Neuromorphic machine learning for audio processing : from bio-inspiration to biomedical applications

Acharya, Jyotibdha

2020

Acharya, J. (2020). Neuromorphic machine learning for audio processing : from bio-inspiration to biomedical applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/142608>

<https://doi.org/10.32657/10356/142608>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 17:51:15 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**NEUROMORPHIC MACHINE LEARNING FOR AUDIO
PROCESSING: FROM BIO-INSPIRATION TO BIOMEDICAL
APPLICATIONS**

JYOTIBDHA ACHARYA

**Interdisciplinary Graduate Programme
NTU Institute for Health Technologies**

2020

**NEUROMORPHIC MACHINE LEARNING
FOR AUDIO PROCESSING: FROM
BIO-INSPIRATION TO BIOMEDICAL
APPLICATIONS**

JYOTIBDHA ACHARYA

**Interdisciplinary Graduate Programme
NTU Institute for Health Technologies**

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

June 25, 2020

.....

Date

Jyotibdha Acharya

.....

JYOTIBDHA ACHARYA

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

June 25, 2020

.....

Date

A handwritten signature in black ink, reading "Arindam Basu", is written over a horizontal line.

.....

Assoc. Prof. ARINDAM BASU

Authorship Attribution Statement

This thesis contains material from 9 publications published or under review in the following peer-reviewed journals / conferences in which I am listed as an author.

Chapter 2, section 2.7.1 is published as Jyotibdha Acharya, Vandana Padala and Arindam Basu, “Spiking Neural Network Based Region Proposal Networks for Neuromorphic Vision Sensors,” *2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5.*

The contributions of the authors are as follows:

- I contributed in designing the simulation, writing the codes, interpreting the results and drafting and revising the manuscript.
- Vandana Padala collected and pre-processed the data and helped design performance metrics.
- Arindam Basu has contributed in overall conception and design of the experiments and drafting and revising the manuscript.

Chapter 2, section 2.7.2 is published as Jyotibdha Acharya, Andres Ussa, Vandana Reddy Padala, Rishiraj Singh Sidhu, Garrick Orchard, Bharath Ramesh, Arindam Basu, “EBBIOT: A Low-complexity Tracking Algorithm for Surveillance in IoVT Using Stationary Neuromorphic Vision Sensors,” *32nd IEEE International System-On-Chip Conference, Singapore, 2019.*

The contributions of the authors are as follows:

- I contributed in writing the codes, comparing the algorithms, interpreting the results and drafting and revising the manuscript.
- Andres Ussa performed the hardware experiments data collection.
- Vandana Padala and Rishiraj Singh has helped develop the object tracking code and performance metrics.
- Garrick Orchard, Bharath Ramesh and Arindam Basu have contributed in overall conception and design of the experiments and drafting and revising the manuscript.

The rest of chapter 2 is published as Jyotibdha Acharya, Aakash Patil , Xiaoya Li, Yi Chen, Shih-Chii Liu and Arindam Basu, “A Comparison of Low-complexity Real-Time Feature Extraction for Neuromorphic Speech Recognition,” *Frontiers in neuroscience* 12 (2018):160.

The contributions of the authors are as follows:

- I contributed in data analysis, software simulations, drafting and revising the manuscript.
- Aakash Patil has contributed in experiment design and data collection using ELM IC.
- Xiaoya Li has contributed in experiment design and data acquisition using silicon cochlea.

-
- Shih-Chii Liu has contributed in overall conception and design of the experiments and revising the manuscript.
  - Arindam Basu has also contributed in conception and design of the experiments and drafting and revising the manuscript.

Chapter 3 is submitted for publication as Rohit Abraham John, Jyotib-dha Acharya, Chao Zhu, Sumon Kumar Bose, Apoorva Chaturvedi, Abhijith Surendran, Keke K. Zhang, Xu Manzhong, Wei Lin Leong, Zheng Liu, Arindam Basu, and Nripan Mathews, “Optogenetics-Inspired Light-Driven Computational Circuits Enable In-Memory Computing for Deep Recurrent Neural Networks,” *Nature Communications* (under review).

The contributions of the authors are as follows:

- Rohit Abraham John contributed in performing all optoelectronic characterizations of the device and drafting and revising the manuscript.
- I contributed in designing the software experiments, performing the simulations and drafting and revising the manuscript.
- Chao Zhu, Apoorva Chaturvedi, Keke K. Zhang and Xu Manzhong fabricated the device under the supervision of Wei Lin Leong and Zheng Liu.
- Sumon Kumar Bose has contributed in designing LIF neuron circuits.

-
- Arindam Basu and Nripan Mathews have contributed in overall conception and design of the experiments, interpreting the results and revising the manuscript.

Chapter 4, section 4.3 and section 4.4 is published as Jyotibdha Acharya, Arindam Basu, and Wee Ser, “Feature extraction techniques for low-power ambulatory wheeze detection wearables,” *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE. IEEE, 2017.*

The contributions of the authors are as follows:

- I contributed in writing the codes, designing the experiments, interpreting the results and drafting and revising the manuscript.
- Wee Ser provided the data and helped analyze the results.
- Arindam Basu has contributed in overall conception and design of the experiments and revising the manuscript.

Chapter 4, section 4.5 and section 4.6 is submitted for publication as Jyotibdha Acharya and Arindam Basu, “Deep Neural Network for Respiratory Sound Classification Enabled by Transfer Learning,” *IEEE Transactions on Biomedical Circuits and Systems* (under review).

The contributions of the authors are as follows:

-
- I contributed in designing the experiments, writing the codes, interpreting the results and drafting and revising the manuscript.
  - Arindam Basu has contributed in overall conception and design of the experiments and revising the manuscript.

Chapter 1 and the literature review of chapters 3 and 5 contain contents from three publications:

- Arindam Basu, Jyotibdha Acharya, Tanay Karnik, Huichu Liu, Member, Hai Li, Jae-sun Seo and Chang Song, “Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8.1 (2018): 6-27.
- Jyotibdha Acharya, and Arindam Basu, “Neuromorphic spiking neural network algorithms,” *Springer Handbook of Neuroengineering*, edited by Nitish V. Thakor (under review).
- Sumon Kumar Bose, Jyotibdha Acharya, and Arindam Basu, “Is my Neural Network Neuromorphic? Taxonomy, Recent Trends and Future Directions in Neuromorphic Engineering,” *Proceedings of the 2019 Asilomar Conference on Signals, Systems, and Computers*.

The contributions of the authors are as follows:

-
- I contributed in overall reviewing relevant literature, comparing algorithms and drafting and revising the manuscript.
  - Arindam Basu has contributed in conception of the publications and drafting and revising the manuscript.
  - Sumon Kumar Bose has reviewed neuromorphic hardware implementations and proposed several metrics for comparison.
  - Tanay Karnik, Huichu Liu, Member, Hai Li, Jae-sun Seo and Chang Song reviewed literature for specific fields of neuromorphic systems.

June 25, 2020

Jyotibdha Acharya

.....

.....

Date

JYOTIBDHA ACHARYA

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor Dr. Arindam Basu. This work would not have been possible without his vision, guidance and valuable suggestions. I could not have sailed through these past four years of my PhD journey without his endless patience with me and constant encouragements during moments of despair. Aside from his genius and immense knowledge, his enthusiasm and passion towards research is something that has been a real inspiration for me and I aspire to carry that with me in my future career. I could not have hoped for a better supervisor.

I would also like to thank members of my thesis advisory committee, Dr. Ser Wee, Dr. Cuntai Guan and Dr. Albert Lim Yick Hou. I sincerely appreciate their insightful comments, support and guidance during every step of my research. I am grateful to Dr. Ser Wee for allowing me to attend his monthly team meetings since it helped me greatly in gaining perspective on biomedical research over the years. I am indebted to Dr. Albert Lim for his invaluable insights from an experienced medical practitioner's point of view which helped me formulate my research problems.

I would also like to thank my friend and ex-colleague Dr. Subhrajit Roy for his support and encouragement. From the beginning of my PhD journey, he has been a guiding star for me both academically and non-academically. I would also like to thank my colleagues Akash Patil and Vandana Reddy Padala for their valuable insights and contributions in this project.

My sincere gratitude goes to Suriani Rabu, admin of Healthtech, NTU for being so helpful and assisting me through all the administrative hurdles over the years.

Last but not the least, I would like to thank my parents, Pradip Kumar Acharya and Pranita Acharya and my girlfriend, Banishree Ghosh for their

---

love and unfailing mental support throughout this project.

Table of Contents

Table of Contents

Abstract	i
List of Figures	iv
List of Tables	xii
List of Acronyms	xiv
1 Introduction	1
1.1 Motivation and Objectives	4
1.1.1 Speech Recognition Using Neuromorphic Auditory Sensors	4
1.1.2 Post-CMOS Hardware for Ultra Energy Efficient Neu- romorphic Computing	5
1.1.3 Respiratory Anomaly Detection for Wearable Devices	8
1.1.4 Spiking Neural Networks for Biomedical Applications	9
1.2 Contributions	10
1.2.1 Ultra Low Power Speech Recognition Using Neuro- morphic Sensors	11
1.2.2 Optogenetics-Inspired Light-Driven Neuromorphic Computing Platform	12
1.2.3 Audio Based Ambulatory Respiratory Anomaly De- tection	13
1.2.4 Spiking Neural Networks for Heart Sound Anomaly Detection	15
1.3 Outline of the Thesis	16

TABLE OF CONTENTS

---

2 Ultra Low Power Speech Recognition Using Neuromorphic Sensors	17
2.1 Introduction	17
2.2 Literature Review	20
2.3 Materials and Methods	22
2.3.1 Silicon Cochlea and Recordings	22
2.3.2 Preprocessing Methods	24
2.3.3 Classification Methods	30
2.4 Hardware Complexity	33
2.4.1 Feature Extraction	34
2.4.2 Classification	35
2.5 Results and Discussions	36
2.5.1 Software Simulations	36
2.5.2 Hardware Measurements	44
2.5.3 Memory and Energy Requirement	45
2.5.4 Hardware vs Software ELM	48
2.5.5 Feature Representations in Combined Binning	49
2.5.6 Comparison with other methods	52
2.5.7 Real-time detection of Word Occurrence	54
2.6 DVS Based Object Tracking	55
2.6.1 SNNRPN	57
2.6.2 EBBIOT	59
2.7 Conclusion	61
3 Optogenetics-Inspired Light-Driven Neuromorphic Computing Platform	62
3.1 Introduction	62
3.2 Literature Review	63
3.3 Materials and Methods	66
3.3.1 Photo-sensitive Synapses	66

TABLE OF CONTENTS

3.3.2	Optical Write and Electrical Erase	67
3.3.3	Linear Dynamic Range	68
3.3.4	Two-shot Write Scheme	69
3.4	Results and Discussions	73
3.4.1	Device Measurements	73
3.4.2	Neural Network Simulation	78
3.5	Conclusion	87
4	Audio Based Ambulatory Respiratory Anomaly Detection	89
4.1	Introduction	89
4.2	Literature Review	92
4.3	Custom Strategy: Materials and Methods	95
4.3.1	Software Method	98
4.3.2	Hardware Method	102
4.4	Custom Strategy: Results and Discussions	103
4.5	General Strategy: Materials and Methods	107
4.5.1	Dataset	107
4.5.2	Evaluation Metrics	107
4.5.3	Related Work	108
4.5.4	Proposed Method	110
4.6	General Strategy: Results and Discussions	116
4.6.1	Generalized Model	116
4.6.2	Patient Specific Transfer Learning Model	117
4.6.3	Memory and Computational Complexity	121
4.7	Conclusion	124
5	Spiking Neural Networks for Heart Sound Anomaly Detection	126
5.1	Introduction	126
5.2	Literature Review	129

TABLE OF CONTENTS

5.3	Materials and Methods	133
5.3.1	Dataset	133
5.3.2	Preprocessing	133
5.3.3	Feature Extraction	134
5.3.4	Proposed CNN Model and Conversion to SNN	136
5.4	Results and Discussions	138
5.4.1	Weight Normalization	139
5.4.2	Simulation Duration	141
5.4.3	Computational Complexity	144
5.5	Conclusion	146
6	Conclusion	148
6.1	Ultra Low Power Speech Recognition Using Neuromorphic Sensors	149
6.2	Optogenetics-Inspired Light-Driven Neuromorphic Comput- ing Platform	151
6.3	Audio Based Ambulatory Respiratory Anomaly Detection .	153
6.4	Spiking Neural Networks for Heart Sound Anomaly Detection	154
6.5	Future Work	156
A	Appendix: SNNRPN	158
A.1	Introduction	158
A.2	Materials and Methods	159
A.2.1	Data Collection	159
A.2.2	Proposed Architecture	160
A.2.3	Computational Complexity & Memory	162
A.2.4	Evaluation Metrics	165
A.3	Results and Discussions	166
A.4	Conclusion	170

TABLE OF CONTENTS

B Appendix: EBBIOT	171
B.1 Introduction	171
B.2 Materials and Methods	172
B.2.1 Event based Binary Image and Noise Filtering	173
B.2.2 Region Proposal	176
B.2.3 Overlap Based Tracker	179
B.3 Results and Discussions	182
B.3.1 Datasets	182
B.3.2 Tracker Evaluation Metrics	182
B.3.3 Performance and Resource Requirements	183
B.4 Conclusion	184
Publications	186
Bibliography	188

Abstract

The recent success of Deep Neural Networks (DNN) has renewed interest in machine learning and, in particular, bio-inspired machine learning algorithms. DNN refers to neural networks with multiple layers (typically two or more) where the neurons are interconnected using tunable weights. Although these architectures are not new, availability of massive amount of data, huge computing power and new training techniques have led to its great success in recent times. DNN has been applied to a variety of fields such as image classification, face recognition in images, word recognition in speech, natural language processing, game playing etc. and the success stories of DNN continue to increase every day. With the progress in software, there has been a concomitant push to develop better hardware architectures to support the deployment as well as training of these algorithms.

While these methods are loosely inspired by the brain, in terms of actual implementation, the similarity between mammalian brain and these algorithms is merely superficial. More often than not, these algorithms require huge energy for real world tasks due to their computation and memory heavy nature, which limits their potential application in energy constrained scenarios such as IoT or wearables. Internet of Things (IoT) is a rapidly growing phenomenon where millions of connected sensors are distributed to improve a variety of applications ranging from precision agriculture to smart factories. In recent years, there has also been a large shift in biomedical industry towards reliable wearable devices for monitoring of health conditions and early detection of diseases. To make IoT systems scalable to millions of nodes/sensors, one has to overcome limits of data rate and energy dissipation. A possible solution is edge computing where part of the processing is done at the sensor (at the edge of the network) instead of shifting

all processing to the cloud. The common challenge for wide scale adaptation of edge computing in Internet of Things (IoT) and wearable applications is the constraints posed by the limited energy and memory available in these devices. Neuromorphic engineering is a possible solution to this problem where different approaches such as analog or physical based processing, non von-Neumann architectures, low-precision digital datapath and event or spike based processing are used to overcome energy and memory bottlenecks. Therefore, it is no surprise that Neuromorphic engineering was recently voted as one of the top ten emerging technologies by the World Economic Forum and the market for neuromorphic hardware is expected to grow to $\sim \$1.8B$ by 2023/2025. However, cross-layer innovations on neuromorphic algorithms, architectures, circuits, and devices are required to enable adaptive intelligence especially on embedded systems with severe power and area constraints.

Since the success story of deep learning began with massive improvements in computer vision tasks offered by deep neural networks, the same trend is repeated in neuromorphic engineering. Spiking neural networks are already reaching performance close to their traditional DL counterparts and several post-CMOS neuromorphic platforms have been shown to perform basic computer vision tasks such as digit recognition. The primary focus of this thesis is a less explored cognitive task from neuromorphic perspective, audio processing. To this end, neuromorphic audio systems have been explored from a diverse set of perspectives, neuromorphic audio sensors, novel neuromorphic nano-devices as well as potential biomedical application areas for such systems.

In the first work, low power feature extraction and data preprocessing techniques customized towards neuromorphic audio sensors were explored. The developments of neuromorphic spiking cochlea sensors and population encoding based ELM hardware were brought together to design a real-time,

power and memory efficient neuromorphic speech recognition system. The proposed hybrid feature extraction strategies were also extended to neuromorphic vision sensor based object tracking. In the second work, a unique neuromorphic computing platform comprising of photo-excitabile neuristors (PENs) is proposed that expands the potential of memristive-based implementations to advance beyond simple pattern matching to complex cognitive tasks such as speech recognition. Combining optical writing and electrical erasing schemes, a new method is developed to transfer offline learnt weights of a deep recurrent neural network on to the memristive device resulting in highly energy efficient speech classification due to electrical mode inference and in-memory computing. In the third work, different feature extraction and classification strategies are implemented for audio based respiratory anomaly detection and then low precision representations of the proposed networks are explored for memory efficient wearable implementation. In the final work, the viability of spiking neural networks and bio-realistic features are examined for wider biomedical applications. By employing spiking neural networks for audio based detection of cardiac anomalies, it is demonstrated that spiking neural networks can achieve similar level of performance as their ANN counterpart at a fractional computational cost.

Overall, the primary goal of this work is to devise brain-inspired strategies and algorithms to enable power and memory efficient audio processing that can be beneficial to a wide area of resource constrained applications ranging from speech processing to audio based biomedical monitoring.

List of Figures

2.1	Block diagram of the proposed speech recognition system. The shaded block for feature extraction is implemented in software in this work while the other two blocks are implemented in hardware	22
2.2	(A) Circuit architecture of one ear of the Dynamic Audio Sensor. The input goes through a cascaded set of 64 bandpass filters. The output of each of the filters is rectified. This rectified signal then drives an integrate-and-fire neuron model. (B),(C) Two sample spikes of digit "2". Dots correspond to spike outputs from the 64 channels of one ear of the cochlea.	23
2.3	Fixed number of bins: Both the short (A) and long (B) samples have the same number of bins but the bin width (W) is shorter for short samples and longer for long samples.	27
2.4	Fixed Bin Size: Both the short (A) and long (B) samples have the same bin width (W). A short sample produces smaller number of bins and a long sample produces larger number of bins	29
2.5	(A) ELM network architecture: The weights w_{ij} in the first layer are random and fixed while only the second layer weights need to be trained. (B) Architecture of the neuromorphic ELM IC	30
2.6	Fixed number of bins: Accuracy vs. number of hidden nodes for different number of bins. (A),(B): Time based binning (1B):10 bins per sample shows highest overall accuracy. (C), (D): Spike count based binning (2B): 10 bins per sample shows highest overall accuracy	39

2.7	Fixed bin size: accuracy vs. number of hidden nodes for different bin sizes. (A),(B): Time based binning (1A): 40 ms bin size shows highest overall accuracy. (C), (D): Spike count based binning (2A): 400 spikes/bin shows highest overall accuracy	41
2.8	(A) Combined Binning architecture for fixed bin size case by fusing the decisions of two ELMs operating in time based and spike count based modes respectively. (B),(C) Comparison of binning modes, fixed bin size: Accuracy vs. Number of Hidden Nodes using different binning modes for fixed bin size, Combined Mode shows highest overall accuracy, comparable to fixed number of bins	43
2.9	Hardware classification accuracies for different binning strategies, Combined Binning strategy shows highest classification accuracy	44
2.10	Histogram of correlation coefficients of input weights	49
2.11	Confusion matrices for different binning strategies exhibit peaks at different locations for time based and spike count based binning. Hence, a combination of these two methods can eliminate some of these errors.	51
2.12	Correlation between confusion matrices	52
2.13	Visualization of RPN input and output: input frame shows a scene with one car and two humans (a) and the corresponding output frame shows the region proposals in red (b). The denoising in the output frame is done by the refractory layer while the region proposal is done by convolution layer and clustering layer.	58

2.14	Flowchart depicting all the important blocks in the system: binary frame generation, region proposal and overlap based tracking.	60
2.15	A sample EBBI with corresponding X and Y histogram based region proposals.	60
3.1	Highly linear weight update of PENs allow us to transfer high precision weights from offline learnt Deep NNs	69
3.2	Two shot write scheme for transferring learned weights to PEN crossbar. After an initial optical potentiation of the entire array, one measurement is done to estimate the con- ductance G_n^{op} for the n-th device. Next, one electrical write (w1) operation is done for duration T_p followed by a measure- ment m2 to estimate change in conductance or slope. Finally, the second write pulse (w2) is applied with the duration T_{wn} calculated based on the earlier estimated slope.	72
3.3	Fitting a straight line to the measured conductance show very little deviation from linearity (smaller than half of step size).	74
3.4	(A) The slope estimated from the first 16 states (correspond- ing to $T_p = 16T_s$) results in error in predicting the conduc- tances compared to the best fit line which measures all the states. Device data from Figure 3.3 is replotted showing the last few states where the difference between the best-fit line and the estimated line based on the two shot write scheme are shown. (B) Deviation of the conductances for all the states in Figure 3.3 are replotted showing the slope estimation method results in a systematic error component that increases mono- tonically reducing the effective number of states within linear range.	75

- 3.5 Violin plot of Slope estimation variance (r_p) as a function of write noise. Noise level σ corresponds to $\Delta G_{avg}/\sigma = 8$. The length of each violin corresponds to the spread of r_p for a given write noise while the width corresponds to the frequency of occurrence. Higher write noise results in higher variance of r_p . 77
- 3.6 Accuracy vs linear dynamic range for CNN with device weights when tested on the MNIST handwritten digit recognition task. 79
- 3.7 A deep neural network with 12 trainable layers including convolutional, fully connected and recurrent LSTM layers is used to classify 12 different spoken digits. The detailed architecture of the network is shown with all filter sizes and dimensions mentioned. 81
- 3.8 The accuracy obtained in electrical inference without slope estimation errors is plotted for various number of device states and write noise with constant-LDR lines shown. σ corresponds to measured write noise for one device with $\Delta G_{avg}^2/\sigma^2 \approx 64$. The network can reach close to ideal floating-point accuracy ($\sim 96\%$) with $LDR > 64$ while measured device LDR exceeds 6000 (marked by stars with colour denoting wavelength used). The iso-accuracy contours align with the lines of constant LDR showing the importance of this metric. 82

3.9	Network accuracy is plotted by fixing number of states to 980 while varying write noise as Figure 3.8 for various slope estimation errors in the two shot write scheme. There is a drop of $\sim 0.15 - 2.46\%$ in accuracy for the measured PEN characteristics (bold circles with colour denoting used wavelength). Original accuracies without slope estimation errors are also plotted using stars for reference. The effective number of states (Y-axis) reduces from the actual number of states due to slope-estimation errors.	83
3.10	Network accuracy is plotted by fixing number of states to 980 while including write noise induced slope estimation errors r_p . While the low noise regime is similar to the earlier case showing floating-point accuracy for $LDR > 64$, the network is unable achieve this accuracy for noise exceeding $\sim 4\sigma$. In the intermediate noise region, the required number of states shoots up rapidly to > 2000	84
3.11	Accuracy is plotted against LDR with different layers quantized. LSTM layer shows highest sensitivity towards LDR. .	85
3.12	Reducing conductance slope estimation error by increasing measurement time results in same accuracy as floating-point implementation when the least square fit line using all 980 states is used. Using only 16 points to estimate the slope, results in $\sim 2\%$ drop in accuracy while using 8 points result in $\sim 30\%$ drop in accuracy.	86
4.1	Spectrogram of Breathing Sounds of Wheeze Patient and Normal Subject: frequency contours corresponding to wheeze are marked	96

4.2	Spectrogram of Breathing Sounds of Wheeze Patient and Normal Subject After applying FCT : Spectral lines corresponding to wheeze are amplified while noise is suppressed	97
4.3	Feature Vectors Computed Before and After Applying Feature Extraction Algorithm: Sharp peaks at frequencies corresponding to spectral contours are visible after FCT is applied	99
4.4	Accuracy vs. No. of Frequency Channels: At least 50 Channels are required for reasonable accuracy	101
4.5	L_{opt} vs. No. of Frequency Channels: L_{opt} is almost linearly dependent on No. of Channels	101
4.6	Accuracy vs. SNR: Frame and Sample accuracies are 65% and 70% respectively for SNR=-10dB and increase to 89% and 99% respectively at SNR=7dB.	104
4.7	Hybrid CNN-RNN: a three stage deep learning model. Stage 1 is a CNN that extracts abstract feature maps from input Mel-spectrograms, stage 2 consists of a Bi-LSTM layer that learns temporal features and stage 3 consists of fully connected (FC) and softmax layers that convert outputs to class predictions.	111
4.8	Boxplot of normalized breathing cycle duration: Intra-patient normalized breathing cycle duration is computed by normalizing each breathing cycles duration by average breathing cycle duration of the corresponding patient while for Inter-patient variability, the normalization is done by average breathing cycle duration of the entire dataset.	113

4.9	Screen and transfer learning model: First the patients are screened into healthy and unhealthy based on % of breathing cycles predicted as unhealthy. For patients predicted to be unhealthy, trained model is re-trained on patient specific data to produce patient specific model which then performs the four class prediction on breathing cycles.	118
4.10	Local log quantization: Score achieved by VGG-16, MobileNet and hybrid CNN-RNN with varying bit precision under local log quantization. VGG-16 requires minimum bit precision to achieve full precision (fp) accuracy while MobileNet requires maximum bit precision.	121
4.11	Resource comparison: Comparison of normalized computational complexity (GFLOPS/sample) and minimum memory required (Mbits) by VGG-16, MobileNet and hybrid CNN-RNN. MobileNet and hybrid CNN-RNN present a trade-off between computational complexity and memory required for optimum performance.	122
5.1	Lyon Cochlear Model: The model uses a series of cascaded notch filters along with resonators to model the basilar membrane in human ear. The half wave rectifiers (HWR) detect the energy of the signals and multiple stages of automatic gain control (AGC) is used to output neuron firing rates. . .	136
5.2	Heart sound classification: Preprocessing, feature extraction and architecture of CNN classifier for heart sound classification. Initially, PCG recordings are preprocessed and segmented (described in section 5.3.2), then spectrogram and cochleagram features are extracted (described in section 5.3.3) and finally normal and abnormal heart sounds are classified by a CNN classifier (described in section 5.3.4)	137

5.3	Effect of different weight normalization methods: both Max Norm and Percentile Norm shows significant performance improvement over no normalization and overall 99.9 Percentile Norm achieves accuracies closest to the original ANN. Spectrogram performs better than cochleagram without normalization while cochleagram performs slightly better than spectrogram after normalization.	140
5.4	Effect of simulation duration: Error rates of SNNs approach the original ANN as simulation duration is increased and it reaches a plateau at around 100 ms	142
5.5	Effect of simulation time: classification accuracies are measured at each timestep for both spectrogram and cochleagram. Total simulation duration is set at $T=100$ ms. The network reaches close to optimum accuracy at only about 50% of the simulation time	143
5.6	classification error vs computational complexity: Computational complexity of the SNN is shown as a fraction of computational complexity of the equivalent ANN. Firstly, we see that there is a clear trade-off between classification error and computations required. Secondly, the SNN can achieve accuracies within 3 – 4% of the ANN using $5\times$ less computations.	145
A.1	Visualization of RPN input and output: input frame shows a scene with one car and two humans (a) and the corresponding output frame shows the region proposals in red (b). The denoising in the output frame is done by the refractory layer while the region proposal is done by convolution layer and clustering layer.	163
A.2	Precision-recall curve for six recordings: sensor-object distance shows significant impact on the recall value.	166

A.3	IoU curve: smaller window size results in more accurate region proposals as evident from higher precision and recall for higher IoU values.	167
A.4	Lateral excitation: precision and recall curve for 100 m (day) measured using IoU and fitness score (FS). Lateral excitation shows better precision at higher overlap ratios for FS measurement. For overlap ratio 0.8, lateral excitation improves precision by 2% without loss of recall (marked by arrow). . .	168
A.5	Comparison with event based mean shift algorithm: precision-recall curve for 100 m (day) measured using IoU and fitness score. SNN-RPN outperforms mean shift for IoU based measurements while mean shift obtains slightly higher precision for fitness score based measurement at significantly smaller recall.	169
B.1	Flowchart depicting all the important blocks in the system: binary frame generation, region proposal and overlap based tracking.	173
B.2	Timing diagram showing interrupt driven operation of the NVS for duty cycled low power operation.	174
B.3	A sample EBBI with corresponding X and Y histogram based region proposals.	177
B.4	Comparison of EBMS, KF and EBBIOT in terms of precision and recall for different IoU thresholds. EBBIOT outperforms others and shows more stable precision and recall values for varying thresholds.	184
B.5	Comparison of EBMS and EBBI+KF with EBBIOT in terms of total computations per frame and memory requirement showing significantly less resource requirement for EBBIOT.	184

List of Tables

2.1	Preprocessing methods	24
2.2	Memory and energy requirements for fixed number of bins method (1B,2B). Highest accuracy case is marked red	46
2.3	Memory and energy requirements for fixed bin size method (1A,2A). Highest accuracy cases are marked red	47
3.1	Mesured value of non-ideality parameters and device accuracy for two devices and three wavelengths of light.	84
4.1	Accuracy Comparison (%)	103
4.2	Power Requirement Comparison Method I:Total Transmis- sion, Method II: Processing + Transmit Feature vector, Method III:Processing+ transmit label, Method IV:Processing + transmit frames with wheeze.	106
4.3	Summary of existing literature on ICBHI dataset	110
4.4	Comparison of results	116
4.5	Comparison of patient specific Models	120
A.1	Details of the Dataset	160
B.1	Details of the Dataset	182

List of Acronyms

ANN Artificial Neural Network

ASR Automatic Speech Recognition

Bi-LSTM Bidirectional Long Short Term Memory

CMA Current Mirror Array

CNN Convolutional Neural Network

COPD Chronic Obstructive Pulmonary Disease

DAS Dynamic Audio Sensor

DFT Discrete Fourier Transform

DL Deep Learning

DVS Dynamic Vision Sensor

DNN Deep Neural Network

DRNN Deep Recurrent Neural Networks

EBWD Entropy-Based Wheeze Detection

ELM Extreme Learning Machine

ENOS Effective Number of States

FCT Frequency Contour Tracking

FFT Fast Fourier Transform

FGMOS Floating Gate Metal-Oxide-Semiconductor

GMM Gaussian Mixture Model

HMM Hidden Markov Model

IoT Internet of Things

IoVT Internet of Video Things

LDR Linear Dynamic Range

LIF Leaky Integrate and Fire

MFCC Mel-Frequency Cepstral Coefficients

NCS Neuromorphic Computing Systems

NEF Neural Engineering Framework

NVM Non-Volatile Memory

PCG Phonocardiogram

PEN Photo-Excitable Neuristor

ReLU Rectified Linear Unit

RF Random Forest

RNN Recurrent Neural Network

RRAM Resistive Random-Access-Memory

SGD Stochastic Gradient Descent

SNN Spiking Neural Network

STFT Short Time Fourier Transform

SVM Support Vector Machine

Chapter 1

Introduction

Artificial neural networks (ANN) trained by deep learning has shown tremendous success in audio, visual and decision making tasks. While these methods are loosely inspired by the brain, in terms of actual implementation, the similarity between mammalian brain and these algorithms is merely superficial. Moreover, more often than not, these algorithms require huge energy for real world tasks due to their computation and memory heavy nature, which limits their potential application in energy constrained scenarios. "Neuromorphic Engineering"—a term coined in 1990 by Carver Mead in his seminal paper [1], is a possible solution to this energy efficiency problem. In this paper, he claimed that hardware implementations of algorithms like pattern recognition would be more energy and area efficient if it adopts biological strategies of analog processing.

While the above idea of brain-inspired analog processing is very appealing and showed initial promise with several interesting sensory prototypes, it failed to gain increased traction over time possibly due to the potential difficulties of creating robust, programmable, large-scale analog designs that can benefit from technology scaling in an easy manner.

However, the necessity of power efficient bio-inspired computing paradigms such as neuromorphic computing is predicted to become more and more prominent in upcoming years with continuous developments in human centric computing which integrates IoT, edge computing and wearable devices to enable seamless information processing at the nodes for improved human experience [2]. Therefore, in the last 5 years, there has been

renewed interest in this topic, albeit with a slightly expanded connotation of the term "neuromorphic". The renaissance in Neuromorphic research over the past years has seen the term being used in a wider sense than the original definition [1]. This is partially due to the fact that scientists from different communities (not only circuits or neuroscientists) ranging from material science to computer architects have now become involved. Based on the recent work, we can describe the key principles of this new version of neuromorphic systems as:

- (P1) Use of **analog or physics based processing** as opposed to conventional digital circuits—this is same as the original version of neuromorphic system from a circuits perspective.
- (P2) From the viewpoint of computer architecture, usage of **non von-Neumann architecture** (independent of analog or digital compute) and **low-precision digital datapath** are hallmarks of neuromorphic systems. In other words, conventional computers using von-Neumann architectures read from memory, compute and write back the result—this is very different from brain-inspired systems where memory and computing are interspersed [3].
- (P3) Computer scientists and algorithm developers on the other hand consider a system neuromorphic if it uses a **spiking neural network** (SNN) as opposed to a traditional artificial neural network (ANN). Neurons in an SNN inherently encode time and output a 1-bit digital pulse called a spike or action potential.

These three principles of neuromorphic engineering are adapted from *Bose et al.* [4] where the authors rigorously survey all the papers published from 2017-19 that use the term neuromorphic to describe their work. The papers surveyed contains all the papers from Nature series, Science, Science Advances as well as conferences ISSCC, SOVC, JSSC etc. The details of

the papers surveyed can be found in [5]. While these principles broadly define which innovations can be categorized as neuromorphic, from an implementation standpoint, neuromorphic ecosystem consists of neuromorphic sensors, neuromorphic devices, neuromorphic circuits and neuromorphic algorithms. Different innovations in neuromorphic sensors, algorithms, devices and circuits can follow one of these principles or a combination of multiple principles. We will describe the works in this thesis in the context of these principles in later sections. In the following sections, first we will describe the motivations and objectives of this work and then elaborate on the novel contributions of each work.

Most recent work on neuromorphic ML have been related to computer vision [6,7]. However, since neuromorphic algorithms try to imitate temporal event based information processing capabilities of brain, they are better equipped to process temporally varying signals. Hence, in this work we propose to bridge this gap and focus on using neuromorphic approaches for processing audio signals such as speech. Further, we also show example applications of audio processing in the biomedical domain that need extreme low power operation. In particular, we develop neuromorphic systems for ambulatory monitoring of audio markers of pulmonary and cardiac diseases. To summarize, the four main contributions of this thesis are:

1. Develop novel feature extraction techniques for neuromorphic auditory sensors so that low power analog classifiers can be connected to these sensors.
2. Develop post-CMOS hardware for ultra energy efficient neuromorphic in-memory implementations with high weight precision needed to implement Deep Recurrent Neural Networks for speech classification.
3. Design different power and memory efficient feature extraction and classification strategies for ambulatory respiratory anomaly detection.

4. Develop SNN based neuromorphic platform for audio based cardiac abnormality detection

More details about each of these topics are described next.

1.1 Motivation and Objectives

In this section, we will discuss the motivations and the objectives of each part of our work in details.

1.1.1 Speech Recognition Using Neuromorphic Auditory Sensors

To achieve the incredible performance and power efficiency of sensors in human biology, neuromorphic visual [8,9], auditory [10] and tactile [11] sensors have been developed which operate in an asynchronous, spike based manner similar to human retina, cochlea and mechanoreceptors, respectively. This asynchronous sampling results in most informative part of the signal being readout first leading to quick reaction times [12–14]. Also, some problems that are difficult to solve with traditional sampling methods often have elegant solutions in this asynchronous spike domain [15]. Though traditional computer vision and speech processing algorithms can be applied to the data collected through event based sensors, since these sensors are fundamentally different from their traditional counterpart, those methods are not always effective in taking full advantage of the unique properties and power efficiency of neuromorphic sensors unless novel feature extraction methods are developed.

In the past decade, a class of shallow neural networks with random weights in the first layer has become popular due to their fast training speed, good generalization abilities and need for less training data. Termed extreme

learning machine (ELM) [16] in the machine learning community, it has relations to earlier machine learning methods [17] as well as methods proposed in computational neuroscience [18, 19]. Compared to the reservoir computing methods [19], the major difference of ELM is the lack of feedback or recurrent connections. Due to the majority of weights in the network being random, it is very amenable to neuromorphic analog implementations [20–23]. Combination of the aforementioned neuromorphic sensors with such analog implementations of ELM classifier can result in a very low power end-to-end system that can accomplish complex tasks as a fraction of power required for the traditional systems if proper feature extraction is used to bridge the sensor with the classifier.

Therefore, **the first objective** of this work is to find low power feature extraction and data preprocessing techniques customized towards neuromorphic audio sensors and evaluate their performance using neuromorphic ELM IC. We also hope to extend these techniques to dynamic vision sensors for object tracking due to the similarity and temporal nature of the signals in both these domains.

1.1.2 Post-CMOS Hardware for Ultra Energy Efficient Neuromorphic Computing

A number of novel devices have been proposed over the years to perform the functions of neurons, synapses or adaptive elements, in general in neuromorphic systems. Without losing generality, the relationship between the activity patterns of the input neurons $\mathbf{x}$ and the output neurons $\mathbf{y}$ in a neural network can be expressed as:

$$\mathbf{y}_n = \mathbf{W}_{n \times m} \mathbf{x}_m \quad (1.1)$$

Most of the novel devices have been used to implement the synaptic function that is denoted by W in the equation 1.1. Some desirable properties of adaptive synapses are: 1) Non-volatile weight 2) Compact size and 3) Low energy.

Historically, one of the earliest non-volatile storage elements proposed as a learning synapse was the floating-gate MOS (FGMOS) [24, 25]. Due to its compatibility with CMOS, FGMOS devices have been integrated into various adaptive neuromorphic circuits in the past [26]. Due to the ability of using a single transistor as a learning synapse in neuromorphic systems and ability to integrate tightly with CMOS circuits [27], FGMOS is a good potential candidate for building large scale adaptive neuromorphic systems.

Emerging nonvolatile memory (NVM) denotes a series of new types of memory technologies that does not rely on electrical charge to store the data (e.g., as SRAM and DRAM). Some representative embedded NVM (eNVM) technologies are phase change memory (PCM) [28], spin-transfer-torque random-access-memory (STT-RAM) [29], resistive random-access-memory (RRAM) [30], ferroelectric field-effect-transistor (FeFET) memory [31] etc. Many of these eNVM technologies can be utilized to implement neuromorphic computing systems (NCS) where the programmable resistance of the eNVM cells represents the synaptic weights of the DNNs, especially RRAM (a.k.a. memristor). The resistance state (often referred to as memristance) of a memristor can be tuned by applying an electrical excitation. Similarity between the programmable resistance state of memristors and the variable synaptic strengths of biological synapses dramatically simplifies the design of NCS. One attractive property for memristors over FGMOS devices is their potentially lower write energy.

Recent advancement in spintronic research has shown path towards ultra-low voltage, low current and energy-efficient computing beyond traditional CMOS. These devices exploit the new materials, device designs as well as

novel switching phenomena such as spin-torque-transfer (STT), domain-wall (DW) movement, spin-hall effect (SHE), spin-orbital torque (SOT), and magnetoelectric (ME) switching [32–34]. These innovations not only augment energy-delay scaling roadmap in CMOS systems with disruptive solutions in energy-efficient power management, dense on-die non-volatile memories, non-volatile logic, etc., but also expand the computing paradigm with non-von Neumann architectures (e.g. neuromorphic hardware). Besides the memristor-crossbar-based synapse, spintronic devices can also be used to implement synaptic operation. Comparing with CMOS-based approaches, spintronic devices offer opportunities to realize compact neuron and synapse models with a potential energy-efficiency improvement (e.g. lower supply voltage ($<1V$) and lower current compared to memristor and floating gate [35–37]). The low current requirements have a direct effect on noise and hence special stochastic algorithms might be needed to train these networks [38].

Since most of the works are still in early research phase and simulation stage, main challenges remain in large-scale implementation including variation, reliability and demonstrating real world applications of systems designed using these devices. While we have seen small neural networks designed using such novel devices performing small benchmark tasks [39, 40], for novel neuromorphic devices to become a viable low power alternate to traditional deep learning in complex tasks, we need to investigate the effect of scaling and different non-idealities of these devices in the context of designing truly complex deep architectures that can perform real world tasks at a competitive level to that of traditional computing paradigms. Therefore, **the second objective** of this work is to use novel neuromorphic devices to design truly deep speech recognition networks with similar level of complexity and performance as traditional deep learning and to find strategies to exploit or mitigate unique device properties while doing so.

1.1.3 Respiratory Anomaly Detection for Wearable Devices

Following the incredible success of deep learning in computer vision problems in recent years [41, 42], significant focus has been attributed to solving similar problems in neuromorphic paradigm. The progress in this field can be exemplified by development in neuromorphic vision sensors [43] [44]), new event based algorithms [45] and application of these algorithms in a variety of applications ranging from motion estimation [46] and stereo-vision [7] to motor control [47] and gesture recognition [48]. A variety of SNN architectures have also been developed and produced admirable accuracies on traditionally used image datasets such as MNIST [49], CIFAR-10 [50], Imagenet [51] etc.

While vision systems have received significant attention, another success area of DL, audio processing [52], has not received its fair share of attention as of yet. Audio processing provides its unique set of challenges for the neuromorphic systems due to the spatial and temporal nature of the signal. To build effective neuromorphic audio systems, we require careful handling of the signal in both time and frequency domain as well as challenges pertaining to signal sampling, noise robustness etc.

Due to the higher energy efficiency and lower area overhead of neuromorphic systems, one interesting area of application for neuromorphic audio processing can be in biomedical domain for wearable devices. Due to massive success of machine learning and deep learning, it has been used extensively in a variety of applications including clinical diagnostics and biomedical engineering [53]. While image, ECG or EEG based biomedical applications have shown significant promise, audio based biomedical applications are increasingly coming into focus of research community due to the simplicity of designing audio recording devices and non-invasive nature of audio based diagnostics. Some prominent areas of audio based diagnostics include sleep

apnea detection [54], cough sound identification [55], heart sound classification [56] etc. Therefore, examination of neuromorphic audio systems in context of biomedical applications is worth exploring. Hence, **the third objective** of this work is to examine strategies and algorithms for audio based biomedical applications for wearable devices, specifically in audio based respiratory anomaly detection and exploring how neuromorphic solutions can be used to reduce the memory and energy footprint of the proposed systems.

1.1.4 Spiking Neural Networks for Biomedical Applications

Artificial neural networks (ANN) traditionally used in the machine learning community use neurons that are continuous function approximators that communicate with each other using connections with precise weights while biological neurons use asynchronous and sparse spikes for computation and communication [57]. In order to bridge this gap between artificial and biological neurons, researchers have proposed spiking neural networks (SNN) that closely resemble their biological counterparts.

With advances in computational neuroscience, neuromorphic engineering [45] and deep learning, there is a significant rise in interest in spiking neural networks over the past decade. The future promise of spiking neural networks can be attributed to their low power consumption, event-driven processing, inherent ability to incorporate temporal information and large scale parallelism [57] [58].

The primary difference between SNN and traditional ANN is that the former uses more bio-realistic spiking neurons that communicate in the network using binary signals or spikes through connections of synapses. Spiking neurons were originally studied to model the biological neurons in mammalian brain to understand their information processing and pattern recognition capability [6]. Though different neuron models in SNNs differ greatly, typ-

ically the input to the neurons are a series of spikes that get converted to analog synaptic currents. These synaptic currents are added and integrated in time to generate membrane potential. When the membrane potential reaches a certain threshold, it generates an output spike and the generated spike induces further change in the next neuron.

The learning algorithms of SNNs can be broadly classified into two classes: spike learning and conversion learning. For spike learning, the SNNs are directly trained while for conversion learning, an equivalent ANN is first trained using traditional learning algorithms and then converted to equivalent SNNs. A careful examination of recent literature on SNNs reveal an increasing bias toward ANN to SNN conversion methods compared to spike learning methods. This can be attributed to the fact that it is relatively easy to take advantage of availability of extensive resources, tried and tested algorithms and well developed frameworks of traditional deep learning to design state-of-the-art SNNs using conversion based methods.

Since the research on deep SNNs is still at an early stage, majority of recent works ([59], [60], [61], [62]) report results on benchmarks used on traditional deep learning research such as MNIST [49], CIFAR-10 [50], TIDIGITS [63], Imagenet [51] etc. But we are yet to see application of SNNs on wider real world cognitive tasks and applications. As discussed previously, biomedical domain can be a lucrative area of exploration for neuromorphic algorithms and hardware in general. Therefore, **the fourth objective** of this work is to determine the viability of SNN based neuromorphic systems for audio based biomedical applications.

1.2 Contributions

The major contribution of this work spans different areas of neuromorphic audio processing from innovations in feature extraction techniques for neuro-

morphic sensors to utilizing novel devices in designing large scale neuromorphic computing systems to developing power efficient algorithms for audio based wearable biomedical systems. The contributions of different parts of this work are described below.

1.2.1 Ultra Low Power Speech Recognition Using Neuromorphic Sensors

This work combines the **first and third principles** of neuromorphic systems described above. Here we employ neuromorphic audio sensor which uses analog filtering circuits and produces time encoded spike outputs. For speech classification we use neuromorphic ELM IC that utilizes device mismatch in current mirror array to generate random weights. The main contributions of this work are as follows:

- (1) In this work, we developed a real-time, low-complexity neuromorphic speech recognition system using a spiking silicon cochlea, a feature extraction module and a population encoding method based Neural Engineering Framework (NEF)/Extreme Learning Machine (ELM) classifier IC.
- (2) Several feature extraction methods with varying memory and computational complexity are presented along with their corresponding classification accuracies. We introduce two different binning modes (time and spike based) and two different binning strategies (fixed bin size and fixed number of bins) and explore these feature extraction techniques in terms of accuracy, computational and memory overhead.
- (3) The proposed fixed number of bins and fixed bin size methods presented a clear trade-off between classification accuracy and hardware overhead where using fixed number of bins gives $\approx 2 - 33\%$ higher

accuracy with $\approx 50\times$ more hardware overhead compared to the fixed bin size method.

- (4) We also show that a fixed bin size based feature extraction method that votes across both time and spike count features can achieve an accuracy of 95% in software similar to previously report methods that use fixed number of bins per sample while using $\approx 3\times$ less energy and $\approx 25\times$ less memory for feature extraction ($\approx 1.5\times$ less overall).
- (5) The proposed speech classification algorithms were not only tested on software, but also on a neuromorphic ELM IC described in [64] by feeding the chip with feature vectors produced by the methods described above.
- (6) Finally, we also show how similar asynchronous event driven algorithms and strategies can be extended to computer vision domain to design power efficient object tracking based on dynamic vision sensors.

1.2.2 Optogenetics-Inspired Light-Driven Neuromorphic Computing Platform

In this work, we combine the **first and second principles** of neuromorphic systems. Here we design an analog computing platform utilizing the physical properties of novel devices and employ them in non-von Neumann architecture (in-memory computing) for highly energy efficient speech recognition. The main contributions of this work are as follows:

- (1) In this work, we propose a unique neuromorphic computing platform comprising of photo-excitabile neuristors (PENs) to selectively control the excitability of artificial neurons and update the weights of synapses with high linearity to obtain precise weights necessary for deep recurrent neural networks(DRNN).

- (2) Combining optical writing and electrical erasing schemes, we propose a new method of optical writing to transfer offline learnt weights on to the device, after which the device is used for inference in electrical mode only without requiring optical inputs resulting in high energy efficiency due to in-memory computing.
- (3) In comparison to state-of-the-art, the proposed PEN features an order of magnitude higher Linear Dynamic Range (LDR) facilitating an order of magnitude lower iterations for weight programming, and enabling us to simulate a DRNN for speech recognition with an order of magnitude higher parameters than digit recognition networks.
- (4) We also perform in depth analysis of the effect of different non-idealities that can arise from real world implementation of such a neuromorphic system, such as write noise, non-linearity, measurement errors etc. on the performance of the system.
- (5) Our work extends the frontiers of current neuromorphic devices by enabling unprecedented accuracy and scale of parameters required for online, adaptive and truly intelligent systems for applications in speech recognition and natural language processing.

1.2.3 Audio Based Ambulatory Respiratory Anomaly Detection

In this work, we utilize the **second principle** of neuromorphic systems. Here we first design different feature extraction and classification strategies for ambulatory respiratory anomaly detection and then find low precision representations of our proposed networks for memory efficient wearable implementation. The main contributions of our work are as follows:

- (1) In this work, we present a novel low-complexity wheeze detection

method based on custom T-F feature extraction technique (frequency contour tracking) for automatic wheeze detection. Two hardware friendly variants of the algorithm have also been proposed.

- (2) Applying the proposed feature extraction algorithm on a respiratory sound dataset for binary classification, we achieved very high classification accuracy ($>99\%$) at considerably low computational complexity ($3 \times -6 \times$) compared to earlier methods and the power consumption of the proposed method is shown to be significantly less ($70 \times -100 \times$) compared to ‘record and transmit’ strategy in wearable devices.
- (3) Next, we have developed a hybrid CNN-RNN model to perform four class classification (normal, wheeze, crackle, both) on a very large respiratory sound database, ICBHI’17 respiratory audio dataset and the proposed architecture produced state of the art results.
- (4) We also propose a patient screening and transfer learning strategy to identify unhealthy patients and then build patient specific models through transfer learning. This proposed strategy provides significant improvements over previous results.
- (5) We also develop a local log quantization strategy for reducing the memory cost of the models that achieves $\approx 4 \times$ reduction in minimum memory required without loss of performance.
- (6) Finally, we compare the performance, memory and computational complexity with several popular deep architectures and show that our proposed model achieves highest classification performance at minimum memory cost.

1.2.4 Spiking Neural Networks for Heart Sound Anomaly Detection

In this work, we propose an SNN based neuromorphic platform for audio based cardiac abnormality detection. The main contributions of this work are as follows:

- (1) In this work, we explore broader applications of SNNs by designing an SNN based classifier for detecting heart sound anomaly on 2016 PhysioNet/CinC Challenge dataset for classification of normal/abnormal heart sound recordings [65].
- (2) First we segment the phonocardiograms using a logistic regression based hidden markov model and design a CNN that produces $\approx 88\%$ accuracy on the classification of individual hear beats into normal and abnormal category.
- (3) Next, we convert the proposed CNN architecture to equivalent SNN and evaluate its performance. We show that the SNN can achieve accuracies very close to their ANN equivalent.
- (4) We explore different weight normalization techniques and show the effect of Max Norm and Percentile Norm on the performance of SNN.
- (5) We explore the latency-accuracy trade-off for the SNN and show that the SNN approaches accuracies close to equivalent ANN as the simulation duration is increased.
- (6) Finally, we calculate the computational complexity of the SNN and show the computational complexity-accuracy tradeoff for the SNN. We also show that the SNN can achieve performance very close to ANN at $\approx 5\times$ less computations.

1.3 Outline of the Thesis

The remainder of this thesis is structured as follows. In Chapter 2, we proposed a low power bio-inspired real-time sound recognition system based on dynamic audio sensors and we also briefly describe how the algorithmic developments in this work can be extended to object tracking based on dynamic vision sensors. In Chapter 3, we propose a unique neuromorphic computing platform comprising of photo-excitabile neuristors (PENs) to implement a memristive DRNN with unprecedented complexity—twelve trainable layers with more than a million parameters to recognize spoken commands with $>90\%$ accuracy. Chapter 4 describes feature extraction techniques and network architectures for power and memory efficient audio based respiratory diagnostics. In Chapter 5, we explore a potential application of SNN in biomedical domain by designing an SNN based classifier to detect heart sound anomalies. Chapter 6 describes the conclusions obtained from the research performed in this work and suggestions for the future work. Appendix A and B elaborate on dynamic vision sensor (DVS) based object tracking techniques mentioned in chapter 2. There is no dedicated chapter for literature review since each chapter focuses on different aspects of neuromorphic audio systems and deserves its own literature review. Therefore, general concepts regarding neuromorphic computation are introduced in this chapter and the literature relevant to each chapter is discussed in the respective chapters.

Chapter 2

Ultra Low Power Speech Recognition Using Neuromorphic Sensors

2.1 Introduction

Event based neuromorphic sensors have received significant attention from research community in recent years. Two of the most popular sensors in this space are neuromorphic retina and neuromorphic cochlea. Neuromorphic retina or more commonly known as asynchronous dynamic vision sensors are bio-inspired visual sensors that produce spikes corresponding to each pixel in their visual fields (also termed address-event representation or AER) where there is a change in light intensity [66]. Similar approaches have been proposed in auditory domain in developing silicon models of cochleas that operate in an event-driven asynchronous fashion [67]. These event-based asynchronous cochlea sensors implement a bio-mimetic filtering circuit that produces spikes at the output in response to input sounds. The primary advantage of these sensors over traditional audio and video sensors stem from their high power efficiency resulting from asynchronous spike based nature. Though traditional computer vision and speech processing algorithms can be applied to the data collected through event based sensors, most of these algorithms are not efficient in taking advantage of lower power and memory footprint of neuromorphic sensors [68]. With the rapid growth in Internet

of Things (IoT), Internet of Video Things (IoVT), edge computing in recent years, there is a significant demand in power and memory efficient signal processing techniques and algorithms that can sufficiently take advantage of these low power sensors.

Speech recognition has been one of the most studied problems in computer science over the past decades . Since humans evolved to have speech as the primary mode of communication, if we envision a future where we will interact with hundreds of low-power IoT devices on a daily basis to improve our quality of life, using speech as the medium of human-machine interaction is an obvious solution. To achieve this future, finding power efficient solutions for Automatic Speech Recognition (ASR) systems is of prime importance. Neuromorphic audio sensor and compatible low power processing techniques provide an interesting approach to handle this problem.

This chapter is primarily focused on the development of a low power bio-inspired real-time sound recognition system based on Dynamic Audio Sensor(DAS) but we also briefly describe Dynamic Vision Sensor(DVS) based solution for power efficient object tracking. While DAS based speech recognition and DVS based object tracking may seem two completely different problems at the surface, there are surprising similarities between the two from an algorithmic perspective. Both of these problems require efficient handling of temporally varying asynchronous spiking signals, both of them require real time processing and both of them require devising effective binning/framing strategies to minimize memory requirement while achieving high accuracy.

In this work, we bring together the developments of neuromorphic spiking cochlea sensors and population encoding based ELM hardware to lay the groundwork for a low power bio-inspired real-time sound recognition system. Several different low-complexity feature extraction methods that do not require storage of entire spike trains are explored in this chapter and tradeoffs

between memory / computation requirements and recognition accuracy are presented. Measured accuracy results using the silicon cochlea in ([67]) and ELM chip in ([64]) are presented for the TIDIGITS dataset with 11 spoken digit classes. Though the entire processing of the signal does not use spike times, our method still uses “physical” computation in the cochlea and NEF/ELM blocks which is the essence of neuromorphic engineering as described in [69].

For DVS based object tracking, we briefly discuss event based object tracking using Spiking Neural Network (SNN) and then discuss a hybrid frame and event based approach called EBBIOT. In our work related to event based audio processing, we explore the trade-offs between two feature extraction strategies, a fully frame based approach where the entire audio is stored first before feature extraction and a hybrid approach where time or spike count based binning is used for more memory efficient and computationally less expensive feature extraction. For our DVS based object tracking, we employ a similar strategy. In our first work, we use multiple SNN layers to process the input spikes in asynchronous event driven manner and only in the final clustering layer we employ frame based processing. For our second work, we use a accumulate events for a fixed duration and generate an event based binary image for further processing. This method is analogous to time based binning for DAS based audio feature extraction.

The remainder of this chapter is organized as follows: Section 2.2 discusses the relevant literature. Section 2.3 details the hardware and the proposed methods. Section 2.4 computes the hardware complexity for the proposed methods. Section 2.5 reports the results for both software simulation and hardware measurements and presents a discussion on the obtained results. Finally, section 2.6 briefly introduces DVS based object tracking and section 2.7 outlines key takeaways of this work.

2.2 Literature Review

Considerable progress has been made recently in machine learning for speech recognition tasks with the developments in traditional Gaussian Mixture Models and Hidden Markov Models to the more recent deep neural networks [52]. However, these models require very complicated processing of the input speech and are not suited for simple sensor nodes with limited power; nor do they perform well in the presence of large background noise (cocktail party problem). In contrast, the human auditory system is able to perform sound stream segregation easily.

This has led to an interest in studying the biological auditory system and developing silicon models of cochleas that operate in an event-driven asynchronous fashion [67] much like the neurons in the auditory pathway. These event-based asynchronous cochlea sensors implement a bio-mimetic filtering circuit that produces spikes at the output in response to input sounds ([67], [70], [71]). The AEREAR2 sensor has been used previously for typical speech recognition problems such as speaker identification ([72], [73]) and digit recognition ([74], [75]). The inter-spike intervals and channel specific spike counts are used as features for these tasks. High classification accuracy (95%) was reported using these features for a speaker independent digit recognition task using a software implementation of support vector machine (SVM) based implementation ([74]). However, this method required the storage of the entire spike response of the cochlea channels to one spoken digit so that the spikes can be pre-processed prior to classification resulting in huge memory requirements.

In parallel, there has been considerable progress in developing neural models of cognition and a particularly popular one based on population coding is the Neural Engineering Framework (NEF) [76], [77]. It proposes a framework for neural simulations where the input is non-linearly encoded using random projections and linearly decoded to model the required func-

tion. The typical NEF architecture consists of three layers, the input layer, a hidden layer consisting of a large number of non-linear neurons and an output layer consisting of linear neurons. In the encoding phase, the inputs are multiplied with random weights and passed to the non-linear neurons. The non-linear function can be any neural model from the spiking Leaky-Integrate-and-Fire model to more complex biological models ([78]). With the use of recurrent connections, NEF can also be used for modeling even dynamic functions. NEF has been proved to be an efficient tool for implementing large scale brain models like SPAUN ([79]) and therefore, is being widely used in neuromorphic research community.

A similar model has been separately developed in the machine learning community. Termed as the Extreme Learning Machine (ELM) ([80]), it also uses a three layered architecture with random projection of the input and linear decoding. It is essentially a feedforward network and does not have feedback connections allowed in NEF—hence, it may be considered as a sub-category of NEF architectures. It has been used in a variety of applications ranging from neural decoding ([81]) and epileptic seizure detection ([82]) to speech recognition ([83]) and big data applications ([84]) in the past. Since we also use a feedforward network in this work, we will refer to our algorithm as ELM in the rest of the chapter acknowledging that it can be referred to as NEF as well. Low power hardware implementations of this algorithm have also been reported recently ([64]). In this work, the authors developed a neuromorphic analog ELM IC where input signals are converted to analog currents and current mirror arrays are used for multiplication. The random weights for ELM are generated by the physical mismatch of transistors in the current mirror array.

2.3 Materials and Methods

The basic architecture of our proposed speech recognition system is shown in Fig. 2.1. The speech input is acquired by the Dynamic Audio Sensor and the spikes produced are then passed to the feature extraction block. The extracted features are then sent to an Extreme Learning Machine for classification. For the experiments in this chapter, we have simulated the feature extraction block in software only, but the feature extraction techniques described here can easily be implemented in hardware using standard micro-controllers. Measured results from hardware are presented for the cochlea and the ELM chip.

Figure 2.1: Block diagram of the proposed speech recognition system. The shaded block for feature extraction is implemented in software in this work while the other two blocks are implemented in hardware

2.3.1 Silicon Cochlea and Recordings

The N-TIDIGITS18 dataset [75] used in this work, consists of recorded spike responses of a binaural 64-channel silicon cochlea ([70]) in response to audio waveforms from the original TIDIGITS dataset [85]. The silicon cochlea and later generations of this design, model the basilar membrane, inner hair cells and spiral ganglion cells of the biological cochlea. The basilar membrane is implemented by a cascaded set of 64 second-order band-pass filters, each with its own characteristic frequency. The output of each filter goes to an inner hair cell block which performs a half-wave rectification of its input. The output of the inner hair cell goes to a ganglion cell block implemented by a spiking neuron circuit. The spike output is transmitted

off-chip using the asynchronous address-event representation (AER). The binaural chip is connected to microphones emulating left and right ears. The circuit architecture of one ear is shown in Fig. 2.2a. Circuit details are described in ([70], [67]).

Figure 2.2: (A) Circuit architecture of one ear of the Dynamic Audio Sensor (adopted from [67]). The input goes through a cascaded set of 64 bandpass filters. The output of each of the filters is rectified. This rectified signal then drives an integrate-and-fire neuron model. (B),(C) Two sample spikes of digit "2". Dots correspond to spike outputs from the 64 channels of one ear of the cochlea.

In the recordings, impulses are added at the beginning and end of the audio digit files so that the start and end points of the spike recordings are visible. The impulses lead to spike responses from all channels.

Table 2.1: Preprocessing methods

Binning \ Mode	Time	Spike Count
Fixed Bin Size	1A	2A
Fixed No. of Bins	1B	2B

2.3.2 Preprocessing Methods

To obtain the feature vectors from the spike recordings of the silicon cochlea, we used the spike count per window or bin for two modes of binning with two binning strategies which resulted in four preprocessing techniques as shown in Table 2.1. In the methods described, we used bins of width W and used counters to count the number of spikes across different channels within that bin. The output of the i^{th} bin can be represented as $X_W(i)$ where X_W is a $[1 \times C]$ vector containing spike counts across C channels. Next, we cascaded the bin outputs to produce the feature vectors. The 4 modes differ in the choice of W and the number of vectors to be cascaded.

2.3.2.1 Binning Modes

We used two modes for binning the cochlea images to extract features. The first one is time based binning (1A, 1B) where the whole spike sample is divided into several bins based on time or duration of the sample (T_{sample}). The second one is spike count based binning (2A, 2B) where we binned the spike trains based on the total number of spikes in the sample (N_{sample}). While the time based strategy captures the spike density variation in cochlear images quite well, it completely ignores the temporal variation (longer vs. shorter samples). On the other hand, the spike count based strategy captures the temporal variation but ignores the spike density variation (dense vs. sparse samples).

2.3.2.2 Binning Strategies

For all modes, we used two binning methods, (A) fixed bin size and (B) fixed number of bins. These methods are described below for the time based binning mode only to avoid repetition. A similar philosophy applies to the case of spike count based binning.

Fixed Number of Bins In this method, the total number of bins per sample is fixed or static. As a result, in the time mode of binning, the longer samples produce longer bins than shorter samples (as shown in Fig. 2.3). If the number of bins per sample is fixed at B_{sta} , and the corresponding bin width is w_{sta} for a sample, the total duration of the sample, T_{sample} is given by:

$$T_{sample} = w_{sta} \times B_{sta} \quad (2.1)$$

In this method, we explicitly set the value of B_{sta} and w_{sta} is determined by:

$$w_{sta} = T_{sample}/B_{sta}. \quad (2.2)$$

If the total number of spikes per sample is denoted as N_{sample} and the average number of spikes/bin/channel is denoted by $\bar{n}_{spikes}$, we can write:

$$N_{sample} = B_{sta} \times C \times \bar{n}_{spikes} \quad (2.3)$$

The output of each bin ($Xw(i)$) is cascaded to produce the feature vector $F=[Xw(1) Xw(2) \dots Xw(B)]$. So the dimension of the feature vector is $C \times B_{sta}$. Thus, there is a clear trade-off between the feature vector size and temporal resolution of the bins. Higher temporal resolution leads to a larger feature vector size and therefore higher classification complexity and vice-versa. The primary disadvantage of this method is that it requires a priori information about the duration of total spike count of the sample before the

binning. So, the entire sample needs to be stored first and afterwards binning is done on the sample. Thus, the memory requirement of this method is quite high and the latency is equal to the sample duration. Finally, use of a dynamic bin size removes inter-sample variability of temporal resolution by performing an intrinsic normalization. The longer samples are compressed as a result of longer bin sizes while shorter samples expanded as a result of shorter bin sizes. This is the feature extraction method used in previous work such as [74]. Fixed number of bins is a commonly preferred technique in machine learning community since it ensures fixed feature vector sizes without loss of information and thus facilitates lesser complexity in machine learning model designing and better performance. While this is an intuitive strategy from a signal analysis perspective, from a neuromorphic point of view, it is not biologically plausible due to significantly long latency required for this strategy to work.

In the spike count mode, the total number of spikes N_{sample} summed across all channels and time is divided into a fixed number of bins (B_{sta}) leading to a limit (N_{sample}/B_{sta}) on total number of spikes per bin. Whenever this limit is reached, it defines the formation of a bin. Spike counts in all channels are frozen to create a feature vector and this process repeats.

Fixed Bin Size In the fixed bin size method, the size of bins is predetermined in terms of time duration or spike count based on the mode of binning. As a result, the longer samples produce larger number of bins while shorter samples produce smaller number of bins (as shown in Fig. 2.4).

Denoting the number of bins per sample using this strategy as B_{fix} , setting the bin width to w_{fix} and using the same notations as the previous method, we can write:

$$T_{sample} = w_{fix} \times B_{fix} \quad (2.4)$$

Figure 2.3: Fixed number of bins: Both the short (A) and long (B) samples have the same number of bins but the bin width (W) is shorter for short samples and longer for long samples.

In this method, we explicitly set the value of w_{fix} and the corresponding value of B_{fix} is determined by:

$$B_{fix} = T_{sample}/w_{fix} \quad (2.5)$$

The total number of spikes per sample is given by:

$$N_{sample} = B_{fix} \times C \times \bar{n}_{spikes} \quad (2.6)$$

As the number of bins produced by the samples (B_{fix}) is different for different samples and the ELM classification algorithm requires a fixed feature vector size, we needed to find an optimum number of bins that produce overall high accuracy irrespective of sample duration. Larger number of bins results in increased feature vector size which in turn makes the classification task more difficult and computationally expensive while smaller

number of bins result in feature vectors that sample the spike recordings coarsely and thus, miss the finer variations over the sample durations. Our initial experiments suggested that, for number of bins 8-12 the classification accuracy is optimum. Therefore, we decided to fix the number of bins to 10. So, the dimension of the feature vector is $10 \times C$. Based on the bin size and total sample duration, one of two cases can occur:

Case I: $B_{fix} \geq 10$

If the sample produced more than 10 bins, we will keep the output of only first 10 bins to produce the feature vectors while ignoring the rest. These bins are then cascaded to produce the feature vector $F=[Xw(1)Xw(2)...Xw(10)]$. In this case,

$$T_{sample} \geq w_{fix} \times 10 \quad (2.7)$$

For this case, we only use a fraction of total spikes to produce the feature vector. If the number of spikes used is given by N_{used} , we can write:

$$N_{used} = 10 \times C \times \bar{n}_{spikes} \leq B_{fix} \times C \times \bar{n}_{spikes} = N_{sample} \quad (2.8)$$

Case II: $B_{fix} < 10$

For the samples that produce less than 10 bins for a given bin size, zero padding is used to produce the feature vectors. In this case,

$$T_{sample} < w_{fix} \times 10$$

For this case, we use all the spikes in the sample to produce the feature vector. So,

$$N_{used} = B_{fix} \times C \times \bar{n}_{spikes} = N_{sample} \quad (2.9)$$

So, generalizing the two cases, we can express N_{used} as:

$$N_{used} = \min\{10 \times C \times \bar{n}_{spikes}, B_{fix} \times C \times \bar{n}_{spikes}\} \quad (2.10)$$

There is no need to store the sample in memory for this method since the

feature vectors are directly produced from the samples with predetermined bin sizes. Thus, memory required for this method is quite low. As we require only 10 bin outputs to form a feature vector, the latency is independent of the sample duration unlike the previous strategy. The primary drawback of this strategy is that to obtain fixed feature vector sizes, we have to use a fixed number of bins (10 in our case) to produce the feature vectors and therefore, for larger samples, the rest of the bin outputs are discarded. So, there is a loss of information in this strategy. Moreover, as the bin size is fixed, this method does not provide any input duration normalization like the earlier strategy. A similar fixed spike count based frame size strategy has been used by [86] for feature extraction.

Figure 2.4: Fixed Bin Size: Both the short (A) and long (B) samples have the same bin width (W). A short sample produces smaller number of bins and a long sample produces larger number of bins

2.3.3 Classification Methods

2.3.3.1 Extreme Learning Machine: Algorithm

Figure 2.5: (A) ELM network architecture: The weights w_{ij} in the first layer are random and fixed while only the second layer weights need to be trained. (B) Architecture of the neuromorphic ELM IC (adopted from [87])

The ELM is a three layer feedforward neural network introduced in [80] shown in Fig. 2.5a. The output of the ELM network with L hidden neurons

is given by:

$$\mathbf{o} = \sum_i^L \beta_i \mathbf{H}_i = \sum_i^L \beta_i \mathbf{g}(\mathbf{w}_i^T \mathbf{x} + \mathbf{b}_i) \quad (2.11)$$

where x is a d -dimensional input vector, b_i is the bias of individual neurons, w_i and β_i are input and output weights respectively. $g(\cdot)$ is the non-linear activation function (sigmoid function is commonly used) and h_i is the output of the i^{th} hidden neuron. While the weights w_i and b_i are chosen from any random distribution and need not be tuned, the output weights β_i need to be tuned during training. So the basic task in this architecture is to find the least square solution of β given targets of training data:

$$\text{Minimize}_{\beta}: \|\mathbf{H}\beta - \mathbf{T}\|^2, \quad (2.12)$$

where $\mathbf{T}$ is the target of training data. The optimal solution of β is given by

$$\tilde{\beta} = \mathbf{H}^\dagger \mathbf{T}, \quad (2.13)$$

where $H^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{H}$ [88]. The simplest method to find $H^\dagger$ is using orthogonal projection:

$$\begin{aligned} H^\dagger &= (H^T H)^{-1} H^T \text{ if } H^T H \text{ is non-singular} \\ H^\dagger &= H^T (H H^T)^{-1} \text{ if } H H^T \text{ is non-singular.} \end{aligned} \quad (2.14)$$

Moreover, taking advantage of the concepts from ridge regression ([89]), a constant is added to the diagonal of $H^T H$ or $H H^T$ which results in a solution that is more stable and has better generalization performance. C is a tunable hyperparameter. Several regularization techniques have been explored for determining the optimal value of H to reduce training time and number of hidden neurons [90]. The simple architecture of the ELM network makes it a suitable candidate for hardware implementation.

2.3.3.2 Extreme Learning Machine: Hardware

For the classification task, we have used software ELM as well as hardware measurements on the neuromorphic ELM chip described in [64].

The digital implementations of ELM can benefit from the software simulations of the ELM shown in this chapter. The architecture of the ELM chip is shown in Fig. 2.5b. The 128 input digital values are converted to analog currents using current mode DACs which are multiplied by random weights in a 128×128 current mirror array (CMA). The random weights are generated by the physical mismatch of transistors in the CMA. The 128 output currents are converted to spikes using an array of 128 integrate and fire neurons. The corresponding firing rates are obtained by an array of digital counters while the second stage of ELM is performed in digital on a FPGA. While the software ELM uses random weights with a uniform random distribution, the chip generates random weights w_{ij} with lognormal distribution. This is due to the exponential relation of current and threshold voltage (V_T) in the sub-threshold regime which leads to mismatch induced weights of the form

$$w = e^{\Delta V_T / U_T} \quad (2.15)$$

where ΔV_T denotes mismatch between threshold voltages of a pair transistors forming a current mirror. However, lognormal distributions have positive mean and software simulations show that zero mean weights result in higher classification accuracy. Hence, a simple digital post-processing is used on the outputs to obtain zero mean random numbers. Instead of directly feeding the chip output h_i to the second stage, the difference h'_i of neighboring neurons were used. So, the modified output of the hidden layer is given by:

$$h'_i = h_i - h_{(i+1) \bmod(128)}, \quad i = 1, 2, \dots, 128 \quad (2.16)$$

As shown in [87], any weight distribution w_{ij} can become a zero mean distri-

bution w'_{ij} using this technique. We will refer to this as log difference weight for the rest of this chapter. Finally, instead of using typical non-linearities like sigmoid or tanh as $g(\cdot)$, we have used an absolute value (abs) function as the preferred nonlinearity. While software simulations show similar or slightly better classification accuracy for an absolute value non-linearity compared to typical non-linearities, it has several other advantages over them. Absolute value is a non-saturating non-linearity and so feature vectors need not be normalized before being passed to the ELM unlike saturating non-linearities. This reduces the computational burden. Moreover, the hardware implementation of abs non-linearity is much simpler than sigmoid or similar non-linearities.

2.4 Hardware Complexity

In this section we will discuss the hardware complexity comprising computations and memory requirements for the classifier and the two feature extraction methods described earlier. For our calculation, we assume that the time stamp of a spike is encoded using 32 bits and the channel address of the spike is 6 bits. The average number of spikes per sample is assumed to be N_{sample} and the spike counter size is $b_{counter}$ bits. The number of computations (N_{comp}) can be written as the sum of two components:

$$N_{comp} = N_{feature} + N_{ELM} \quad (2.17)$$

where $N_{feature}$ is the number of computations for feature extraction while N_{ELM} is the number of computations required for classification by ELM.

The total memory required (M_{total}) can be written as sum of two components:

$$M_{total} = M_{feature} + M_{ELM} \quad (2.18)$$

where $M_{feature}$ is the memory required for feature extraction while M_{ELM} is the memory required for classification by ELM.

2.4.1 Feature Extraction

2.4.1.1 Fixed Number of Bins

For the fixed number of bins method, the entire sample needs to be stored first and bin sizes are to be determined later. So, the memory required to store the spike information of an entire sample (time stamp and channel count) is

$$M_{samples} = 38 \times N_{sample} \text{ bits} \quad (2.19)$$

Now, if the number of bins is B_{sta} , a total of $B_{sta} \times C$ counters are required to count the spikes and produce the feature vector. Therefore, the memory required to store a feature vector is given by:

$$M_{feature\_vector} = B_{sta} \times C \times b_{count} \text{ bits} \quad (2.20)$$

So, from Eqs. 2.19 and 2.20 the total memory requirement for fixed number of bins method is

$$M_{feature} = 38 \times N_{sample} + B_{sta} \times C \times b_{count} \text{ bits} = 38 \times B_{sta} \times C \times \bar{n}_{spikes} + B_{sta} \times C \times b_{count} \text{ bits} \quad (2.21)$$

In terms of computations, there will be a counter increment for each spike resulting in N_{sample} operations per sample. Also, for each spike, the time stamp needs to be compared with the bin boundary to determine when to reset counters. Hence the total number of operations per sample is given by:

$$N_{feature} = N_{sample} + N_{sample} = 2N_{sample} \quad (2.22)$$

2.4.1.2 Fixed Bin Size

For the fixed bin size method, the feature vectors are produced directly from the sample as the bin sizes are pre-determined. Thus, there is no need for storing the sample in memory. The only memory required in fixed bin size method is for storing the feature vectors. Since we cascade 10 bin outputs to produce a feature vector in this method, using calculations similar to above, we get:

$$M_{feature} = M_{feature\_vector} = 10 \times C \times b_{count} \text{ bits} \quad (2.23)$$

Finally, the total number of operations per sample is the total number of counter increments which is equal to the number of spikes used to produce the feature vector. So,

$$N_{feature} = N_{used} = \min\{10 \times C \times \bar{n}_{spikes}, B_{fix} \times C \times \bar{n}_{spikes}\}, \quad (2.24)$$

For the fixed bin size method, the memory requirement is significantly less than the fixed number of bins method as there is no need for storing the entire sample before feature extraction. Furthermore, pre-determined bin sizes enable this method to be compatible with real-time speech recognition systems. The significant advantage of this method over the fixed number of bins method in terms of memory and energy requirements is further quantified in Section 2.5.3.

2.4.2 Classification

N_{ELM} again has two parts due to multiply and accumulate (MAC) in the first and second layers of the network. Hence, N_{ELM} is given by the following:

$$N_{ELM} = D \times L + L \times C_o \quad (2.25)$$

where C_o is the number of output classes, D is the dimension of the feature vector and L is the number of hidden nodes. For our classification problem, number of output classes $C_o = 11$. Moreover, calculating log difference weights requires some additional subtractions ($= L$). Hence, the final value of N_{ELM} is given by:

$$N_{ELM} = D \times L + L \times C_o + L \quad (2.26)$$

Finally, the amount of memory (M_{ELM}) needed by the classifier is given by:

$$M_{ELM} = D \times L \times b_W + L \times C_o \times b_\beta \quad (2.27)$$

where b_W and b_β denote the number of bits to represent the first and second layer weights.

The energy requirement for the ELM in the custom implementation will depend on the energy required for each of these operations. Since multiplications are dominant, E_{MAC} is the prime concern. Since it has been shown that $E_{MAC}^{ana} < E_{MAC}^{dig}$ for the first stage with maximum number of multiplies ([81]), we have used an analog neuromorphic ELM hardware in this work. However, the findings of this work are applicable to a digital implementation of ELM on ASIC or on a microprocessor.

2.5 Results and Discussions

2.5.1 Software Simulations

In this section, we show the classification accuracies for different pre-processing strategies described in Section 2.3.2 using a software ELM with uniform random weights and log difference weights. Though there are 64 (max. channel count) channels available in AEREAR2, only the first 54 channels were active for all the samples, therefore $C=54$. All the results

were obtained by averaging the classification accuracies over five randomized 90% – 10% train-test splits .

2.5.1.1 Fixed Number of Bins (1B,2B)

For the fixed number of bins method, we have used $B_{sta} = 5, 10, 20$ and 30 bins per sample for both time based and spike count based modes with number of hidden nodes in the classifier varying from $L= 500$ to 3000. The results for this experiment are plotted for both uniform random and log difference weights in Fig. 2.6 (A) and (B) for time based and in Fig. 2.6 (C) and (D) for spike based binning respectively. It can be seen that, for both modes, $B_{sta}= 10$ bins per sample produced maximum overall classification accuracy of around 96% for uniform random and 94.2% for log difference weights respectively. Also, the accuracies tend to initially increase with increasing values of L but eventually saturate and start decreasing due to over-fitting.

2.5.1.2 Fixed Bin Size (1A,2A)

For the fixed bin size method (1A, 2A in Table 1), we have used 10ms to 40ms bin sizes for time based binning and 300spikes/bin to 600spikes/bin bin sizes for spike count based binning with number of hidden nodes varying from 500 to 3000. The results for this experiment are plotted for both uniform random and log difference weights in Fig. 2.7 (A) and (B) for time based and in Fig. 2.7 (C) and (D) for spike based binning respectively. It can be seen that, for time based mode, the maximum overall classification accuracy was obtained for 40 ms. We tried a bin size of up to 80 ms and found that the accuracy decreases beyond 40 ms. This is probably due to the fact that, while larger bin sizes ensure less loss of information at the end of a digit, it produces very small number of bins for shorter samples which results in their misclassification. For spike count based mode the maximum overall

(A) Uniform random weights (1B)

(B) Log difference weights (1B)

classification accuracy was obtained for 400 spikes/bin. Interestingly, even with fixed bin size features, we can obtain maximum classification accuracies 95.2% for uniform random and 92.5% for log difference weights using time based binning. Hence, this points to a method for low hardware complexity feature extraction that also allows usage of analog sub-threshold ELM circuits with log difference weights. Second, the trend of increasing accuracies with increasing temporal bin size is due to the ELM being able to access larger parts of the speech sample. Lastly, the difference between spike count based binning and time based binning is very large in this case indicating that spike count alone is not a good distinguishing feature for fixed bin size.

(C) Uniform random weights (2B)

(D) Log difference weights (2B)

Figure 2.6: Fixed number of bins: Accuracy vs. number of hidden nodes for different number of bins.

(A),(B): Time based binning (1B): 10 bins per sample shows highest overall accuracy.

(C), (D): Spike count based binning (2B): 10 bins per sample shows highest overall accuracy

(A) Uniform random Weights (1A)

(B) Log difference weights (1A)

(C) Uniform random weights (2A)

(D) Log difference weights (2A)

Figure 2.7: Fixed bin size: accuracy vs. number of hidden nodes for different bin sizes.

(A),(B): Time based binning (1A): 40 ms bin size shows highest overall accuracy.

(C), (D): Spike count based binning (2A): 400 spikes/bin shows highest overall accuracy

2.5.1.3 Combined Binning

Out of the two binning strategies described in this chapter, the fixed bin size method is more convenient to implement from a hardware perspective. Moreover, the memory and energy requirements of the fixed bin size method are much less than its counterpart as discussed in Section 2.5.3. But as we have shown in Section 2.5.1.2, the best case accuracy of the fixed bin size method is typically 1 – 2% less than that of fixed number of bins method. This is due to two factors: lack of input temporal normalization and loss of information due to discarded bins. To increase the accuracy of the fixed bin size method, we adopted a combined binning approach as shown in Fig. 2.8 (A). In this fixed bin size strategy, the input data is processed in parallel using both time based and spike count based binning. The feature vectors produced are applied to their respective ELMs and the ELM outputs are combined (added) in the decision layer. The final output class is defined as the strongest class based on both strategies. Figure 2.8 (B) and (C) compares the best case accuracies of time based binning (40 ms bin size), spike count based binning (400 spikes/bin bin size) and combined binning mode (combination of both). The combined binning mode not only outperforms both the time and spike count based modes, but also shows accuracies similar to the best case accuracies of fixed number of bins method for both type of weights. The reasons for this increased accuracy is further discussed in Section 2.5.5.

(A) Combined binning architecture

(B) Comparison of binning modes
(Uniform random weights)

(C) Comparison of binning modes
(Log difference weights)

Figure 2.8: (A) Combined Binning architecture for fixed bin size case by fusing the decisions of two ELMs operating in time based and spike count based modes respectively. (B),(C) Comparison of binning modes, fixed bin size: Accuracy vs. Number of Hidden Nodes using different binning modes for fixed bin size, Combined Mode shows highest overall accuracy, comparable to fixed number of bins

2.5.2 Hardware Measurements

Finally, the proposed feature extraction methods were tested on a neuromorphic ELM IC described in [64] by feeding the chip with feature vectors produced by the methods described above. Due to the long testing times needed, we only tested the best accuracy cases of time based binning (40 ms bin size), spike count based binning (400 spikes/bin bin size) and combined binning (combination of the two). The accuracies obtained are shown in Fig. 2.9. The optimum accuracy obtained by time based binning is slightly higher than that of spike count based binning while combined binning approach outperforms both of the methods. However, comparing this result with the earlier software simulations, we notice two differences. First, the accuracies obtained are slightly less than software and second, the accuracy increases with increasing L . Possible reasons for this reduction in accuracy

Figure 2.9: Hardware classification accuracies for different binning strategies, Combined Binning strategy shows highest classification accuracy

and its subsequent increase with increasing L are discussed in section 2.5.4.

2.5.3 Memory and Energy Requirement

In this section, we will determine the memory and energy requirements of different post processing methods described. We have used the formulae derived in Section 2.4 to determine the memory requirement and the computational complexity of different strategies. Moreover, we used the specifications of Apollo2 Ultra-Low Power Microcontroller for calculating pre-processing energy requirement ($10\mu A/MHz$ at $3.3V$ ¹) and specifications of the neuromorphic ELM chip for calculating the classification energy requirement ($0.47pJ/MAC$, [64]). Tables 2.2 and 2.3 show the memory requirement, computational complexity and average energy per sample of fixed number of bins and fixed bin size strategies assuming 1500 hidden nodes for the ELM. If we compare the best accuracy cases of both fixed bin size and fixed number of bins methods, these results show that fixed binning requires $\sim 50\times$ less memory for feature extraction ($\sim 3\times$ overall) and $\sim 30\%$ less energy compared to that of fixed number of bins method. Furthermore, as the combined binning requires approximately twice the memory and computational complexity than that of the simple time or spike count based binning methods, we can conclude that the combined binning strategy is able to produce accuracies similar to fixed number of bins method using $\sim 25\times$ less memory for feature extraction ($\sim 1.5\times$ overall). Moreover, since the neuromorphic ELM chip uses mismatch induced random weights for the first layer of the ELM, no memory is required to store the first layer weights. Only, the second layer trained weights need to be stored in memory. The minimum resolution of the second layer weights (b_β) required for no loss of accuracy is found to be 8 bits.

---

¹<http://ambiqmicro.com/apollo-ultra-low-power-mcu/apollo2-mcu/>

Table 2.2: Memory and energy requirements for fixed number of bins method (1B,2B). Highest accuracy case is marked red

Bins/ Sample	5	10	20	30
Memory Required (Feature Extraction) (Kbits)	213	215	219	223
Memory Required (ELM Layer 2) (Kbits)	132	132	132	132
No.of Ops/sample (Feature Extraction) (Kops)	11	11	11	11
No. of MACs/sample (ELM Layer 1) (KMACs)	405	810	1620	2430
No. of MACs/sample (ELM Layer 2) (KMACs)	18	18	18	18
Energy Required (nJ/sample)	3061	3251	3632	4013

Table 2.3: Memory and energy requirements for fixed bin size method (1A,2A). Highest accuracy cases are marked red

	Time Based Binning				Spike Count Based Binning			
	10 ms	20 ms	30 ms	40 ms	300 spikes /bin	400 spikes /bin	500 spikes /bin	600 spikes /bin
Bin Size								
Memory Required (Feature Extraction) (Kbits)	4.3	4.3	4.3	4.3	4.3	4.3	4.3	4.3
Memory Required (ELM Layer 2) (Kbits)	132	132	132	132	132	132	132	132
No.of Ops/sample (Feature Extraction) (Kops)	0.7	1.4	1.8	2	2	3	4	5
No. of MACs/sample (ELM Layer 1) (KMACs)	810	810	810	810	810	810	810	810
No. of MACs/sample (ELM Layer 2) (KMACs)	18	18	18	18	18	18	18	18
Energy Required (nJ/sample)	2232	2301	2340	2360	2360	2459	2558	2657

2.5.4 Hardware vs Software ELM

One key observation from the results obtained is that the hardware ELM requires larger number of hidden nodes to obtain accuracies similar to the software simulations (compare Fig. 2.8 and Fig. 2.9). While software simulations required around 2000 hidden nodes to obtain optimum accuracy, the hardware required more than 5000 hidden nodes to obtain comparable accuracies. This discrepancy can be ascribed to the higher correlation between input weights in the ELM IC. In an ideal ELM, the input weights are assumed to be random and so, the correlation between successive columns of weights should be low. But in the ELM IC, the correlation between successive columns of weights is relatively higher due to chip architecture. Since the DACs converting the input digital number to a current is shared for each row, mismatch between the DACs introduce a systematic mismatch between rows. This systematic variation of the input weight matrix results in increased correlation between columns of input weights. Fig. 2.10 shows the histogram of inter column correlation coefficients for hardware weights and software simulated log normal weights. Greater correlation between hardware weights can alternatively be thought of as a reduction in effective number of uncorrelated weights and thereby, a reduction in number of uncorrelated hidden nodes compared to software simulations. Therefore, the "effective" number of hidden nodes in hardware case is in fact smaller than the number of hidden nodes used in the IC. This explains the requirement of higher number of hidden nodes in hardware to match the performance of software simulations. One major drawback of using general purpose ELM IC for speech recognition is that the ELM based classification requires fixed feature size. Speech or audio signals in general are time varying and the length of the signals can vary from sample to sample. Therefore, we had to use zero padding or clipping in the fixed number of bins strategy to ensure the generated features have same dimension or had to use more memory intensive

Figure 2.10: Histogram of correlation coefficients of input weights

and higher latency fixed number of bins strategy. While we still obtained excellent performance for the dataset, this strategy may lead to decreased accuracy for more complex datasets. There are variants of ELM that are able to handle dynamic feature sizes such as OS-ELM [91], OR-ELM [92] etc. ASICs based on such sequential ELM models might be more suitable for speech recognition.

2.5.5 Feature Representations in Combined Binning

Another significant observation about the experimental results is that the combined strategy consistently outperforms both time based binning and

spike count based binning methods for software as well as hardware simulations. This can be attributed to the synergy produced by combining two disparate representations of the input data (time based features and spike count based features) using a decision layer. To prove the importance of using two different representations, we have obtained the average confusion matrices for both time based binning and spike count based binning using several randomized training and testing sets. The resulting confusion matrices are plotted alongside the confusion matrix for the combined strategy in Fig. 2.11. It can be clearly seen from the confusion matrices that while some of the peaks of the confusion matrices are at the same locations for both time based and spike count based methods, a significant number of minor peaks are at different locations. Therefore, a significant number of those misclassifications occurring for only one of the two binning methods are correctly classified in the combined strategy. We have also tried the combined strategy with fixed number of bins but the accuracy did not improve significantly unlike fixed bin size strategy. The decision layer in the combined strategy modifies the overall accuracy only when the time based and spike count based classifiers have classified the same digit differently. Therefore, to quantitatively analyze the reason for this anomaly, we have checked the correlation between the output of time based and spike count based classifier for both fixed number of bins and fixed bin size strategies. The correlation between the time based and spike count based classifiers is significantly higher for fixed number of bins strategy compared to that of fixed bin size strategy. This might be the reason why fixed number of bins strategy did not offer improved accuracy for combined binning technique while fixed bin size strategy did.

To quantitatively analyze our hypothesis that the correlation matrices produced by time based binning and spike count based binning have peaks at different locations, we have used correlation coefficients. We have cal-

Spike Count Based Binning

Time Based Binning

Combined Binning

Figure 2.11: Confusion matrices for different binning strategies exhibit peaks at different locations for time based and spike count based binning. Hence, a combination of these two methods can eliminate some of these errors.

culated the correlation coefficients between confusion matrices produced by time (and spike count) based binning for different randomized training and testing sets. We have also obtained the cross-correlation coefficients between confusion matrices produced by time and spike count based binning

for same training and testing sets. The spread of the correlation coefficients obtained is shown using the box-plots in Fig.2.12. It is quite evident from the box-plots that confusion matrices produced by the same feature extraction method for different training and testing sets are highly correlated while confusion matrices produced by different feature extraction methods for same training and testing set have lower correlation.

Figure 2.12: Correlation between confusion matrices

2.5.6 Comparison with other methods

Next, we compare our results with reported accuracies in existing literature using the N-TIDIGITS18 dataset. For fixed bin size strategy, [93] obtained an accuracy of 87.65% using CNN and an accuracy of 82.82% using GRU RNN. [75] also obtained 88.6% accuracy using GRU RNN and 86.1% accuracy using LSTM RNN for the same feature extraction technique. For fixed number of bins strategy, [74] obtained an accuracy of 95.08% using SVM. Thus, we can see that the accuracies reported in this chapter outperform those obtained using fixed bin size or fixed number of bins techniques in existing literature. The best case accuracies obtained in this chapter are comparable to that of MFCC based features in previous works (using MFCC based features, [74] obtained an accuracy of 96.83% using SVM while [75]

obtained an accuracy of 97.90% using GRU RNN). However, this comparison is imperfect since we need to account for the power needed in generating more complex features like MFCC. [94] has shown that the power required for MFCC feature extraction is 122 mW on FPGA based implementation and 62.3 mW on ARM based implementation for TIDIGITS dataset using a 32 ms frame size. This is significantly higher than that of feature extraction techniques described in this chapter (Tables 2.2 and 2.3). Also, it is difficult to compare power dissipation of RNN approaches since very few hardware implementations of these networks are reported. As one example, [95] reports a Delta RNN network that uses $453K$ operations per frame of 25 ms (excluding FFT operations to generate features) which is quite comparable to the number of operations needed by the ELM first stage. However, it should be noted that the ELM first stage operations were simple random multiplications which could be easily implemented in low power using analog techniques while the same cannot be said for the RNN.

While more recent and computationally expensive methods such as deep recurrent neural networks can directly handle inputs with varying temporal dimension, most of the existing machine learning based classification algorithms require fixed feature vector size. This puts a restriction on the extracted features. For fixed bin size method, shorter samples require zero padding at the end while longer samples get clipped. For the fixed number of bins method, the entire sample duration is divided into fixed number of bins. Larger number of bins result in features with higher temporal resolution at the cost of higher feature vector size and therefore higher computational cost and vice versa. Therefore, both of these methods result in some loss of information during the feature extraction process but these methods do provide a power efficient way to convert the outputs of event based asynchronous output of neuromorphic sensors to useful fixed length features that can be easily used by existing machine learning algorithms.

2.5.7 Real-time detection of Word Occurrence

For the classification of the dataset so far we have assumed that the start and end of a digit is clearly marked for both training and testing data. But for real time applications, this assumption will not hold. So, we have decided to employ a sliding window technique for automatic detection of start and end of a digit. For the spike N-TIDIGITS18 dataset we have used, no noise was added to the waveforms of the original TIDIGITS dataset. So, the detection of start and end of the digit will become a relatively trivial task. However, the more challenging task is to detect the start and end of the signal in presence of noise. Therefore, we have implemented a threshold-based start and end detection using a sliding window assuming presence of noise. The algorithm detects the start of a digit if the total spike count within the window is higher than the given threshold and rejects the frame as noise if the total spike count is less than the threshold. Once the start of a digit is detected, the upcoming spikes are assumed to be part of the digit until the total spike count within a window is less than the threshold for a certain number of consecutive windows. At this point, the last window where the spike count was higher than the threshold is assumed to be the end of the digit. This ensures that the false end detection is avoided in case there are low spike count windows within the digit. We have set the threshold as a certain % of average spike count per window over all samples and the number of consecutive low spike count windows required to determine the end of a digit is a parameter dependent on the sliding window size.

We have tested this algorithm on best accuracy cases of both fixed number of bins strategy (time based binning, 10 bins/sample) and fixed bin size strategy (time based binning, bin size= 40 ms). We used a non-overlapping sliding window size of 40 ms and 2 consecutive windows with sub-threshold spike count for end detection. For fixed bin size strategy, the accuracy remained same for 10% threshold level and decreased by 0.8% for 20% thresh-

old level. For fixed number of bins strategy, the reductions in accuracy were 2.5% and 3.6% respectively for 10% and 20% threshold level respectively. The diminished effect of start and end detection on the classification accuracy for fixed bin size strategy can be attributed to its indifference towards digit duration and thereby exact start and end time unlike its counterpart. Thus, the fixed bin size strategy seems relatively more noise robust.

In this proposed algorithm, the loss of accuracy stems from three sources, (a) loss of bins at the beginning, (b) loss of bins at the end and (c) loss of part of the digits due to false detection. For the fixed bin size case, only (c) is the major contributor to loss in accuracy while for sliding window case, all three factors contribute to the accuracy loss. Moreover, this sliding window technique introduces some additional latency depending upon the number of sub-threshold spike count windows used for end detection.

2.6 DVS Based Object Tracking

With increasing demand in autonomous vehicles, smart surveillance, human-computer interaction etc, accurate real time object tracking has become a primary research area in computer vision community [96]. Traditional cameras require of hugely increased bandwidth and energy to wirelessly transmit the huge volume of video data. The unique challenges and opportunities offered by camera sensors have led to a sub-field of IoT called the internet of video things (IoVT). Edge computing becomes important in this case to process data locally to reduce wireless transmission [2]. Neuromorphic sensors and algorithms offer a unique low power solution for this case.

With the advent of CNN and deep learning, a number of deep learning based object tracking algorithms have been proposed [97] [98]. Most of these object tracking algorithms have two distinct phases: *a*) region proposal and *b*) object classification. While the region proposal network proposes

multiple bounding boxes per frame where there might be an object, the object classification network runs on the proposed regions and predicts the class of the object. Recent object tracking algorithms have used selective search [99], CNN based region proposal networks [100] etc. for generating region proposals.

Research work on tracking using DVS has mostly been focused on taking advantage of the high temporal resolution to faithfully track high speed objects which is a problem for frame based cameras [101] [102]. [102] demonstrated the capabilities of DVS [103] by performing low latency ball tracking and blocking by a robotic goalie with a 3 ms reaction time. In order to show the benefits of these high speed sensors other research has achieved computationally complex tasks like contour motion estimation, corner detection, natural scene detection accuracy and pose tracking for high speed maneuvers [104] [105] [106]. Mean shift [102], combination of CNN and particle filtering [107] and Kalman Filters [108] have been employed in the past for tracking NVS outputs. While such applications demonstrate the ability of NVS based systems to handle complex tasks, they do not show their applicability to resource constrained systems which is a hallmark of IoT.

We have developed two different region proposal algorithms both of which aim to leverage properties of DVS sensors and spike based data to produce power and memory efficient region proposal networks. The first one is SNN based region proposal network where the asynchronous data from DVS sensors are treated in an asynchronous event based manner and the second one is EBBIOT where we use a hybrid frame based method analogous to fixed bin size method with time based binning (1A) described previously for audio data. We briefly introduce these two methods below.

For this work, AER based event data is acquired using a DAVIS sensor (resolution - 240×180) setup at a traffic junction. This setup captures the movement of various moving entities in the scene and the typical objects in

the scene include humans, bikes, cars, vans, trucks and buses. The sizes of various moving objects vary by an order of magnitude in any given scene (eg: Humans vs Buses) and their velocities also span over a wide range (sub-pixel for humans to 5-6 pixels/frame for other fast moving vehicles) in the same recording. These recordings were manually annotated to generate the ground truth annotations of these objects in the scene.

2.6.1 SNNRPN

In this work, we developed a three layer spiking neural network based region proposal network operating on data generated by the aforementioned neuromorphic vision sensors. The proposed architecture consists of refractory, convolution and clustering layers designed with bio-realistic leaky integrate and fire (LIF) neurons and synapses. The performance of the region proposal network has been compared with event based mean shift algorithm and is found to be far superior ($\approx 50\%$ better) in recall for similar precision ($\approx 85\%$). Computational and memory complexity of the proposed method are also shown to be similar to that of event based mean shift [102]. The proposed algorithm is summarized in algorithm 1. Figure 2.13 shows a sample frame of the input data and corresponding output frame. This work is discussed in detail in Appendix A.

Algorithm 1 SNN based RPN

```

1: for each input event do
2:   modify refractory layer membrane voltages
3:   feed-forward any refractory spike produced
4: end for
5: for each refractory spike do
6:   modify refractory synaptic currents
7:   modify convolution layer membrane voltages
8:   modify recurrent synaptic currents (optional)
9:   feed-forward any region proposal produced
10: end for
11: for each frame do
12:   accumulate all region proposal boxes
13:   for each region proposal box do
14:     if There is an adjacent region proposal box then
15:       combine both boxes to produce larger region proposals
16:     end if
17:   end for
18: end for

```

Figure 2.13: Visualization of RPN input and output: input frame shows a scene with one car and two humans (a) and the corresponding output frame shows the region proposals in red (b). The denoising in the output frame is done by the refractory layer while the region proposal is done by convolution layer and clustering layer.

2.6.2 EBBIOT

Different from fully event based tracking or fully frame based approaches, we developed a mixed approach where we created event-based binary images (EBBI) that can use memory efficient noise filtering algorithms. We exploited the motion triggering aspect of neuromorphic sensors to generate region proposals based on event density counts with $> 1000X$ less memory and computes compared to frame based approaches. We also proposed a simple overlap based tracker (OT) with prediction based handling of occlusion. Our overall approach required $7X$ less memory and $3X$ less computations than conventional noise filtering and event based mean shift (EBMS) tracking [102]. Finally, we showed that our approach results in significantly higher precision and recall compared to EBMS approach as well as Kalman Filter tracker [109] when evaluated over 1.1 hours of traffic recordings at two different locations. A flowchart depicting the entire algorithm pipeline is shown in Figure 2.14 and Figure 2.15 shows the histogram based region proposal generation process. Details of this work can be found in B.

Figure 2.14: Flowchart depicting all the important blocks in the system: binary frame generation, region proposal and overlap based tracking.

Figure 2.15: A sample EBBI with corresponding X and Y histogram based region proposals.

2.7 Conclusion

In this chapter, we have presented several low-complexity feature extraction techniques to construct an end-to-end speech recognition system using a neuromorphic spiking cochlea and neuromorphic ELM IC. Moreover, the computational complexity, power requirement and memory requirement of the proposed techniques were calculated. Furthermore, we have used both software and hardware simulations of the neuromorphic ELM IC to obtain high classification accuracies ($\sim 96\%$) for the N-TIDIGITS18 dataset.

The proposed fixed number of bins and fixed bin size methods presented a clear trade-off between classification accuracy and hardware overhead where using fixed number of bins gives $\sim 2 - 3\%$ higher accuracy with $\sim 50\times$ more hardware overhead compared to the fixed bin size method. Our strategy of combining two different feature space representations of the input data gives high classification accuracy while using $\sim 25\times$ less memory compared to the fixed number of bins method.

We have also briefly described a spiking neural network based and a hybrid frame and event based region proposal and object tracking algorithms that can efficiently handle asynchronous DVS data and outperforms existing algorithms using a significantly lower power and memory overhead.

Chapter 3

Optogenetics-Inspired Light-Driven Neuromorphic Computing Platform

3.1 Introduction

The success of deep learning in diverse fields such as image classification [42] and face recognition [110] has spurred a renewed interest in the area of artificial intelligence (AI). Despite the impressive progress already demonstrated with conventional CMOS-based programmable architectures [111], innovative neuromorphic hardware approaches are required to emulate the scale, connectivity and energy efficiency of biological neural networks.

Shallow feed-forward networks are incapable of addressing complex tasks like natural language processing that requires learning of temporal signals. To address these requirements, we need neuromorphic architectures with recurrent connections and deeper architectures such as deep recurrent neural networks (DRNNs). However, the training of such DRNNs demand a very high precision of weights, excellent conductance linearity and low write-noise- not satisfied by current memristive implementations. Pure-electrical implementations fall behind due to their abrupt switching dynamics and limited number of addressable states, while all-photon systems are disadvantaged by their footprint and complex read-out circuitry.

Optogenetics- a photo-stimulated neuromodulation technique, utilizes

optical pulses to monitor and manipulate neuronal activities by controlling ionic currents in biological tissues [112, 113]. Inspired from optogenetics, we propose a unique neuromorphic computing platform comprising of photo-excitabile neuristors (PENs) to selectively control the excitability of artificial neurons and update the weights of synapses with high linearity to obtain precise weights necessary for DRNNs.

Combining optical writing and electrical erasing schemes, we propose a new method of optical writing to transfer offline learnt weights on to the device, after which the device is used for inference in electrical mode only without requiring optical inputs resulting in high energy efficiency due to in-memory computing. Offline learning enables DRNN training with advanced learning rules while the exquisite write linearity afforded by the optical gating is the major phenomenon we exploit to get very accurate weight transfer with a two-shot write scheme. It is also critical to note that the electrical erase operation now becomes linear due to the previously used optical write pulse. The proposed PEN features an order of magnitude higher LDR than other recent state-of-the-art reports, enabling us to simulate a DRNN for speech recognition with an order of magnitude higher parameters than digit recognition networks.

The remainder of this chapter is organized as follows: Section 3.2 discusses the relevant literature. Section 3.3 details the materials used and proposed methods. Section 3.4 describes the results and simulation details. Finally, section 3.5 summarizes the key conclusions of the work.

3.2 Literature Review

The first explicit theoretical depiction of memristor was given by Leon Chua in 1971 [114] though earlier references to a similar element may be found in Bernard Widrow's work [115]. According to circuit theory, the four fun-

damental electrical quantities of a two terminal device can be: 1) voltage (v), 2) current (i), 3) charge (q) and 4) flux (ϕ). Thus a two terminal device is characterized by the static relationship [116] between any two of the aforementioned quantities. Resistors represent a static relationship between v and i , capacitors represent a static relationship between q and v while inductors represent the relationship between i and ϕ . Building upon this rationale, Leon Chua predicted a new fundamental two terminal device that represents similar relationship between charge (q) and flux (ϕ) and named it memristor. Memristors are programmable two terminal resistors. The resistance state (often referred to as memristance) of a memristor can be tuned by applying an electrical excitation.

In past two decades, memristive properties has been demonstrated in a number of nano-scale devices, such as titanium dioxide ([117–119]), amorphous Silicon ([120]) etc. The prevalence of memristance in nano-devices can be attributed to the fact that in such small devices, small voltages can produce large electric fields capable of altering structural properties of the devices.

Similarity between the programmable resistance state of memristors and the variable synaptic strengths of biological synapses dramatically simplifies the design of NCS. As a result, we have seen significant interest in developing memristor based neuromorphic systems ([121–123]). Due to compact size and stackability there are also considerable research effort into developing resistive memories as a replacement for flash memory ([124, 125]). Besides, memristors can be easily integrated into the crossbar architecture which naturally supports parallel computing and high-density storage. With the nanoscale, two-terminal, solid-state memristors, the memristor crossbar (MC) is a parallel computing circuit boasting a high density and low power consumption. Overall, neuromorphic computing with MC represents a possible path for efficient implementation of neural networks.

Therefore, the most popular neuromorphic architectures based on memristive devices is memristive crossbars since the parallelism of this architecture is inherently advantageous to neuromorphic neural network implementations [126]. Memristive crossbars are particularly suitable for in-memory computing since each cell is capable of both processing and storing information. Therefore, NCS based on memristive crossbar provides significant energy scaling over traditional systems [127].

The first wave of neuromorphic hardware solutions based on electrical memristors have demonstrated advantages in parallel summing and update operations, scalability, cost, and power consumption [128]. However, the switching dynamics of conventional memristors result in abrupt transitions and write non-linearity, resulting in asymmetric weight updates and limited number of accessible states [33]. Such limited precision weights can only be used for shallow feed-forward networks and are incapable of addressing applications like speech recognition and natural language processing (NLP) that require learning of temporal signals. To address these requirements, we therefore need neuromorphic architectures with recurrent connections and deeper architectures (> 10 hidden layers) such as deep recurrent neural networks (DRNNs). However, the efficient training of DRNNs calls for fully-parallel write/read operations, which demands non-volatile memory elements with multiple linearly-distributed conductance states addressable via blind updates. This not satisfied by current electrical implementations. A potential way to accelerate the training process of DRNN is to exploit the increased speed offered by optical processing platforms. However, such all-photonics-based computing platforms require meticulous integration of numerous optical components with a large footprint [129] and are incapable of updating weights in a precise, linear manner, necessary for training DRNNs via parallel updates [130]. Present electrical and optical approaches therefore do not sufficiently address the effective bit precision and accuracy required

for energy-efficient computation-in-memory implementation of DRNNs.

This motivates us to re-examine neuroscience, specifically those based on optogenetics, for a second wave of neuromorphic devices for advanced AI applications [131]. With higher speed and spatiotemporal precision over its electrical counterparts, optogenetics has facilitated precise probing of neuronal circuits [131], unravelling underpinnings of cognition and memory. Beyond probing the neural pathways, optogenetics is now extensively employed to stimulate and silence neural activity to manipulate locomotion [132] and rewire neural pathways to cure disorders [133]. This strategy of utilizing light to tune the learning and memory behaviours can be adopted to selectively activate artificial neurons and synapses in hardware circuitry to address the drawbacks of limited number of accessible states, non-linearity and asymmetric weight updates.

3.3 Materials and Methods

3.3.1 Photo-sensitive Synapses

An optoelectronic neuromorphic architecture calls for the need of atomically-thin low-bandgap semiconducting channels as active switching matrices, that would enable intimate opto-electronic control over the charge carrier concentration and conductance transitions to functionally mimic the role played by rhodopsins in optogenetics. 2D transition metal dichalcogenides (TMDCs) offer several attractive features in this regard with their atomic-level scalability, tunable bandgap and opto-electronically modulatable carrier concentrations [134]. we demonstrate PENs based on Rhenium Disulfide (ReS₂) as optogenetic actuators modulating neural excitation and synaptic plasticity with optical stimuli. Designed in a field-effect configuration, the PENs harness the gate-controlled memresistance of the semiconducting channel, modulated additionally via photon pulses.

From the algorithmic point of view, an ideal synapse should typically satisfy linear weight updates for high classification accuracies using online blind programming schemes [135]. Even for transferring offline learnt weights with high bit precision to a neuromorphic hardware necessary for speech recognition or NLP, blind weight updates without iterative read-write procedures are necessary to avoid prohibitively long write times for large-scale DRNNs [136]. Update linearity and write noise have been identified as the prime analog device properties which degrade the accuracy of neural networks [33]. However, most memristive systems depict a nonlinear and asymmetric response due to their abrupt filamentary switching physics, failing to sufficiently address the bit precision and accuracy required for in-memory computations in DRNNs. We propose to exploit the exquisite write linearity and low write noise afforded by the optical gating to get very accurate weight transfer with a two-shot write scheme. It is also critical to note that the electrical erase operation now becomes linear due to the previously used optical write pulse.

To demonstrate the extremely high density and linearity of non-volatile states available for analog computation, the PENs were subjected to an input optical pulse train of constant pulse width and interval. Upon illumination, the conductance updates exhibited excellent linearity, non-volatility and retention characteristics. Real-time monitoring of the conductance changes revealed a precise stepped linear increase in conductance from 1.25 to 180 nS with a step size of ~ 0.18 nS, equivalent to 980 distinct conductance states (~ 10 -bit). Shorter activation wavelengths resulted in higher slopes, wider conductance ranges, higher retention and SNR.

3.3.2 Optical Write and Electrical Erase

During each optical “write” operation, the increase in conductance was attributed to the photo-generation of carriers in the semiconducting channel,

while during each “read” operation, the conductance state remained stable demonstrating excellent non-volatility and retention due to the persistent photoconductivity (PPC) effect [137,138]. The subsequent photo-generation pulses added on to the carrier concentration resulting in a near-ideal conductance linearity. The switching transitions depended solely on the accumulation and retention of photo-generated carriers. Increased photo-dosage resulted in a larger number of carriers occupying the sites of the local potential minima, leading to slower recombination, higher retention and a slower forgetting process, in accordance with the random local potential fluctuation (RLPF) model [139,140]. The number of distinct states were determined primarily by the programming pulse resolution and recombination kinetics of the photo-generated carriers, and hence, the programming pulses could be optimized accordingly to achieve a near-perfect linearity. Hence, the number of states demonstrated in this work is by no means an upper limit for this concept. The extent/range of the number of possible linearly accessible states depends on the linear/triode region of operation of our PENs and the resolution of the pulsing measurement set-up. While optical gating enabled linear incremental non-volatile “write” steps, electrical gating facilitated the “erase” process via defect-assisted recombination of the photo-generated carriers. Defects at the semiconductor-dielectric interface have been observed to act as carrier trapping and detrapping centers, causing hysteresis in current transient measurements [141,142]. Here, the “erase” process modulated via the electrical gating created electron recombination centers, erasing the excess photo-generated charge carriers accumulated during the “write” process.

3.3.3 Linear Dynamic Range

To assess the benefit of the high write linearity and low write noise provided by the opto-electronic write-erase operation, we simulate several neural net-

works for image and speech recognition. The two parameters of linear range and write noise can be combined into one metric, linear dynamic range (LDR), defined as follows:

$$LDR = \frac{\text{Range of conductance over which updates are linear}}{\sigma} \quad (3.1)$$

where σ denotes standard deviation of write noise. Intuitively, LDR points to the maximum number of states that can be available from a device for blind updates since the standard deviation of write noise sets the minimum step size. For either potentiation or depression, we denote by ΔG the step size of conductance change for application of a pulse of duration T_s .

3.3.4 Two-shot Write Scheme

Figure 3.1: Highly linear weight update of PENs allow us to transfer high precision weights from offline learnt Deep NNs

In this work, we propose to train neural networks offline and then transfer the weights by electro-optic means to the PEN crossbar for electrical inference. We use offline learning of weights followed by optically-assisted weight transfer to the PEN crossbar which can then perform the inference operation in electrical mode with extremely low energy dissipation (Figure 3.1). The advantages of this method are as follows:

- Previously reported work [39, 40] using online learning for memristors could only use stochastic gradient descent (SGD) to train fully connected networks (FCN) to classify handwritten digits from the MNIST dataset. However, to train DRNN for classifying complex datasets for speech recognition, it is necessary to use sophisticated momentum based learning rules such as ADAM [143]. Hence, we propose to train the network offline and then transfer the learnt weights on the PEN array with high write accuracy.
- ADAM has several advantages over traditional stochastic gradient descent. Firstly, it contains a momentum term which accelerates the learning process and prevents it from getting stuck at local optima. Secondly, in ADAM, the learning parameters are adapted on per-parameter basis based on previous values of parameter updates which results in smaller updates for frequent features and larger updates for infrequent features. This improvement is particularly useful for sparse gradients. Since the learning parameter adaptation is done based on only past few parameter updates, this optimization technique is also more useful than SGD for online learning and non-stationary objectives. Thirdly, since the learning parameters in ADAM are adapted on per-parameter basis, it is considered more robust towards choice of hyper parameters. Finally, ADAM calculates the bias-corrected parameter estimations which prevents it from being biased towards initial gradient and momentum values. These advantages have made ADAM the most popular choice for gradient descent optimization in past few years [144].
- Our method of weight writes involve optical stimulation, but after the write the inference operation is performed fully in the electrical mode thus still enjoying high energy efficiency afforded by in-memory

computing on PEN memristive arrays.

We next describe the high accuracy weight transfer scheme. From the measured data, we can estimate $\overline{\Delta G}$ and σ as the mean and standard deviation of ΔG across different trials and various initial conductance states. Based on the linearity of the conductance change, we can estimate the write pulse width T_w to reach a desired conductance G_{des} from an initial conductance G_{init} as:

$$T_w = T_s \times \frac{G_{des} - G_{init}}{\Delta G} \quad (3.2)$$

where T_w : Write pulse width to achieve desired conductance T_s : Unit pulse width used in characterization experiments G_{des} : Desired final conductance G_{init} : Initial conductance.

However, it is impractical to estimate the average of ΔG with many measurements to get a best fit line for every PEN. Note that for other non-volatile memory with non-linear characteristics [145, 146], it typically requires 20 – 30 iterations of successive write and measurement to converge to a precise conductance (>1% error). The exquisite linearity and low write noise of the PEN allows us to simplify this procedure and reduce the number of iterations by an order of magnitude as described next.

The quickest method is to estimate the slope $\Delta G/T_p$ as difference in conductance after one write pulse of known pulse-width T_p —we refer to it as the two-shot write scheme where the first write is used to estimate the rate of conductance change of the PEN while the second write is used to program the PEN to the desired conductance value based on the earlier estimated slope. This is illustrated in Figure 3.2 where the initial optical write (Op W) is used to increase the conductance of all devices on a chip to a very large value (greater than largest desired conductance) denoted by G_n^{op} for the n-th device. Note that a global optical write is easier to implement in hardware since it does not require optical selectivity. This is followed by a

Figure 3.2: Two shot write scheme for transferring learned weights to PEN crossbar. After an initial optical potentiation of the entire array, one measurement is done to estimate the conductance G_n^{op} for the n-th device. Next, one electrical write (w_1) operation is done for duration T_p followed by a measurement m_2 to estimate change in conductance or slope. Finally, the second write pulse (w_2) is applied with the duration T_{wn} calculated based on the earlier estimated slope.

measurement (m_1) to read this conductance thus eliminating the mismatch across devices in optical write efficiencies. Next, the first electrical write pulse (w_1) of width T_p is applied to help in slope estimation or calibration.

The following conductance measurement (m_2) allows the controller to estimate the slope for the n-th device as $\Delta G_n / T_p$. Then, it can apply the second and final write pulse (w_2) for the calculated width T_{wn} to reach desired final conductance following equation 3.2 but replacing the ideal slope of the best-fit line $\Delta G / T_s$ with the estimate $\Delta G / T_p$ and using $G_{init,n}$ as the conductance of the n-th device after w_1 .

However, this will be prone to estimation error due to write noise, measurement noise induced variability as well as any non-linearity in the device

write characteristics over a large conductance range. To counter this partially, we use a long write pulse with width $T_p = nT_s$ to make this estimation ($n > 1$). This ensures that the mean value of ΔG_n obtained after the calibration write w1 is large compared to the write noise. Also note that measurement noise is assumed negligible compared to write noise in this analysis which is reasonable since measurement circuits shared along columns can be made precise due to relaxed area/power requirements compared to elements within the crossbar. Hence, the actual device conductance, G_{act} obtained after the second write w2 differs from G_{des} and is modelled as follows:

$$G_{act} = (1 + p)G_{des} + \sigma \quad (3.3a)$$

$$p = \tilde{p} + r_p \quad (3.3b)$$

where $(1 + p)$ is the slope and σ is the standard deviation or write noise calculated from actual measured device conductances. The slope p again comprises two components—the deterministic part $\tilde{p}$ due to systematic bias and the random part due to write noise during the first calibration write w1. It is represented by a zero mean random variable r_p whose distribution is determined by the write noise σ .

3.4 Results and Discussions

3.4.1 Device Measurements

Deviation of the measured conductance of one device from a best fit straight line is very little (Figure 3.3) demonstrating excellent linearity throughout the entire conductance range. Combining this with the measured write noise standard deviation, we can calculate the LDR for our PEN as $35.4 \times 980 \approx 34692$. Compared to other recently reported devices [39, 40], our PENs show at least an order of magnitude higher LDR. LDR has been calculated across

Figure 3.3: Fitting a straight line to the measured conductance show very little deviation from linearity (smaller than half of step size).

5 devices and 3 wavelengths and found to vary in the range 6311 – 34692 with an average value of 15102.

Figure 3.4: (A) The slope estimated from the first 16 states (corresponding to $T_p = 16T_s$) results in error in predicting the conductances compared to the best fit line which measures all the states. Device data from Figure 3.3 is replotted showing the last few states where the difference between the best-fit line and the estimated line based on the two shot write scheme are shown. (B) Deviation of the conductances for all the states in Figure 3.3 are replotted showing the slope estimation method results in a systematic error component that increases monotonically reducing the effective number of states within linear range.

Next, we focus on the effect of the systematic nonlinearity component $\tilde{p}$. For the best fit straight line obtained by fitting a line to the conductance of all 980 states (for the device data in Figure 3.3) averaged across multiple write cycles, we get the slope error $p \approx 0$. For $T_p = 16T_s$, $|\tilde{p}| = 0.0008$, which results in an LDR ~ 600 lower than the earlier calculated value of 34692. The difference between the best-fit line and the estimated line based on the two-shot write scheme are plotted in Figure 3.4a for the data corresponding to the last few states of one device. Note that the effect of write noise is minimized here by averaging the data across multiple write cycles. Here, the slope estimation is done based on the initial 16 states corresponding to $T_p = 16T_s$. It can be clearly seen that there is a systematic error in the prediction at the higher states for a straight line fit based on the initial 16 states. The error is plotted for all the states in Figure 3.4b and indeed shows a distinct increasing trend with the states. Hence, it crosses the limit of half step size at an earlier point reducing the effective number of states (ENOS) to ~ 600 from the original number of 980 states.

Both the potentiation and depression curves are modelled in the above manner and the corresponding parameters are calculated for all 3 wavelengths across 5 devices. The resulting values of $\tilde{p}$ are in the range of $[-0.0048, 0.006]$ resulting in ENOS varying between 82 – 980. The minimum change in conductance that can be reliably obtained is limited by σ , while the effective number of bits can be obtained by the range of conductances available.

We can theoretically derive an equation for effective number of states (ENOS) limited by slope error by equating the error to half step-size as follows:

$$ENOS(1 + \tilde{p})\Delta G \leq ENOS + \frac{\Delta G}{2} \quad (3.4a)$$

$$\implies ENOS \leq \frac{1}{2\tilde{p}} \quad (3.4b)$$

where ΔG denotes the step size. Combining this equation with the earlier equation 3.2, we can obtain a final equation for LDR as:

$$LDR = \min(ENOS, \frac{\text{Range of conductance}}{\sigma}) \quad (3.5)$$

Figure 3.5: Violin plot of Slope estimation variance (r_p) as a function of write noise. Noise level σ corresponds to $\Delta G_{avg}/\sigma = 8$. The length of each violin corresponds to the spread of r_p for a given write noise while the width corresponds to the frequency of occurrence. Higher write noise results in higher variance of r_p .

Next, we analyze the effect of write noise induced slope error captured by the variable r_p . Intuitively, we expect the variance of r_p to increase with increasing amounts of write noise σ . Since our devices show low noise or high

SNR, we explored the effect of noise on r_p over a much larger range of noise levels. Figure 3.5 shows the distribution of r_p for varying amount of write noise. The figure shows that the variance of r_p increases as we increase the noise level. This can also act as a guideline for determining slope estimation variability for other devices with different write noise.

3.4.2 Neural Network Simulation

We simulate the case where the network is trained offline and the trained weights are written to the neuromorphic device by the earlier described two-shot write scheme. It should be noted that we can also perform online learning with SGD using blind updates as is typically shown for resistive memory crossbars trained to do handwritten digit recognition tasks based on the linearized electrical write and erase operations. However, we focus on the results of the offline learning procedure since the focus of the paper is implementing DRNN which cannot be trained efficiently by SGD.

For the experiments, we trained a CNN model [147] to classify digits from MNIST [49] dataset and a hybrid CNN+LSTM model to classify audio from tensorflow speech recognition challenge dataset [148].

3.4.2.1 MNIST

The MNIST dataset consists of 28×28 pixel images of handwritten digits (0 – 9) from multiple writers and is relatively small compared to the speech dataset. It has been used by most earlier works to report their classification accuracy. Hence, we perform MNIST classification as a proof of concept of our two-shot write method and to enable us to compare with other earlier works as well. The CNN model was trained on 60,000 training samples to classify the 10 handwritten digits and tested on 10,000 testing samples. The network architecture used can be described as: $28 \times 28 \times 1 - 32c3 - 64c3 - m2 - 0.5d - fc128 - 0.5d - fc10$ where the input dimension is $28 \times 28 \times 1$, XcY

Figure 3.6: Accuracy vs linear dynamic range for CNN with device weights when tested on the MNIST handwritten digit recognition task.

represents a convolution layer with X convolution filters with $Y \times Y$ size, mZ represents a $Z \times Z$ 2D max-pooling layer, Pd represent a dropout layer with dropout rate P and fcL represent a fully connected dense layer with L neurons. The maximum accuracy obtained by the network for 980 states and ΔG estimation using $T_p = 16T_s$ is 99.09% ($\pm 0.1\%$) which is almost same as original model accuracy (99.2%). Figure 3.6 shows the accuracy of the model with varying linear dynamic range (obtained by varying the number of states). It can be seen that for the MNIST task, even a low

value of LDR (~ 50) is good enough to achieve high accuracy. Since the MNIST task is quite simple, it was not suitable to explore the effect of each device non-ideality separately. We next proceed to do that with the speech classification task.

3.4.2.2 Speech Recognition

For the Tensorflow speech recognition challenge, the dataset consists of 64,727 audio files spoken by thousands of speakers. The audio recordings are 1 sec clips of voice commands such as ‘yes’, ‘no’, ‘up’, ‘down’ etc. The dataset was divided into 90% – 10% for training and testing for 12 class classification (‘yes’, ‘no’, ‘up’, ‘down’, ‘left’, ‘right’, ‘on’, ‘off’, ‘stop’, ‘go’, ‘unknown’ or ‘silence’). The audio clips were converted to 256×101 spectrograms and transposed spectrograms were given as input to the hybrid CNN-RNN model (Figure 3.7). The architecture of the network can be described using aforementioned notations as: $256 \times 101 \times 1 - bn - 64c(5, 7) - 64c3 - m3 - 64c3 - 64c3 - m(2, 3) - 128c3 - 128c3 - m(2, 3) - m(2, 3) - 128c3 - 128c3 - bl256 - 0.5d - fc200 - 0.5d - fc12$ where bn is batch normalization layer, bl is Bidirectional LSTM layer and $Xc(Y, Z)$ represents a convolution layer with X convolution filters with $Y \times Z$ size. The 3.62 million trainable weights in this model is an order of magnitude higher than other simulated networks using resistive memory device to classify handwritten digits. The software model of our proposed network with floating point weights achieved an accuracy of 95.71% on the word recognition task. Then we explored the effect of different non-idealities (write noise, slope estimation error) on the classification accuracy.

Figure 3.7: A deep neural network with 12 trainable layers including convolutional, fully connected and recurrent LSTM layers is used to classify 12 different spoken digits. The detailed architecture of the network is shown with all filter sizes and dimensions mentioned.

Figure 3.8: The accuracy obtained in electrical inference without slope estimation errors is plotted for various number of device states and write noise with constant-LDR lines shown. σ corresponds to measured write noise for one device with $\Delta G_{avg}^2/\sigma^2 \approx 64$. The network can reach close to ideal floating-point accuracy ($\sim 96\%$) with $LDR > 64$ while measured device LDR exceeds 6000 (marked by stars with colour denoting wavelength used). The iso-accuracy contours align with the lines of constant LDR showing the importance of this metric.

First, we investigated the effect of write noise on device accuracy in absence of any slope estimation error ($p = 0$). The results are shown in Figure 3.8. Iso-LDR regions show similar accuracy and the device reaches floating point accuracy for $LDR > 64$.

Next, we explore the effect of both $\tilde{p}$ resulting from systematic bias and write noise. The accuracy variation with number of states fixed at 980 is plotted in Figure 3.9. For the low noise region, the accuracy is primarily determined by the slope estimation error ($\tilde{p}$) and thereby, Effective Number of States (ENOS), while for high noise region, the accuracy is limited by noise. The accuracy is $\sim 0.15 - 2.46\%$ less than floating point accuracy for the measured PEN characteristics.

Figure 3.9: Network accuracy is plotted by fixing number of states to 980 while varying write noise as Figure 3.8 for various slope estimation errors in the two shot write scheme. There is a drop of $\sim 0.15 - 2.46\%$ in accuracy for the measured PEN characteristics (bold circles with colour denoting used wavelength). Original accuracies without slope estimation errors are also plotted using stars for reference. The effective number of states (Y-axis) reduces from the actual number of states due to slope-estimation errors.

Finally, we incorporate the effect of both write noise and estimation error variance resulting from it (Figure 3.5) in determining the classification accuracy. In this experiment, the device is assumed to have no systematic bias ($\tilde{p} = 0$) and the slope estimation error is randomly chosen from r_p distribution corresponding to a specific noise level as shown in Figure 3.5. The results are shown in Figure 3.10. Here, the accuracy conforms to iso-LDR lines for lower noise levels (similar to Figure 3.8) due to low r_p variance. For higher noise levels, the accuracy is significantly reduced due to higher r_p variance. The maximum accuracy obtained by the network for 980 states and ΔG estimation using $T_p = 16T_s$ is in the range of $93.31 - 95.7\%$ for two devices and three wavelengths of light while the accuracy of the original model using floating-point weights is 95.71% . These results are obtained by

Figure 3.10: Network accuracy is plotted by fixing number of states to 980 while including write noise induced slope estimation errors r_p . While the low noise regime is similar to the earlier case showing floating-point accuracy for $LDR > 64$, the network is unable achieve this accuracy for noise exceeding $\sim 4\sigma$. In the intermediate noise region, the required number of states shoots up rapidly to >2000

Table 3.1: Mesured value of non-ideality parameters and device accuracy for two devices and three wavelengths of light.

Case	$\tilde{p}$	$\Delta G_{avg}/\sigma$	Accuracy
1	0.00051	6.438	95.71%
2	0.00083	35.398	95.56%
3	-0.00490	9.633	93.79%
4	0.00605	8.192	93.31%
5	-0.00477	11.766	94.21%
6	0.00463	21.036	94.32%

incorporating all the device non idealities (write noise, $\tilde{p}$ and r_p) for each of the devices. The detailed results are shown in Table 3.1.

From the accuracy plots of the word recognition task (Figures 3.8-3.10) we can conclude that relatively larger LDR is required for word recognition compared to the MNIST task. We hypothesize that this is because of the recurrent neural layers (LSTM) in the speech recognition network which have

Figure 3.11: Accuracy is plotted against LDR with different layers quantized. LSTM layer shows highest sensitivity towards LDR.

the highest number of parameters. Also, any errors in mapping weights may be magnified by the recurrence. To test this hypothesis, we perform three experiments where only the convolutional layer, fully or densely connected layer or the recurrent LSTM layers are quantized using linear quantization. For this experiment, the number of quantization levels is not explicitly set, rather it is implicitly determined by the device non-idealities described earlier. The accuracies are plotted against LDR instead of bit precision. While LDR is not exactly same as bit precision, it provides a measure of implicitly

set number of quantization levels. For the convolutional and fully connected layers, those with the largest number of parameters are chosen. The results plotted in Figure 3.11 indeed show that the accuracy of the LSTM layer is most sensitive to bit precision and drops the earliest when LDR reduces.

Figure 3.12: Reducing conductance slope estimation error by increasing measurement time results in same accuracy as floating-point implementation when the least square fit line using all 980 states is used. Using only 16 points to estimate the slope, results in $\sim 2\%$ drop in accuracy while using 8 points result in $\sim 30\%$ drop in accuracy.

Finally, we explored the effect of reduced estimation error during weight mapping by taking larger estimation times as well as using the best fit straight line to estimate the conductances. As can be seen in Figure 3.12, the classification accuracy of $\sim 93.5\%$ drops by around 30% when the slope estimation time is reduced by half from $16T_s$ to $8T_s$. This accuracy drop

is in accordance with a drop in ENOS from 82.6 to 19.8. The best case corresponds to a least square fit line through all the 980 conductance states. In that case, we obtain back the original accuracy of 95.7% similar to using floating-point numbers.

3.5 Conclusion

In this chapter, we demonstrate that optoelectronic neuromorphic devices can be adapted to execute highly-parallel energy-efficient blind weight-update protocols for DRNNs, accelerated by in-memory computing. In comparison to state-of-the-art, the proposed PEN features an order of magnitude higher LDR facilitating an order of magnitude lower iterations for weight programming, and enabling us to simulate a DRNN for speech recognition with an order of magnitude higher parameters than digit recognition networks [39,40]. Thus, our work extends the frontiers of current neuromorphic devices by enabling unprecedented accuracy and scale of parameters required for online, adaptive and truly intelligent systems for applications in speech recognition and natural language processing.

Although practical implementation of such a large scale neural network using opto-electrical devices is challenging in its current form. However, new, scalable and reproducible growth methods in halide perovskites continue to be developed [149–151]. Numerous demonstrations of large-scale light emitting and photodetector arrays also points to the possibility that such neuromorphic systems could be feasible in future [152]. Through significant progress in the wafer-scale growth of halide perovskites and their heterointegration, the realization of large arrays of neuromorphic elements based on halide perovskites might not be too farfetched. The portability of our demonstrated concepts to other semiconductor systems also point to alternative routes for the concepts to be realized. Opto-electrical conversion

efficiency and selectivity is another challenge . Hence, to get around selectivity we do global optical potentiation followed by selective electrical erase. We show that using optical potentiation makes the electrical erase more linear. Second, to get around high write energy of optical (equivalent to poor conversion efficiency), unlike online learning, we only write weights from offline trained system and then deploy for low-energy electrical inference.

The precise control of device conductances via optical stimuli also opens up possibilities of using light as a non-contact debugging tool to diagnose and correct electrical weights. In short, our optoelectronic neuromorphic computing platform has the potential to allow memristive-based implementations to advance beyond simple pattern matching to complex cognitive tasks such as visual question answering, machine translation and dialogue generation.

Chapter 4

Audio Based Ambulatory Respiratory Anomaly Detection

4.1 Introduction

One of the most prominent application areas of machine learning and deep learning has been in biomedical domain for detection, diagnosis and monitoring of diseases. While image, ECG or EEG based diagnosis have received a significant amount of attention from the machine learning community in past decades, automated audio based diagnosis deserves further exploration. The primary advantages of audio based diagnosis are 1) it is inexpensive and hence has a better affordability for patients 2) it is non-invasive and hence can be used for long term monitoring 3) it doesn't require complex devices and equipment and hence can be easily integrated with wearable devices. In this chapter we will explore strategies for designing audio based respiratory anomaly detection algorithms that can be suitable for long term monitoring of chronic diseases through wearable solutions.

Two most clinically significant lung sound anomalies are wheeze and crackle. Wheeze is a continuous high pitched adventitious sound that results from obstruction of breathing airway. While normal breathing sounds have majority of their energy concentrated in 80-1600Hz [153], wheeze sounds have been shown to be present in the frequency range 100Hz-2KHz. Wheeze

is normally associated with patients suffering from asthma, chronic obstructive pulmonary disease (COPD) etc. Crackles are explosive and discontinuous sounds present during inspiratory and expiratory parts of breathing cycle with a significantly smaller duration compared to the total breathing cycle. Crackles have been associated with obstructive airway diseases and interstitial lung diseases [154].

Auscultation has been used historically for screening and monitoring respiratory diseases. It provides a simple and non-invasive approach to detect respiratory and cardiovascular diseases based on lung sound abnormalities. But these methods suffer from two disadvantages. Firstly, a trained medical professional is required to diagnose a patient based on adventitious lung sounds and therefore, disproportionate number of medical practitioners compared to overall population hinders the speed at which patients are tested. Secondly, even if the patients are diagnosed by experienced professionals, there might be subjectivity in the diagnosis due to dissimilar interpretation of the respiratory sounds by different medical professionals [155].

There can be two possible approaches in designing algorithms and feature extraction techniques for automated detection of breathing sound anomalies. One possible solution is to devise custom feature extraction techniques and algorithms that are highly accurate and very low power but limited in scope i.e. it can only be used for binary classification (normal vs wheeze, normal vs crackle etc.), works with ideal recording condition (relatively low ambient noise), performs ideally only with specific recording device etc. Another strategy is using simple generalized feature extraction techniques (e.g. spectrogram) along with deep learning which will be better at handling multiple types of respiratory anomalies, will be more robust against noise and will have lower dependence on specific recording device used but at the cost of increased power and memory requirement.

From a wearable device perspective, the custom strategy has significant

benefits since the primary constraints of wearable devices are limited memory and computation power. But wearable devices can not be assumed to operate under ideal noiseless environments and the commercial viability of devices geared towards a specific disease or respiratory anomaly becomes limited. Therefore, we need algorithms and architectures that can achieve performance similar to generalized strategy while being able to operate under limited resources of the wearable devices. This is where neuromorphic improvements come in handy.

In this chapter we explore both of these strategies namely custom strategy and general strategy. In custom strategy, we describe a low complexity T-F continuity based algorithm for feature extraction and wheeze detection with high accuracy. Two hardware friendly variants of the algorithm with reasonably high detection accuracy have also been proposed. It has been tested on a small dataset for binary classification. Next, in general strategy, we propose a hybrid CNN-RNN model to perform four class classification (normal, wheeze, crackle, both) of breathing sounds on International Conference on Biomedical and Health Informatics (ICBHI'17) scientific challenge respiratory sound database [156] and then devise a screen and transfer learning strategy to build patient specific diagnosis models from limited patient data. For comparison of our model with more commonly used CNN architectures, we applied the same methodology on VGGnet [157] and Mobilenet [158] architecture. While the proposed model performs admirably in a diverse dataset, the memory requirement of such deep networks is conceivably quite significant. Therefore, we look into neuromorphic techniques and propose a layerwise logarithmic quantization scheme that can reduce the memory footprint of the networks without significant loss of performance.

The remainder of this chapter is organized as follows: Section 4.2 discusses the relevant literature. Section 4.3 and section 4.4 describes the methods and results for custom strategy. Section 4.5 and section 4.6 details the

methods and results for general strategy. Finally, section 4.7 summarizes the key conclusions.

4.2 Literature Review

In the past decade several attempts were made to design algorithms and feature extraction techniques for automated detection of breathing anomalies. Some popular feature extraction techniques used include Mel-Frequency Cepstral Coefficients (MFCC) [159], wavelet coefficients [160], entropy based features [161] etc. Several machine learning (ML) algorithms have been developed in past few years to detect breathing sound anomalies such as logistic regression [162], Dynamic Time Wrap (DTW), Gaussian mixture model (GMM) [163], Hidden Markov Model (HMM) [164] etc. An exploration of existing literature reveals some conspicuous issues with these approaches. Firstly, most of the ML algorithms use manually crafted highly complex features suitable for their algorithms and due to absence of publicly available datasets, it was hard to compare the efficacy of the feature extraction methods and algorithms proposed [165]. Secondly, most of the strategies were developed for a binary classification problem to identify either wheeze or crackle and therefore, not suitable for multi-class classification to detect wheeze and crackle simultaneously [166]. These drawbacks make these approaches difficult to apply in real world scenarios.

Deep learning has gained a lot of attention in recent years due to its unparalleled success in a variety of applications including clinical diagnostics and biomedical engineering [53]. A significant advantage of these deep learning paradigms is that there is no need to manually craft features from the data since the network learns useful features and abstract representations from the data through training. Due to wide success of convolutional neural networks (CNN) in image related tasks, they have been extensively used in

biomedical research for image classification [167], anomaly detection [168], image segmentation [169], image enhancement [170], automated report generation [171] etc. There have been multiple successful applications of deep CNNs in diagnosis of cardiac diseases [172], neurological diseases [173], cancer [174] and ophthalmic diseases [175]. While CNNs have shown significant promise for analyzing image data, recurrent neural networks (RNN) are better suited for learning long term dependencies in sequential and time-series data [176]. The state of the art systems in natural language processing (NLP), audio and speech processing use deep RNNs to learn sequential and temporal features [177]. Finally, hybrid CNN-RNN models have shown significant success in video analytics [178] and speech recognition [179]. These hybrid models show particular promise in cases where both spatial and temporal/sequential features need to be learned from the data.

Since deep learning came into prominence, it is also being used by researchers for audio based biomedical diagnosis and anomaly detection. Some significant areas of audio based diagnosis using deep learning include sleep apnea detection, cough sound identification, heart sound classification etc. Amoh *et al.* [55] used a chest mounted sensor to collect audio containing both speech and cough sounds and then used both CNN and RNN to identify cough sounds. Similarly, Nakano *et al.* [54] trained a deep neural networks on tracheal sound spectrograms to detect sleep apnea. In [56], authors train a CNN architecture to classify heart sounds into normal and abnormal classes. The non-invasive nature of audio based diagnosis make them an attractive choice for biomedical applications.

A major handicap in training a deep network is that a significantly large dataset and considerable time and resources need to be allocated for the training. While the second issue can be solved by using dedicated deep learning accelerators (GPU,TPU etc), the first issue is even more exacerbated for medical research since medical datasets are very sparse and difficult to ob-

tain. One way to circumvent this issue is to use transfer learning. The central idea behind transfer learning is following: a deep network trained in a domain $D1$ to perform task $T1$ can successfully use the learned data representations to perform task $T2$ in domain $D2$. Most commonly used method for transfer learning is using a large dataset to train a deep network and then re-training a small section of the network on the available data (often significantly small) for the specific task and specific domain. Transfer learning has been used in medical research for cancer diagnosis [180], prediction of neurological diseases [181] etc.

Finally, for employing machine learning methods for medical diagnosis, two primary approaches are used. The first one is generalized models where the models are trained on a database of multiple patient data and it is tested on new patient data. This type of models learns generalized features present across all the patients. While these kind of models are often easier to deploy, they often suffer from inter-patient variability of features and may not produce reliable results for unseen patient data. The second approach is patient specific models, where the models are trained on patient specific data to produce more precise results for the patient specific diagnosis. While these models are harder to train due to difficulty in collecting large amount of patient specific data, they often produce very reliable and consistent results [182].

Since a large fraction of medical diagnosis algorithms are geared toward wearable devices and mobile platforms, large memory and computational power requirement of deep learning methods present a considerable challenge for commercial deployment. Weight quantization [161], low precision computation [183] and lightweight networks [158] are some of the approaches used to address this challenge. Quantizing the weights of the trained network is the most straight-forward way to reduce the memory requirement for deployment. DNNs with 8 or 16 bit weights have been shown to achieve com-

parable accuracy compared to their full precision counterpart [161]. Though linear quantization is most commonly used, log quantization has been shown to achieve similar accuracy at lower bit precision [184]. Finally, lightweight networks like MobileNet [158] reduces computational complexity and memory requirement without significant loss of accuracy by replacing traditional convolution layers by depthwise separable convolution.

4.3 Custom Strategy: Materials and Methods

Spectrograms are one of the most common tools to analyze and visualize audio signals. To produce the spectrograms, the audio signals are broken into small time frames and discrete Fourier transform (DFT) is applied on each frame to generate a 2D time-frequency representation of the audio. As evident from Fig.4.1, the spectrograms of wheeze signals are characterized by continuous frequency contours which distinguishes them from normal breathing sounds. These frequency contours are 1) continuous in time 2) varying in shape for different patients and 3) present in different frequency bands for different patients. In the proposed method we amplify the amplitude of the frequency channels which have a similar high amplitude frequency in its neighboring channels in previous time instants while diminish the amplitude of the frequency channels that does not have any such prior high amplitude frequencies. As a result, temporally continuous lines are gradually amplified while isolated noise points are suppressed. We call this algorithm frequency contour tracking (FCT).

(a) Normal Breathing

(b) Wheeze

Figure 4.1: Spectrogram of Breathing Sounds of Wheeze Patient and Normal Subject: frequency contours corresponding to wheeze are marked

(a) Normal Breathing

(b) Wheeze

Figure 4.2: Spectrogram of Breathing Sounds of Wheeze Patient and Normal Subject After applying FCT : Spectral lines corresponding to wheeze are amplified while noise is suppressed

4.3.1 Software Method

In the first step of our proposed method, we use short-time Fourier transform (STFT) with a fixed size overlapping window to obtain the spectrogram of the wheeze signal. The input to the feature extraction algorithm is the spectrogram of the signal. The algorithm requires three predefined parameters, Ch_num : Total number of frequency channels to consider from each window output, L : No. of high amplitude channels selected from each window output, F_d : Neighborhood size. The algorithm works as follows:

Algorithm 2 FCT: Software Method

```

1: for each window(i) output do
2: Find  $L$  channels with largest amplitude:  $f_L(i)$ 
3:   for each  $f_n \in f_L$  do
4:     if  $\exists f_k \in f_L(i-1)$  s.t.
5:        $f_n - F_d/2 \leq f_k \leq f_n + F_d/2$  then
6:          $Amp(f_n(i)) \leftarrow Amp(f_k(i-1)) + \Delta$ 
7:       else
8:          $Amp(f_n(i)) \leftarrow 0$ 
9:       end if
10:    end for
11:  end for
12: for  $j = 1: Ch\_num$  do
13: Feature Vector(j)  $\leftarrow \sum_i^N Amp(f(i, j))$  ( $N$ =No. of windows per frame)
14: end for

```

In this algorithm, first the L frequencies with largest amplitudes are selected from each window. Then the amplitude corresponding to a certain frequency channel (out of L frequencies) increases if there is any large amplitude frequency channel within its neighborhood in the previous time window and the amplitude becomes 0 otherwise. The feature vector is the sum of amplitudes corresponding to each frequency over the duration of a frame. Thus the length of the feature vector is equal to the number of frequency channels (ch_num). As the amplitude increases linearly ($O(n)$) with the duration of continuous contours, the amplitude of the feature vector grows by $O(n^2)$. In Fig. 4.2 we can see the output spectrogram after applying the algorithm. It can be clearly seen that most of the noise points are removed while only frequency contours are present. Fig. 4.3 shows the sum of

(a) Before FCT

(b) After FCT

Figure 4.3: Feature Vectors Computed Before and After Applying Feature Extraction Algorithm: Sharp peaks at frequencies corresponding to spectral contours are visible after FCT is applied

channel outputs (extracted feature) before and after applying the algorithm. Wheeze and normal signals show indistinguishable channel response before the algorithm is applied. But the feature vector corresponding to wheeze signal shows a clear peak corresponding to the location of spectral contours after the algorithm is applied. The trade-off between noise suppression and signal amplification is achieved by tuning algorithm parameters F_d and L .

We checked the validity of the feature extraction on the dataset which contains breathing sounds of 6 normal subjects and 18 wheeze patients collected in practical environment (either in the clinic or a local hospital). IRB approval and informed consent of patients were obtained prior to the data collection. Breathing sounds were recorded over the right side of the chest using an acoustic sensor. The details of the database and data collection method are described in [185]. Each sample (duration : 9-12sec, re-sampled at $4kHz$) was divided into 3 sec frames and each frame was used to obtain one feature vector. We used a majority voting layer to obtain the sample accuracy from frame accuracies. The spectrograms were obtained using a 60 ms hanning window with 50% overlap.

For the binary classification task based on the extracted features, random forest (RF) algorithm has been used throughout this work. For consistency, the size of the random forest is kept constant (50 trees) for all the experiments. 3-fold cross-validation was used to obtain the classification accuracies.

A design space exploration was done to determine optimum parameter settings of the algorithm. From Fig.4.4, we can infer that the accuracy reaches its peak value at $Ch\_num = 100$ and then slightly decreases for larger Ch_num values (larger Ch_num results in larger feature vector size and therefore, higher classification complexity). Fig. 4.5 shows the optimum values of L as a function of Ch_num . While the value of L increases almost linearly with number of channels, the maximum value of F_d is limited to 5.

Figure 4.4: Accuracy vs. No. of Frequency Channels: At least 50 Channels are required for reasonable accuracy

Figure 4.5: L_{opt} vs. No. of Frequency Channels: L_{opt} is almost linearly dependent on No. of Channels

Higher value of F_d results in a significant increase in false contour detection.

4.3.2 Hardware Method

We developed two low complexity variants of this algorithm for hardware implementation. Here instead of sorting the frequencies based on amplitudes and finding L largest components, we used thresholding to find the frequencies with amplitudes larger than a threshold, since it is computationally less expensive. We used two variants, one with fixed threshold (FT) and one with adaptive threshold (AT). The algorithm is as follows:

Algorithm 3 FCT: Hardware Method

```

1: for each window(i) output do
2:   Update Threshold. (Only for adaptive threshold)
3:   Apply threshold.
4:   for each channel(j) do
5:      $out(j) = out(j - F_d/2) + \dots + out(j) + \dots + out(j + F_d/2)$  (where '+' represents logical OR)
6:     if  $out(j) = 1$  then
7:       Counter Output(j)  $\leftarrow$  Counter Output(j) + 1
8:     else
9:       Counter Output(j)  $\leftarrow$  0
10:    end if
11:  end for
12: end for
13: Add Counter outputs till end of frame.
```

Each channel output is considered 1 if its amplitude is greater than the threshold and 0 if its amplitude is less than the threshold. In adaptive thresholding, an additional adder is used to check the number of channel outputs above threshold and the threshold value is adjusted (multiplied or divided by a constant learning parameter) based on the value. We apply an array of OR gates to F_d consecutive channel outputs to determine if there is any high(1) amplitude frequency in the neighborhood. In the next stage the counter increments for OR gate output 1 and resets for OR gate output 0. Finally in the last stage an array of adders (one for each channel) sums up the outputs of each channel over a frame to produce the feature vectors. For $Ch\_num = 100$, $L = 10$ and $F_d = 3$, fixed threshold requires: 100 comparisons/window, 200 ORs/window, 15 counter increment and decrements/window (avg.) and 100 additions/window. Adaptive threshold method requires an additional 100 additions/window. As frame

size is $3sec$ and one window output is generated each $30ms$, the total no. of operations per frame is approximately $42kops$ for fixed thresholding and $52kops$ for adaptive thresholding.

4.4 Custom Strategy: Results and Discussions

Table 4.1 compares the highest accuracies obtained by software and hardware methods. We also compared the performance of two earlier feature extraction methods 'EBWD' [161] and '2-D Threshold' [186] with our results since the datasets used in these papers are similar to ours. All the results were obtained using 3-fold cross-validation. The performance of our method is superior to that of EBWD while it is similar to '2-D Threshold' method. However, in our hardware methods, after the threshold is applied, the data is purely binary numbers while in 2-D threshold method, they are floating point numbers and complex functions like logarithm needs to be calculated. Thus, in an microcontroller based implementation our method requires $3\times$ less computations and an ASIC based implementation requires $6\times$ less computations. We checked the performance of the algorithm in

Table 4.1: Accuracy Comparison (%)

	EBWD	2-D Thresh- old	Proposed SW Method	Proposed HW Method (FT)	Proposed HW Method (AT)
Frame Acc.	71.47	90.65	92.33	87.23	89.22
Sample Acc.	84.38	95.63	99.38	93.13	96.25

presence of noise by adding white Gaussian noise to the signal as shown in

Fig. 4.6. Here the algorithm parameters are set to $Ch\_num = 100$, $L = 5$, $F_d = 3$.

Figure 4.6: Accuracy vs. SNR: Frame and Sample accuracies are 65% and 70% respectively for SNR=-10dB and increase to 89% and 99% respectively at SNR=7dB.

Finally, compared to traditional record and transmit strategy (Method I), several on-node processing strategies can provide significant reduction in power consumption. The audio signals can be processed on wearable and feature vectors acquired using FCT can be transmitted (method II) or only the class labels obtained after classification can be transmitted (Method III). Alternatively, only the frames labeled as wheeze can be transmitted for further analysis (Method IV). Obviously, these methods present us with a trade-off between processing and transmission power and choice of optimum method depends on hardware and power constraints. Assuming the audio recording is sampled at $4kHz$ at a bit depth of 16 bit and frame size is 3sec, we compared the power requirement for arithmetic operations and transmission for these strategies (Table. 4.2.). We used the specifica-

tions of *Apollo2 Ultra-Low Power Microcontroller* for calculating processing power ($10\mu A/MHz$ at $3.3V$ [187]) and *Bluetooth low energy (BLE)* (power $\approx bitrate \times 40nJ/bit$ [188]) for calculating transmission power. We assume a trained random forest (with 50 trees and 20 avg. nodes/tree) is used for binary classification. We also assume each arithmetic operation requires 3 instruction cycles. The signal is analyzed using 256 point FFT (approx. $6.7K$ operations per FFT) with window size 60 ms and 50% overlap.

With traditional methods (record and transmit), the transmission power is the main bottleneck. The use of feature extraction algorithm and on-device classification considerably reduces transmission power overhead and power consumption is primarily comprised of processing power which can be reduced significantly using power efficient microcontrollers. Moreover, as we can see from the Table. 4.2 the bulk of processing power is spent on computing the FFTs. This power can be reduced even further using low power digital filter banks.

Table 4.2: Power Requirement Comparison
Method I:Total Transmission, Method II: Processing + Transmit Feature vector, Method III:Processing+ transmit label, Method IV:Processing + transmit frames with wheeze.

Methods	Proc.	Ops/ frame	Proc. Power	Tx. bits/ sec	Tx. Power	Remarks
I	-	-	-	$4k \times 16 = 64k$	2560 μW	-
II	FFT + FCT	$670k + 52k = 732k$	24 μW	$8 \times 100/3 \approx 267$	10.7 μW	Feature vec.=100 numbers (8-bit), Tx.once /frame
III	FFT +FCT +RF	$670k + 52k + 1k = 723k$	24 μW	1/3	13 nW	Binary Label Tx.once /frame.
IV	FFT + FCT + RF	$670k + 52k + 1k = 723k$	24 μW	$\alpha \times 4k \times 16 = \alpha \times 64k$	$\alpha \times 2560 \mu W$	α =frac. of frames with wheeze, e.g. α =0.3, Tx.power =768 μW

4.5 General Strategy: Materials and Methods

4.5.1 Dataset

For this work we have used the International Conference on Biomedical and Health Informatics (ICBHI'17) scientific challenge respiratory sound database [156]. This is the largest publicly available respiratory sound database. The database contains 920 recordings from 126 patients. Each breathing cycle in a recording is annotated by respiratory experts as one of the four classes: normal, wheeze, crackle and both (wheeze and crackle). The database contains a total of 6898 respiratory cycles out of which 1864 cycles contain crackles, 886 contain wheeze, 506 contain both and rest are normal. The dataset contains samples recorded with different equipment (AKG C417L Microphone, 3M Littmann Classic II SE Stethoscope, 3M Littmann 3200 Electronic Stethoscope and WelchAllyn Meditron Master Elite Electronic Stethoscope) from hospitals in Portugal and Greece. The data is recorded from different locations of chest: 1) Trachea 2) Anterior left 3) Anterior right 4) Posterior left 5) Posterior right 6) Lateral left and 7) Lateral right. Furthermore, a significant number of samples are noisy. These characteristics make the classification problem more challenging and much closer to real world scenarios compared to manually curated datasets recorded under ideal conditions. Further details about the database and data collection methods can be found in [156].

4.5.2 Evaluation Metrics

In the original challenge, out of 920 recordings, 539 recordings were marked as training samples and 381 recordings were marked as testing samples. There are no common patients between training and testing set. The train-

ing set contains recordings from 79 patients while the testing set contains recordings from 49 patients. For this work we used the officially described evaluation metrics for the four-class (normal(N), crackle(C), wheeze(W) and both(B)) classification problem defined as follows:

$$Sensitivity(Se) = \frac{C_{correct} + W_{correct} + B_{correct}}{C_{total} + W_{total} + B_{total}} \quad (4.1)$$

$$Specificity(Sp) = \frac{N_{correct}}{N_{total}} \quad (4.2)$$

$$Score(Sc) = \frac{Se + Sp}{2} \quad (4.3)$$

where $i_{correct}$ and i_{total} represent correctly classified and total breathing cycles of $class_i$ respectively. Since deep learning models require a large amount of data for training, we use an 80-20 split of patients for training and testing in all our experiments.

4.5.3 Related Work

A number of papers have been published so far analyzing this dataset. Jakovljevic *et al* [164] used hidden Markov model with Gaussian mixture model to classify the breathing cycles. They have used spectral subtraction based noise suppression to pre-process the data and MFCC features are used for classification. Their models obtained a score of 39.56% on the original train-test split and 49.5% on 10-fold cross-validation of the training set.

Kochetov *et al.* [189] proposed a noise marking RNN for the four-class classification. Their proposed model contains two sections: an attention network for binary classification of respiratory cycles into noisy and non-noisy classes and an RNN for four class classification. The attention network learns to identify noisy parts of the audio and suppress those sections and

passes the filtered audio to the RNN for classifications. With a 80-20 split, they obtained a score of 65.7%. They didn't report the score for the original train-test split. Though this method reports relatively higher scores, a primary issue with this method is that there are no noise labels in the metadata of the ICBHI dataset and the paper doesn't mention any method for obtaining these labels. Since there are no known objective methods to measure the noise labels in these types of audio signals, this kind of manual labeling of the respiratory cycles makes their results unreliable and irreproducible.

Perna *et al.* [190] used a deep CNN architecture to classify the breathing cycles into healthy and unhealthy and obtained an accuracy of 83% using a 80-20 train-test split and MFCC features. They also did a ternary classification of the recordings into healthy, chronic and non-chronic diseases and obtained an accuracy of 82%.

Chen *et al.* [166] used optimized S-transform based feature maps along with deep residual nets (ResNets) on a smaller subset of the dataset (489 recordings) to classify the samples (not individual breathing cycles) into three classes (N, C and W) and obtained an accuracy of 98.79% on a 70-30 train-test split.

Finally, Chambres *et al.* [191] have proposed a patient level model where they classify the individual breathing cycles into one of the four classes using lowlevel features (melbands, mfcc, etc), rythm features (loudness, bpm etc), SFX features (harmonicity and inharmonicity information) and the tonal features (chords strength, tuning frequency etc). They used boosted tree method for the classification. Next, they classified the patients as healthy or unhealthy based on the percentage of breathing cycles of the patient classified as abnormal. They have obtained an accuracy of 49.63% on the breathing cycle classification and an accuracy of 85% on patient level classification. The justification for this patient level model is that medical professionals do not take decisions about patients based on individual breathing cycles but

rather based on longer breathing sound segments and the trends represented by several breathing cycles of a patient can provide more consistent diagnosis. A summary of the literature is presented in table 4.3.

Table 4.3: Summary of existing literature on ICBHI dataset

Paper	Features	Classification Method	Results
Jakovljevic <i>et al.</i> [164]	MFCC	GMM+ HMM	sc: 39.56% (original train test split), 49.5% (training data, 10-fold cross-validation)
Kochetov <i>et al.</i> [189]	MFCC	Noise masking RNN	sc: 65.7% (80 – 20 split, four class classification)
Perna <i>et al.</i> [190]	MFCC	CNN	Acc: 83% (80 – 20 split, healthy-unhealthy classification), Acc: 82% (healthy, chronic and non-chronic classification)
Chen <i>et al.</i> [166]	optimized S-transform	ResNets	sc: 98.79% (smaller subset of original data, 70 – 30 split, sample level classification)
Chambres <i>et al.</i> [191]	Multiple features	boosted tree	sc: 49.63% (original train test split), Acc: 85% (original train test split, patient level healthy-unhealthy classification)

4.5.4 Proposed Method

4.5.4.1 Feature Extraction and Data augmentation

Since the audio samples in the dataset had different sampling frequencies, first all of the signals were downsampled to $4kHz$. Since both wheeze and crackle signals are typically present within frequency range $0 - 2kHz$, down-sampling the audio samples to $4kHz$ should not cause any loss of relevant information.

As the dataset is relatively small for training a deep learning model, we used several data augmentation techniques to increase the size of the dataset. We used noise addition, speed variation, random shifting, pitch shift etc. to

create augmented samples. Aside from increasing the dataset size, these data augmentation methods also help the network learn useful data representations in spite of different recording conditions, different equipments, patient age and gender, inter-patient variability of breathing rate etc.

For feature extraction we have used Mel-frequency spectrogram with a window size of 60 ms with 50% overlap. Each breathing cycle is converted to a 2D image where rows correspond to frequencies in Mel scale and columns correspond to time (window) and each value represent log amplitude value of the signal corresponding to that frequency and time window.

4.5.4.2 Hybrid CNN-RNN

Figure 4.7: Hybrid CNN-RNN: a three stage deep learning model. Stage 1 is a CNN that extracts abstract feature maps from input Mel-spectrograms, stage 2 consists of a Bi-LSTM layer that learns temporal features and stage 3 consists of fully connected (FC) and softmax layers that convert outputs to class predictions.

We propose a hybrid CNN-RNN model (figure 4.7) that consists of three stages: the first stage is a deep CNN model that extracts abstract feature representations from the input data, the second stage consists of a bidirectional long short term memory layer (Bi-LSTM) that learns temporal relations and finally in the third stage we have fully connected and softmax layers that convert the output of previous layers to class prediction.

While these type of hybrid CNN-RNN architectures have been more commonly used in sound event detection ([192], [193]), due to sporadic nature of wheeze and crackle as well as their temporal and frequency variance, similar hybrid architectures may prove useful for lung sound classification.

The first stage consists of batch-normalization, convolution and max-pool layers. The batch normalization layer scales the input images over each batch to stabilize the training. In the 2D convolution layer the input is convolved with 2D kernels to produce abstract feature maps. Each convolution layer is followed by Rectified Linear activation functions (ReLU). The max-pool layer selects the maximum values from a pixel neighborhood which reduces the overall network parameters and results in shift-invariance [53].

LSTM have been proposed by Hochreiter and Schmidhuber [194] consisting of gated recurrent cells that block or pass the data in a sequence or time series by learning the perceived importance of data points. Each current output and the hidden state of a cell is a function of current as well as past values of the data. Bidirectional LSTM consists of two interconnected LSTM layers, one of which operates on the same direction as data sequence while the other operates on the reverse direction. So, the current output of the Bi-LSTM layer is function of current, past and future values of the data. We used tanh as non-linear activation function for this layer.

The final fully connected and softmax layers take the output of the Bi-LSTM layer and convert it to class probabilities $p_{class} \in [0, 1]$. Finally, the model is trained with categorical crossentropy loss and Adam optimizer for the four class classification problem. We also used dropout regularization in the fully connected layer to reduce overfitting.

To benchmark the performance of our proposed model, we compare it to two standard CNN models, VGG-16 [157] and Mobilenet [158]. Since our dataset size is limited even after data augmentation, it can cause overfitting if we train these models from scratch on our dataset. Hence, we used Imagenet

trained weights instead and replaced the dense layers of these models with an architecture similar to the fully connected and softmax layers of our proposed CNN-RNN architecture. Then the models are trained with a small learning rate.

4.5.4.3 Patient Specific Transfer Learning

Figure 4.8: Boxplot of normalized breathing cycle duration: Intra-patient normalized breathing cycle duration is computed by normalizing each breathing cycles duration by average breathing cycle duration of the corresponding patient while for Inter-patient variability, the normalization is done by average breathing cycle duration of the entire dataset.

Though most of the existing researches concentrate on developing generalized models for classifying respiratory anomalies, the problem with this kind of models is that their performance can often deteriorate for a completely new patient due to inter-patient variability. This kind of inconsistent performance of classification models make them unreliable and thus hinder their wide scale adoption. As an example of inter-patient variability, we

show the boxplot of inter-patient and intra-patient variability of breathing cycle duration in fig. 4.8. For the intra-patient variability, we normalized each breathing cycle duration of a patient by mean breathing cycle duration of that specific patient while for inter-patient variability, we normalized the breathing cycle durations by the mean breathing cycle duration of the entire dataset. As evident from the figure, inter-patient variability is significantly larger when compared to intra-patient variability.

Also, from a medical professional's perspective, for most of the chronic respiratory patients, some patient data is already available or can be collected and automated long-term monitoring of patient condition after initial treatment is often very important. Though training a model based on existing patient specific data to extract patient specific features result in a more consistent and reliable patient specific model, it is often very difficult to collect enough data from a patient to sufficiently train a machine learning model. Since deep learning models require much larger amount of data for training, the issue is further exacerbated.

To address these shortcomings of existing methods, we propose a patient specific transfer learning model that can take advantage of deep learning techniques even with small amount of patient data available. In this proposed model, the deep network is first trained on a large database to learn domain specific feature representations. Then a smaller part of the network is re-trained on the small amount of patient specific data available. This enables us to transfer the learned domain specific knowledge of the deep network to patient specific models and thus produce consistent patient specific class predictions with high accuracy. In our proposed model we train the 3 stage network on the training samples. Then, for a new patient, only the last stage is re-trained with patient specific breathing cycles while the learned CNN-RNN stage weights are frozen in their pre-trained values. For our proposed strategy only 1.4% of the network parameters are retrained

for patient specific models. For VGG-16 and MobileNet, the same strategy is applied.

4.5.4.4 Weight Quantization

In this proposed weight quantization scheme, the magnitudes of weights of each layer are quantized in log domain. The quantized weight ($\tilde{w}$) can be represented as:

$$\tilde{w} = \lfloor 2^N \times \frac{w_{log} - w_{log}^{min}}{w_{log}^{max} - w_{log}^{min}} \rfloor \quad (4.4)$$

where w_{log} represents the weights (w) mapped to log domain ($\log_{10}(w)$) and N is the bit precision. The total number of bits required to store each weight in this scheme is $(N + 1)$ since one bit is required to store the sign of the bit. Now, the minimum and maximum weights (w_{log}^{min} and w_{log}^{max}) used for normalization can be calculated globally (over the entire network) or locally (for each layer). Since the architectures used here have different types of layers (convolution, batchnormalization, LSTM etc.) which often show different ranges of weights [184], local weight normalization seems to be more logical choice. While local normalization requires minimum and maximum weights of each layer to be saved in memory for retrieving the actual weights, this is very insignificant compared to total memory required to save the quantized weights. Finally, we rounded very small weights to zero before applying log quantization to limit the quantization range in log domain.

4.6 General Strategy: Results and Discussions

4.6.1 Generalized Model

Firstly, we evaluated our model on four class breathing cycle classification. To compare our results with traditionally used CNN architectures, we used VGGnet and Mobilenet. All models were trained and tested on a workstation with Intel Xeon E5-2630 CPU with 128GB RAM and NVIDIA TITAN Xp GPU. The results are tabulated in table 4.4. The results are averaged over five randomized train-test sets. As it can be seen from the table 4.4, the proposed hybrid CNN-RNN model trained with data augmentation produces state of the art results. Both VGG-16 and Mobilenet produces slightly lower scores. The score obtained by the proposed model also outperform results reported by Kochetov *et al.* using noise labels on a similar 80-20 split (Table 4.3). We have also performed a 10-fold cross-validation on the dataset

Table 4.4: Comparison of results

Model	Sensitivity	Specificity	Score
VGG-16	40.63%	86.03%	63.33%
Mobilenet	46.4%	78.2%	62.3%
Our work	48.58%	84.04%	66.31%

for our proposed model and the average score obtained is 66.43%. Due to unavailability of similar audio datasets in biomedical field, we have also tested the proposed hybrid model on Tensorflow speech recognition challenge [195] to benchmark its performance. For an eleven-class classification with 90% – 10% train-test split, it produced a respectable accuracy of 96%. For the sake of completeness, we also tested the dataset using same train-test split strategy with a variety of commonly used temporal and spectral features (RMSE, ZCR, spectral centroid, roll-off frequency, entropy, spectral contrast etc. [196]) with non-DL methods such as SVM, shallow neural network,

random forest and gradient boosting. The resulting scores were significantly lower (44.5 – 51.2%).

4.6.2 Patient Specific Transfer Learning Model

It has been shown by Chambres *et al.* [191] that though it is difficult to achieve high scores for breathing cycle level classification, it is much easier to achieve high accuracy in patient level binary classification (healthy/sick). Hence, we propose a screen and transfer learning model. First, the patients are screened using the pre-trained model and if a patient is found to be unhealthy, the pre-trained model is retrained on the patient data to build the patient specific model that can monitor the patient condition in future with higher reliability. The proposed model is shown in Fig. 4.9. To evaluate the performance of the proposed methodology, we used leave one out validation. Since there are variable number of recordings from each patient in the dataset, of the n samples from a patient, $n - 1$ samples are used to retrain the model and it is tested on the other sample. This method is repeated so that all the samples are in the test set once. We trained the proposed model on the patients in the train set and evaluated it on the patients in test set. Since leave one out validation is not possible for patients with only one sample, we only considered patients with more than one sample. The dataset contains different number of recordings from each patient and the length of the recordings and number of breathing cycles in each recording varies widely. But on average, ≈ 47 patient breathing cycles are used for the fine-tuning of patient specific models.

Now, for an objective evaluation of the proposed method, we need to consider a few things. Firstly, if for a patient, most of the breathing cycles belong to one of the classes, the simplest strategy is to assign that class to any of the breathing cycles in the test set. We define this strategy as majority class and use it as a baseline to evaluate the performance of our

Figure 4.9: Screen and transfer learning model: First the patients are screened into healthy and unhealthy based on % of breathing cycles predicted as unhealthy. For patients predicted to be unhealthy, trained model is re-trained on patient specific data to produce patient specific model which then performs the four class prediction on breathing cycles.

model. The predicted class probability of k^{th} class is defined as:

$$\begin{aligned}
 p_{class^k} &= 1 \quad \text{if} \quad N_B^k > N_B^i \quad \forall \quad i \neq k \\
 &= 0 \quad \text{otherwise}
 \end{aligned} \tag{4.5}$$

where N_B^i is the number of breathing cycles belonging to class i to the specific patient.

Secondly, since we are using patient specific data to train the models, we have to verify if our proposed transfer learning model provides any advantage over a simple classifier trained on only patient specific data. To verify this, we used an ImageNet [51] trained VGG-16 [157] as a feature extractor along with an SVM classifier to build patient specific models. Variants of VGG trained on ImageNet dataset have been shown to be very efficient feature extractors not only for image classification, but also for audio classification [197]. Here we use the pre-trained CNN to extract features from patient recordings and train an SVM based on those features only on the patient specific data.

Thirdly, we are proposing that by pre-training the hybrid CNN-RNN

model on the respiratory data, the model learns domain specific feature representations that are transferred to the patient specific model. To justify this claim, we trained the same model on tensorflow speech recognition challenge dataset [195]. Then we used the same transfer learning strategy to re-train the model on patient specific data. If the proposed model learns only the audio feature specific abstract representations from the data, then a model trained on any sufficiently large audio database should perform well. But, if the model learns respiratory sound domain specific features from the data, the model pre-trained on respiratory sounds should outperform the model pre-trained on any other type of audio database. Finally, we compare the results of our model with pure CNN models VGG-16 and MobileNet using the same experimental methodology.

The results are tabulated in table 4.5. Firstly, Our proposed strategy outperforms all other models and strategies and obtains a score of 71.81%. Secondly, VGG-16 and MobileNet achieves scores 68.54% and 67.60% which signifies pure CNNs can be employed for respiratory audio classification, albeit not as effective as a CNN-RNN hybrid model. Thirdly, results corresponding to speech trained network shows that speech domain pre-training is not very effective for respiratory domain feature extraction. Finally, Imagenet trained VGG-16 shows promise as a feature extractor for respiratory data, although it does not reach the same level of performance as ICBHI trained models.

Table 4.5: Comparison of patient specific Models

Pre-trained Model	Patient Specific Training	Se	Sp	Sc
NA.	Majority Class	46.25%	80.45%	63.35%
VGG-16 (ImageNet)	SVM	50.35%	82.38%	66.36%
VGG-16 (ICBHI)	Dense + Softmax	54.41%	82.66%	68.54%
MobileNet (ICBHI)	Dense + Softmax	51.28%	83.92%	67.60%
Hybrid CNN-RNN (TF speech dataset)	Dense + Softmax	43.01%	74.28%	58.4%
Hybrid CNN-RNN (ICBHI)	Dense + Softmax	56.91%	86.70%	71.81%

4.6.3 Memory and Computational Complexity

Figure 4.10: Local log quantization: Score achieved by VGG-16, MobileNet and hybrid CNN-RNN with varying bit precision under local log quantization. VGG-16 requires minimum bit precision to achieve full precision (fp) accuracy while MobileNet requires maximum bit precision.

Even though the proposed models show excellent performance in the classification task, the memory requirement for storing huge number of weights for these models make them unsustainable for application in mobile and wearable platforms. Hence, we apply the local log quantization scheme proposed in section 4.5.4.4. Figure 4.10 shows the score achieved by the models as a function of bit precision of weights. As expected, VGG-16 outperforms the other two models due to its over-parameterized design [184]. MobileNet shows particularly poor performance in weight quantization and is only able to achieve optimum accuracy at 10 bit precision. This poor quantization performance can be attributed to large number of batch-normalization layers and RELU6 activation of MobileNet architecture [184]. While several ap-

proaches have been proposed to circumvent these issues [198], these methods are not compatible with Imagenet pre-trained MobileNet model since they focus on modifications in the architecture rather than quantization of pre-trained weights. The hybrid CNN-RNN model performs slightly worse than VGG-16 since it has LSTM layer which requires higher bit precision compared to the CNN counterpart [199].

Figure 4.11: Resource comparison: Comparison of normalized computational complexity (GFLOPS/sample) and minimum memory required (Mbits) by VGG-16, MobileNet and hybrid CNN-RNN. MobileNet and hybrid CNN-RNN present a trade-off between computational complexity and memory required for optimum performance.

Finally, we compare the computational complexity and minimum memory requirement of the models. The computational complexity is calculated in GFLOPS per sample during inference. The minimum memory required for a model is calculated as (total number of parameters $\times N^{opt}$) where N^{opt} is the minimum bit precision required for the model to achieve optimum accuracy. The only other major source of computational complexity

for end-to-end classification of respiratory audio is computations required for converting audio samples to their respective mel-spectrograms. Since this computational overhead for feature extraction is significantly less (< 1 MFLOPS) [200] compared to the DL architecture, we can ignore it in computational complexity calculations. Both computational complexity and memory are normalized with respect to hybrid CNN-RNN model and shown in figure 4.11. Due to large number of parameters, even after significant memory compression through weight quantization, VGG-16 requires significantly higher memory. Hybrid CNN-RNN model requires least amount of memory followed by MobileNet. In terms of computational complexity, while VGG-16 requires very high computational overhead, MobileNet is computationally most efficient followed by hybrid CNN-RNN. Hence, MobileNet is more suitable for a hardware system where power is the primary constraint while hybrid CNN-RNN, while less power efficient than MobileNet, achieves better performance at minimal memory footprint.

Now, our proposed system requires data pre-processing, feature extraction and classification only once in each breathing cycle. Therefore, if we consider a ping-pong buffer architecture [201] for audio acquisition and processing, our system needs to perform end to end classification of breathing cycles at a latency smaller than minimum breathing cycle duration for real time operation. The primary computational bottleneck of the proposed system is the DL architecture as mentioned earlier. The number of computations of the proposed architecture is of the same order as Mobilenet as shown in fig. 4.11. Since the minimum breathing cycle duration is > 1 second [202] and the per sample latency of Mobilenet on modern mobile SoCs is only ~ 100 ms [203], the proposed system should easily be able to perform real time classification of respiratory anomalies.

4.7 Conclusion

In this chapter, we have proposed a low complexity T-F continuity based feature extraction algorithm and its power efficient hardware implementation for wheeze detection. The algorithm produces highly distinguishable features for wheeze signals using nominal computational overhead. High classification accuracies were obtained for both software and hardware simulations. The method (coupled with a suitable classifier) can be used for low power implementation of both on-chip wheeze detection and selective transmission strategies. It can be used to develop a low power wearable wheeze detection platform using microcontroller based implementation of the algorithm with commercially available audio recording devices.

Next, we have developed a hybrid CNN-RNN model that produces state of the art results for ICBHI'17 respiratory audio dataset. It produces a score of 66.31% score on 80-20 split for four-class respiratory cycle classification. We also propose a patient screening and transfer learning strategy to identify unhealthy patients and then build patient specific models through transfer learning. This proposed model provides significantly more reliable results for the original train-test split achieving a score of 71.81% for leave-one-out cross-validation. It is observed that trained models from image recognition field, surprisingly, perform better in transferring knowledge than those pre-trained on speech. We also develop a local log quantization strategy for reducing the memory cost of the models that achieves $\approx 4\times$ reduction in minimum memory required without loss of performance. The primary significance of this result is that this weight quantization strategy is able to achieve considerable weight compression without any architectural modification to the model or quantization aware training. Finally, while the proposed model has higher computational complexity than MobileNet, it has minimal memory footprint among the models under consideration. Since the amount of data from a single patient is still very small for this dataset, in future,

this strategy should be explored with larger amount of patient specific data. Further, reductions in computational complexity can be explored using a neuromorphic spike based approach [204, 205].

Chapter 5

Spiking Neural Networks for Heart Sound Anomaly Detection

5.1 Introduction

Artificial neural networks (ANN) trained by deep learning has shown tremendous success in audio, visual and decision making tasks. While these methods are loosely inspired by the brain, in terms of actual implementation, the similarity between mammalian brain and these algorithms is merely superficial. Moreover, more often than not, these algorithms require huge energy for real world tasks due to their computation and memory heavy nature, which limits their potential application in energy constrained scenarios. A prime reason for that is unlike their biological counterparts, these algorithms were designed with the primary goal of increasing accuracy on some benchmark tasks. Spiking neural networks (SNN) bridge the gap between artificial algorithms and the biological model of brain due to their asynchronous spike based signal processing model that closely resembles that of the brain.

While spiking neural networks have largely been interesting due to the promise of delivering brain-like natural intelligence, the recent rise in interest for SNN can be attributed to three primary factors. Firstly, in recent years, deep neural networks have been applied to a variety of fields such as image classification [206, 207], object tracking [98], speech recognition [208, 209],

natural language processing [207, 210], game playing [211] etc. The promise of artificial neural networks in solving real world problems invigorated interest in investigating the capability of spiking neural networks in solving similar real world datasets instead of traditionally used toy problems.

Secondly, in spite of massive success of traditional neural networks, the primary obstacle for real world implementation of these architectures is computational resources. Since most of these deep learning architectures require huge amount of memory and power, they are not particularly compatible with the fast growing internet of things, edge computing and mobile computing paradigms. One probable solution to this problems is power efficient neuromorphic hardware that circumvent the power bottleneck of traditional von-Neumann architectures [45]. Due to the asynchronous spike based data processing architecture of SNNs, they seem particularly lucrative for implementation on these low power neuromorphic hardware.

Thirdly, there have been massive improvements in event based sensors in past few years. A number of very power efficient audio and vision sensors have been developed in recent years [212]. Though traditional computer vision and speech processing algorithms can be applied to the data collected through event based sensors, asynchronous event driven nature of SNNs make them particularly suitable to work in tandem with these sensors.

The primary difference between SNN and traditional ANN is that the former uses more bio-realistic spiking neurons that communicate in the network using binary signals or spikes through connections of synapses. Spiking neurons were originally studied to model the biological neurons in mammalian brain to understand their information processing and pattern recognition capability [6].

If we examine the state of the art works on SNNs ([59], [60], [61], [62]), though they are catching up fast with traditional deep learning, for evaluating the performance of the proposed algorithms, these papers only re-

port accuracies on benchmark datasets such as MNIST [49], CIFAR-10 [50], TIDIGITS [63] etc. But in traditional DL, the results are not limited to benchmark datasets. We have seen real world application and wide spread adoption of deep learning in fields varying from sentiment analysis [213] to stock market analysis [214]. But we are yet to see such real world applications from SNN research community. Due to the higher energy efficiency of SNNs, one interesting area of application can be in biomedical domain for wearable devices. We have already seen success of DL in varied fields of medicine and medical diagnosis [53] and sometimes even surpassing human capabilities [215]. Therefore, for this work, we turn our attention to finding potential biomedical application for deep SNNs.

Cardiac disorders are time critical and needs to be diagnosed as early as possible and with high precision. Heart sounds have been a primary screening tool for diagnosing heart diseases from early days of medicine. Therefore, digital heart sound recording or phonocardiogram (PCG) based detection of heart problems is a central problem in biomedical community [216]. The primary appeal of PCG based cardiac disorder detection stems from the non-invasive nature of auscultation and cheaper cost. Over past decades, researchers have proposed a multitude of feature extraction techniques (MFCC, wavelet, spectrogram etc.) and classification techniques (HMM, kNN, SVM etc.) for automated classification of normal and abnormal PCG [217–219]. With recent success of DL, there have been multiple works that implement PCG based cardiac anomaly detection using CNNs [220–222].

In this work, we first develop and train a CNN based model for classification of normal vs abnormal heart sounds and then convert the proposed model to its SNN equivalent and evaluate its performance. We explore different weight normalization techniques, latency-accuracy trade-off for SNNs and compute computational complexity of the proposed SNN. We also show

computational complexity-accuracy trade-off for the proposed SNN as well as compare it with the equivalent ANN both in terms of accuracy and computational complexity.

The remainder of this chapter is organized as follows: Section 5.2 discusses the relevant literature. Section 5.3 details the dataset, feature extraction and proposed classification methods. Section 5.4 describes the results and simulation details. Finally, section 5.5 summarizes the key observations.

5.2 Literature Review

Different models of spiking neurons have been proposed over the years such as leaky integrate-and-fire model ([223](LIF), spike response model [224](SRM), or Hodgkin-Huxley model [225] etc. These models describe the membrane potential and generation of output spikes by a neuron with relation to experimental neuro-biological results.

Though these models differ in amount of details, typically the inputs to the neurons are a series of spikes that get converted to analog synaptic currents. These synaptic currents are added and integrated in time to generate membrane potential. When the membrane potential reaches a certain threshold, it generates an output spike and the generated spike induces further change in the next neuron. Hence, the internal state variable (membrane potential) of a neuron is a continuous variable but all of the information flow in and out of it is described by binary spikes. It is observed in biological neurons that a neuron can not generate a spike immediately after generating another. In some neuron models, this is taken into consideration as refractory period, a short time period after a spike when the neuron is inactive/does not generate another spike.

As an example, the neuron dynamics of a LIF neuron can be described

by:

$$\tau_m \frac{dV}{dt} = -(V(t) - V_{rest}) + RI(t) \quad (5.1)$$

Where $V(t)$ and V_{rest} are the membrane potential and rest potential, $I(t)$ is the total synaptic current, R is the membrane resistance and τ_m is the membrane time constant. The neuron spikes when $V(t) \geq V_{threshold}$ and then resets.

In an SNN the neurons communicate through direct neuron to neuron connections called synapses. Primarily, these synapses convert the input spikes to continuous analog synaptic currents which in turn affect the post-synaptic membrane potential. Hence, the synapses are often modelled as:

$$I(t) = W \sum_{t_s} f(t - t_s) \quad (5.2)$$

Where $I(t)$ is the synaptic current, W is the synaptic weight and $f(t - t_s)$ is synaptic current kernel that converts an instantaneous input spike at $t = t_s$ to an analog current spread out over time. These synaptic current kernels are often formulated as exponentially decaying functions. The synaptic currents are scaled by synaptic weights and these weights contribute to the learning process of an SNN through synaptic plasticity. Synaptic plasticity is the biologically observed variation of synaptic weights based on pre- and post-synaptic activities. The synaptic weights can be positive or negative and the synapse is called excitatory and inhibitory accordingly. Finally, in some models of the synapses, along with synaptic weights, synaptic delays are also used. Synaptic delays can be thought of as a proxy for delay of signal transmission in dendrites or finite synaptic rise time [226].

There are several factors that contribute to difficulty in training and implementing learning algorithms for SNNs. Firstly, the input and output of spiking neurons are spike trains which result in inherent temporal properties of the neurons unlike simple activation values of their non-spiking counter-

part. This necessitates learning algorithms that can incorporate time varying spike patterns for input and output. Secondly, due to the nature of spike trains, most neuron models are non-differentiable at the time of spike while zero otherwise. Hence, traditional gradient based learning methods such as backpropagation, that are the cornerstone of deep learning, can not be directly applied to SNNs. Finally, the temporal nature of the information processing in SNNs makes credit assignment a much difficult issue than their ANN counterparts.

The learning algorithms of SNNs can be broadly classified into two classes: spike learning [227, 228] and conversion learning [59, 229]. For spike learning, the SNNs are directly trained while for conversion learning, an equivalent ANN is first trained using traditional learning algorithms and then converted to equivalent SNNs. The spike learning algorithms can be further grouped based on several criteria. Single spike algorithms [227] need inputs and output spike trains to have a single spike while multi spike algorithms [228, 230] can handle multiple spike times. While some algorithms are more suited for rate encoded spike trains and try to learn the overall number of spikes instead of their exact timings [228, 231], some are designed for time encoded spike trains and learn precise spike times instead [232, 233]. Although some of learning algorithms use backpropagation with different modifications for learning spike patterns [61, 234], a lot of other biologically inspired learning rules are also used to design other algorithms [231, 235, 236].

In spike learning, the SNN is trained directly using some modified form of backpropagation or bio-realistic local learning rules. The primary advantage of this class of algorithms is that they can use both firing rates and individual spike times to train the network and hence, can take advantage of inherent temporal data processing ability of spiking neurons. But these algorithms often have to handle the discontinuous nature of spike signals to implement effective training algorithms. Another prohibitive factor for

this class of algorithms is the simulation time. Since realistic simulation of spiking neurons are often very time consuming, it is difficult to scale up the spike learning algorithms to deep enough models that can handle real world data. Early example of this class of algorithms are Spikeprop [227], ReSuMe [232], Tempotron [228] etc. While these algorithms try to imitate their biological counterparts effectively by integrating spiking neurons bio-realistic weight learning mechanisms, they are not particularly suitable for large networks and real world applications. Spatio-temporal Backpropagation [237], SLAYER [61] etc are some more modern implementations that can be used to design deep SNNs and have been tested on a number of benchmarking datasets [49, 63] typically used by DL community. The primary advantage of these modern algorithms over their older counterparts is the ability to incorporate multilayer backpropagation in the weight learning process so that these algorithms can take advantage of years of developments in traditional ANNs.

Conversion Learning algorithms rely on training a standard ANN using backpropagation and then converting them into SNN with minimal loss in accuracy. The success of this class of algorithms stems from the fact that they can take full advantage of all the algorithmic advances of traditional neural networks and deep learning. Moreover, since the training is done on the proxy ANN, these algorithms do not need to handle the discontinuous nature of spike signals. For these learning algorithms the activation of neurons are used as proxies for their firing rates in SNN domain. Different strategies in this class of algorithms are concerned with imposing constraints on the ANN so that it can be easily converted to SNN [60, 229, 238] or exploring different models of SNN that can easily approximate different layers of their ANN counterpart [239–241]. The second approach seems more interesting since it imposes no constraint or specific criteria on training the ANN and therefore, any deep network can be easily trained with standard DL tools

and then converted to respective SNNs. This way, these algorithms can take advantage of rich and extensive research developments in deep learning as well as low power operation of SNNs during inference.

5.3 Materials and Methods

In this section, we will describe the dataset, preprocessing and feature extraction strategies as well as CNN and SNN architectures and details of the conversion method.

5.3.1 Dataset

For this work, we have used 2016 PhysioNet/CinC Challenge dataset for classification of normal/abnormal heart sound recordings [65]. The dataset consists of heart sound recordings from 3126 patients with the recording duration varying from 5 seconds to 120 seconds. Each recording was sampled at 2kHz. The data were collected from all over the world and includes recordings from both clinical and non-clinical environment. The sounds were recorded at different locations of the body such as aortic area, pulmonic area, tricuspid area etc and the subjects include a varied age group (from children to adults).

5.3.2 Preprocessing

Firstly, since for some recordings, the diagnosis were marked unsure by clinicians, we removed those samples from the dataset. Then, the signals were filtered by a low pass filter with cutoff frequency 500 Hz to remove background noise. Finally, the signals were down-sampled to 1kHz. The signals were also normalized using their mean and standard deviation.

Since any heart sound recording is likely to include a series of heart sounds, for the success of any heart sound recording classifier, the first step is

segmenting the recording by identifying each heartbeat. Earlier approaches to heart sound segmentation relied on threshold based algorithms [242, 243]. The primary problem with this class of algorithms is their lack of noise robustness and inability to handle wider range of recordings from different equipments, different recording setups etc. More recent approaches use probabilistic frameworks such as hidden Markov models (HMM) [244, 245]. Though these algorithms performed much better than threshold based methods, standard HMM methods are memoryless and current output only depends on the previous state. This was improved upon by Schmidt *et al.* [246] using duration dependent HMM, where the state transition in HMM also depends on likely duration of previous states. Finally, while this work used Gaussian distribution to model observation probabilities, Springer *et al.* [247] improved upon their results by using logistic regression based observational probabilities. In this work, we have used the heart sound segmentation methods developed in [247] to segment each recording into individual heart beats. So, our classification task can be defined as identifying heart sound abnormalities based on individual heart beats.

5.3.3 Feature Extraction

For the classification, we have used two features, spectrogram and cochleagram. To generate the spectrogram, discrete fourier transform is applied to windowed signal:

$$D(k, t) = \sum_{n=0}^{N-1} s(n)w(n-t)e^{\frac{-2\pi kn}{N}} \quad (5.3)$$

Where $s(n)$ is the original signal of length N , $w(n)$ is the window function and $D(k, t)$ represent the k^{th} frequency component at time t . The spectro-

gram is given by the log of magnitude of the frequency components:

$$D'(k, t) = \log(|S(k, t)|) \quad (5.4)$$

We have used a 64 ms window with 75% overlap for the spectrogram. Clipping and zero padding were used to account for variation in length of each heartbeat. Thereby, each heartbeat was converted to a 2D image with its 2 axes being time and frequency. While spectrogram is a very commonly used feature for audio based classification [248, 249], use of cochleagram is relatively rare. The ability of human ear in classification, localization and separation of sound has given rise to interest in studying and modelling human ear by scientists from diverse fields. One of the most prominent computational models of human ear is Lyon passive cochlear model proposed by Richard F. Lyon where the cochlea is modelled using a cascade filter bank with half wave rectifiers and automatic gain control [250]. When an audio is passed through this model, it also results in a time-frequency 2D representation similar to spectrogram but here the values represent firing rates of auditory nerves (figure 5.1). This is called a cochleagram. As mentioned before, the primary advantage of SNNs for biomedical application results from their power efficiency. But another advantage of SNNs over traditional DL is their easy integration with low power event based sensors. So, even though we are using audio recorded from traditional recording devices, we want to check the viability of using event based audio sensors for such tasks since an end-to-end system with neuromorphic audio sensors along with SNN classifier will result in a very power efficient wearable platform. Dynamic Audio Sensors (DAS) (such as the one described in Chapter 1) use models similar to Lyon Cochlear Model to implement bio-realistic audio sensors 2.2a. Therefore, we also passed the heart sounds through Lyon cochlear model to obtain cochleagrams.

Figure 5.1: Lyon Cochlear Model [251] : The model uses a series of cascaded notch filters along with resonators to model the basilar membrane in human ear. The half wave rectifiers (HWR) detect the energy of the signals and multiple stages of automatic gain control (AGC) is used to output neuron firing rates.

5.3.4 Proposed CNN Model and Conversion to SNN

For the classification of heart sound, we used a standard CNN architecture. The architecture of the network can be described using notations similar to chapter 3.4.2.1 as: $F \times T \times 1 - 32c3 - 32c3 - a2 - 0.5d - 32c3 - 0.5d - fc128 - 0.5d - fc2$ where the input dimension is $F \times T \times 1$, XcY represents a convolution layer with X convolution filters with $Y \times Y$ size, aZ represents a $Z \times Z$ 2D average pooling layer, Pd represent a dropout layer with dropout rate P and fcL represent a fully connected dense layer with L neurons. The model is shown in figure 5.2. We used ReLu activation function for all convolutional layers. One deviation of this model from standard architecture is use of average pooling rather than max-pooling. While max-pooling is a staple for most common CNN architectures, it is a bit complicated for SNNs. Different forms of winner-take-all strategies [60, 252] have been proposed over the years to replicate max-pooling in spike domain, but they often result in loss of accuracy. We have also verified that for this specific task, use of

Figure 5.2: Heart sound classification: Preprocessing, feature extraction and architecture of CNN classifier for heart sound classification. Initially, PCG recordings are preprocessed and segmented (described in section 5.3.2), then spectrogram and cochleagram features are extracted (described in section 5.3.3) and finally normal and abnormal heart sounds are classified by a CNN classifier (described in section 5.3.4)

average pooling instead of max pooling does not result in any statistically significant difference in accuracy.

As mentioned earlier, the primary principle of conversion of ANNs to their corresponding SNNs is modelling the spiking neurons such that their firing rates are proportional to the corresponding activation values of the ANN neurons. In this work, we adopt the strategies developed in [240] for ANN to SNN conversion. Membrane potential of a neuron i in a given layer l is given by:

$$V_l^i(t) = V_l^i(t-1) + h_l^i(t) - V_{th}S_l^i(t) \quad (5.5)$$

Where $h_l^i(t)$ represent total input current obtained by summing all individual spike trains multiplied by corresponding synaptic weight. V_{th} represents the threshold voltage and the neuron spikes if the membrane voltage

exceeds V_{th} . $S_l^i(t)$ represent the output spike train. So, the membrane potential integrates total input current over time and whenever it exceeds threshold voltage, the threshold voltage is subtracted from the membrane potential. It has been shown in [240] that this model leads to firing rates proportional to corresponding activation values along with an additive error term.

$$r_l^i(t) = a_l^i r_{max} - \frac{V_l^i(t)}{T \cdot V_{th}} \quad (5.6)$$

Where $r_l^i(t)$ is the firing rate, a_l^i is the corresponding activation value of the ReLu function of the ANN, r_{max} is the maximum firing rate given by the inverse of temporal resolution and T is the simulation duration.

Now, since the input to the network is 2D images, we have to consider conversion of these input values to compatible formats for the SNN. There are different possible solutions such as generating stochastic poisson spike trains with firing rates proportional to the input value. In this work, we use another approach where each input value is represented by constant currents spanning over all timesteps with its magnitude proportional to the analog values. The biases are also represented by constant currents. The average pooling layer is replicated by averaging the firing rates over multiple neurons and the softmax layer is replaced by Relu function in the SNN simulation.

5.4 Results and Discussions

For the experiments, we have divided the dataset in 50-50 train-test split and ensured that heartbeats from same recording do not appear in both training and testing dataset. The model is trained with categorical crossentropy loss and Adam optimizer to classify the heartbeats into two classes, normal and abnormal. The initial learning rate was set at 0.0001. All models were trained and tested on a workstation with Intel Xeon E5-2630 CPU with 128GB RAM and NVIDIA TITAN Xp GPU. For the spectrogram,

we obtained an accuracy of 88.07% while for cochleagram we obtained a slightly lower accuracy of 87.42%. We did not perform further experiments into exploring different features and modification of the DL architecture to improve the accuracy since its already very close to state-of-the-art accuracy [253] and the primary focus of this work is investigating the viability of SNN for such classification tasks. We used SNN toolbox [240] to convert the proposed CNN to its SNN equivalent.

5.4.1 Weight Normalization

The main step in converting an ANN to its SNN equivalent is conversion of activation values to equivalent firing rates. While ANN activation values can assume any value, since the SNN simulation has finite temporal resolution and firing rates are always positive, firing rates of the SNN are limited to the range $[0, r_{max}]$. As a result, there can be two issues, 1) if the number of input spikes is small enough, the neuron will never spike and therefore, no information is transmitted to subsequent neurons. 2) if the number of input spikes is too high, it can lead to firing rates higher than r_{max} and due to finite temporal resolution of the neuron, that information is cut-off from propagating to further layers. The first problem can be solved with a sufficiently large simulation time to ensure the neurons spike while the second issue can be solved by ensuring high temporal resolution. Since both of these approaches require higher amount of time or computational resources, a possible solution to this problem was suggested in [59]. In their work, they suggested passing a subset of training data through the SNN first to measure the maximum possible activation value in a layer and using that value to normalize the weights such that the firing rates always remain in the desired range. We use Max Norm to refer to this method. A drawback of this method is that since there can be some outlier activation values (much larger than the rest), normalizing all the weights using the

Figure 5.3: Effect of different weight normalization methods: both Max Norm and Percentile Norm shows significant performance improvement over no normalization and overall 99.9 Percentile Norm achieves accuracies closest to the original ANN. Spectrogram performs better than cochleagram without normalization while cochleagram performs slightly better than spectrogram after normalization.

maximum activation may result in very low firing rates in other neurons of the layer. A possible solution was suggested in [240] where instead of MaxNorm, Percentile Norm was used. Here, instead of normalizing the weights by the maximum activation, p -th percentile of activation is used.

Therefore, we explore the effect of these weight normalization techniques for our SNN. With a temporal resolution of 1 ms and simulation duration

of 50 ms, we obtained error rates for no normalization, Max Norm and Percentile Norm. The results are shown in figure 5.3. As can be seen from the figure, normalization indeed increases the accuracy significantly. While the difference in accuracy between Max Norm and Percentile Norm is smaller, overall we get best accuracy for 99.9 Percentile Norm there the accuracy reaches within 3 – 4% of the original ANN. Another interesting observation is that while for the ANN and SNN without normalization spectrogram outperforms cochleagram slightly, after normalization, cochleagram produces slightly higher accuracy than spectrogram.

5.4.2 Simulation Duration

As mentioned in the previous section, smaller timestep size and larger simulation duration result in a more accurate SNN simulation. This accuracy-latency trade-off has been reported by a number of previous works [59, 93]. Smaller timestep size and larger simulation time ensures that the information from one neuron to another is represented by a larger number of timesteps. Therefore, this temporal resolution in SNN can be thought of as an analogue of bit resolution in ANN. A more formal relationship between the performance of SNN and simulation duration is described in equation 5.6 where we see the additive error term $(\frac{V_L^i(t)}{T.V_{th}})$ is inversely proportional to total simulation duration (T). This is again analogous to quantization error in ANN. Therefore, for a fixed temporal resolution, larger simulation duration should result in better accuracy. To verify this, we used 99.9 Percentile Norm along with a stepsize of 1 ms and varied the simulation duration to obtain corresponding accuracies. The results are shown in figure 5.4. The errors keep decreasing with increase in simulation time and reach a plateau at around 100 ms. It can also be inferred from the figure that the errors converge slightly faster for cochleagram feature.

Now, during inference, ANN produces the class prediction only once for

Figure 5.4: Effect of simulation duration: Error rates of SNNs approach the original ANN as simulation duration is increased and it reaches a plateau at around 100 ms

each sample while for the SNN, a prediction is produced at each timestep. Therefore, SNN produces continuous prediction unlike the ANN equivalent. While the previous experiment shows the accuracy for different simulation durations, all the accuracies are measured at the end of the simulation. So, for the next experiment, we measure the classification accuracies at each timestep for entire simulation duration. The results are shown in 5.5 for total simulation duration of 100 ms. As can be seen from the figure, with each timestep, a larger fraction of input is presented to the network

and the classification accuracy improves. Both curves for spectrogram and cochleogram show similar convergence pattern and the errors keep gradually decreasing initially but the error rate becomes flat after a certain point. The network reaches very close to its optimum accuracy at only about 50% of the simulation time.

Figure 5.5: Effect of simulation time: classification accuracies are measured at each timestep for both spectrogram and cochleogram. Total simulation duration is set at $T=100$ ms. The network reaches close to optimum accuracy at only about 50% of the simulation time

5.4.3 Computational Complexity

As mentioned earlier, one primary advantage of using SNNs for inference is their smaller energy footprint compared to ANN counterpart. Therefore, we explore the computational complexity of the SNN. In this proposed model of SNN, each computation is instigated by presence of an input spike. Whenever an input spike is presented to a neuron, the membrane voltage updates accordingly. Therefore, if a neuron's output is connected to n neurons in the next layer, each output spike of that neuron will result in n membrane voltage updates. If we consider a neuron i at a layer l (with a total of K_l neurons) connected to n_l^i neurons of the next layer, we can calculate the total number of operations for a network with L layers as:

$$C_{SNN} = \sum_{t=1}^T \sum_{l=1}^L \sum_{i=1}^{K_l} n_l^i \times S_l^i(t) \quad (5.7)$$

where T is the total simulation time and $S_l^i(t)$ is the output spike train. Now, while for fully connected layers, the number of output neurons connected to a given neuron is straight forward, for convolutional layers, the size of the filters and number of channels determine this number. Now, we can see from this equation that the SNN model not only provides us with a latency-accuracy trade-off as discussed previously (figure 5.4, but also presents a computational complexity-accuracy trade-off. Since the SNN produces continuous predictions at each timestep as shown in figure 5.5, and the computational complexity increases with each timestep as shown in equation 5.7, we can explore the relationship of accuracy and computational complexity by calculating the accuracy and number of computations at each timestep during a simulation. Since we have already seen that 100 ms duration is good enough for this dataset, we explored the aforementioned trade-off for 100 ms simulation duration. The results are shown in figure 5.6. The computational complexity of the SNN is shown as a fraction of computational

complexity of the equivalent ANN. As evident from the figure, the SNN reaches within 3 – 4% of the error rate of equivalent ANN with $5\times$ less computations. Moreover, in SNN, the computations are only additions while for the ANN, the computations are MAC operations. Therefore, SNN based inference presents us with an order of magnitude higher power efficiency compared to ANN at the cost of very small accuracy loss.

Figure 5.6: classification error vs computational complexity: Computational complexity of the SNN is shown as a fraction of computational complexity of the equivalent ANN. Firstly, we see that there is a clear trade-off between classification error and computations required. Secondly, the SNN can achieve accuracies within 3 – 4% of the ANN using $5\times$ less computations.

5.5 Conclusion

In this chapter, we explore the viability of audio based biomedical diagnostics using SNN. Firstly, we have shown that bio-realistic cochleagram can provide similar level of performance as traditionally used spectrogram. Secondly, we explore different weight normalization techniques and show that while weight normalization is an essential step for ANN to SNN conversion, the difference in performance between Max Norm and Percentile Norm is rather small for this dataset. Thirdly, we examine the accuracy-latency trade-off for SNNs and show that while with increasing simulation duration the SNN accuracy approaches closer to ANN accuracy, the drop in error rate flattens out at a certain point and increasing simulation duration afterwards result in very small improvement in accuracy. Finally, we calculate the computational complexity of the SNN and show that we can obtain accuracies very close to ANN using the equivalent SNN at approximately $5\times$ less computations. This result is comparable with similar results reported for image classification in [240] where they achieved SNN accuracies withing 1 – 4% of of original ANN at $2 – 3\times$ less computational cost for different image datasets and network architectures.

While this chapter proves the potential of SNN for much more power efficient implementations of audio based biomedical applications, there are a lot of areas for exploration regarding this work. Firstly, since we have already shown the effectiveness of cochleagram for this application, in future similar SNN implementations can be integrated with neuromorphic audio sensors that produce similar feature representations to design much more power efficient end-to-end wearable biomedical solutions. Secondly, although we used constant currents to represent input to the network, different spike encoding schemes need to be explored to improve the performance of the SNN. Finally, while we use the ANN to SNN conversion methods used in [59] and [240], there are other conversion methods such as [241] that need

exploring from an application perspective.

Chapter 6

Conclusion

The recent success of “deep neural networks” (DNN) has renewed interest in machine learning and, in particular, bio-inspired machine learning algorithms. Although these architectures are not new, availability of massive amount of data, huge computing power and new training techniques (such as unsupervised initialization, use of rectified linear units as the neuronal nonlinearity, regularization using dropout or sparsity, etc. [207, 254]) to prevent the networks from over-fitting have led to its great success in recent times. DNN has been applied to a variety of fields such as image classification [255, 256], face recognition in images [257], word recognition in speech [208, 258], natural language processing [207, 210], game playing [211], and the success stories of DNN continue to increase every day.

While these methods are loosely inspired by the brain, in terms of actual implementation, the similarity between mammalian brain and these algorithms is merely superficial. Moreover, more often than not, these algorithms require huge energy for real world tasks due to their computation and memory heavy nature, which limits their potential application in energy constrained scenarios. The mammalian brain is surprisingly efficient at pattern recognition tasks, learning from few examples and spending very little power to compute [259]. Hence, it is natural that scientists and engineers interested in artificial intelligence should draw inspiration from neuroscience.

This is what has drawn researchers from diverse fields such as computer science, electrical engineering and neuroscience towards neuromorphic engineering. Neuromorphic engineering was recently voted as one of the top

ten emerging technologies by the World Economic Forum [260] and the market for neuromorphic hardware is expected to grow to $\sim \$1.8B$ by 2023/2025 [261, 262]. However, cross-layer innovations on algorithms, architectures, circuits, and devices are required to enable adaptive intelligence especially on embedded systems with severe power and area constraints.

In this work, we have explored neuromorphic audio systems from a diverse set of perspectives, neuromorphic audio sensors, novel neuromorphic nano-devices as well as potential biomedical application areas for such systems. In this chapter, we will summarize the key results and observations from this body of work. We will also present some potential ideas for future work.

6.1 Ultra Low Power Speech Recognition Using Neuromorphic Sensors

In this work, we bring together the developments of neuromorphic spiking cochlea sensors and population encoding based ELM hardware to lay the groundwork for a low power neuromorphic real-time sound recognition system and tested it on N-TIDIGITS18 dataset. We introduce two different binning modes (time and spike based) and two different binning strategies (fixed bin size and fixed number of bins) and explore these feature extraction techniques in terms of classification accuracy, computational and memory overhead.

Out of the two binning strategies described, the fixed bin size method is more convenient to implement from a hardware perspective. Moreover, the memory and energy requirements of the fixed bin size method are much less than its counterpart. But the best case accuracy of the fixed bin size method is typically $\sim 2 - 3\%$ less than that of fixed number of bins method. This is due to two factors: lack of input temporal normalization and loss

of information due to discarded bins. If we compare the best accuracy cases of both fixed bin size and fixed number of bins methods, these results show that fixed binning requires $\sim 50\times$ less memory for feature extraction ($\sim 3\times$ overall) and $\sim 30\%$ less energy compared to that of fixed number of bins method. Therefore, these methods presented a trade-off between classification accuracy and hardware overhead where using fixed number of bins gives $\sim 2 - 3\%$ higher accuracy with $\sim 50\times$ more computational overhead compared to the fixed bin size method.

To increase the accuracy of the fixed bin size method, we adopted a combined binning approach. In this fixed bin size strategy, the input data is processed in parallel using both time based and spike count based binning. The feature vectors produced are applied to their respective ELMs and the ELM outputs are combined (added) in the decision layer. The final output class is defined as the strongest class based on both strategies. The combined binning mode not only outperforms both the time and spike count based modes, but also shows accuracies similar to the best case accuracies of fixed number of bins. As the combined binning requires approximately twice the memory and computational complexity than that of the simple time or spike count based binning methods, we can conclude that the combined binning strategy is able to produce accuracies similar to fixed number of bins method using $\sim 25\times$ less memory for feature extraction.

Finally, the proposed feature extraction methods were tested on a neuro-morphic ELM IC described in [64] by feeding the chip with feature vectors produced by the methods described above. One key observation from the results obtained is that the hardware ELM requires larger number of hidden nodes to obtain accuracies similar to the software simulations. While software simulations required around 2000 hidden nodes to obtain optimum accuracy, the hardware required more than 5000 hidden nodes to obtain comparable accuracies. This discrepancy can be ascribed to the higher cor-

relation between input weights in the ELM IC. In an ideal ELM, the input weights are assumed to be random and so, the correlation between successive columns of weights should be low. But in the ELM IC, the correlation between successive columns of weights is relatively higher due to chip architecture. Greater correlation between hardware weights can alternatively be thought of as a reduction in effective number of uncorrelated weights and thereby, a reduction in number of uncorrelated hidden nodes compared to software simulations. Therefore, the "effective" number of hidden nodes in hardware case is in fact smaller than the number of hidden nodes used in the IC.

For real time applications, a speech recognition system also needs to identify the beginning and end of a speech signal even when noise is present. Therefore, we have also implemented a threshold-based start and end detection using a sliding window assuming presence of noise. The experiments show that the fixed bin size method is much less affected by error in detection of start and end compared to fixed number of bins methods.

Finally, we introduce similar strategies for object detection based on DVS sensors. We develop one SNN based and one hybrid technique for power efficient object detection particularly suited for IoVT applications.

6.2 Optogenetics-Inspired Light-Driven Neuromorphic Computing Platform

In this work, we propose a unique neuromorphic computing platform comprising of photo-excitabile neuristors based on Rhenium Disulfide (ReS₂) and simulate a DRNN for speech recognition with an order of magnitude higher parameters than previous works.

Combining optical writing and electrical erasing schemes, we propose a new method of optical writing to transfer offline learnt weights on to the

device, after which the device is used for inference in electrical mode only without requiring optical inputs resulting in high energy efficiency due to in-memory computing. We propose to train neural networks offline and then transfer the weights by electro-optic means to the PEN crossbar for electrical inference. We use offline learning of weights followed by optically-assisted weight transfer to the PEN crossbar which can then perform the inference operation in electrical mode with extremely low energy dissipation. Offline learning enables DRNN training with advanced learning rules while the exquisite write linearity afforded by the optical gating is the major phenomenon we exploit to get very accurate weight transfer with a two-shot write scheme.

For the experiments, we trained a CNN model [147] to classify digits from MNIST [49] dataset and a hybrid CNN+LSTM model to classify audio from tensorflow speech recognition challenge dataset [148]. We explore different non-idealities (write noise, non-linearity, measurement errors etc.) from an implementation standpoint and examine the effect of each of these factors on the performance of the proposed computation platform in great detail.

We also introduce different measurement metrics such as linear dynamic range, slope estimation variability etc. and their effect on performance of the neuromorphic platform. The application of these metrics are not limited to this work only, it can also be used as guidelines for similar works in future.

While existing works on similar novel devices are limited to only simple simulations and basic digit recognition tasks, our optoelectronic neuromorphic computing platform has shown the potential of memristive-based implementations to advance beyond simple pattern matching to complex cognitive tasks. Therefore, this work extends the frontiers of current neuromorphic devices by enabling unprecedented accuracy and scale of parameters required for online, adaptive and truly intelligent systems for applications in speech recognition and natural language processing.

6.3 Audio Based Ambulatory Respiratory Anomaly Detection

In this work, we explore feature extraction and classification strategies for audio based ambulatory respiratory anomaly detection. Since the goal of this work is focused towards wearable devices, we develop low power feature extraction techniques as well as memory efficient implementations of classification techniques.

We explore two different strategies to design feature extraction techniques for power and memory constrained wearable devices, namely custom strategy and general strategy. The custom strategy is to devise custom feature extraction techniques and algorithms that are highly accurate and very low power but limited in scope i.e. it can only be used for binary classification (normal vs wheeze, normal vs crackle etc.), works with ideal recording condition (relatively low ambient noise), performs ideally only with specific recording device etc. The general strategy is using simple generalized feature extraction techniques (e.g. spectrogram) along with deep learning which will be better at handling multiple types of respiratory anomalies, will be more robust against noise and will have lower dependence on specific recording device used at the cost of increased power and memory requirement. Since the higher resource requirement of the general strategy is somewhat incompatible with wearable devices, we introduce novel weight compression techniques that can reduce the memory overhead of deep networks without compromising the performance.

In the custom strategy, we have proposed a low complexity T-F continuity based feature extraction algorithm and its power efficient hardware implementation for wheeze detection. The algorithm produces highly distinguishable features for wheeze signals using nominal computational overhead. High classification accuracies were obtained for both software and hardware

simulations. The method (coupled with a suitable classifier) can be used for low power implementation of both on-chip wheeze detection and selective transmission strategies.

In the general strategy, we have developed a hybrid CNN-RNN model to perform four class classification (normal, wheeze, crackle, both) on a very large respiratory sound database, ICBHI'17 respiratory audio dataset and the proposed architecture produced state of the art results. It produces a score of 66.31% score on 80-20 split for four-class respiratory cycle classification. We also propose a patient screening and transfer learning strategy to identify unhealthy patients and then build patient specific models through transfer learning. This proposed model provides significantly more reliable results for the original train-test split achieving a score of 71.81% for leave-one-out cross-validation.

We also develop a neuromorphic weight compression technique called local log quantization to reduce the memory cost of the models that achieves $\sim 4\times$ reduction in minimum memory required without loss of performance. The primary significance of this result is that this weight quantization strategy is able to achieve considerable weight compression without any architectural modification to the model or quantization aware training. The proposed model along with weight compression outperforms traditionally used models such as VGG-16 and Mobilenet at a reduced memory cost.

6.4 Spiking Neural Networks for Heart Sound Anomaly Detection

In this work, we first develop and train a CNN based model for classification of normal vs abnormal heart sounds and then convert the proposed model to its SNN equivalent and evaluate its performance. We show that SNNs can achieve accuracies similar to ANN at a much lower computational cost.

For this work, we have used 2016 PhysioNet/CinC Challenge dataset for classification of normal/abnormal heart sound recordings [65]. The dataset consists of heart sound recordings from 3126 patients from varied age group recorded at different locations of the body and using different equipments and setups. We have used LR-HSMM heart sound segmentation methods developed in [247] to segment each recording into individual heart beats. So, our classification task can be defined as identifying heart sound abnormalities based on individual heart beats. For the classification, we have used two features, traditionally used spectrogram and more bio-realistic cochleogram. For the classification of heart sound, we used a CNN architecture with 6 layers. The proposed CNN achieved $\sim 88\%$ accuracy on the binary classification task. Then, we adopt the strategies developed in [59, 240] for ANN to SNN conversion.

We explore different weight normalization techniques and show that while weight normalization is an essential step for ANN to SNN conversion, the difference in performance between Max Norm and Percentile Norm is rather small for this dataset. Next, we explore the latency-accuracy trade-off of the SNN reported by a number of previous works [59, 93] and show that while with increasing simulation duration the SNN accuracy approaches closer to ANN accuracy, the drop in error rate flattens out at a certain point and increasing simulation duration afterwards result in very small improvement in accuracy. Finally, we calculate the computational complexity of the SNN and show the computational complexity-accuracy trade-off for the SNN. We also show that the SNN can achieve performance very close to ANN at $\sim 5\times$ less computations.

6.5 Future Work

Our first work shows that combining neuromorphic audio sensors coupled with neuromorphic ASIC can produce a very low power solution for real time speech recognition. While we show the computational and memory cost of all the feature extraction techniques, the feature extraction block is purely simulated in software. A microcontroller based implementation of the feature extraction block as well as threshold based digit detection can be integrated with the DAS and ELM IC to obtain a complete hardware setup for the speech recognition task. Moreover, while this setup is tested only for spoken digit recognition task, similar setup can also be expanded for other audio and more complex speech recognition tasks. Moreover, this study is constrained by the hardware design (signal encoding scheme) of the neuromorphic audio sensors. N-TIDIGITS18 represents only one of the possible spike encoding schemes of audio signals. In future, we plan to explore other spike encoding schemes such as [263] for similar tasks.

In our second work, we develop an optoelectronic neuromorphic computing platform capable of handling deep neural networks with millions of parameters. The precise control of device conductances using optical stimuli shown in this work opens up possibilities of using light as a non-contact debugging tool to diagnose and correct electrical weights. Our optoelectronic neuromorphic computing platform has shown that memristive implementations of neural networks can advance beyond simple pattern matching to complex cognitive tasks. Since this work is mainly focused speech recognition application, further exploration is required to examine the potential of such implementations in complex computer vision, natural language processing and other fields of computer intelligence.

In our third work, we have developed a hybrid CNN-RNN model that produces state of the art results for four-class respiratory cycle classification on ICBHI'17 respiratory audio dataset. We also propose a patient screening

and transfer learning strategy to identify unhealthy patients and then build patient specific models through transfer learning. The performance of this strategy was limited by the lack of availability of large amount of patient specific data. In future, this strategy can be explored further by collecting more patient data. Now, for this dataset, the beginning and end of each breathing cycle was properly annotated. But for real world applications, this will not be the case. Therefore, automated audio segmentation algorithms need to be explored for segmenting breathing cycles in respiratory sound recordings. Finally, while we use weight quantization techniques to reduce the memory footprint of the proposed hybrid CNN-RNN, use of neuromorphic audio sensors and other low resolution deep networks or deep SNNs can be tested for this application for more energy efficient solutions.

In our final work we explore the potential of spiking neural networks and bio-realistic features for wider biomedical applications. Although this work shows that SNNs can achieve close to ANN accuracy at a fraction of computational cost for audio based cardiac abnormality detection, there are significant areas for further exploration related to this work. Since we have already shown the effectiveness of bio-realistic audio feature extraction for this application, in future similar SNN implementations can be integrated with dynamic audio sensors that produce similar feature representations to design much more power efficient end-to-end wearable biomedical solutions. While this work deals in heart sound classification, there are several other potential areas of audio based biomedical applications (such as works described in chapter 4) where similar strategies can be implemented. Furthermore, different spike encoding techniques and other ANN-SNN conversion methods should be explored for similar applications. Finally, while we do compute computational complexity of the proposed models, we could not arrive at exact power calculations due to unavailability of existing benchmarks. This requires further investigation.

Appendix A

Appendix: SNNRPN

A.1 Introduction

Asynchronous dynamic vision sensors are bio-inspired visual sensors that produce spikes corresponding to each pixel in their visual fields (also termed address-event representation or AER) where there is a change in light intensity [66]. These sensors received significant attention from research community in recent years due to their distinct advantages over traditional frame based video cameras in terms of both power efficiency and memory requirement. Several hardware implementations of these AER sensors have been made in the past decade [43] [44]. A number of new event based algorithms have been proposed in recent years to successfully process the data from these sensors [45]. These algorithms are applied in various applications ranging from motion estimation [46] and stereo-vision [7] to motor control [47] and gesture recognition [48].

However, most of these event based algorithms are inspired by traditional computer vision algorithms and therefore, not particularly suitable for neuromorphic processing [68]. Biologically plausible spiking neural networks (SNN) have been shown to perform successfully in complex tasks like image classification [6] and stereo vision [68]. Due to their unique asynchronous spike based data processing architecture, SNNs are inherently suitable for spiking input data.

With increasing demand in autonomous vehicles, smart surveillance and human-computer interaction etc, accurate real time object tracking has be-

come a primary research area in computer vision community [96]. With the advent of CNN and deep learning, a number of deep learning based object tracking algorithms have been proposed [97] [98]. Most of these object tracking algorithms have two distinct phases: *a)* region proposal and *b)* object classification. While the region proposal network proposes multiple bounding boxes per frame where there might be an object, the object classification network runs on the proposed regions and predicts the class of the object. Recent object tracking algorithms have used selective search [99], CNN based region proposal networks [100] etc. for generating region proposals.

With the development of several low-power SNN processors ([111], [264]), it is timely to revisit signal processing algorithms and recast them in terms of SNN building blocks. In this work, we propose a SNN based region proposal network (RPN) – the first stage for most tracking algorithms [100] and apply it to real recordings using an event based neuromorphic vision sensor (NVS) [44]. While the benefit of NVS in foreground extraction for stationary cameras is well known, it has not been properly quantified to the best of our knowledge. We propose the first SNN based RPN as well as use standard tracking metrics of precision vs recall to evaluate the RPN operating on the NVS recording of traffic data.

A.2 Materials and Methods

A.2.1 Data Collection

AER based event data is acquired using a DAVIS sensor (resolution - 240×180) setup at a traffic junction. This setup captures the movement of various moving entities in the scene and the typical objects in the scene include humans, bikes, cars, vans, trucks and buses. Multiple recordings of varying duration are obtained at different distances and day/night settings and the comprehensive details of the recordings used in this work are presented in

Table A.1: Details of the Dataset

Dataset Details				
Distance (m)	Lighting Con- dition	Duration (s)	Average Car Size	Number of Events
50	Day	58.9898	40x20	927242
50	Night	59.9599	38x18	771646
100	Day	60.0291	28x14	630885
100	Night	59.9599	27x14	480272
150	Day	58.9897	19x11	583646
150	Night	59.9593	19x11	479242

Table . The sizes of various moving objects vary by an order of magnitude in any given scene (eg: Humans vs Buses) and their velocities also span over a wide range (sub-pixel for humans to 5-6 pixels/frame for other fast moving vehicles) in the same recording. For reference purpose, typical size of a particular class of object (car in this case) across various recordings is provided in the fourth column in Table A.1. These recordings were manually annotated to generate the ground truth annotations of these objects in the scene.

A.2.2 Proposed Architecture

The basic building blocks of our proposed SNN are leaky integrate and fire (LIF) neurons and synapses. The membrane potential ($V(t)$) is governed by the following differential equation:

$$\tau_m \frac{dV}{dt} = -(V(t) - V_{rest}) + RI(t) \quad (\text{A.1})$$

where V_{rest} is the rest potential, $I(t)$ is the total synaptic current, R is the membrane resistance and τ_m is the membrane time constant. When the membrane potential reaches threshold voltage (V_{th}), the neuron fires (produces one output spike) and then resets to reset voltage (V_{reset}). After one

spike, the neuron can not spike again within a refractory period ($t_{refractory}$). The synapses are modelled using exponentially decaying EPSCs i.e., when a spike arrives, the conductance (g) of the synapse increases instantaneously and decays otherwise according to:

$$\tau_g \frac{dg}{dt} = -g \quad (\text{A.2})$$

where τ_g is the synaptic time constant. All the neurons and synapses in a layer have the same neuron and synaptic parameters.

Our proposed architecture is a three layered network with first two asynchronous event driven layers and one final clustering layer that converts the event based outputs of previous layer to frame based outputs for visualization and evaluation. The architecture is as follows:

A.2.2.1 Refractory Layer

The size of the refractory layer is same as the input image ($H \times L$) and each input is connected to one neuron in the refractory layer in 1:1 connections. The neurons in this layer have large refractory period and small threshold voltage. The importance of this layer is two-fold. Firstly, the output of DVS sensors often contains significant amount of noise [265] and the poor SNR affects the effectiveness of further processing of the events. With proper tuning of the refractory period, a sizeable fraction of these noisy events are eliminated without any considerable loss of signal and thereby, SNR is improved. As a result, we get significantly smoother tracking boxes in further layers. Secondly, as this layer filters off a large fraction of the input events, the computational complexities of the further event based layers are considerably reduced.

A.2.2.2 Convolution Layer

The convolution layer operates on the spikes produced by the refractory layer. Each neuron in the convolution layer is connected to a sliding square window of size $W \times W$ and stride S . Each convolutional neuron spikes in response to increased activity within its corresponding window and produces one region proposal box of size $W \times W$. These neurons have no refractory period (since the input to this layer is already sparse enough) and have relatively larger threshold to handle larger synaptic currents coming from multiple input neurons.

A.2.2.3 Local Excitation

In a variant of the model, we have also proposed recurrent lateral connections in the convolution layer where each neuron is connected to its immediate neighbors through excitatory connections.

A.2.2.4 Clustering Layer

This is the only frame based layer of our proposed architecture. In this layer, all the region proposal boxes generated by the convolution layer within a given frame duration are accumulated and all the neighboring boxes are clustered together to form larger boxes. Since the convolution layer only produces fixed size region proposal boxes, this layer is necessary to combine the boxes to actual shapes of objects.

The proposed algorithm is summarized in algorithm 4. Figure A.1 shows a sample frame of the input data and corresponding output frame.

A.2.3 Computational Complexity & Memory

For variable updates during each event, we have used Euler method [266] for calculating event based parameter updates. For the refractory layer, for

Algorithm 4 SNN based RPN

```

1: for each input event do
2:   modify refractory layer membrane voltages
3:   feed-forward any refractory spike produced
4: end for
5: for each refractory spike do
6:   modify refractory synaptic currents
7:   modify convolution layer membrane voltages
8:   modify recurrent synaptic currents (optional)
9:   feed-forward any region proposal produced
10: end for
11: for each frame do
12:   accumulate all region proposal boxes
13:   for each region proposal box do
14:     if There is an adjacent region proposal box then
15:       combine both boxes to produce larger region proposals
16:     end if
17:   end for
18: end for

```

Figure A.1: Visualization of RPN input and output: input frame shows a scene with one car and two humans (a) and the corresponding output frame shows the region proposals in red (b). The denoising in the output frame is done by the refractory layer while the region proposal is done by convolution layer and clustering layer.

each neuron, we need to store the membrane potential and the timestamp when the neuron received the last input spike. When a new spike comes at a input neuron, 5 operations are required to update the corresponding refractory membrane potential. So, if a b bit number is used to store each variable, refractory layer requires $2 \times H \times L \times b$ bit memory and 5 operation per event.

Similarly, in the convolution layer, $5 \times W \times W$ operations are required

to update each synaptic conductance corresponding to one convolution layer neuron, $W \times W - 1$ additional operations to compute the total input current and 5 operations to update the membrane voltage. So, in total, $6 \times W \times W + 4$ operations are required to update each convolution layer neuron. Now, when there is a refractory spike, the membrane voltage of the convolutional neurons connected to that neuron will be updated. Since we are using a square window with $< 50\%$ overlap, one refractory neuron is connected to at most 4 convolution neurons. So, a total of maximum $24 \times W \times W + 16$ operations are required per refractory spike. Now, as for memory requirement of this layer, $2 \times H \times L \times b$ bits are required to store the synaptic conductances and corresponding last spike times. An additional $2 \times M \times N \times b$ bits are required to store membrane potentials and corresponding last spike times for a convolution layer of dimension $M \times N$.

Finally, For the clustering layer, for each frame, all the region proposal boxes are iterated over and the neighboring boxes are combined. So, for r region proposals, $r \times (r - 1)$ operations are required to compare box locations. The clustering layer will require a buffer of size $2 \times r \times b$ bits (2 numbers representing each proposed box) to accumulate the region proposals generated during a frame.

So, for total input spikes K_{inp} , total refractory layer spikes $\alpha \times K_{inp}$ ($\alpha \ll 1$ since refractory layer produces significantly less spikes at output) and total frames F , the total number of computes for tracking a recording without lateral excitation is given by:

$$C_{total} = K_{inp} \times 5 + \alpha \times K_{inp} \times (24 \times W \times W + 16) + F \times r \times (r - 1) \quad (\text{A.3})$$

The total memory required in bits is given by:

$$B_{total} = 2 \times H \times L \times b + 2 \times H \times L \times b + 2 \times M \times N \times b + 2 \times r \times b \quad (\text{A.4})$$

A.2.4 Evaluation Metrics

To evaluate our region proposal network, we have adopted a precision vs recall curve based metric, traditionally used by computer vision community. The metric used to quantify the quality of proposed boxes is Intersection over Union ratio (IoU) defined as:

$$IoU = \frac{\text{intersection of proposed \& ground truth box}}{\text{union of proposed \& ground truth box}}$$

A certain threshold is defined based on IoU (e.g. IoU 0.5). Proposal boxes with IoU values larger than that threshold value are considered correct region detection (true positive box). Then, the performance of the tracker is evaluated on precision (true positive boxes/total proposal boxes) and recall (true positive boxes/total ground truth boxes) calculated over all the frames of the video. Parameter variation of the architecture produces different precision and recall values and therefore, the precision vs recall curve represent the performance of the region proposal algorithm in its entirety.

Although, IoU is perfectly suitable to evaluate region proposals for end to end object tracking, in case of standalone region proposal networks like the one we described, having proposal boxes larger than ground truth boxes is more advantageous than having proposal boxes smaller than ground truth boxes since larger boxes will ensure no loss of information to the object classifier and the classifier can be trained to tighten the proposal box if required. Since IoU is symmetric with respect to both ground truth and proposed boxes, it does not capture this distinction. So, we proposed another

metric, fitness score, to evaluate the fitness of region proposal.

$$\text{Fitness Score} = \frac{\text{intersection of proposed \& ground truth box}}{\text{area of ground truth box}}$$

A.3 Results and Discussions

For our first experiment, we have obtained the precision-recall curves for all 6 recordings in our dataset by varying the spiking threshold of convolution layer neurons (Fig. A.2). All the curves are plotted for a fixed IoU 0.3. Lower thresholds result in lower precision and higher recall while higher thresholds result in higher precision and lower recall. The recall value is higher for both 50 m day and night. This is the result of a smaller sensor-object distance. 150 m night curve shows significantly smaller precision due to unfavourable light condition (night) and much larger sensor-object distance.

Figure A.2: Precision-recall curve for six recordings: sensor-object distance shows significant impact on the recall value.

Now, in our proposed algorithm, the clustering is done entirely on the convolution layer proposals instead of original pixels. Although this ensures a significant saving in computational complexity, the resolution of the final boxes are limited by the dimension of convolution layer boxes. To explore

the performance-complexity trade-off, we varied the IoU for fixed threshold value with different window sizes. Fig. A.3 shows the precision and recall as a function of IoU for two different recordings with two different window sizes. Smaller window size results in a overall higher recall and better precision at higher IoU values. This goes to show that smaller window sizes produce more accurate region proposals due to availability of better resolution at the cost of higher computational complexity.

Figure A.3: IoU curve: smaller window size results in more accurate region proposals as evident from higher precision and recall for higher IoU values.

To examine the effect of lateral excitatory connections, we plotted the precision and recall curves for the same recording (100 m day) with and without lateral connections. We have seen in fig. A.3 that IoU is not particularly suitable for measuring the performance of our architecture for higher IoU values due to resolution limitation of the algorithm. So, we evaluated the curves here using both IoU and fitness score (FS) parameters (Fig. A.4). While for IoU, lateral connections does not seem to yield any improvement, for FS, lateral connections show performance improvement over base architecture. While the recall remains similar with or without lateral connections, precision of the RPN shows improvement for lateral excitation case. Quantifying the effect of recurrent lateral connections, both excitatory and

inhibitory, will require further investigation.

Figure A.4: Lateral excitation: precision and recall curve for 100 m (day) measured using IoU and fitness score (FS). Lateral excitation shows better precision at higher overlap ratios for FS measurement. For overlap ratio 0.8, lateral excitation improves precision by 2% without loss of recall (marked by arrow).

To benchmark our algorithm, we have compared it with the event based mean shift algorithm described in Delbruck et. al (2013) [47]. For a fair comparison, we applied the mean shift algorithm on the de-noised data at the output of the refractory layer. Fig. A.5 shows the precision-recall curves (measured using both IoU and fitness score) for both algorithms on 100 m (day) recording. While our algorithm shows clear advantage both in terms of precision and recall for IoU based measurement, for fitness score based measurement, mean shift achieves slightly higher precision at the cost of significantly reduced recall value. Moreover, our proposed algorithm generates a much stable ROC curve which signifies lesser dependence on fine tuning the threshold parameter.

Finally, based on the formula developed in section A.2.3, we calculate the total number of operations and memory requirements for the algorithm. For the recording in our dataset the value of α ranges from 0.1 – 0.2 and

Figure A.5: Comparison with event based mean shift algorithm: precision-recall curve for 100 m (day) measured using IoU and fitness score. SNN-RPN outperforms mean shift for IoU based measurements while mean shift obtains slightly higher precision for fitness score based measurement at significantly smaller recall.

mean value of r ranges from 4 – 6. So, assuming $\alpha = 0.15$ and $r = 5$, for sensor input dimension 180×240 , convolution layer size 15×20 , window size 16×16 and a frame rate of 30 FPS the architecture requires 0.9 Kops/event. For reference, if the convolution and clustering layer is replaced by mean shift tracker with similar configurations, it will require approximately 1.2 Kops/event. Now, if the variables are saved as 8-bit numbers, i.e. $b = 8$, the total memory required is given by 1.3 Mbits. It can be clearly seen from equation A.4 that the total memory requirement is dominated by the memory required by noise filtering layer (refractory layer). For the mean shift tracker, the required memory will be approximately 1.05 Mbits. So, the proposed algorithm performs significantly better than event based mean shift for similar memory requirement and computational cost.

A.4 Conclusion

In this work, we have proposed a three-layer SNN based region proposal network for event based processing of neuromorphic vision sensor recordings of traffic scenes. We have also introduced evaluation metrics for the region proposal network analogous to traditional computer vision techniques. The proposed algorithm is tested for different sensor-object distance and light conditions (day/night). The precision-recall trade-off is parameterized by neuron firing threshold. The resolution of the proposed boxes and thereby their accuracy is dependent on convolution window size and therefore, there is also an apparent computation-performance trade-off. Although this work is limited to only region proposals, this work can be extended in future to include a classification layer to evaluate its performance more accurately. Moreover, by combining a SNN based classifier similar to Diehl et. al. [6] with the proposed architecture an end to end SNN based object detection framework can be designed.

Appendix B

Appendix: EBBIOT

B.1 Introduction

Internet of Things (IoT) is a rapidly growing phenomenon where millions of connected sensors are distributed to improve a variety of applications ranging from precision agriculture to smart factories. Among the sensors, cameras offer unique opportunities due to the wealth of information they provide [267] at the cost of hugely increased bandwidth and energy to wirelessly transmit the huge volume of video data. The unique challenges and opportunities offered by camera sensors has led to a sub-field of IoT called the internet of video things (IoVT). Edge computing becomes important in this case to process data locally to reduce wireless transmission [204]. Neuromorphic sensors and processors offer an unique low power solution for this case.

In the past, neuromorphic vision sensors (NVS) have been employed for a variety of applications and tasks including microsphere tracking, multiple person tracking, vehicle-speed estimation, controlling robotic-arm, gesture recognition, etc [101] [102] [268]. While the role of these sensors in IoVT have been envisioned [204], there has not been any concrete work showing details of resource (energy, area) required by NVS based solutions for IoVT.

Object tracking forms the essential first step in most computer vision applications. Research work on tracking using NVS has mostly been focused on taking advantage of the high temporal resolution to faithfully track high speed objects which is a problem for frame based cameras [101] [102].

Mean shift [102], combination of CNN and particle filtering [107] and

Kalman Filters [108] have been employed in the past for tracking NVS outputs. While such applications demonstrate the ability of NVS based systems to handle complex tasks, they do not show their applicability to resource constrained systems which is a hallmark of IoT.

In this work, we propose EBBIOT—a low-complexity tracking algorithm for surveillance applications in IoVT using a NVS. The focus of our approach is to make the whole system less memory intensive (thus reducing chip area) and less computationally complex leading to savings in energy. Different from purely event based or purely frame based approaches, we accumulate the events from a NVS into a binary image and perform tracking on these frames. We further propose an event density based region proposal network (RPN) that has far less computations compared to traditional RPN. Lastly, we demonstrate a simple overlap based tracker (OT) that requires far less resources than the conventional event based mean shift (EBMS) [102] while producing superior performance compared to it. This is of immense importance when using NVS for IoVT applications like remote surveillance where long battery life of the sensor node is critical. We describe the details of our approach in the following sections.

B.2 Materials and Methods

NVS differ from a traditional frame based sensor in that they output an event $[e_i = (x_i, y_i, t_i, p_i)]$ whenever there is a positive or negative change in the intensity of light falling on that pixel. Here, e_i represents the event, (x_i, y_i) represent the event's location on the sensor array, t_i represents the time stamp of the event generally at a microsecond resolution and $p_i = 1$ (ON Event) whenever the change in intensity is increases beyond a threshold and $p_i = -1$ (OFF Event) whenever the change in intensity decreases below a threshold [269]. We define a list of symbols that we will use throughout

Figure B.1: Flowchart depicting all the important blocks in the system: binary frame generation, region proposal and overlap based tracking.

this work:

- $A \times B$ Image resolution
- B_t Number of bits to store time stamp t_i
- N_T Number of trackers
- t_F Frame duration
- p Neighbourhood size for noise filtering

The sensor used in this work is DAVIS [269] with $A = 240$ and $B = 180$. Also, we use $t_F = 66$ ms which is sufficient for tracking vehicles—a longer exposure is needed for humans. A flowchart depicting the entire algorithm pipeline is shown in Figure B.1 and is described in details in the following subsections.

B.2.1 Event based Binary Image and Noise Filtering

While most work on NVS has focussed on its event driven nature where number of computations are proportional to event rate, noise prevalent in

Figure B.2: Timing diagram showing interrupt driven operation of the NVS for duty cycled low power operation.

such sensors invariably lead to spurious spikes even in the absence of any objects in the scene [270]. This is a problem for IoT nodes which rely on saving energy by heavy duty cycling—using the NVS events as interrupts would rarely allow the processor to sleep.

Instead, we propose to use an interrupt based sensing scheme where the EBBIOT processor generates an interrupt at regular time intervals t_F to collect the events accumulated since the last interrupt (Figure B.2). Such a scheme makes it possible to interface NVS with others FPGA and micro-processors commonly used in IoT. This scheme is feasible for two reasons:

- Frame rates (≈ 15 Hz) are good enough for traffic surveillance as shown later in the chapter. This scheme loses appeal as t_F becomes smaller.
- We exploit the fact that the pixels firing an event are not reset till the event is readout in an NVS. Thus it can effectively store a binary image of events occurring while the processor is sleeping. In other words, we reuse the sensor as a memory.

Since we readout a binary image with only one possible event per pixel (ignoring polarity), we call the image an event based binary image (EBBI). Note that the NVS is always awake in this scenario and it is the processor which goes to sleep and wakes up regularly. However, this binary image is useful only if sufficient information can be extracted from it—we show corresponding results in later sections. For a binary frame, noise removal may be easily done by a median filter [271] (with patch size $p \times p$) since

spurious events result in salt and pepper noise. In this work, we used $p = 3$. The total number of computes per pixel of the filtered image is then equal to incrementing a counter every time a 1 is encountered in the p^2 pixels of that patch followed by a comparison with $\lfloor p^2/2 \rfloor$. This has to be added with the memory writes for creating the EBBI (ignoring memory reads due to lower energy requirement). The total memory requirement is twice the frame size—one frame to store the original image and one for the filtered version. We chose to keep the original frame since it might carry more information necessary for classification at a later stage. Thus we can summarize the computation C_{EBBI} and memory M_{EBBI} required by the proposed method as:

$$\begin{aligned} C_{EBBI} &\approx (\alpha p^2 + 2) \times A \times B (\because p \ll A, B) \\ M_{EBBI} &= 2 \times A \times B \end{aligned} \tag{B.1}$$

where α denotes the fraction of active pixels in a patch on average.

It may be argued that the NVS may be operated in a traditional mode while a noise filter block can generate an interrupt for the processor after filtering noise events. We show next that our approach is more memory efficient than this one. A commonly used filter for event-based output stream is a Nearest Neighbour filter (NN-filt) [270] that stores timestamps (using B_t bits per timestamp) for every incoming event and marks this event as a noisy event if it does not get temporal support in a $p \times p$ spatial neighbourhood. The number of computes per event is then equal to $p^2 - 1$ comparisons and counter increments followed by one B_t bit memory write to update the timestamp. The memory requirement for the two approaches may now be compared as:

$$\begin{aligned}
C_{NN-filt} &= (2(p^2 - 1) + B_t) \times \bar{n} \\
M_{NN-filt} &= B_t \times A \times B
\end{aligned} \tag{B.2}$$

where $\bar{n}$ is the average number of events per frame. Note that $\bar{n} = \beta \times \alpha \times A \times B$ ($\beta \geq 1$) where β represents average number of times an active pixel fires in t_F . Since the objects generally take up less than 10% of the image, we have a conservative estimate of $C_{EBBI} = 125.2$ kops/frame while $C_{NN-filt} \approx 276.4$ kops/frame. For the memory requirement, with typical values of $B_t = 16$, our proposed method provides $8X$ memory savings. For the DAVIS sensor used, the reduced memory requirement of our proposed EBBI is only 10.8 kB.

B.2.2 Region Proposal

A region proposal network (RPN) is the first block in a tracking pipeline [98]. In our application of traffic monitoring using a stationary camera, NVS offers the advantage of almost perfect foreground background separation inherently since the pixels only respond to changes in contrast [269]. Thus background pixels will generate little or no events while moving objects will generate a significantly larger number of events. This allows us to locate valid regions without having to perform costly CNN operations [272].

A traditional approach to detect regions in our case would have been to perform connected components analysis (CCA) on binary images using morphological operators [271]. While this is still less costly than CNN operations, we propose a further simplified approach in our case by exploiting the fact that our application only requires a side view of the ongoing traffic. Instead of doing CCA on a 2-D image, we create X and Y histograms (H_X and H_Y) for the image and find regions in these two 1-D signals by finding

Figure B.3: A sample EBBI with corresponding X and Y histogram based region proposals.

consecutive entries that are higher than a threshold (Figure B.3). The RPN and the following tracker both operate on these 1-D data structures reducing the computational burden. The actual 2D region is obtained by finding intersections of the X and Y regions (Figure B.3). Histogram based location of objects for NVS has been earlier proposed [273], but the authors only had one moving object without any realistic occlusion and did not implement a full tracker on these region proposals.

To further reduce the computation and memory requirement, we create the histograms from a scaled image, I^{s_1, s_2} downsampled from the original one (I) by factors s_1 and s_2 in X and Y directions respectively. Mathematically we can write the scaled image as:

$$I^{s_1, s_2}(i, j) = \sum_{n=0}^{s_2-1} \sum_{m=0}^{s_1-1} I(is_1 + m, js_2 + n) \quad i < \lfloor A/s_1 \rfloor, j < \lfloor B/s_2 \rfloor \quad (\text{B.3})$$

where $I(i, j) \in \{0, 1\}$. Based on this, the histograms are defined as:

$$\begin{aligned} H_X^{s_1}(i) &= \sum_j I^{s_1, s_2}(i, j) \\ H_Y^{s_2}(j) &= \sum_i I^{s_1, s_2}(i, j) \end{aligned} \quad (\text{B.4})$$

X and Y regions are then found from $H_X^{s_1}$ and $H_Y^{s_2}$ by finding contiguous

elements that are higher than a threshold (set to 1 in this case). This is acceptable since we need a coarse location for the objects which will be smoothed by the tracker. In fact, this helps in overcoming *fragmentation* of the object into smaller parts. As an example, the car in Fig. B.3 displays two peaks in H_X and would normally generate two separate regions. But in the low resolution histogram $H_X^{s_1}$, these mini regions get merged to create one region albeit with a slightly larger size than desired. One issue with this approach is that if there are multiple regions in both X and Y directions, false regions may be proposed by considering all overlaps between the two. In such cases, a check needs to be done in the original image to see if there are any valid pixels in that region. A better solution in that case is to perform the 2-D CCA, a task which we leave for future generalization of this approach. The total number of computes and memory requirement may now be summarized as follows:

$$\begin{aligned}
C_{RPN} &= A \times B + 2 \frac{A \times B}{s_1 s_2} \\
M_{RPN} &= \frac{A \times B}{s_1 s_2} [\log_2(s_1 s_2)] + \\
&\quad \left(\frac{A}{s_1} [\log_2(B \times s_1)] + \frac{B}{s_2} [\log_2(A \times s_2)] \right)
\end{aligned} \tag{B.5}$$

Here the first term for C_{RPN} (M_{RPN}) denotes the computes (memory) needed for I^{s_1, s_2} while the second term denotes the same for $H_X^{s_1}$ and $H_Y^{s_2}$. For our specific case, $s_1 = 6$ and $s_2 = 3$ were found to work well and in that case, $C_{RPN} = 45.6$ kop/frame while $M_{RPN} \approx 1.6$ kB. Both of the equations are dominated by the first term.

In comparison to this, even the simplest CNN-based object detector like YOLO [98] would need GPUs for real-time performance (30 fps) with RAM usage in the order of Gigabytes ($> 1GB$).

B.2.3 Overlap Based Tracker

In this chapter, we implement a multi-tracker system that can have up to $N_T = 8$ trackers simultaneously active on the frame. Two main assumptions have driven the design of this simplified tracker:

- t_F is small enough such that there is significant overlap between objects from one frame to next.
- Distractors such as trees which create spurious events can be removed by a manually provided definition of region of exclusion (ROE). Static occlusion from posts etc can also be included in ROE.

A major issue faced by trackers for EBBI is object *fragmentation* as shown in Fig. B.3 and discussed in Section B.2.2. This happens because big vehicles like a bus have a lot of plane surface on their sides that do not generate much events.

A block diagrammatic representation of the major steps in the algorithm are shown in Fig. B.1. The basic philosophy is to make predictions about the current position of a tracker from earlier frame and then correcting it based on current region proposals [109]. Denoting the position vectors (bottom left corner co-ordinates (x,y), width (w) and height (h) of tracker box) for region proposals and trackers by P_i and T_i respectively ($1 \leq i \leq 8$), we summarize the major steps below for every frame:

1. Get predicted position $T_i^{pred}(x, y)$ of all valid trackers by adding $T_i(x, y)$ with corresponding X and Y velocities.
2. Match T_i^{pred} for each valid tracker i with all available region proposals P_j . A match is found if overlapping area between the two is larger than a certain fraction of area of T_i^{pred} or P_j —hence the name overlap based tracker (OT).

3. If a P_j has no match and there are available free trackers, seed a new tracker k with $T_k = P_j$.
4. If a T_i^{pred} matches single or multiple P_j , assign all P_j to it and update T_i and velocities as a weighted average between prediction and region proposal. Here, past history of tracker is used to remove *fragmentation* in current region proposal if multiple P_j had matched.
5. If a P_j matched multiple T_i^{pred} , we can have two cases—dynamic occlusion between two moving objects or the earlier region proposals for this object were fragmented leading to multiple trackers being assigned to it. Occlusion is detected if the predicted trajectory of those trackers for upto $n = 2$ future time steps result in overlap. In that case, T_i is updated entirely based on T_i^{pred} and previous velocities are retained. Otherwise, the multiple T_i^{pred} are merged into one tracker based on P_j and corresponding velocity updated. The other trackers are freed up for future use.

The memory requirement for this tracker is negligible (< 0.5 kB) compared to the other modules and can be implemented in registers. The computation depends on which of the above cases is true. An average number of computes per frame can be obtained as follows:

$$C_{OT} = 134\overline{N_T}^2 + \gamma_3 N_3 + \gamma_4 N_4 + \gamma_5 N_5 \quad (\text{B.6})$$

where $\overline{N_T}$ denotes the average number of valid trackers, γ_j and N_j denote probability and number of computes for step j in the tracker description above. The first term dominates the others due to low values of γ_j . For the recordings used, $\overline{N_T} \approx 2$ resulting in $C_{OT} \approx 564$.

A Kalman Filter (KF) tracking algorithm based on [109] was used as comparison. The implementation follows a constant velocity motion model,

hence contains a state vector of length 2 ($X_{centroid}, Y_{centroid}$) for each track. Using [274] to approximate its computational complexity, Eq. B.7 shows the approximate computes for a Kalman filter with $\overline{N}_T = 2$ tracks.

$$C_{KF} = 4m^3 + 6m^2n + 4mn^2 + 4n^3 + 3n^2 \quad (\text{B.7})$$

Where n and m are the state and measurement vector size, respectively. Therefore, for this implementation with $n = 2 \times \overline{N}_T$ and $m = 2 \times \overline{N}_T$, $C_{KF} = 1200$.

Memory requirement of the KF is ≈ 1.1 kB which is also much smaller compared to the earlier blocks in the processing pipeline.

As an example of an event based tracker to be used in a fully event based pipeline after NN-filt, we chose [102]. For the event based mean shift (EBMS), average number of computes per frame (C_{EBMS}) and memory requirement in bits (M_{EBMS}) can be given as:

$$\begin{aligned} C_{EBMS} &= \overline{N}_F [9 \overline{CL}^2 + (169 + 16 \gamma_{merge}) \overline{CL} + 11] \\ M_{EBMS} &= 408 CL_{max} + 56 \end{aligned} \quad (\text{B.8})$$

Where $\overline{N}_F$ is the average number of events per frame at the output of NN-filter, $\overline{CL}$ is the average number of active clusters at any given time ($\approx \overline{N}_T$), γ_{merge} is the probability of two clusters merging and CL_{max} is the maximum number of potential clusters. For these calculations we have assumed that past 10 positions of a cluster is used to calculate the current velocity of the cluster using least square regression. For our experiments we have used $CL_{max} = 8$ and for our dataset, $\overline{CL} \approx 2$, $\gamma_{merge} \approx 0.1$ and $\overline{N}_F \approx 650$. So, EBMS requires 252 kops per frame which is $\approx 500X$ higher than EBBIOT. The total memory required for EBMS is 3.32 kB which is again negligible compared to the earlier processing blocks for noise filtering.

B.3 Results and Discussions

B.3.1 Datasets

Address Event Representation (AER) based event data is acquired from a DAVIS setup at a traffic junction. This scenario captures moving entities in the scene and the typical objects in the scene include humans, bikes, cars, vans, trucks and buses. Two sample recordings of varying duration are obtained at different lens settings and the comprehensive details of those two recordings used in the work are presented in table B.1. The sizes of various moving objects vary by an order of magnitude in any given scene and their velocities also range over a wide range (sub-pixel to 5-6 pixels/frame). These recordings were manually annotated to generate the Ground Truth tracker annotations of these objects in the scene.

Table B.1: Details of the Dataset

Location	Lens (mm)	Duration (s)	Num Events
ENG	12	2998.4	107.5M
LT4	6	999.5	12.5M

B.3.2 Tracker Evaluation Metrics

To assess the performance of the tracker, we examine how closely the tracks generated by the proposed tracker match with the ground truth tracks. The first step in this evaluation process involves obtaining the boxes encapsulating the objects in the scene from both Ground Truth and the proposed tracker annotations at multiple instances of time (with a fixed time interval) in the entire duration of the recording. For each instance, if the area of a ground truth box enclosing an object in the scene is $A_{GroundTruth}$, the area of a tracker box enclosing an object is $A_{ProposedTracker}$, the area of intersection of these two boxes is denoted as $A_{Intersection}$ and the area of the union of

these two boxes is denoted as A_{Union} . We adopted Intersection over Union (IoU) typically used in computer vision community (as defined in B.9) to measure the effectiveness of our tracker algorithm.

$$IoU = \frac{A_{Intersection}}{A_{Union}} \quad (B.9)$$

The proposed tracker box is assumed to be a correct region proposal only if the IoU of that box is greater than a threshold.

A certain parametric threshold is defined based on IoU (e.g. IoU 0.5) to determine if a proposed box is a correct region proposal. Proposal boxes with IoU values larger than that threshold value is considered correct region detection (true positive box). Then, the performance of the tracker is evaluated on precision (true positive boxes/total proposal boxes) and recall(true positive boxes/total ground truth boxes) calculated over all the frames of the video.

B.3.3 Performance and Resource Requirements

Finally, for a fair comparison of the EBBIOT algorithm with EBMS and Kalman Filter (KF) we compare the weighted average of precision and recall across multiple recordings where the weights correspond to the number of ground truth tracks present in a given recording. The results are shown in fig. B.4.

We also calculated total computes per frame and total memory required for KF and EBMS relative to EBBIOT (fig. B.5). For EBBIOT and KF total memory and computes are calculated considering memory and computes required for generating EBBI, RPN and tracker while for EBMS we consider memory and computes of NN-filt and EBMS tracker.

Figure B.4: Comparison of EBMS, KF and EBBIOT in terms of precision and recall for different IoU thresholds. EBBIOT outperforms others and shows more stable precision and recall values for varying thresholds.

Figure B.5: Comparison of EBMS and EBBI+KF with EBBIOT in terms of total computations per frame and memory requirement showing significantly less resource requirement for EBBIOT.

B.4 Conclusion

In this work, we presented EBBIOT – a novel paradigm for object tracking using stationary neuromorphic vision sensors in low-power sensor nodes for IoVT applications and demonstrated reliable tracking performance compared to both event-based tracking methodologies and traditional frame-based tracking frameworks. In particular, the mixed approach of creating event-based binary images resulted in $\approx 7X$ reduced memory requirement

and $3X$ less computes over conventional event-based approaches. On the other hand, region proposals based on event density resulted in $> 1000X$ less memory and computes compared to frame based approaches. Further, we have not tracked slow and small objects like humans—this can be done by a two time scale approach where a second frame is generated with longer exposure times to capture activity of humans.

Publications

Journal Papers

- (J1) Jyotibdha Acharya, Aakash Patil , Xiaoya Li, Yi Chen, Shih-Chii Liu and Arindam Basu, “A Comparison of Low-complexity Real-Time Feature Extraction for Neuromorphic Speech Recognition,” *Frontiers in neuroscience* 12 (2018):160.
- (J2) Jyotibdha Acharya and Arindam Basu, “Deep Neural Network for Respiratory Sound Classification Enabled by Transfer Learning,” *IEEE Transactions on Biomedical Circuits and Systems* (under review).
- (J3) Arindam Basu, Jyotibdha Acharya, Tanay Karnik, Huichu Liu, Member, Hai Li, Jae-sun Seo and Chang Song, “Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8.1 (2018): 6-27.
- (J4) Rohit Abraham John, Jyotibdha Acharya, Chao Zhu, Sumon Kumar Bose, Apoorva Chaturvedi, Abhijith Surendran, Keke K. Zhang, Xu Manzhong, Wei Lin Leong, Zheng Liu, Arindam Basu, and Nripan Mathews, “Optogenetics-Inspired Light-Driven Computational Circuits Enable In-Memory Computing for Deep Recurrent Neural Networks,” *Nature Communications* (under review).

Book Chapter

- (B1) Jyotibdha Acharya, and Arindam Basu, “Neuromorphic spiking neural network algorithms,” *Springer Handbook of Neuroengineering*, edited by Nitish V. Thakor (under review).

Conference Papers

- (C1) Jyotibdha Acharya, Arindam Basu, and Wee Ser, “” Feature extraction techniques for low-power ambulatory wheeze detection wearables,” *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE. IEEE, 2017.*
- (C2) Jyotibdha Acharya, Vandana Padala and Arindam Basu, “Spiking Neural Network Based Region Proposal Networks for Neuromorphic Vision Sensors,” *2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 2019, pp. 1-5.*
- (C3) Jyotibdha Acharya, Andres Ussa, Vandana Reddy Padala, Rishiraj Singh Sidhu, Garrick Orchard, Bharath Ramesh, Arindam Basu, “EBBIOT: A Low-complexity Tracking Algorithm for Surveillance in IoVT Using Stationary Neuromorphic Vision Sensors,” *32nd IEEE International System-On-Chip Conference, Singapore, 2019.*
- (C4) Andres Ussa, Luca Della Vedova, Vandana Reddy Padala, Deepak Singla, Jyotibdha Acharya, Charles Zhang Lei, Garrick Orchard, Arindam Basu, Bharath Ramesh, “A low-power end-to-end hybrid neuromorphic framework for surveillance applications,” *BMVC 2019 Workshop on Object Detection and Recognition for Security Screening.*
- (C5) Sumon Kumar Bose, Jyotibdha Acharya, and Arindam Basu, “Is my Neural Network Neuromorphic? Taxonomy, Recent Trends and Future Directions in Neuromorphic Engineering,” *Proceedings of the 2019 Asilomar Conference on Signals, Systems, and Computers.*

Bibliography

- [1] C. Mead, “Neuromorphic electronic systems,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.
- [2] J. M. Rabaey, “Human-centric computing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 3–11, 2019.
- [3] G. Indiveri and S. C. Liu, “Memory and information processing in neuromorphic systems,” *Proc. of IEEE*, vol. 103, no. 8, pp. 1379–97, 2015.
- [4] S. K. Bose, J. Acharya, and A. Basu, “Is my neural network neuromorphic? taxonomy, recent trends and future directions in neuromorphic engineering,” in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 1522–1527.
- [5] Bose, Sumon Kumar and Acharya, Jyotibdha and Basu, Arindam, “Survey of neuromorphic and machine learning accelerators in sovc, isscc and nature/science series of journals from 2017 onwards,” 2019.
- [6] P. U. Diehl and M. Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity,” *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.
- [7] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck, “Asynchronous event-based binocular stereo matching,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 347–353, 2012.

- [8] M. Yang, S. C. Liu, and T. Delbruck, "A Dynamic Vision Sensor With 1Temporal Contrast Sensitivity and In-Pixel Asynchronous Delta Modulator for Event Encoding," *IEEE Journal of Solid State Circuits*, vol. 50, no. 9, 2015.
- [9] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS," *IEEE Journal of Solid State Circuits*, vol. 46, no. 10, 2010.
- [10] M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, "A 0.5V 55 μ W 64 $\times$ 2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing," *IEEE Journal of Solid State Circuits*, vol. 51, no. 11, 2016.
- [11] M. Rasouli, Y. Chen, A. Basu, N. V. Thakor, and S. L. Kukreja, "Spike-based tactile pattern recognition using an extreme learning machine," in *IEEE Biomedical Circuits and Systems (BioCAS)*, 2015.
- [12] J. C. et. al., "A pencil balancing robot using a pair of AER dynamic vision sensors," in *IEEE Intl. Symp. Circuits and Systems (ISCAS)*, 2009.
- [13] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, no. 223, 2013.
- [14] W. W. lee, S. L. Kukreja, and N. V. Thakor, "Discrimination of Dynamic Tactile Contact by Temporally Precise Event Sensing in Spiking Neuromorphic Networks," *Frontiers in Neuroscience*, vol. 11, no. 5, 2017.
- [15] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous Event-Based Binocular Stereo Matching," *IEEE Trans. on Neural Networks*, vol. 23, no. 2, pp. 347 – 353, 2012.

- [16] G. B. Huang, Q. Y. Zhu, and C. K. Siew, “Extreme Learning Machines: Theory and Applications,” *Neurocomputing*, vol. 70, pp. 489–501, 2006.
- [17] Y. H. Pao, G. H. Park, and D. Sobajic, “Learning and generalization characteristics of the random vector functional-link net,” *Neurocomputing*, vol. 6, pp. 163–180, 1994.
- [18] C. Eliasmith and C. H. Andersen, *Neural Engineering: Computation, Representation and Dynamics in Neurobiological Systems*. MIT Press, 2003.
- [19] W. Maass, T. Natschläger, and H. Markram, “Real-time computing without stable states: a new framework for neural computation based on perturbations,” *Neural Computation*, vol. 14, no. 11, pp. 2531–60, 2002.
- [20] A. Basu, S. Shuo, H. Zhou, G. Huang, and M. Lim, “Silicon Spiking Neurons for Hardware Implementation of Extreme Learning Machines,” *Neurocomputing*, vol. 102, pp. 125–134, 2013.
- [21] E. Yao and A. Basu, “VLSI Extreme Learning Machine: A Design Space Exploration,” *IEEE Trans. on VLSI*, vol. 25, no. 1, pp. 60–74, 2017.
- [22] C. S. Thakur and R. W. et. al, “A Low Power Trainable Neuromorphic Integrated Circuit That Is Tolerant to Device Mismatch,” *IEEE Trans. on CAS-I*, vol. 63, no. 2, pp. 211–221, 2016.
- [23] A. Patil, S. Shen, E. Yao, and A. Basu, “Hardware Architecture for Large Parallel Array of Random Feature Extractors applied to Image Recognition,” *Neurocomputing*, vol. 261, pp. 193–203, 2017.

- [24] P. Hasler, C. Diorio, B. Minch, and C. Mead, “Single Transistor Learning Synapses,” in *Neural Information Processing Systems (NIPS)*, 1994, pp. 817–824.
- [25] C. Diorio, P. Hasler, B. A. Minch, and C. Mead, “A floating-gate MOS learning array with locally computed weight updates,” *IEEE Trans. on Electron Devices*, vol. 44, no. 12, 1997.
- [26] C. Diorio, D. Hsu, and M. Figueroa, “Adaptive CMOS: From Biological Inspiration to Systems-on-a-Chip,” *Proceedings of IEEE*, vol. 90, no. 3, 2002.
- [27] J. Hasler and B. Marr, “Finding a roadmap to achieve large neuro-morphic hardware systems,” *Frontiers in Neuroscience*, vol. 7, p. 118, 2013.
- [28] H. Li and Y. Chen, *Nonvolatile memory design: magnetic, resistive, and phase change*. CRC Press, 2011.
- [29] Y. Chen, X. Wang, H. Li, H. Xi, Y. Yan, and W. Zhu, “Design margin exploration of spin-transfer torque ram (stt-ram) in scaled technologies,” *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 12, pp. 1724–1734, 2010.
- [30] Y. Chen, W. Tian, H. Li, X. Wang, and W. Zhu, “Pcmo device with high switching stability,” *IEEE Electron Device Letters*, vol. 31, no. 8, pp. 866–868, 2010.
- [31] I. Bayram, E. Eken, X. Wang, X. Sun, T. Ma, and Y. Chen, “Adaptive refreshing and read voltage control scheme for fedram,” in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1154–1157.

- [32] N. Locatelli, A. Mizrahi, A. Accioly, D. Querlioz, J.-V. Kim, V. Cros, and J. Grollier, “Spin torque nanodevices for bio-inspired computing,” in *Cellular Nanoscale Networks and their Applications (CNNA), 2014 14th International Workshop on*. IEEE, 2014, pp. 1–2.
- [33] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola *et al.*, “Neuromorphic computing using non-volatile memory,” *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.
- [34] Y. Chen, X. Wang, H. Li, H. Liu, and D. V. Dimitrov, “Design margin exploration of spin-torque transfer ram (spram),” in *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*. IEEE, 2008, pp. 684–690.
- [35] G. Narasimman, S. Roy, X. Fong, K. Roy, C.-H. Chang, and A. Basu, “A low-voltage, low power stdp synapse implementation using domain-wall magnets for spiking neural networks,” in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 914–917.
- [36] A. Sengupta and K. Roy, “A Vision for All-Spin Neural Networks: A Device to System Perspective,” *IEEE Trans. on CAS-I*, vol. 63, no. 12, pp. 2267–77, 2016.
- [37] A. Sengupta, Z. A. Azim, X. Fong, and K. Roy, “Spin-orbit torque induced spike-timing dependent plasticity,” *Applied Physics Letters*, vol. 106, no. 093704, 2015.
- [38] A. Sengupta and K. Roy, “Encoding neural and synaptic functionalities in electron spin: A pathway to efficient neuromorphic computing,” *Applied Physics Reviews*, vol. 4, no. 041105, 2017.

- [39] E. J. Fuller, F. E. Gabaly, F. Léonard, S. Agarwal, S. J. Plimpton, R. B. Jacobs-Gedrim, C. D. James, M. J. Marinella, and A. A. Talin, “Li-ion synaptic transistor for low power analog computing,” *Advanced Materials*, vol. 29, no. 4, p. 1604310, 2017.
- [40] Y. van de Burgt, E. Lubberman, E. J. Fuller, S. T. Keene, G. C. Faria, S. Agarwal, M. J. Marinella, A. A. Talin, and A. Salleo, “A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing,” *Nature materials*, vol. 16, no. 4, p. 414, 2017.
- [41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [42] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [43] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, “Retinomorphic event-based vision sensors: bioinspired cameras with spiking output,” *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014.
- [44] R. Berner, C. Brandli, M. Yang, S.-C. Liu, and T. Delbruck, “A 240×180 10mw 12us latency sparse-output vision sensor for mobile applications,” in *VLSI Circuits (VLSIC), 2013 Symposium on*. IEEE, 2013, pp. C186–C187.
- [45] A. Basu, J. Acharya, T. Karnik, H. Liu, H. Li, J.-S. Seo, and C. Song, “Low-power, adaptive neuromorphic systems: Recent progress and future directions,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 6–27, 2018.

- [46] 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, 2015.
- [47] T. Delbruck and M. Lang, “Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor,” *Frontiers in neuroscience*, vol. 7, p. 223, 2013.
- [48] A. Amir, B. Taba, D. J. Berg, T. Melano, J. L. McKinstry, C. Di Nolfo, T. K. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, “A low power, fully event-based gesture recognition system.” in *CVPR*, 2017, pp. 7388–7397.
- [49] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [50] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-10 and cifar-100 datasets,” *URL: <https://www.cs.toronto.edu/kriz/cifar.html>*, vol. 6, 2009.
- [51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [52] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

- [53] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, “A survey on deep learning in medical image analysis,” *Medical image analysis*, vol. 42, pp. 60–88, 2017.
- [54] H. Nakano, T. Furukawa, and T. Tanigawa, “Tracheal sound analysis using a deep neural network to detect sleep apnea,” *Journal of Clinical Sleep Medicine*, vol. 15, no. 08, pp. 1125–1133, 2019.
- [55] J. Amoh and K. Odame, “Deep neural networks for identifying cough sounds,” *IEEE transactions on biomedical circuits and systems*, vol. 10, no. 5, pp. 1003–1011, 2016.
- [56] H. Ryu, J. Park, and H. Shin, “Classification of heart sound recordings using convolution neural network,” in *2016 Computing in Cardiology Conference (CinC)*. IEEE, 2016, pp. 1153–1156.
- [57] M. Pfeiffer and T. Pfeil, “Deep learning with spiking neurons: opportunities and challenges,” *Frontiers in neuroscience*, vol. 12, 2018.
- [58] W. Maass, “Networks of spiking neurons: the third generation of neural network models,” *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [59] 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, 2015, pp. 1–8.
- [60] 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.

- [61] S. B. Shrestha and G. Orchard, “Slayer: Spike layer error reassignment in time,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421.
- [62] Y. Jin, W. Zhang, and P. Li, “Hybrid macro/micro level backpropagation for training deep spiking neural networks,” in *Advances in Neural Information Processing Systems*, 2018, pp. 7005–7015.
- [63] R. G. Leonard and G. Doddington, “Tidigits speech corpus,” *Texas Instruments, Inc*, 1993.
- [64] E. Yao and A. Basu, “VLSI Extreme Learning Machine: A design space exploration,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 60–74, 2017.
- [65] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, “Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals,” *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.
- [66] A. Yousefzadeh, G. Orchard, E. Stamatias, T. Serrano-Gotarredona, and B. Linares-Barranco, “Hybrid neural network, an efficient low-power digital hardware implementation of event-based artificial neural network,” in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.
- [67] S.-C. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, “Asynchronous binaural spatial audition sensor with $2 \times 64 \times 4$ channel output,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 4, pp. 453–464, 2014.

- [68] M. Osswald, S.-H. Ieng, R. Benosman, and G. Indiveri, “A spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems,” *Scientific reports*, vol. 7, p. 40703, 2017.
- [69] C. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley, Reading, MA, 1989.
- [70] V. Chan, S.-C. Liu, and A. van Schaik, “AER EAR: A matched silicon cochlea pair with address event representation interface,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.
- [71] S.-C. Liu and T. Delbruck, “Neuromorphic sensory systems,” *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.
- [72] C.-H. Li, T. Delbruck, and S.-C. Liu, “Real-time speaker identification using the AEREAR2 event-based silicon cochlea,” in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2012, pp. 1159–1162.
- [73] S. Chakrabartty and S.-C. Liu, “Exploiting spike-based dynamics in a silicon cochlea for speaker identification,” in *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2010, pp. 513–516.
- [74] M. Abdollahi and S.-C. Liu, “Speaker-independent isolated digit recognition using an aer silicon cochlea,” in *2011 IEEE Biomedical Circuits and Systems Conference (BIOCAS)*. IEEE, 2011, pp. 269–272.
- [75] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu, “Feature representations for neuromorphic audio spike streams,” *Frontiers of Neuroscience*, 2018.

- [76] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.
- [77] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, “A large-scale model of the functioning brain,” *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012.
- [78] T. C. Stewart, “A technical overview of the neural engineering framework,” 2012.
- [79] T. Stewart, F.-X. Choo, and C. Eliasmith, “Spaun: A perception-cognition-action model using spiking neurons,” in *Proceedings of the Cognitive Science Society*, vol. 34, no. 34, 2012.
- [80] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.
- [81] Y. Chen, E. Yao, and A. Basu, “A 128-channel extreme learning machine-based neural decoder for brain machine interfaces,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 3, pp. 679–692, 2016.
- [82] Y. Song, J. Crowcroft, and J. Zhang, “Automatic epileptic seizure detection in EEGs based on optimized sample entropy and extreme learning machine,” *Journal of neuroscience methods*, vol. 210, no. 2, pp. 132–146, 2012.
- [83] J. Deng, S. Fruhholz, Z. Zhang, and B. Schuller, “Recognizing emotions from whispered speech based on acoustic feature transfer learning,” *IEEE Access*, 2017.

- [84] A. Akusok, K.-M. Björk, Y. Miche, and A. Lendasse, “High-performance extreme learning machines: a complete toolbox for big data applications,” *IEEE Access*, vol. 3, pp. 1011–1025, 2015.
- [85] R. Leonard, “A database for speaker-independent digit recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '84.*, vol. 9, Mar 1984, pp. 328–331.
- [86] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr, and T. Delbrück, “Steering a predator robot using a mixed frame/event-driven convolutional neural network,” in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2016, pp. 1–8.
- [87] A. Patil, S. Shen, E. Yao, and A. Basu, “Random projection for spike sorting: Decoding neural signals the neural network way,” in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2015, pp. 1–4.
- [88] R. Penrose, “A generalized inverse for matrices,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, no. 03, pp. 406–413, 1955.
- [89] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
- [90] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2012.
- [91] S. Zhang, W. Tan, and Y. Li, “A survey of online sequential extreme learning machine,” in *2018 5th International Conference on Control,*

- Decision and Information Technologies (CoDIT)*. IEEE, 2018, pp. 45–50.
- [92] J.-M. Park and J.-H. Kim, “Online recurrent extreme learning machine and its application to time-series prediction,” in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1983–1990.
- [93] D. Neil and S.-C. Liu, “Effective sensor fusion with event-based sensors and deep network architectures,” in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 2282–2285.
- [94] W.-Y. Tsai, D. R. Barch, A. S. Cassidy, M. V. DeBole, A. Andreopoulos, B. L. Jackson, M. D. Flickner, J. V. Arthur, D. S. Modha, J. Sampson *et al.*, “Always-on speech recognition using truenorth, a reconfigurable, neurosynaptic processor,” *IEEE Transactions on Computers*, vol. 66, no. 6, pp. 996–1007, 2017.
- [95] C. Gao, D. Neil, E. Ceolini, S.-C. Liu, and T. Delbruck, “Deltarnn: A power-efficient rnn accelerator,” in *Twenty-Sixth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2018.
- [96] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.
- [97] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [98] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

- [99] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.
- [100] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in *Advances in neural information processing systems*, 2016, pp. 379–387.
- [101] Z. Ni, C. Pacoret, R. Benosman, S. Ieng, and S. RÉGNIER, “Asynchronous event-based high speed vision for microparticle tracking,” *Journal of Microscopy*, vol. 245, no. 3, pp. 236–244, 2012.
- [102] T. Delbruck and M. Lang, “Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor,” *Frontiers in Neuroscience*, vol. 7, p. 223, 2013.
- [103] P. Lichtsteiner and T. Delbruck, “A 64x64 aer logarithmic temporal derivative silicon retina,” in *Research in Microelectronics and Electronics, 2005 PhD*, vol. 2. IEEE, 2005, pp. 202–205.
- [104] F. Barranco, C. Fermüller, and Y. Aloimonos, “Contour motion estimation for asynchronous event-driven cameras,” *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1537–1556, 2014.
- [105] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, “Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS),” in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2016, pp. 1–7.
- [106] E. Mueggler, B. Huber, and D. Scaramuzza, “Event-based, 6-DOF pose tracking for high-speed maneuvers,” in *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 2761–2768.

- [107] H. Liu, D. P. Moeys, G. Das, D. Neil, S.-C. Liu, and T. Delbrück, “Combined frame-and event-based detection and tracking,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 2511–2514.
- [108] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, S.-H. Ieng, R. Benosman, and B. Linares-Barranco, “Event-driven stereo visual tracking algorithm to solve object occlusion,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4223–4237, 2018.
- [109] L. Lin, B. Ramesh, and C. Xiang, “Biologically Inspired Composite Vision System for Multiple Depth-of-field Vehicle Tracking and Speed Detection,” in *Asian Conference on Computer Vision*, 2015, pp. 473–486.
- [110] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.
- [111] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [112] E. S. Boyden, F. Zhang, E. Bamberg, G. Nagel, and K. Deisseroth, “Millisecond-timescale, genetically targeted optical control of neural activity,” *Nature neuroscience*, vol. 8, no. 9, p. 1263, 2005.
- [113] K. Deisseroth, “Optogenetics,” *Nature methods*, vol. 8, no. 1, p. 26, 2011.

- [114] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.
- [115] B. W. et. al, "Birth, Life, and Death in Microelectronic Systems," Office of Naval Research, Tech. Rep., May 1961.
- [116] B. Linares-Barranco, T. Serrano-Gotarredona, L. A. Camuñas-Mesa, J. A. Perez-Carrasco, C. Zamarreño-Ramos, and T. Masquelier, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," *Frontiers in neuroscience*, vol. 5, p. 26, 2011.
- [117] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.
- [118] J. Borghetti, Z. Li, J. Straznicky, X. Li, D. A. Ohlberg, W. Wu, D. R. Stewart, and R. S. Williams, "A hybrid nanomemristor/transistor logic circuit capable of self-programming," *Proceedings of the National Academy of Sciences*, vol. 106, no. 6, pp. 1699–1703, 2009.
- [119] T. Prodromakis, B. P. Peh, C. Papavassiliou, and C. Toumazou, "A versatile memristor model with nonlinear dopant kinetics," *IEEE transactions on electron devices*, vol. 58, no. 9, pp. 3099–3105, 2011.
- [120] S. H. Jo, K.-H. Kim, and W. Lu, "High-density crossbar arrays based on a si memristive system," *Nano letters*, vol. 9, no. 2, pp. 870–874, 2009.
- [121] T. Chang, Y. Yang, and W. Lu, "Building neuromorphic circuits with memristive devices," *IEEE Circuits and Systems Magazine*, vol. 13, no. 2, pp. 56–73, 2013.
- [122] Y. Kim, Y. Zhang, and P. Li, "A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive comput-

- ing,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 11, no. 4, p. 38, 2015.
- [123] R. Hasan and T. M. Taha, “Enabling back propagation training of memristor crossbar neuromorphic processors,” in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 21–28.
- [124] H.-Y. Chen, B. Gao, H. Li, R. Liu, P. Huang, Z. Chen, B. Chen, F. Zhang, L. Zhao, Z. Jiang *et al.*, “Towards high-speed, write-disturb tolerant 3d vertical rram arrays,” in *2014 Symposium on VLSI Technology (VLSI-Technology): Digest of Technical Papers*. IEEE, 2014, pp. 1–2.
- [125] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, “Overcoming the challenges of crossbar resistive memory architectures,” in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 476–488.
- [126] X. Zhang, A. Huang, Q. Hu, Z. Xiao, and P. K. Chu, “Neuromorphic computing with memristor crossbar,” *physica status solidi (a)*, vol. 215, no. 13, p. 1700875, 2018.
- [127] S. Agarwal, T.-T. Quach, O. Parekh, A. H. Hsia, E. P. DeBenedictis, C. D. James, M. J. Marinella, and J. B. Aimone, “Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding,” *Frontiers in neuroscience*, vol. 9, p. 484, 2016.
- [128] M. Prezioso, F. Merrih-Bayat, B. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, “Training and operation of an integrated

- neuromorphic network based on metal-oxide memristors,” *Nature*, vol. 521, no. 7550, p. 61, 2015.
- [129] J. Feldmann, N. Youngblood, C. Wright, H. Bhaskaran, and W. Pernice, “All-optical spiking neurosynaptic networks with self-learning capabilities,” *Nature*, vol. 569, no. 7755, p. 208, 2019.
- [130] Z. Cheng, C. Ríos, W. H. Pernice, C. D. Wright, and H. Bhaskaran, “On-chip photonic synapse,” *Science advances*, vol. 3, no. 9, p. e1700160, 2017.
- [131] M. Häusser, “Optogenetics: the age of light,” *Nature methods*, vol. 11, no. 10, p. 1012, 2014.
- [132] K. Honjo, R. Y. Hwang, and W. D. Tracey Jr, “Optogenetic manipulation of neural circuits and behavior in drosophila larvae,” *Nature protocols*, vol. 7, no. 8, p. 1470, 2012.
- [133] E. Burguière, P. Monteiro, G. Feng, and A. M. Graybiel, “Optogenetic stimulation of lateral orbitofronto-striatal pathway suppresses compulsive behaviors,” *Science*, vol. 340, no. 6137, pp. 1243–1246, 2013.
- [134] Q. H. Wang, K. Kalantar-Zadeh, A. Kis, J. N. Coleman, and M. S. Strano, “Electronics and optoelectronics of two-dimensional transition metal dichalcogenides,” *Nature nanotechnology*, vol. 7, no. 11, p. 699, 2012.
- [135] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti *et al.*, “Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element,” *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.

- [136] S. Kim, J. Hasler, and S. George, “Integrated floating-gate programming environment for system-level ics,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2244–2252, 2015.
- [137] W. Zhang, J.-K. Huang, C.-H. Chen, Y.-H. Chang, Y.-J. Cheng, and L.-J. Li, “High-gain phototransistors based on a cvd mos2 monolayer,” *Advanced materials*, vol. 25, no. 25, pp. 3456–3461, 2013.
- [138] V. Klee, E. Preciado, D. Barroso, A. E. Nguyen, C. Lee, K. J. Erickson, M. Triplett, B. Davis, I.-H. Lu, S. Bobek *et al.*, “Superlinear composition-dependent photocurrent in cvd-grown monolayer mos2 (1-x) se2 x alloy devices,” *Nano letters*, vol. 15, no. 4, pp. 2612–2619, 2015.
- [139] Q. Wang, Y. Wen, K. Cai, R. Cheng, L. Yin, Y. Zhang, J. Li, Z. Wang, F. Wang, F. Wang *et al.*, “Nonvolatile infrared memory in mos2/pbs van der waals heterostructures,” *Science advances*, vol. 4, no. 4, p. eaap7916, 2018.
- [140] S. Ghatak, A. N. Pal, and A. Ghosh, “Nature of electronic states in atomically thin mos2 field-effect transistors,” *Acs Nano*, vol. 5, no. 10, pp. 7707–7712, 2011.
- [141] A. J. Arnold, A. Razavieh, J. R. Nasr, D. S. Schulman, C. M. Eichfeld, and S. Das, “Mimicking neurotransmitter release in chemical synapses via hysteresis engineering in mos2 transistors,” *ACS nano*, vol. 11, no. 3, pp. 3110–3118, 2017.
- [142] R. A. John, F. Liu, N. A. Chien, M. R. Kulkarni, C. Zhu, Q. Fu, A. Basu, Z. Liu, and N. Mathews, “Synergistic gating of electro-ionophotocactive 2d chalcogenide neuristors: Coexistence of hebbian and

- homeostatic synaptic metaplasticity,” *Advanced Materials*, vol. 30, no. 25, p. 1800220, 2018.
- [143] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [144] A. Karparthy, “A peek at trends in machine learning,” 2017.
- [145] A. Basu and P. E. Hasler, “A fully integrated architecture for fast and accurate programming of floating gates over six decades of current,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 6, pp. 953–962, 2010.
- [146] F. M. Bayat, X. Guo, M. Klachko, N. Do, K. Likharev, and D. Strukov, “Model-based high-precision tuning of nor flash memory cells for analog computing applications,” in *2016 74th Annual Device Research Conference (DRC)*. IEEE, 2016, pp. 1–2.
- [147] “Mnist cnn - keras documentation,” https://keras.io/examples/mnist_cnn/, (Accessed on 11/11/2019).
- [148] P. Warden, “Speech commands: A public dataset for single-word speech recognition,” *Dataset available from http://download.tensorflow.org/data/speech_commands_v0*, vol. 1, 2017.
- [149] L. Lee, J. Baek, K. S. Park, Y.-E. Lee, N. K. Shrestha, and M. M. Sung, “Wafer-scale single-crystal perovskite patterned thin films based on geometrically-confined lateral crystal growth,” *Nature communications*, vol. 8, no. 1, pp. 1–8, 2017.
- [150] C. Ge, W. Zhai, C. Tian, S. Zhao, T. Guo, S. Sun, W. Chen, and G. Ran, “Centimeter-scale 2d perovskite (pea) 2 pbbr 4 single crystal plates grown by a seeded solution method for photodetectors,” *RSC advances*, vol. 9, no. 29, pp. 16 779–16 783, 2019.

- [151] C.-K. Lin, Q. Zhao, Y. Zhang, S. Cestellos-Blanco, Q. Kong, M. Lai, J. Kang, and P. Yang, “Two-step patterning of scalable all-inorganic halide perovskite arrays,” *ACS nano*, vol. 14, no. 3, pp. 3500–3508, 2020.
- [152] C. Bao, W. Xu, J. Yang, S. Bai, P. Teng, Y. Yang, J. Wang, N. Zhao, W. Zhang, W. Huang *et al.*, “Bidirectional optical signal transmission between two identical devices using perovskite diodes,” *Nature Electronics*, vol. 3, no. 3, pp. 156–164, 2020.
- [153] N. Gavriely, Y. Palti, G. Alroy, and J. B. Grotberg, “Measurement and theory of wheezing breath sounds,” *Journal of Applied Physiology*, vol. 57, no. 2, pp. 481–492, 1984.
- [154] P. Piirila and A. Sovijarvi, “Crackles: recording, analysis and clinical significance,” *European Respiratory Journal*, vol. 8, no. 12, pp. 2139–2148, 1995.
- [155] M. Bahoura and C. Pelletier, “Respiratory sounds classification using gaussian mixture models,” in *Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No. 04CH37513)*, vol. 3. IEEE, 2004, pp. 1309–1312.
- [156] B. Rocha, D. Filos, L. Mendes, I. Vogiatzis, E. Perantoni, E. Kaimakamis, P. Natsiavas, A. Oliveira, C. Jácome, A. Marques *et al.*, “A respiratory sound database for the development of automated classification,” in *Precision Medicine Powered by pHealth and Connected Health*. Springer, 2018, pp. 33–37.
- [157] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [158] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient

- convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [159] B.-S. Lin and B.-S. Lin, “Automatic wheezing detection using speech recognition technique,” *Journal of Medical and Biological Engineering*, vol. 36, no. 4, pp. 545–554, 2016.
- [160] M. Bahoura, “Pattern recognition methods applied to respiratory sounds classification into normal and wheeze classes,” *Computers in biology and medicine*, vol. 39, no. 9, pp. 824–843, 2009.
- [161] J. Zhang, W. Ser, J. Yu, and T. Zhang, “A novel wheeze detection method for wearable monitoring systems,” in *Intelligent Ubiquitous Computing and Education, 2009 International Symposium on*. IEEE, 2009, pp. 331–334.
- [162] P. Bokov, B. Mahut, P. Flaud, and C. Delclaux, “Wheezing recognition algorithm using recordings of respiratory sounds at the mouth in a pediatric population,” *Computers in biology and medicine*, vol. 70, pp. 40–50, 2016.
- [163] I. Sen, M. Saraclar, and Y. P. Kahya, “A comparison of svm and gmm-based classifier configurations for diagnostic classification of pulmonary sounds,” *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 7, pp. 1768–1776, 2015.
- [164] N. Jakovljević and T. Lončar-Turukalo, “Hidden markov model based respiratory sound classification,” in *Precision Medicine Powered by pHealth and Connected Health*. Springer, 2018, pp. 39–43.
- [165] R. X. A. Pramono, S. Bowyer, and E. Rodriguez-Villegas, “Automatic adventitious respiratory sound analysis: A systematic review,” *PloS one*, vol. 12, no. 5, p. e0177926, 2017.

- [166] H. Chen, X. Yuan, Z. Pei, M. Li, and J. Li, “Triple-classification of respiratory sounds using optimized s-transform and deep residual networks,” *IEEE Access*, vol. 7, pp. 32 845–32 852, 2019.
- [167] E. Hosseini-Asl, G. Gimel’farb, and A. El-Baz, “Alzheimer’s disease diagnostics by a deeply supervised adaptable 3d convolutional network,” *arXiv preprint arXiv:1607.00556*, 2016.
- [168] M. J. van Grinsven, B. van Ginneken, C. B. Hoyng, T. Theelen, and C. I. Sánchez, “Fast convolutional neural network training using selective data sampling: Application to hemorrhage detection in color fundus images,” *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1273–1284, 2016.
- [169] Y. Song, L. Zhang, S. Chen, D. Ni, B. Lei, and T. Wang, “Accurate segmentation of cervical cytoplasm and nuclei based on multiscale convolutional network and graph partitioning,” *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 10, pp. 2421–2433, 2015.
- [170] O. Oktay, W. Bai, M. Lee, R. Guerrero, K. Kamnitsas, J. Caballero, A. de Marvao, S. Cook, D. O’Regan, and D. Rueckert, “Multi-input cardiac image super-resolution using convolutional neural networks,” in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 246–254.
- [171] P. Kisilev, E. Sason, E. Barkan, and S. Hashoul, “Medical image description using multi-task-loss cnn,” in *Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 121–129.
- [172] P. V. Tran, “A fully convolutional neural network for cardiac segmentation in short-axis mri,” *arXiv preprint arXiv:1604.00494*, 2016.
- [173] H. K. van der Burgh, R. Schmidt, H.-J. Westeneng, M. A. de Reus, L. H. van den Berg, and M. P. van den Heuvel, “Deep learning pre-

- dictions of survival based on mri in amyotrophic lateral sclerosis,” *NeuroImage: Clinical*, vol. 13, pp. 361–369, 2017.
- [174] T. Kooi, B. van Ginneken, N. Karssemeijer, and A. den Heeten, “Discriminating solitary cysts from soft tissue lesions in mammography using a pretrained deep convolutional neural network,” *Medical physics*, vol. 44, no. 3, pp. 1017–1027, 2017.
- [175] X. Chen, Y. Xu, D. W. K. Wong, T. Y. Wong, and J. Liu, “Glaucoma detection based on deep convolutional neural network,” in *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE, 2015, pp. 715–718.
- [176] Y. Bengio, P. Simard, P. Frasconi *et al.*, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [177] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent advances in recurrent neural networks,” *arXiv preprint arXiv:1801.01078*, 2017.
- [178] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, “Action recognition in video sequences using deep bi-directional lstm with cnn features,” *IEEE Access*, vol. 6, pp. 1155–1166, 2018.
- [179] Y. Zhao, X. Jin, and X. Hu, “Recurrent convolutional neural network for speech processing,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5300–5304.
- [180] H. Chang, J. Han, C. Zhong, A. M. Snijders, and J.-H. Mao, “Unsupervised transfer learning via multi-scale convolutional sparse coding for biomedical applications,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1182–1194, 2018.

- [181] A. Payan and G. Montana, “Predicting alzheimer’s disease: a neuroimaging study with 3d convolutional neural networks,” *arXiv preprint arXiv:1502.02506*, 2015.
- [182] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, “Convolutional neural networks for patient-specific ecg classification,” in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 2608–2611.
- [183] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in *Advances in neural information processing systems*, 2016, pp. 4107–4115.
- [184] T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, and M. Aleksic, “A quantization-friendly separable convolution for mobilenets,” in *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*. IEEE, 2018, pp. 14–18.
- [185] J. Zhang, W. Ser, and D. Y. T. Goh, “A novel respiratory rate estimation method for sound-based wearable monitoring systems,” in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE, 2011, pp. 3213–3216.
- [186] X. Liu, W. Ser, J. Zhang, and D. Y. T. Goh, “Detection of adventitious lung sounds using entropy features and a 2-d threshold setting,” in *Information, Communications and Signal Processing (ICICS), 2015 10th International Conference on*. IEEE, 2015, pp. 1–5.
- [187] “Apollo2 mcu – ambiq micro,” <http://ambiqmicro.com/apollo-ultra-low-power-mcu/apollo2-mcu/>, (Accessed on 01/11/2017).
- [188] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, “A micro-power eeg acquisition soc with integrated

- feature extraction processor for a chronic seizure detection system,” *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 804–816, 2010.
- [189] K. Kochetov, E. Putin, M. Balashov, A. Filchenkov, and A. Shalyto, “Noise masking recurrent neural network for respiratory sound classification,” in *International Conference on Artificial Neural Networks*. Springer, 2018, pp. 208–217.
- [190] D. Perna, “Convolutional neural networks learning from respiratory data,” in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2018, pp. 2109–2113.
- [191] G. Chambres, P. Hanna, and M. Desainte-Catherine, “Automatic detection of patient with respiratory diseases using lung sound analysis,” in *2018 International Conference on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2018, pp. 1–6.
- [192] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.
- [193] J. Sang, S. Park, and J. Lee, “Convolutional recurrent neural networks for urban sound classification using raw waveforms,” in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 2444–2448.
- [194] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [195] P. Warden, “Speech commands: A public dataset for single-word speech recognition,” *Dataset available from http://download.tensorflow.org/data/speech_commands_v0*, vol. 1, 2017.

- [196] R. X. A. Pramono, S. A. Imtiaz, and E. Rodriguez-Villegas, "Evaluation of features for classification of wheezes and normal respiratory sounds," *PloS one*, vol. 14, no. 3, p. e0213659, 2019.
- [197] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.
- [198] S. Alyamkin, M. Ardi, A. Brighton, A. C. Berg, B. Chen, Y. Chen, H.-P. Cheng, Z. Fan, C. Feng, B. Fu *et al.*, "Low-power computer vision: Status, challenges, opportunities," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019.
- [199] T. Gokmen, M. Rasch, and W. Haensch, "Training lstm networks with resistive cross-point devices," *Frontiers in neuroscience*, vol. 12, p. 745, 2018.
- [200] J. Acharya, A. Basu, and W. Ser, "Feature extraction techniques for low-power ambulatory wheeze detection wearables," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2017, pp. 4574–4577.
- [201] D. Katz, "Fundamentals of embedded audio, part 3," Sep 2007. [Online]. Available: <https://www.eetimes.com/fundamentals-of-embedded-audio-part-3/>
- [202] W. Q. Lindh, M. Pooler, C. D. Tamparo, B. M. Dahl, and J. Morris, *Delmar's comprehensive medical assisting: administrative and clinical competencies*. Cengage Learning, 2013.
- [203] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "Ai benchmark: Running deep neural networks on an-

- droid smartphones,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.
- [204] A. Basu, J. Acharya, and T. K. an et . al, “Low-power, adaptive neuromorphic systems: Recent progress and future directions,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 6–27, 2018.
- [205] J. Acharya, A. Patil, X. Li, Y. Chen, S. C. Liu, and A. Basu, “A comparison of low-complexity real-time feature extraction for neuromorphic speech recognition,” *Frontiers in neuroscience*, vol. 12, p. 160, 2018.
- [206] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [207] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature*, vol. 521, p. 436–444, 2015.
- [208] G. Hinton, L. Deng, and D. Y. et. al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, p. 82–97, 2012.
- [209] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams et al., “Recent advances in deep learning for speech research at microsoft,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8604–8608.
- [210] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *Journal of Machine Learning Research*, vol. 12, p. 2493–2537, 2011.

- [211] D. Silver, A. Huang, and C. J. Maddison, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, p. 484–489, 2016.
- [212] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-based neuromorphic systems*. John Wiley & Sons, 2014.
- [213] O. Araque, I. Corcuera-Platas, J. F. Sanchez-Rada, and C. A. Iglesias, “Enhancing deep learning sentiment analysis with ensemble techniques in social applications,” *Expert Systems with Applications*, vol. 77, pp. 236–246, 2017.
- [214] E. Chong, C. Han, and F. C. Park, “Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies,” *Expert Systems with Applications*, vol. 83, pp. 187–205, 2017.
- [215] S. Dodge and L. Karam, “A study and comparison of human and deep learning recognition performance under visual distortions,” in *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE, 2017, pp. 1–7.
- [216] G.-Y. Son, S. Kwon *et al.*, “Classification of heart sound signal using multiple features,” *Applied Sciences*, vol. 8, no. 12, p. 2344, 2018.
- [217] C. Kwak and O.-W. Kwon, “Cardiac disorder classification by heart sound signals using murmur likelihood and hidden markov model state likelihood,” *IET signal processing*, vol. 6, no. 4, pp. 326–334, 2012.
- [218] S. R. Thiyagaraja, R. Dantu, P. L. Shrestha, A. Chitnis, M. A. Thompson, P. T. Anumandla, T. Sarma, and S. Dantu, “A novel heart-mobile interface for detection and classification of heart sounds,” *Biomedical Signal Processing and Control*, vol. 45, pp. 313–324, 2018.

- [219] T. Chen, K. Kuan, L. A. Celi, and G. D. Clifford, “Intelligent heart-sound diagnostics on a cellphone using a hands-free kit,” in *2010 AAAI Spring Symposium Series*, 2010.
- [220] T. Nilanon, J. Yao, J. Hao, S. Purushotham, and Y. Liu, “Normal/abnormal heart sound recordings classification using convolutional neural network,” in *2016 Computing in Cardiology Conference (CinC)*. IEEE, 2016, pp. 585–588.
- [221] M. Tschannen, T. Kramer, G. Marti, M. Heinzmann, and T. Wiatowski, “Heart sound classification using deep structured features,” in *2016 Computing in Cardiology Conference (CinC)*. IEEE, 2016, pp. 565–568.
- [222] Y. Zhang, S. Ayyar, L.-H. Chen, and E. J. Li, “Segmental convolutional neural networks for detection of cardiac abnormality with noisy heart sound recordings,” *arXiv preprint arXiv:1612.01943*, 2016.
- [223] R. B. Stein, “Some models of neuronal variability,” *Biophysical journal*, vol. 7, no. 1, pp. 37–68, 1967.
- [224] W. Gerstner, “Time structure of the activity in neural network models,” *Physical review E*, vol. 51, no. 1, p. 738, 1995.
- [225] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.
- [226] J. Arthur and K. Boahen, “Synchrony in silicon: The gamma rhythm,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 18, no. 6, pp. 1815–25, 2007.
- [227] S. M. Bohte, J. N. Kok, and H. La Poutre, “Error-backpropagation in temporally encoded networks of spiking neurons,” *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

- [228] R. Gütig and H. Sompolinsky, “The tempotron: a neuron that learns spike timing-based decisions,” *Nature neuroscience*, vol. 9, no. 3, p. 420, 2006.
- [229] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, “Backpropagation for energy-efficient neuromorphic computing,” in *Advances in Neural Information Processing Systems*, 2015, pp. 1117–1125.
- [230] R. V. Florian, “The chronotron: a neuron that learns to fire temporally precise spike patterns,” *PloS one*, vol. 7, no. 8, p. e40233, 2012.
- [231] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, “Swat: a spiking neural network training algorithm for classification problems,” *IEEE Transactions on Neural Networks*, vol. 21, no. 11, pp. 1817–1830, 2010.
- [232] F. Ponulak and A. Kasiński, “Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting,” *Neural computation*, vol. 22, no. 2, pp. 467–510, 2010.
- [233] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, “Method for training a spiking neuron to associate input-output spike trains,” in *Engineering Applications of Neural Networks*. Springer, 2011, pp. 219–228.
- [234] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, “Event-driven random back-propagation: Enabling neuromorphic deep learning machines,” *Frontiers in neuroscience*, vol. 11, p. 324, 2017.
- [235] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, “First-spike-based visual categorization using reward-modulated stdp,” *IEEE transactions on neural networks and learning systems*, no. 99, pp. 1–13, 2018.

- [236] Q. Yu, H. Tang, K. C. Tan, and H. Li, “Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns,” *Plos one*, vol. 8, no. 11, p. e78318, 2013.
- [237] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, “Spatio-temporal back-propagation for training high-performance spiking neural networks,” *Frontiers in neuroscience*, vol. 12, 2018.
- [238] E. Hunsberger and C. Eliasmith, “Spiking deep networks with lif neurons,” *arXiv preprint arXiv:1510.08829*, 2015.
- [239] Q. Liu and S. Furber, “Noisy softplus: a biology inspired activation function,” in *International Conference on Neural Information Processing*. Springer, 2016, pp. 405–412.
- [240] 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.
- [241] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going deeper in spiking neural networks: Vgg and residual architectures,” *Frontiers in neuroscience*, vol. 13, 2019.
- [242] H. Liang, S. Lukkarinen, and I. Hartimo, “Heart sound segmentation algorithm based on heart sound envelopogram,” in *Computers in Cardiology 1997*. IEEE, 1997, pp. 105–108.
- [243] D. Kumar, P. Carvalho, M. Antunes, J. Henriques, L. Eugenio, R. Schmidt, and J. Habetha, “Detection of s1 and s2 heart sounds by high frequency signatures,” in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2006, pp. 1410–1416.

- [244] C. S. Lima and M. J. Cardoso, “Phonocardiogram segmentation by using hidden markov models,” 2007.
- [245] D. Gill, N. Gavrieli, and N. Intrator, “Detection and identification of heart sounds using homomorphic envelopogram and self-organizing probabilistic model,” in *Computers in Cardiology, 2005*. IEEE, 2005, pp. 957–960.
- [246] S. E. Schmidt, C. Holst-Hansen, C. Graff, E. Toft, and J. J. Struijk, “Segmentation of heart sound recordings by a duration-dependent hidden markov model,” *Physiological measurement*, vol. 31, no. 4, p. 513, 2010.
- [247] D. B. Springer, L. Tarassenko, and G. D. Clifford, “Logistic regression-hmm-based heart sound segmentation,” *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 4, pp. 822–832, 2015.
- [248] Y. Han, J. Kim, and K. Lee, “Deep convolutional neural networks for predominant instrument recognition in polyphonic music,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 208–221, 2016.
- [249] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, “Deep speech: Scaling up end-to-end speech recognition,” *arXiv preprint arXiv:1412.5567*, 2014.
- [250] R. Lyon, “A computational model of filtering, detection, and compression in the cochlea,” in *ICASSP’82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7. IEEE, 1982, pp. 1282–1285.
- [251] M. Slaney, *Lyon’s cochlear model*. Apple Computer, Advanced Technology Group, 1988, vol. 13.

- [252] T. Masquelier and S. J. Thorpe, “Unsupervised learning of visual features through spike timing dependent plasticity,” *PLoS computational biology*, vol. 3, no. 2, p. e31, 2007.
- [253] F. Noman, C.-M. Ting, S.-H. Salleh, and H. Ombao, “Short-segment heart sound classification using an ensemble of deep convolutional neural networks,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 1318–1322.
- [254] F.-F. Li, A. Karpathy, and J. Johnson, “Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition,” <http://cs231n.stanford.edu/>.
- [255] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Neural Information Processing Systems (NIPS)*, 2012, pp. 1106–1114.
- [256] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *CVPR*, 2016.
- [257] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” in *CVPR*, 2014.
- [258] L. Deng, J. Li, and J.-T. H. et. al., “Recent advances in deep learning for speech research at Microsoft,” in *IEEE Intl. Conferences on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.
- [259] F. Javed, Q. He, L. E. Davidson, J. C. Thornton, J. Albu, L. Boxt, N. Krasnow, M. Elia, P. Kang, S. Heshka et al., “Brain and high metabolic rate organ mass: contributions to resting energy expenditure beyond fat-free mass,” *The American journal of clinical nutrition*, vol. 91, no. 4, pp. 907–912, 2010.

- [260] “Top 10 emerging technologies of 2015,” <https://www.weforum.org/agenda/2015/03/top-10-emerging-technologies-of-2015-2/>.
- [261] “Global Neuromorphic Chip Market,” <https://www.transparencymarketresearch.com/pressrelease/neuromorphic-chip-market.html>.
- [262] “Neuromorphic Chip Market Worth US 1.8 Billion by 2023,” <https://www.marketwatch.com/story/neuromorphic-chip-market-worth-us18-billion-by-2023-tmr-2017-05-18-82033139>.
- [263] Z. Pan, Y. Chua, J. Wu, M. Zhang, H. Li, and E. Ambikairajah, “An efficient and perceptually motivated auditory neural encoding and decoding algorithm for spiking neural networks,” *Frontiers in Neuroscience*, vol. 13, p. 1420, 2020. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fnins.2019.01420>
- [264] 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.
- [265] V. Padala, A. Basu, and G. Orchard, “A noise filtering algorithm for event-based asynchronous change detection image sensors on truenorth and its implementation on truenorth,” *Frontiers in neuroscience*, vol. 12, p. 118, 2018.
- [266] M. J. Skocik and L. N. Long, “On the capabilities and computational costs of neuron models,” *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 8, pp. 1474–1483, 2014.
- [267] A. Mohan, K. Gauen, Y.-H. Lu, W. W. Li, and X. Chen, “Internet of video things in 2030: A world with many cameras,” in *2017 IEEE*

- International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.
- [268] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, “A low power, fully event-based gesture recognition system,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252.
- [269] C. Brandli, R. Berner, M. Yang, S.-C. Liu, V. Villeneuve, and T. Delbruck, “Live demonstration: The “davis” dynamic and active-pixel vision sensor,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2014, pp. 440–440.
- [270] V. Padala, A. Basu, and G. Orchard, “A noise filtering algorithm for event-based asynchronous change detection image sensors on truenorth and its implementation on truenorth,” *Frontiers in Neuroscience*, vol. 12, p. 118, 2018.
- [271] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.
- [272] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 91–99, 2015.
- [273] S. Afshar, T. J. Hamilton, J. Tapson, A. van Schaik, and G. Cohen, “Investigation of event-based surfaces for high-speed detection, unsupervised feature extraction, and object recognition,” *Frontiers in Neuroscience*, vol. 12, 2018.

- [274] A. Valade, P. Acco, P. Grabolosa, and J.-Y. Fourniols, “A Study about Kalman Filters Applied to Embedded Sensors.” *Sensors*, vol. 17, no. 12, p. 2810, dec 2017.