Deep Spiking Neural Networks with Applications to Human Gesture Recognition

Yannan Xing

Neuromorphic Sensor Signal Processing Laboratory

Centre for Signal and Image Processing(CeSIP)

Department of Electronic and Electrical Engineering

University of Strathclyde, Glasgow

This thesis is submitted for the degree of

Doctor of Philosophy

November 10, 2020

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Professor John Soraghan, for his consistent support and guidance during my PhD career. Without his persistent help, the goal of this novel neuromorphic research would not have been realized. In addition, I would like to express my gratitude to Dr. Gaetano Di Caterina and Dr. Lykourgos Petropoulakis for their insight and knowledge that support my research. Further more, I owe my sincere appreciation to my colleague Paul Kirkland who is the person working on this leading-edge with me. I have traveled to many places in the world with Paul. During our study, there are numerous idea exchange and inspiration collision between us.

A big thanks to the University of Strathclyde, who provided me with a world learning research environment and research sponsorships. This research would not have been possible without the financial support of the Strathclyde.

I would also like to thank my friends Fanfei, Ming, Weijie and Shangen, who have lived and worked with me during my stay in the UK. Your company was a treasure when I encountered difficulties. And thanks for the happiness that my lovely kitten Baozi has always brought to me, and hope you will like the life in a different country.

Last, for my parent, thanks for all your support and set me off on the road to this PhD degree a long time ago.

Abstract

The spiking neural networks (SNNs), as the 3rd generation of Artificial Neural Networks (ANNs), are a class of event-driven neuromorphic algorithms that potentially have a wide range of application domains and are applicable to a variety of extremely low power neuromorphic hardware. The work presented in this thesis addresses the challenges of human gesture recognition using novel SNN algorithms. It discusses the design of these algorithms for both visual and auditory domain human gesture recognition as well as event-based pre-processing toolkits for audio signals.

From the visual gesture recognition aspect, a novel SNN-based event-driven hand gesture recognition system is proposed. This system is shown to be effective in an experiment on hand gesture recognition with its spiking recurrent convolutional neural network (SCRNN) design, which combines both designed convolution operation and recurrent connectivity to maintain spatial and temporal relations with address-event-representation (AER) data. The proposed SCRNN architecture can achieve arbitrary temporal resolution, which means it can exploit temporal correlations between event collections. This design utilises a backpropagation-based training algorithm and does not suffer from gradient vanishing/explosion problems.

From the audio perspective, a novel end-to-end spiking speech emotion recognition system (SER) is proposed. This system employs the MFCC as its main speech feature extractor as well as a self-designed latency coding algorithm to efficiently convert the raw signal to AER input that can be used for SNN. A two-layer spiking recurrent architecture is proposed to address temporal correlations between spike trains. The robustness of this system is supported by several open public datasets, which demonstrate state of the arts recognition accuracy and a significant reduction in network size, computational costs, and training speed.

In addition to directly contributing to neuromorphic SER, this thesis proposes a novel speech-coding algorithm based on the working mechanism of humans auditory organ system. The algorithm mimics the functionality of the cochlea and successfully provides an alternative method of event-data acquisition for audio-based data. The algorithm is then further simplified and extended into an application of speech enhancement which is jointly used in the proposed SER system. This speech-enhancement method uses the lateral inhibition mechanism as a frequency coincidence detector to remove uncorrelated noise in the time-frequency spectrum. The method is shown to be effective by experiments for up to six types of noise.

Contents

List of Figures

List of Tables

Abbreviations

1D	One dimension
2D	Two dimension
3D	Three dimension
AER	Address-Event-Representation
AI	Artificial Intelligence
ANN	Artificial Neural Network
BLSTM	Bidirectional Long short-term Memory
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
CWT	Continuous Wavelet Transform
DAS	Dynamic Audio Sensor
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
DSPs	Digital Signal Processors
DVS	Dynamic Vision Sensor
EEG	Electroencephalogram
ELU	Exponential Linear Unit
EMG	Electromyography

EPSP	Excitatory Post-Synaptic Potential
FBE	Frequency Band Energy
FC	Fully-Connected
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
GRU	Gate Recurrent Unit
HCI	Human-Computer Interaction
HMM	Hidden Markov Model
IF	Integrated and Fire
IHC	Inner Hair Cell
IMCRA	Improved Minima Controlled Recursive Averaging
IoTs	Internet of Things
IPSP	Inhibitory Post-Synaptic Potential
k-NN	K-nearest Neighbourhoods
LIF	Leaky-Integrated and Fire
LSTM	Long Short-Term Memory
MFCC	Mel Frequency Cepstral Coefficient
MFSC	Mel Frequency Spectral Coefficient
MLP	Multi-Layer Perceptron
MMSE	Minimum Mean-Square Error
NM	Neuromorphic
OMLSA	Optimally Modified-Log Spectral Amplitude
PSP	Post-Synaptic Potential

ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SCNN	Spiking Convolutional Neural Network
SCRNN	Spiking Convolutional Recurrent Neural Network
SER	Speech Emotion Recognition
SNN	Spiking Neural Network
SNR	Signal-to-Noise Ratio
SRM	Spike Response Model
SRNN	Spiking Recurrent Neural Network
STDP	Spike-Timing-Dependent-Plasticity
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
SWaP	Size, Weight and Power
TBPTT	Truncated Backpropagation Through Time
TDS	Time-Depth-Separable
WFMM	Weighted Fuzzy Min-Max

Chapter 1

Introduction

1.1 Preface

Over the past few years, artificial intelligence (AI) technologies have significantly affected infrastructure development in modern smart cities. Human-computer interaction (HCI) plays a vital role in bridging the machine and the user's demand. Human gesture recognition is an important area for development in an HCI system. The unimodal gesture recognition systems can be broadly categorized as three class regarding their inputs, as shown in Figure 1.1. The first class is vision-based gesture recognition such as face emotion recognition [7–9], body movement tracking [10], and gaze detection [11, 12]. The second category is the audio-based human gesture recognition, these tasks for example are speech emotion recognition [13], auditory speaker identification [14] and speech recognition [15]. The third type is the recognition systems that are based on other forms of sensors like electroencephalogram (EEG) signal [16] or Electromyography(EMG) signal [17] based gesture recognition.

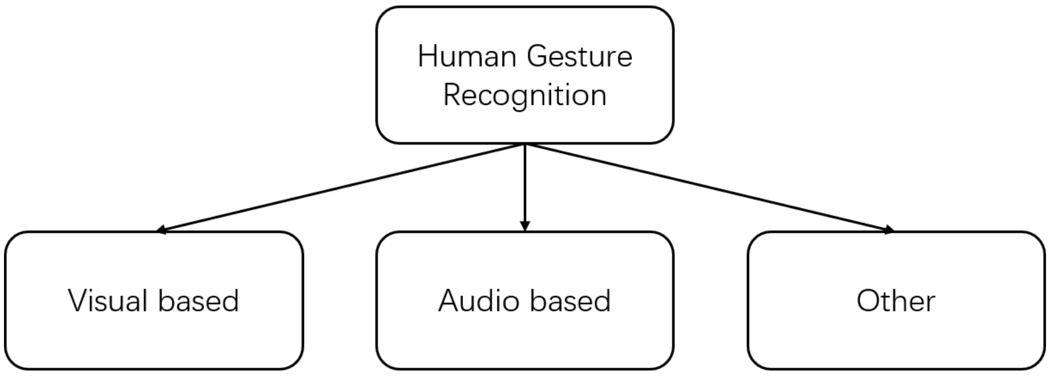


Figure 1.1: Illustration of Human gesture recognition

In a non-verbal human to machine communication framework, gesture recognition establishes a channel for acquiring intentions from human behaviour. The hand gesture recognition forms a key part of such interface since controlling or communicating with machines without physical contact particular has the advantages of convenience and effectiveness. The vision-based hand gesture recognition has been shown significant value in real-world HCI systems such as virtual reality [18, 19], robot control [20, 21] and sign language recognition [22, 23].

In the context of audio-based gesture recognition tasks, speech emotion recognition(SER) is one of the most natural and unaware ways of communication and exchanging in HCI. The SER is a challenging and active research area that uses generated speech signals to recognize speakers' qualitative emotional state, which usually has more information than spoken words [24]. The SER has a considerable potential to be applied in many real-time applications, such as it can be used in the board systems of a self-driving car to detect the mental conditions or emotional stability of drivers to ensure the safety of passengers [25]. Besides, literature has shown the SER has a significant achievement in the applications of automatic voice customer service [26], crowds violent detection [27] and smart health care [28].

1.2 Motivation

Recent developments in machine learning and deep learning techniques have demonstrated significant achievements in the field of HCI. Significantly, Artificial Neural Networks(ANNs) have pushed forward the state-of-the-art performance of various gesture recognition tasks. The current gesture recognition algorithms are mostly developed using statistical machine learning and neural network methods [21]. For example, Neverova et al. [29] successfully build a Convolutional Neural Network (CNN) based visual sign language gesture recognition system with 20 Italian upper-body skeletal motion data. Molchanov et al. [30] utilize the fused input feature successfully jointly developed a visual-based hand gesture recognition system for the in-car board system.

With the increasingly SWaP (Size, Weight, Power) demand, the mobile units, Internet of Things (IoTs) and embedded systems will consume the most AI techniques [31]. Applying statistical approaches like ANNs to large-scale HCI gesture recognition problems is still a challenging task in edge AI applications due to their limitations in computation capacity and power such as drones and robots. Compared to ANNs, the next generation of ANN spiking neural networks (SNNs) utilize event-based computation by employing a simplified bio-inspired neuron model as the fundamental processing unit and the event-based spike train as the information carrier [32]. SNNs, since their biologically plausible model and the nature of processing mechanism, has shown significant potential in terms of both power efficiency and computation speed.

Along with the development of SNN, another advancement in the field has come from in terms of SNN specialized hardware. These involve extremely low power neuromorphic (NM) chips (SNN computing processors) and neuromorphic sensors. One of the NM chip milestones is the IBM TrueNorth system [33] which consists of 5.4 billion transistors with only 70mW power density consumption, this only accounts for 1/10000 of traditional processing units. As for the sensor side, event-based vision and audio sensors mimic the biological retina and cochlear to provide sparse, asynchronous events to represents the input stimulus [34]. For example, the Dynamic Vision Sensor (DVS) [35] is a visual sensor for acquiring asynchronous events whenever a single pixel detects a change in terms of light intensity. The benefits of using a DVS involve high temporal resolution, ultra-low power consumption and high dynamic range.

Real-time vision-based hand gesture recognition is an HCI problem that is well suited to NM computing. Traditional systems suffer from various motion-related interferences (such as motion blur, light conditions, shutter) from the camera and the system challenged by power hungry and high latency problems. The related recent works [36–39], are either difficult to be applied into the real neuromorphic hardware or overemphasis the spatial feature but ignores temporal relations in such dynamic scenes problem. Therefore, developing a high performance, efficient, event-based, NM hardware applicable SNN is needed. A well designed hand gesture recognition SNN should be able to not only extensively take the advantages of event-based data from the NM sensors to achieve state-of-art recognition accuracy but also can be applied to developed neuromorphic hardware.

Another identified research gap is the field of applying NM technology into the recognition of auditory gesture tasks. Although researches have demonstrated many

1.3 Aims and Objectives

The aim of this research is to investigate the-state-of-art NM technologies and identify the applications of SNN in the field of NM human gesture recognition, which provides a biological plausible, energy efficient, reduced complexity solution compared to conventional statistical-based machine learning and ANN approaches.

The objectives of this study are identified as follows.

- To identify and design appropriate SNN models and training algorithms that match the engineering application requirements(high performance, energy/computing efficient, can be used on NM hardwares)
- To investigated and design novel SNN for the application of visual hand gesture recognition, which can utilize the benefits of event-based data.
- To evaluate and analyse the performance of designed SNN on a real DVS dataset.
- To develop an end-to-end spiking speech SER system.
- To develop spike coding and noise reduction algorithms that can improve the performance of speech emotion recognition system.

- To analyse the performance of the SER system by experiments on public datasets.

1.4 Original Contributions

The work conducted in this research has led to a number of contributions in the field of NM engineering which includes:

(i) Design a novel Spiking Convolutional Recurrent Neural Network (SCRNN) that takes advantage of both convolution operation and recurrent connectivity to maintain the spatial and temporal relations from event-based data. The use of recurrent architecture enables the network to have an arbitrary length of sampling window allowing the network to exploit temporal correlations between event collections(Chapter 4).

(ii) Applying the SCRNN architecture to the visual-based hand gesture recognition problem with an evaluation and analysis on an event-based gesture recognition dataset(Chapter 4).

(iii) Develop a software simulation tool that can efficiently convert the digitized audio signal to address event representation (AER) data. This work is inspired by the operating mechanism of the biological cochlear and the dynamic audio sensor (DAS), which can act as an NM audio sensor equivalent function in an NM system(Chapter 5.2).

(iv) Design a novel noise reduction algorithm that is based on neuron rate coding and bio-inspired SNN architecture. The excitatory-inhibitory topology in the network acts as the temporal characteristic synchrony and coincidence detector that removes uncorrelated noisy spikes. LIF source encoder is introduced along with the network. The network uses generated binary Short-Time Fourier Transform (STFT) masks according to the rate of processed spike train, which is used to reconstruct the denoised speech signal(Chapter 5.3).

(v) Design a novel end-to-end speech emotion recognition SNN system that takes the advantages of proposed speech processing algorithms(Chapter 5.2 and Chapter 5.3). The system directly takes the speech signal fed by microphone as the input with a cascaded SNN to perform pre-processing, spiking coding, denoising and recognition tasks(Chapter 6).

1.5 Thesis Outline

The thesis consists of 7 chapters. The first chapter contains the background, introduction, motivation and identified aims and objectives of this research.

Chapter 2 and Chapter 3 are two review chapters that consist of the literature review that covers relevant techniques and previous researches related to this study. Chapter 2 covers the review of conventional neural network structures that inspired the development of SNNs and the related ANN techniques for the gesture recognition tasks. This chapter provides the basic concepts of ANN with different architectures that popular with human gesture recognition. Chapter 3 introduces the principles of SNN along with the current developments of NM technologies such as Hardware, Software and corresponded implementations. The concept of SNN is explained and discussed in this chapter, which contains the introduction for spiking neurons, neural coding algorithms and various spiking learning strategies. The chapter builds up the knowledge foundation for the novel contributions that proposed in the novel chapters.

Chapters 4 to 6 present novel contributions of this research. Chapter 4 presents a novel Spiking Convolutional Recurrent Neural Network(SCRNN) structure that designed for recognising the event-based hand gesture data. The structure has the ability to maintain both spatial and temporal correlations that coded in the address event representation(AER) data, which potentially provides a neuromorphic solution to visual based hand gesture recognition challenge. Chapter 5 presents two novel event-driven speech processing algorithms that contributes to the preprocessing stage of the audio based gesture recognition problems. The first algorithm is designed for speech neural coding which offers an alternative routine to bridge the outside auditory stimulus to SNNs. The second contribution in this chapter is a speech enhancement algorithm which employs the modified version of the speech coding algorithm and neural lateral inhibition mechanism. The developed method is able to improve the speech quality in the time domain with the adaptability of 6 types of noise. A novel speech emotion gesture recognition system is presented in Chapter 6. The system consists of a novel preprocessing algorithm and a spiking recurrent neural network. The method is validated to be effective with three different open public datasets. In addition, the speech enhancement algorithms that developed in Chapter 5 is embedded into the system to enables an anti-noise ability of the system. Finally, the conclusion of the contributions in this research as well as the relevant future works are provided in Chapter 7.

1.6 Publications

Some aspects of the research work in this thesis have been published or submitted for publication.

Xing, Y., Kirkland, P., Di Caterina, G., Soraghan, J., & Matich, G. (2018, October). Real-time embedded intelligence system: emotion recognition on Raspberry Pi with Intel NCS. In *International Conference on Artificial Neural Networks* (pp. 801-808). Springer, Cham.

Xing, Y., Ke, W., Di Caterina, G., & Soraghan, J. (2019). Noise reduction using neural lateral inhibition for speech enhancement. In *International Journal of Machine Learning and Computing.*(Accepted for publication)

Ke, W., Xing, Y., Di Caterina, G., Petropoulakis, L., & Soraghan, J. (2020, February). Intersected EMG heatmaps and deep learning based gesture recognition. In *Proceedings of the 2020 12th International Conference on Machine Learning and Computing* (pp. 73-78).

Ke, W., Xing, Y., Di Caterina, G., Petropoulakis, L., & Soraghan, J. Deep Convolutional Spiking Neural Network Based Hand Gesture Recognition. In *International*

Joint Conference on Neural Networks. (Accepted for publication)

Xing, Y.,Di Caterina, G.,& Soraghan, J. A New Spiking Convolutional Recurrent Neural Network (SCRNN) with applications to Event-based Hand Gesture Recognition. In frontiers of Neuroscience. (Submitted for publication)

Xing, Y.,Di Caterina, G.,& Soraghan, J. A novel spiking neural network based speech emotion recognition system. IEEE Transactions of Neural Network and Learning System. (In preparation)

Chapter 2

Review of Conventional DNN Techniques with Applications to Human Gesture Recognition

2.1 Introduction

In this chapter, a review of conventional DNN techniques in terms of different statistical neural models and how they are used in the gesture recognition tasks is discussed. The ANN/DNN techniques will be described while highlighting three outstanding structures which inspires the novel designs in chapter 4 and chapter 6.

In recent years, DNNs with a range of different remarkable neural processing designs have been successfully applied to a range of human gesture recognition applications. For example, the development of convolutional neural network (CNN) [40] and its variants significantly improved the performance of many visual-based gesture recognition tasks due to the use of convolution and pooling operation to extract the spatial feature. Another example is the recurrent neural network (RNN) [41], which employs the mechanism of cyclical information propagation, which provides an effective computational model especially for 1D signal based gesture recognition.

The critical literature review of a range of DNN based human gesture recognition is provided in section 2.2. Section 2.3 provides a detailed explanation of the fundamentals of ANNs, which covers the knowledge of artificial neurons, multi-layer perceptron, activation function and training method. Then three outstanding human gesture recognition DNN architectures are described, which includes convolutional neural network(CNN) in section 2.4, the recurrent neural network in section 2.5 and convolutional recurrent neural network in section 2.6 respectively.

2.2 Previous Works on Human Gesture Recognition with DNNs

The previous researches on DNN based human gesture recognition can generally be categorized into 3 classes regard to the different formats of the input signal which have been described in section 1.1. The noticeable works in the field are shown in Table 2.1.

Table 2.1: The summary of recent ANN works for human gesture recognition

Vision Based Gesture Recognition			
Method	**Application**	**Dataset**	**Accuracy**
3D CNN [42]	Hand gesture recognition	VIVA	77.5%
CNN with 3D receptive field [43]	Dynamic hand gesture recognition	self-collected	80% -97.5%
Multi-channel CNN [44]	Face expression recognition	JAFFE	93.8%
CNN [45]	Body movement detection	self-collected	N/A
Attention CNN [46]	Human activities recognition	UCI HAR	91.58%
Attention CNN [47]	Facial expression recognition	FER2013,CK+	99.3%, 98%
Temporal CNN [48]	Human action recognition	NTU-RGBD	83.1%
CNN [22]	Sign language recognition	N/A	91.7%
BLSTM-3D residual network [49]	Dynamic sign language recognition	DEVISIGN_D	89.8%
Deep GRU [50]	Human action recognition	UT-kinect	100%
3D CNN [51]	Facial Palsy Grading	SLR_Dataset.	86.9%
Audio Based Gesture Recognition			
BLSTM [52]	Phoneme Recognition	TIMIT	77.4%
LSTM [53]	Speaker identification	LIEPA	94.46%
Attention LSTM [54]	Speech emotion recognition	eNTERFACE	85.7%
TDS CNN [55]	Speech recognition	LibriSpeech	N/A
Deep Self attention [56]	Speech recognition	Switchboard	N/A
Other sensor Based Gesture Recognition			
Radar signal + LSTM encoder [57]	Hand gesture recognition	self-collected	98.48%
IMU sensor + RCE neural network [58]	Hand gesture recognition	self-collected	98.6%
EMG signal + CNN [59]	Hand gesture recognition	self-collected	75%+
sEMG + feedforward [60]	Hand gesture recognition	self-collected	98.7%

CNN is the one technique that prevalent in a variety of gesture recognition works.

CNN and its variants are especially effective in terms of accuracy and computational cost to visual based gesture recognition tasks. Compared to traditional machine learning-based algorithms like support vector machine(SVM) [61] or k-Nearest neighbors (k-NN) [62], CNNs do not need to identify and pre-extract visual features manually but can learn high quality features automatically by their own [63]. Molchanov et al. [42] introduced a visual-based hand gesture recognition system that takes advantage of multiple intensity channels with a 3D CNN. Kim et al. [43] presented a weighted fuzzy min-max(WFMM) combined CNN that significantly increases the efficiency for spatio-temporal pattern extraction for video-based hand gesture recognition task. Hamester et al. [44] proposed a multi-channel convolutional neural network architecture that uses less computational resources for the application of facial expression recognition. In addition to these, CNNs have been used in gesture recognition tasks such as human body-movement identification [45], human activities recognition [46, 64], action recognition [48, 65, 66], sign language recognition [22, 67]. In addition, It is clear in the table that CNNs are not only efficient for visual-based gesture recognition tasks but also can be applied to auditory and other sensor-based problems. For instance, researchers from Facebook [55] developed a Time-Depth-Separable(TDS) convolution operation and performed experiments on the 960 hour speech corpus for the speech recognition problem, which gives superior results to the strong RNN baseline. Asif et al. [59] successfully employed CNN into an Electromyographic (EMG) signal based hand gesture recognition application and demonstrates a state-of-art recognition accuracy on a self-collected dataset.

Due to the advantages of long-term sequence processing of RNNs, they have been found to be very efficient in sequential information processing. In the domain of human gesture recognition, RNNs are particularly powerful in audio-based gesture recognition which can be regarded as a many-to-one problem. Graves et al. [52] proposed a hybrid approach that successfully applied a bidirectional LSTM to a phoneme classification and recognition task. A speaker identification LSTM is built by Dovydaitis el al. [68], which demonstrates superior results using an LSTM than the conventional Hidden Markov Model (HMM) and DNN. Xie et al. [54] proposed an attention gate processing unit that enables LSTM to accept the arbitrary size of the input and simultaneously reduces the computational complexity of LSTM, which significantly raises the recognition accuracy of LSTM in the SER task. The work in [53] shows the possibility of using RNNs for speaker identifications.

2.3 Artificial Neural Networks

Despite the fact that ANN is not a new concept that has been used since 1950s [69], the actual development wave of ANNs and DNNs started since around 2006 [70]. ANNs are commonly known as non-linear statistical data processing models that can easily model complex tasks [71]. A common feature of the various ANN/DNNs is that these networks have many stacked layers of hidden neurons combined with gradient descent based backpropagation [72] based training algorithms.

ANNs are initially biologically inspired by using approximated activation values and a series of weighted inputs. The neurons employ non-linear, differentiable activation functions that enable the network to achieve arbitrary computational complexity by stacking the neuron layers topologically [73]. The existence of neuron derivatives values allows several gradient-based optimization algorithms to be applied to reduce the error between the network input and output. With recent development in the acquiring of large scale labeled datasets and the computation capacity of Graphical Processing Unit(GPU), ANNs have become an effective solution to many big data-driven based tasks.

A DNN typically means an ANN that has more than 3 layers of hidden neurons, which modeled as a multilayer perceptron [74] that is trained to learn a statistical representation from a dataset without any manual feature extraction. As the name of Deep Neural Network suggests, it is using a higher number of hidden layers to represent data with multiple levels of abstraction [75]. In recent years, DNN based models have been successfully applied in many human gesture recognition tasks. For instance, Zhang et al. [76] build a cascaded video-based hand gesture recognition network using 3DCNN and ConvLSTM network, which efficiently learn the global temporal correlation information completely. Fayek et al. [77] evaluated multi-layered feedforward and DNNs/RNNs on the SER tasks with different configuration settings and yields state-of-the-art results on the specific dataset.

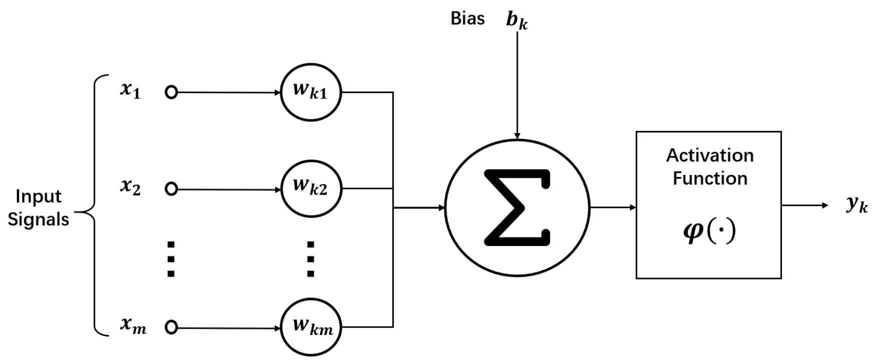


Figure 2.1: Schematic diagram an artificial neuron

2.3.1 Artificial Neurons

"Neurons" in ANNs are the basic statical computing units connected to other neurons via weight connectors. Artificial neurons sometimes named perceptron [69] calculate the weighted sum of the incoming information and then apply an activation function and bias value to generate the output. Figure 2.1 illustrates a single artificial neuron k receives an input signal $x_i\{i = 1, 2, 3...m\}$ and produce a output of y_k. There are three basic elements that can be identified in this neuron model. The first element is the synaptic weight between the inputs and neuron k, which is the value that will be multiplied to the corresponding input. The second element is an integrated which sums all the weighted input. The third element is an activation function $\varphi(\cdot)$ that introduces the nonlinearity into the computation and produces the output of a neuron. The activation function [78] sometimes also names as a "squashing function" that limits the output range to a finite value. Also, a bias value b_k is commonly used in a neuron model to shifting the neuron output to the desired range. Thus, the overall computation model of the neuron k can be written as:

$$y_k = \varphi(\sum_{i=1}^{m} x_i w_{ki} + b_k) \tag{2.1}$$

2.3.2 Multi-Layer Perceptron and Neural Networks

In 1986, Rumelhart et al. [79] introduced the concept of backpropagation and hidden layers, which creates a new era of ANNs. The backpropagation is a procedure to iteratively adjust the network parameter to reduce the difference between target and actual output. More advancements of the backpropagation technique will be reviewed in section 2.3.4 of this chapter. The hidden layer transforms the previous single-layer network to a multilayer perceptron(MLP) as is shown in Figure 2.2. This means introducing more layers of neurons between the input and output layers. A neuron in a hidden layer receives all the input signal from the previous layer and produce a processed output using (2.1) and send it to the neurons connected to it in the next layer. This process will iterate over all the hidden layers neurons, and then the final output is generated by applying a dot product for the last hidden layer outputs and weights.

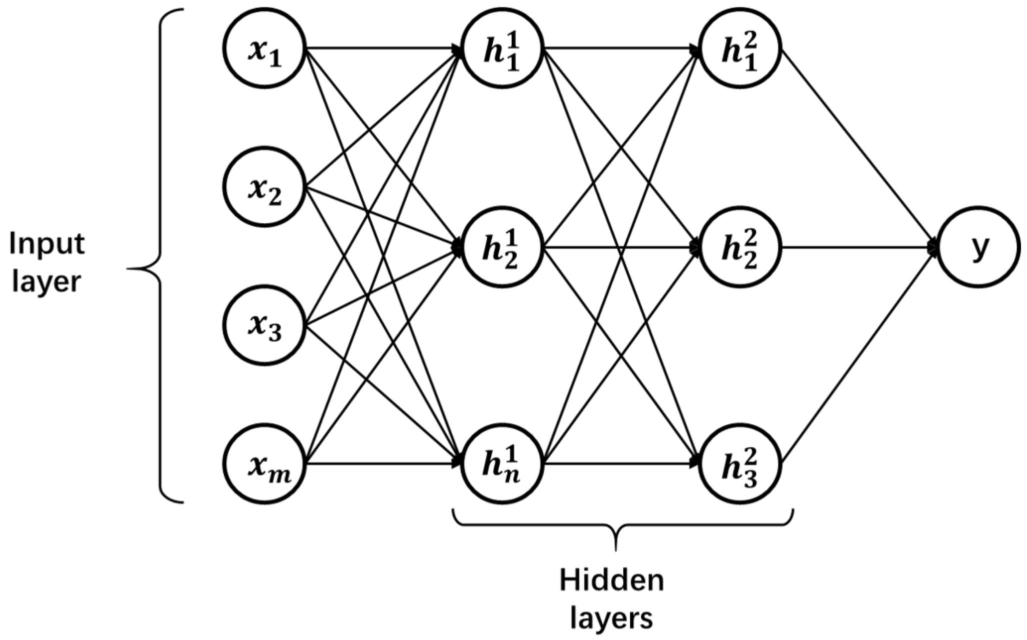


Figure 2.2: Schematic diagram of multiplayer perceptrons

2.3.3 Activation Functions

Activation functions are mathematical models attached to each neuron in a network and determine whether a neuron should be activated regarding the relevance between the input and network prediction. It also normalizes the entire network output in a range depends on the types of the activation function.

An important factor of an activation function selected for a big network is that it has to be computationally efficient since it is used across each neuron in a network for each data. Especially for a deep neural network, an inefficient activation function can waste numerous computation resources.

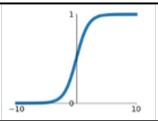
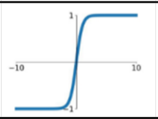
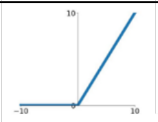
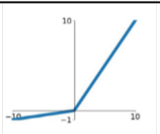
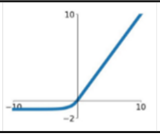
Recent works in ANN/DNNs employ gradient-based back-propagation training algorithms to reduce the loss between the network output and predictions, which highly rely on the derivatives of the activation functions [80, 81]. Thus, the nonlinearity of the activation functions is crucial. It enables multilayer networks to create complex mappings and allows the gradients to flow smoothly within the network.

Table 2.2 shows the recent popular activation functions commonly used in ANNs with their pros and cons. The Rectified linear unit(ReLU) [82] is one of the landmark activation functions in modern neural nets due to its simplicity of derivative and computationally efficient. It does not suffer from the problems in terms of gradient saturating like the sigmoid and the tanh function since its linear behaviour for positive input and it also makes the network converge faster than previous activation functions. The negative input to ReLU will lead to its derivative becomes zeros, which prevents the network from training. This is known as the dead ReLU problem, and many works have been done to addressing this problem such as Leaky ReLU [83] and ELU [84]. However, there is no perfect activation function that can satisfy all the requirements regarding different computational efficiency, performance and network training, the selection of activation for a neural network is still objective dependent.

2.3.4 Training Method

Training an ANN is not an easy task since the initial weights and biases in a network are totally random and weights in layers are highly interreliant. Thus a change in any

Table 2.2: Summary of development of activation functions

Method	Graph	Advantages	Disadvantages
Sigmoid		• Smooth gradient • Output value boundary • Clear predictions	• Gradient Vanishing • Output shifted from zero center • Computationally expensive
tanh		• Smooth gradient • Output value boundary • Clear predictions • Output zero centered	• Gradient Vanishing • Computationally expensive
ReLU		• Computationally efficient • Solve the gradient explosion ploblem	• When inputs approach zero, the gradient becomes zero(cannot learn)
Leaky ReLU		• Computationally efficient • Solve the gradient vanishing and explosion ploblem • Prevents ReLU dying	• Does not provide consistent outputs for negative values
ELU		• No dead ReLU situation • Closer to zero mean	• Higher computational costs since exponential function

connection will result in not only on the corresponding neuron but the effect to all the neurons in subsequent layers. Therefore, it is impossible to get the finest weights of the entire network by a single step optimizing operation. The optimum result is attained by iteratively evaluating the datasets until the overall weight sets converge to desired results. [85].

The modern neural network training algorithm can be classified into two categories: supervised learning and unsupervised learning. The supervised learning is a learning model that network both input X and an output/target variable Y. The goal is to approximate a function of $f(\cdot)$ that can map the input to output (i.e. $Y = f(X)$). This learning process from pre-known labels which can be regarded as a teacher supervising the learning process. The algorithm repeatedly predicts the output according to the training input samples and the output is corrected by the teacher. Unsupervised learning is commonly used when the label of the input data is not available. Unsupervised learning differently from supervised learning is more emphasis on modeling the underlying distributions or patterns of input data rather than specifically outputs a prediction value. In the deep learning domain, supervised learning is suitable for classification and regression based tasks and Unsupervised learning is applied to clustering and association tasks.

It is worth mentioning the gradient descent method [86] that is one of the most powerful supervised algorithms to optimize complex function by repeatedly calculating the current state of gradient and then takes step proportional to the negative direction of the gradient to find a local minimum(optimized value) of the loss function. The result modifications made to network parameters tend to reduce the error between the network outputs and labels. This process is then repeated until the network outputs a satisfactory result.

The gradient descent is a straightforward method that can automatically choose the learning rate (the level of changes in each weight modification) for the neural network training. Many works have been done in order to balance the network training performance and speed. Table 2.3 demonstrates an overview of recent advanced developments in DNN training algorithms. The gradient descent [86] is the most basic optimization algorithm used in a variety of works. It approximates the routine that the weights should be modified so that the loss function can reach the minima. The traditional gradient descent, however, was found many disadvantages. First, the learning process is found to be easily trapped in a local minima due to the lack of proper adjustment of learning rate. The learning rate of a training algorithm affects the stability and convergence speech of the process [87]. If a learning rate is chosen too small, it leads to an expensive computational cost. In contrast, a big learning rate will results in violent oscillation during the training process and completely miss the global minimum point. In addition, weights are modified after the gradient calculated on the entire training dataset.

Kiefer et al. [88] proposed a stochastic gradient descent algorithm to enables fewer memory requirements based on the theory of traditional gradient descent. Later, Rumelhart et al. [79] introduced momentum, which overcame the disadvantages of noisy weight updates and accelerates the convergence speed of the network by applying an exponential weighting average term. Adagrad proposed in [89] firstly applied an adaptive learning rate for each weight updating process. The learning rate is then decreasing along with the number of training iterations. Recently, many advanced versions of Adagrad such as AdaDelta [90] and Adam [91] are presented to further improve the training performance in terms of training stability, computational cost and convergence speed.

Table 2.3: A summary of gradient decent training algorithm families

Method	Description	Advantages	Disadvantages
Gradient Decent [86]	1.First-order derivative dependent 2. Used heavily in linear regression and classification tasks	1. computational efficient 2. Easy to implement 3. Easy to understand	1. May stuck at local minima 2. weights changed after gradient computation(long training time) 3. Require large memory
Stochastic Gradient Descent [88]	1.A variant of Gradient Descent algorithm 2.Weights updated after computation loss on each iteration	1.Frequently weight update (faster convergence) 2. Less memory requirement without storage of loss function 3.Probably get new minima	1. High variance in model parameters 2. May shifting after getting global minima 3. Need of small learning rate to get high performance
Momentum [79]	1.Invented for reducing high variance 2. Accelerate convergence towards relevant direction 3. Reduce fluctuation to irrelevant direction	1. Reduce oscillation and high variance 2. Converges faster than gradient descent	1. One more hyper-parameter is introduced and need to be selected manually
Adagrad [89]	1. Solve the constant learning rate problem. 2. Dynamic weight update strategy	1. Learning rate changes along with training 2. Do not need manually learning rate modifications 3. Able to be applied on sparse data	1.Computationally expensive as it need to calculate second order derivative 2. The learning rate always decreasing
AdaDelta [90]	1. An extension of Adagrad that tends to remove decay learning rate problem	1.The learning rate does not always decay and training does not stop	1. The higher requirements of computation resources
Adam [91]	1. Works with momentums of first and second order. 2. Keeps an exponentially decaying of average past gradient	1. Very fast and converges rapidly 2. Rectifies vanishing learning rate and high variance	1.Computationally expensive

2.4 Convolutional Neural Network

The convolutional neural network(CNN) [40] is one of the key landmarks that contributes to the success and development of deep learning [92]. It has been applied in various domains due to the utilization of outstanding convolution operation which decomposes the high-level visual objects into low-level features. As is shown in Figure 2.3, typical CNN consists of an input layer, an output layer and several hidden layers. Differently from standard MLP, hidden layers in addition to fully connected(FC) layer have convolutional layers and pooling layers. CNNs have wide applications particularly in computer vision tasks such as face recognition [93], object detection [94, 95] and object segmentation [96].

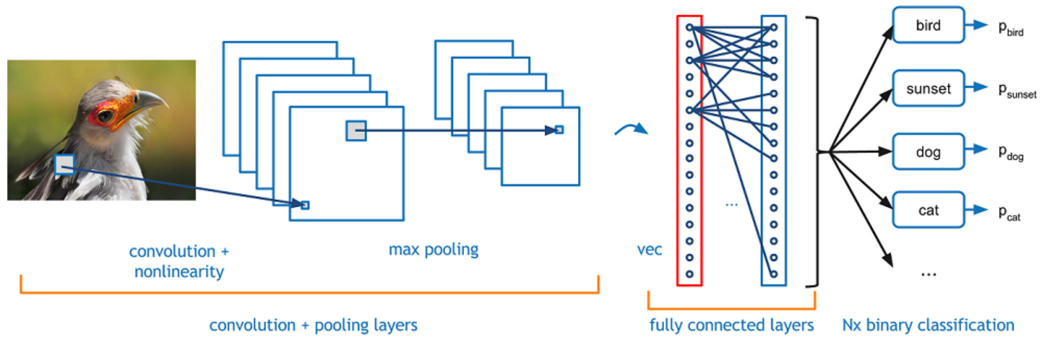


Figure 2.3: An example of convolutional neural network structure [1]

2.4.1 Convolutional Layer

The convolutional layer is the key and the most important layer of a CNN. It uses convolution operation to convolves the feature pixel matrix with the given pixel image to produce a feature map representing the spatial arrangements of a given feature.

The distinction of using the convolution operation for images is that it extracts all the distinguishing features into several feature maps and simultaneously reducing the size of the pixel matrix to be processed. The feature pixel matrix is called convolution kernel/filter which basically is a feature detector that includes various low-level visual features.

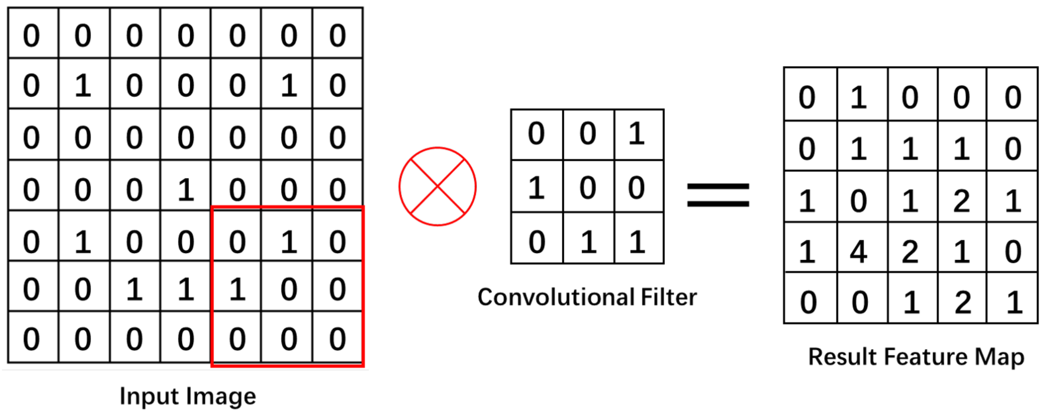


Figure 2.4: The convolution operation in a convolutional layer

Figure 2.4 demonstrates a graphical explanation of the process of an input image is being convolved with a 3x3 filter. The size of a convolution filter is equal to the receptive field of a neuron in the convolution layer, where in this case a single value in the feature map represents the intensity of the feature in a 3x3 dimension of the input image. Each neuron in the convolution layer only processes data only for its receptive field, and the size of the feature map depends on the size of the kernel and stride(Distance of each kernel shifting within the image). For example, A 3x3 kernel convolves with an image size of 7x7 with the stride of 1 results in a 5x5 feature map. The 2D convolution that generates the feature map G can be expressed as follow.

$$G[m,n] = (I * k)[m,n] = \sum_{j}\sum_{k} k[j,k]{\cdot}I[m-j,n-k] \tag{2.2}$$

where I denotes to input pixel matrix and k to the convolution kernel. G is the output feature map and m, n are horizontal and vertical pixel index respectively. The subsequent feature map is guaranteed to has a shrink in terms of dimension every time a convolution operation is performed, thus the convolution layer brings a significant benefit which reduces the number of free parameters and allows the network to be deeper.

2.4.2 Pooling Layer

In addition to convolution layers, the pooling layer is very often used in CNNs to further reduce the dimension of the feature maps and speed up the training process. The idea of pooling operation is to keep only the most important information within a receptive field while also reducing the spatial invariance. This also reduces the amount of learnable free parameters for a model. There are different types of pooling operations such as max pooling and average pooling.

Figure 2.5 demonstrates a pooling process of the max pooling operation. The max pooling operation simply takes the greatest value from each sub-matrix(receptive field) and forms a shrunk version of the original feature map. For example, a 5x5 feature map is pooled by a 2x2 pooling kernel in Figure 2.5, the pooled feature map only preserves

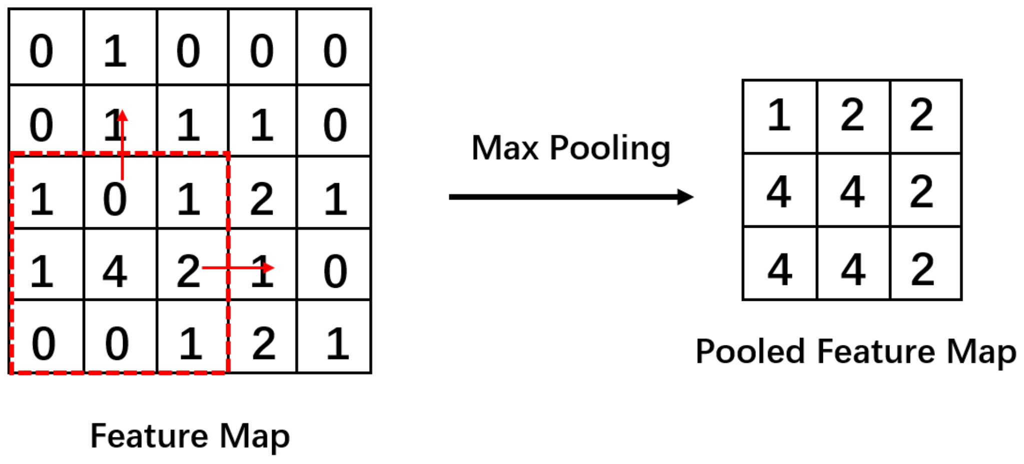


Figure 2.5: The pooling operation in a pooling layer

the highest value of a 2x2 region and this gives a pooled feature map with the dimension of 3x3.

2.4.3 Fully Connected Layer

The fully connected(FC) layer is often used in the last layer of a CNN. The processed feature map is flattened from 2D to 1D before fed into FC layers. Thus the extracted features are broken into vectors that can be processed by standard MLP. Thus it is hard to follow the data after this layer due to the loss of spatial arrangements.

2.5 Recurrent Neural Network

Recurrent neural networks (RNNs) [41] have been commonly recognized as an ANN structure that powerful in dealing with sequential data, such as text, audio and video data. Unlike the standard feedforward neural network structure, RNNs capture the critical dynamics of a sequence via cycle connections in network nodes. Although recurrent structure makes an RNN is more difficult to train and it usually contains millions of free parameters, recent developments in network optimization, parallel computing and tuning method have enabled many different successful training ways that can be used in RNNs.

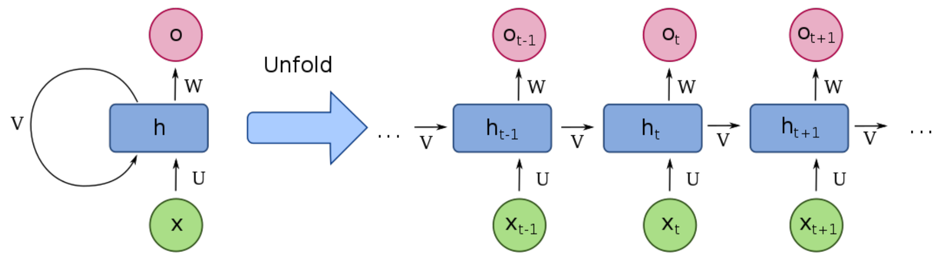


Figure 2.6: Diagram of a one-unit RNN [2]

Figure 2.6 illustrates a basic RNN which from bottom to top consists of 3 fundamental components: input state, hidden state and output state respectively. RNN introduces a notion of time to the model, where the processed information in the current step will be transferred as a part of input in the next time step. For example at time t, current hidden state $h^{(t)}$ is formed by combining current input data $x^{(t)}$ and previous hidden state $h(t-1)$. The output $o^{(t)}$ is then computed at each time step via correspond hidden state $h^{(t)}$. With W^{xh} denotes to input-hidden weights and W^{hh} denotes to hidden-hidden weights, W^{ho} denotes to hidden-output weight, vectors b_h and b_y to bias value of hidden state and output state respectively, the process of a standard RNN can be expressed as:

$$h^{(t)} = \varphi(W^{xh}x^{(t)} + W^{hh}h^{(t-1)} + b_h) \tag{2.3}$$

$$o^{(t)} = \varphi(W^{ho}h^{(t)} + b_y) \tag{2.4}$$

Despite the advantages of RNNs in sequence learning, training an RNN has long been challenging due to the gradient vanishing/explosion problems. This phenomenon occurs when backpropagation across too many layers. This is particularly common in an RNN since the input sequences are usually very long (e.g. text recognition tasks) [97]. Assuming an example of RNN with a single input node, hidden state and the output node(Figure 2.6)and a sequence is injected to this RNN at time *tau* and an error signal calculated at time t. As $t - \tau$ goes large, the contribution of input at t to output at time t will either grows to infinity or decays to zero. Thus the corresponding derivative of the error signal is also vanished or exploded.

Technically, RNN can either be trained with only a few parameters or as an unfolded feedforward manner across many time steps. The selection of the training algorithms for RNNs is a trade-off between network performance and computational costs. It is clear that training a unfolded RNN dramatically increased free parameters and the length of backpropagation in the model, which can usually get better results than folded RNN. In this way, the backpropagation will be passing through all the unfolded parameters in each unfolded time step, which is named backpropagation through time (BPTT) [98]. A folded RNN has very few parameters and the information is recurrently propagating through the same model thus require much less computational resources. This strategy however usually degrades the network performance since it does not contain enough parameters to well modeling the problem.

Truncated backpropagation through time(TBPTT) [99] is a solution that can balance both the gradient problem and the computational performance of RNN, which enables continuously training for RNNs. It defines a limit number of time steps that backpropagation can propagates and segments the input sequence to several fragments with the same length as the number of time steps.

2.5.1 Long short-term Memory

Long short-term memory(LSTM) [100] is a brilliant RNN design that introduced by Hochreiter and Schmidhuber to solve the gradient vanishing/explosion problems and handle long-term dependencies. LSTM replaces the original RNN neurons with a hidden cell that consists of 5 elements Input node as is shown in Figure 2.7. For an introduction purpose, these five key elements Input gate, Hidden state, Forget gate and Output gate are described as below.

A. **Input Node $g^{(t)}$**

   The Input node is a channel that receives the value from input $x^{(t)}$ at the current time step and hidden state $h^{(t-1)}$ from the previous time step. A tanh or sigmoid activation function is used to process the sum weighted input signal.

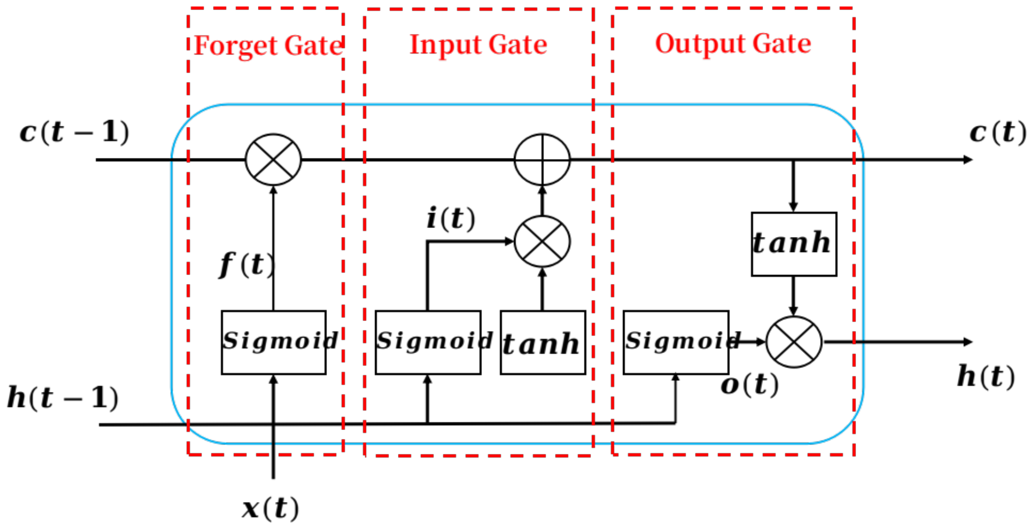


Figure 2.7: Diagram of a LSTM.

B. **Input Gate** $i^{(t)}$

The input gate controls the flow of the data from the input node. It is a sigmoidal unit that takes the activation value from input $x^{(t)}$ as well as from the previous layers. It is called a gate since its value is used to multiply the values from other gates. If the gate value is 0, then the correspond propagation route is cut-off. In contrast, the gate value with 1 will allow all the information pass through.

C. **Forget Gate** $f^{(t)}$

The forget is introduced into LSTM by Gers et al [101]. The forget provides a method that can learn to what extent the information in the hidden state should be kept and thrown away. This is particularly useful in maintaining the network stability during the training process.

D. **State** $s^{(t)}$

The state is the key in a LSTM cell which represents the status of the information at the current time step. It spans the integrated information from forget gate and input gate to adjacent time steps and simultaneously propagates them to output gate. With the forget gate, the process that generates the hidden state of current time step can be written as:

$$s^{(t)} = g^{(t)} \odot i^{(t)} + f^{(t)} \odot h^{(t-1)} \tag{2.5}$$

where $\odot$ represents the pointwise multiplication.

E. **Output Gate** $o^{(t)}$

An output produced by a LSTM cell at a single time step is the value of hidden state $h^{(t)}$ multiplied by the value of the output gate. It summarizes the hidden state by applying a tanh function which limits the output of each LSTM cell within the same dynamic range.

In a summary, the overall calculations of a LSTM can be expressed as follows.

$$g^{(t)} = \phi(W^{(xg)}x^{(t)} + W^{(gh)}h^{(t-1)} + b_g) \tag{2.6}$$

$$i^{(t)} = \sigma(W^{(xi)}x^{(t)} + W^{(ih)}h^{(t-1)} + b_i) \tag{2.7}$$

$$f^{(t)} = \sigma(W^{(xf)}x^{(t)} + W^{(fh)}h^{(t-1)} + b_f) \tag{2.8}$$

$$o^{(t)} = \sigma(W^{(xo)}x^{(t)} + W^{(oh)}h^{(t-1)} + b_o) \tag{2.9}$$

$$s^{(t)} = g^{(t)} \odot i^{(t)} + f^{(t)} \odot h^{(t-1)} \tag{2.10}$$

$$h^{(t)} = \phi(s^{(t)}) \odot o^{(t)} \tag{2.11}$$

where ϕ and σ denotes to tahn and sigmoid activation function respectively. $h^{(t)}$ represents the hidden layer value of the LSTM at the time t. The superscription/subscription x, g, h, x, f, o of W are the correspond gate abbreviation which denotes to weight values of different connections. For example, $W^{(xf)}$ is the input to forget gate weight values, b_g is the bias of input nodes.

It should be noted that gate components in LSTM can learn when to switch ON/OFF gate to control the forward/backward information propagate [41]. This demonstrates a significant ability to learning long-term dependencies compared to conventional RNNs.

2.5.2 Variations of RNN

Currently, the term RNNs in the modern ANN domain are mostly mean LSTM and its variations. Since the successful development of LSTM, several variations have been proposed based on this structure such as a Gated Recurrent Unit (GRU) [102] that integrates the forget gate and input gate of the original LSTM cell to reduce the parameters. Schuster and Paliwal [103] presented a bidirectional RNN that enables the network can be trained both in the forward and backward direction(Along with the input sequence time axis). To determine the optimal architecture of an LSTM cell, Jozefowicz et al. [104] evaluated over 10000 different RNN structures and identified the MUT series that outperform the performance of both LSTMs and GRUs.

RNNs have been found that can be used for sequential input and sequential output [105]. Figure 2.8 demonstrates 5 types of structure that can model numerous tasks into RNNs. The structure in Figure 2.8(a) is an individual RNN cell that acts as a standard feedforward network that takes non-sequential input and produces non-sequential output. Figure 2.8(b) is commonly known as a many-to-one problem that suitable for tasks requiring a sequential input with a single output(e.g. text, audio, video classification). By contrast, the model in Figure 2.8(c) represents a one-to-many processing strategy that can be used for captioning related tasks. Last but not least, Figure 2.8(d) and (e) are two forms of the many-to-many RNN structure which have been widely used in natural language processing(NLP)and text predictions.

2.6 Convolutional Recurrent Neural Network

The convolutional recurrent neural network (CRNN) structure has been well studied in the second generation of ANNs. The convolution operation in the ANNs usually acts as a spatial visual feature extractor that assumes features are in different levels of hierarchy. The recurrent structure introduces memory to the network and an ability to deal with sequential data dependently.

A significant design of the CRNN structure is the ConvLSTM structure [106] that was initially designed for forecasting precipitation. By replacing the general gate acti-

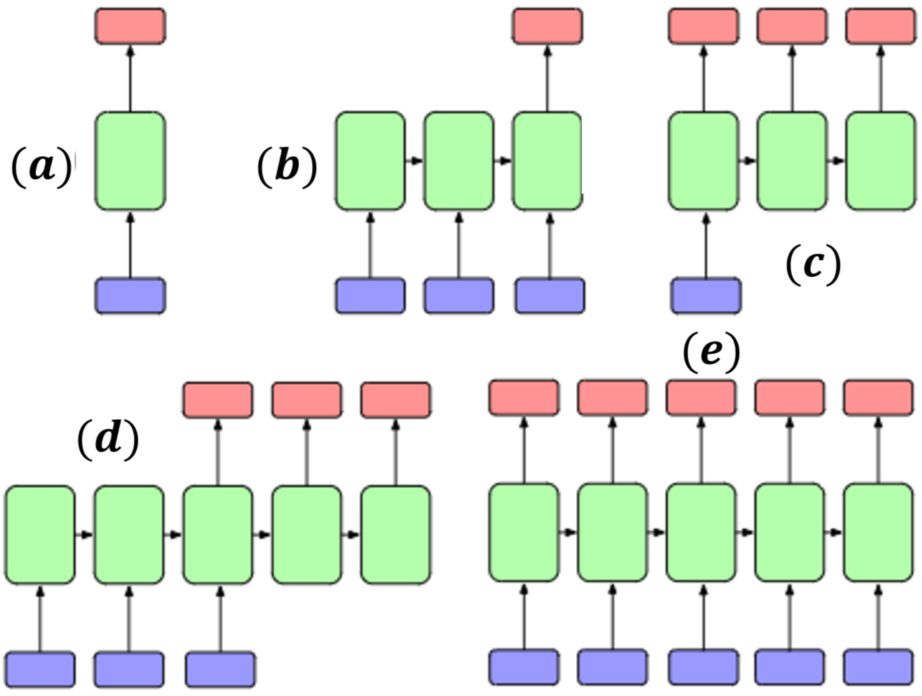


Figure 2.8: Graphical illustration of RNN tasks models

vation by the convolutional operation, the network is able to exploit an extracted 3D tensor as the cell state. The ConvLSTM was also evaluated on the moving MNIST [107] dataset and was shown to successfully separate the overlapping digits and predicted the overall motion with a high level of accuracy.

Another CRNN structure CNN-LSTM concatenates a CNN and an LSTM to formulate a joint network. The LSTM in the structure is placed behind a pretrained CNN that directly takes the output feature vector from the CNN as the input sequence. The implementation of this structure however is highly dependent on a well pre-trained CNN that was designed for the interest as the feature extractor. The CNN-LSTM is proved powerful in many application domains such as acoustic scene classification [108], emotion recognition [109], action recognition [110].

Over the past few years, researchers have successfully applied CRNN in medical applications [111], speech processing [15, 112], music classification [113]. Adopting a recurrent structure enables the neural network to encapsulate the global information while local features are extracted by the convolution layers. Yang et al [114] demonstrated a Convolutional LSTM network that was successfully evaluated on various human hand gesture recognition datasets. The importance of using CRNN structure in the application of visual-based gesture recognition is that unlike the recognition tasks in images, the same task in videos relies on motion dynamics in addition to visual appearance. Although CNNs and its variants like 3D convolution [115, 116] achieved good performance, they still do not make sufficient use of temporal relations between frames. More recently, Maj et al. [117] designed a motion-ware ConvLSTM for the hand gesture recognition task which is an LSTM unit that considers the correlation of consecutive video frames in addition to the spatio-temporal information.

2.7 Conclusion

This chapter illustrates a review of ANN/DNN techniques and how they have been used in human hand gesture recognition tasks. The fundamentals of statistical neural processing in terms of the neuron model, training method and activations were introduced. CNN and RNN are discussed as two important structures that have significantly contributed to visual and auditory based gesture recognition. These neural network landmarks form the basis for the chosen research inspirations and contributions in chapter 4 and 6 of this thesis.

The combination of CNN and RNN(CRNN) is reviewed since its importance in solving the video-based gesture recognition since it can well preserve both spatial and temporal information for dynamic scene recognition tasks. Despite the different processing mechanisms, the idea of using spatio-temporal feature for gesture recognition is also applicable in the SNN domain. The CRNN structure provides a key inspiration for the work in chapter 4.

Chapter 3

Review of Spiking Neural Networks and Neuromorphic Computing

3.1 Introduction

In this chapter, the literature review of the spiking neural network (SNN) is presented. Neurons in the second generation ANNs sending float points and employ continuous activation functions(e.g. sigmoid and tanh). SNNs, as the third generation of ANN, are inspired by way of biological information processing which typically sparse and asynchronous binary spikes are transported in a parallel manner. SNN is fundamentally different from ANNs in terms of the information carrier, basic processing unit, sensing technique and training methods. In the human brain, the communication between neurons is achieved by broadcasting trains of action potentials which also known as spike trains that have sample amplitude and sparse in time.

Form the scientific aspect, it is widely recognized that a series of processing states with learning mechanisms in a multilayer neural network significantly assist the human brain perception system to recognize complex visual patterns or auditory information in a noisy environment [118–120]. Compare to ANNs, SNNs provides the structure to better process temporal information. The representation of spikes in the spatial and temporal domain makes SNNs unique to address the event-based data which is closer to how actual neural systems works. More details of the SNN representations have been introduced in [121].

Form the engineering perspective, SNNs have several advantages compared to traditional ANNs/DNNs in terms of energy efficiency and the implementations in NM hardware. The SNN process takes the input, process, and output spike trains which sparse in time. An advantage of this is that the binary spike events consume only very few energy but can contain high information content in the spike timing [122]. Bio-inspired SNNs, technically, have a higher potential of power consumption and capacity than traditional neural networks [123]. Thus, it is not hard to consider the engineering application potential of the low energy NM systems that is highly responsive to event-based sensors regards to the property of SNNs.

3.2 Spiking Neuron

The Neuron in SNN represents the elementary processing unit which communicates with each other by sending and receiving spikes. Hodgkin and Huxley [124–126] carried out an experiment on the giant axon of a squid and build up the first conductance-based neuron model which can reconstruct the electrophysiological behaviour of the biological neuron. In 1996, Knig etc al. [127] suggests that a neuron can be regarded as an integrator or coincidence detector, which is commonly used as the fundamental idea of modern spiking neuron models. The spikes in the spiking neuron model are only identified at the time instant when they arrive at the neuron. A spiking neuron integrates the incoming spikes and transfers these spikes to a voltage change that is commonly termed as postsynaptic potential(PSP). Then, the overall PSP is compared to a pre-defined threshold. If the PSP reaches the threshold, then a spike is emitted by the spiking neuron. Figure 3.1 demonstrates a spiking neuron internal process where in (a) a single spiking neuron that receives incoming spike trains from s_1, s_2 and s_3 and generates an output spike.

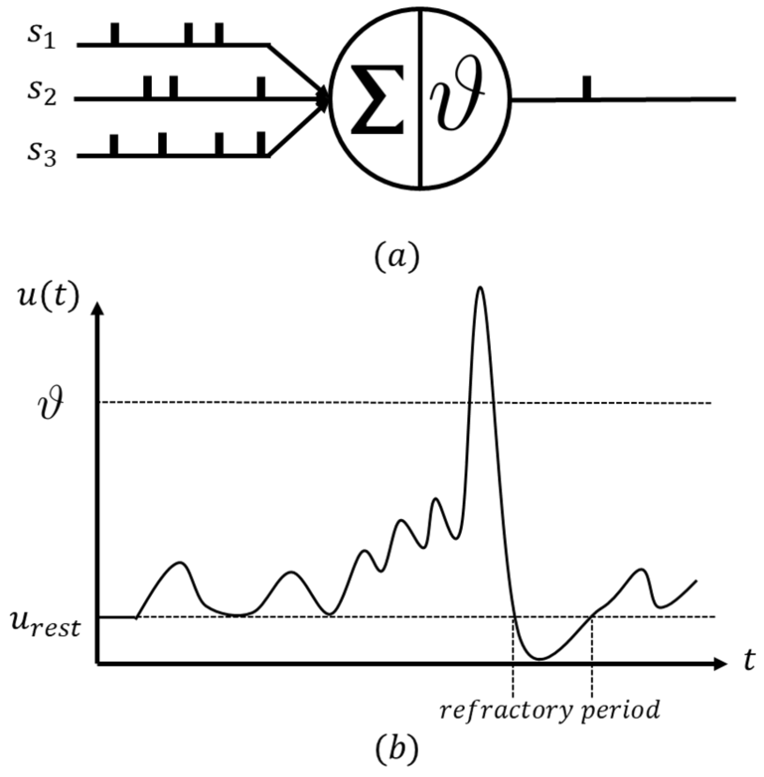


Figure 3.1: The illustration of spiking neuron operating mechanism (a): An example of a single spiking neuron that consists of integrator and threshold operator (b): A simulation of membrane potential $u(t)$ change of a spiking neuron

The incoming spikes to a neuron are integrated and transferred to the membrane potential dynamics $u(t)$ as is shown in Figure 3.1(b). Whenever the membrane potential reaches a certain threshold value ϑ, the spiking neuron will emit a spike and reset the membrane potential to its resting value u_{rest}. After a spike activity, the neuron enters the refractory period and cannot fire any further spikes until its membrane potential resets to its resting value. A typical spiking neuron model can contain additional parameters that approximate the membrane potential changes in the neural cortex. Commonly used spiking neuron model in SNNs include: Integrate and fire neurons(IF) [128, 129], Leaky integrated and fire neurons(LIF) [130], Izhikevich neuron model [131] and Spike Response Model(SRM) [132].

3.2.1 Leaky Integrate-and-Fire(LIF) Neuron Model

The integrated and fire neuron (IF) model [133] is the first model that is applied in event-based information processing. When the input spikes arrive at the neuron in time, the internal PSP potential then correspondingly increases or decreases depends on the value of the synaptic weight. The PSP due to positive weights commonly termed as excitatory PSP(EPSP) and the negative weights lead to an inhibitory PSP(IPSP). Whenever the PSP reaches a threshold, an output spike is released. The LIF introduces a decay term to the original IF model which leads to the PSP decay over time [134]. This means that when a neuron is not receiving any input spikes, the PSP will gradually decrease to the resting potential. This mechanism can be modeled as a simple RC circuit. The differential equation of a LIF neuron can be expressed as:

$$C_m \frac{dV_m(t)}{dt} = -\frac{V_m(t)}{R_m} + I(t) \tag{3.1}$$

where $V_m(t)$ denotes to the membrane potential of the neuron, R_m denotes to the membrane resistance, C_m is the membrane capacitance and $I(t)$ represents the input current source. Figure 3.2 further shows an example of LIF neuron dynamics. The input spikes $\theta_i(t - t_f)$(a spike from i^{th} neuron at time f) is weighted by synaptic weights w_i to generate PSPs. In this case, the input current $I(t)$ is defined as the weighted integration of incoming spikes as follows.

$$I(t) = \sum_{i=1}^{N}(w_i \sum_{k} \theta_i(t - t_f)) \tag{3.2}$$

where N represents the number of synapses connected from input stimulus to LIF neuron. The neuron receives the summed current input and transfer the input current to membrane potential(V_m) according to (3.1) then compare with the threshold V_{th}. Whenever the membrane potential satisfies the criteria of $V_m \geq V_{th}$, the neuron sends out a spike and the membrane potential reset to its resting value(0 in this example).

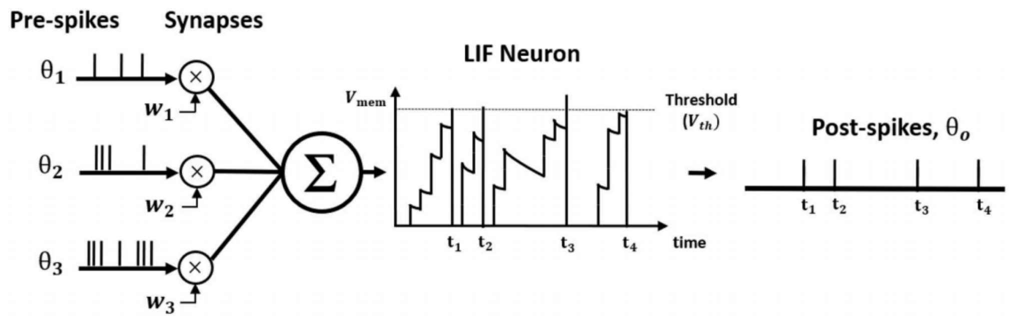


Figure 3.2: The illustration of LIF neuron dynamics [3]

3.2.2 Izhikevich Model

The Izhikevich model (IZ) [135] introduced additional parameters to the differential potential, which creates a 2D ordinary differential equation system. The model combines the original Hodgkin-Huxley dynamics and the computational efficiency of LIF neurons. The differential equation of an IZ model is expressed as:

$$\frac{dV_m}{dt} = 0.04V_m^2 + 5V_m + 140 - u + I(t) \tag{3.3}$$

$$\frac{du}{dt} = a(bV_m - u) \tag{3.4}$$

$$\text{If } V_m \geq 30\text{mV then } v \leftarrow c, u \leftarrow u + d \tag{3.5}$$

where u is the adjusting function, a,b,c,d are the additional hyperparameters to control the functionality of the model. When the V_m reach the limit(30mV) of voltage, then a spike is generated by IZ model, then the V_m and u are reset to its values according to (3.3). The standard factor of a,b,c,d are set as 0.02, 0.2, -65mV and 2 respectively [136].

3.2.3 Spike Response Model

The spike response model (SRM) [132] is a representative spiking neuron model that generalized the LIF and provides a simplified model to simulate the action potential generation process. Just like other neuron models, the spikes are generated when the

PSP above a pre-defined threshold. A significant difference for SRM is using the filters to construct the PSP curve rather than differential equations. The formula for an SRM model membrane potential generation is given as follows.

$$V_{mem}(t) = \eta(t - \hat{t}) + \int_0^\infty \kappa(t - \hat{t}, s) I(t - s) ds \tag{3.6}$$

where $\hat{t}$ is the actual spike firing time. η denotes to the form of the action potential. κ is the linear spike response kernel. The spike is generated if the $V_{mem}(t)$ reach a threshold V_{th} in which the $\hat{t}$ is updated. The main features of an SRM include:

- The threshold of an SRM is dynamic and it depends on the last spiking time. Typically the threshold will increase at the time of firing and decay back to its original value.
- The spike curve η is a function of spiking time. This means it can change along with the spiking activities of the neural processing.
- when the time constant of kernel κ is modelled include single exponential or combinations of exponentials, the SRM has a Hodgkin-Huxley model [124] equivalent behaviour.

3.3 Neural Coding Algorithms

The relation between spike trains and the transmitted information forms the requirements of neural coding. The information propagation in an event-based processing system differs from the conventional system. Thus the outside input stimulus has to be converted to the form of spikes that contains the information. A range of neural coding paradigms has been developed over the past decades. The most common coding strategy is called rate coding [137,138], which the information is coded in the forms of the frequency/rate of spikes over a limited time period. The latency coding mechanisms encode the information into the arrival time of the first spike [139,140]. These two coding methods are commonly known as the temporal coding method, which modulates the timed pattern of spikes to represent corresponding information. Besides, there is also population coding [141] which manages the firing behaviour of a group of neurons to express the information efficiently.

3.3.1 Rate Coding and Spike Count Coding

Rate coding was originally presented by ED Adrian and Y Zotterman [137] with a weight experiment on a muscle. With the weight increases, the number of spikes recorded from muscle nerves also increased. A rate coding model commonly states that as the intensity/magnitude of an input stimulus increases, the firing rate/frequency of the neuron increases. In other words, the value of the input stimulus is regularized to the frequency of the spike trains. Figure 3.3 demonstrates an example of the rate coding model. The blue bar at the left side represents the magnitude of the input stimulus where the higher the input is, the more spikes are generated.

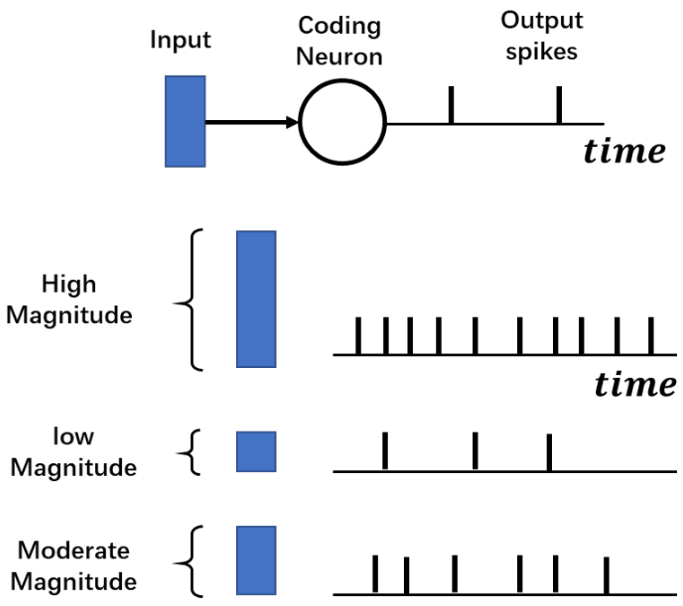


Figure 3.3: Example of neural rate coding model

It should be noted that the rate coding can also be termed as the spike count coding, which defines how many spikes are generated by a stimulus intensity. When a time window is applied, the number of spikes is equivalent to the sense of the frequency of overall spike trains. However, the disadvantages of rate coding are distinct as it only focuses on the magnitude but ignores the temporal structure that encoded in the spike trains. Rate coding is also time consuming since each coding neuron needs to average the temporal spike number at every time step.

3.3.2 Latency Coding

A number of researches have identified that the typical temporal resolution of neural coding is in the millisecond range, which implies that a precise spike timing is significant [142]. The temporal coding can efficiently map the information into the sequence order of spike trains rather than the average firing rate.

As is shown in Figure 3.4, unlike rate coding, a latency coding model transfer the information of stimulus to individual spikes rather than a spike train. Within a certain time window, the magnitude of the stimulus is converted to the precise time of the spikes [140]. For example, a high intensity of stimulus will lead to a low latency of spike generation. In contrast, spike generation is delayed for those input with a low magnitude.

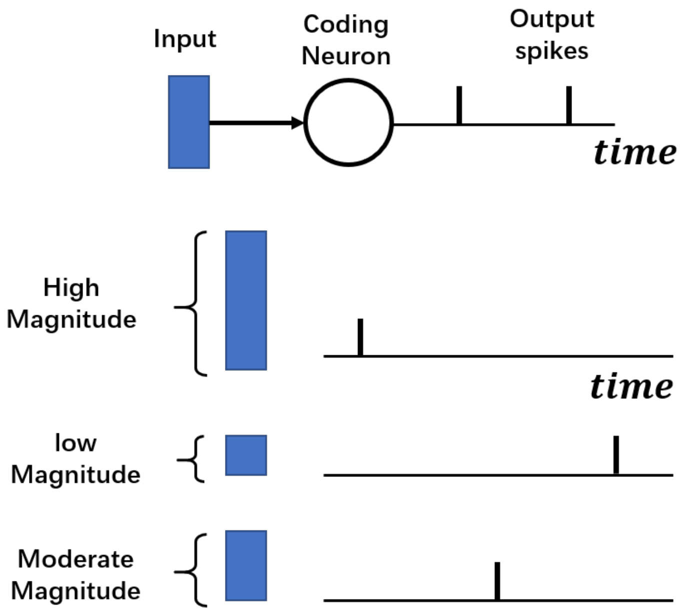


Figure 3.4: Example of latency coding model

The latency coding can express features of spike trains that cannot be described just by the firing rate. For example, time to first spike after the stimulus onset or timed groups of spikes [143].

3.3.3 Neural Coding in Applications

In addition to rate coding and latency coding methods, there are variants of coding scheme including population coding [141], correlation coding [144], sparse coding [145]. Despite the remarkable contributions of these coding methods in the field of neuroscience, it is challenging to map all of the existing neural coding methods to real-world stimuli.

Diehi et al. [146] presented an SNN system to perform a handwritten digit recognition task using the 28x28 greyscale MNIST dataset. The neural coding in this work is modeled using 28x28= 784 neurons to represent each pixel of the image. For each pixel location, a Poisson spike train with firing rates proportional to the corresponding pixel intensity is generated. An example of the coding process for digit 7 in the MNIST dataset is shown in Figure 3.5. With these paradigms, the inked region with higher pixel intensity will result in a high-frequency spike train and the blank region with low pixel intensity will produce no spikes or low-frequency Poisson spike trains.

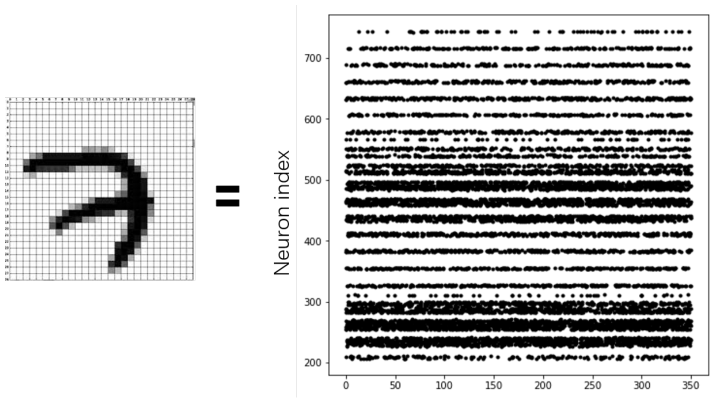


Figure 3.5: Example of image neural coding

Dong et al. [4] proposed an SNN based speech recognition system that utilized the latency coding method to transfer the speech signal to spikes. As is shown in Figure 3.6, the speech signal is firstly transferred to Mel-Frequency Spectral coefficients(MFSCs) matrix by applying mel-scaled filter banks and take the logarithm of the results. Then, the energy components in the MFSC matrix are encoded in the response latency of the time of the first spike to the value onset. With this latency coding, each neuron represents a frequency bands only need to emit a single spike during a presentation of an input sample and all after spikes can be ignored since they are less important than the first spike.

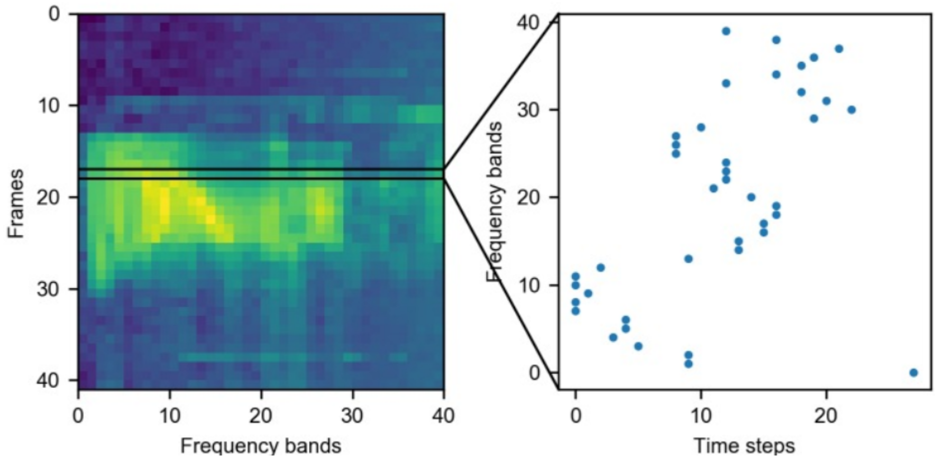


Figure 3.6: Example of speech neural coding [4]

The neural coding is a vital process in the development of event-based algorithms. From the engineering perspective, it provides alternatives to the NM hardware which can transfer the information of output stimulus to an event-based processing system. However, these spiking information is hard to be tracked and analysed. It can be clearly seen that the format of spiking information is fundamentally different from the data captured by conventional sensors. Thus, to efficiently make use of the advancement of spiking data, applicable spiking learning algorithms(section 3.5) is vital for developing an SNN application.

3.4 SNN Architecture

An SNN is typically similar to ANNs in topology but different in neuron activations and information carrier. With the introduced knowledge of neuron models (section 3.2) and input coding methods (section 3.3), SNN can be regarded as a result of the integration of both action potential generation dynamics and network dynamics. Spike trains in an SNN are propagated via synaptic weights which reflects the relevance of the connections between spiking neurons. These synaptic weights as a result are the key learning parameters in SNN as it controls the spike activations and the output target spike train patterns. An example of a standard multilayer spiking neural network is shown in Figure 3.7. It can be seen that the architecture of an SNN is very similar to the conventional ANN. If an event-based input is not available, a coding layer is usually placed at the front of the spiking neuron layers to transfer the outside signals into spike trains.

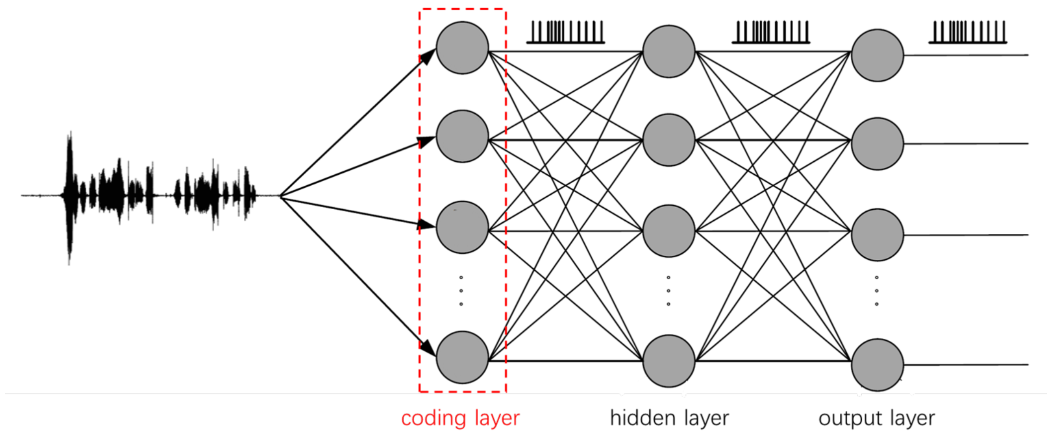


Figure 3.7: Graphical illustration of a feed forward multi-layer SNN structure

The architectures of SNNs are not limited to only the standard feedforward but can be modified by introducing extra signal processing techniques or changing topological neuron connections. Such as spiking CNNs(SCNNs) have been developed by introducing difference-of-Gaussian(DoG) as edge detection and spiking coding methods and several weight-sharing neuron groups(convolution kernel) represents a specific receptive field [147–152]. In addition to SCNNs, many ideas in the conventional ANN domain have been brought into the construction of SNN architecture such as spiking deep belief network [153, 154], Recurrent spiking neural networks [155–157] and liquid state machine based reservoir spiking neural networks [158–160].

3.5 SNN Learning Algorithms

The second generation of ANNs are relying on the gradient descent based backpropagation training algorithm(section 2.3.4), which requires differentiable activation functions to represents the error gradients. However, the backpropagation can not be easily applied to SNN since the discrete asynchronous events produced by spiking neurons are not differentiable.

During the past few years, many works demonstrated successful employment of spike-timing-dependent-plasticity(STDP) in various applications. The STDP technically is able to solve unsupervised clustering tasks [161, 162]. Encouragingly, [163, 164] has shown that the convolutional SNN can be learned in an unsupervised manner using STDPs. More recently, spiking CNNs [165, 166] were proved can be used in the field of frame-level object recognition with a design of identical kernel weight sharing method.

However, Hebbian learning based bio-inspired SNN learning algorithms does not offer the desired accuracy compared with supervised learning algorithms in classification or recognition tasks. Employment of STDP is still experiencing several obstacles as a training method such as the parameters of STDP and spiking neurons have to obey the spike distributions of data. The excitatory and inhibitory strength performed by coupled spiking neurons is really sensitive to the spike timing and weights, which can easily cause neuron domination(not learn) problem.

Currently, the exact learning method for bio-inspired neurons is still remaining as an open question. Recent research has shown various types of synaptic plasticity that enable weight and axon delay learning to be compatible with SNNs [167]. Similar to the learning mechanism described in section 2.3.4, supervised learning and unsupervised learning are the two known types of training strategies. To date, the developed SNN training methods can be categorized as the following 3 types.

- **Conversion from standard ANNs**: Train a conventional ANN with backpropagation and then transfer the weight, input data and neurons to spiking version.
- **Supervised learning with SNNs**: Although the spikes are not differentiable, the backpropagation can still be used by approximating gradient of other dynamics in SNN.
- **Bio-inspired learning rule**: Some biologically realistic synaptic plastic rules can be applied to spiking neuron model which enables learning of SNNs such as STDP.

It should note that most current SNN training algorithms are highly dependent on the preliminaries of the spiking neuron model(section 3.1) and the way of information coupled in spikes (section 3.3) since different spiking processing model can perform different learning behaviour on the spike patterns.

3.5.1 Unsupervised Bio-inspired Learning Rule in SNN

In addition to transfer the learning mechanisms in traditional ANNs (backpropagation and ANN-SNN conversion), a wide range of the neuroscience synaptic plasticity learning mechanisms can be applied into SNN training such as Spike Timing Dependent Plasticity(STDP) [168, 169] or Hebbian learning [170]. The bio-inspired learning rules is very attractive for practical applications since it would allow online learning to be implemented on hardware [171].

Unsupervised learning is implemented only according to local events that do not have any extra information to supervise the SNN outputs. The unsupervised learning in SNN may be constructed by one or a combination of following synaptic plasticity rule.

- A reward or decay of synaptic weights according to the presence of any spiking activity [172].
- A reward or decay of synaptic weights independent of pre-synaptic spikes [173].

- The synaptic weight modulation caused by only pre-synaptic spikes but exclude post-synaptic spikes [174].
- The synaptic weight modulation caused by both pre-synaptic and post-synaptic spiking activities such as Hebbian learning [175].

Although these types of plasticity methods, currently, only the Hebbian term based learning mechanisms are used in SNN, other weight modulation rules are still found to be challenged to be applied to practical SNN training.

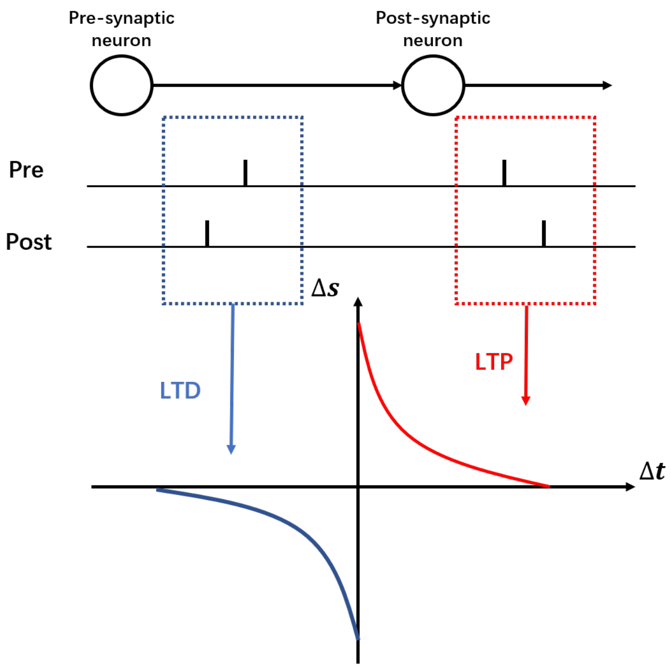


Figure 3.8: An example of working principle of STDP

STDP is an interesting variant of Hebbian learning, which is popularly involved as a part of the unsupervised SNN learning algorithm [176,177]. Figure 3.8 demonstrates an example of the working principle of STDP. When a pre-neuron is sending spikes to post-neuron, the amount of weight modulation Δs is determined based on the actual spike time of the pre-neuron t_{pre} and post-neuron t_{post}. The synaptic weights is strengthened if presynaptic spike comes before a postsynaptic spike $t_{post} - t_{pre} > 0$, this potential increase commonly termed as long-term-potentiation(LTP). In contrast, if $t_{post} - t_{pre} <$

0, the synaptic weights is penalized which refers to long-term depression(LTD). A standard formula of STDP can be written as:

$$\Delta s = \sum_{t_{pre}} \sum_{t_{post}} W(t_{post} - t_{pre}) \tag{3.7}$$

$$W(\Delta t) = \begin{cases} A_{pre} e^{\frac{-\Delta t}{\tau_{pre}}} & \Delta t > 0 \\ A_{post} e^{\frac{\Delta t}{\tau_{post}}} & \Delta t < 0 \end{cases} \tag{3.8}$$

where A_{pre} and A_{post} are the constant learning rate parameters, τ_{pre} and τ_{post} denote to the time constant of LTP and LTD for the temporal STDP learning window [178,179]. The STDP is found can affect neurons' spiking behaviour in response to spike train patterns.

Previous works have demonstrated that repeating temporal patterns can be detected by a single neuron with STDP embedded synaptic plasticity [180, 181]. STDP and its variants can also be tuned to solve complex problems if a precise time reference is coded within a spike train [161]. A significant development of STDP was proposed by Nessler et al. [182]. They approximated the STDP as a stochastic winner-take-all(WTA) circuit [183]. This method is able to estimate an expectation maximization(EM) algorithm to learn parameters for multinomial data distribution.

3.5.2 Training SNN by ANN Conversion

The conversion algorithms were originally developed to process the event-based data generated by NM hardware without SNN. Early researches manually mapped convolutional filters to spike train inputs [184, 185]. The main idea of their work follows the idea of rate coding which translates the information of ANNs into neuron firing rates. The weights are scaled according to the spiking neuron models.

Transferring an ANN to SNN has the advantages of high performance and short development period. The developed ANN frameworks can be used for the design as well as numerous state-of-art ANN architectures [186]. The trained ANN inference model should be converted to SNNs by adapting weights and parameters of the spiking neurons. The method not only needs to map the whole processing system from ANN to SNN but also includes the input and output spike encoding and decoding. The ANN-SNN conversion technique [187] is possible to provide performance guaranteed SNN that can satisfy the expected accuracy.

However, the ANN-SNN conversion also comes with limitations. First, not all ANNs can be easily transferred to SNN. One of the main challenging issues is the negative activations in ANN where the latencies or firing rate of spikes are always positive values. To address this problem, many solutions have been proposed. Such as Prez-Carrasco et al. [185] suggests using a pair of spiking neuron to represents the positive and negative activations.

Second, modern ANN structures have introduced several mathematical operations into network structure which is difficult to be considered used in SNNs. One of the examples is the pooling operation in CNNs(described in section 2.4.2). The max-pooling operation is not a linear function that can be approximated in a spike to spike basis. The average pooling [188] although is a linear operation but it is proved that may degrade the performance of the original work.

Third, the parameters conversion and normalization will bring the additional cost of spikes which is significantly less computationally efficient. This comes to a trade-off between the performance of converted SNN and the computational costs.

3.5.3 Supervised Learning in SNN

The supervised learning reduces the SNN training method back to the methods in traditional ANNs. Many works have been proposed to directly manage the gradient flow through spiking units on the different aspects of spikes. This form of training method does not really aim to mimic the biological plausible learning mechanism but to efficiently adapt SNNs to many problems with state-of-art backpropagation learning rule.

As mentioned in section 2.3.4, supervised learning must rely on the error signal generated by the difference between the label and actual outputs. In the context of event-based SNN, the goal of spiking supervised learning rule is to reduce the error between input and output spike trains. SpikeProp [189] is the first spiking backpropagation based training algorithm. The loss function in SpikeProp looks into the approximation of firing time of neurons rather than non-differentiable spikes. This method showed an example of classification for non-linear data XOR problems with SRM(section 3.2.3) spiking neuron model. The use of the SRM model is a significant progress in the development of backpropagation based learning algorithm. The problem that computing derivatives on specific spikes was avoided since the spiking unit's response can be modeled as PSPs which can be straightforwardly applied to synapses. The limitation of SpikeProp is that the output of SNN has to be strictly constrained to a single spike.

With the appearance of SpikeProp, many variants were developed to solve the limitation of it. Using the same neural model and architecture of SpikeProp but a different choice in formulations of spike coding and loss function [190, 191]. More recent works that focus on the training of synapses to cause output neurons to generate spike trains with expected spike times include ReSuMe(remote supervised learning) [192, 193], Chronotron [194] and SPAN(spike pattern association neuron) [195]. Huh and Sejnowski [196] in 2017 identified the problem that all of the previous gradient descent based spiking learning mechanisms still suffers from a constrain which the number or the precise times of output spikes from SNN has to be satisfied a specific range. They proposed a hard spike threshold function that enables the spike generation process to be continuity by releasing the modeled postsynaptic currents when the PSP approaches the threshold.

Later advanced development of supervised training algorithms in SNN is focusing on the management of alternatives of spike functions. Such as backpropagating the membrane potentials of a spiking neuron. Lee et al. [197] approximated a small signal at every spike time which is applicable for the gradient descent method. Zenke et al. [198] proposed a Superspike surrogate function to the membrane potential generation which can serve as derivatives. Among these methods, it is worth mentioning a state-of-art spiking supervised learning algorithm Spike Layer Error Reassignment(SLAYER) [199]. SLAYER successfully approximates the derivative of the spike function based on the neuron state changes and assigns the error to previous layers. Due to the selection of SLAYER training algorithms for the contributions in Chapter 4 and Chapter 6, a simplified SLAYER training algorithm with its embedded SNN model is provided.

The neuron model used for the SLAYER is the Spike Response Model(SRM). As described in section 3.2.3, the membrane potential generation process of a SRM neuron is achieved by convolving a spike response kernel $\sigma(t)$ with the incoming spike train $s_i(t)$ to this neuron to form a spike response signal as $a(t) = (\sigma(t) * s_i(t))$. Here the index i represents the i_{th} input channel. The spike response signal is further weighted by the synaptic weight w. Similarly, the refractory response signal can be obtained via convolving a refractory kernel $\nu(t)$ with the neuron output spike train $s_o(t)$ as $r(t) = (\nu(t) * s_o(t))$. The overall neuron membrane potential $u(t)$ can be obtained by summing all the spike response signal and refractory response signal as:

$$\begin{aligned} u(t) &= \sum w_i(\sigma(t) * s_i(t)) + (\nu(t) * s_o(t)) \\ &= \mathbf{W}^\top \mathbf{a}(t) + r(t) \end{aligned} \tag{3.9}$$

The bold character $\mathbf{W}$ represents the weight matrix. The generated membrane potential $u(t)$ is then compared with a predefined threshold ϑ and output spike when $u(t) > \vartheta$. In a multilayer feedforward spiking neural network architecture, with n_l layers and synaptic weights $\boldsymbol{W^l}$, the forward propagation can be modelled as 3 part as is shown in (3.10) to (3.12). First, generate the spike response signal for all neurons in layer l based on incoming spikes $s^{(l)}(t)$. Secondly, calculate the membrane potential for each neuron based on the weighted spike response signal $a^{(l)}(t)$ and refractory response signal. Third, generate spike trains by applying thresholing operation on the membrane potential.

$$a^{(l)}(t) = ((\sigma(t) * s^{(l)}(t)) \tag{3.10}$$

$$u^{(l+1)}(t) = \boldsymbol{W^{(l)}} \boldsymbol{a^{(l)}}(t) + (\nu(t) * s^{(l+1)}(t)) \tag{3.11}$$

$$s^{(l+1)}(t) = f_s(u^{(l+1)}(t)) \tag{3.12}$$

Instead of directly managing the non-differentiable spike neuron equations, SLAYER approximates the derivative of the spike function as a probability density function(PDF) of spike state changes. Further details of the model and its use in training the SNN can be found in [199]. With a good estimation PDF as the derivative term of spike change state, the SLAYER can easily derive the gradient of weights and delays in each layer from a feedforward SNN. This allows the network to adapt developed gradient descent method for optimization purposes such as ADAM [91], RmsProp [200].

3.6 Neuromorphic Hardware

During the past decade, there has been a significant exploration of novel devices and technologies as the enablers of brain-inspired computation. Multiple NM systems using SNNs to perform energy efficient computing, which was inspired by the biological working mechanism of human brain. Many researchers and industrial partners have tackled the difficulty of design of NM chips that is scalable in architecture and able to handle various SNN architecture. The most challenging problem in the NM chip design is to handling numerous semiconductor with a great number of artificial neurons and synapses.

In addition to NM computing hardware, the NM sensors are another influential field that attracted increasing attention from both researchers and industries. The NM sensor not only provides a way of acquiring the event-based data to SNNs but also upgraded the sensing techniques beyond the conventional visual/auditory sensors.

In this section, the noticeable NM hardware will be reviewed. The NM chips will be illustrated in section 3.6.3. Section 3.6.1 introduces a popular vision sensor called Dynamic Vision Sensor(DVS) and Section 3.6.2 introduces a type of audio sensor named Dynamic Audio Sensor(DAS).

3.6.1 Dynamic Vision Sensor

The traditional vision sensor is a digital camera that repeatedly refreshes its entire array of pixel values at a predefined frame rate. However, using the digital camera has three drawbacks for dynamic motion recognition. First, a digital camera normally operates with a predefined frame sampling rate(typically range 25-50 frames per second), which limits the temporal resolution of activities observed. Secondly, consecutive frames and redundant pixels in each frame waste significant storage resources and computation. Thirdly, the dynamic range of traditional image sensors is limited by its exposure time and integration capacity. Most cameras suffer from saturating linear response with dynamic range limited to 60-70dB where light from natural scenes can reach approximately 140dB of dynamic range [201].The dynamic vision sensor(DVS) [34,35,202] provides a solution to these problems. The DVS using address event representation(AER) is an event-driven technology based on the human vision system. The benefit of the event-based sensor on dynamic scene recognition task is that it offers very high temporal resolution when a large fraction of scene changes, which can only be matched by a high-speed digital camera with the requirement of high power and a significant amount of resources.

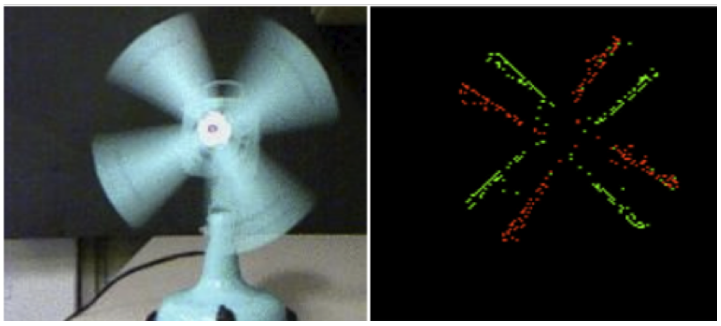


Figure 3.9: Output from (left)conventional sensor, (right)Dynamic vision sensor [5]

Figure 3.9 demonstrates an example of the visualised DVS output with a comparison to the conventional sensor, where DVS can produce crisper edges for rapidly moving objects. This is significant in both the process of object identification and memory saving. In DVS, information is coded and transmitted as electric pulses(or spikes), which is similar to the processing mechanism in biological sensory systems. The output of DVS is generated asynchronously by comparing each activity of a retina pixel with a certain threshold. The emergence of dynamic vision sensor(DVS) demonstrated significant potential in applications of ultra-fast power efficient computing. Compared to traditional vision sensors, DVS returns unsynchronized events rather than sampled time-based frame series [35]. For a given real-world input, DVS records only changes in pixel intensity values and outputs a stream of ON/OFF discrete events regarding to the changing polarity. Such an event-based acquisition mechanism offers many advantages such as low power consumption, less redundant information, low latency and high dynamic range.

With the successful design of DVS, a benchmark was built up for characteristics that NM sensor should deal with and provide guidance for further research in event-based vision sensing. Many enhanced version of DVS is proposed to improve either the spatial resolution or the dynamic range of the devices. Such as Brandli et al. [34] proposed a hybrid method to addressing the frame-based and frame-free visual sensing problem. The DAVIS(Dynamic and Active-Pixel Vision Sensor) integrated both frame-based active pixel sensing and asynchronous DVS sensing in a single camera.

3.6.2 Dynamic Audio Sensor

Just like the DVS mimic the human visual system, the working mechanism of a dynamic audio sensor is inspired by the sensory organ in the human auditory system [203]. The DAS is an asynchronous event-driven silicon cochlea which takes the stereo audio inputs. The DAS uses microphone pre-amplifiers and 64 binaural channels which set up a benchmark in NM audio sensing. The DAS integrates the local digital-to-analog converters(DACs) to allow the quality factors in each channel can be modified. The function of cochlear is modeled by cascaded second-order sections analog components which include half-wave rectifiers, frequency modulator, digital to analog converters and serverl amplifiers and buffers.

An example of binaural DAS output of speech is demonstrated in Figure 3.10 and 3.11. Figure 3.10 demonstrates a DAS response to the speech signal where green and red color are corresponds to the left and right sampling channels respectively and each dot is an acquired event. The chirp response of DAS is shown in Figure 3.11, which the input signal has a dynamic frequency change ranging from 30Hz to 10kHz.

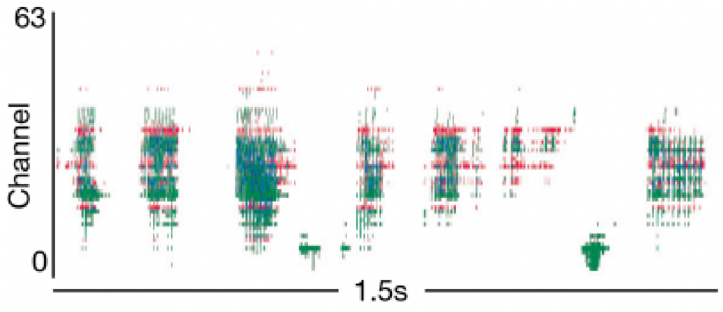


Figure 3.10: DAS output response of speech signal [6]

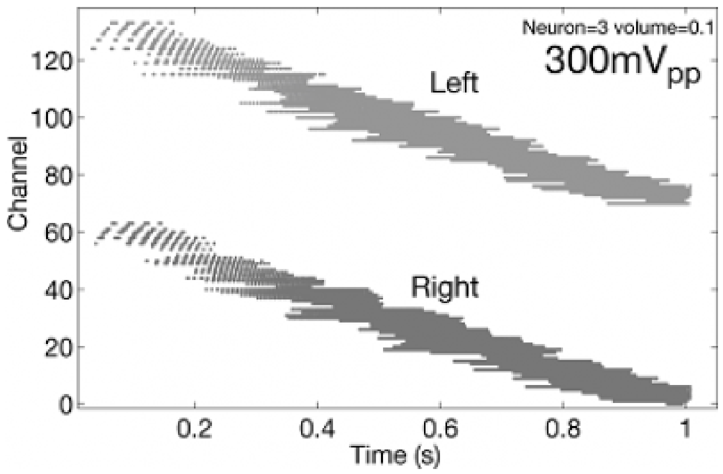


Figure 3.11: DAS output of chirp signal [6]

3.6.3 Neuromorphic Computing Platforms

Initially, NM chips were only investigated by research institutions. As the researchers have shown significant potential of these outstanding brain-like computing models, many big companies have started to involve in the development of NM chips.

IBM company has recently developed the TrueNorth system [33] which is a part of the defense advanced research projects Agency SyNAPSE development program [204]. One single TruNorth chip consists of 4096 computation core, which can achieve the dynamic mapping of neural synapses and neuron arrangement. Each core can maximumly implement 256 IF neurons with 1024 axonal circuits for input connection, which organized as static random access memory crossbar [205]. An attractive feature of the IBM TrueNorth systems is that a single chip consists of 5.4 billion transistors with only 70mW power density consumption, which accounts for only 1/10000 of traditional computing units.

The SpiNNaker NM platform [206] is developed by researchers at the University of Manchester which is a part of the Human Brain Project(HBP) [207] that funded by the European Union. The SpiNNaker provides ASIC solutions to hardware implementations of SNNs. It utilized multiple ARM cores and FPGAs to configure the hardware and PyNN [208] software API to enable the scalability of the platform. The ARM processors allow the platform can configure billions of spiking neurons with a biological realistic connectivity with only 1ms simulation time step. Furthermore, the second generation platform SpiNNaker2 is under the development, which allows the simulation of significantly larger and more complex SNNs with over 10 millions processors [209]. In addition to SpiNNaker, the BrainScaleS [210–212] is also another NM computing platform as a result of the HBP project. BrainScaleS is a mixed signal NM chip developed using waferscale integration technology which allows a utilize of 40 million synapses and up to 180k neurons. The next generation of BarianScaleS is being designed and named BrianScaleS-2 which is capable of using more complex neuron models and simultaneously supports non-linear synapses and customized structure neurons.

Stanford University created two NM hardware designs which are Neurogrid and Braindrop respectively. The neurocore in the Neurogrid [213] is built up by a 256x256 array of fabricated CMOS, which enables a mixed analog-digital implementation of SNNs. Neurogrid is able to provide biological plausible computation with the capacity of millions neuron and billions of synapses. The Braindrop [214] like the Neurogrid is a mixed-signal NM processor but at a high level of abstraction. Braindrop is designed with a 28-nm FDSOI process and integrates 4096 spiking neurons on a single chip which is neuron capacity limited for large scale SNN implementation.

The Loihi NM chip [215] is a digital NM computing platform that was recently announced by Intel. The most attractive feature of Loihi is the potential of chip online learning. Loihi has a special programmable microcode engine for SNN training on the fly. Loihi has 3 unique Lakemont NM cores that are designed specifically to assist with advanced learning rules. There are totally 128 NM cores in a single Loihi chip, which is able to implement 130K LIF neurons and 130M synapses. The maximum size of the Loihi system can supports a scale up to 4096 on-chip cores with 16384 chips [216].

The Brainchip company delivered Akida neuromorphic computing platform [217] which can effectively implement 1.2 million neurons and 10 billion synapses with one NSoC. The platform has several on-board processors includes the functionalities of event-based processing, digital processing, memory, input/output interface and multi-chip expansion.

In addition to these NM chips, there are still many emerging NM chips that demonstrated significant potential in NM computing. Such as the DARWIN chip [218] from Zhejiang University which targeted for embedded low power applications. The DYNAP-SEL [219] developed by researchers at the University of Zurich combines the asynchronous digital logic and analog circuits to achieve analog SNN implementation. The researchers at Tsinghua university successfully design the hybrid Tianjic NM chip [220] which allows both conventional neural network and SNN implementation. The summary of the features of mentioned NM computing platforms is shown in Table 3.1.

Table 3.1: Summary of existing NM platforms

Name	Type	Learning	Simulation time	Capacity	Connection
TrueNorth	Digital	No	Faster than real time	4096 core per chip	AXI bus to SoC
SpiNNaker	Digital	No	Real time	1% of brain capacity	Ethernet
SpiNNaker2	Digital	No	Real time	10M processing	Ethernet
Loihi	Digital	Yes	Faster than real time	4096 core per chip	Ethernet, USB
Neurogrid	Analog-Digital	No	Real time	256x256 CMOS	USB via FX2
Darwin	Digital	No	70MHz clock	2048 neurons per chip	UART to USB
BranScaleS	Analog, Digital	No	Slower than real time	180K neurons	Ethernet
Dynap-SEL	Analog, Digital	No	Real time	512 neurons per chip	FPGA
Braindrop	Analog, Digital	No	Real time	4096 neuron per chip	Not specified
Akida	Analog, Digital	No	Not specified	1.2M neuron	USB/UART/PCI
Tianjic	Digital	No	Real time	40k neuron per chip	Not specified

3.7 Spiking Neural Network Software

Despite the advantages of SNNs, the computational problems in terms of simulating spiking neurons are relatively large. In some cases, the detailed differential representation of biophysical spiking neurons is required like the IZ neuron model. On the other hand, the simplified neuron model which does not need to reconstruct the biological spiking generation mechanism realistically can be simulated easier (i.e. IF neuron model) [221]. The simulation strategies of an SNN can be divided into two families: synchronously or asynchronously. The synchronous algorithms update all the neurons at every time steps which causes higher computational resources than asynchronous or "event-driven" algorithms. The asynchronous method only updates the neuron status when they receive or send out spikes just like the working paradigm as a DVS sensor.

Unlike the unified neural network frameworks in ANN(such as tensorflow [222] and Pytorch [204]), neither the SNN model nor the training method for SNNs is generalized. The way of simulating the SNNs still remains diverse and objectively oriented. Currently, the process of design an SNN is not only considering the feasibility of the network itself but also can extend to features like biological plausible, computational cost and learning mechanisms. To comprehensively review the software implementations of SNN, the noticeable SNN simulators are summarised as Table 3.2, which includes key features of these emerging software.

Table 3.2: Summary of SNN simulators

Name	**Enviorment**	**Hardware**	**Synaptic Plasticity**	**Synchrony**	**GPU support**	**Feature**
BRIAN [223]	Python,C++	N/A	Hebbian Synaptic Plasticity	syn & asyn	No	Biological Plasible
NEST	Python/nest	N/A	short-term plasticity, STDP	syn	No	Biological Plasible
Spyketorch [224]	Python /MATLAB	N/A	STDP. R-STDP	syn	YES	Convolutional operation and Pooling operation avaliable
NeuronFlow [225]	Python	FPGAs	pair-based nearest neighbor STDP	syn	No	Ability of mapping FPGAs
PyNN [208]	Python	SpiNNaker & BainScaleS	STDP variants	syn	Yes	Hardware applicable
Nengo [226]	Python,C++	SpiNNaker, Loihi, FPGA	STDP and backpropagation	syn & asyn	Yes	Hardware friendly; Supports deep learning and conversion
SLAYER [199]	Python, C++	Loihi	Supervised backpropagation	syn	Yes	Convolution and pooling supervised learning
NeuCube [158]	Pythom, Matlab, JAVA, C++	SpiNNaker	reservoir	syn	Yes	end to end module to address applications
SNN toolbox [187]	Python	N/A	N/A	syn	N/A	ANN-SNN conversion toolbox
BindsNET [227]	Python, C++	N/A	machine learning, reinforcement learning, Reservoir	syn	YES	Programming flexible
Brian2GeNN [228]	Python,C++	DSP,FPGA	Hebbian Synaptic Plasticity	syn	Yes	BRIAN with GPU support

3.8 Neuromorphic Technology in Human Gesture Recognition

Recently, using neuromorphic technology for orientated engineering applications have attracted increasing attention. Due to the availability of visual and audio NM sensors, various SNN architecture and learning rules, researchers have successfully applied the NM in many fields. Human gesture recognition as described in section 1.1 is an inter-discipline problem which is well suited to the event-based processing such as for those scenarios that require SWaP profiles(UAV, IoT, robots). With the described NM related techniques in section 3.1 to 3.7, an extensive review of the noticeable works is provided in Table 3.3.

Table 3.3: Summary of previous works on NM human gesture recognition

Application	SNN type	Learning Rule	Dataset	Accuracy	NM Hardware	Feature	NM Sensor
Hand Gesture Recognition [32]	CNN-SNN conversion	Backpropagation	DVSGesture 128	89.64%-10class	Loihi	End-End Recognition	DVS
Hand Gesture Recognition [36]	CNN with filters and weight quantization	Backpropagation	DVSGesture 128	96.49%-10class	TrueNorth	Ultra low power	DVS
Hand Gesture Recognition [199]	Spiking CNN	Spiking Bachpropagation	DVSGesture 128	93.64(+-0.49%)-11class	N/A	Pure Spiking	DVS
Hand Gesture Recognition [229]	Spiking CRNN	Spiking Bachpropagation	DVSGesture 128	96.59%-10class	N/A	CRNN structure	DVS
Early Prediction in Human Robot Collaboration [230]	Turn-taking SNN	STDP	self collected	N/A	N/A	ability of early prediction	N/A
Spoken Digit Classification [231]	SNN-SOM	SOM and Tempotron learning	TIDIGIT	97.40%	N/A	The use of SOM	N/A
Head Pose Estimation [232]	Objective Oriented	N/A	self collected	N/A	Loihi	No training	N/A
Human action recognition [233]	Reservoir	STDP	UCF101	81.30%	N/A	Reservoir based learning	N/A
Human Motion Recognition [234]	Feedforward	Backpropagation	MSR-Action3D	82%	N/A	3D coordinates data	N/A
EMH signal Hand Movements Classification [235]	ANN-SNN conversion	Backpropagation	limb position sEMG dataset	89.79%	N/A	EMG classification	N/A
Speech Recognition [4]	MFSC + Convolution SNN	STDP	TIDIGITS and TIMIT	97.50%	N/A	Robust in speech recognition	N/A
Speech Recognition [236]	Constrained CNN-SNN conversion	Backpropagation	self collected	92.21%	N/A	Novel auditory sensor	FPGA DAS
Speech Recognition [237]	STFT+SOM+ SNN	Tempotron learning	TIDIGITS	97.60%	N/A	The use of SOM	N/A
Speech Recognition [238]	Reservoir	STDP	Emo-DB	82.35%	N/A	N/A	N/A
Speech Emotion Recognition [239]	Convolutional SNN	STDP	RAVDESS(only 6/8 class used) eNTERFace	80.30%	N/A	Multi-model	N/A
EEG Emotion Recognition [240]	wavelet + FFT +NeuCube	Hebbian Learning	SEED	96.67%	N/A	EEG emotion recognition	N/A
Text Emotion Recognition [241]	CNN-SNN conversion	Backpropagation	self collected	N/A	TrueNorth	NM text processing	N/A

It can be seen that the SNN has been applied to HCI human gesture recognition tasks with or without hardware. Due to the event address representation of the input data, researchers in the NM visual-based domain mainly focus on the tasks with the input has a characteristic of the dynamic scene such as action recognition, hand gesture recognition, action recognition and motion recognition. For the audio domain, the speech-related applications were extensively investigated.

The IBM DVSGesture 128 dataset [36] which was collected using a 128X128 resolution of DVS is one that most popular used for visual-based hand gesture recognition applications. Researchers from IBM created an offline inference model which trained by the dataset and successfully transferred the model to the TrueNorth chip, which delivered a real-time edge neuromorphic computing example for hand gesture recognition. Similarly, Massa et al. [32] successfully implemented an efficient SNN on the Loihi chip using the CNN-SNN conversion technique. In addition to conversion techniques, Shrestha et al. [199] using a spiking backpropagation equivalent method SLAYER delivered a nine layer convolutional SNN for the DVSgesture dataset, which enables completely event-based processing without and preprocessing of the data. The work in [233] presented a reservoir based SNN trained with STDP that can effectively handle the challenge of visual-based action recognition on UCF101 dataset [242]. This essentially demonstrates the ability of bio-inspired unsupervised learning on large scale gesture recognition dataset.

Apart from the visual side, a variety of SNN solutions were proposed to address the audio-based gesture recognition problem. However, unlike the vision data, the information in the audio signals are difficult to be extracted by the network along without prior information to the frequency components. This causes most of the researches to employ additional preprocessing techniques to raw audio signals. Wu et al. [231] suggested an SNN based spoken digit classification strategy using the frequency band energies along with the self-organizing map to extract time-frequency features in the speech signal. It utilizes the tempotron supervised learning rule for a feedforward fully connected SNN as the classifier which offered a 97.4% accuracy for TIDIGIT dataset [243]. Additionally, Dong et al. [4] developed an event-based speech recognition system using the

Mel-Frequency-Spectral-Coefficient (MFSC) as the feature extractor. The MFSCs were treated as frames and processed by a convolutional SNN with STDP as the learning rule. A speech emotion recognition(SER) model was developed in [239] , which employs Mel-frequency cepstral coefficients(MFCC) as the pre feature extractor and a convolutional SNN as the main spiking unit. The method provides a recognition accuracy of 80.30% for 6 class of different emotion gestures on the RAVDESS dataset.

Other noticeable works that using input signals from different sensors is described as follows. An emotional recognition SNN model using Electroencephalogram(EEG) signals was presented in [240]. The model combines the continuous wavelet transform and NeuCube reservoir based SNN together, which offers an accuracy of 96.67% on the SEED dataset. Diehl et al. [241] proposed an interesting CNN-SNN based model 'True-Happiness' that can predict the emotion from written text. The model was successfully implemented on TrueNorth NM computing platforms.

3.9 Conclusion

In this chapter, the fundamentals of NM technology and SNN working principles are presented. The SNN is a key element in a neuromorphic system that receives, processes, and outputs the event-based information. The emerging neuromorphic sensors and computing platform allow SNNs can maximize the potential of SNNs in terms of energy efficient and real-world performance. The chapter forms up the fundamentals of original contributions in SNN algorithms design in chapter 4, chapter 5 and chapter 6.

Despite the advantages of NM technology, there are still many challenges in the NM computing such as limited accuracy compared to traditional ANN, the difficulty of large-scale network implementations, lack of reliable supervise training algorithms, difficult to train with bio-inspired synaptic plasticity. There is still no universal frameworks that either for SNN algorithm design or NM circuit layout.

Chapter 4

A Novel Spiking Convolutional Recurrent Neural Network(SCRNN) Structure

4.1 Introduction

This chapter presents a novel spiking neural network structure that can adapt to neuromorphic vision data based recognition problem especially for those data that contains strong spatiotemporal correlations such as visual based human gesture recognition. The convolutional operation and recurrent neural network connections are combined in a SNN that uses a supervised learning based spiking convolutional recurrent neural network(SCRNN). By adjusting the integration period of input data sequence and convolution kernel, SCRNN can achieve arbitrary spatio-temporal resolution according to the recognition demand. Besides, The Spike Layer Error Reassignment (SLAYER) training algorithm as described in section 3.5.3 is successfully deployed to the SCRNN for the purpose of generalization and training stability. The use of SLAYER effectively prevents the common gradient vanishing and explosion problem associated with recurrent neural networks. Since the recurrent propagation between the SCRNN cells rely on the information fusion from inputs of current timestamps and output from previous timestamps. Particularly for SCRNN, a spiking feature map integration method is developed in the SCRNN cell to maintain information continuity in temporal domain.

To validate the robustness of the SCRNN, the network structure is evaluated by performing the recognition task on the IBM DVS gesture dataset [36]. The experimental result of action recognition using SCRNN will is presented in section 4.8 with discussions.

4.2 3D spiking Convolution Operation

Consider an input sequence $S(n) =, n = 0, 1, 2, ...N$ as is illustrated in Figure 4.1. At each time step, $S(n)$ is a 3D tensor with shape $\{u, v, t\}$ where u and v denote the width and height of each frame and t correspond to the pre-defined time resolution. For a given event based video stream, it can be arbitrarily segmented into several tensors according to the desired temporal frequency. For example, for a 1.5sec 128x128 resolution events data stream with 30ms temporal resolution and 1ms sampling time can form a input sequence $S(n), n = 0, 1, 2, ...50$. For each segment, the tensor shape is $\{128, 128, 30\}$.

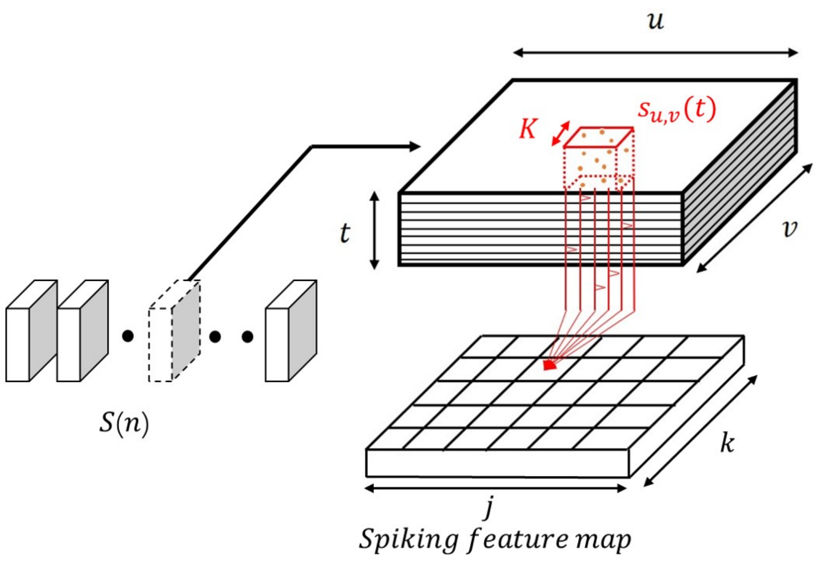


Figure 4.1: The 3D spiking convolutional operation

The sampled input tensor $S(n)$ with shape of $\{u, v, t\}$ is convolved with a 3D convolutional kernel to generate a spiking neuronal feature map. The spikes within a arbitrary kernel can be regarded as a bunch of spike trains $s_{u,v}(t)$ where each spike train corresponds to the spikes at a specific coordinate (u, v) within the temporal resolution window t. Each neuron in the feature map receives the spikes from the neurons in the 3D convolutional kernel. The spikes in the region of the kernel are integrated to generate membrane potential for a single neuron in the feature map. The neurons in a map detect the spatio-temporal dynamic patterns in different 3D volumes. Unlike the standard feature map generated by CNN, the information at each coordinate in a spiking feature map is expressed by spike trains which is a spiking representation of detected patterns.

The convolutional kernel is highly overlapped to ensure the proper detection of features. The SRM neuron model is used to describe the 3D spiking convolution operation, which gathers all the input spikes from pre-synaptic neurons and outputs spike when the membrane potential reaches the pre-defined threshold. In the SLAYER, this is done by convolving the spike trains in the kernel with spike response kernel and followed by the threshold function. Each spike train will be transferred to the spike response signal then further to the membrane potential of the postsynaptic neuron. The process can be expressed as:

$$a_{u,v}(t) = s_{u,v}(t) * \sigma(t) \tag{4.1}$$

$$u_{j,k}(t) = \sum_{m=1}^{K}\sum_{n=1}^{K} \mathbf{W}_{m,n} a_{j+m-1,k+n-1}(t) + (s_{j,k}(t) * \nu(t)) \tag{4.2}$$

$$s_{j,k}(t) = 1 \;\;\&\;\; u_{j,k}(t) = 0 \quad \text{when } u_{j,k}(t) \geq V_{thr} \tag{4.3}$$

where $\mathbf{W}$ denotes the synaptic weights. u and v are the vertical and horizontal coordinate index of the input tensor. j and k represent the vertical and horizontal coordinates in the feature map. K represents the convolution kernel width and height.

The 3D spiking convolution can decompose the input event based data into several spatio-temporal pattern feature maps, where each spike in the map corresponds to a specific pattern. When multiple spiking convolution layers are used, the feature in a layer is a combination of several low level features extracted from the previous layer.

4.3 The SCRNN cell

The SCRNN cell is designed as the fundamental unit of the SCRNN system. The idea was inspired by the structure of ConvLSTM cell as is introduced in section 2.6. A graphical illustration of a single SCRNN cell is shown in Figure 4.2.

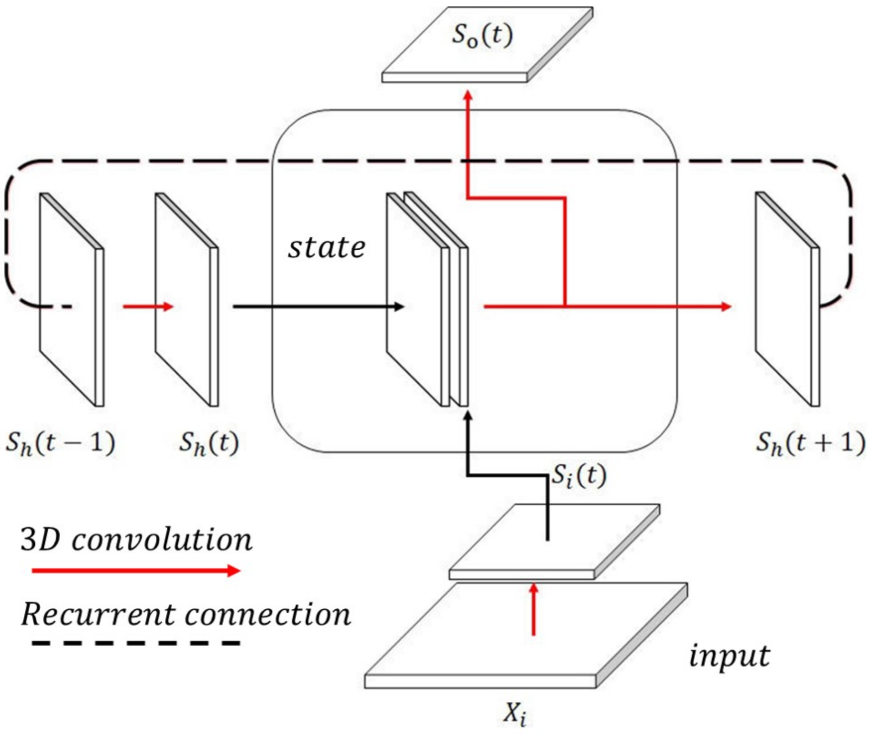


Figure 4.2: The proposed single SCRNN cell. The state spiking feature map and input feature map are combined in the cell with an output feature map recurrently connected to the cell

The inputs to the cell comprises two parts: First is the spiking feature map generated by the outside events(e.g. a fragment from a event-based action data). The second part is the hidden spiking states which represent the fused feature map of previous states and the feature map generated by current input. To ensure the state feature map has the same shape as the input, a padding technique is needed before the actual convolution operation, which means padding empty events(zeros) on the boundary of state maps. This can be viewed as the current state having no prior knowledge in terms of the region outside the current receptive field. At zero time index, the internal state needs to be initialized randomly or set empty which represents no prior knowledge at the beginning from the temporal perspective. Consequently the 3D spiking convolution operation is applied on both input-to-internal state transitions and state-to-state transitions in a SCRNN cell. The future state to state transition is achieved by utilizing another 3D convolution layer that contains a pre-defined number of hidden neurons. Two feature maps are concatenated to form a single map. Then the spikes in the same kernel of the fusion map are accumulated and activated to generate the membrane potential signal for future states. Consider an input segment X_i. The entire computation process within a SCRNN cell can be written as:

$$s_i(t) = \theta\{\sum W_{ih}(X_i * \sigma(t))\} \tag{4.4}$$

$$s_h(t) = \theta\{\sum W_{hi}(s_h(t-1) * \sigma(t))\} \tag{4.5}$$

$$s_h(t+1) = \theta\{\sum W_{hh}(s_i(t) * \sigma(t) + s_h(t) * \sigma(t))\} \tag{4.6}$$

$$s_o(t) = \theta\{\sum W_{ho}(s_i(t) * \sigma(t) + s_h(t) * \sigma(t))\} \tag{4.7}$$

where θ represents the thresholding operation. W_{ih}, W_{hi}, W_{hh} and W_{ho} denotes the weight input to state, state to input, state to state and state to output respectively. It can be seen from (4.6) and (4.7) that the output of a SCRNN cell comprises two terms: $s_h(t+1)$ is the spiking states that can be used for future cells and the $s_o(t)$ represents the output spike train. The output from the cell represents the 3D feature map extracted from the current cell that allows the network to go deeper by using the $s_o(t)$ as the input of next layer.

4.4 The Spiking Convolution Recurrent Neural Network Architecture

The overall SCRNN architecture shown in Figure 4.3 comprises a combination of single cells that are stacked in both temporal and spatial processing domain. From a temporal point of view, the cells can process the input sequence separately using the internal state correlations. Furthermore, the input can be further decomposed by adding additional cells at each time step, thus allowing the network to form greater computational complexity and processing higher level spatial features. In other words, at a specific time step, the concatenated SCRNN cells(layers) can be treated as a standard spiking convolutional neural network wherein each input of a SCRNN cell is the output signal of previous cell. It should be noted that additional initial states are needed for every added layer.

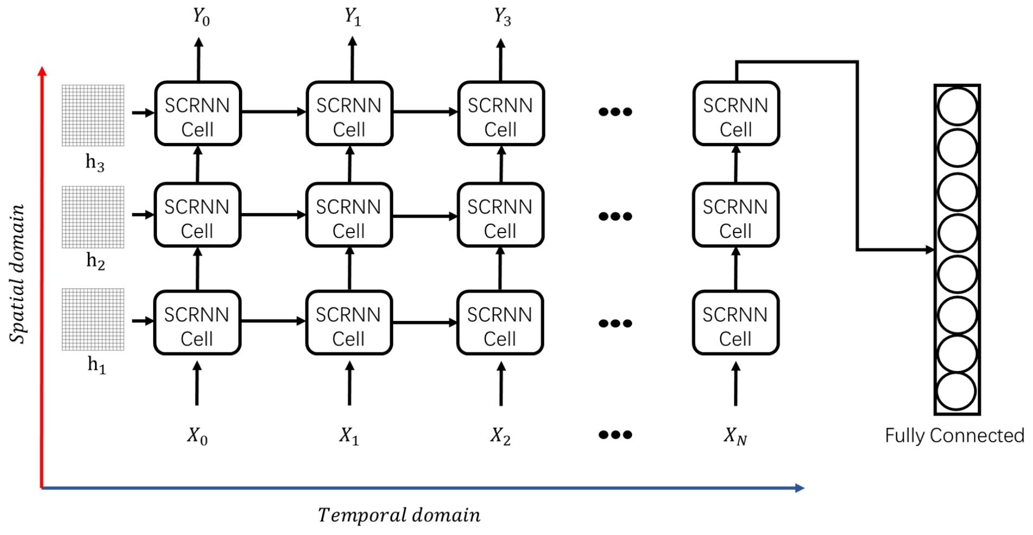


Figure 4.3: The proposed SCRNN structure which is comprised by prior defined individual SCRNN cells

Similarly to the conventional recurrent neural network, the SCRNN can also be unrolled to form a short-term feed-forward structure that increases the network parameter capacity. Unrolling a recurrent structure represents a trade off between the network performance and the computational cost. Although theoretically the cells can be unrolled up to the length of the input sequence, the computation cost in the training process increases dramatically along with the number of cells. Moreover, to guarantee the network performance in terms of temporal information, the backpropagation through time(BPTT) as described in section 2.3.4 is used which is another factor that affects the training speed. BPTT calculates and accumulates errors across each time step, which can be computationally expensive as the number of time step increases.

4.5 DVS Gesture Dataset

The DVS gesture dataset comprises of recordings of 29 different actors carrying out 10 different hand gesture actions. All recordings are captured by an Inilabs 128 x 128 dynamic vision sensor under three different lighting conditions. Each gesture sample has a duration of approximately 6 second. Figure 4.4 shows an example of hand waving gesture with 0.5s integral time interval in nature light condition. The goal is to classify the gesture event video data into correspond label. The DVS gesture dataset is split as 1176 samples for training and 288 samples for testing as annotated. Note that the amount of samples of arm roll is twice than other gestures in the original dataset.

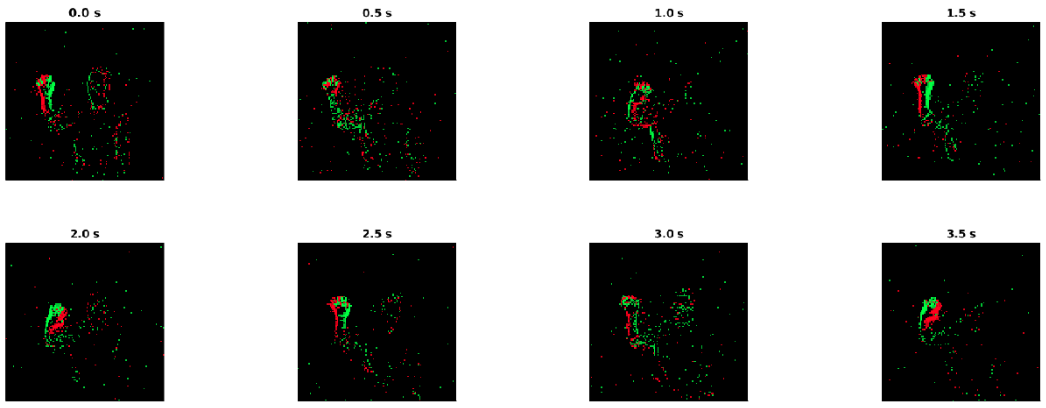


Figure 4.4: The demonstration of DVS gesture dataset with integral time of 0.5s. The gesture showing in the example is hand waving. The green and red edges in each Figure 5 represents the ON/OFF polarities of spikes.

4.6 SCRNN Setup

A 3 layer SCRNN was constructed to solve this problem as is shown in Figure 4.5. The network uses a standard SRM response neuron that introduced in section 3.2, the detailed neuron parameters are shown in Table 4.1. The parameters define the standard neuron dynamics behavior which is used in all SCRNN networks. Where ϑ_{neuron} is the neuron firing threshold. τ_{neuron} is the neuron time constant, τ_{ref} is the neuron refractory time constant, C_{ref} is the refractory response scaling coefficient, τ_f is the neuron spike function derivative time constant, and the C_f is the neuron spike function derivative scaling coefficient.

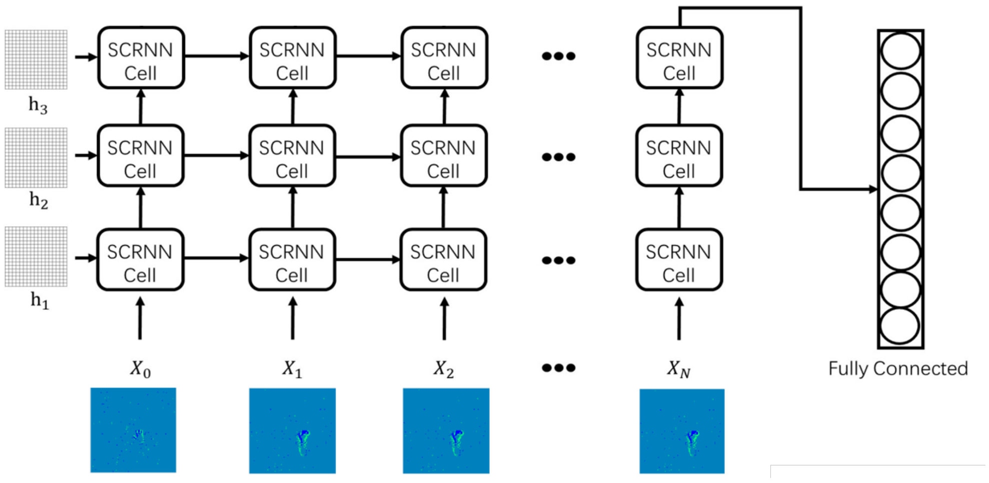


Figure 4.5: The SCRNN structure that used for IBM DVS gesture dataset hand gesture recognition.

Table 4.1: The neuron parameter setting for the SCRNN simulation.

ϑ_{neuron}	τ_{neuron}	τ_{ref}	C_{ref}	τ_f	C_f
5	10	1	2	1	1

As the gesture recognition is a many-to-one problem(described in section 2.5.2), only the output from last layer and last time step SCRNN cell is taken into account for the loss calculation. The loss function used in this method is defined as the square error based on the number of spikes between the target and actual output in a time window according to [199]. With the S_o as the output spike train of the last layer of SCRNN and $\hat{S}$ denotes to the target spike train, the loss function L can be expressed as follows.

$$L = \frac{1}{2}\sum_{1}^{N}\left(\int S_o(\tau)d\tau - \int \hat{S}(\tau)d\tau\right)^2 \tag{4.8}$$

where N is the number of output neuron of the last layer. At each time step, the error signal is calculated according to the current output spike count and target spike count. It should be noted that the backpropagation pipeline covers both spatial and temporal propagating routes through the recurrent connection. To save on computation resources, only 1s out of 6s of each gesture samples were used for the experiment. The input event sequence is integrated to several frames based on pre-defined segmentation length l_s. The segmentation length significantly affects the sparsity and the number of integrated frames. A small l_s will results in a large number of sparse frames, on the contrary a chosen of large l_s will reduce the amount of frames but increase the number of events in each frame.

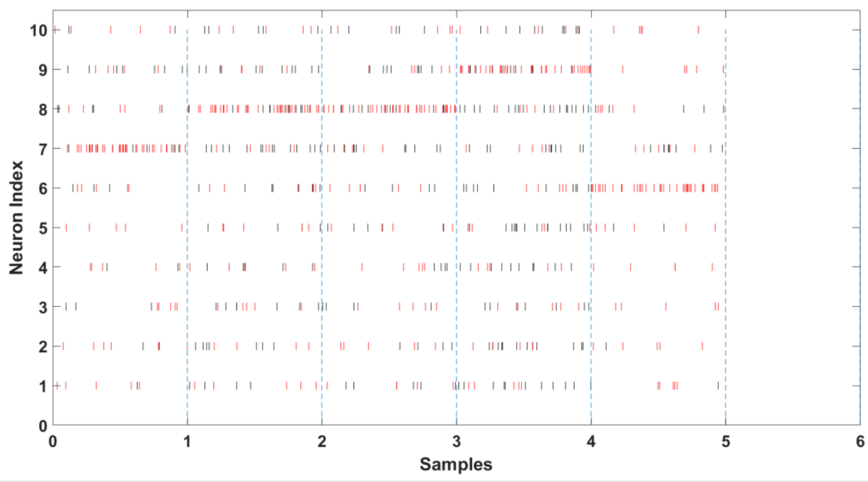
4.7 SCRNN Output Activities

To evaluate the performance of SCRNN, different combinations of network parameters are carried out to perform the action recognition task. The following hyper-parameters were used in the experiments: Number of filters in convolutional layer, the segmentation length(time resolution)l_s, the target true spike count Tg_{True} and target false spike count Tg_{False}. Figure 4.6 illustrates the output spike activities before and after training of the last layer of the SCRNN. The vertical dash line in the figures simulates the time window that spikes will be counted for a input sample. In other words, the spikes between two dash lines are the output from a single input instance. The output neuron index from 1 to 10 represents 10 different gesture classes. The red bars are target spike(labels) and black bars are actual network output spikes. It should be noted that the loss for the SLAYER training algorithms is calculated from the error signal that was generated according to the difference between the number of actual output spikes from the network and the target spikes(Tg_{True} and Tg_{False}). If the actual spikes count of output neuron match that from the target spike count then a correct prediction is implied. As shown in Figure 4.6(a) the SCRNN have zero output before training and gradually learns to generate spikes that match the target spike in terms of the target spike quantity. Figure 4.6(b) demonstrates the output spike monitoring after-training the SCRNN. It can be clearly seen from Figure 6(b) that the actual spikes(shown in black) now have similar spike counts as target spikes(shown in red) for the input samples. It should be noted that, the target spikes and actual spikes have different spike timings but similar spike counts in each window.

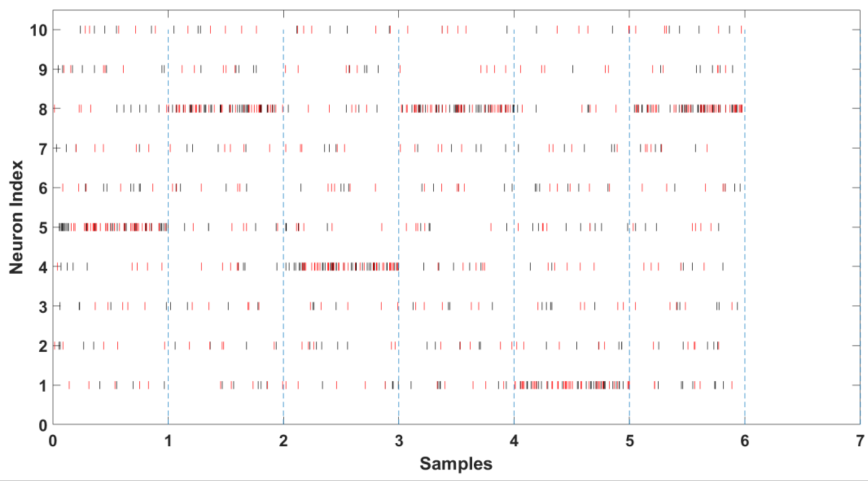
4.8 SCRNN Experimental on DVS gesture Dataset

The experiment results are shown in Table 4.2, where each listed architecture is simulated for 100 epoch over the full dataset. For each structure listed in the table, the accuracy is obtained by averaging the best testing accuracy among the 5 repeated experiments with different randomly initialized weights. Among these experiments, the best testing accuracy of 10 classes gesture is 96.59% with the 3 layer SCRNN structure with the first convolutional layer consisted of 32 5x5 convolutional filter, second and third convolution layer has 64 and 128 3x3 convolutional kernel respectively. The l_s is 50ms which represents there are total of $1000/50 = 20$ time steps. The loss and training curve for the best network structure is shown in Figure 4.7(a) and Figure 4.7(b). This structure also was used to train the 11 classes gesture (plus a random other gesture action) and obtained a testing accuracy of 90.28%.

The loss can be very large at the start compared with normal loss value since the network can have empty output with untrained weights and delays. It was found that setting the $l_s = 50ms$ produces the best result for SCRNN structure which can be explained as follows. First, the time resolution is matched with the frame continuity for this dataset, which means the. to the training process. A proper selection of l_s can make sure of the sparsity of frames which guaranteed the stability of the training



(a)



(b)

Figure 4.6: A demonstration of the last layer SCRNN output: (a)Before Training (b)After Training

process.

The confusion matrix in Figure 4.7(c) shows a detailed performance of the SCRNN for the 10 gestures recognition tasks. Note that the amount of samples of arm roll is twice than other gestures in the original dataset. It can be seen that the SCRNN achieved an overall good performance except that the confusion between the hand clapping and air drums gesture where there are totally $3+4=7$ instances that SCRNN misclassified the hand clapping or air drum as each other. This is due to the dynamic

Table 4.2: Comparisons of SCRNNs performance on DVS gesture dataset with different hyper-parameters. Tg_{Ture}: The preliminary setting of target True spikes count; Tg_{Ture}: The preliminary setting of target False spike count; $l_s(ms)$: The segmentation length(time resolution)

$Conv1$	$Conv2$	$Conv3$	$FC1$	$FC2$	Tg_{Ture}	Tg_{False}	$l_s(ms)$	$Trainacc$	$Testacc$
5x5x16	3x3x32	3x3x64	1024	512	30	5	25	91.67%	85.23%
3x3x16	3x3x32	3x3x64	512	128	30	5	25	88.64%	88.64%
5x5x32	3x3x64	3x3x128	1024	512	30	5	25	94.32%	89.15%
5x5x16	3x3x32	3x3x64	1024	512	60	10	50	95.45%	91.67%
3x3x16	3x3x32	3x3x64	512	128	60	10	50	95.08%	86.37%
5x5x32	3x3x64	3x3x128	1024	512	60	10	50	98.48%	**96.59%**
5x5x16	3x3x32	3x3x64	1024	512	80	15	75	95.45%	88.64%
3x3x16	3x3x32	3x3x64	512	128	80	15	75	93.18%	81.06%
5x5x32	3x3x64	3x3x128	1024	512	80	15	75	96.59%	90.9%

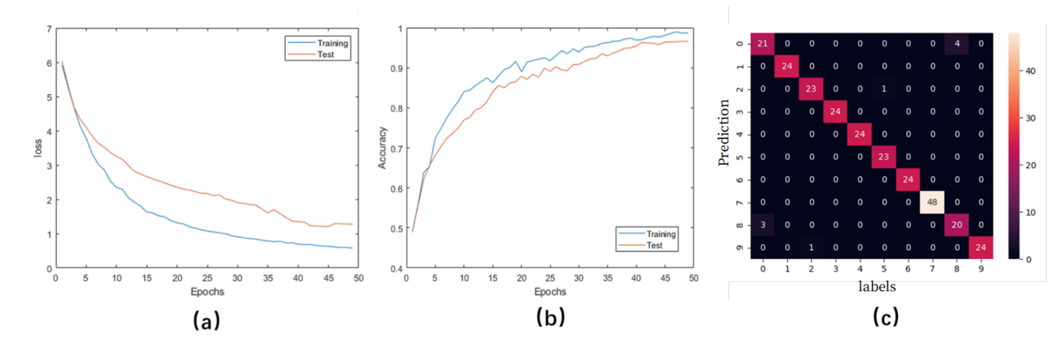


Figure 4.7: (a): The training and testing loss changes for 3 layer SCRNN; l_s=50ms (b):The training and testing accuracy changes for 3 layer SCRNN; l_s=50ms; (c): The confusion matrix for 3 layer SCRNN; l_s=50ms;

similarity of these two gestures for some instances.

In order to identify the reason why the SCRNN performs commonly misclassification on the hand clapping and drum gesture dataset. Deeper analysis of these two gestures was conducted. Figure 4.8 demonstrates an example of misclassification which shows both 3D and 2D view of dynamics of these two gestures. From our observations, some of hand clapping and air drum gestures exhibit strong similar spike change pattern which is a potential reason that leads to misclassification. This further matches our initial design purpose of SCRNN, which is an action dynamics sensitive, event stream pattern based recognition network.

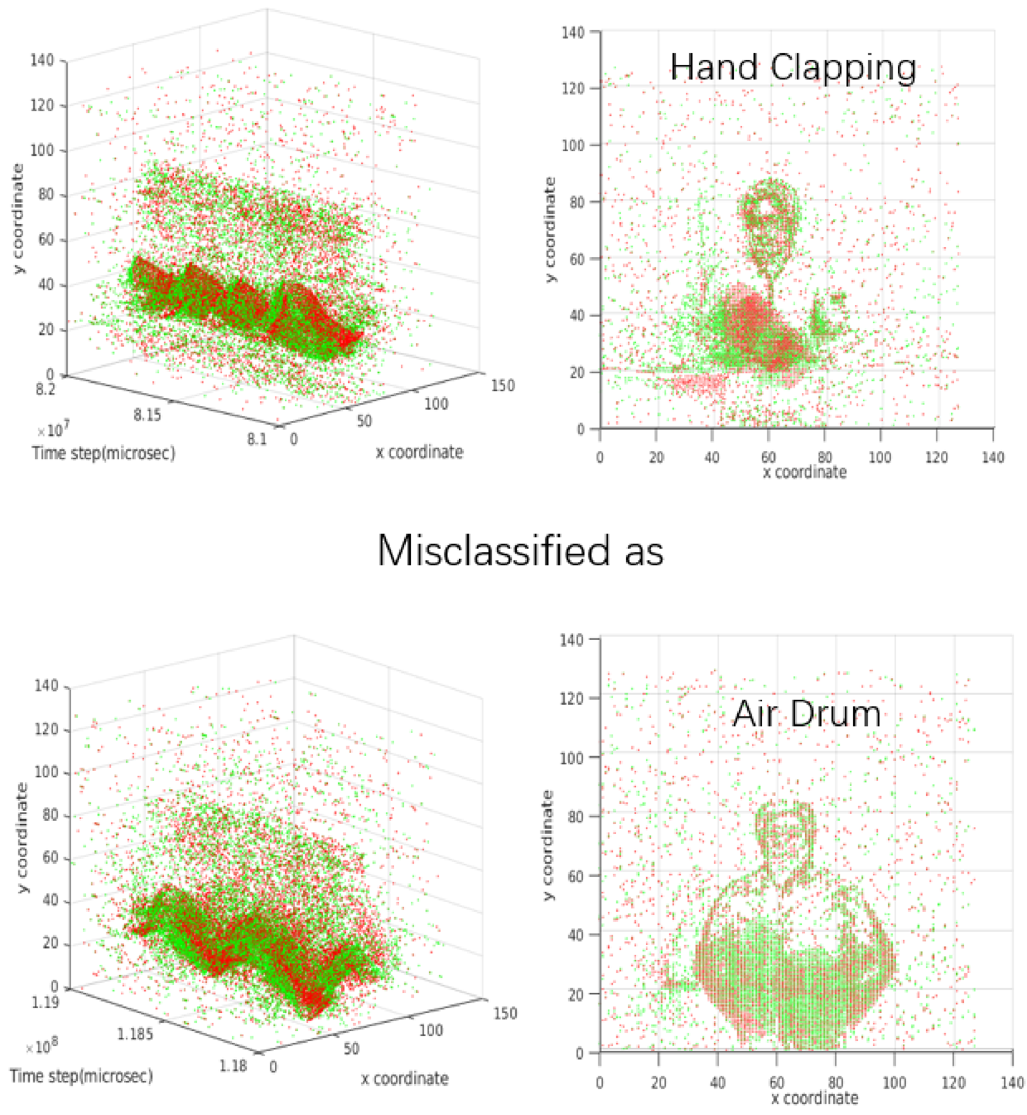


Figure 4.8: The example of 3 layer SCRNN misclassification case

For comparison purposes, results from previous published works on IBM DVS gesture dataset is carried out which is shown in Table 4.3. It can be seen that the SCRNN approaches the state of the art recognition accuracy and surpassing the benchmark accuracy of the IBM's work in 10 categories gesture classification task. The original work from IBM that running on TrueNorth was trained with Eedn [36] and requires extra filters and preprocessing before the CNN. On the other hand the SCRNN take the neuromorphic data directly from the sensor and the training process do not require any

Table 4.3: Comparison of SCRNN gesture recognition results with previous works

Method	Type of processing	10 class	11 class
IBM TrueNorth Eedn [36]	spiking	96.49%	94.59%
SLAYER CNN [199]	spiking	unknown	93.64% $\pm$ 0.49%
PointNet++ [244]	Non-spiking	97.08%	95.32%
SCRNN	**spiking**	**96.59%**	**90.28%**

addition processing to the data. The SLAYER algorithms [199] using CNN achieved 93.64% on average. Although the SCRNN does not outperform the SLAYER based CNN network in 11 class classification, the SCRNN is still competitive at 90.28%. We conclude this accuracy drop for the 11 class recognition task is due to the introduction of the additional class of random gesture. The "other" class in the DVS gesture dataset consists of random samples and each of those is neither same as other samples nor does it fall into the first ten categories. The SCRNN with designed recurrent convolution operation is found to be less effective to such type of training data. The pointnet++ [244] processed individual event data by a MLP based feedforward neural network which achieved the best accuracy in both 10 and 11 category gesture recognition tasks. However, the pointnet++ is not a spiking based training algorithm that has less potential to be applied to neuromorphic hardwares and the DVS data in their method needs to be modelled as multiple points cloud with each spike{x,y,z} fed into a MLP.

4.9 Effect of Recurrent Connection

To further demonstrate the effectiveness of SCRNN for the category-limited dynamic scene recognition, an experiment was designed to directly compare the effect of the recurrence for the 10 class gesture recognition. A feedward spiking convolutional neural network and a SCRNN were designed following a "same learning capacity rule" as is shown in Figure 4.9. The two structures are exactly same in neuron parameters, number of neurons and number of layers except the SCRNN has a recurrent connection in each convolution layer. For both structures, with the segmentation length of l_s, the first layer is a pooling layer with a kernel size of 4x4xl_s which reduced the dimension of data from 128x128xl_s to 32x32xl_s. The second layer is a convolutional layer that has a kernel size of 3x3xl_s with 16 hidden neurons. The third layer is a pooling layer using 2x2 kernels to further reduce the dimension of each feature map to 16x16xl_s. The fourth layer is a convolutional layer with 32 hidden neurons with the kernel size of 3x3xl_s, which the output is flattened and fed into a fully connected layer with 5256 neurons followed by the output layer to perform the classification.

The feedforward spiking CNN is different from the SCRNN in the training phase. For CNN, the first 1s event data of each sample with a temporal resolution of 1ms(l_s = 1000) is used as the input data which only need to be fed to the network once per sample. The SCRNN takes the same length of input data in total for each sample but a segmentation length of $l_s = 50$ is selected to partition the input into 20 subsets. This represents that the SCRNN needs to take the data to perform the recurrent processing iteratively.

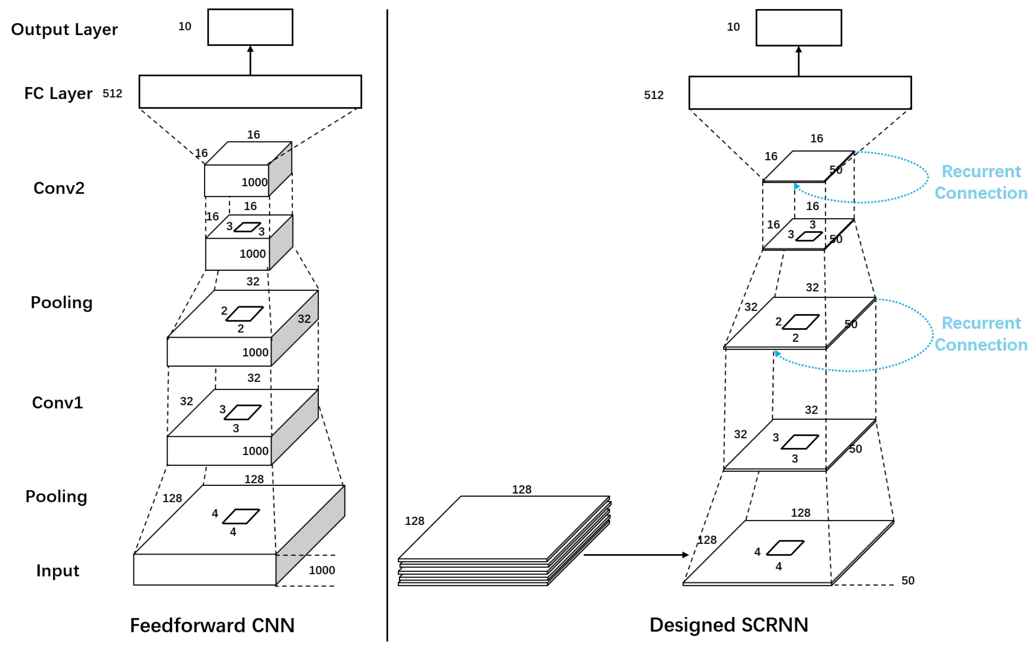


Figure 4.9: The network structure for the experiment of the feedforward Spiking Convolutional Neural Network and SCRNN

Both of the designed structures are trained 100 epochs for 5 trials with different weight initializations, the averaged testing accuracy dynamics of these two experiments are plotted in Figure 4.10. It is seen that the performance of the SCRNN compared to standard feedforward spiking CNN with the similar learning condition can provide a faster convergence speed. As is shown in Figure 4.10, the averaged testing accuracy of SCRNN stabilises after approximately 40 epochs while the CNN requires about an additional 25 epochs to fully converge with the data. Besides, the SCRNN without the inference of the unknown class can provide a recognition accuracy of 88.64% on the 10 class gesture recognition in this particular structure, while the feedforward CNN only achieves 84.09%.

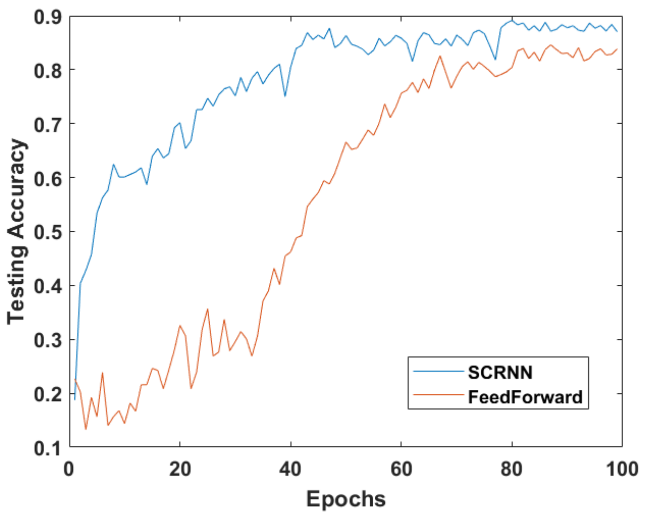


Figure 4.10: The testing accuracy curve for the designed experiments

4.10 Conclusion and Discussion

This chapter presented a novel spiking convolutional recurrent neural network that is designed for efficient event-based hand gesture recognition. The individual cell is able to extract the spatial features by 3D spiking convolution operation and transferring the information recurrently. The SLAYER is used in the SCRNN for the network training.

The SCRNN was successfully deployed on the DVS 128 gesture dataset. The SCRNN tested on the IBM DVS gesture dataset achieved an averaged recognition accuracy of 96.59% for 10 category classification which outperformed the original IBM's work. An accuracy of 90.28% was obtained for 11 category classification. Section 4.9 has shown that the designed SCRNN compared to standard feedforward CNN structure performs less competitive for the 'unknown' class but has the advantages in terms of convergence speech and accuracy for the fixed amount of categories.

However, the usage of SCRNN is not only limited to the action recognition but can be extended to various dynamic scene recognition and prediction tasks since it can also handle many-to-many and one-to-many tasks.Additionally, using new neuromorphic hardware with low SWaP(Size, Weight and Power) profile, the SCRNN has a potential to be implemented as a efficient training algorithms for neuromorphic action recognition based applications.

Chapter 5

Novel Spiking Speech Processing Algorithms

5.1 Introduction

In this chapter, two novel spiking speech processing algorithms are presented. Firstly, a cochlear bio-inspired audio spike converting algorithm is proposed in section 5.2 as an efficient coding strategy that mimicked the sensing methodology of human auditory system. The output spike trains effectively expressed the time-frequency information of the speech signal into the spike firing rate.

Secondly, a spiking speech enhancement algorithm is proposed in section 5.3 that is based on the lateral inhibition mechanism in SNN. This algorithm employs simplified idea of speech coding algorithm to firstly convert the input speech signal to event-based data, then the connectivity of lateral inhibitory SNN is applied in a layerwise local to global competitive fashion. The proposed architecture does not need to be trained to react for specific noise type but only uses forward propagation with naturally event based information processing.

5.2 Cochlear inspired auditory spike coding algorithm

5.2.1 Introduction

The utilizing of spike train as main processing unit allows network to incorporate strong temporal information which is especially useful for speech processing. Unlike the DVS(section 3.6.1) which replicates the visual cortex function for image processing, acoustic sensing using spiking technologies has not been well developed. Furthermore, very few information encoding methodologies can be applied to input data. In order to take advantage of SNNs in audio processing domain, it is essential to develop the correspond compressive information converting method which is the main idea of this work.

A commonly used neural coding scheme is rate coding as described in section 3.3 [137] where the frequency of rate of spike trains is dependent on the intensity of the input stimulus. For example in [245], SNNs incorporating STDP on computer vision tasks where the input image is encoded into spike trains according to the pixel value at every locations. Proper managing the arrangement of synapses connections and input sequence between neurons could involve the spatial relations between pixels to pixels from the image. Unlike images, 1D signal especially speech has more temporal information correlated and at most circumstance time-frequency domain information is also vital for orientated applications like speech recognition and speech enhancement.

Sergey etc [246] showed a time window based multichannel coding scheme. A fixed size time widow is applied over the signal of each channel, then compute root mean square (RMS) of the signal being the intensive of the rate coding. This strategy however is not suitable for signals that contain high time-frequency information due to the RMS calculation only gives a vague estimation of the signal strength averaging but actually missing the information either in temporal domain or frequency domain.

Humans auditory sensory system has ability to decompose the spectral components in real time with its efficient compressive coding scheme that can transporting the acoustic signal to central nervous system in proper fashion. Scientists have identified its sensing principles through many experimental observations on primates. The cochlea known as the main signal processing unit in the auditory system, which receives the perceived outside stimulus (i.e. acoustic) and translates it to electrical signals that higher level auditory cortex could understand.

Based on previous studies on cochlear partitions [247–249], the basilar membrane of the cochlear encodes the induced activation into electrical spikes. It is able to decompose the sound into several frequency components and based on this property we can assume that different spatial location of auditory cortex responds to certain frequency band. Another important organ in cochlear known as inner hair cells(IHCs) [250] which exhibits primary function that encode the graded potential from basilar membrane into neural spikes through mechanoelectrical transduction. The neural spike train is a series of binary electrical spikes that only fire when a certain membrane potential is reached. Single spike would not carry information within its amplitude but instead all information is encoded in the frequencies of a series spikes.

5.2.2 Conversion Algorithm

The cochlear has an ideal root that serves as the biological auditory cortex. In engineering domain this process could be modelled as a sensing algorithm that can achieve orientated fast spiking coding from outside stimulus. As is shown in Figure 5.1, the primary function of cochlear can be described in a order as signal segmentation in frequency domain, peak extraction followed by lower/upper value thresholding and finally rate coding process.

The main function of basilar membrane has been usually modelled as bandpass filtering function [251]. Thus, signal decomposition is designed as passing input signals through several bandpass filters - this can be regarded as using a window to segments the signal in frequency domain. Considering the unique properties of speech signal, lower frequency components usually contain higher energy and information than high frequencies, pass band range can be arbitrarily altered in terms of the analysing purpose. Each n^{th} filter convolved with given signal with central characteristic frequency ω_n that represents the corresponds frequency bands. Letting $x(t)$ be the input signal and $h_n(t)$ be the impulse response of the n_{th} bandpass filter, output from n_{th} filter $y_n(t)$ can be

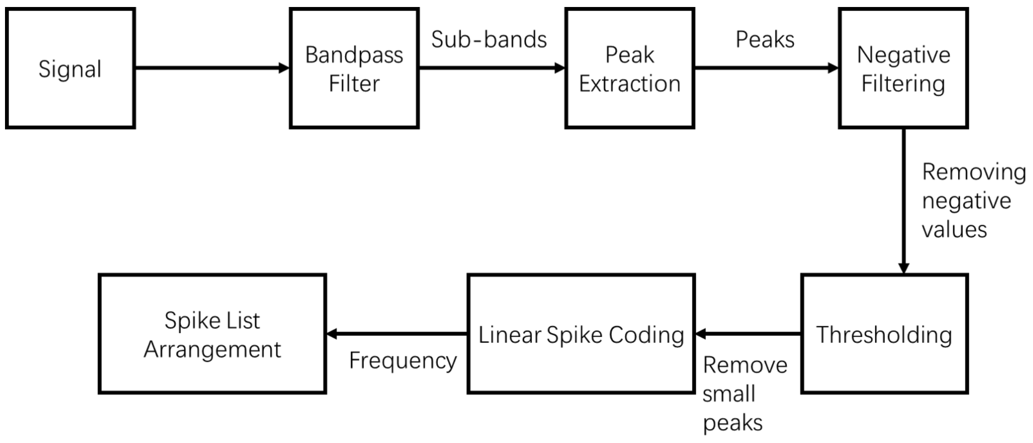


Figure 5.1: The block diagram description of speech spike coding algorithm

found as follows.

$$y_n(t) = \int_0^N x(\tau)h_n(\tau - t)d\tau \tag{5.1}$$

Then the reconstruction of $x(t)$ is the sum of all the bandpass filter outputs as (5.2).

$$x(t) \approx \sum_{n=1}^{N} y_n(t) \tag{5.2}$$

It should be noted that reconstruction quality directly related to the number of bandpass filters. The resulting filtered signal can be considered as the decomposed sub-band feature in desired frequency which is determined by central characteristic frequency of bandpass filters. This estimates the input stimulus over all the interested frequency bands. Avissar et al. [252] suggested that in auditory sensing system the firing rate is linear correlated with the peaks of sinusoidal signal. Regarding the output from bandpass filter containing large amounts of samples depending on the sampling frequency, a peak extraction algorithm used after bandpass filter will significantly reduce the coding computational intensity while saving the majority of temporal features of signal in forms of coding. The detected peak is further filtered to eliminate the negative/accident peaks values which are considered meaningfulness to speech frequency content. The remaining detected peak magnitude will then be linearly encoded into frequency of spike trains. Let M_n^i being the magnitude of i^{th} peak of the output signal from n^{th} bandpass filter, u and l are the upper and lower firing rate boundary respectively, the firing rate f_n^i can be calculated as:

$$f_n^i = l + \frac{(u-l)M_n^i}{Max(M_n^i)} \tag{5.3}$$

The number of encoded spikes are proportional to the frequency, which means higher frequency bands will result in larger number of spikes. The reconstruction procedure in turn is that decodes the peak values locating in each frequency bands from firing rates of spikes. It has been pointed out that the algorithms reconstruction quality and compressing rate highly dependent on the number of bandpass filters, bandwidth of each filter and the striding steps of each filter [253]. These parameters are highly application dependent, when assuming the compressing rate is not prior consideration while these spikes being the input of SNNs.

5.2.3 Experiment Results

To evaluate the distinction features detected by the sensing conversion algorithm, the Edinburgh speech enhancement dataset [254] is selected as the input and then visualize the output spikes in raster plot form. The dataset contains 28 speakers clean sound recording which sampled at 48kHz. Table 5.1 illustrates the information of a speech sample in the database which will be carried out for discussion later on. The bandpass filtered bands were designed using a minimum-order filter with the bandwidth of 50Hz and 60dB stopband attenuation. Each two adjacent passband of bandpass filters are 50% overlapped to satisfy a 25Hz frequency resolution which is also the interval of the central characteristic frequencies of two adjacent bandpass filter. As stated in last section, the quantity of spikes will increase significantly along with the frequency. The sensing range was set from 20Hz to 2.5kHz, where the lower boundary matches

Table 5.1: The experiment speech corpus information

Index	Description	Duration(s)
P232_001	please call stella	1.743

humans auditory system and upper boundary is sufficient for analysis purpose for the speech signal. This results in 199 bandpass channels filtered the signals that appear in parallel, which means separating the signal into 199 sub-bands with 25Hz frequency resolution. The frequency resolution is arbitrarily adjustable without losing time domain resolution by changing the bandwidth of bandpass filters and overlapping rate. This provided a sensing friendly characteristic of the algorithm that could achieved both high frequency resolution and time resolution if needed when a large number of narrow-bandwidth bandpass filters involved in. In the experiment, the parameters were chosen as l=50Hz and u=100Hz that limits both the highest and lowest firing rate that can be encoded from each peak value. The large difference between upper and lower firing rates boundary will be helpful to distinguish the time-frequency pattern visually for study purpose. Each peak value will be encoded into 5 spikes then being placed in an order at the time location where original peak located at.

The Figure 5.2 demonstrates a sample results of the output from the proposed algorithm. To clearly demonstrate the spike activities that encoded in the results, a zoomed in figure among the region approximately in time range of 0.72s to 0.88s and frequency range approximately of 100Hz to 700Hz is shown in Figure 5.3.

5.2.4 Comparison with DAS

For comparison purpose, the output from the spectrogram generated by the same STFT window size is shown in Figure 5.4. It can be clearly seen that the proposed spike coding algorithm well estimated the time-frequency information by comparing the spike activities(Figure 5.2) to the energy spectrogram(Figure 5.4). The spike coding algorithm only preserves the high-energy valued temporal-frequency components due to the proper use of the threholding technique. To further validate the robustness of the proposed coding technique, a chirp signal is input to the algorithm to compare with the output from the output signal from DAS sensor(described in section 3.6.2). Figure 5.5 shows the response from the DAS sensor, where red and green coloured signal represents the left and right ear channel. Figure 5.6 demonstrates the output signal that generated from the coding algorithm. It is clear that the algorithm can output a DVS equivalent signal and do not suffer the analog noise.

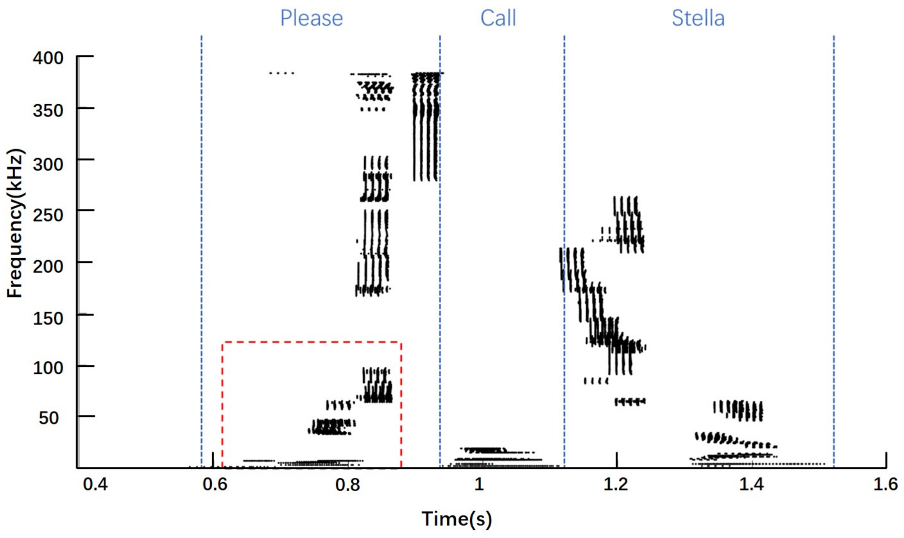


Figure 5.2: The spike coding results for speech 'please call stella', red box:zoom in region

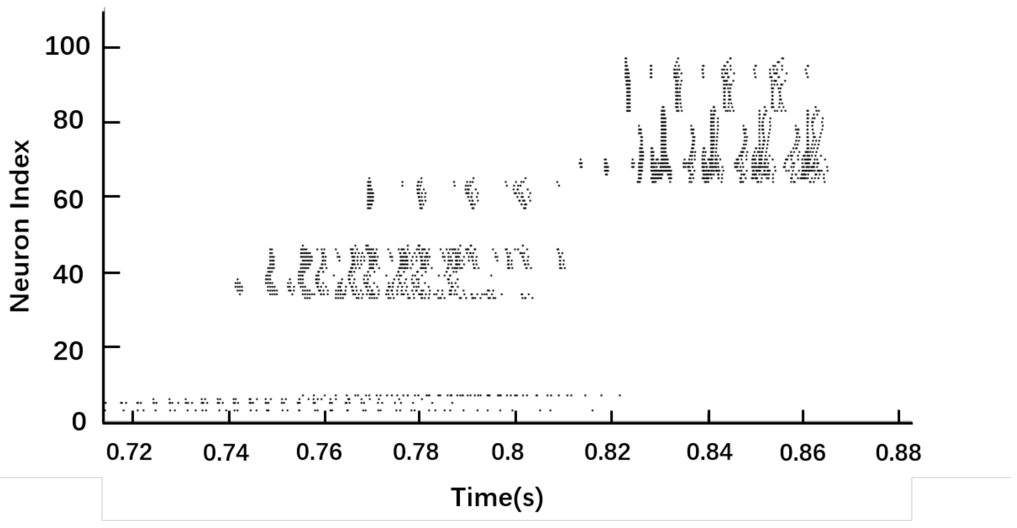


Figure 5.3: The detailed spike activities of low frequency bands of word 'please'

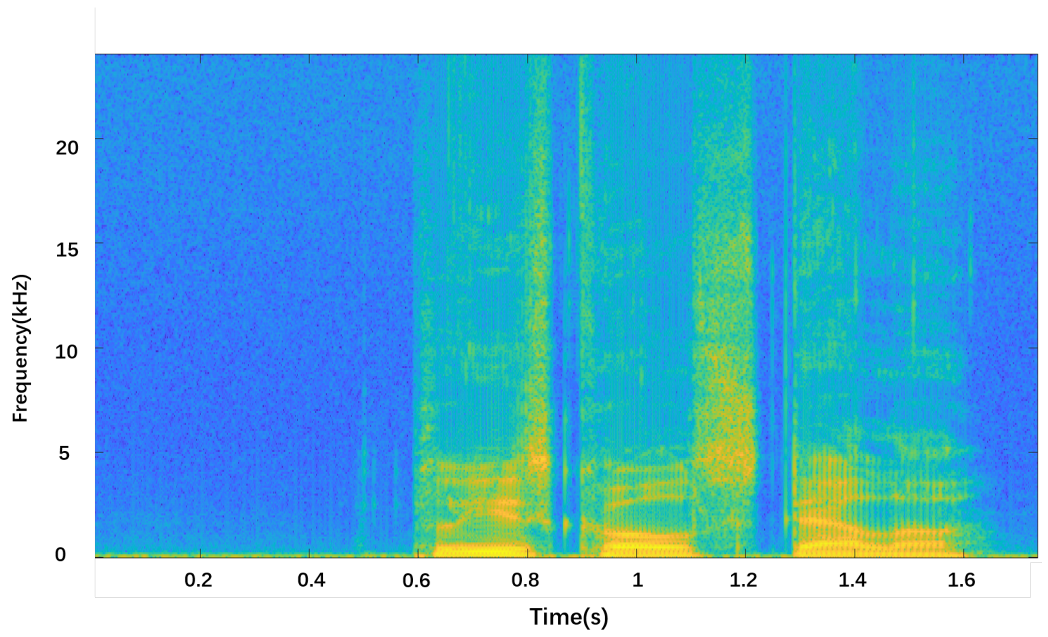


Figure 5.4: The spectrogram results for speech 'please call stella'.

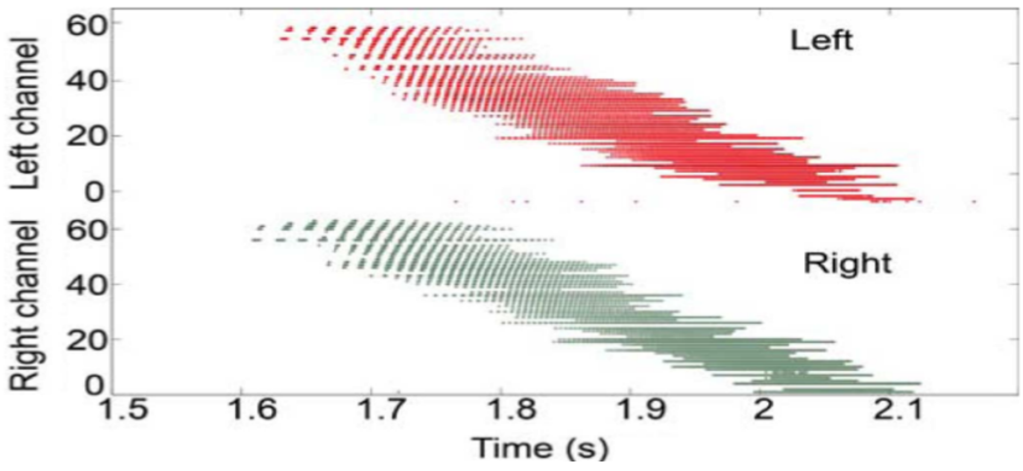


Figure 5.5: Chirp signal response from DAS

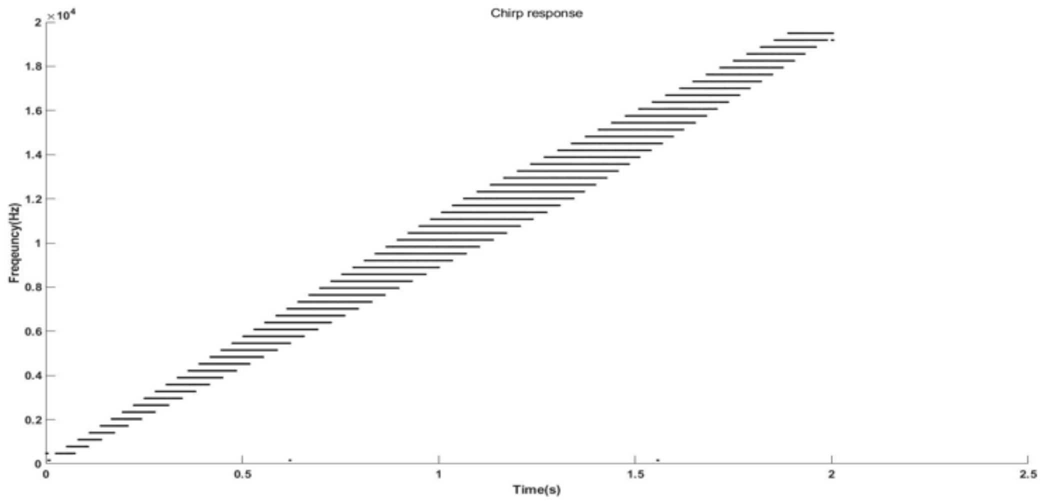


Figure 5.6: Chirp signal response from proposed spike coding method

The successful design of the speech coding method provides a alternative event-data acquisition method for audio based data. The technique can simulate the functionality of DVS to efficiently transfer digitised audio signal to spike trains that encoded time-frequency information.

5.3 A novel spiking speech enhancement system

5.3.1 Introduction

In this section, a novel noise reduction method that is based on neuron rate coding (introduced in section 3.3 and section 5.2) and bio-inspired spiking neural network architecture is proposed. This work extend the contribution of audio coding to a real application, which demonstrates how the coding method can affect the way of event-based processing in terms of SNN design and performance. To improve the computational efficiency of the coding, the bandpass filters are is simplified with short-time-fourier-transform(STFT), the spike counting stage is replaced by bio-inspired LIF neurons(described in section 3.2.1) which can automatically transfer the input stimulus to rate coded spike trains.

5.3.2 Motivation

In the past decades, numerous speech enhancement techniques have been investigated by researchers. Spectral subtraction [255] subtracts an estimated noise spectrum from the noisy signal to produce the denoised spectrum. Ephraim and Malah introduced the minimum mean-square error (MMSE) [256] that reduces the residual noise level without significantly affecting the original speech components. The optimally modified log spectral amplitude estimator (OMLSA) [257] and improved minima controlled recursive averaging (IMCRA) [258] offer high performance in speech enhancement tasks.

Early work using shallow neural networks [259, 260] estimated Signal-to-Noise-ratio (SNR) based on the spectrogram which is then subsequently used to reduce the noise in each frequency band. In [261] and [262] speech enhancement was considered as a classification problem to predict an ideal mask in the time frequency domain to estimate the presence of speech components. Although DNN based models can achieve effective noise reduction, they usually require large datasets to represent various types of noise and multiple hidden layers with a significant number of free parameters. The computational cost and power-hungry nature of DNN based speech enhancement technique makes it difficult for them to be applied on SWaP limited devices.

With limitation of conventional techniques on speech enhancement tasks and a requirement of a efficient pure-spiking algorithms for speech processing, a need to develop a spiking speech enhancement algorithm is identified.

5.3.3 Neural Synchronization

The use of lateral inhibition as neural synchrony and coincidence detector was investigated by Abbott [263]. A simple SNN is illustrated in Figure 5.7. It comprises 3 spiking neurons that each produce a spike train output. In Figure 5.7(a) neuron A and B are Leaky-integrated and fire (LIF) neurons(described in section 3.2) that interacts with each other via lateral inhibitory connections. A simplified differential equation that describes the membrane potential dynamics of a LIF neuron model can be expressed as:

$$\frac{dv}{dt} = \frac{R_{mem}I(t) - v}{\tau_{mem}} \tag{5.4}$$

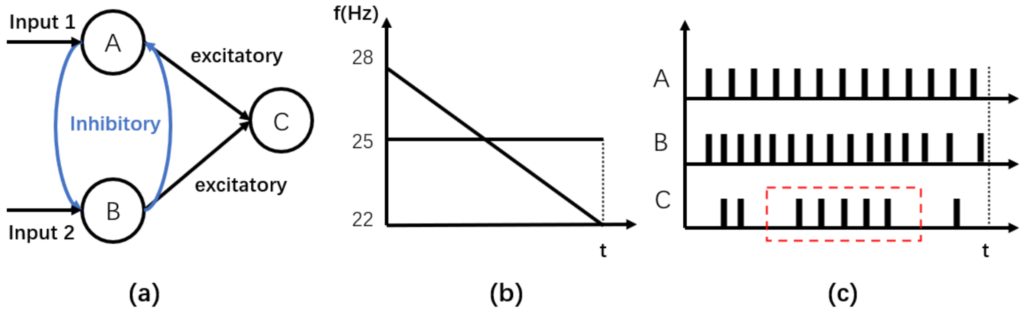


Figure 5.7: Illustration of two LIF neuron SNN that behave neural synchronization phenomenon

where v is the membrane potential, R_{mem} denotes the membrane resistance, τ_{mem} refers to the membrane constant and $I(t)$ stands for the synaptic input current. The LIF neuron reacts to input stimuli that raises a certain amount of membrane potential. Once the membrane potential is greater than a pre-defined membrane threshold, the neuron will emit constant amplitude spikes at a certain frequency which is dependent on the magnitude of membrane potential. The inhibition in the example is modelled as one decreasing its membrane potential due to the other neurons firing activity. As illustrated in Figure 5.7, Neuron B is inhibited from firing if Neuron A is firing and vice versa. Output neuron C simply receives the output spikes from A and B to generate output spike trains. Figure 5.7(b) shows the input stimuli (synaptic input current) to neurons A and B are different over a certain time period t. One input excitation makes the neuron firing at constant rate of 25Hz (A in Figure 5.7(c)) while the other input makes the neurons firing rate linear changing from 28Hz to 22Hz (B in Figure 5.7(c)). As shown in the highlighted red rectangular in C in Figure 5.7(c), the output neuron C will fire maximally in a short period when A and Bs firing rates are approximately equal. When they have different firing rates, the two neurons tend to inhibit each other in turn leading to sparse events, until their firing rates reach the range of synchronization (nearly the same).

Cornelius et al. demonstrated that multiple fully connected lateral inhibitions are able to exploit the inhibitory process between neurons, to remove uncorrelated spikes (frequency difference) [264]. This mechanism can be extremely useful when the speech components are able to be coded into spike trains with similar frequencies.

5.3.4 Speech Coding Method

The Short-Time-Fourier-Transform (STFT) is used in this work to form the complex spectrogram. The STFT can perform equivalent time-frequency domain information extraction process as the strategy proposed in section 5.2.2 which utilizes cascaded structure consists of bandpass filter, peak extraction, negative filtering and thresholding. The benefit using STFT to approximate the time-frequency characteristics of audio is that it significantly reduced the computational costs caused by a series of filtering operations.The STFT is formed using a Hamming window length of 1024 samples to provide high frequency resolution with 80% overlap which results in 514 frequency channels ranging 0 to 8kHz.

Recall the dynamic behaviour of LIF neuron model mentioned in section3.2.1, the LIF neuron can transfer the input stimulus to spike trains which frequency is non-linearly proportional to the corresponding input magnitude. This mechanism, therefore, is considered to be used in the design of the input coding stage. The absolute value of the output complex matrix from the STFT is log scaled and normalised to the input current to LIF neurons. The number of input neurons is set to be the same as the number of frequency channels according to the spectrogram. The use of the lateral inhibitory connections preserves the spike trains that have approximately the same frequency. Each LIF neuron responds to noise and speech components to generate fixed frequency spike train during every single time resolution bin. The LIF neuron is expected to have a higher firing rate in a STFT time resolution bin to the speech components. In contrast, the noise components should be converted to low frequency spike trains which are easier to be distinguished by lateral inhibition. The firing frequency of a LIF neuron usually is proportional to its input, but this is not obvious when simulating it at a very small time step.

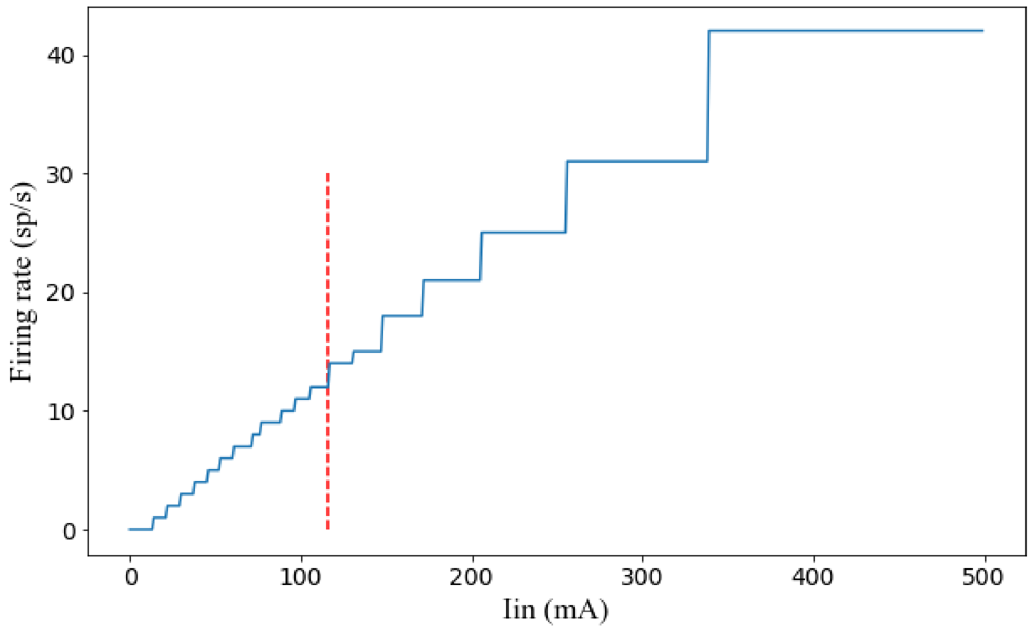


Figure 5.8: The firing rate dynamics experiment of a LIF neuron with simulation time step of 100μs, the input current linear changes from 0 to 500mA where the neuron firing rate becomes unstable after 115mA.

Figure 5.8 shows the firing rate dynamics of a LIF neuron with input current linear changing from 0 to 500mA with the simulation time step of 100us. The firing rate profile displayed in Figure 5.8 ensures the LIF neurons are able to response differently to a certain range of synaptic input currents during a single time bin of STFT temporal resolution (i.e. the time difference between two adjacent value in a same frequency band). The neuron firing frequency becomes unstable when the input current is over approximately 115mA. Thus, the range of input current from 0mA to 115mA is scaled to provide a balanced input current normalization. In our case, the input neuron firing rate ranges 0-15Hz which means there will be maximally 15 spikes that can be observed in a single time resolution bin of STFT. The full spectrogram is input to 514 LIF neurons by updating the input current of each input neuron based on STFT time resolution. Figure 5.10 shows the results of spike coding method applied on the clean speech sample. For comparison purposes, the spectrogram of the same speech sample is shown in Figure 5.9. It can be seen that the method can represent the temporal-frequency pattern of the speech signal. Figure 5.11 demonstrates the coding results for white noise corrupted speech sample with SNR=1dB. The resulting raster plot shows the speech components are densely packed (high frequency spike trains). The goal of the SNN is to remove the sparse distributed spikes resulting from noise and preserve the speech elements.

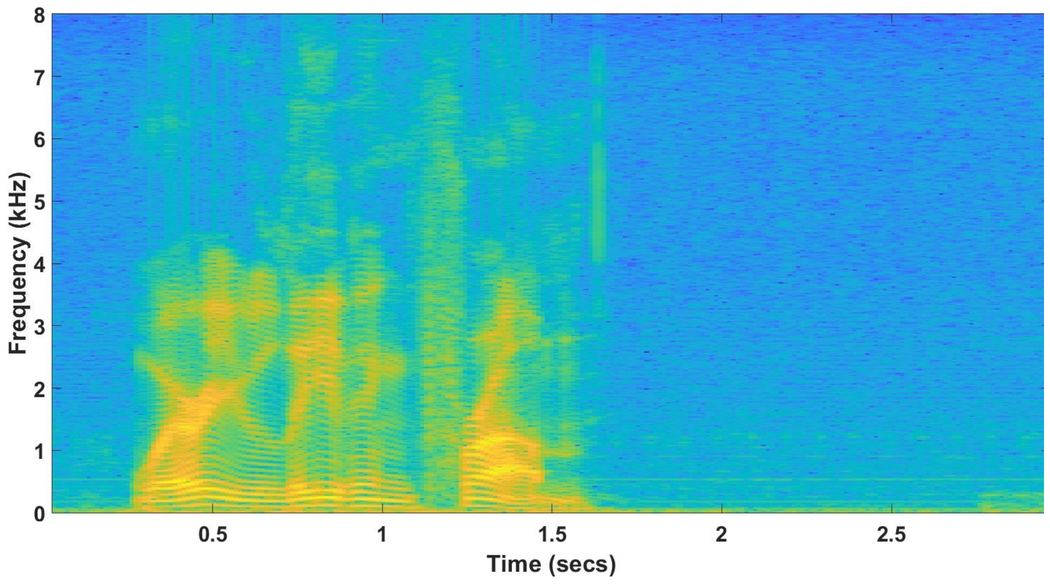


Figure 5.9: Log Spectrogram of clean speech signal

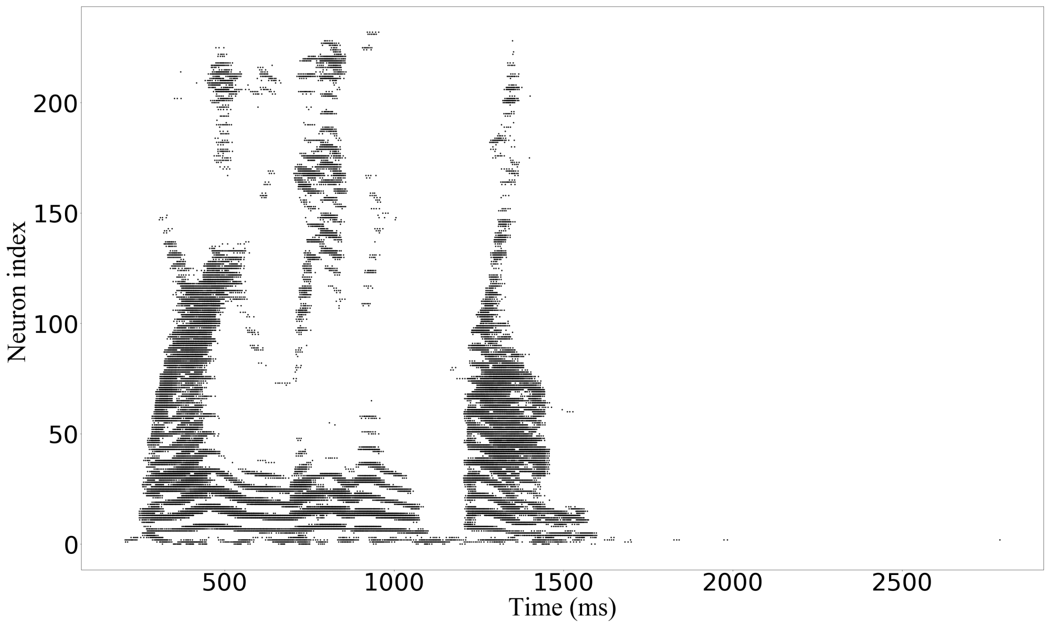


Figure 5.10: Spike coding result of clean speech samples

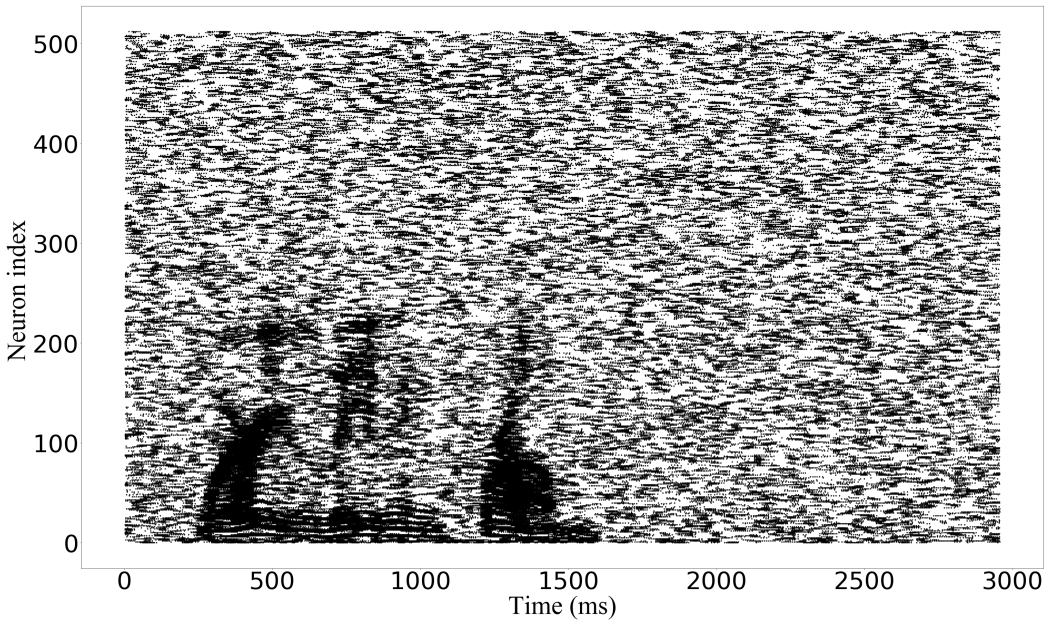


Figure 5.11: Spike coding result of noisy speech signal (with white noise SNR =1)

5.3.5 SNN with lateral inhabitation

In [265], lateral inhibitory SNN with neighbourhood connectivity [266] has been successful demonstrated on Gaussian while noise corrupted speech. Unlike the approaches described in [265] and [266] which uses global inhibitory actions we consider a local to global inhibitory connection strategy. The lateral inhibitory connections are built with different inhibitory radius for each layer. The inhibitory radius defines how neurons in a layer that are close to one another are connected laterally. The basic idea is described in Fig 5.12. The dynamic inhibitory radius, results in a local to global neural temporal competition while the lateral inhibition simultaneously removes spikes that are sparsely distributed in time from small set of frequency channel to all frequency bands.

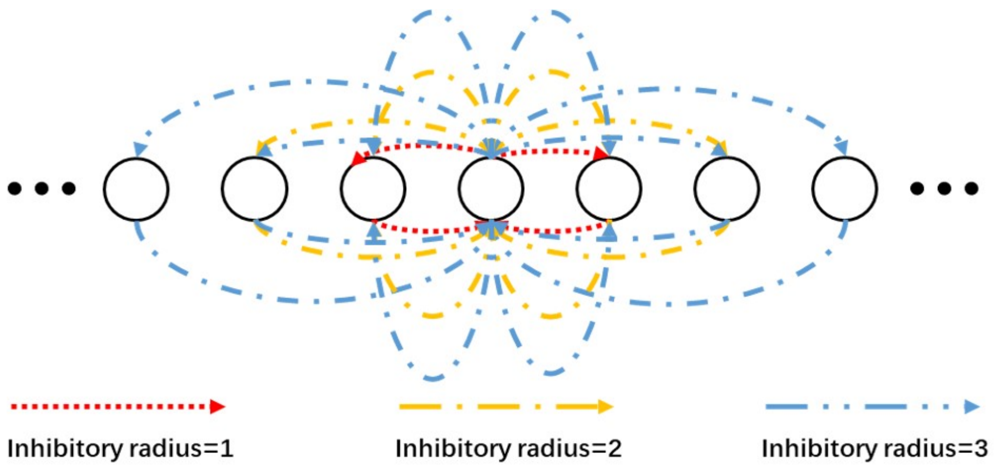


Figure 5.12: Lateral inhabitation in terms of connection radius which defines how close the adjacent neurons can be lateral connected

To adapt the different inhibitory connection radius for each layer, the inhibitory strength is modelled as (5.5) to exponentially decay in terms of synapse length (distance between two neurons) rather than a constant inhibitory strength for all inhibitory connections.

$$W_n = A_n \cdot e^(- \frac{|i-j|}{D_n}), i \neq j \tag{5.5}$$

where n is the layer index, W represents the weights (strength) of the inhibitory synapse, A represents the maximum inhibitory strength, i and j denote to the neuron index of two lateral connected neurons, and D is decay constant. This is because it was discovered through simulation that strong inhibition for large inhibitory radius (global competition) will result in information loss. On the contrary weak inhibition for small inhibitory radius (local competition) can have little effect on removing unsynchronized spikes.

5.3.6 Experiment Results

A test clean speech corpus was obtained from the VoxForge open public dataset [266], where speech sample is sampled at 16kHz. We selected 5 types of real-world environmental noise corrupted speech from the DEMAND noise database [254], including: living, office, river, kitchen and white noise. A range of SNR were chosen for test performance of proposed model (-5, 0.1, 1, 5 and 10) dB.

The Python and the BRIAN are used as the main simulator for this study(described in section 3.7) [223]. A 3-layer SNN is used with each layer contains 514 LIF neurons that are laterally connected with different inhibitory radii. The connection between each layer is one to one using excitatory synapses. The inhibitory radius from layer one to three are set as: 10, 50 and 250 respectively. The inhibitory synapses parameters are set as A_1 =7, A_2 =5, A_3 =1, D_1 =5, D_2 = 30, D_3=250. The selection process of these parameters was according to the experiments. It was found that, a lower inbitory strenth with large radius performes bad in terms of remove noisy spikes. A high inbitory strenth can result in dominant probelm. The output from SNN is spike times of 514 neurons which represents 514 frequency channels of STFT. The quantity of spikes can accurately represent the log intensity of correspond time frequency element. Thus, the processed spectrogram can be obtained by summing the number of spikes and linear decoding for each time resolution cell in the spectrogram. Due to the lack of phase in the log spectrogram we use the processed spectrogram as a binary mask for the original complex spectrogram. The mask is constructed by comparing the number of spikes in a spectrogram cell to a certain threshold, which determines the ON(1)/OFF(0) status of correspond location. The mask is then element by element multiplied to both real and imaginary parts of the original noisy spectrogram. This preserves the original phase information from the complex numbers of spectrogram which can be used to perform ISTFT to reconstruct time domain signal.

Figure 5.13 shows an example of raster plot output from SNN. The spikes that are not densely packed in Figure 5.13 are further reduced by the thresholding during decoding process (from raster plot to spectrogram). Figure 5.14 demonstrate the reconstructed spectrogram using the binary mask. Compare to the Figure 5.13, most sparsely distributed noise is supressed. An example of time domain signal comparison of before and after denoising is shown that Figure 5.15(a) demonstrate the original time domain clean speech, Figure 5.15(b) is the noisy speech corrupted by white noise with SNR = 1dB and Figure 5.15(c) is the noise reduced and reconstructed time domain speech signal.

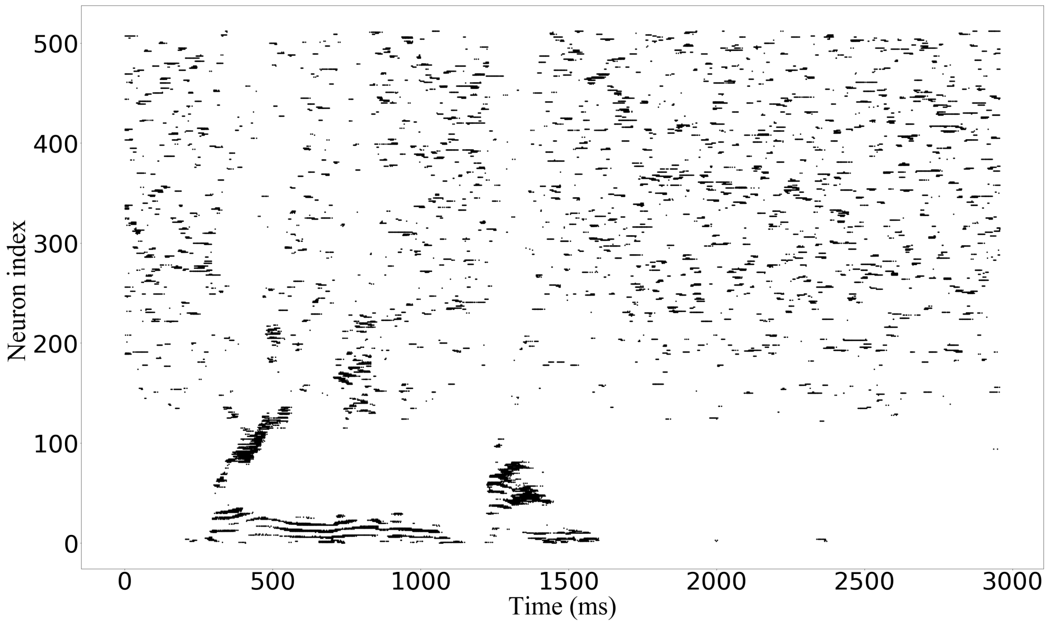


Figure 5.13: The SNN spike output, noisy spikes are significantly reduced compared to Figure 5.11 (white noise SNR=1)

Five types of noise sources were used to evaluate the effectiveness of the SNN based method. During the set up of simulations, informal listening was carried out to subjectively determine how successful the method performed. To determine the numerical speech improvement, the clean signal is pass through SNN and the output signal is used as the reference target speech signal. The power of residual noise signal is determined by estimating the signal power during the time when no speech is presented. This assumed that the noise signal is stationary over the presence of speech. The reason

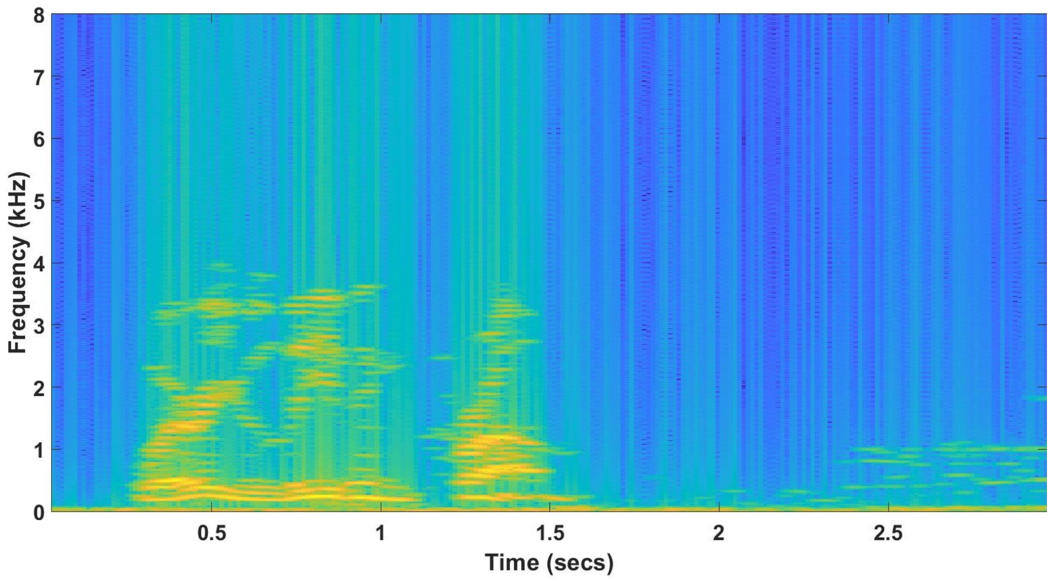


Figure 5.14: Reconstructed spectrogram (white noise SNR =1) Note sparse spikes are removed during the thresh hold process

of not using the subtraction to obtain the residual noise is because of the non-linear and unsynchronized information processing property of the SNN.

Table 5.2 presents the results on 5 types of noise with different noise level. The lateral inhibitory based SNN can have an average improvement of SNR of 10.915dB among 5 types noises. However, the improvement is noise type dependent, for example

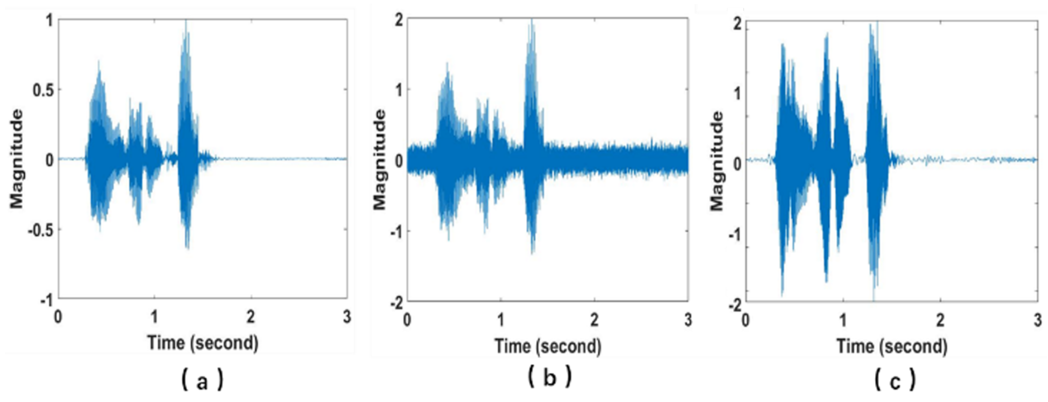


Figure 5.15: The time domain signal representation. (a) Time domain clean speech.(b) Time domain noisy speech with white noise SNR=1 (c) Time domain reconstructed denoised signal

it improved approximately 19dB for white noise but only 8dB for the living noise. We strongly believe that the parameter of proposed SNN should be dynamically tuned in terms of the noise level and type. In future work, we will investigate an automated way of tuning the parameters.

5.3.7 Discussion

The work presented has demonstrated successful noise reduction on real world noise using multilayer lateral inhibitory spiking neural networks. The lateral inhibition strengthens correlations in the time-frequency domain and naturally suppresses the noise which are usually sparsely distributed. Unlike standard artificial neural networks, the lateral inhibitory based SNN does not need to train with datasets. However, during our experiment, it should be noted that the performance of lateral inhibition is highly dependent on the presence of speech. In Figure 5.13 and Figure 5.14, the noise cannot be efficiently removed by lateral inhibition without the presence of the speech components. This is nothing to do with the SNN structure but is due to the natural property of inhibitions. A possible solution to this is to separate the speech element from the noise using effective speech detection algorithms. In next stage of our work, we will further improve the performance of lateral inhibitory SNN by applying speech detection algorithms to detect the presence of speech.

5.4 Conclusion

In this chapter, two novel event-based speech processing methods are proposed.

Firstly, section 5.2.1 to 5.2.3 reported a novel idea that could lead to efficient coding the speech signal into high time-frequency pattern correlated spike representations. The main objective of this work is to address a software-based solution that can help current SNN effectively sensing 1D signal that contains complex temporal-spectral information. The resulting raster plots demonstrate how the converted spike trains can be useful to further SNN speech processing applications. Combining the the techniques that illustrated in chapter 3, SNNs processing systems can achieve real-time power efficient

Table 5.2: Experiment results of proposed speech enhancement algorithm on variable types of noisy signals

Type of Noise	**Original SNR**	**Measured SNR**	**Enhanced SNR**
White	-5	-5.033	13.62
	0.1	0.0662	19.48
	1	0.9662	23.08
	5	4.9662	24.48
	10	9.9662	24.62
Living	-5	-5.0436	3.22
	0.1	0.0109	8.29
	1	1.0111	9.61
	5	5.0254	13.95
	10	10.0229	15.022
Office	-5	-4.9554	2.515
	0.1	0.1049	8.016
	1	1.0234	9.07
	5	4.995	13.95
	10	10.00	15.022
River	-5	-4.8716	3.26
	0.1	0.0109	8.28
	1	1.149	9.75
	5	5.0254	13.96
	10	10.0229	18.72
Field	-5	-4.133	4.62
	0.1	0.103	9.66
	1	0.9961	10.68
	5	4.991	15.48
	10	9.996	20.8
Kitchen	-5	-4.8275	7.08
	0.1	0.1059	11.91
	1	1.0022	13.48
	5	5.0094	18.60
	10	9.9077	23.83
Average Improvement			10.915

neuromorphic computing with their bio-inspired processing mechanism.

Secondly, section 5.3.1 to 5.3.6 described a novel spectrogram coding method and a lateral inhibitory SNN structure that naturally suppresses uncorrelated noise in time-frequency domain. It demonstrated an average of 10.915dB SNR improvement on 5 types noise. Furthermore, with the emergence of novel SNN learning rule(section 3.5), it is envisaged large scale SNN with ability of denoising and recognition.

Chapter 6

A Novel Bio-inspired Spiking Speech Emotional Recognition System

6.1 Introduction

This chapter presents a novel design of a SNN based end-to-end speech emotion recognition(SER) system. A SNN is built for addressing the temporal correlations that are coded in the extracted spectral features. The SNN model is set up by the SLAYER algorithm described in section 3.5.3 using a novel spiking recurrent neural network(SRNN) architecture. The Mel frequency cepstral coefficients(MFCCs) is used as the input speech feature for the proposed SRNN structure. The eNTERface and RAVESS emotion dataset was used for validating the proposed SER system, where the proposed system achieves both state-of-art SER recognition accuracy and computational efficiency compared to previous work.

Many of proposed SER works have applied bio-inspired auditory working mechanism such as Gammatone filter bank [267] and Mel-scaled filter bank [268]. However, both feature extractors and classifiers used in previous works are mainly based on conventional machine learning or deep learning techniques. These methods usually utilize a self-designed feature extraction method followed by a range of classification structures such as support vector machines [269, 270], CNNs [271–274] and LSTMs [275].

The remainder of this chapter is organized as follows. Section 6.2 describes the overall proposed SER system. The pre-processing techniques used to convert the input signal to spikes is provided in Section 6.3. Section 6.4 illustrates the design of novel SRNN structure in detail. Section 6.5 describes the information in terms of the datasets used in this study. The experiments and results of the designed SER system are proposed in Section 6.6. Besides, a noise reduction spiking SER system is described in Section 6.8. Finally, conclusions are provided in 6.9.

6.2 The Proposed Speech Emotion Recognition System

As is shown in Figure 6.1, the proposed event-based SER system consists of 2 stages. The first stage of the system is the pre-processing stage, which transforms the input speech signals into spike represented features using designed feature extraction and spike coding algorithms. Secondly, the converted spike trains are fed into a designed SRNN to perform classifications of emotions.

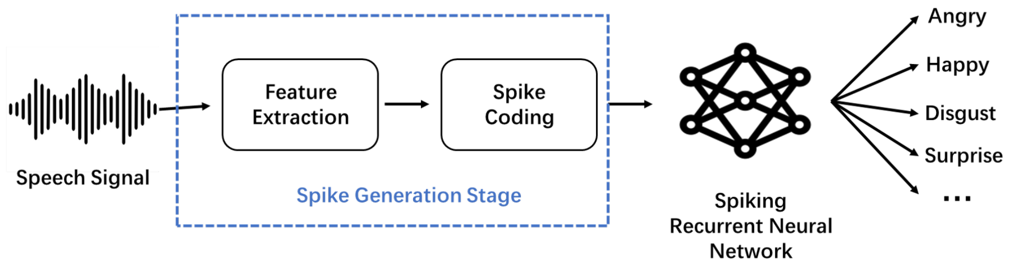


Figure 6.1: The proposed SNN based SER system

6.3 Feature Extraction and Spike Coding

Figure 6.2 illustrates the proposed speech spike generation block of the SNN SER system. MFCCs are used in the feature extraction stage since it is more discriminative to human hearing frequency ranges. As shown in the figure, the speech signal fed into the SNN is first processed so as to better extract the spectral information. The speech signal is firstly pre-emphasised to amplify the high energy frequency components. Hamming windows with a duration of 20ms and an overlap of 10ms are used in order to sufficiently acquire the temporal variations of signal. The the Short-Time-Fourier-transform(STFT) is performed on all the speech segments to extract the time-frequency energy spectrogram of the entire speech signal. The resulting spectrograms are convolved with mel-scaled filter banks to reduce the information in desired non-linear frequency bands. The generated mel-scaled frequency spectra are transformed to MFCCs by taking the DCT of the mel-scaled log powers of the spectrogram.

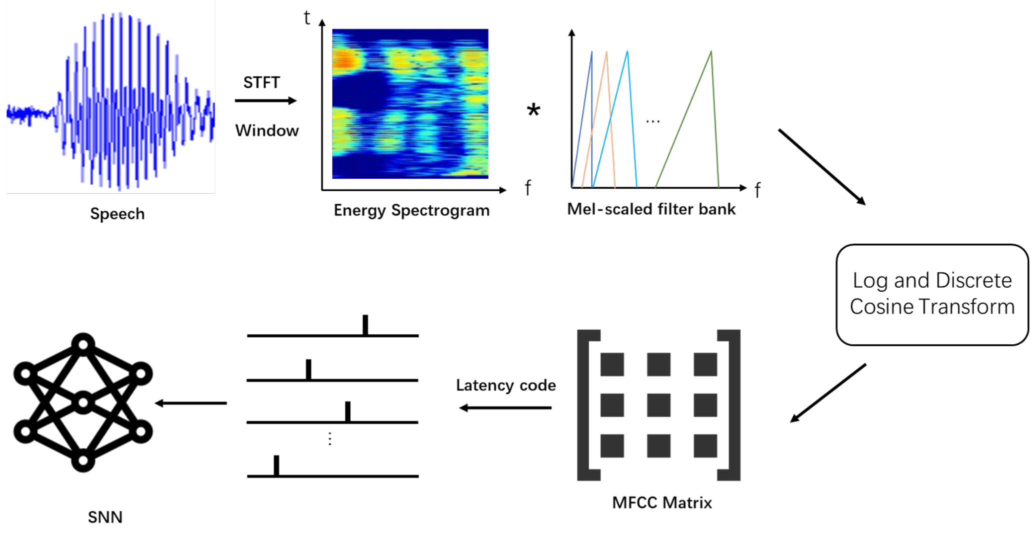


Figure 6.2: The proposed speech preprocessing block diagram for SNN SER system

The latency coding schedule is applied to the generated MFCC matrix to transform the feature value to spikes. There is evidence that the neurons in the auditory nervous system respond faster to the early individual stimulus [276]. The latency coding, as described in section 3.3.2, encodes the values in the precise spike timings. The higher intensity MFCC values are transformed to spikes with short delays in the encoding window while the lower intensity values will evoke spikes that appear later. Due to the use of single spike based latency coding, the computational cost in both pre-processing stage and SNN is significantly reduced compared to rate-based spike coding scheme. The formulation of the latency coding is given as follows:

$$t_i = (1 - \frac{X_i - min(X_i)}{max(X_i) - min(X_i)}) * T \tag{6.1}$$

where t denotes the firing time of a spike, i represents the i^{th} frequency channel which corresponds to the output from the i^{th} mel-scaled filter, X is the generated MFCCs and t_i denotes to the actual spike occurance time. The term max and min denote to the maximum and minimum operation and T is the length encoding time window. The spiking coding described above is a normalization process that normalizes the MFCCs with the same scale. The setting of the window size can be flexible, as long as the connected SNN can provide sufficient temporal resolution to distinguish individual spikes.

6.4 Spiking Recurrent Neural Network

As shown in Figure 6.3, a recurrent spiking neural network structure is designed to address the temporal correlations that are encoded in spike trains. It can be seen that the proposed SRNN consists of an input layer and a hidden cell that has a internal hidden layer within it. Typically, a layer of spiking neurons firstly integrating the incoming spike trains to PSPs by applying a thresholding operation on the generated PSPs to generate outbound spike trains. In the proposed method, the thresholding operation is omitted and the information recurrent propagation is simply achieved by sums the PSPs from the current input layer and the PSPs from the previous hidden layer. It should note that throughout the entire information flow within a cell, the spikes are only being generated after the PSPs are fused. With a train $s_i(t)$ as the input spike train to the SRNN, an output spike train $s_o(t)$ is generated, membrane potential changes in the cell is given based on the modification on the feedforward structure as follows:

$$u^k(t) = \sum w_i(\sigma(t) * s_i^k(t)) + (\nu(t) * s_i^k(t)) + u^{k-1}(t) \tag{6.2}$$

where u is the post synaptic potential, w denotes to the synaptic weights, $\sigma(t)$ is the spike response kernel and $\nu(t)$ is the refractory response kernel. k represents the time step index which is defines the order of input spike train batches and i defines the input channel index.

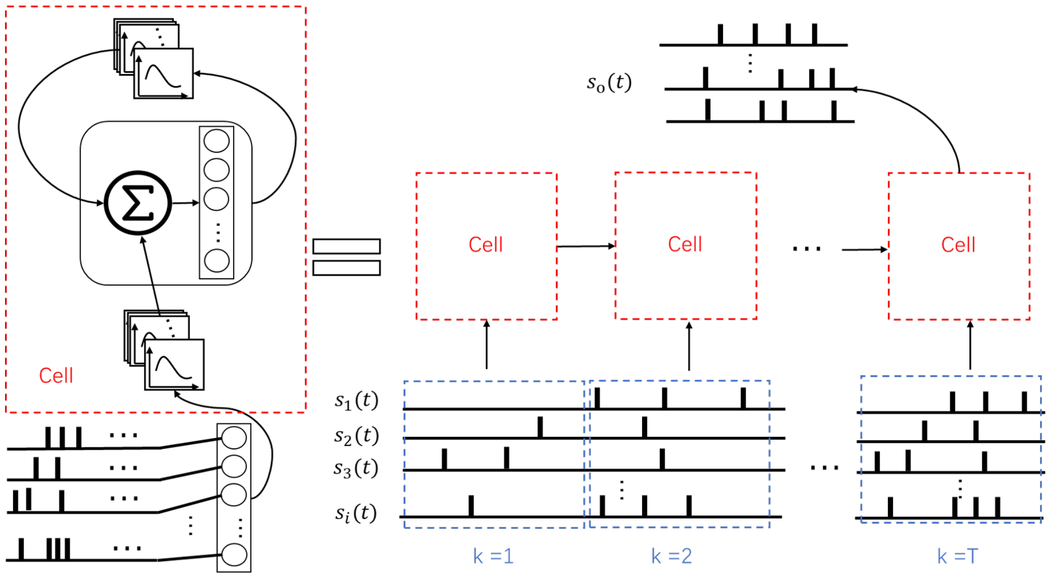


Figure 6.3: The proposed Spiking Recurrent Neural Network Architecture

It can be seen in the Figure 6.3, the SRNN structure can either be treated as a feedforward structure by unfolding the structure with stacked cells or as a independent structure which propagates information cyclically. The unfolded structure can increase the network capacity but also increases the computational costs of the network. A compromise solution is to unfold the SRNN in a short-term and segments the input sequence into subsets which will be fed into SRNN at each corresponding time step.

6.5 Dataset

Three datasets were selected to conduct experiments of speech emotional recognition, namely(i)Ryerson Audio-Bisual Database of Speech and Song dataset(RAVDESS) [277] , (ii)eNTERFACE dataset [278] (iii)EMO-DB dataset [279]. These three datasets contain the audio and video samples wherein only the speech corpus were used for experiment in this work.

The sample and gender distribution of the RAVDESS dataset are shown in Figure 6.4. The RAVDESS dataset contains emotional speech corpus that performed by 24 professional actors(12 male and 12 female), which vocalize two statements in neutral American accents. The dataset covers 8 category of emotional gestures which include Neutral, Calm, Happy, Sad, Angry, Surprise, Fearful and Disgust. The speech samples were acquired using a 16 bit-format with a sampling frequency of 48kHz.

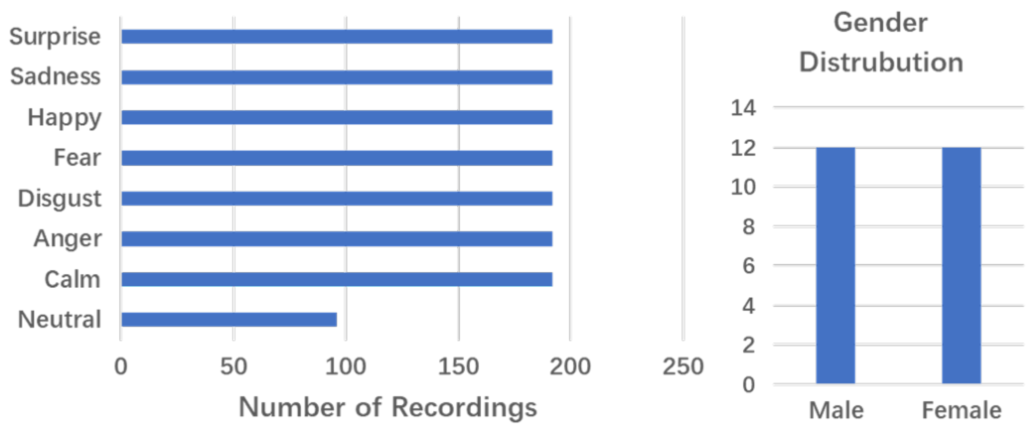


Figure 6.4: The sample and gender distribution of the RAVDESS dataset

The distribution of eNTERFACE data set is demonstrated in Figure 6.5 . The eNTERFACE records the emotional videos of 42 independent English speakers from 14 different nationalities with 81% male participants and 19% female participants. It contains 6 types of emotion which are Happy, Sad, Angry, Surprise, Fearful and Disgust. The audio samples are recorded with a sampling frequency of 48kHz and 16-bit format. It should note that, the eNTERFACE dataset is unbalanced in both the samples per class and the gender as is shown in the Figure6.5. Due to the audio samples of eNTERFACE dataset are embedded in videos and not separately available, the data used for experiment is the speech corpus extracted from videos.

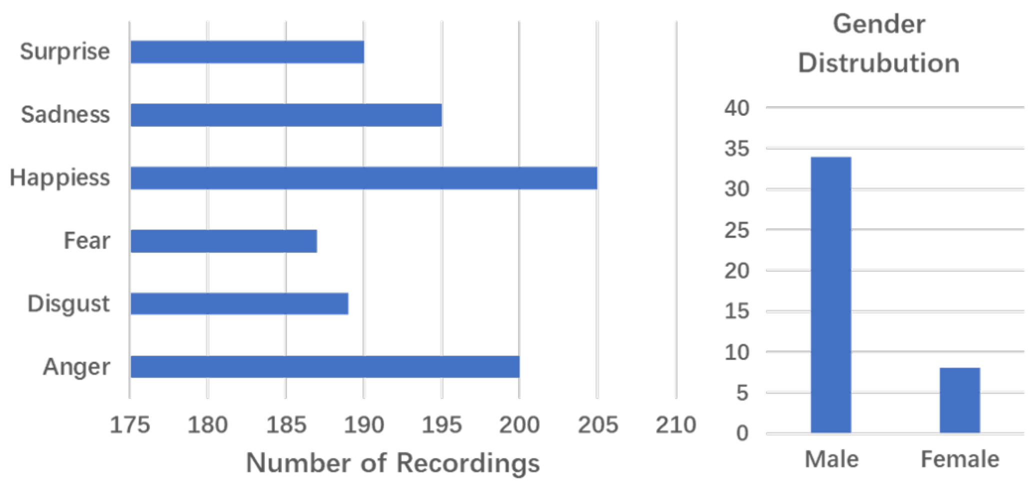


Figure 6.5: The sample and gender distribution of the eNTERFACE dataset

In addition, the EMO-DB [280] is also used in the proposed system. The dataset contains 535 audio utterances with 10 different utterances performed by German speakers. This dataset is recorded in a 16-bit format with a sampling frequency of 16kHz. It contains 7 emotion class which are Anger, Sad, Fear/Anxiety, Neutral, Disgust, Happiness and Boredom. The sample and gender distribution of the dataset is shown in Figure 6.6.

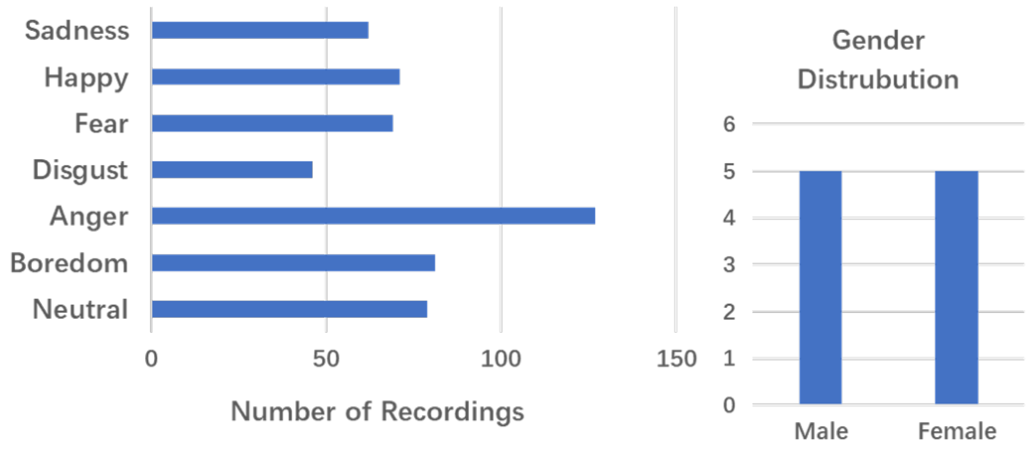


Figure 6.6: The sample and gender distribution of the EMO-DB dataset

6.6 Experimental Results on SER Tasks

The experiments conducted are described in Table 6.1. The RAVDESS dataset is used to conduct 6 class and 8 class SER tasks with proposed system. The reason for this is that the emotion of Neutral and Calm are considered as the same, which potentially corrupts the system performance. The feedforward SNN structure was also used to show the effectiveness of the proposed recurrent structure on the RAVDESS dataset. Besides, the system is applied to Emo-DB and eNTERFACE dataset to perform SER to show its generalization capability. Last, the speech enhancement technique from section 5.3 is applied to this system to enable the anti-noise ability of the system.

Table 6.1: Description of SRNN experiments

Dataset	**Description**
RAVDESS	8 class emotion recognition
RAVDESS	6 class emotion recognition
RAVDESS	SRNN vs feedforward
RAVDESS	Comparison to feedforward structure
Emo-DB	6 class emotion recognition
eNTERFACE	7 class emotion recognition
RAVDESS	6 class emotion recognition under the noisy enviorment

Recall the proposed spike generation methods illustrated in section 6.3, all of the speech signals are segments into sub-frames with 20ms hamming window and 10ms overlap between adjacent samples to well capture the temporal variations in the raw signal. The result segments are further transformed to spectrogram using the STFT. Then mel-scaled filter banks that non-linearly covers the frequency range from 100Hz to 5000Hz are applied to spectrogram. Subsequently, the first 23 MFCCs are taken from the DCT output of the mel-scaled power spectrum and converted to 23 channel spike representations by employing (6.1) with T = 100ms. This represents the spiking time that each coded spike resides in a 0-100ms window. An example of MFCC spectrum and the converted raster plot are shown in Figure 6.7 and Figure 6.8 respectively. By comparing these two figures it is noted that the MFCC features become hard to track after latency coding since all the information are transferred to the time of occurrence of each spikes. Nevertheless, these spikes can be used in the by proposed SRNN which contains strong temporal information across frequency bands.

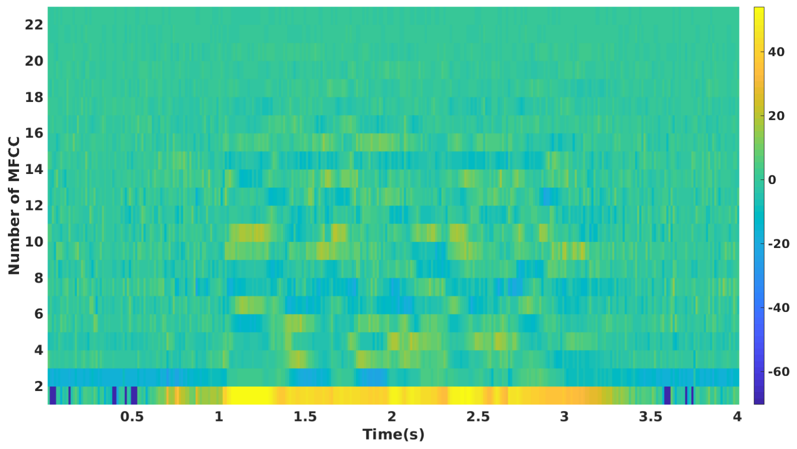


Figure 6.7: An example of MFCC spectrum. Samples taken from RAVDESS dataset('03-01-04-01-01-01-01.wav')

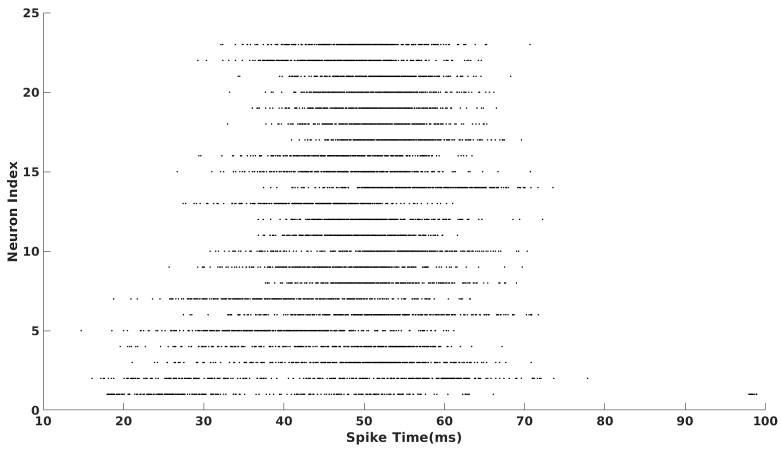


Figure 6.8: An example of result pre-processed spike output. Samples taken from RAVDESS dataset('03-01-04-01-01-01-01.wav')

To investigate the adaptability and capability of the proposed SRNN structure, a carefully designed SRNN is used for training three datasets. The input layer consists of spiking neurons which takes the coded spike train in a one to one fashion. In other words, each neuron receives the information only from one mel-scaled frequency bands.

The size of the output layer is equal to the number of the target emotion classes, which are only activated when the cell completes the propagation for the entire spike trains. The input spike train is split into N subset where each has segments of spike trains with a duration of $T = (100/N)$. It is important to highlight that the SRNN in the experiments was not unfolded to a feedward structure and this means the SRNN is sharing the same set of weights at each time step and samples. The simulation setting of the neuron parameters that used in SRNN is shown in Table 6.2.

Table 6.2: The neuron parameter setting for the SRNN SER task

ϑ	τ	τ_{ref}	C_{ref}	τ_f	C_f
10	10	1	2	1	1

where ϑ denotes the neuron firing threshold. τ represents the neuron time constant, τ_{ref} denotes the neuron refractory time constant, C_{ref} is the refractory response scaling coefficient, τ_f refer to the neuron spike function derivative time constant, and the C_f is the neuron spike function derivative scaling coefficient.

6.6.1 Eight Class SER for RAVDESS Dataset

To investigate the performance of the proposed SRNN structure, different network topologies were tested on the entire RAVDESS dataset with different choice of hidden layer size L_h, the number of input segments T and the number of MFCC coefficients N_{mfcc}, which is shown in Table 6.3.

Table 6.3: The SRNN SER experiments with different structure on RAVDESS dataset 8 class

N_{mfcc}	L_h	$T(ms)$	**Train Acc**	**Test Acc**
11	64	250	45.83%	48.61%
11	128	250	56.9%	53.81%
11	128	125	60.07%	57.64%
11	256	250	60.42%	55.21%
23	64	125	76.74%	58.33%
23	128	125	77.23%	59.31%
23	128	250	72.76%	59.72%
23	**256**	**125**	**84.13%**	**64.07%**

The 5-fold cross-validation is used in the experiment, which randomly partitions the entire dataset into 5 subsets with 4 of them used for training and the remaining one for testing. Thus the accuracy shown in the table is obtained by averaging the results from 5 experiments. It can be seen that the SRNN with the designed structure of $L_h = 128$, $T = 125ms$ and $N_{mfcc} = 23$ achieves the best resulting in a training accuracy of 84.12% and testing recognition accuracy of 69.07% for the entire dataset. Among these parameters, SRNN is especially sensitive to the number of MFCCs which determines the amount of input frequency-time features of speech where the SRNN running with 23 MFCCs yields a significant improvement in both training and test accuracy.

Apart from this, the results suggests that SRNN performs better with higher number of hidden neurons and segmentation time steps. This is due to the higher number of neurons which introduce additional learning capacity to the model, and the utilization of smaller event segments which can increases the temporal resolution for the input spikes trains.

The training accuracy curve, loss curve and the confusion matrix for the optimal run is shown in Figure 6.9 and Figure 6.10 respectively. It can be seen that the SRNN performs best on the Angry emotion which classifies 89% samples correctly and worst on Surprise wherein only correctly predicted 59% of the testing samples. Besides, Calm was very easily misclassified as Neutral, where 27% of Calm samples are recognized as Neutral.The correspond classification score of this experiment is shown in Table 6.4, including the precision, recall and f1 score for each class of the emotion according to the following equations:

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.4}$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6.5}$$

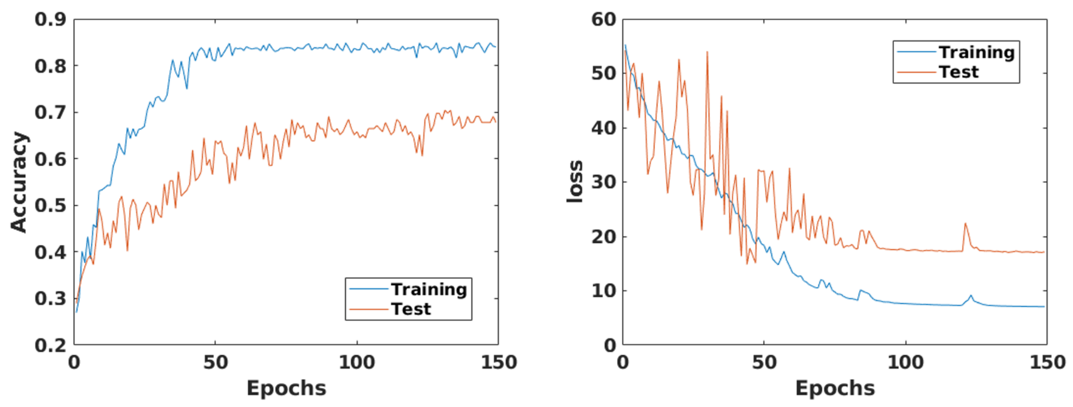
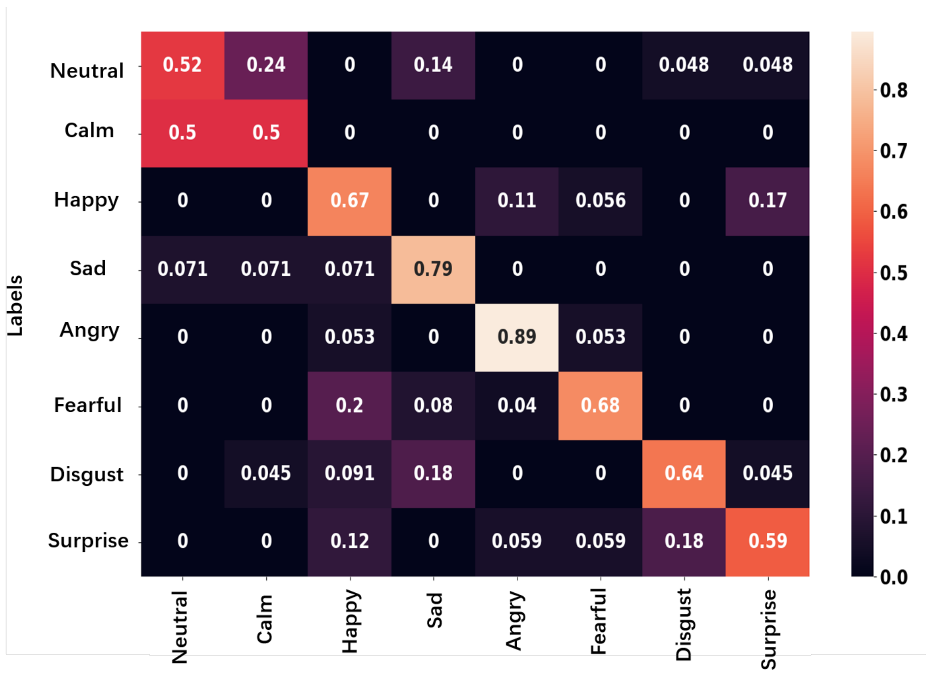


Figure 6.9: Training(a) and loss(b) curve for 8 class RAVDESS SER task



Labels	Neutral	Calm	Happy	Sad	Angry	Fearful	Disgust	Surprise
Neutral	0.52	0.24	0	0.14	0	0	0.048	0.048
Calm	0.5	0.5	0	0	0	0	0	0
Happy	0	0	0.67	0	0.11	0.056	0	0.17
Sad	0.071	0.071	0.071	0.79	0	0	0	0
Angry	0	0	0.053	0	0.89	0.053	0	0
Fearful	0	0	0.2	0.08	0.04	0.68	0	0
Disgust	0	0.045	0.091	0.18	0	0	0.64	0.045
Surprise	0	0	0.12	0	0.059	0.059	0.18	0.59

Figure 6.10: Confusion Matrix for RAVDESS SER 8 class with testing data

Table 6.4: Classification score accuracy table for SER on RAVDESS with 8 class

Emotion	**Precision**	**Recall**	**F1 Score**
Neutral	52.38	52.38	52.38
Calm	50	56.25	52.94
Happy	66.67	52.17	58.53
Sad	78.57	55	63.71
Angry	89.47	80.95	85
Fearful	68	85	71.56
Disgust	63.64	77.78	66.67
Surprise	58.82	66.67	62.5

6.6.2 Six Class SER for RAVDESS Dataset

Another experiment was carried out to compare the overall performance of the system with the work reported by Mansouri et al. in [281]. The SRNN is trained with only 6 class of emotions from RAVDESS dataset, which excludes Neutral and Calm emotions due to the consideration of the disturbances caused by Neutral and Calm. The network setting remains the same as the setting for the 8 class except the number of spiking neurons L_h in the hidden layer was set to 64 and the output layer was set to 6. The training curve and loss curve for this experiment is shown in Figure 6.11(a) and (b) respectively. It can be seen that the SRNN performs more effectively on 6 classes SER without the disturbances of Neutral and Calm, which offers a testing accuracy of 92.85% over the 6 class emotion recognition task. The SRNN performs perfectly on the Angry and Fearful with 100% recognition accuracy and weakly on the Disgust and Surprise emotion. The confusion matrix is demonstrated in Figure 6.12 and the classification score table is shown in Table 6.5.

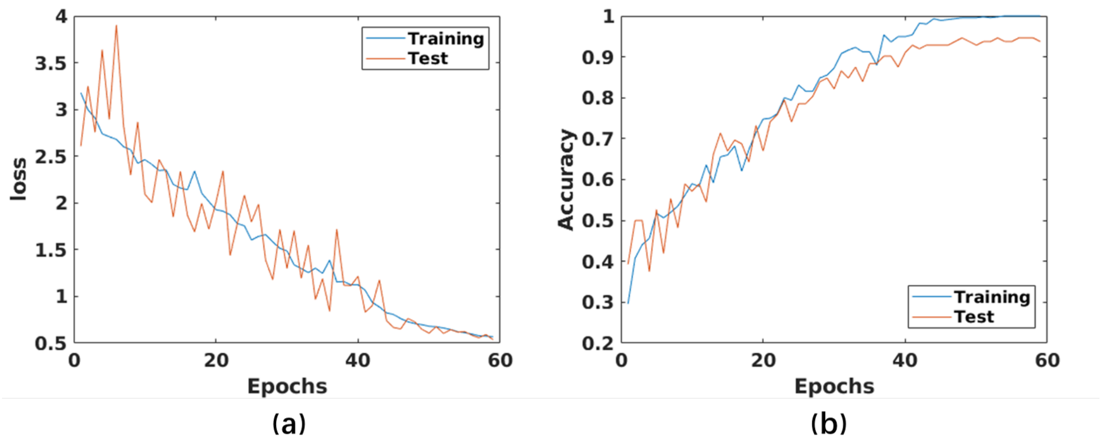


Figure 6.11: Training(a) and loss(b) curve for 6 class RAVDESS SER task

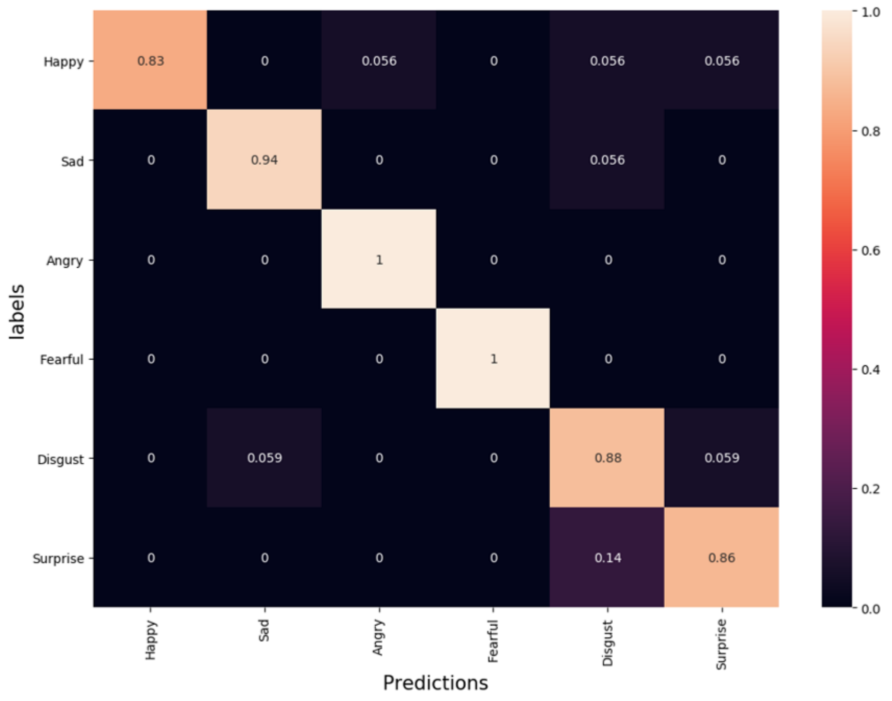


Figure 6.12: Confusion Matrix for RAVDESS 6 class SER with testing data

Table 6.5: Classification score table for 5 class SER on RAVDESS with

Emotion	**Precision**	**Recall**	**F1 Score**
Happy	83.33%	100%	90.90%
Sad	94.44%	94.44%	94.44%
Angry	100%	94.44%	97.14%
Fearful	100%	100%	100 %
Disgust	88.24%	75%	81.08%
Surprise	86.36%	90.48%	88.37%

6.6.3 Effect of Recurrent Connection

To access the effectiveness of the recurrent structure used in SRNN, a feedforward SNN is trained to directly compare the results with SRNN. This experiment uses exactly the same number of spiking neurons as SRNN, with a input layer contains 23 neurons, a hidden layer with 256 neurons and an output layer with 6 spiking neurons. With the same neuron settings as shown in Table 6.2, training algorithms and the MFCC based AER input, the feedforward structure only provides a test accuracy of 66.36% which is approximately 26.5% lower than use the recurrent structure. In fact, the recurrent connectivity enables the extracted spiking patterns to be accumulated along with the temporal axis. At each time step, the SCRNN integrates the PSPs from current input segments and the PSPs that is accumulated and processed over all the previous time steps. From the viewpoint of feature extraction(MFCCs), the SCRNN actually is constructing and learning the temporal relations that is represented with spike trains, for each frequency bands independently. It then summarizes the information across all specified frequency bands with a fully connected output layer.

6.6.4 Experiment on Emo-DB and eNTERFACE datasets

To further explore the universality of SRNN on the SER task, the SRNN was also trained with eNTERFACE and EMO-DB emotional dataset with exactly the same neuron parameter setting as used for RAVDESS dataset. The 5-fold cross validation method is applied to both dataset. The training dynamics and the confusion matrix for the eNTERFACE dataset and EMO-DB are shown in Figure 6.13 to Figure 6.16.

Table 6.6 illustrates the network settings for the conducted experiments.

For EMO-DB dataset, considering the limited amount of samples and unbalanced data distribution, the SRNN is constructed with 64 hidden neurons and input segments length of 10 with the MFCC feature size of 13. The experiment results yield an average testing accuracy of 65.12% with a best testing accuracy of 74.05% for 7 class emotion recognition. The confusion matrix in Figure 6.16 shows that the SCRNN performs well on Bored and Sad and with precision of 78% and 80% respectively. One of reasons of this is due to the fact that Bored and Sad samples in the EMO-DB are found to be distincly different from other types of emotions in terms of volume, intonation and speech rate. The number of Angry samples in EMO-DB as described in Figure 6.6 are approximately two times more than other types of emotions, which can be a reason that the model is over-fitted for Angry and it can be seen that 17% of Disgust, 23% of Fearful and 27% of Happy were classified falls into Angry. The model perform worst for the Neutral with only an accuracy of 57%.

For eNTERFACE dataset, SRNN was constructed with 128 hidden neurons using a segments amount of 8 with first 23 MFCCs, an averaging testing accuracy of 61.34% with a best testing accuracy of 69.84% and was obtained for 6 class emotion recognition. SRNN provided an accuracy of 70% for Disgust, 71% for Fearful and worst on the Surprise and Happy with only 50% and 42% respectively. Although the eNTERFACE provides an almost balanced number of emotions in each category, the utterance durations in the dataset vary from 1.12s to 106.92s, which introduced significant interference to the SRNN due to the emotion states in a long duration sample are usually non-stationary and it is difficult to assign the prediction as only one category.

Table 6.6: The SRNN experiment setup for EMO-DB and eNTERFACE dataset

Dataset	L_h	T	N_{mfcc}
EMO-DB	64	10	13
eNTERFACE	128	8	23

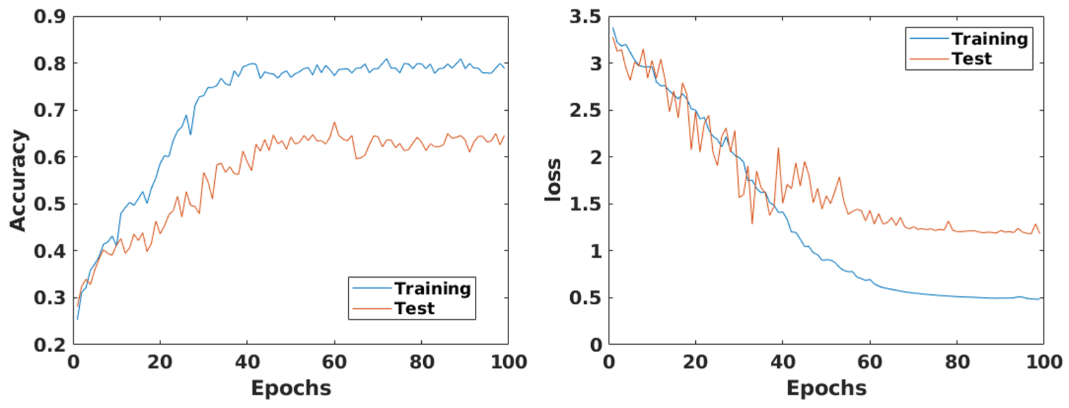
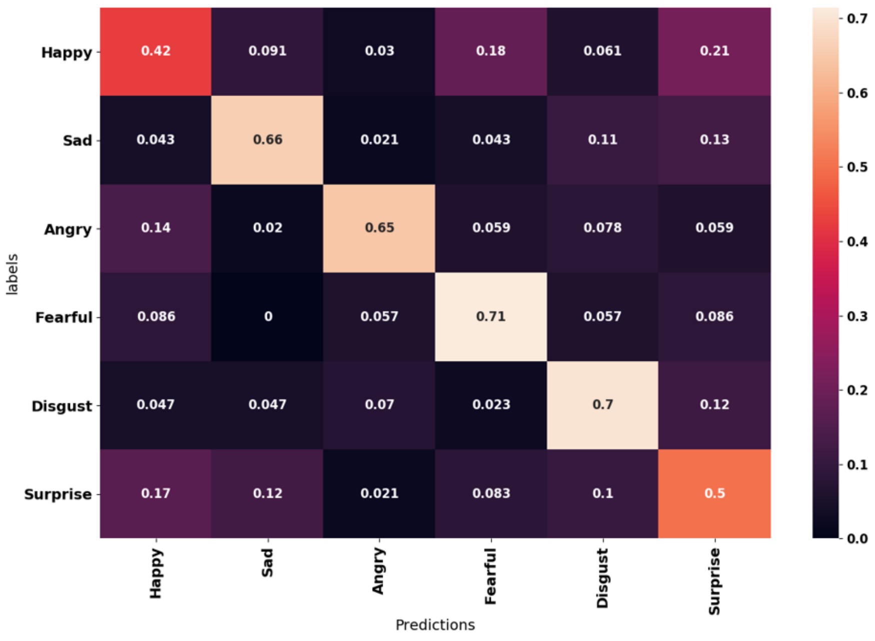


Figure 6.13: Training(a) and loss(b) curve for 6 class eNTERFACE SER



labels	Happy	Sad	Angry	Fearful	Disgust	Surprise
Happy	0.42	0.091	0.03	0.18	0.061	0.21
Sad	0.043	0.66	0.021	0.043	0.11	0.13
Angry	0.14	0.02	0.65	0.059	0.078	0.059
Fearful	0.086	0	0.057	0.71	0.057	0.086
Disgust	0.047	0.047	0.07	0.023	0.7	0.12
Surprise	0.17	0.12	0.021	0.083	0.1	0.5

Figure 6.14: confusion matrix for 6 class eNTERFACE SER

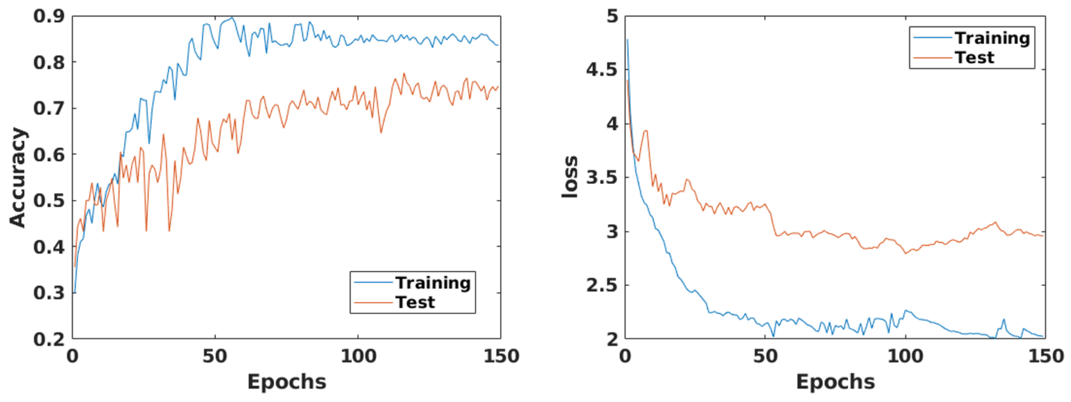
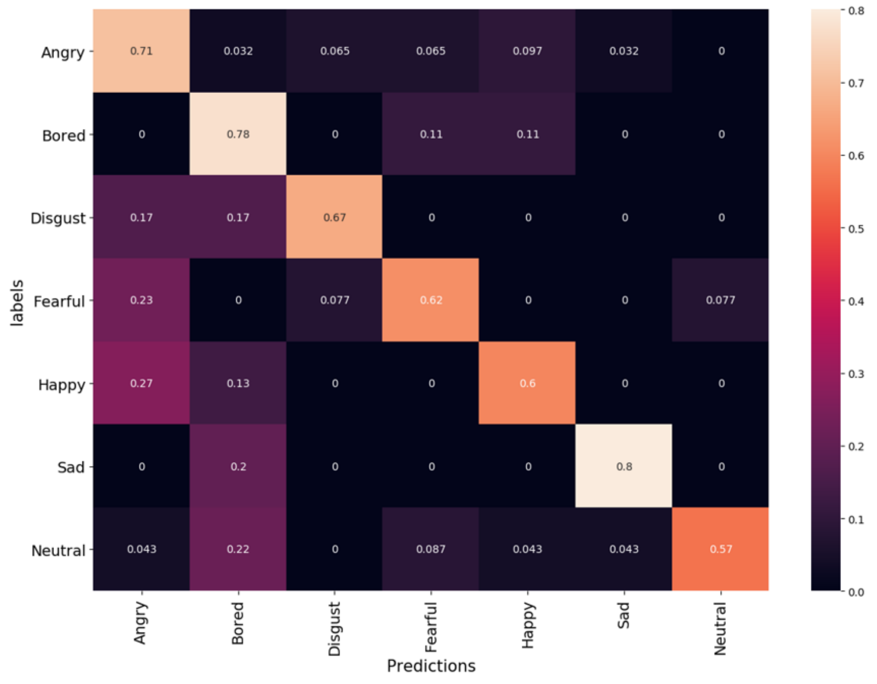


Figure 6.15: Training(a) and loss(b) curve for 7 class EMO-DB SER



labels \ Predictions	Angry	Bored	Disgust	Fearful	Happy	Sad	Neutral
Angry	0.71	0.032	0.065	0.065	0.097	0.032	0
Bored	0	0.78	0	0.11	0.11	0	0
Disgust	0.17	0.17	0.67	0	0	0	0
Fearful	0.23	0	0.077	0.62	0	0	0.077
Happy	0.27	0.13	0	0	0.6	0	0
Sad	0	0.2	0	0	0	0.8	0
Neutral	0.043	0.22	0	0.087	0.043	0.043	0.57

Figure 6.16: Confusion matrix for 7 class EMO-DB SER

6.7 Comparison to Previous SER Works

Table 6.7 illustrates a comparison of the SRNN with previous works. It can be seen that most previous SER works are based on conventional machine learning or deep learning methods. Many additional manually selected features have been used such as continuous wavelet transform(CWT) in [282], spectrograms in [283], MFCCs [284]. For the RAVDESS dataset, the SRNN achieves 64.07% which outperforms some conventional method based work such as the CWT based SVM model [282] and capsule CRNN in [285]. The work presented in [284] provided an accuracy of 75.69%, however they manually extracted and selected 183 different input features such as MFCCs, MEDCs, Energy, Prosodic, Spectral features, while SRNN only uses the first 23 order of MFCCs. Zeng et al. [286] employed spectrogram and a gated residual convolutional neural network to reduce the SER task as an image classification task. Each input audio file in their work was transformed to a spectrogram with dimension of 257x399 as the input feature to the network. The SRNN is still competitive to this work with only 128 neurons and the input feature size of approximately 23x229. Noticeably, the BLSTM network proposed in [275] offers the best recognition accuracy for 8 class SER task on RAVDESS dataset. This work requires multiple learning and processing techniques include K-means, STFT, Resnet101 and BLSTMs to achieve the 77.02%, while SRNN compare to this work has the significant advantage in computational and energy costs. It is important to note that compared to the only previous SNN based SER system in [239], SRNN outperforms the state-of-art 6 class RAVDESS SER test accuracy by approximately 12.55%.

Table 6.7: The comparison of SER previous works with the performance of SRNN

Method	Type	Dataset	No.Class	Accuracy
CWT, prosodic cofficient +SVM [282]	Non-Spiking	RAVDESS	8	60.1%
Features + bagged SVMs [284]	Non-Spiking	RAVDESS	8	75.69%
Spectrograms + DNN [286]	Non-Spiking	RAVDESS	8	64.52%
Features + BLSTM [275]	Non-Spiking	RAVDESS	8	77.02%
Spectrogram + capsule CRNN [285]	Non-Spiking	RAVDESS	8	56.2%
MFCC +Feedforward SNN +STDP [239]	Spiking	RAVDESS	6	80.3%
MFCC+SRNN+SLAYER	**Spiking**	**RAVDESS**	**6**	**92.85%**
MFCC+SRNN+SLAYER	**Spiking**	**RAVDESS**	**8**	**69.07%**
Spectrogram + CNN [287]	Non-Spiking	EMO-DB	7	85.2%
Spectrogram + 1D2DCNN [283]	Non-Spiking	EMO-DB	7	82.41%
Spectrograms + BLSTM [275]	Non-Spiking	EMO-DB	7	85.57%
Spectrograms + CNN+BLSTM [288]	Non-Spiking	EMO-DB	7	88.01%
Spectrograms + MultitimeCNN [289]	Non-Spiking	EMO-DB	7	70.97%
MFCC+SRNN+SLAYER	**Spiking**	**EMO-DB**	**7**	**74.05%**
MFCC+RF [290]	Non-Spiking	eNTERFACE	6	47.1%
Spectrogram + 3DCNN + Kmeans [291]	Non-Spiking	eNTERFACE	6	72.33%
Sparse autoencoder [292]	Non-Spiking	eNTERFACE	6	59.1%
Feature selection + MLP [293]	Non-Spiking	eNTERFACE	6	69.23%
MFCC +Feedforward SNN +STDP [239]	Spiking	eNTERFACE	6	72.2%
MFCC+SRNN+SLAYER	**Spiking**	**eNTERFACE**	**6**	**69.84%**

To the best of the author's knowledge, there were no SNN based SER baseline for EMO-DB dataset. Therefore, SRNN is compared with several deep learning based methods which has advantages in terms of float-point based processing and significantly larger network learning capacity. Despite these disadvantages, the SRNN is still competitive with a best accuracy of 74.05% with only 64 hidden neurons with 13 MFCCs and it outperformed the performance of the multi-time scale CNN structure in [289]. Among the SER research for the eNTERFACE dataset, SRNN still was able to cope with the unbalanced dataset with an accuracy of 69.84% which outperforms the performance of random forest based work in [290], the sparse autoencoder in [292] and the MLP in [293]. Although from the recognition accuracy perspective the SRNN does not achieve the best results among all the previous works, section 6.6.1 has shown a possibility that the performance of SRNN can be increased by increasing the network capacity or reducing the segmentation steps.

It is also important to highlight that the SRNN has a strong potential to be applied in to the Loihi NM chip since the use of SLAYER training algorithm, whereas none of the above listed conventional machine learning or deep learning based method has such advantage.

Most ANN/DNNs have a fixed amount of neurons which can only receive features(e.g. spectrogram) with a fixed dimension. A common method for this is to resize or re-sample all the speech samples to same length or use different size of STFT windows to force a equivalent dimension for the spectrogram. Another significant advantage of SRNN based SER system is that it can takes the input with arbitrary length since all the information will be coded into spikes in a fixed size window (such as 100ms). This efficiently prevents information loss due to the prior pre-processing stage.

Most importantly, the SRNN approached the state-of-art recognition accuracy on the SER tasks with only 3 layer of binary spiking neurons. The SRNN with 13-128-6 structure only has (128+7) neurons with 2432 learnable parameters, which is over 40 times smaller compare to the work [283] which uses parallel 1D-2D CNN-LSTM structure and multiple local feature learning blocks(LFLBs), approximately have over 1.1x10^5 free parameters. The size of the SRNN is almost equal to only the last classification layer in the conventional complex ANN/DNN architecture. With the characteristic of event-driven processing and NM hardware applicable, the SRNN is especially significant for SWaP based HCI systems.

6.8 Spiking SER under White Noise

In this section, the lateral inhibition based speech enhancement algorithm that proposed in Chapter 5.3 is applied to the developed SRNN SER system to construct an anti-noise SER system. As illustrated, the lateral inhibition can efficiently remove the spike trains that has a lower firing rate. The de-noising section is designed to use the rate coded spectrogram which is the intermedia between the MFCC and raw speech signal. The process of the speech enhancement is developed as a part of spike generation stage which is shown in Figure 6.17.

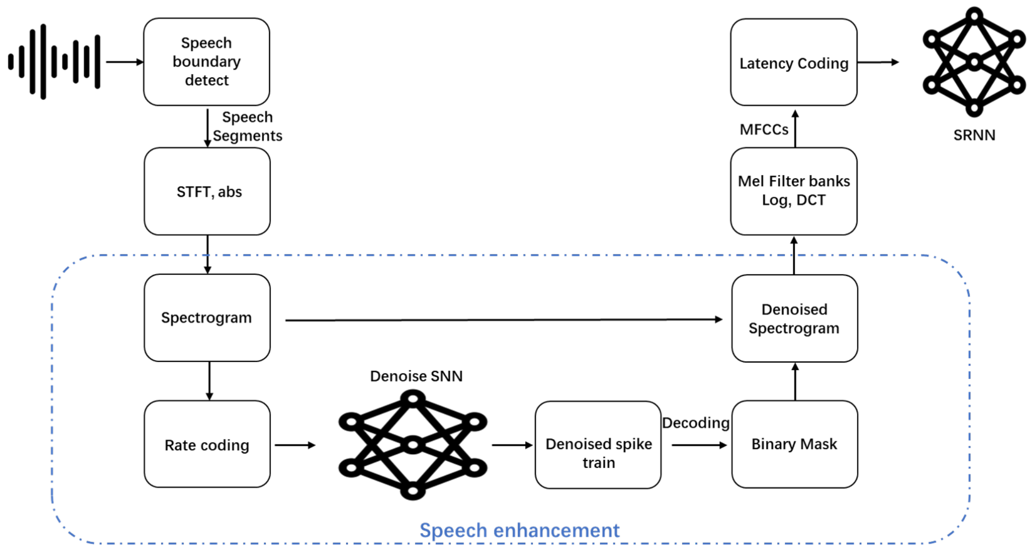


Figure 6.17: The proposed Speech enhancement spike generation algorithms that used for SRNN SER

From Figure 6.17 it can be seen that, the input speech signal is firstly fed to a speech boundary detection algorithm. Due to the limitation of the proposed speech enhancement algorithm described in section 5.3.7 where the algorithms is speech presence dependent, the detection algorithm is implemented to reduce the unrelated interference with the '*detectSpeech*' function in MATLAB audio toolbox [294]. All the speech corpusses are cropped to remove the silence (the time period without presence of speech) from raw signals to improve the speech enhancement performance by using the detected speech time-domain boundaries.

Then the cropped speech signals are processed following the preprocessing stages described in section 6.3 with the speech enhancement stage are inserted before applying Mel-scaled filter banks. The spectrogram generated by the signal is taken as a reference to construct the denoised spectrogram with the binary mask. Due to the fact that the SER process does not need to use the time domain speech signal, the signal reconstruction stage that is required in original speech enhancement algorithm is omitted and the denoised spectrogram is directly connected to the preprocessing stage to ultimately produce the latency coded MFCC.

The SRNN with speech enhancement algorithm is tested with the 6 class RAVDESS dataset to demonstrate the usefulness of the method. The clean speech corpus in the dataset are firstly corrupted with different levels of Gaussian white noise to generate noisy test samples which has measured SNR of 0.1,5 and 10dBs. The noisy test samples are fed into the well-trained SRNN with and without the speech enhancement section respectively to obtain the recognition result which is shown in Table 6.8.

Table 6.8: The speech enhancement experiment results for SRNN based SER(RAVDESS 6 Class)

SNR	Without speech enhancement	With speech enhancement	Improvement
clean speech	92.85%	N/A	N/A
0.1	53.04%	58.61%	5.57%
5	51.30%	55.46%	4.16%
10	63.41%	67.74%	4.33%

It can be seen that the speech enhancement algorithm successfully increases the SER accuracy under the noisy environment. The results suggest that the system has an average SER improvement of 4.69% under the noisy environment.

However, the method in SER system requires two forms of neural spike coding (one rate coding, one latency coding) operation and one decoding and mapping operation, which potentially causes information loss in the spectrogram and degrades the performance of the SRNN. Despite the limit improvement of accuracy, it is still noteworthy that the proposed SNN speech enhancement algorithm does not require prior knowledge to the speech signal as well as training process. The method acts as a non-linear asynchronous event-based filter which can remove the spikes which are not densely packed to preserve the high frequency spike trains.

6.9 Conclusion

In this chapter, a novel SER system was presented along with a design of novel preprocessing algorithm, a spiking recurrent neural network structure and a signal denoising SER strategy. The proposed system has been validated using open public SER datasets and achieving a state-of-art result on both accuracy and computational efficiency. The systems achieves 8 class SER accuracy of 64.07% and 6 class accuracy of 92.85% on RAVDNESS dataset, 74.05% on EMO-DB dataset and 69.84% on the eNTERFACE dataset. In addition, the speech processing algorithms presented in Chapter 5 are combined with this work to further improve the performance of the system under the noisy environment that achieves an average SER accuracy improvement of 4.69%.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

Throughout this thesis, a range of novel event-driven processing techniques and SNNs have been proposed to address the concept of neuromorphic human gesture recognition. The novel research especially covers the field of visual hand gesture recognition, audio speech emotion gesture recognition and the event-driven speech processing algorithms. These algorithms have strong potentials contributes to the engineering field espicially in the applications of human gesture recognitions. With the development of neuromorphic processors and sensors, the contributions in this work can be transformmed to the applications that adavanced in terms of extremely low power and low latency.

In Chapter 2, a review of conventional neural network and its applications on gesture recognition was provided. The concept of ANNs and its working mechanism was extensively reviewed which provide the foundation of the new generation SNNs. The noticeable structure of ANNs that is significant in human gesture recognition is discussed in detail with several examples of previous works, which includes convolutional neural network, recurrent neural network and convolutional recurrent neural network. CNNs are especially effective for the visual gesture recognition tasks with static frame input. The RNN uses recurrent connectively is well suited to those audio gesture recognition tasks that has a 1D sequential input. The CRNN combines the advantages over CNN and RNN which is found to be useful to solve the video based dynamic gesture recognition tasks. The concepts reviewed in this chapter significantly inspired the novel designs which enables these remarkable structure in SNN domain for human gesture recognition.

Chapter 3 presented a review of neuromorphic(NM) technology, which covered the aspects of spiking neural networks(SNNs), NM software, NM hardware and previous research on SNN based human gesture recognition. The concept of SNN is explained in detail that included spiking neuron models, neural coding strategies, learning algorithms, which are fundamentally different from the traditional statistics based ANNs. In addition, a variety of state-of-the-art NM hardware were illustrated with their different capabilities and corresponding SNN software implementation adaptabilities. Furthermore, the recent works of SNN based human gesture recognition in both visual and auditory domain were provided and discussed.

Chapter 4 proposed a novel spiking convolutional recurrent neural network(SCRNN) that was specially designed to solve the visual based hand gesture recognition challenge. The proposed SCRNN transferred the idea of CRNN into event-driven processing domain to provide an SNN spiking model that can effectively learn the spatio-temporal pattern in the dynamic gesture AER data. The correspond design of fundamental units in SCRNN such as 3D spiking convolution operation and individual SCRNN cell were explained. Moreover, the model was extensively analysed and discussed in terms of the network topology, training setting and dynamic behaviours. The model was positively validated by a series of experiments on IBM DVSgesture dataset and performed a state-of-art hand gesture recognition accuracy.

Novel event-based speech processing algorithms that potentially contribute to the audio speech gesture recognition were presented in Chapter 5. The first method presented in this chapter consists of a method for bio-inspired neural coding for audio signals. This method provided a possible route that bridges the standard digital speech raw signals to event-based one. The results of this algorithm were compared to the dynamic audio sensor(DAS) which shows it can offer a DAS equivalent output spiking signal. The second algorithm presented in Section 5.3 was a speech enhancement algorithm which was build upon the upgraded version of speech coding algorithm. The spike coding strategy in this method was simplified with STFT and the use of LIF neurons. With a proper design of lateral inhibition connection between layers of spiking neurons, the method successfully provides an average SNR improvement of 10.915dB up to 6 types of noise.

In Chapter 6, a novel end-to-end spiking speech emotion recognition(SER) system was proposed. The systems consists of a novel preprocessing algorithm and a novel spiking recurrent neural network(SRNN). In addition, the speech enhancement algorithm developed in Chapter 5 was additionally embedded in this SER system that enabled a signal denoising feature to this method. The SRNN was firstly tested on 3 different SER datasets and resulted in a state-of-art recognition accuracy where it provides an accuracy of 69.07% for RAVDESS 8 class, 92.85% for 6 class, 74.05% on EMO-DB and 69.84% on eNTERFACE. The significance of SRNN in low power, energy efficient NM SER was discussed in a detail. The speech enhancement algorithm combined in the proposed system was able to provide an average recognition improvement of 4.69% under the interference of vary level of Gaussian white noises.

7.2 Future Work

The work presented throughout this thesis has a wide range of directions that can be further explored for future research.

A. *Transfer algorithms to NM hardware*

Both of the algorithm SCRNN(Chapter 4) for hand gesture recognition and SRNN for SER systems(6) are SNNs that were developed under the SLAYER spiking layer model which could be implemented on Intel Loihi chip. The proposed human gesture recognition algorithms thus have a large potential to be implemented as an offline well trained inference model on NM hardware such as SpiNNaker, TrueNorth and especially Loihi chip. Particularly, the SCRNN can directly take the NM sensor DVS input without any preprocessing stage, which is envisaged to create a fully event based system.

B. *Automated network parameter tuning*

The novel speech enhancement algorithms that was proposed in section 5.3 at present still needs to manually identify the inhibitory strength and radius. This is a time consuming and repetitive task where a automated way of parameter tuning method is needed to improve the convenience of this technique. Further more, the speech enhancement algorithms can potentially be tested for additional variety of noises.

C. *Applying the algorithms to other applications*

The SNN algorithms presented in this thesis although have demonstrated promising results in human gesture recognition in particular visual hand gestures and speech emotion gestures, there is still a need to further investigated the applications of these algorithms in other event-based applications such as SRNN for speech recognition, SCRNN for dynamic scene recognition. These and other potential applications of SNN have been left for future work.

D. *Use of further feature for SER*

The SRNN algorithm presented in Chapter 6 efficiently used the MFCC as the main prior feature extractor. However, there are still a range of speech features can potentially contributes to the improvement of system. Future research may include the applications of other features such as CWT and frequency band energy(FBE) in the recognition tasks by using SRNN.

Bibliography

[1] A. Deshpande, "A Beginner's Guide To Understanding Convolutional Neural Networks Adit Deshpande Engineering at Forward — UCLA CS '19," 2016. [Online]. Available: https://adeshpande3.github.io/A-Beginner{%}27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[2] F. Deloche, "File:Recurrent neural network unfold.svg - Wikimedia Commons," 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:Recurrent{_}neural{_}network{_}unfold.svg

[3] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, 2020.

[4] M. Dong, X. Huang, and B. Xu, "Unsupervised speech recognition through spike-timing-dependent plasticity in a convolutional spiking neural network," *PLoS ONE*, vol. 13, no. 11, 2018.

[5] "Stereo Vision Using Computing Architecture Inspired by the Brain." [Online]. Available: https://www.ibm.com/blogs/research/2018/06/stereo-vision/

[6] "Dynamic Audio Sensor — iniLabs." [Online]. Available: https://inilabs.com/products/dynamic-audio-sensor/

[7] Y. Xing, P. Kirkland, G. Di Caterina, J. Soraghan, and G. Matich, *Real-time embedded intelligence system: Emotion recognition on raspberry Pi with intel NCS*, 2018, vol. 11139 LNCS.

[8] G. Sandbach, S. Zafeiriou, M. Pantic, and L. Yin, "Static and dynamic 3D facial expression recognition: A comprehensive survey," *Image and Vision Computing*, vol. 30, no. 10, pp. 683–697, 2012.

[9] S. Kumar, M. K. Bhuyan, and B. K. Chakraborty, "Extraction of informative regions of a face for facial expression recognition," *IET Computer Vision*, vol. 10, no. 6, pp. 567–576, 2016.

[10] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer vision and image understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[11] M. Milders, J. K. Hietanen, J. M. Leppänen, and M. Braun, "Detection of Emotional Faces Is Modulated by the Direction of Eye Gaze," *Emotion*, vol. 11, no. 6, pp. 1456–1461, 2011.

[12] D. W. Hansen and Q. Ji, "In the Eye of the Beholder: A Survey of Models for Eyes and Gaze," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 478–500, 2010.

[13] M. El Ayadi, M. S. Kamel, and F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognition*, vol. 44, no. 3, pp. 572–587, 2011.

[14] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12–40, 2010.

[15] E. Cakir, 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.

[16] Y. Liu, O. Sourina, and M. K. Nguyen, "Real-time EEG-based emotion recognition and its applications," in *Lecture Notes in Computer Science (including*

subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6670 LNCS, 2011, pp. 256–277.

[17] W. Ke, Y. Xing, G. Di Caterina, L. Petropoulakis, and J. Soraghan, "Intersected EMG Heatmaps and Deep Learning Based Gesture Recognition," in *ACM International Conference Proceeding Series*, 2020, pp. 73–78.

[18] D. Wickeroth, P. Benölken, and U. Lang, "Markerless gesture based interaction for design review scenarios," in *2nd International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2009*, 2009, pp. 682–687.

[19] V. Frati and D. Prattichizzo, "Using Kinect for hand tracking and rendering in wearable haptics," in *2011 IEEE World Haptics Conference, WHC 2011*, 2011, pp. 317–321.

[20] D. Droeschel, J. Stückler, and S. Behnke, "Learning to interpret pointing gestures with a time-of-flight camera," in *HRI 2011 - Proceedings of the 6th ACM/IEEE International Conference on Human-Robot Interaction*, 2011, pp. 481–488.

[21] H. Liu and L. Wang, "Gesture recognition for human-robot collaboration: A review," *International Journal of Industrial Ergonomics*, vol. 68, pp. 355–367, 2018.

[22] L. Pigou, S. Dieleman, P. J. Kindermans, and B. Schrauwen, "Sign language recognition using convolutional neural networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8925, 2015, pp. 572–578.

[23] R. H. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in *Proceedings - 3rd IEEE International Conference on Automatic Face and Gesture Recognition, FG 1998*, 1998, pp. 558–567.

[24] L. L. Grewe and C. Hu, "ULearn: understanding and reacting to student frustration using deep learning, mobile vision and NLP," 2019, p. 30.

[25] U. Fiore, A. Florea, and G. Pérez Lechuga, "An interdisciplinary review of smart vehicular traffic and its applications and challenges," *Journal of Sensor and Actuator Networks*, vol. 8, no. 1, p. 13, 2019.

[26] Mustaqeem and S. Kwon, "A CNN-assisted enhanced audio signal processing for speech emotion recognition," *Sensors (Switzerland)*, vol. 20, no. 1, 2020.

[27] S. Kim, S. J. Guy, K. Hillesland, B. Zafar, A. A. A. Gutub, and D. Manocha, "Velocity-based modeling of physical interactions in dense crowds," *Visual Computer*, vol. 31, no. 5, pp. 541–555, 2015.

[28] Q. Mao, M. Dong, Z. Huang, and Y. Zhan, "Learning salient features for speech emotion recognition using convolutional neural networks," *IEEE Transactions on Multimedia*, vol. 16, no. 8, pp. 2203–2213, 2014.

[29] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout, "Multi-scale deep learning for gesture detection and localization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8925, 2015, pp. 474–490.

[30] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Multi-sensor system for driver's hand-gesture recognition," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, FG 2015*, 2015.

[31] "Deep learning has a size problem. Shifting from state-of-the-art accuracy… — by Jameson Toole — Heartbeat." [Online]. Available: https://heartbeat.fritz.ai/deep-learning-has-a-size-problem-ea601304cd8

[32] R. Massa, A. Marchisio, M. Martina, and M. Shafique, "An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor," *arXiv preprint arXiv:2006.09985*, 2020.

[33] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "TrueNorth:

Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[34] C. Brandli, R. Berner, M. Yang, S. C. Liu, and T. Delbruck, "A 240 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[35] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[36] A. Amir, B. Taba, D. Berg, T. Melano, J. Mckinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, 2017, pp. 7388–7397.

[37] J. Botzheim, T. Obo, and N. Kubota, "Human gesture recognition for robot partners by spiking neural network and classification learning," in *6th International Conference on Soft Computing and Intelligent Systems, and 13th International Symposium on Advanced Intelligence Systems, SCIS/ISIS 2012*, 2012, pp. 1954–1958.

[38] L. Cheng, Y. Liu, Z. G. Hou, M. Tan, D. Du, and M. Fei, "A Rapid Spiking Neural Network Approach with an Application on Hand Gesture Recognition," *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

[39] Y. Liuy and L. Chengy, "Spiking-Neural-Network Based Fugl-Meyer Hand Gesture Recognition for Wearable Hand Rehabilitation Robot," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2018-July, 2018.

[40] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-13, no. 5, pp. 826–834, 1983.

[41] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[42] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2015-Octob, 2015, pp. 1–7.

[43] H. J. Kim, J. S. Lee, and J. H. Park, "Dynamic hand gesture recognition using a CNN model with 3D receptive fields," in *2008 IEEE International Conference Neural Networks and Signal Processing, ICNNSP*, 2008, pp. 14–19.

[44] D. Hamester, P. Barros, and S. Wermter, "Face expression recognition with a 2-channel Convolutional Neural Network," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2015-Septe, 2015.

[45] G. Batchuluun, R. A. Naqvi, W. Kim, and K. R. Park, "Body-movement-based human identification using convolutional neural network," *Expert Systems with Applications*, vol. 101, pp. 56–77, 2018.

[46] K. Wang, J. He, and L. Zhang, "Attention-based convolutional neural network for weakly labeled human activities' recognition with wearable sensors," *IEEE Sensors Journal*, vol. 19, no. 17, pp. 7598–7604, 2019.

[47] S. Minaee and A. Abdolrashidi, "Deep-emotion: Facial expression recognition using attentional convolutional network," *arXiv preprint arXiv:1902.01019*, 2019.

[48] T. S. Kim and A. Reiter, "Interpretable 3D Human Action Analysis with Temporal Convolutional Networks," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, 2017, pp. 1623–1631.

[49] Y. Liao, P. Xiong, W. Min, W. Min, and J. Lu, "Dynamic Sign Language Recognition Based on Video Sequence with BLSTM-3D Residual Networks," *IEEE Access*, vol. 7, pp. 38 044–38 054, 2019.

[50] M. Maghoumi and J. J. LaViola Jr, "Deepgru: Deep gesture recognition utility," in *International Symposium on Visual Computing.* Springer, 2019, pp. 16–31.

[51] G. Storey, R. Jiang, S. Keogh, A. Bouridane, and C.-T. Li, "3dpalsynet: a facial palsy grading and motion recognition framework using fully 3d convolutional neural networks," *IEEE access*, vol. 7, pp. 121 655–121 664, 2019.

[52] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3697 LNCS, 2005, pp. 799–804.

[53] M. K. Nammous and K. Saeed, "Natural language processing: Speaker, language, and gender identification with LSTM," in *Advances in Intelligent Systems and Computing*, vol. 883, 2019, pp. 143–156.

[54] Y. Xie, R. Liang, Z. Liang, C. Huang, C. Zou, and B. Schuller, "Speech Emotion Classification Using Attention-Based LSTM," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 27, no. 11, pp. 1675–1685, 2019.

[55] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions," 2019, pp. 3785–3789.

[56] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, and A. Waibel, "Very Deep Self-Attention Networks for End-to-End Speech Recognition," 2019, pp. 66–70.

[57] J. W. Choi, S. J. Ryu, and J. H. Kim, "Short-Range Radar Based Real-Time Hand Gesture Recognition Using LSTM Encoder," *IEEE Access*, vol. 7, pp. 33 610–33 618, 2019.

[58] M. Kim, J. Cho, S. Lee, and Y. Jung, "Imu sensor-based hand gesture recognition for human-machine interfaces," *Sensors (Switzerland)*, vol. 19, no. 18, 2019.

[59] A. R. Asif, A. Waris, S. O. Gilani, M. Jamil, H. Ashraf, M. Shafique, and I. K. Niazi, "Performance evaluation of convolutional neural network for hand gesture recognition using EMG," *Sensors (Switzerland)*, vol. 20, no. 6, 2020.

[60] Z. Zhang, K. Yang, J. Qian, and L. Zhang, "Real-time surface EMG pattern recognition for hand gestures based on an artificial neural network," *Sensors (Switzerland)*, vol. 19, no. 14, 2019.

[61] W. S. Noble, "What is a support vector machine?" pp. 1565–1567, 2006.

[62] S. Singh, J. Haddon, and M. Markou, "Nearest-neighbour classifiers in natural scene analysis," *Pattern Recognition*, vol. 34, no. 8, pp. 1601–1612, 2001.

[63] A. S. Vyas, H. B. Prajapati, and V. K. Dabhi, "Survey on face expression recognition using cnn," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE, 2019, pp. 102–106.

[64] C. Avilés-Cruz, A. Ferreyra-Ramírez, A. Zúñiga-López, and J. Villegas-Cortéz, "Coarse-fine convolutional deep-learning strategy for human activity recognition," *Sensors (Switzerland)*, vol. 19, no. 7, 2019.

[65] A. Kar, N. Rai, K. Sikka, and G. Sharma, "AdaScan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, 2017, pp. 5699–5708.

[66] E. P. Ijjina and K. M. Chalavadi, "Human action recognition using genetic algorithms and convolutional neural networks," *Pattern Recognition*, vol. 59, pp. 199–212, 2016.

[67] G. A. Rao, K. Syamala, P. V. Kishore, and A. S. Sastry, "Deep convolutional neural networks for sign language recognition," in *2018 Conference on Signal Processing And Communication Engineering Systems, SPACES 2018*, vol. 2018-Janua, 2018, pp. 194–197.

[68] L. Dovydaitis and V. Rudžionis, "Building LSTM neural network based speaker identification system," *Computational Science and Techniques*, vol. 6, no. 1, 2018.

[69] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[70] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[71] M. Nikzad, K. Movagharnejad, and F. Talebnia, "Comparative Study between Neural Network Model and Mathematical Models for Prediction of Glucose Concentration during Enzymatic Hydrolysis," *International Journal of Computer Applications*, vol. 56, no. 1, pp. 43–48, 2012.

[72] H. J. KELLEY, "Gradient Theory of Optimal Flight Paths," *ARS Journal*, vol. 30, no. 10, pp. 947–954, 1960.

[73] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An introductory review of deep learning for prediction models with big data. front," *Artif. Intell*, vol. 3, no. 4, 2020.

[74] S. Lek and Y. S. Park, "Multilayer Perceptron," in *Encyclopedia of Ecology, Five-Volume Set*, 2008, pp. 2455–2462.

[75] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," 2018.

[76] L. Zhang, G. Zhu, P. Shen, J. Song, S. A. Shah, and M. Bennamoun, "Learning spatiotemporal features using 3DCNN and convolutional LSTM for gesture recognition," in *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, vol. 2018-Janua, 2017, pp. 3120–3128.

[77] H. M. Fayek, M. Lech, and L. Cavedon, "Evaluating deep learning architectures for Speech Emotion Recognition," *Neural Networks*, vol. 92, pp. 60–68, 2017.

[78] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[79] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[80] ., "7 Types of Activation Functions in Neural Networks: How to Choose?" p. 19, 2019. [Online]. Available: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/

[81] S. Jadon, "Introduction to different activation functions for deep learning," *Medium, Augmenting Humanity*, vol. 16, 2018.

[82] V. Nair and G. E. Hinton, "Rectified linear units improve Restricted Boltzmann machines," in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[83] C. Valentini-Botinhao, I. Cohen, Y. Xing, P. Kirkland, G. Di Caterina, J. Soraghan, G. Matich, D. W. Hansen, Q. Ji, S. Kumar, M. K. Bhuyan, B. K. Chakraborty, M. Milders, J. K. Hietanen, J. M. Leppänen, M. Braun, G. Sandbach, S. Zafeiriou, M. Pantic, L. Yin, T. Kinnunen, H. H. Li, M. El Ayadi, M. S. Kamel, F. Karray, S. Kim, S. J. Guy, K. Hillesland, B. Zafar, A. A. A. Gutub, D. Manocha, Mustaqeem, S. Kwon, L. L. Grewe, C. Hu, Y. Y. H. Y. Y. Liu, O. Sourina, M. K. Nguyen, W. Ke, Y. Xing, G. Di Caterina, L. Petropoulakis, J. Soraghan, M. El Ayadi, M. S. Kamel, F. Karray, Q. Mao, M. M. Dong, Z. Huang, Y. Zhan, S. Loiselle, J. Rouat, D. Pressnitzer, S. J. Thorpe, E. Ohn-Bar, M. M. Trivedi, N. Neverova, C. Wolf, G. W. Taylor, F. Nebout, Y. Liuy, L. Chengy, L. Cheng, Y. Y. H. Y. Y. Liu, Z. G. Hou, M. Tan, D. Du, M. Fei, J. Botzheim, T. Obo, N. Kubota, P. Molchanov, S. Gupta, K. Kim, K. Pulli, J. Wu, Y. Chua, H. H. Li, J. P. Dominguez-Morales, Q. Liu, R. James, D. Gutierrez-Galan, A. Jimenez-Fernandez, S. Davidson, S. B. Furber, ., P. Smolensky, S. Lek, Y. S. Park, D. Yu, L. L. Deng, I. Jang, P. Kudumakis, M. Sandler, K. Kang, A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Y. H. Y. Y. Liu, F. E. Alsaadi, L. L. L. Zhang, G. G. Zhu, P. Shen, J. J. Song, S. A. Shah, M. Bennamoun, H. M. Fayek, M. Lech, L. Cavedon, M. Nikzad,

K. Movagharnejad, F. Talebnia, F. Rosenblatt, G. E. Hinton, S. Osindero, Y. W. Teh, H. J. KELLEY, R. Ranjan, S. Sankaranarayanan, A. Bansal, N. Bodla, J. C. J. Chen, V. M. Patel, C. D. Castillo, R. Chellappa, Go, Y. Lecun, Y. Bengio, G. E. Hinton, H. B. Curry, J. Kiefer, J. Wolfowitz, D. A. Clevert, T. Unterthiner, S. Hochreiter, V. Nair, G. E. Hinton, A. L. Maas, A. Y. Hannun, A. Y. Ng, J. W. Choi, S. J. Ryu, J. H. Kim, Y. Y. Liao, P. Xiong, W. W. Min, W. W. Min, J. Lu, T. S. Kim, A. Reiter, A. R. Asif, A. Waris, S. O. Gilani, M. Jamil, H. Ashraf, M. Shafique, I. K. Niazi, Z. Z. Zhang, K. Yang, J. Qian, L. L. L. Zhang, A. Deshpande, F. Deloche, P. Konig, A. K. Engel, P. R. Roelfsema, W. Singer, T. J. Hamilton, A. A. Van Schaik, E. M. Izhikevich, D. Huh, T. J. Sejnowski, M. Schuster, K. K. Paliwal, A. Kasiński, F. Ponulak, A. Mohemmed, S. Schliebs, S. Matsuda, N. K. Kasabov, R. V. Florian, F. Zenke, S. Ganguli, A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, A. S. Maida, J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, S. Millner, N. Ketkar, N. Ketkar, H. Markram, K. Meier, T. Lippert, S. Grillner, R. Frackowiak, S. Dehaene, A. Knoll, H. Sompolinsky, K. Verstreken, J. DeFelipe, S. Grant, J. P. Changeux, A. Sariam, A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, K. Boahen, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, K. Boahen, M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, H. H. H. Wang, M. Abadi, P. Barham, J. C. J. Chen, Z. Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison,

S. El Boustani, A. Destexhe, M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, T. Masquelier, S. Moradi, N. Qiao, F. Stefanini, G. Indiveri, J. Pei, L. L. Deng, S. Song, M. Zhao, Y. Y. Zhang, S. S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, H. H. Li, H. Wu, R. Zhao, Y. Y. Xie, L. Shi, D. Ma, J. J. Shen, Z. Gu, M. M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, G. Pan, B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, S. C. Liu, T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. R. Voelker, C. Eliasmith, O. Moreira, A. Yousefzadeh, F. Chersi, G. Cinserin, R. J. Zwartenkot, A. Kapoor, P. Qiao, P. Kievits, M. Khoei, L. Rouillard, A. Ferouge, J. Tapson, A. Visweswara, M. Stimberg, D. F. Goodman, T. Nowotny, H. Hazan, D. J. Saunders, H. Khan, D. Patel, D. T. Sanghavi, H. T. Siegelmann, R. Kozma, S. C. Liu, A. A. Van Schaik, B. A. Minch, T. Delbruck, R. Kreiser, A. Renner, V. R. Leite, B. Serhan, C. Bartolozzi, A. Glover, Y. Sandamirskaya, J. Wu, Y. Chua, M. M. Zhang, H. H. Li, K. K. C. Tan, E. Mansouri-Benssassi, J. Ye, R. Lotfidereshgi, P. Gournay, P. U. Diehl, B. U. Pedroni, A. S. Cassidy, P. A. Merolla, E. Neftci, G. Zarrella, T. Tashan, T. Allen, L. Nolle, O. Martin, I. Kotsia, B. Macq, I. Pitas, S. R. Livingstone, F. A. Russo, T. Giannakopoulos, N. Hajarolasvadi, H. Demirel, F. Noroozi, M. Marjanovic, A. Njegus, S. Escalera, G. Anbarjafari, T. Özseven, J. Deng, Z. Z. Zhang, E. Marchi, B. B. Schuller, M. A. Jalal, E. Loweimi, R. K. Moore, T. Hain, Z. J. Chuang, C.-h. Wu, J. Zhao, X. Mao, L. Chen, A. Bhavan, P. Chauhan, Hitkul, R. R. Shah, P. Shegokar, P. Sircar, Z. J. Chuang, C.-h. Wu, Y. Zeng, H. Mao, D. Peng, Z. Yi, F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, B. Weiss, M. Kim, J. Cho, S. Lee, Y. Jung, N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, A. Waibel, A. Y. Hannun, A. Lee, Q. Xu, R. Collobert, E. G. Walsh, A. Narayanan, D. L. Wang, L. F. Abbott, E. D. Adrian, Y. Zotterman, J. Tchroz, B. Kollmeier, Y. Wang, D. L. Wang, C. Glackin, L. Maguire, L. McDaid, J. Wade, C. Valentini-Botinhao, D. F. Goodman, R. Brette, D. J. Felleman, D. C. Van Essen, A. Sengupta, Y. Ye, R. Wang, C. Liu, K. Roy, T. Serre, W. A.

Freiwald, D. Y. Tsao, A. L. Hodgkin, A. F. Huxley, H. Paugam-Moisy, S. M. Bohte, W. Gerstner, A. L. Hodgkin, A. F. Huxley, B. Katz, T. J. Gawne, T. W. Kjaer, B. J. Richmond, E. D. Adrian, Y. Zotterman, S. S. Wu, S. I. Amari, H. Nakahara, L. Kostal, P. Lansky, J. P. Rospars, S. S. Wu, S. I. Amari, H. Nakahara, A. L. Hodgkin, A. F. Huxley, O. Booij, H. Tat Nguyen, S. M. Bohte, J. N. Kok, H. La Poutré, M. M. Dong, X. Huang, B. Xu, Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, S. C. Liu, R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbrück, S. C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Civit Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, B. Linares-Barranco, S. Song, K. D. Miller, L. F. Abbott, R. Guyonneau, R. Vanrullen, S. J. Thorpe, T. Masquelier, R. Guyonneau, S. J. Thorpe, A. Coates, H. Lee, A. Y. Ng, T. Masquelier, S. R. Kheradpisheh, B. Nessler, M. Pfeiffer, W. Maass, S. K. Esser, R. Appuswamy, P. A. Merolla, J. V. Arthur, D. S. Modha, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. Di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, D. S. Modha, G. Q. Bi, M. M. Poo, H. Markram, J. Lübke, M. Frotscher, B. Sakmann, G. G. Turrigiano, S. B. Nelson, J. R. Knott, E. Vasilaki, M. Giugliano, E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, S. C. Liu, A. Artola, S. Bröcher, W. Singer, G. Indiveri, F. Corradi, N. Qiao, H. Markram, W. Gerstner, P. J. Sjöström, R. Pyle, R. Rosenbaum, C. Clopath, L. Büsing, E. Vasilaki, W. Gerstner, E. Rueckert, D. Kappel, D. Tanneberg, D. Pecevski, J. Peters, G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, W. Maass, N. K. Kasabov, Z. Q. Zhao, P. Zheng, S. T. Xu, X. Wu, K. He, G. Gkioxari, P. Dollár, R. Girshick, S. Milyaev, I. Laptev, P. Molchanov, S. Gupta, K. Kim, J. Kautz, H. J. Kim, J. S. Lee, J. H. Park, D. Boehning, S. H. Snyder, D. P. Kingma, J. L. Ba, D. E. Rumelhart, G. E. Hinton, R. J. Williams, J. Duchi, E. Hazan, Y. Singer, K. Fukushima, S. Miyake, T. Ito, S. Hochreiter, J. Schmidhuber,

F. A. Gers, J. Schmidhuber, F. Cummins, Y. Bengio, P. Simard, P. Frasconi, R. J. Williams, D. Zipser, W. S. Noble, S. Singh, J. Haddon, M. Markou, D. Hamester, P. Barros, S. Wermter, T. S. Kim, A. Reiter, A. Kar, N. Rai, K. Sikka, G. Sharma, K. Wang, J. He, L. L. L. Zhang, C. Avilés-Cruz, A. Ferreyra-Ramírez, A. Zúñiga-López, J. Villegas-Cortéz, D. Povey, X. Zhang, S. Khudanpur, P. J. Werbos, E. P. Ijjina, K. M. Chalavadi, G. A. Rao, K. Syamala, P. V. Kishore, A. S. Sastry, A. Karpathy, K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, R. Jozefowicz, W. Zaremba, I. Sutskever, M. K. Nammous, K. Saeed, L. Dovydaitis, V. Rudžionis, Y. Y. Xie, R. H. R. Liang, Z. Liang, C. Huang, C. Zou, B. B. Schuller, A. Graves, S. Fernández, J. Schmidhuber, VoxForge, Y. Ephraim, D. Malah, M. Avissar, A. C. Furman, J. C. Saunders, T. D. Parsons, C. A. Peckens, J. P. Lynch, F. Xie, D. van Compemolle, Q. Zhang, M. Wang, S. F. Boll, J. Wall, C. Glackin, N. Cannings, G. Chollet, N. Dugan, A. Tavanaei, A. S. Maida, S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, T. Masquelier, S. Lobov, V. Mironov, I. Kastalskiy, V. Kazantsev, L. Robles, M. A. Ruggero, A. J. Hudspeth, R. F. Lyon, P. Dallos, P. Lichtsteiner, C. Posch, T. Delbruck, D. Matolin, R. Wohlgenannt, G. G. Zhu, C. Xu, Q. Huang, W. Gao, L. Xing, S. Ji, W. Xu, M. M. Yang, K. Yu, K. Simonyan, A. Zisserman, R. Poppe, A. Krizhevsky, I. Sutskever, G. E. Hinton, H. H. H. Wang, C. Schmid, S. Danafar, N. Gheissari, J. A. Pérez-Carrasco, C. Serrano, B. Acha, T. Serrano-Gotarredona, B. Linares-Barranco, O. Bichler, D. Querlioz, S. J. Thorpe, J. P. Bourgoin, C. Gamrat, X. J. Wang, L. Gao, J. J. Song, H. Shen, A. Amir, B. Taba, D. J. Berg, T. Melano, J. L. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. K. Esser, T. Delbruck, M. D. Flickner, D. S. Modha, X. Shi, Z. Z. Chen, H. H. H. Wang, D. Y. Yeung, W. K. Wong, W. C. Woo, J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, T. Darrell, K. Zhou, Y. Zhu, Y. Zhao, H. Song, W. Wang, S. Zhao, J. J. Shen, K. M. Lam, S. M. Bohte, J. N. Kok, H. La Poutré, S. McKennoch,

L. Dingding, L. G. Bushnell, W. Gerstner, S. B. Shrestha, G. Orchard, G. E. Hinton, S. Nitish, K. Swersky, D. P. Kingma, J. L. Ba, B. Schrauwen, J. Van Campenhout, N. Srivastava, E. Mansimov, R. Salakhutdinov, E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, F. F. Li, K. K. C. Tan, D. L. Wang, K. Choi, G. Fazekas, M. Sandler, K. Cho, L. L. Wang, K. Li, X. Chen, X. P. Hu, H. Yang, J. Zhang, S. Li, J. Lei, S. Chen, M. Majd, R. Safabakhsh, S. H. Bae, I. Choi, N. S. Kim, X. J. Wang, L. Gao, J. J. Song, H. Shen, Y. Fan, X. Lu, D. Li, Y. Y. H. Y. Y. Liu, A. L. Hodgkin, A. F. Huxley, J. Vreeken, V. Nair, G. E. Hinton, W. Gerstner, W. M. Kistler, R. Naud, L. Paninski, Y. Y. H. Y. Y. Liu, X. J. Wang, N. K. Kasabov, K. Dhoble, N. Nuntalid, G. Indiveri, A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, P. Yger, A. Mohemmed, S. Schliebs, S. Matsuda, N. K. Kasabov, F. Akopyan, J. Sawada, A. S. Cassidy, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, D. S. Modha, A. Tavanaei, A. S. Maida, S. B. Furber, F. Galluppi, S. Temple, L. A. Plana, B. Han, T. M. Taha, S. G. Wysoski, L. Benuskova, N. K. Kasabov, S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, T. Masquelier, P. U. Diehl, M. Cook, J. Feng, D. Brown, S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, A. D. Brown, W. Gerstner, S. Loiselle, J. Rouat, D. Pressnitzer, S. J. Thorpe, K. Dhoble, N. Nuntalid, G. Indiveri, N. K. Kasabov, J. M. Bower, D. Beeman, M. Nelson, J. Rinzel, P. J. Werbos, S. Miao, G. Chen, X. Ning, Y. Zi, K. Ren, Z. Bing, A. Knoll, Q. Q. Wang, Y. Y. Zhang, J. Yuan, Y. Lu, G. I. Parisi, J. Tani, C. Weber, S. Wermter, Q. Q. Wang, K. Chen, M. Mahowald, C. Posch, D. Matolin, R. Wohlgenannt, P. Lichtsteiner, C. Posch, T. Delbruck, C. Brandli, R. Berner, M. M. Yang, S. C. Liu, T. Delbruck, A. L. Hodgkin, A. F. Huxley, L. F. Abbott, C. R. Qi, L. Yi, H. Su, L. J. Guibas, W. Gerstner, W. Teka, T. M. Marinov, F. Santamaria, S. S. Rautaray, A. Agrawal, A. Haria, A. Subramanian, N. Asokkumar,

S. Poddar, J. S. Nayak, S. Mitra, T. Acharya, L. Pigou, S. Dieleman, P. J. Kindermans, B. Schrauwen, D. Droeschel, J. Stückler, S. Behnke, H. Liu, L. L. Wang, R. H. R. Liang, M. Ouhyoung, V. Frati, D. Prattichizzo, R. Yang, S. Sarkar, B. Loeding, D. Wickeroth, P. Benölken, U. Lang, S. Hochreiter, and J. Schmidhuber, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Neural Computation*, vol. 13, no. 1, pp. 25–71, 2019. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.7093{&}rep=rep1{&}type=pdfhttps://www.frontiersin.org/article/10.3389/fnins.2019.00434http://www.voxforge.org/http://karpathy.github.io/2015/05/21/rnn-effectiveness/https://datashare.is.e

[84] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[85] K. Chakraborty, S. Bhattacharyya, R. Bag, and A. Hassanien, "Sentiment analysis on a set of movie reviews using deep learning techniques," in *Social Network AnalyticsComputational Research Methods and Techniques.* Elsevier, 2018.

[86] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, 1944.

[87] Go, "Deep learning Deep Learning ," *Nature*, vol. 29, no. 7553, pp. 1–73, 2019.

[88] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[89] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *COLT 2010 - The 23rd Conference on Learning Theory*, 2010, pp. 257–269.

[90] D. Boehning and S. H. Snyder, "Novel neural modulators," in *Annual Review of Neuroscience*, vol. 26, 2003, pp. 105–131.

[91] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[92] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," pp. 436–444, 2015.

[93] R. Ranjan, S. Sankaranarayanan, A. Bansal, N. Bodla, J. C. Chen, V. M. Patel, C. D. Castillo, and R. Chellappa, "Deep Learning for Understanding Faces: Machines May Be Just as Good, or Better, than Humans," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 66–83, 2018.

[94] S. Milyaev and I. Laptev, "Towards reliable object detection in noisy images," *Pattern Recognition and Image Analysis*, vol. 27, no. 4, pp. 713–722, 2017.

[95] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," pp. 3212–3232, 2019.

[96] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020.

[97] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[98] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do It," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[99] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[100] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[101] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[102] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, pp. 1724–1734.

[103] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[104] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of Recurrent Network architectures,” in *32nd International Conference on Machine Learning, ICML 2015*, vol. 3, 2015, pp. 2332–2340.

[105] A. Karpathy, “The Unreasonable Effectiveness of Recurrent Neural Networks,” pp. 1–28, 2015. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[106] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in *Advances in Neural Information Processing Systems*, vol. 2015-Janua, 2015, pp. 802–810.

[107] N. Srivastava, E. Mansimov, and R. Salakhutdinov, “Unsupervised learning of video representations using LSTMs,” in *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, 2015, pp. 843–852.

[108] S. H. Bae, I. Choi, and N. S. Kim, “Acoustic Scene Classification Using Parallel Combination of LSTM and CNN,” *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, no. September, pp. 11–15, 2016.

[109] Y. Fan, X. Lu, D. Li, and Y. Liu, "Video-Based emotion recognition using CNN-RNN and C3D hybrid networks," in *ICMI 2016 - Proceedings of the 18th ACM International Conference on Multimodal Interaction*, 2016, pp. 445–450.

[110] X. Wang, L. Gao, J. Song, and H. Shen, "Beyond Frame-level CNN: Saliency-Aware 3-D CNN with LSTM for Video Action Recognition," *IEEE Signal Processing Letters*, vol. 24, no. 4, pp. 510–514, 2017.

[111] L. Wang, K. Li, X. Chen, and X. P. Hu, "Application of convolutional recurrent neural network for individual recognition based on resting state fMRI Data," *Frontiers in Neuroscience*, vol. 13, no. MAY, p. 434, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2019.00434

[112] K. Tan and D. L. Wang, "A convolutional recurrent neural network for real-time speech enhancement," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2018-Septe, 2018, pp. 3229–3233.

[113] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017, pp. 2392–2396.

[114] H. Yang, J. Zhang, S. Li, J. Lei, and S. Chen, "Attend it again: Recurrent attention convolutional neural network for action recognition," *Applied Sciences (Switzerland)*, vol. 8, no. 3, p. 383, 2018.

[115] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[116] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[117] M. Majd and R. Safabakhsh, “A motion-aware ConvLSTM network for action recognition,” *Applied Intelligence*, vol. 49, no. 7, pp. 2515–2521, 2019.

[118] D. J. Felleman and D. C. Van Essen, “Distributed hierarchical processing in the primate cerebral cortex,” *Cerebral Cortex*, vol. 1, no. 1, pp. 1–47, 1991.

[119] W. A. Freiwald and D. Y. Tsao, “Functional compartmentalization and viewpoint generalization within the macaque face-processing system,” *Science*, vol. 330, no. 6005, pp. 845–851, 2010.

[120] T. Serre, “Hierarchical Models of the Visual System,” in *Encyclopedia of Computational Neuroscience*, 2014, pp. 1–12.

[121] N. Kasabov, “Time-space, spiking neural networks and brain-inspired artificial intelligence, springer,” 2018.

[122] J. V. Stone, “Principles of neural information theory,” *Computational Neuroscience and Metabolic Efficiency*, 2018.

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

[124] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *Bulletin of Mathematical Biology*, vol. 52, no. 1-2, pp. 25–71, 1990.

[125] A. L. Hodgkin, A. F. Huxley, and B. Katz, “Measurement of currentvoltage relations in the membrane of the giant axon of Loligo,” *The Journal of Physiology*, vol. 116, no. 4, pp. 424–448, 1952.

[126] A. L. Hodgkin and A. F. Huxley, “The components of membrane conductance in the giant axon of Loligo,” *The Journal of Physiology*, vol. 116, no. 4, pp. 473–496, 1952.

[127] P. Konig, A. K. Engel, P. R. Roelfsema, and W. Singer, “How precise is neuronal synchronization?” *Neural Computation*, vol. 7, no. 3, pp. 469–485, 1995.

[128] J. Feng, “Is the integrate-and-fire model good enough? - A review,” pp. 955–975, 2001.

[129] J. Feng and D. Brown, “Integrate-and-fire models with nonlinear leakage,” *Bulletin of Mathematical Biology*, vol. 62, no. 3, pp. 467–481, 2000.

[130] Y. H. Liu and X. J. Wang, “Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron,” *Journal of Computational Neuroscience*, vol. 10, no. 1, pp. 25–45, 2001.

[131] A. van Schaik, C. Jin, A. McEwan, and T. J. Hamilton, “A log-domain implementation of the izhikevich neuron model,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems.* IEEE, 2010, pp. 4253–4256.

[132] W. Gerstner, “Spike-response model,” *Scholarpedia*, vol. 3, no. 12, p. 1343, 2008.

[133] T. J. Hamilton and A. Van Schaik, “Silicon implementation of the generalized integrate-and-fire neuron model,” in *Proceedings of the 2011 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2011*, 2011, pp. 108–112.

[134] W. Gerstner, “Spiking Neuron Models,” in *Encyclopedia of Neuroscience*, 2009, pp. 277–280.

[135] E. M. Izhikevich, “Simple model of spiking neurons,” pp. 1569–1572, 2003.

[136] H. Paugam-Moisy and S. Bohte, “Computing with spiking neuron networks,” in *Handbook of Natural Computing*, 2012, vol. 1-4, pp. 335–376.

[137] E. D. Adrian and Y. Zotterman, “The impulses produced by sensory nerveendings: Part II. The response of a Single EndOrgan,” *The Journal of Physiology*, vol. 61, no. 2, pp. 151–171, 1926.

[138] M. Meister, J. Pine, and D. A. Baylor, “Multi-neuronal signals from the retina: acquisition and analysis,” *Journal of neuroscience methods*, vol. 51, no. 1, pp. 95–106, 1994.

[139] S. J. Thorpe, “Spike arrival times: A highly efficient coding scheme for neural networks,” *Parallel processing in neural systems*, pp. 91–94, 1990.

[140] T. J. Gawne, T. W. Kjaer, and B. J. Richmond, “Latency: Another potential code for feature binding in striate cortex,” *Journal of Neurophysiology*, vol. 76, no. 2, pp. 1356–1360, 1996.

[141] S. Wu, S. I. Amari, and H. Nakahara, “Population Coding and Decoding in a Neural Field: A Computational Study,” *Neural Computation*, vol. 14, no. 5, pp. 999–1026, 2002.

[142] D. A. Butts, C. Weng, J. Jin, C.-I. Yeh, N. A. Lesica, J.-M. Alonso, and G. B. Stanley, “Temporal precision in the neural code and the timescales of natural vision,” *Nature*, vol. 449, no. 7158, pp. 92–95, 2007.

[143] L. Kostal, P. Lansky, and J. P. Rospars, “Neuronal coding and spiking randomness,” pp. 2693–2701, 2007.

[144] K. Johnson, “Sensory discrimination: neural processes preceding discrimination decision,” *Journal of Neurophysiology*, vol. 43, no. 6, pp. 1793–1815, 1980.

[145] B. A. Olshausen and D. J. Field, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[146] 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.

[147] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, “Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition,” *Neurocomputing*, vol. 205, pp. 382–392, 2016.

[148] A. Tavanaei, T. Masquelier, and A. S. Maida, “Acquisition of visual features through probabilistic spike-timing-dependent plasticity,” in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 307–314.

[149] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, “Feedforward categorization on aer motion events using cortex-like features in a spiking neural network,” *IEEE transactions on neural networks and learning systems*, vol. 26, no. 9, pp. 1963–1978, 2014.

[150] S. K. Esser, R. Appuswamy, P. A. Merolla, J. V. Arthur, and D. S. Modha, “Backpropagation for energy-efficient neuromorphic computing,” in *Advances in Neural Information Processing Systems*, vol. 2015-Janua, 2015, pp. 1117–1125.

[151] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. Di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, “Convolutional networks for fast, energy-efficient neuromorphic computing,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[152] G. Indiveri, F. Corradi, and N. Qiao, “Neuromorphic architectures for spiking deep neural networks,” in *Technical Digest - International Electron Devices Meeting, IEDM*, vol. 2016-Febru, 2015, pp. 4.2.1–4.2.4.

[153] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S. C. Liu, “Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms,” *Frontiers in Neuroscience*, vol. 9, no. JUN, 2015.

[154] P. O’Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, “Real-time classification and sensor fusion with a spiking deep belief network,” *Frontiers in neuroscience*, vol. 7, p. 178, 2013.

[155] E. Rueckert, D. Kappel, D. Tanneberg, D. Pecevski, and J. Peters, “Recurrent Spiking Networks Solve Planning Tasks,” *Scientific Reports*, vol. 6, 2016.

[156] R. Pyle and R. Rosenbaum, “Spatiotemporal Dynamics and Reliable Computations in Recurrent Spiking Neural Networks,” *Physical Review Letters*, vol. 118, no. 1, 2017.

[157] W. Zhang and P. Li, “Spike-train level backpropagation for training deep recurrent spiking neural networks,” in *Advances in Neural Information Processing Systems*, 2019, pp. 7802–7813.

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

[159] L. Paulun, A. Wendt, and N. Kasabov, “A retinotopic spiking neural network system for accurate recognition of moving objects using neucube and dynamic vision sensors,” *Frontiers in Computational Neuroscience*, vol. 12, p. 42, 2018.

[160] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, “Long short-term memory and learning-to-learn in networks of spiking neurons,” in *Advances in Neural Information Processing Systems*, vol. 2018-Decem, 2018, pp. 787–797.

[161] T. Masquelier, R. Guyonneau, and S. J. Thorpe, “Competitive STDP-based spike pattern learning,” pp. 1259–1276, 2009.

[162] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, “Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity,” *PLoS Comput Biol*, vol. 9, no. 4, p. e1003037, 2013.

[163] A. Coates, H. Lee, and A. Y. Ng, “An analysis of single-layer networks in unsupervised feature learning,” in *Journal of Machine Learning Research*, vol. 15, 2011, pp. 215–223.

[164] A. Dundar, J. Jin, and E. Culurciello, “Convolutional clustering for unsupervised learning,” *arXiv preprint arXiv:1511.06241*, 2015.

[165] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, “STDP-based spiking deep convolutional neural networks for object recognition,” *Neural Networks*, vol. 99, pp. 56–67, 2018.

[166] C. Lee, G. Srinivasan, P. Panda, and K. Roy, “Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity,” *IEEE*

Transactions on Cognitive and Developmental Systems, vol. 11, no. 3, pp. 384–394, 2018.

[167] J.-W. Lin and D. S. Faber, "Modulation of synaptic delay during synaptic plasticity," *Trends in neurosciences*, vol. 25, no. 9, pp. 449–455, 2002.

[168] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs," *Science*, vol. 275, no. 5297, pp. 213–215, 1997.

[169] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.

[170] J. R. Knott, "The organization of behavior: A neuropsychological theory," *Electroencephalography and Clinical Neurophysiology*, vol. 3, no. 1, pp. 119–120, 1951.

[171] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, p. 774, 2018.

[172] G. G. Turrigiano and S. B. Nelson, "Homeostatic plasticity in the developing nervous system," pp. 97–107, 2004.

[173] A. Artola, S. Bröcher, and W. Singer, "Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex," *Nature*, vol. 347, no. 6288, pp. 69–72, 1990.

[174] E. Vasilaki and M. Giugliano, "Emergence of connectivity motifs in networks of model neurons with short- and long-term plastic synapses," *PLoS ONE*, vol. 9, no. 1, 2014.

[175] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner, "Connectivity reflects coding: A model of voltage-based spike-timing-dependent-plasticity with homeostasis," *Nature Precedings*, 2009.

[176] N. Caporale and Y. Dan, “Spike timing–dependent plasticity: a hebbian learning rule,” *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[177] H. Markram, W. Gerstner, and P. J. Sjöström, “A history of spike-timing-dependent plasticity,” pp. 1–24, 2011.

[178] R. Guyonneau, R. Vanrullen, and S. J. Thorpe, “Neurons tune to the earliest spikes through STDP,” *Neural Computation*, vol. 17, no. 4, pp. 859–879, 2005.

[179] S. Song, K. D. Miller, and L. F. Abbott, “Competitive Hebbian learning through spike-timing-dependent synaptic plasticity,” *Nature Neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[180] T. Masquelier, R. Guyonneau, and S. J. Thorpe, “Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains,” *PloS one*, vol. 3, no. 1, p. e1377, 2008.

[181] T. Masquelier and S. R. Kheradpisheh, “Optimal Localist and Distributed Coding of Spatiotemporal Spike Patterns Through STDP and Coincidence Detection,” *Frontiers in Computational Neuroscience*, vol. 12, 2018.

[182] B. Nessler, M. Pfeiffer, and W. Maass, “STDP enables spiking neurons to detect hidden causes of their inputs,” in *Advances in Neural Information Processing Systems 22 - Proceedings of the 2009 Conference*, 2009, pp. 1357–1365.

[183] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks,” pp. 47–63, 2019.

[184] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbrück, S. C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Civit Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, and B. Linares-Barranco, “CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and

tracking," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1417–1438, 2009.

[185] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2706–2719, 2013.

[186] Y. Hu, H. Tang, Y. Wang, and G. Pan, "Spiking deep residual network," *arXiv preprint arXiv:1805.01352*, 2018.

[187] 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, no. DEC, 2017.

[188] 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.

[189] S. M. Bohte, J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," pp. 17–37, 2002.

[190] O. Booij and H. Tat Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," *Information Processing Letters*, vol. 95, no. 6 SPEC. ISS., pp. 552–558, 2005.

[191] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection," *Neural networks*, vol. 22, no. 10, pp. 1419–1431, 2009.

[192] A. Kasiński and F. Ponulak, "Comparison of supervised learning methods for spike time coding in spiking neural networks," pp. 101–113, 2006.

[193] F. Ponulak, "Supervised learning in spiking neural networks with resume method," *Phd, Poznan University of Technology*, vol. 46, p. 47, 2006.

[194] R. V. Florian, “The chronotron: A neuron that learns to fire temporally precise spike patterns,” *PLoS ONE*, vol. 7, no. 8, 2012.

[195] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, “Span: Spike pattern association neuron for learning spatio-temporal spike patterns,” *International Journal of Neural Systems*, vol. 22, no. 4, 2012.

[196] D. Huh and T. J. Sejnowski, “Gradient descent for spiking neural networks,” in *Advances in Neural Information Processing Systems*, vol. 2018-Decem, 2018, pp. 1433–1443.

[197] J. H. Lee, T. Delbruck, and M. Pfeiffer, “Training deep spiking neural networks using backpropagation,” *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[198] F. Zenke and S. Ganguli, “SuperSpike: Supervised learning in multilayer spiking neural networks,” *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[199] S. B. Shrestha and G. Orchard, “Slayer: Spike layer error reassignment in time,” in *Advances in Neural Information Processing Systems*, vol. 2018-Decem, 2018, pp. 1412–1421.

[200] G. Hinton, S. Nitish, and K. Swersky, “Lecture 6a: Overview of minibatch gradient descent,” *Neural Networks for Machine Learning*, 2017.

[201] 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,” in *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, 2011, pp. 259–275.

[202] ——, “A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS,” in *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, 2011, pp. 259–275.

[203] S. C. Liu, A. Van Schaik, B. A. Minch, and T. Delbruck, “Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms,” in *ISCAS 2010 -*

2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, 2010, pp. 2027–2030.

[204] N. Ketkar and N. Ketkar, "Introduction to PyTorch," in *Deep Learning with Python*, 2017, pp. 195–208.

[205] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE custom integrated circuits conference (CICC).* IEEE, 2011, pp. 1–4.

[206] S. Furber and A. Brown, "Biologically-inspired massively-parallel architectures-computing beyond a million processors," in *2009 Ninth International Conference on Application of Concurrency to System Design.* IEEE, 2009, pp. 3–12.

[207] H. Markram, K. Meier, T. Lippert, S. Grillner, R. Frackowiak, S. Dehaene, A. Knoll, H. Sompolinsky, K. Verstreken, J. DeFelipe, S. Grant, J. P. Changeux, and A. Sariam, "Introducing the Human Brain Project," in *Procedia Computer Science*, vol. 7, 2011, pp. 39–42.

[208] A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "PyNN: A common interface for neuronal network simulators," *Frontiers in Neuroinformatics*, vol. 2, no. JAN, 2009.

[209] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, 2010, pp. 1947–1950.

[210] S. Scholze, S. Schiefer, J. Partzsch, S. Hartmann, C. G. Mayr, S. Höppner, H. Eisenreich, S. Henker, B. Vogginger, and R. Schüffny, "Vlsi implementation of a 2.8 gevent/s packet-based aer interface with routing and event sorting functionality," *Frontiers in neuroscience*, vol. 5, p. 117, 2011.

[211] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence).* IEEE, 2008, pp. 431–438.

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

[213] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, "Braindrop: A Mixed-Signal Neuromorphic Architecture with a Dynamical Systems-Based Programming Model," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 144–164, 2019.

[214] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[215] "Loihi - Intel - WikiChip," 2018. [Online]. Available: https://en.wikichip.org/wiki/intel/loihi{%}0Ahttp://files/5717/loihi.html

[216] "Akida Neural Processor System-on-Chip - BrainChip." [Online]. Available: https://brainchipinc.com/akida-neuromorphic-system-on-chip/

[217] D. Ma, J. Shen, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, and G. Pan, "Darwin: A neuromorphic hardware co-processor based on spiking neural networks," *Journal of Systems Architecture*, vol. 77, pp. 43–51, 2017.

[218] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A Scalable Multicore Architecture with Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106–122, 2018.

[219] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, H. Li, H. Wu, R. Zhao, Y. Xie, and L. Shi, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[220] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani, and A. Destexhe, "Simulation of networks of spiking neurons: A review of tools and strategies," *Journal of Computational Neuroscience*, vol. 23, no. 3, pp. 349–398, 2007.

[221] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016, pp. 265–283.

[222] D. Goodman and R. Brette, "Brian: A simulator for spiking neural networks in python," *Frontiers in Neuroinformatics*, vol. 2, no. NOV, 2008.

[223] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, and T. Masquelier, "Spyke-Torch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron," *Frontiers in Neuroscience*, vol. 13, no. JUL, 2019.

[224] O. Moreira, A. Yousefzadeh, F. Chersi, G. Cinserin, R. J. Zwartenkot, A. Kapoor, P. Qiao, P. Kievits, M. Khoei, L. Rouillard, A. Ferouge, J. Tapson, and A. Visweswara, "NeuronFlow: A neuromorphic processor architecture for Live AI applications," in *Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition, DATE 2020*, 2020, pp. 840–845.

[225] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. R. Voelker, and C. Eliasmith, "Nengo: A Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, no. JAN, 2014.

[226] H. Hazan, D. J. Saunders, H. Khan, D. Patel, D. T. Sanghavi, H. T. Siegelmann, and R. Kozma, "BindsNET: A machine learning-oriented spiking neural networks library in python," *Frontiers in Neuroinformatics*, vol. 12, 2018.

[227] M. Stimberg, D. F. Goodman, and T. Nowotny, "Brian2GeNN: accelerating spiking neural network simulations with graphics hardware," *Scientific Reports*, vol. 10, no. 1, 2020.

[228] J. S. Yannan Xing, Gaetano Di Caterina, "A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition," *Frontiers in Neuroscience(submitted)*, 2020.

[229] T. Zhou and J. P. Wachs, "Spiking neural networks for early prediction in human–robot collaboration," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1619–1643, 2019.

[230] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification," *Frontiers in Neuroscience*, vol. 12, no. NOV, 2018.

[231] R. Kreiser, A. Renner, V. R. Leite, B. Serhan, C. Bartolozzi, A. Glover, and Y. Sandamirskaya, "An On-chip Spiking Neural Network for Estimation of the Head Pose of the iCub Robot," *Frontiers in Neuroscience*, vol. 14, 2020.

[232] P. Panda and N. Srinivasa, "Learning to recognize actions from limited training examples using a recurrent spiking neural model," *Frontiers in neuroscience*, vol. 12, p. 126, 2018.

[233] J. Yang, Q. Wu, M. Huang, and T. Luo, “Real time human motion recognition via spiking neural network,” in *IOP Conference Series: Materials Science and Engineering*, vol. 366, 2018, p. 12042.

[234] A. K. Mukhopadhyay, I. Chakrabarti, and M. Sharad, “Classification of hand movements by surface myoelectric signal using artificial-spiking neural network model,” in *2018 IEEE SENSORS*. IEEE, 2018, pp. 1–4.

[235] J. P. Dominguez-Morales, Q. Liu, R. James, D. Gutierrez-Galan, A. Jimenez-Fernandez, S. Davidson, and S. Furber, “Deep spiking neural network model for time-variant signals classification: a real-time speech recognition approach,” in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[236] J. Wu, Y. Chua, and H. Li, “A biologically plausible speech recognition framework based on spiking neural networks,” in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[237] R. Lotfidereshgi and P. Gournay, “Biologically inspired speech emotion recognition,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017, pp. 5135–5139.

[238] E. Mansouri-Benssassi and J. Ye, “Speech Emotion Recognition with Early Visual Cross-modal Enhancement Using Spiking Neural Networks,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2019-July, 2019.

[239] Y. Luo, Q. Fu, J. Xie, Y. Qin, G. Wu, J. Liu, F. Jiang, Y. Cao, and X. Ding, “Eeg-based emotion classification using spiking neural networks,” *IEEE Access*, vol. 8, pp. 46 007–46 016, 2020.

[240] P. U. Diehl, B. U. Pedroni, A. Cassidy, P. Merolla, E. Neftci, and G. Zarrella, “TrueHappiness: Neuromorphic emotion recognition on TrueNorth,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2016-Octob, 2016, pp. 4278–4285.

[241] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[242] R. G. Leonard and G. Doddington, "Tidigits speech corpus," *Texas Instruments, Inc*, 1993.

[243] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu, "Space-time event clouds for gesture recognition: From RGB cameras to event cameras," in *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, 2019, pp. 1826–1835.

[244] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, 2018.

[245] S. Lobov, V. Mironov, I. Kastalskiy, and V. Kazantsev, "A spiking neural network in SEMG feature extraction," *Sensors (Switzerland)*, vol. 15, no. 11, pp. 27 894–27 904, 2015.

[246] P. Dallos, "The active cochlea," pp. 4575–4585, 1992.

[247] E. G. Walsh, "Experiments in Hearing," *Quarterly Journal of Experimental Physiology and Cognate Medical Sciences*, vol. 45, no. 3, pp. 324–325, 1960.

[248] L. Robles and M. A. Ruggero, "Mechanics of the mammalian cochlea," pp. 1305–1352, 2001.

[249] A. J. Hudspeth, "How the ear's works work," pp. 397–404, 1989.

[250] R. F. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 1982-May, 1982, pp. 1282–1285.

[251] M. Avissar, A. C. Furman, J. C. Saunders, and T. D. Parsons, "Adaptation reduces spike-count reliability, but not spike-timing precision, of auditory nerve responses," *Journal of Neuroscience*, vol. 27, no. 24, pp. 6461–6472, 2007.

[252] C. A. Peckens and J. P. Lynch, "Utilizing the cochlea as a bio-inspired compressive sensing technique," *Smart Materials and Structures*, vol. 22, no. 10, 2013.

[253] C. Valentini-Botinhao, "Noisy speech database for training speech enhancement algorithms and TTS models, 2016 [sound]," University of Edinburgh. School of Informatics. Centre for Speech Technology Research (CSTR)., Edinburgh, Tech. Rep., 2017. [Online]. Available: https://datashare.is.ed.ac.uk/handle/10283/2791

[254] S. F. Boll, "Suppression of Acoustic Noise in Speech Using Spectral Subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979.

[255] Y. Ephraim and D. Malah, "Speech Enhancement Using a Minimum Mean-Square Error Short-Time Spectral Amplitude Estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[256] Q. Zhang and M. Wang, "Speech enhancement for nonstationary noise environments," in *International Conference on Communication Technology Proceedings, ICCT*, vol. 2017-Octob, 2018, pp. 1663–1667.

[257] I. Cohen, "Noise spectrum estimation in adverse environments: Improved minima controlled recursive averaging," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, pp. 466–475, 2003.

[258] F. Xie and D. van Compemolle, "A family of MLP based nonlinear spectral estimators for noise reduction," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2, 1994, pp. II53–II56.

[259] J. Tchroz and B. Kollmeier, "SNR estimation based on amplitude modulation analysis with applications to noise suppression," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 3, pp. 184–192, 2003.

[260] Y. Wang and D. L. Wang, "Towards scaling up classification-based speech separation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 7, pp. 1381–1390, 2013.

[261] A. Narayanan and D. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013, pp. 7092–7096.

[262] L. F. Abbott, "The timing game," pp. 115–116, 2001.

[263] C. Glackin, L. Maguire, L. McDaid, and J. Wade, "Lateral inhibitory networks: Synchrony, edge enhancement, and noise reduction," in *Proceedings of the International Joint Conference on Neural Networks*, 2011, pp. 1003–1009.

[264] J. Wall, C. Glackin, N. Cannings, G. Chollet, and N. Dugan, "Recurrent lateral inhibitory spiking networks for speech enhancement," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2016-Octob, 2016, pp. 1023–1028.

[265] VoxForge, "Free Speech... Recognition (Linux, Windows and Mac) - voxforge.org," 2007. [Online]. Available: http://www.voxforge.org/

[266] R. D. Patterson, I. Nimmo-Smith, J. Holdsworth, and P. Rice, "An efficient auditory filterbank based on the gammatone function," in *a meeting of the IOC Speech Group on Auditory Modelling at RSRE*, vol. 2, no. 7, 1987.

[267] J. N. Gowdy and Z. Tufekci, "Mel-scaled discrete wavelet coefficients for speech recognition," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, vol. 3. IEEE, 2000, pp. 1351–1354.

[268] G. Deshmukh, A. Gaonkar, G. Golwalkar, and S. Kulkarni, "Speech based emotion recognition using machine learning," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2019, pp. 812–817.

[269] A. Bhavan, P. Chauhan, R. R. Shah *et al.*, “Bagged support vector machines for emotion recognition from speech,” *Knowledge-Based Systems*, vol. 184, p. 104886, 2019.

[270] D. Issa, M. F. Demirci, and A. Yazici, “Speech emotion recognition with deep convolutional neural networks,” *Biomedical Signal Processing and Control*, vol. 59, p. 101894, 2020.

[271] A. Christy, S. Vaithyasubramanian, A. Jesudoss, and M. A. Praveena, “Multimodal speech emotion recognition and classification using convolutional neural network techniques.”

[272] B. Raman and P. P. Roy, “A segment level approach to speech emotion recognition using transfer learning.”

[273] K. Venkataramanan and H. R. Rajamohan, “Emotion recognition from speech,” *arXiv preprint arXiv:1912.10458*, 2019.

[274] M. Sajjad, S. Kwon *et al.*, “Clustering-based speech emotion recognition by incorporating learned features and deep bilstm,” *IEEE Access*, vol. 8, pp. 79 861–79 875, 2020.

[275] T. Tashan, T. Allen, and L. Nolle, “Speaker verification inspired by the physiology of hearing using spiking self-organising map,” *Expert Systems*, vol. 34, no. 5, 2017.

[276] S. R. Livingstone and F. A. Russo, “The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north American english,” *PLoS ONE*, vol. 13, no. 5, 2018.

[277] O. Martin, I. Kotsia, B. Macq, and I. Pitas, “The eNTERFACE’05 Audio-Visual emotion database,” in *ICDEW 2006 - Proceedings of the 22nd International Conference on Data Engineering Workshops*, 2006.

[278] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, and B. Weiss, “A database of German emotional speech,” in *9th European Conference on Speech Communication and Technology*, 2005, pp. 1517–1520.

[279] ——, “A database of German emotional speech,” in *9th European Conference on Speech Communication and Technology*, 2005, pp. 1517–1520.

[280] E. Mansouri-Benssassi and J. Ye, “Speech emotion recognition with early visual cross-modal enhancement using spiking neural networks,” in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[281] P. Shegokar and P. Sircar, “Continuous wavelet transform based speech emotion recognition,” in *2016, 10th International Conference on Signal Processing and Communication Systems, ICSPCS 2016 - Proceedings*, 2016.

[282] J. Zhao, X. Mao, and L. Chen, “Speech emotion recognition using deep 1D & 2D CNN LSTM networks,” *Biomedical Signal Processing and Control*, vol. 47, pp. 312–323, 2019.

[283] A. Bhavan, P. Chauhan, Hitkul, and R. R. Shah, “Bagged support vector machines for emotion recognition from speech,” *Knowledge-Based Systems*, vol. 184, 2019.

[284] M. A. Jalal, E. Loweimi, R. K. Moore, and T. Hain, “Learning temporal clusters using capsule routing for speech emotion recognition,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2019-Septe, 2019, pp. 1701–1705.

[285] Y. Zeng, H. Mao, D. Peng, and Z. Yi, “Spectrogram based multi-task audio classification,” *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 3705–3722, 2019.

[286] Z. Huang, M. Dong, Q. Mao, and Y. Zhan, “Speech emotion recognition using cnn,” in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM 14. New York, NY, USA: Association for Computing Machinery, 2014, p. 801804. [Online]. Available: https://doi.org/10.1145/2647868.2654984

[287] W. Lim, D. Jang, and T. Lee, “Speech emotion recognition using convolutional and recurrent neural networks,” in *2016 Asia-Pacific signal and information pro-*

cessing association annual summit and conference (APSIPA). IEEE, 2016, pp. 1–4.

[288] E. Guizzo, T. Weyde, and J. B. Leveson, “Multi-time-scale convolution for emotion recognition from speech audio signals,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2020, pp. 6489–6493.

[289] F. Noroozi, M. Marjanovic, A. Njegus, S. Escalera, and G. Anbarjafari, “Audio-Visual Emotion Recognition in Video Clips,” *IEEE Transactions on Affective Computing*, vol. 10, no. 1, pp. 60–75, 2019.

[290] N. Hajarolasvadi and H. Demirel, “3D CNN-based speech emotion recognition using k-means clustering and spectrograms,” *Entropy*, vol. 21, no. 5, 2019.

[291] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, “Sparse autoencoder-based feature transfer learning for speech emotion recognition,” in *Proceedings - 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, ACII 2013*, 2013, pp. 511–516.

[292] T. Özseven, “A novel feature selection method for speech emotion recognition,” *Applied Acoustics*, vol. 146, pp. 320–326, 2019.

[293] T. Giannakopoulos, “A method for silence removal and segmentation of speech signals, implemented in Matlab,” Tech. Rep., 2009.