LIQUID STATE MACHINE MODEL WITH HOMEOSTASIS AND SUPERVISED STDP ON NEUROMORPHIC LOIHI PROCESSOR

A Thesis

by

ASHVIN SHENOY RENJAL

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Peng Li
Committee Members,	Andrew Jiang
	Srinivas Shakkottai
Head of Department,	Miroslav M. Begovic

December 2019

Major Subject: Computer Engineering

ABSTRACT

This research focuses on the implementation of the Liquid State Machine model with Intrinsic Plasticity (IP) for the reservoir layer and Synaptic Plasticity for the readout layer on Intel's new digital neuromorphic processor Loihi. Synaptic plasticity refers to the modification of weights in order to learn and infer certain patterns using a learning rule. The learning rule adopted for this model is supervised and local. Intrinsic plasticity refers to modification of neuronal states such as threshold voltage to maintain homeostasis. A Liquid State Machine Model with the combination of a homeostatic rule and a local learning rule is created on the Loihi platform and benchmarked on a speech dataset to verify its performance.

DEDICATION

To my mother, my father and my teachers

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Dr. Peng Li for giving me an opportunity to pursue research in neuromorphic computing and giving inputs to nudge me in the right direction. I also thank the committee members, Dr. Andrew Jiang and Dr. Srinivas Shakkottai for guidance and support. Special thanks to Wenrui Zhang who cleared my innumerable queries related to the topic as well as the Loihi chip. I would also thank Renqian Zhang, Yu Liu, Jeongjun Lee and other research group members for providing insight into the topic through interesting discussions which helped me in my initial phase of research.

Last but not the least, I would like to thank Intel's Neuromorphic Computing Lab for providing access to Loihi and clearing related queries in timely manner without which this research wouldn't have been possible.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Peng Li[advisor] and Dr. Srinivas Shakkottai of the Department of Electrical and Computer Engineering and Dr. Andrew Jiang of the Department of Computer Science and Engineering. Intel provided access to Loihi and tutorials on its software development kit.

Wenrui Zhang developed the initial framework for Loihi, data-set in the relevant format and the mathematical derivation of SpiKL-IP rule in the threshold voltage format. All other work conducted for the thesis was completed by me independently.

Funding Sources

Graduate study was supported by a fellowship from the Department of Electrical and Computer Engineering at Texas A&M University.

NOMENCLATURE

IP	Intrinsic Plasticity
SNN	Spiking Neural Network
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
SVM	Support Vector Machine
IF	Integrate and Fire
LIF	Leaky Integrate and Fire
LSM	Liquid State Machine
LTP	Long Term Potentiation
LTD	Long Term Depression
STDP	Spike Timing Dependent Plasticity
S-STDP	Supervised-Spike Timing Dependent Plasticity
ReSuMe	Remote Supervised Method
API	Application Programming Interface
SNIP	Sequential Neural Interfacing Processes
SP	Separation Property
AP	Approximation Property
BSA	Ben's Spiker Algorithm
INRC	Intel Neuromorphic Research Community
TAMU	Texas A&M University

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Spiking neural networks (SNN) [1] [2] [3] consist of fundamental computational units called neurons communicating through sequence of spikes called spike train. The spike train can be modelled as a Dirac Comb or simplified as a sequence of bits in a digital platform. These networks have been demonstrated to be computationally more powerful [4] than the conventional neural networks such as feedforward neural networks and recurrent neural networks[5] popular today. The spiking neuron model is also biologically plausible and provides intrinsic tuning parameters like threshold voltage. In addition, low-power neuromorphic hardware[6][7][8][9] is advantageous in applications involving edge computing and self driving cars which are conventionally done by power consuming GPUs today.

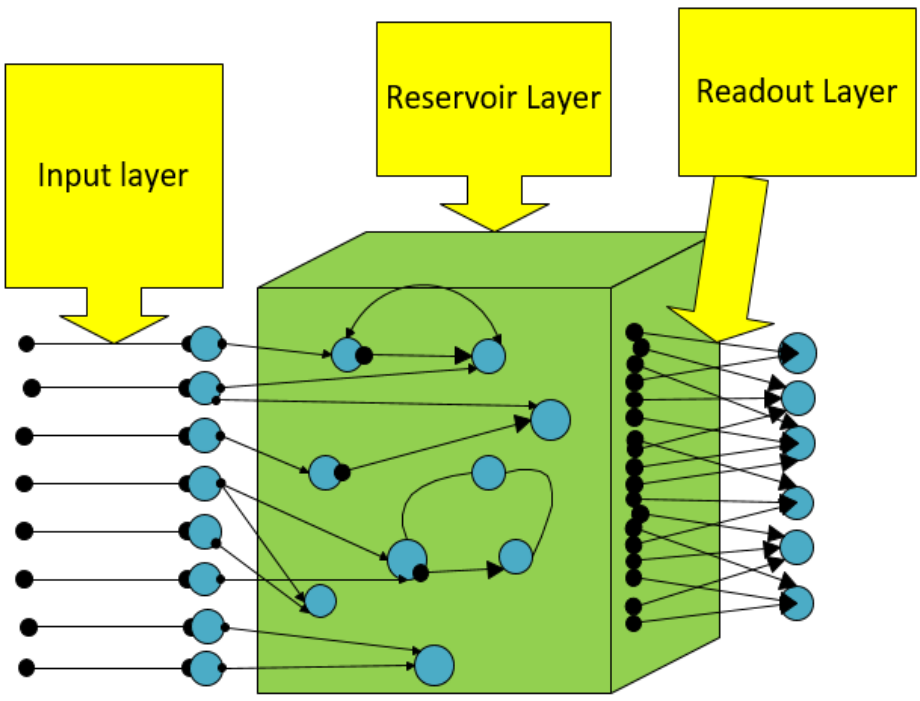

Figure 1.1: The LSM model

One of the biologically plausible models of SNN is called Liquid state machine (LSM) model [10], a specific form of reservoir computing [11]. As shown in figure 1.1, the LSM model consists of the input layer receiving the input spikes, which is randomly connected to the reservoir (also called liquid) layer followed by a fully connected readout layer.

The liquid layer consists of neurons with recurrent, static but random connections. This layer non-linearly projects the dataset to a higher dimension like in the case of Support Vector Machine (SVM) [12]. However, reservoir layer has recurrent connections which gives it a temporal benefit as in the case of Recurrent Neural Network (RNN) [13]. The advantages of this model over RNN are two-fold [14]. Firstly, the weights of liquid layer are static, unlike the recurrent weights in RNN which are harder to train. Secondly, the static liquid layer could be connected to multiple output layers and thus provide parallelism.

The readout weights between the reservoir and the readout layer are fully connected and modified using a local learning rule which is linear in nature. It can range from linear discriminant[15] to bio-inspired rules like Spike Timing Dependent Plasticity (STDP) [3]. The local nature of the rule provides computational advantage over conventional Back-Propagation [16] rules used in multi-layer perceptron in addition to significant power saving [7]. Although STDP is unsupervised in nature, there are supervised versions such as ReSume[17] and S-STDP[18]. Such a modification of weights based on a learning rule is called synaptic plasticity. The term is based on the biological term synapse[19], which is the connection between two neurons through which spikes are transferred.

Although the weights are not modified in the liquid layer, certain parameters like threshold voltage of the reservoir neurons could be modified to give a desired output across the readout layer. This is called Intrinsic Plasticity (IP) [20] where intrinsic refers to the tuning parameter internal to the neuron model (in this case it is threshold voltage). These rules, which are unsupervised in nature, aim to maintain neuronal activity within the desired threshold [21] or create an optimum distribution for the neuronal firing rate. An example for the former is the activity range homeostasis[8] where the threshold voltage is updated based on activity trace of the individual

neurons. The SpiKL-IP rule[22] is an example for the latter where the ultimate goal would be to produce an optimal exponential distribution of firing rate for the reservoir neurons.

2.1 Brief overview of Spiking Neuron Model

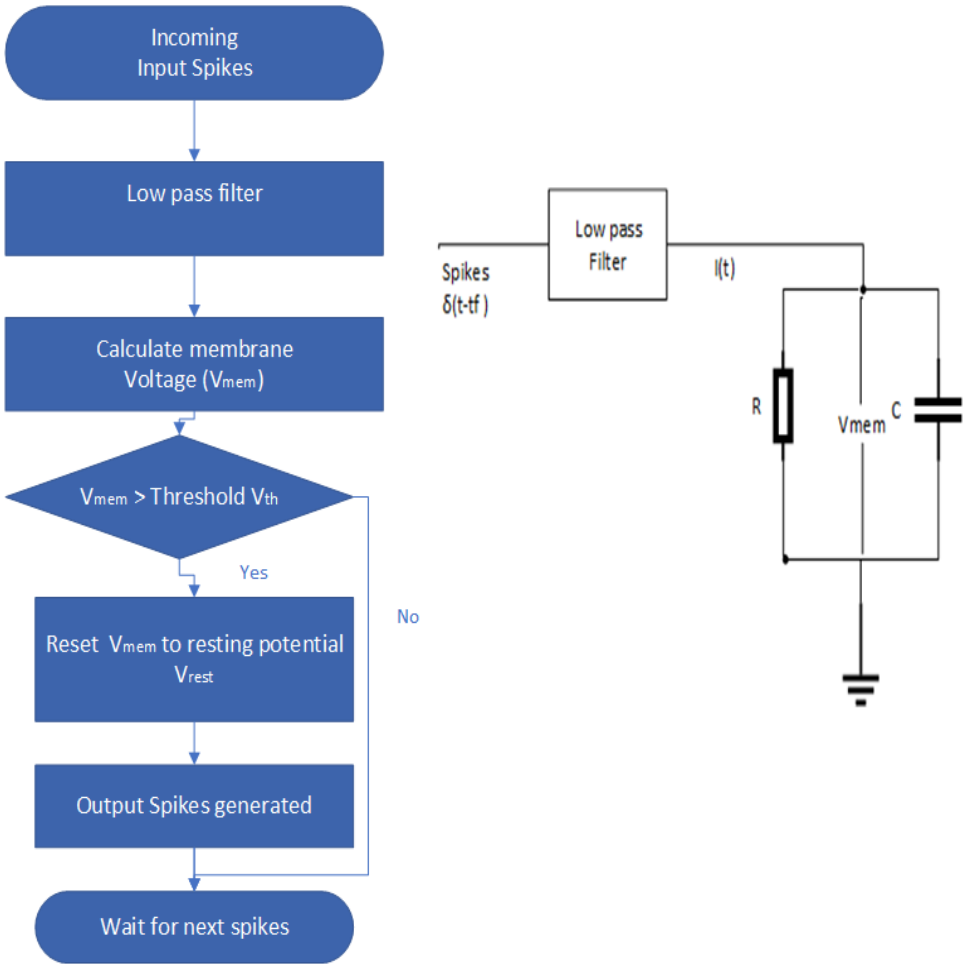

Figure 2.1: The LIF neuron model flowchart and circuit

The brain consists of fundamental units called neurons which communicate through signals called action potentials [23]. The typical action potential is a continuous curve with different portions pointing to different phases in its generation by a neuron. However, the information carried from one neuron to another lies in the sequence, timing and count of these signals rather than their

individual shapes [3]. Hence, action potentials can be represented as spikes and a sequence of them is called a spike train. These spikes trains can be approximated by Dirac comb $\sum_f \delta(t - t_f)$ or sequence of bits in the case of digital signals.

There are several neuron models ranging from rigorous models such as Hodgkin Huxley model [24] to simpler mathematical models such as McCulloch-Pitts neuron [25] and perceptron [26]. However, all the neurons based on these models have the following common principles [3]. They are multiple input single output structures, their output is enhanced by certain excitatory inputs and repressed by inhibitory inputs and their output is governed by at least one state variable. In this document, the neuronal units are based on the Leaky Integrate and Fire (LIF) model [1][3] unless otherwise specified.

2.2 Leaky Integrate and Fire Model [1]

The flowchart and circuit of the LIF model is indicated in figure 2.1. The incoming input spikes are integrated by a low pass filter into a continuous current. The filter may be of first, second or higher order. The current from several inputs (or synapses) are weighted and fed into the RC circuit which models the neuronal membrane. The Capacitance acts as an Integrator and Resistor R acts as the leak,giving it the name Leaky Integrate and Fire. If the membrane voltage (V_{mem}) measured across the capacitance is greater than some threshold voltage (V_{th}), then the neuron fires, producing an output spike train which then acts as an input to other neurons in the network. V_{mem} is then reset to the resting potential (V_{rest} usually 0). This process can be summarized by the differential equation 2.1

$$\frac{\mathrm{d}V_m(t)}{\mathrm{d}t} = -\frac{V_m}{\tau} + \sum_i \sum_j w_{mi} \cdot s(t - t_{ij} - d_i) \tag{2.1}$$

where V_m and τ are the membrane voltage and time constant of the m-th neuron. w_{mi} is the weight connecting the m-th post-synaptic neuron to the i-th pre-synaptic neuron. The function $s()$ indicates the synaptic response which is the output of the low pass filter. t_{ij} is the time of spiking of the j-th spike by the i-th synaptic neuron. d_{ij} is the synaptic delay which is a property

of the synapse. In addition to these parameters, there is the refractory delay t_r not included in the equation and in the figure 2.1. This is the duration after the firing of neuron for which all the inputs will be ignored and the membrane voltage won't be accumulated.

2.3 Hebbian Learning

The principle of Hebbian Learning [27] states that the synaptic strength between the neurons depend solely on their correlated activities. In simple terms, "the neurons that fire together wire together". If x_i and y_j are the activities of presynaptic and postsynaptic neurons respectively, then the change in weight Δw is given by

$$\Delta w \propto x_i \cdot y_j \tag{2.2}$$

Similarly, there is the principle anti-hebbian learning where correlated input and output spikes lead to the depression of weights.

2.4 Spike Timing Dependent Plasticity

There are several unsupervised learning rules based on Hebbian Learning. One of them could be as follows. It was observed in [28] that there is an increase in synaptic weight when the postsynaptic spike occurs after the presynaptic spike thus giving rise to Long Term Potentiation(LTP)[29]. Similarly, there is a decrease in synaptic weight when there is a postsynaptic spike before the presynaptic spike, resulting in Long Term Depression(LTD)[30]. This biological process of synaptic weight change occurring due to temporal correlation of input and output spikes is called Spike Timing Dependent Plasticity(STDP)[3].

These weight changes are governed by the STDP curve as shown in figure 2.2. From this figure, we observe that closer input and output spikes result in greater weight change by magnitude. In the mathematical form, the STDP rule is given by the following equations.

$$\Delta W = \begin{cases} A_+(w) \cdot e^{-\frac{\Delta t}{\tau_+}}, & \text{if } \Delta t \geq 0 \\ A_-(w) \cdot e^{-\frac{\Delta t}{\tau_-}}, & \text{if } \Delta t < 0 \end{cases} \tag{2.3}$$

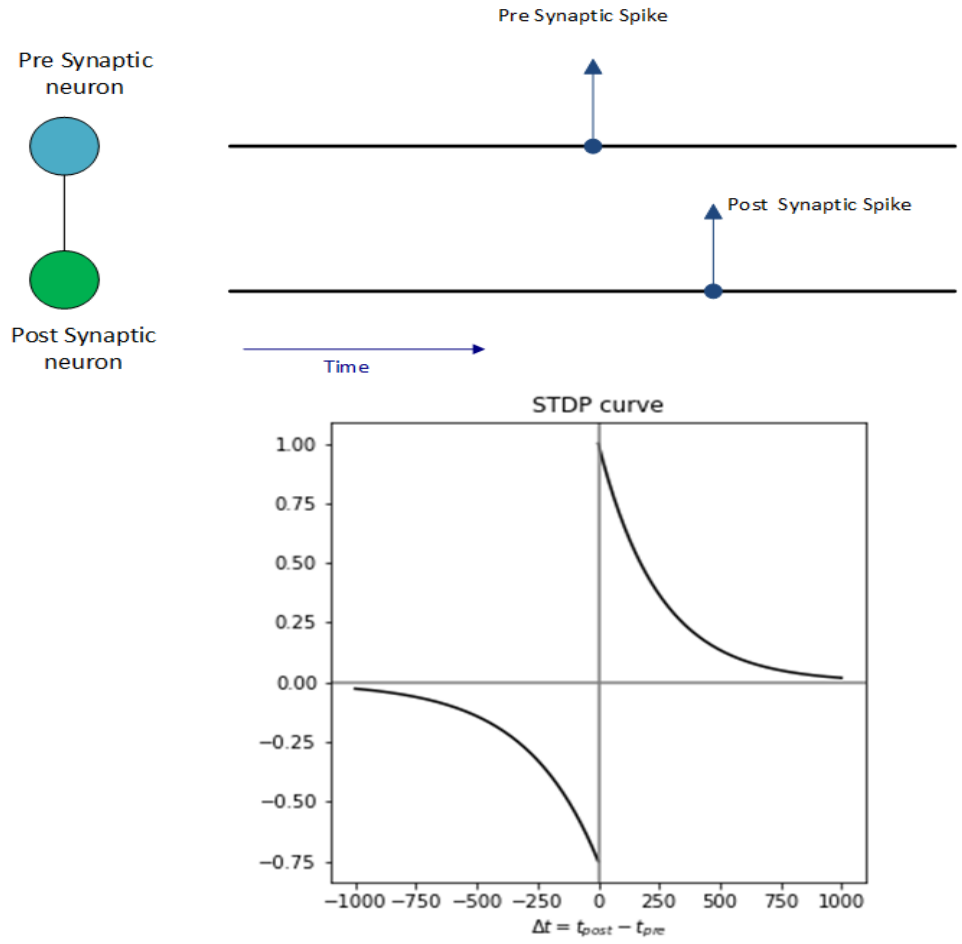

Figure 2.2: STDP curve

An interesting form from the perspective of implementation is the online form of STDP [31] given by the equation 2.4 for a postsynaptic spike j with weight w_j. Here $x(t)$ and $y(t)$ indicates input and output traces which are the low pass filtered versions of the respective spike trains. t_n represents the output spike time and t_f indicates the input spike time.

$$\frac{dw_j}{dt} = A_+(w_j)x(t)\sum_n \delta(t - t^n) - A_-(w_j)y(t)\sum_f \delta(t - t_j^f) \tag{2.4}$$

The STDP rule is a Hebbian rule which is unsupervised in nature. A supervised STDP rule called ReSume[3][17] was formulated combining both the Hebbian and anti-Hebbian principles with a form similar to Widrow-Hoff rule [32]

3. OVERVIEW OF LOIHI INFRASTRUCTURE

3.1 Neuromorphic architectures

The word neuromorphic was coined by Carver Mead in [33] to refer to the analog architectures which mimic the biological brain. Today the term encompasses a wide range of analog, digital and hybrid architectures. But this research will mainly focus on implementation of our LSM based model on Intel's neuromorphic digital processor Loihi [8].

3.2 Spiking Neural Unit in Loihi

The Spiking neural units are based on the simplified version of the CUBA model [34]. According to this model, there are two state variables for the neuron; synaptic current $u_i(t)$ and membrane voltage $v_i(t)$. The synaptic current is obtained by passing the input spikes through a low pass filter and multiplying them with their synaptic weights. It is given by equation 3.1, where $\alpha_u(t) = \frac{e^{-\frac{t}{\tau_u}}}{\tau_u} H(t)$ is the impulse response of the filter with time constant τ_u and the term $H(t)$ indicates the step function. The synaptic current is then accumulated to obtain the membrane voltage which is given by equation 3.2. If the membrane voltage crosses the threshold voltage V_{th}, then it is reset to 0 and a output spike is generated.

$$u_i(t) = \sum_{j \neq i} w_{ij}(\alpha_u * \sigma_j)(t) + b_i \tag{3.1}$$

$$\dot{v}_i(t) = -\frac{1}{\tau_v} v_i(t) + u_i(t) \tag{3.2}$$

3.3 Learning rule engine in Loihi

Loihi supports learning rules which are local in nature such as Spike Timing Dependent Plasticity (STDP) (see section 2.4). The learning rule should be written in the sum of product form [18] given by the following equation

$$z(t) = z(t-1) + \sum_m S_m \prod_n F_n \tag{3.3}$$

where z indicates the synaptic state variable (weight,delay and tag) and S is a constant. F can one of the following: input/output spikes, input/output traces or special functions such as $sign$ of any synaptic state variable. The input and output spikes/traces are denoted by x_k and y_k respectively where k can range from 0 to 3 with 0 indicating spikes and 1,2,3 representing different traces. The learning rule is applied at periodic intervals called epoch time (tEpoch) which can range from 1 to 63 but generally set as a power of 2 for efficiency. For example an online form of unsupervised STDP (see equation 2.4) can be written as

$$w(t) = w(t-1) + y_0 x_1 - x_0 y_1 \tag{3.4}$$

Another example would be the potentiation rule of Supervised STDP (S-STDP) [18] given by the equation 3.5 where S_1 and S_2 are constants. The term u_k is a dependency factor which evaluates the product at every $tEpoch \cdot 2^k$ timestep; at all other time steps the product is 0.

$$w(t) = w(t-1) - S_1 y_0 x_1 + S_2 u_k x_1 \tag{3.5}$$

3.3.1 Trace evaluation

The trace is the filtered version of the spike train which can be utilized for online learning. Loihi provides up to 2 presynaptic traces per input axons (with at most 2048 learning enabled input axons per core) and 3 postsynaptic traces per compartment (with at most 1024 compartments per core). Each trace has two components: the decay α and the impulse δ which is the gain added to the trace whenever there is a spike. The trace is evaluated [8] by the following equation where α indicates the decay and δ is the impulse for every spike $s[t]$

$$x[t] = -\alpha \cdot x[t-1] + \delta \cdot s[t] \tag{3.6}$$

3.4 Homeostasis

Loihi supports threshold voltage homeostasis rules. The chip also has an inbuilt homeostasis mechanism which is governed by a separate trace called activity trace. This trace variable has an impulse value and time constant like in the case of learning rule. If the activity goes beyond the upper and lower limit then threshold voltage is increased and reduced respectively to keep the neuronal firing rate under control. The equation for this rule is given by

$$\Delta V_{th} = \begin{cases} \beta \cdot (y - y_{max}), & \text{if } y > y_{max} \\ \beta \cdot (y - y_{min}), & \text{if } y < y_{min} \end{cases} \tag{3.7}$$

where ΔV_{th} is the change in threshold voltage, y is the activity trace, y_{min} and y_{max} are the lower and upper limits of activity trace respectively.

3.5 Software Infrastructure

The toolchain and procedure of programming Loihi along with architecture can be found in [18]. Loihi toolchain consists of python API at the front end which is used to create neural networks. The programming interface is user oriented and abstracts many details of the underlying hardware. The python code is then compiled and run on the hardware which is accessed via cloud. The programming model can be divided into three portions: Compartments analogous to the body of neuron, connections/synapses and the learning rules formulated with traces (see section 3.3).

3.5.1 Compartment

The compartment receives input spikes and integrates them to current and voltage. Loihi provides API to configure several parameters including refractory delay, homeostasis, axon delay, bias and threshold. A compartment prototype can also be declared which would help in the creation of groups of several compartments (shown in figure 3.1). Loihi also provides a spike generator object which provides input spikes to the compartments at run-time.

3.5.2 Connections

The connection object functions as a synapse between a compartment and another compartment/spike generator. The connections supports feedforward and recurrent connections. An important functionality of the connection object is that it hosts the learning rule engine and a reinforcement channel through which reinforcement spikes can be inserted at run-time.

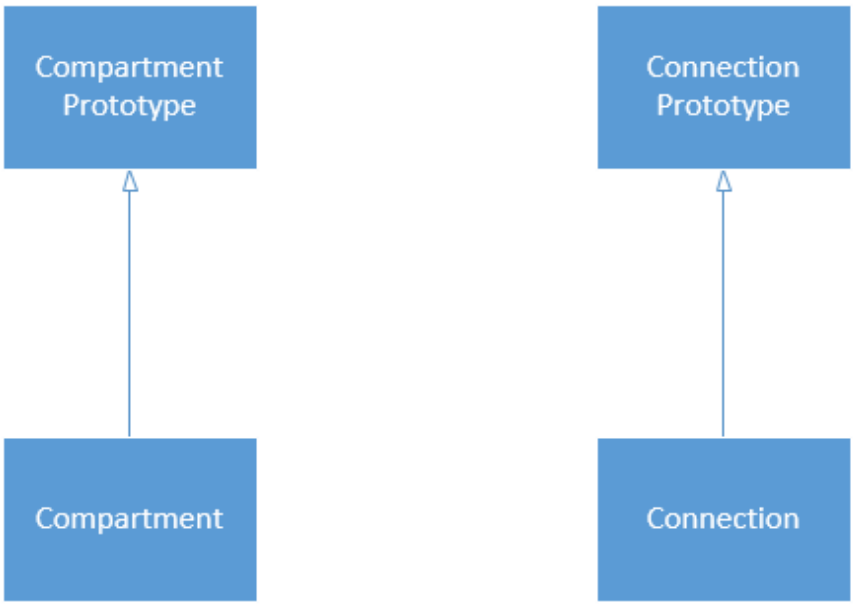

Figure 3.1: Prototypes on Loihi

Figure 3.2 block diagram shows a simple network with a compartment, connection and spike generator

3.5.3 SNIP

The python framework helps in creating a network and running it. But there may be cases where one would require actions such as pruning neurons, disabling and switching learning rules

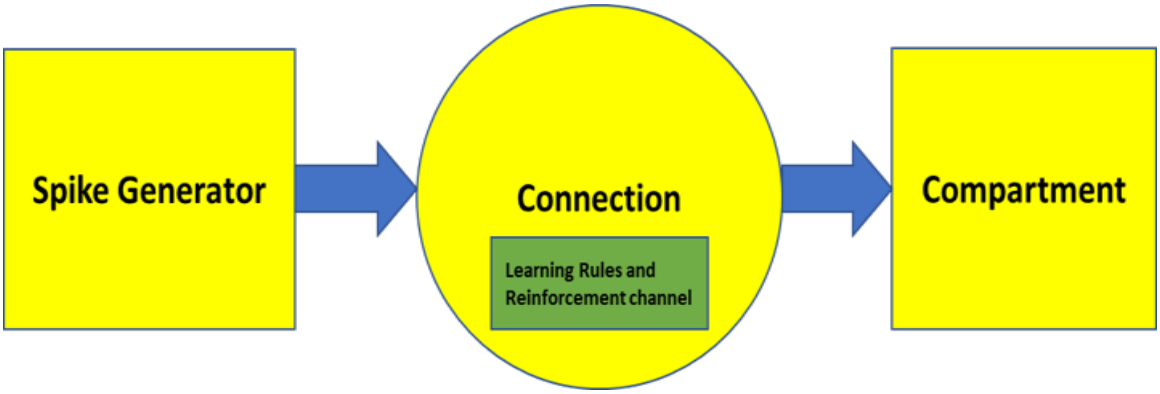

Figure 3.2: Block diagram of simple network on Loihi

or other run-time decisions which depend on fulfillment of spiking and timing conditions. SNIP (Sequential Neural Interfacing Processes) is a C based interface which provides access to certain Loihi parameters at runtime. It behaves like a controller which monitors the state of the neural networks and takes specific actions. The SNIP code directly accesses the registers at run-time to modify the network. The python and the SNIP block communicate via channels as shown in figure 3.3. These channels are initialised on startup. Separate channels can be created to send the data to SNIP as well as receive the data from it. In addition to switching learning rules, SNIPs can also be used to change bias, membrane time constant and threshold voltage at runtime.

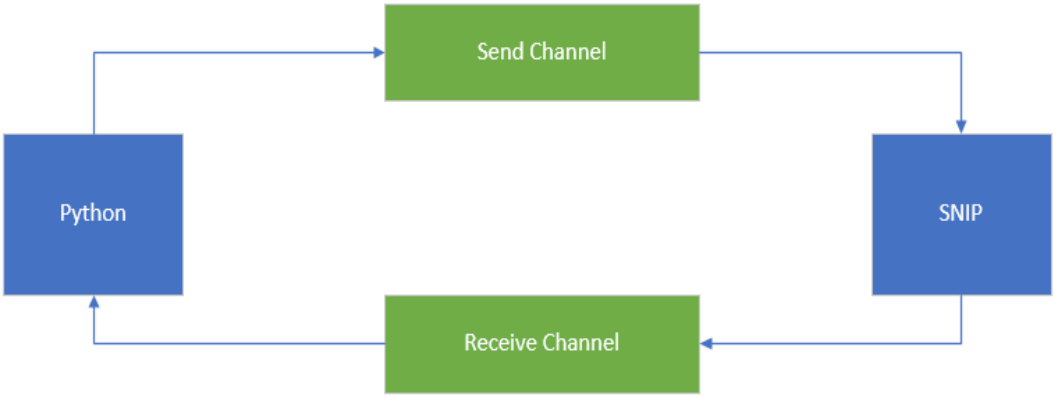

Figure 3.3: Python and SNIP communication

4. EXPERIMENTAL SETUP

The main goal of this research is to setup an LSM model with intrinsic plasticity at the reservoir layer and synaptic plasticity at the readout layer for a classification task. The model is then run on the Loihi processor utilizing its learning engine and software development kit. Since the chip and its setup is recent and novel, we face inherent challenges of adapting our model to this environment. The setup is then benchmarked on single speaker TI-alpha and TI-digits dataset [35][36].

4.1 Brief overview of the Liquid State Machine model

As mentioned in section 1, the LSM model contains a reservoir(liquid) layer which has recurrent connections. However, there can be ensembles of reservoirs too as demonstrated in [37]. The reservoir is a 3 dimensional structure with individual LIF neurons. These neurons project the input dataset to high dimensional space where the dataset is linearly separable. So two different inputs should result in different outputs from the reservoir if they have to be separable. This is called the Separation Property (SP) [10]. For the neurons i,j separated by Euclidean distance $D_{i,j}$ in the liquid layer, the probability of connection is given by equation 4.1 with λ and c as constants. These connections are static in nature. The liquid also demonstrates fading memory[38] and is time invariant which gives it universal computational power [10]

$$P_{i,j} = c \cdot e^{-\frac{D_{i,j}^2}{\lambda^2}} \tag{4.1}$$

The output of the reservoir layer is given to the memoryless (doesn't remember the previous input unlike the liquid) readout layer. The connections between these layers are plastic, modified using a simple learning rule.

4.2 LSM model of the setup

The setup on Loihi will be based on the LSM model. The recurrent structure of Liquid layer along with fading memory helps in processing temporal information efficiently. Hence the setup

will be benchmarked on a speech dataset. However, there are works such as [22] and [37] which were benchmarked on image datasets like MNIST[39] and Cityscape[40]. The LSM model is summarized in the figure 4.1.

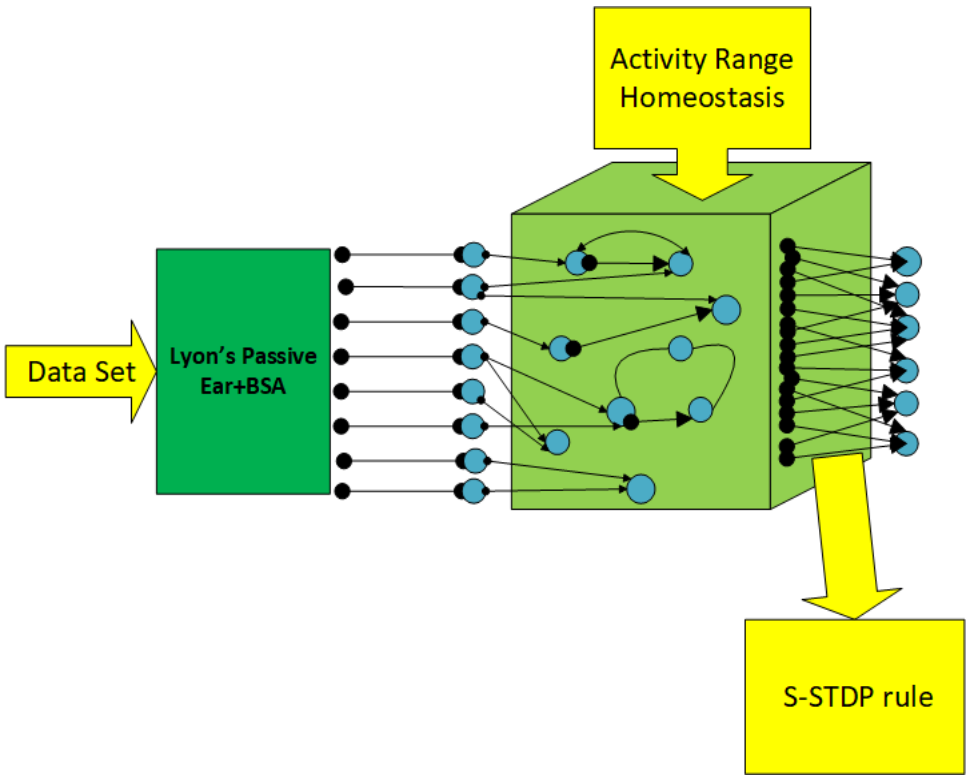

Figure 4.1: The proposed model

4.2.1 Input layer

The input layer has the function of data pre-processing and spike generation. This layer converts the speech dataset into spike trains. This objective is achieved by a combination of Lyon's cochlear model based on human ear [41] and BSA algorithm[42]. The methodology is explained in this work [43]. The Lyon's model consist of 78 cascaded band pass filter each of which produces signals of different frequency with 1 being the highest and 78 being the lowest. Each dataset signal is sent into these cascaded filters followed by a combination of half wave rectifier and automatic gain control layer which compresses the signal. All the 78 inputs generated is then converted to

78 individual spike trains by the BSA layer. However, the Loihi setup will receive the already preprocessed dataset as file which has the ports and the time at which these ports should produce the spikes listed.

4.2.2 Reservoir layer

The Reservoir layer consists of 135 LIF neurons in a 3D structure with dimensions 3x3x15 neurons. As mentioned earlier, the weights are recurrent and static with the probability of connection between any two neurons given by equation 4.1. The connections are either excitatory with positive weights or inhibitory with negative weights with the proportion of these weights decided by the application. We may use unsupervised methods such as homeostasis to give desirable output. In this work an inbuilt homeostasis setup provided by Loihi is used, which tweaks the threshold voltage based on activity range given by equation 3.7.

4.2.3 Readout layer

The readout layer has plastic synapses which are modified using a local learning rule. We employ Supervised STDP rule [18] which is inspired by the ReSuMe rule [17]. Since supervised learning is desired, the rule will have two individual rules. A potentiating rule which is applied to the weights of the desired neuron to strengthen its connections and a depressing rule which is applied to other neurons to suppress their weights.

$$PotentiatingRule : w(t) = w(t-1) - S_1 y_0 x_1 + S_2 u_k x_1 \tag{4.2}$$

$$DepressingRule : w(t) = w(t-1) - S_1 y_0 x_1 \tag{4.3}$$

4.3 Setup on Loihi

The setup consists two blocks: python block which is used to create,run the network and the C based SNIP(see section 3.5.3) which takes run-time decisions such as switching learning rules and disabling learning. The python code is object oriented with the object diagram given by the figure 4.2. All the compartment and connection objects are inherited from their respective prototypes as

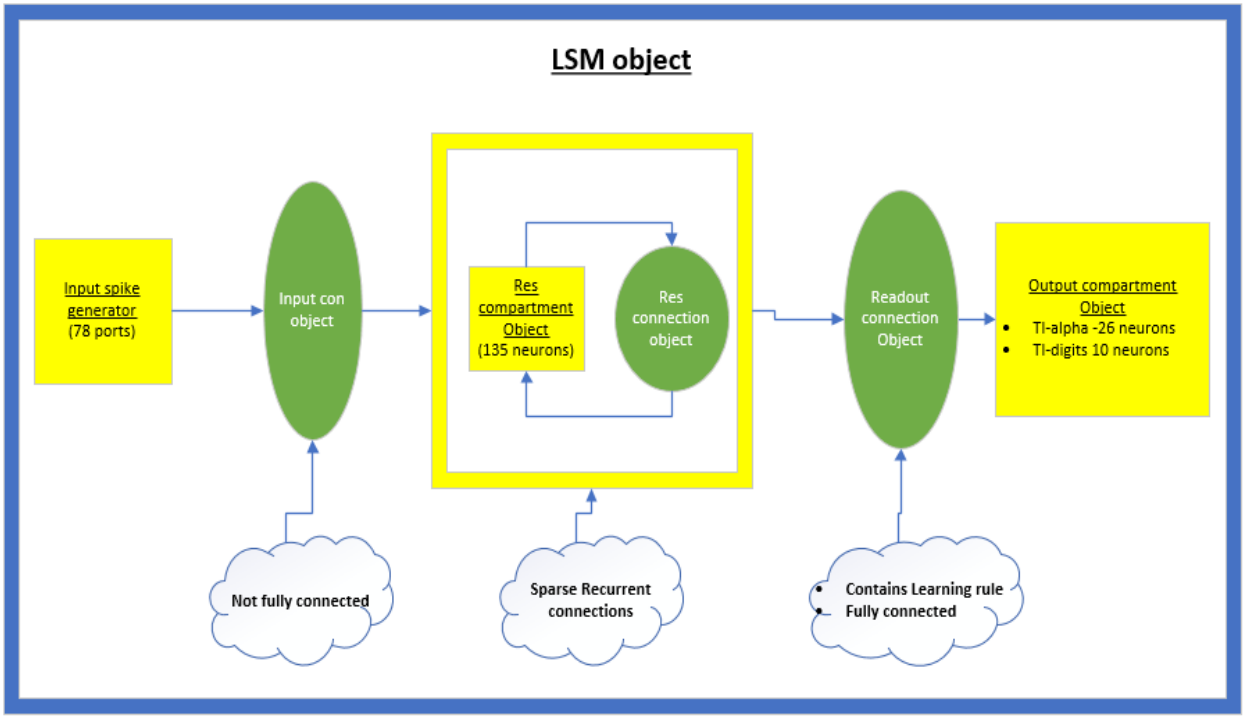

Figure 4.2: Block diagram of objects in network

shown in figure 3.1. As mentioned in the previous section, the Loihi setup will obtain preprocessed spikes directly as a file which is then parsed to give spike data. The LSM object created on start-up will have an input, a reservoir and an output compartment object. The input compartment is a spike generator with 78 input ports which is connected to the reservoir compartment object through the input connection object. Each input port will have 32 random connections to any of the 135 reservoir neurons. The reservoir compartment object is recurrently connected via reservoir connection object. These connections are sparse and probabilistic governed by equation. 4.1. The reservoir and readout compartments are fully connected by the readout connection object which also contains the potentiating and depressing learning rule. At run-time, these learning rules are applied to desired and undesired neurons at the start of each sample. The number of neurons in the output compartment depends on the dataset.

4.3.1 Working of setup

The python setup consists of a main file from which all the decisions are made. The steps consist of initialization, creation of LSM object, compiling and running the network which is shown in figure 4.3.

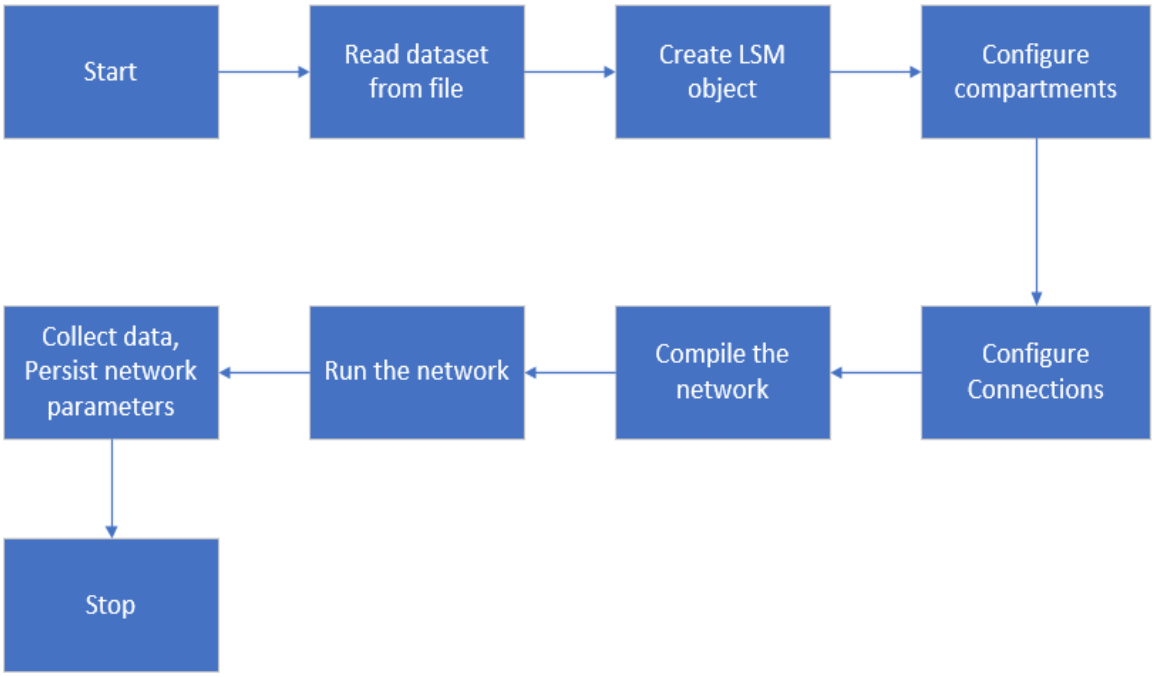

Figure 4.3: Block diagram of working of setup

The dataset is read from the file via a parser and passed to the LSM object. Additional parameters such as number of epochs, duration of each sample,number of testing and training samples are also provided here. The compartments and their respective connections are then configured with the required parameters such as weights. The code is then compiled and run. At the end, the data is collected and the state of the network is persisted on to a text file.

The compile phase also consists of additional activities such as initializing probes (monitors the

state variables for data collection), SNIP (see section 3.5.3) and the send channels. This is shown in figure 4.4. There are two channels to send the data from the running python code to SNIP. Init channel is used to send the network configuration parameters on start-up such as sample duration, number of neurons in reservoir and readout and the core id of Loihi (where the network is located in the chip). Label channel sends the ground truth label on the start of every sample. This is done because the learning rules are switched between the label and undesired neurons in the SNIP. The run phase is shown in figure 4.5

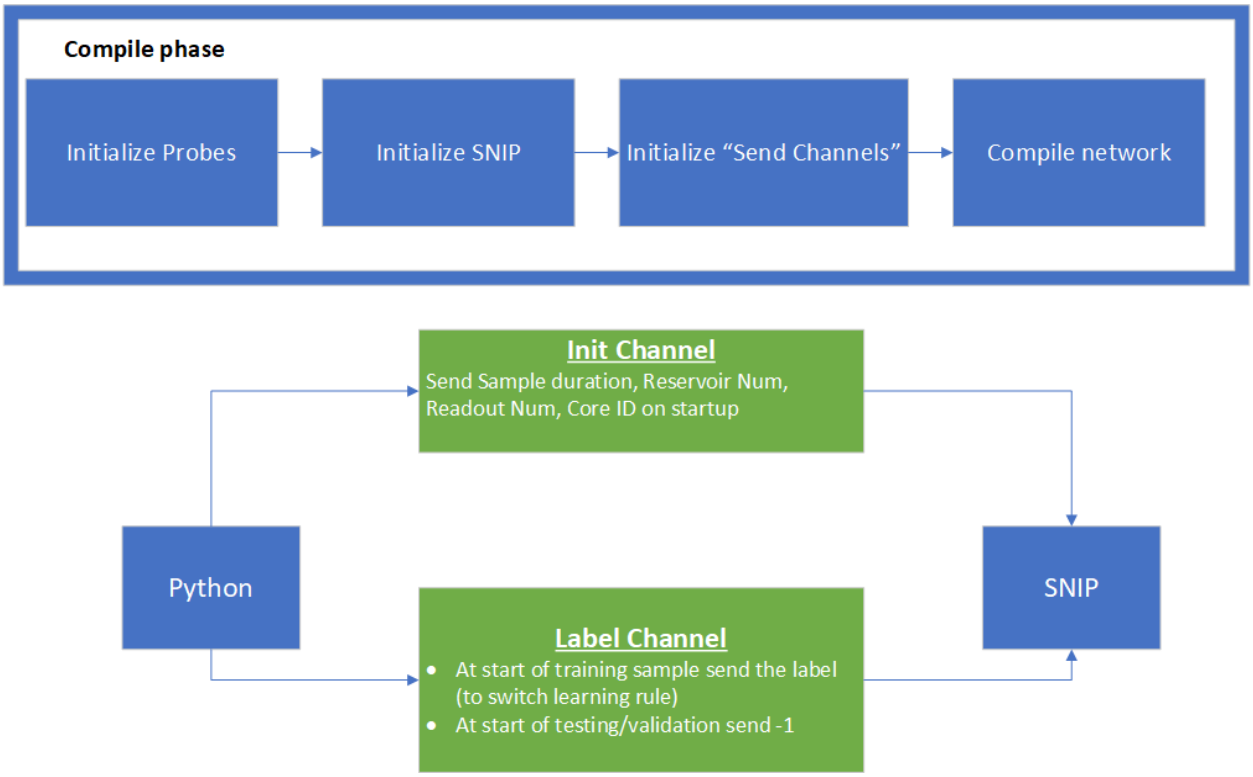

Figure 4.4: Steps in compile phase and the channels

4.3.2 SNIP in the setup

The SNIP performs the task of switching the potentiating and depressing learning rules between the label and other neurons in the training phase and disables learning at the beginning of the testing

phase. It also resets the state variables such as membrane voltage and current, and sets the learning and activity traces to zero at the beginning of each sample.

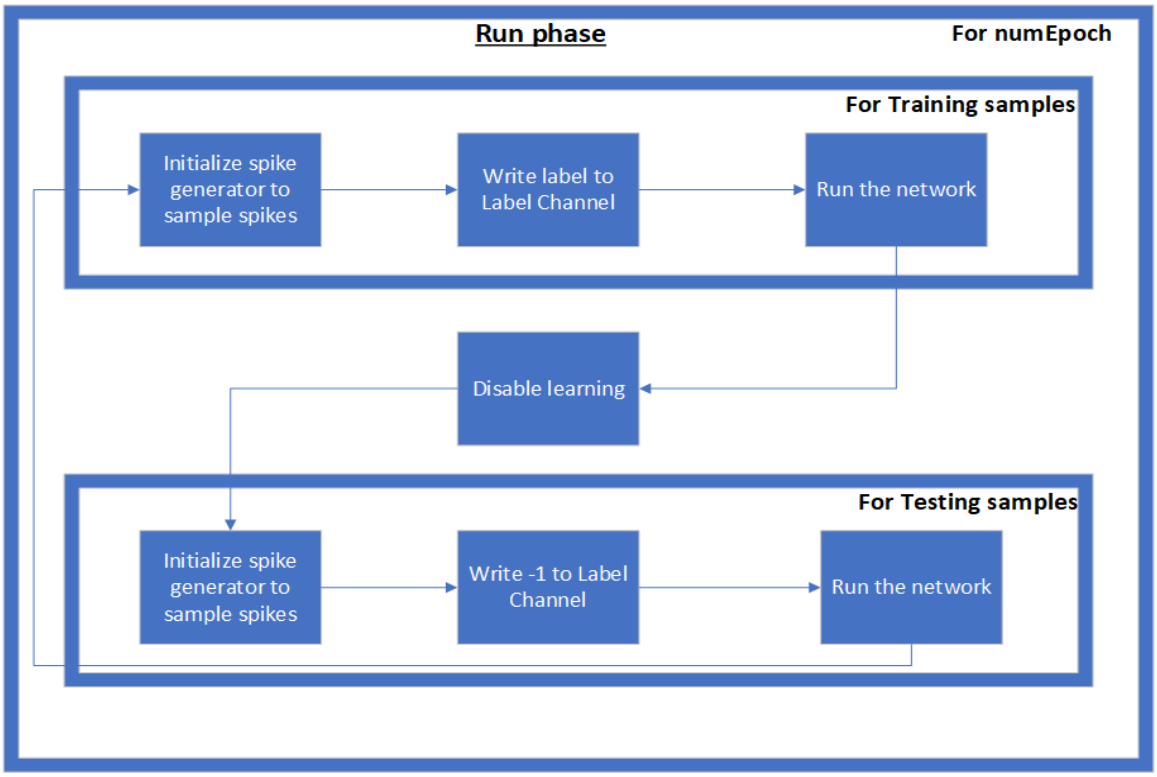

Figure 4.5: Steps in run phase

The network is benchmarked on a subset of TI-alpha and TI-digits datasets[35]. Only a single speaker subset is used, as the Loihi device, accessed via cloud, is time-shared between all the participating teams. The size of the dataset is summarized in the table 5.1. The single speaker dataset is divided in the ratio 8:2 between training and testing samples without cross validation.

Dataset	**Number of classes**	**Number of training samples**	**Number of testing samples**
TI-alpha	26	208	52
TI-digits	10	80	20

Table 5.1: Dataset summary

5.1 Parameters of the network

The sample duration is 700 time steps in Loihi units. The layer wise parameters are shown below.

5.1.1 Input layer

The input layer consists of a spike-generator which is connected to the reservoir layer with parameters in table 5.2. These individual elements of the spike generator aren't neurons but rather simple signal generators without any neuronal parameters such as threshold voltage.

Parameters	**Value**
Ports	78
Connection	Each port connected to 32 random reservoir neurons
Excitatory Probability	0.5
Excitatory weight	128
Inhibitory weight	-128

Table 5.2: Input layer parameters

5.1.2 Reservoir layer

The reservoir layer is recurrently connected as per the equation 4.1 and the parameters are shown in in table 5.3. Activity range homeostasis (see section 3.4) is enabled for this layer. The threshold voltage for the reservoir neurons are lower than the readout neurons. The minimum activity level is kept at 0 (lowest possible value since activity is non-negative) and maximum is kept at 5. The recurrent weights are either excitatory or inhibitory depending on the probability. The connection probability between these neurons also depends on their type.

Parameters	**Value**
Number of neurons	135
Dimensions	3x3x15
Threshold Voltage	80
Compartment voltage time constant	32
Compartment Current time constant	8
Refractory Delay	2
Homeostasis gain	5
Activity Impulse	1
Activity time constant	64
Activity max limit	5
Activity min limit	0
Excitatory Probability	0.8
E->E/I weight	32
I->I/E weight	-32
E->E c (see equation 4.1)	0.3
E->I c	0.2
I->E c	0.4
I->I c	0.1

Table 5.3: Reservoir layer parameters

5.1.3 Readout layer

The reservoir layer and readout layer are fully connected. The range of readout weight is $[-256, 255]$ which is same as the range of weights for Loihi. The weight values are integers, ran-

domly chosen from the range for each connection based on uniform distribution. The connection also has the learning rule enabled given by equation 4.2 and 4.3.

Parameters	**Value**
Number of neurons	26/10
Threshold Voltage	5120
Compartment voltage time constant	32
Compartment Current time constant	8
Refractory Delay	2
Homeostasis	Disabled

Table 5.4: Readout compartment parameters

All the readout connections will have learning rule enabled. As per the Loihi documentation, the tEpoch value (see section 3.3) should be a power of 2. The learning rule parameters for readout connections are shown in table 5.5. If the constants S_1 and S_2 are equal then the potentiating rule reduces to the ReSuMe rule[3]. For the setup we choose a higher tEpoch value 8, with impulse value of 127 (maximum possible) and time constant of input trace as 10. This results in shorter traces which decline quickly.

Parameters	**Value**
Connections	Fully connected to reservoir
Weight range	[-256,255]
tEpoch	8
S_1	2^{-6}
S_2	2^{-5}
u_k (see section 3.3)	u_1
Input trace(x_1) impulse	127
Input trace(x_1) time constant	10

Table 5.5: Readout connection parameters

5.2 Results

As mentioned earlier only the single speaker subset of the TI-alpha and TI-digits are used due to the time limitation in running the setup.

5.2.1 TI-digits

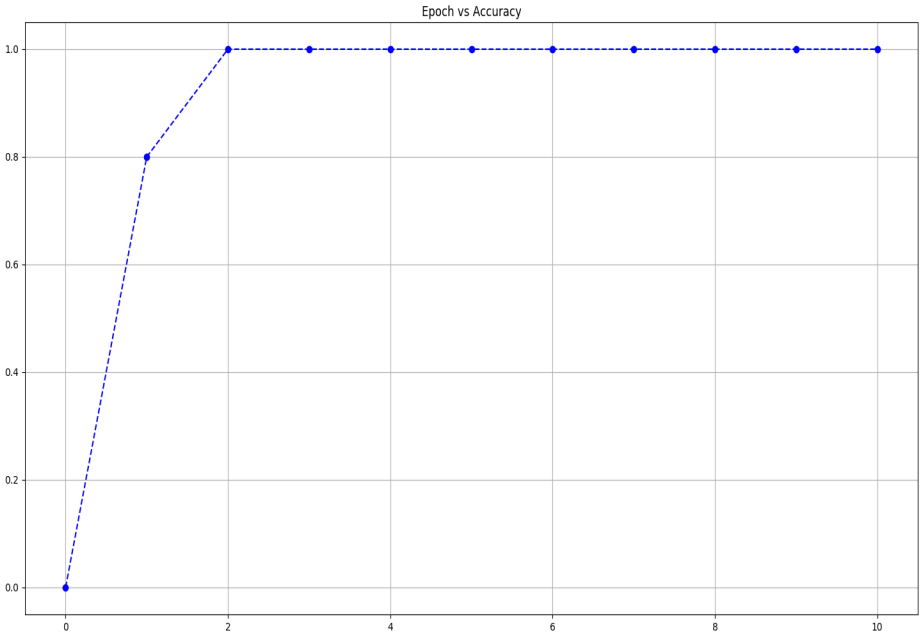

Figure 5.1: Accuracy plot for TI-digits dataset

In the case of TI-digits dataset, the setup is able to complete the classification within 10 epochs. Since the dataset is simpler in nature, the same classification accuracy can be achieved by disabling homeostasis. The result is shown in figure 5.1. The accuracy achieved is 100% for the given subset. However, the same performance cannot be guaranteed for the whole dataset.

5.2.2 TI-alpha

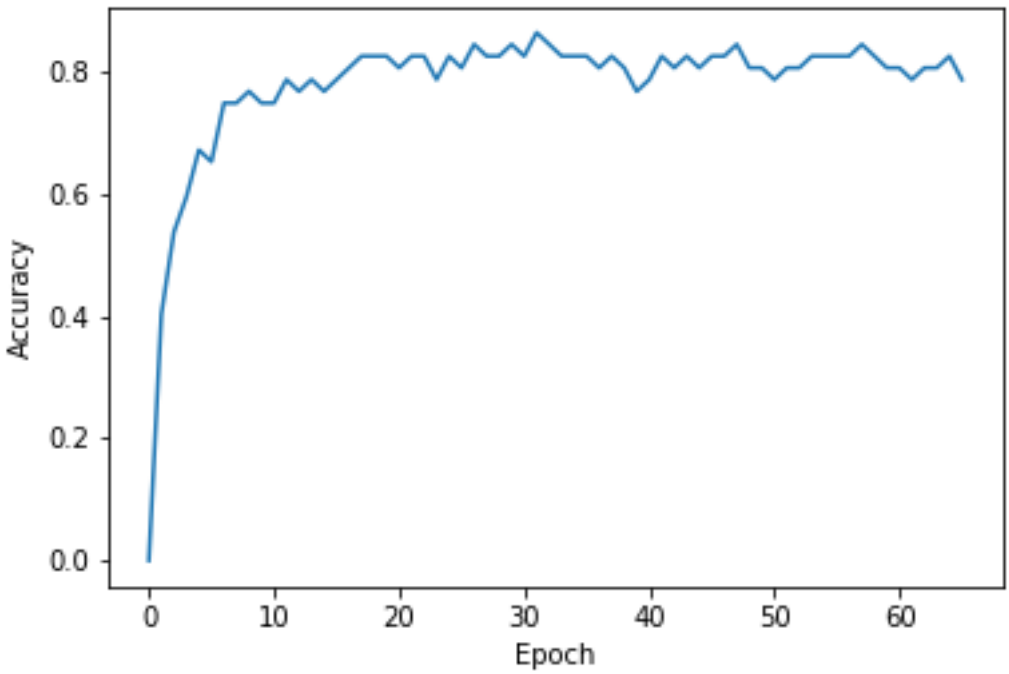

Figure 5.2: Accuracy plot for TI-alpha dataset

In the case of single speaker TI-alpha dataset, the performance is very poor without homeostasis. This is because the dataset contains similar sounding classes(alphabets), which the learning rule finds harder to classify. However with homeostasis enabled, a maximum accuracy of 86.54% (about 45 of 52 samples) could be classified correctly. Similar sounding alphabets such as B-D-G or V-Z couldn't be classified,with the last alphabet encountered during training dominating. The result is shown in figure 5.2.

6.1 Future work

6.1.1 SpiKL-IP

The intrinsic plasticity rule called SpiKL-IP [22] applied to the reservoir layer, utilizes KL divergence to produce an exponential distribution of firing rate. The challenge here is that the rule modifies the resistance (which isn't accessible on Loihi) and membrane time constant of a neuron to achieve the results . So we use a novel unexplored variation which changes the easily accessible membrane threshold voltage V_{th} given by the equations 6.1 and 6.2.

$$w = \frac{V_{Th} \cdot \tau_m}{\left(\frac{1}{y} - t_r\right)} \tag{6.1}$$

$$\Delta V_{Th} = -lr \cdot \left(\frac{\left(\tau_m \left(2 \cdot y - \frac{y^2}{\mu}\right) - 1\right)}{w} - \frac{1}{V_{th}} \right) \tag{6.2}$$

where V_{Th} is the change in threshold voltage, y is the current trace of the neuron, μ is the expected trace ,τ_m is the membrane time constant and t_r is the refractory period.

6.1.2 Calcium based supervised training Rule

In this rule [43], the weights are adjusted taking into account the upper and lower limits of the output trace which is biologically significant as the calcium concentration of the neuron. The rule is given by the following equation 6.3 and 6.4 where y is the trace (current Calcium concentration), c_m is the expected Calcium concentration and δ defines the range. ΔW is constant change in weight.

$$\text{Potentiating Rule: } dw = \begin{cases} \Delta W, & \text{with prob p if } c_m \leq y \leq c_m + \delta \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

$$\text{Depressing Rule: } dw = \begin{cases} -\Delta W, & \text{with prob p if } c_m \geq y \geq c_m - \delta \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

6.2 Conclusion

- A spiking neural network based on liquid state machine model with input, reservoir and output layers was implemented on the Loihi processor.
- For the readout learning rule, Supervised STDP rule was used which is based on ReSuMe rule. Intrinsic Plasticity was explored in the reservoir layer by the implementation of activity range homeostasis.
- A setup was created with a framework composed of python and SNIP. The python block created and ran the network. The SNIP layer undertook run-time decisions like switching learning rules.
- The network was benchmarked on the single speaker TI-alpha and TI-digits dataset.
- An accuracy of 86.54% was obtained for the TI-alpha subset.

REFERENCES

[1] W. Gerstner and W. Kistler, *Spiking Neuron Models: An Introduction*. New York, NY, USA: Cambridge University Press, 2002.

[2] J. Vreeken, "Spiking neural networks, an introduction," 2003.

[3] F. Ponulak and A. Kasinski, "Introduction to spiking neural networks: Information processing, learning and applications.," *Acta neurobiologiae experimentalis*, vol. 71, no. 4, pp. 409–433, 2011.

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

[5] S. Haykin, "Neural networks: A comprehensive foundation," 2007.

[6] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[7] 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)*, pp. 1–4, IEEE, 2011.

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

[9] C. S. T. Thakur, J. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. M. Wang, E. Chicca, J. Olson Hasler, *et al.*, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in neuroscience*, vol. 12, p. 891, 2018.

[10] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[11] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[12] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, "Isolated word recognition with the liquid state machine: a case study," *Information Processing Letters*, vol. 95, no. 6, pp. 521–528, 2005.

[15] M. Welling, "Fisher linear discriminant analysis," *Department of Computer Science, University of Toronto*, vol. 3, no. 1, 2005.

[16] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[17] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting," *Neural computation*, vol. 22, no. 2, pp. 467–510, 2010.

[18] C.-K. Lin, A. Wild, G. N. Chinya, Y. Cao, M. Davies, D. M. Lavery, and H. Wang, "Programming spiking neural networks on intel loihi," *Computer*, vol. 51, no. 3, pp. 52–61, 2018.

[19] M. R. Bennett, "The early history of the synapse: from plato to sherrington," *Brain research bulletin*, vol. 50, no. 2, pp. 95–118, 1999.

[20] R. H. Cudmore and N. S. Desai, "Intrinsic plasticity," *Scholarpedia*, vol. 3, no. 2, p. 1363, 2008. revision #129344.

[21] G. G. Turrigiano and S. B. Nelson, "Homeostatic plasticity in the developing nervous system," *Nature reviews neuroscience*, vol. 5, no. 2, p. 97, 2004.

[22] W. Zhang and P. Li, "Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks," *Frontiers in neuroscience*, vol. 13, 2019.

[23] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[24] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[25] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[26] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[27] D. O. Hebb, *The organization of behavior: a neuropsychological theory*. Science Editions, 1962.

[28] 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. 10464–10472, 1998.

[29] T. Lømo, "The discovery of long-term potentiation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1432, pp. 617–620, 2003.

[30] V. Jacob, D. J. Brasier, I. Erchova, D. Feldman, and D. E. Shulz, "Spike timing-dependent synaptic depression in the in vivo barrel cortex of the rat," *Journal of Neuroscience*, vol. 27, no. 6, pp. 1271–1284, 2007.

[31] J. Sjoestroem and W. Gerstner, "Spike-timing dependent plasticity," *Scholarpedia*, vol. 5, no. 2, p. 1362, 2010. revision #184913.

[32] B. Widrow and M. E. Hoff, "Adaptive switching circuits," tech. rep., Stanford Univ Ca Stanford Electronics Labs, 1960.

[33] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, pp. 1629–1636, Oct 1990.

[34] T. P. Vogels and L. F. Abbott, "Signal propagation and logic gating in networks of integrate-and-fire neurons," *Journal of neuroscience*, vol. 25, no. 46, pp. 10786–10795, 2005.

[35] M. Liberman, L. D. Consortium., and T. I. Incorporated., "TI 46-word," 1993.

[36] G. R. Doddington and T. B. Schalk, "Computers: Speech recognition: Turning theory to practice: New ics have brought the requisite computer power to speech technology; an evaluation of equipment shows where it stands today," *IEEE spectrum*, vol. 18, no. 9, pp. 26–32, 1981.

[37] P. Wijesinghe, G. Srinivasan, P. Panda, and K. Roy, "Analysis of liquid ensembles for enhancing the performance and accuracy of liquid state machines," *Frontiers in neuroscience*, vol. 13, p. 504, 2019.

[38] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with volterra series," *IEEE Transactions on circuits and systems*, vol. 32, no. 11, pp. 1150–1161, 1985.

[39] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[40] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[41] M. Slaney, *Lyon's cochlear model*, vol. 13. Apple Computer, Advanced Technology Group, 1988.

[42] B. Schrauwen and J. Van Campenhout, "Bsa, a fast and accurate spike train encoding scheme," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 4, pp. 2825–2830, IEEE, 2003.

[43] Y. Zhang, P. Li, Y. Jin, and Y. Choe, "A digital liquid state machine with biologically inspired learning and its application to speech recognition," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 11, pp. 2635–2649, 2015.