Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2019-4734122

M.Sc. Thesis

N-Shot Training Methodology

Ninad Joshi - 4734122

Abstract

Traditional Artificial Neural Networks(ANNs)like CNNs have shown tremendous opportunities in various domains like autonomous cars, disease diagnosis, etc. Proven learning algorithms like backpropagation help ANNs in achieving higher accuracy. But there is a serious challenge with the increasing popularity of traditional ANNs is of energy consumption and computational complexity. Spiking Neural Networks (SNNs) are considered to be next-generation neural networks that are capable of doing complex deep learning applications at fraction of energy that is needed in current deep learning applications because of its similarity to biological neurons. However, SNN is still not able to match the classification accuracy of ANNs which poses a big challenge for wide acceptance of SNN in various applications as traditional learning methods like backpropagation are not possible in SNN. During training of a neural network the weight matrix is of the highest importance as it eventually decides the trajectory of learning. Currently, one existing solution is to just manually convert ANNs into SNNs to get weight matrix which doesn't focus on getting weight matrix from a small dataset and doesn't consider spiking neuron parameters. We aim to address this challenge by proposing a novel N-shot training methodology that is capable of providing a weight matrix for SNN and can give sufficient classification accuracy. The methodology not only provides the weight matrix but can do training with a very small dataset(up to 1 image per class) and still can give considerably higher accuracy. For a reduced MNIST dataset, the method can give an accuracy of 71.68% 10 images per class.

N-Shot Training Methodology
For Spiking Neural Networks(SNNs)

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Embedded Systems

by

Ninad Joshi - 4734122
born in Khandwa, India

Thesis Committee:	Prof. Dr Ir. Rene Van Leuken	TU Delft, CAS
	Prof. Dr. Ir. Zaid Al Ars	TU Delft, CE
	Dr. Ir. Amir Zjajo	TU Delft, CAS
	Dr. Ir. Sumeet Kumar	TU Delft, CAS

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Abstract

Traditional Artificial Neural Networks(ANNs)like CNNs have shown tremendous opportunities in various domains like autonomous cars, disease diagnosis, etc. Proven learning algorithms like backpropagation help ANNs in achieving higher accuracy. But there is a serious challenge with the increasing popularity of traditional ANNs is of energy consumption and computational complexity. Spiking Neural Networks (SNNs) are considered to be next-generation neural networks that are capable of doing complex deep learning applications at fraction of energy that is needed in current deep learning applications because of its similarity to biological neurons. However, SNN is still not able to match the classification accuracy of ANNs which poses a big challenge for wide acceptance of SNN in various applications as traditional learning methods like backpropagation are not possible in SNN. During training of a neural network the weight matrix is of the highest importance as it eventually decides the trajectory of learning. Currently, one existing solution is to just manually convert ANNs into SNNs to get weight matrix which doesn't focus on getting weight matrix from a small dataset and doesn't consider spiking neuron parameters. We aim to address this challenge by proposing a novel N-shot training methodology that is capable of providing a weight matrix for SNN and can give sufficient classification accuracy. The methodology not only provides the weight matrix but can do training with a very small dataset(up to 1 image per class) and still can give considerably higher accuracy. For a reduced MNIST dataset, the method can give an accuracy of 71.68% 10 images per class.

Acknowledgments

I would like to express my sincere gratitude to CAS group for providing me with such challenging yet an interesting graduation project. A special thanks to Dr. Ir. Sumeet Kumar and Dr. Ir. Amir Zjajo for their earnest supervision and for pushing me in the right direction whenever I was stuck in any problem. A special thanks to Ir. Johan Mes for his crucial inputs and expertise in SNNs whenever I faced difficulties in understanding the concepts.

I would like to thanks Prof. Dr. Ir.Rene Van Leuken for his supportive supervision during the project and accepting me in CAS group for my thesis. This thesis would have not been possible without your support and your inputs during the project.

A huge thanks to Prof. Dr. Ir Zaid Al-Ars for agreeing to be a part of my thesis committee.

I would also like to thank my fellow master students Davide, Shashanka, Bas, Rahul, Joppe for accompanying me in this journey. A big thanks to my family back in India, who made sure that I didn't get carried away on the good days and didn't become miserable during the bad days.

Last but not the least, a big shout out to all my friends for their continuous support to make sure that I was constantly giving my best so that I could complete my thesis in a timely manner.

Ninad Joshi - 4734122
Delft, The Netherlands
24th October 2019

Contents

List of Figures

List of Tables

1 Introduction

Artificial Neural Networks(ANNs) as shown in Figure 1.1 are computational systems that are inspired, but not identical, by biological brains. ANNs try to "learn" from examples to do tasks and this learning in ANNs is known as "training" of a neural network.

ANNs have shown tremendous opportunities in various fields especially in domain of pattern recognition & computer vision. Thus there is an enormous possibility of applications in healthcare, autonomous cars, data mining, surveillance etc. There has been widespread acceptance of ANNs in commercial applications as well. But due to this acceptance of ANNs there is even more amount of data that is now available and for complex applications like autonomous cars it gets even more difficult. The hardware that is currently being used for such complex applications are GPUs[4.35W for Alexnet(45)] that are able to handle such increased computational loads but are also energy inefficient consuming most of the energy in autonomous cars. There has been a major concern for the engineers and researchers to find a

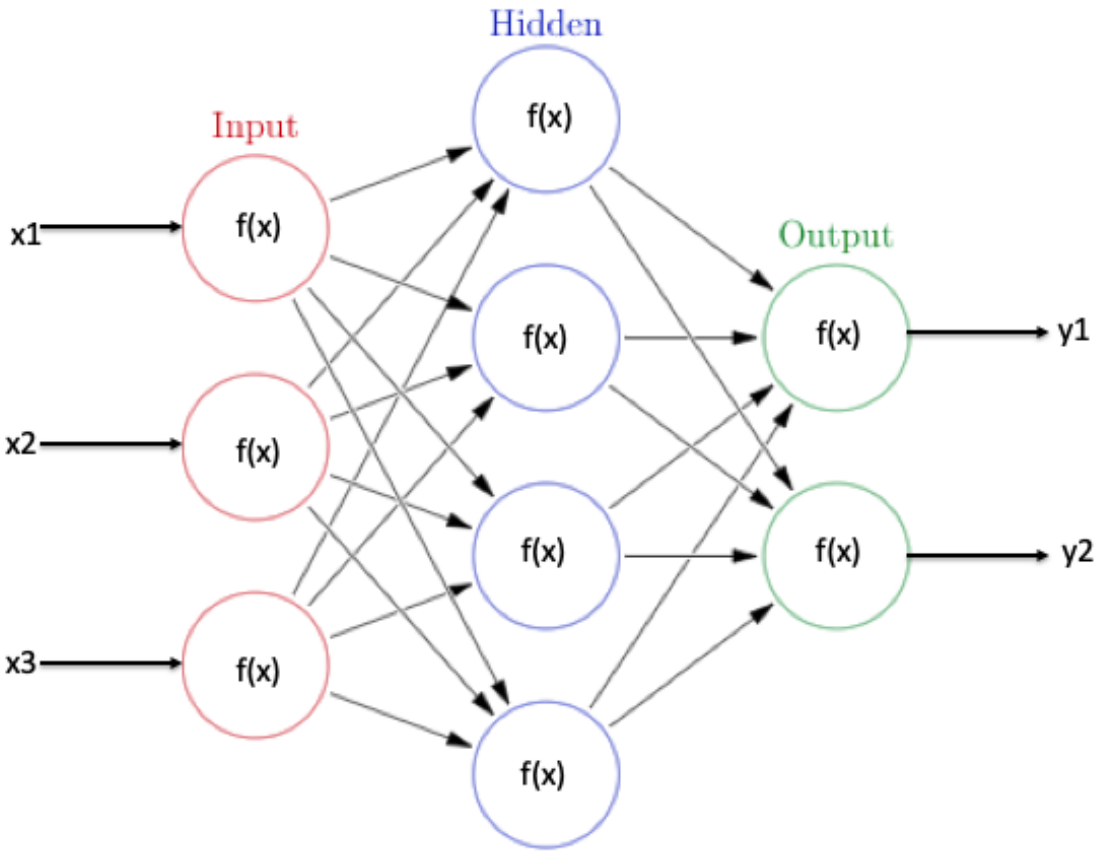

Figure 1.1: An Artificial Neural Network, image taken from(NN)

solution that can do such complex computations with considerably low energy consumption.

1.1 Motivation

Solution to this problem can be found in our brains. Biological brains have shown how complex logical computations can be done in an energy efficient manner. Spiking Neural Networks(SNNs) are special type of ANNs that mimic the biological neural networks. Spiking Neurons are inspired by biological neurons and thus if implemented on hardware, they can be a potential solution to this problem.

SNNs are considered to be as next generation neural networks that are capable of doing complex computations and are extremely energy efficient. However, these biological neuron inspired networks are still in a developmental stage and a lot of research is underway. Learning in SNNs still remains a very big challenge, the inability of spiking neurons to use backpropagation restricts SNNs to achieve higher accuracy. In order to have widespread acceptance of SNNs commercially, a considerable amount of classification accuracy needs to be achieved by SNNs.

1.2 Problem Description

The "training" of a neural network means change in the weight values Figure 1.2 of the synapses/connections that connect the neurons between two layers, thus the values that these synaptic weights eventually attain is responsible how good the network can recognise given input.

Thus, regardless of the type of learning methods used the weight of networks are of highest importance. The values of this weight matrix eventually affects how good the learning is and how well the network can do the classification. In traditional ANNs it is made possible by using various methods like backpropogation which is most popular method, however in SNNs backpropagation can not be easily used due to its inability to do differentiation which makes it difficult to get better classification accuracy.

This becomes even more challenging when there is not enough data to train the network.

1.3 Objective

- Get the weight matrix for the SNNFigure 1.3.
- Should be able to work with as less as possible samples in the dataset.
- Using the weight matrix the SNN should be able to perform classification with sufficient accuracy.

1.4 Contributions

The contributions made by this work are as follows:

- Proposal of a state-of-art training methodology with lesser data samples from each class : N-shot training.

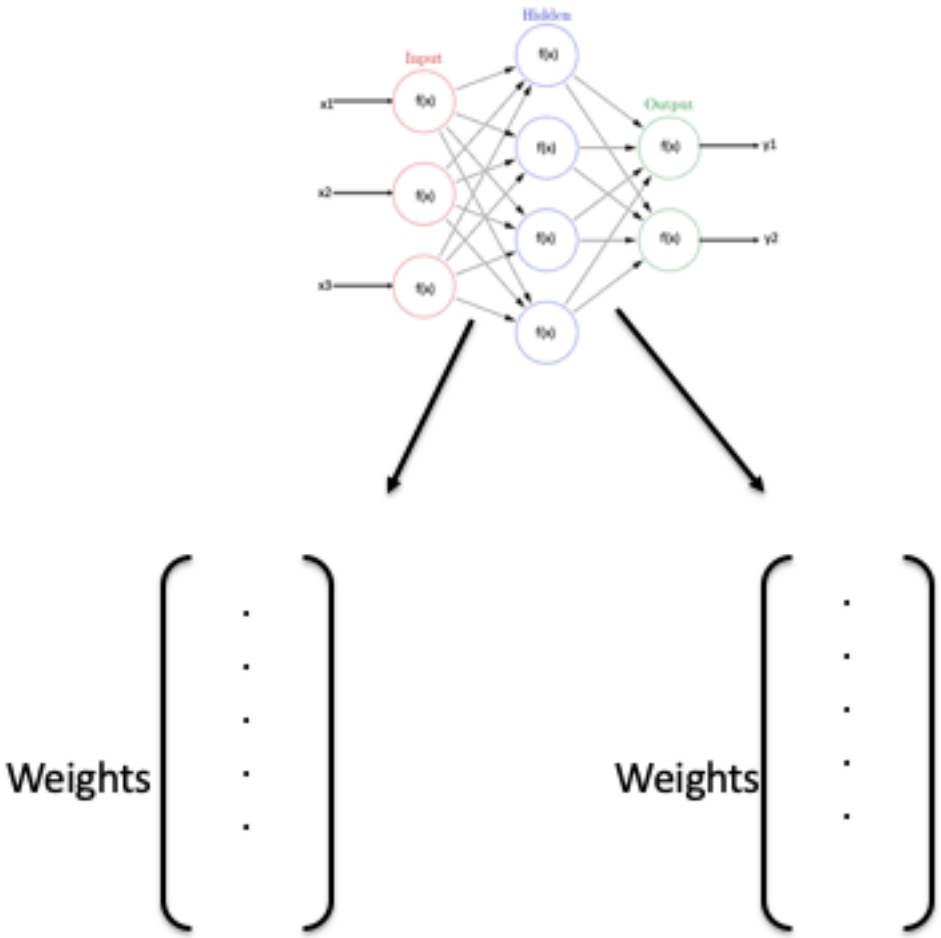

Figure 1.2: Weights of a neural network

- Methodology incorporates a way to get weight matrix that gives sufficient accuracy and in much lesser time.

- The proposed methodology is flexible enough to train network with other datasets and can give estimated accuracy before using weight matrix into SNN.

- Methodology can do training irrespective of learning method(supervised or unsupervised) used by SNN.

- Methodology is applicable to both labelled and unlabelled data.

- A unique distance controlled CD algorithm, where distance works as a Hyper Parameter and strongly controls the learning.

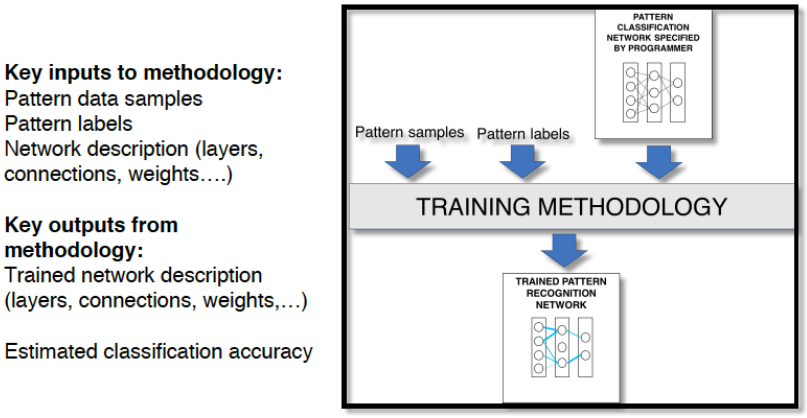

Figure 1.3: Objective

1.5 Outline

This thesis is broadly divided into three parts : Part I - Literature Review which is comprised of chapter 2 and chapter 3 , Part II - Proposed Methodology is comprised of chapter 5 and chapter 4 and at last Part III - Results and Conclusion is comprised of chapter 6 and chapter 7.

Part I:

chapter 2 discusses about Spiking Neural Network , various Spiking neuron models, learning methods, simulators.

chapter 3 discusses about Deep Learning and Restricted Boltzmann Machines, role of RBM in the context of this thesis, working of RBM, training of RBM, relation between RBM and SNN.

Part II: chapter 4 discusses about the proposed Generalised training methodology flow, working of a distance calculator and DBN based N-shot training methodology.

chapter 5 discusses about the proposed modified CD based learning method, why the modification of CD was required and also discusses about the proposed algorithm.

Part III:

chapter 6 provides various results and findings about proposed methodology.

chapter 7 deals with the conclusion, future work and contributions made by this work.

Part I
Literature Review

2 Spiking Neural Networks

As discussed in chapter 1 Spiking Neural Networks (SNNs) are biologically inspired ANNs(Artificial Neural Networks) where information is represented in the form of spikes. The SNN learning rules are operated by the relative timing of spikes between two neurons connected to each other(38).

SNNs are considered to be as next generation neural networks that are capable of doing complex deep learning applications at fraction of energy that is required in current deep learning applications because of its similarity to biological neurons. However, this field is still in its nascent stage and SNN are still not able to match classification accuracy of ANNs which poses as a big hindrance for wide acceptance of SNN in various applications. Hence, training of SNN poses to be a big challenge.

In this chapter we will develop background of SNNs that makes them different from conventional ANNs. In section 2.1 we will understand about biological neurons, section 2.2 deals with various spiking neurons models, section 2.3 deals with learning is SNNs and finally section 2.5 discusses about existing simulators that can be used to simulate various SNN based models.

2.1 Biological Neural components

The Human Brain consists of billions of neurons and up to the order of 10^4 more synaptic connections(6). Even after being such a complicated system the neurons can transmit signals to other cells quite effectively.

2.1.1 Neurons

As it can be seen form Figure 2.1 there are three components of a neuron i.e Dendrites, Cell body and Axon.

The signal is delivered from one neuron to another via a junction called Synapse Figure 2.1(b). The whole setup can be seen as consisting of Pre-Synaptic Neuron(sender), Post-Synaptic Neuron(receiver) and a synapse.

2.1.2 Neuronal Signals

The neuron signals travel from one neuron to another in the form of spikes.Figure 2.1(c) shows the representation of spike in a biological neuron. This spike is of 100mV amplitude and can last for 1-2ms. Once the spike is fired from the neuron, the neuron stops responding to any incoming spike for a specific duration of time and this time period is known as " Refractory Period ".

The timing and frequency of the spike that carries the information. A sequence of spike caused by the neuron is known as "spike train". This spike train is binary in nature where spike events are 'on' and no activity is 'off'.

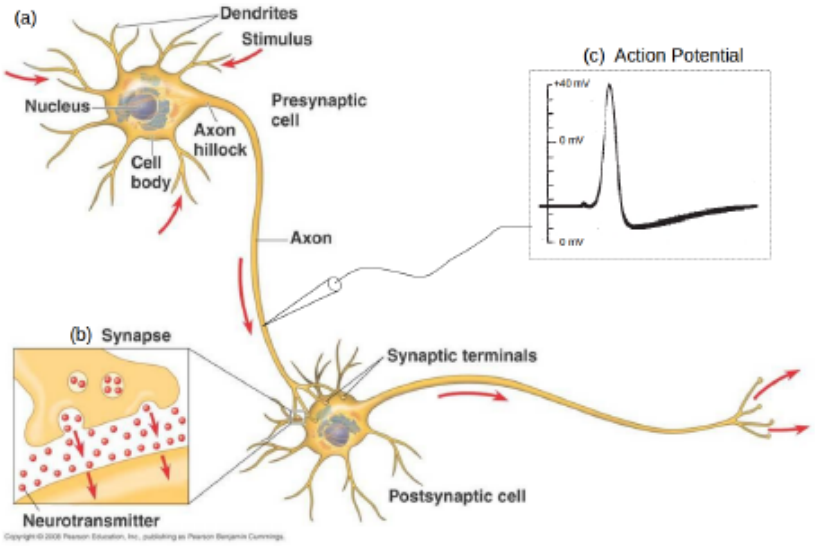

Figure 2.1: Two neurons connected by synapses. A neuron comprises three functional parts: dendrites, the cell body, and the axon.(a) A pre-synaptic cell connects to its post-synaptic cell through synapses (b) (11) , and the neural signal, the action potential(c)(25) propagates in the direction of the red arrows.

When the spiking rate is representing the intensity of the stimulus then such model is called as "Rate Coding Model". As from Figure 2.2 it can be seen that when the stimulus gets stronger the spike frequency also increases. This encoding is most suited for the situations when the input is changing over the long period of observation to estimate the firing rate which is very short in practice. Thus, "Temporal Encoding" i.e encoding that encodes information in the precise timing of spikes is used for fast changing input Figure 2.3 shows the phase locked spike trains.

Time - to - first scheme encodes the information according to the intensity of the input where a spike occurring as soon as the reference signal indicates a strong input whereas spike with a delay indicates a weaker input. There are various other encoding schemes like population encoding, Rank order encoding etc.

2.2 Modelling Spiking Neurons

2.2.1 Neural Dynamics

"Membrane Potential" in the biological neuron is the difference between interior and exterior of the cell body. it stays at "resting potential" when there is no spike presented to the neuron. When spike is presented to the neuron the membrane potential increases or decreases. The change in membrane potential of post-synaptic neuron is termed as "Post-synaptic Potential(PSP)". Spike triggered by excitatory synapse is positive PSP and the one triggered by inhibitory synapse is negative PSP. When multiple PSP aggregate until membrane potential reaches the threshold to generate a spike as output Figure 2.5.

There are two types of synapses :

- Excitatory : Increases neuron's membrane potential

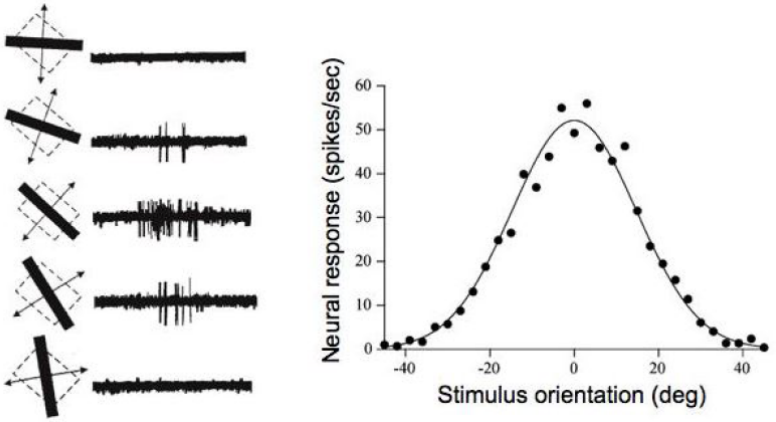

Figure 2.2: Example of rate coding: spike trains for different stimulus orientation (left) of a V1 simple cell of a cat, and the tuning curve (firing rate against stimulus orientation) of the neuron (right)(26). The square indicates the visual receptive field of the neuron, and a bar is placed at different orientations and moves to the direction perpendicular to its orientation. As the stimulus becomes more aligned to the preferred orientation (0°) of the neuron, the firing rate increases.

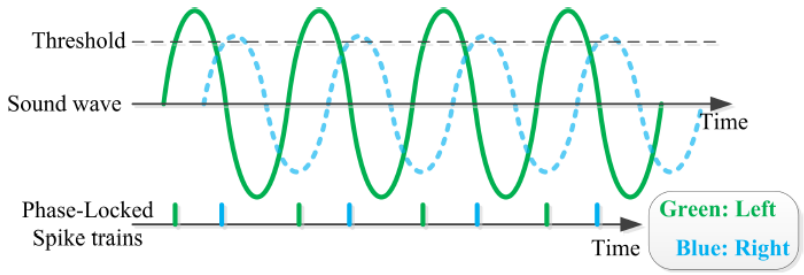

Figure 2.3: Example of temporal coding: phase-locked spike trains generated by simulated Inner Hair Cells in the cochlea (37). A sound source generates a sine wave of a certain frequency and conducts to two ears with a time difference and different amplitude due to the angle and distance of the sound source to the head. Two spike trains respond to different phases manipulated by a threshold of the sound waves. Sound localisation can be resolved by calculating the time difference and/or level difference of these sound waves which are encoded in the spike trains.

- Inhibitory : Decreases neuron's membrane potential

Thus, weights between two layers of a network can be updated as a result of learning method applied to the network.

2.2.2 Spiking Neuron Models

There are several spiking neurons that have been developed, we will be discussing them in this section in brief. LIF neuron model was chosen a spiking neuron model for this thesis because it captures the intuitive properties of biological neuron.

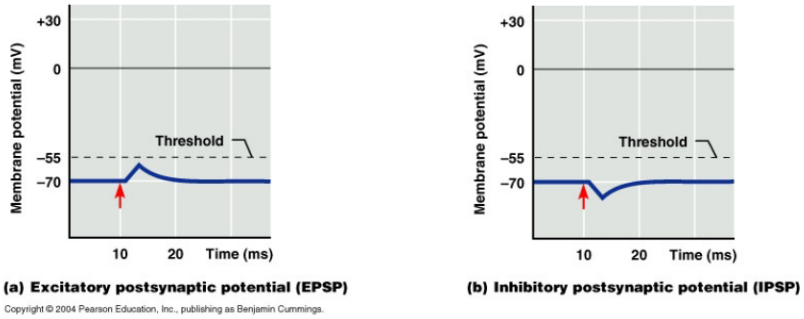

Figure 2.4: Post-synaptic potential driven by a spike, where the red arrow represents a spike arriving at the neuron (39)

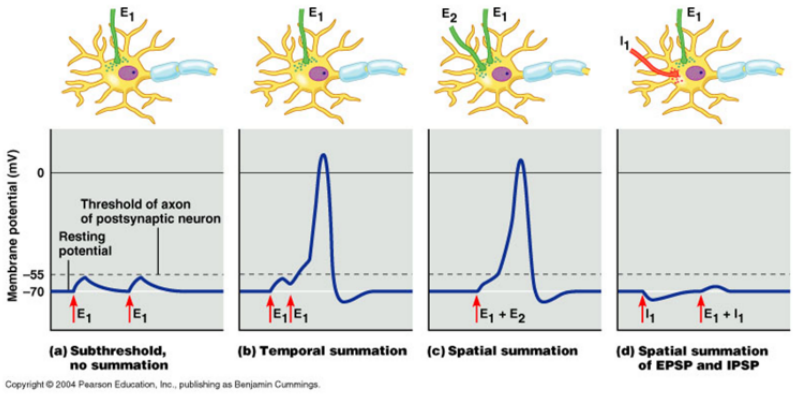

Figure 2.5: Summation of post-synaptic potentials [Reece et al., 2011]. (a) Single positive PSP are usually not strong enough to trigger an action potential without summation. (b) Temporal summation of two positive PSP of the same synapse generates an action potential. (c) Spatial summation of two positive PSP of two synapses generates an action potential. (d) Spatio-temporal summation of both positive PSP and negative PSP.

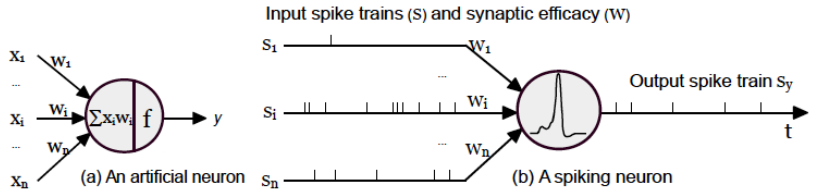

Figure 2.6: Comparisons of processing mechanisms of an artificial and a spiking neuron (36). (a) An artificial neuron takes numerical values of vector x as input, works as a weighted summation followed by an activation function f. (b) Spike trains flow into a spiking neuron as input stimuli, trigger linearly summed PSPs through synapses with different synaptic efficacy w, and the post-synaptic neuron generates output spikes when the membrane potential reaches some threshold.

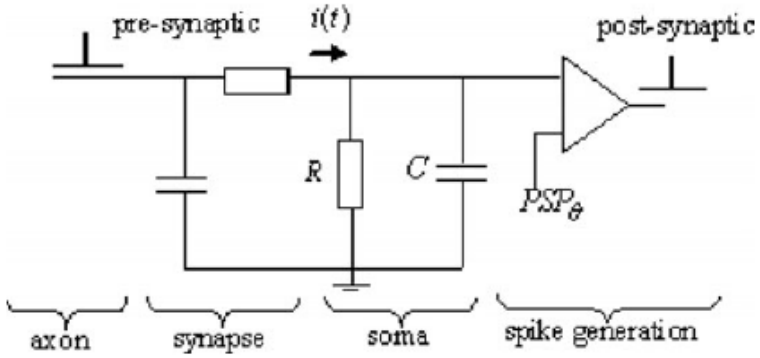

Figure 2.7: An electrical circuit representing the LIF spiking neuron model (17)

2.2.2.1 Leaky Integrate and Fire Model(LIF model)

LIF model is based on leaky integrator which fires the spike if voltage of the neuron reaches a threshold and resets itself to a resting state. LIF can also be visualised as a RC circuit (Lapicque). I(t) is current, u(t) is membrane potential and $\tau_m = \text{RC}$ is the neuron's membrane time constant.

Follwoing equations are taken from (30) that explain its mathematical significance :

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \tag{2.1}$$

the firing time $t^{(f)}$ is defined by a threshold value v_{thr} .

$$t^{(f)} : u(t^{(f)}) = v_{thr}$$
$$\lim_{t \to t(f); t > t(f)} u(t) = u_r$$

LIF model is viewed as the best-known instance of spiking neuron model because of its simplicity and low computational cost.

There are many more neuron models other than LIF, Izekivich and HHM model. As it can be seen from Figure 2.9 that HHM model is most biologically plausible neuron model but has very high implementation cost, while LIF neuron model is most efficient in terms of computation costs making it most suitable for hardware implementation but it has least biological plausibility among all available neuron models.

SNNs are more biologically realistic than other ANNs like CNN, also are more energy efficient and hardware friendly. However, training of SNNs is a challenge and has lesser accuracy.

The non-differentiability of spike train restricts the use of popular backpropagation theorem, which is popularly used in AI applications for finding derivative(49).

2.3 Learning in SNN

In almost all ANNs (spiking or non-spiking) learning is achieved by adjusting synaptic weights. Learning in SNN can be classified into multiple categories, but here in this thesis we will just

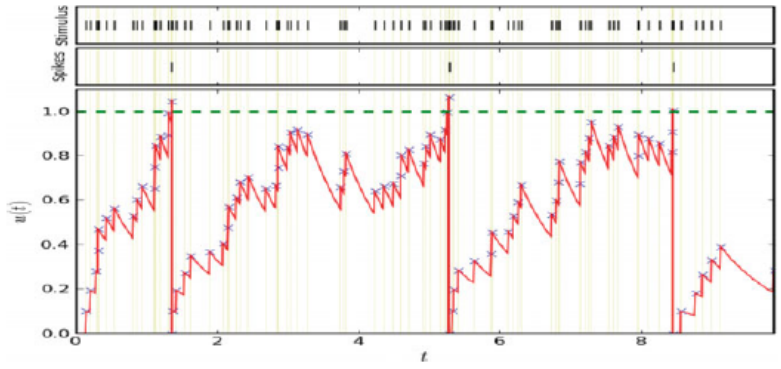

Figure 2.8: The membrane potential of a LIF neuron accumulates input spikes as stimuli. When the potential reaches a threshold, the neuron emits an output spike. (17)

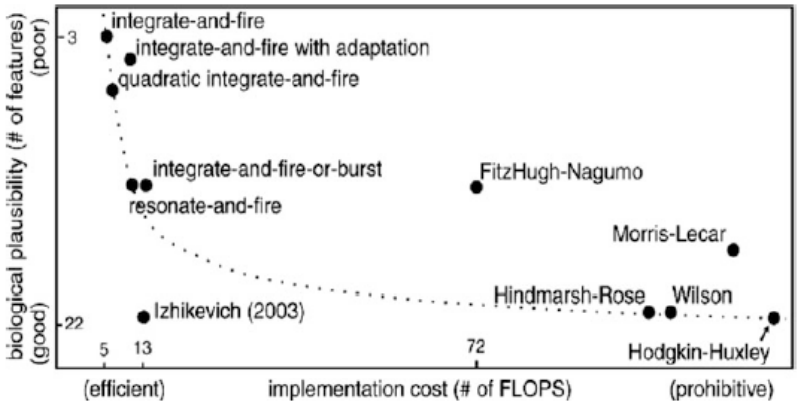

Figure 2.9: Comparison of spiking neuron models in the evaluation landscape of biological plausibility and implementation cost (27)

focus on :

- Unsupervised Learning
- Supervised Learning

2.3.1 Unsupervised Learning via STDP

SNN has a special kind of learning rule i.e STDP(Spike Time Dependent Plasticity). Key feature of STDP is weight connecting a pre and post synaptic neuron is adjusted according to their relative spike times with an interval of tens of milliseconds(ms) in length. Information used for weight adjustment is local i.e to synapse and local in time.

If pre-synaptic neuron fires before post-synaptic neuron the weight is increased or weight is strengthened, conversely if post-synaptic neuron fires before pre-synaptic neuron, then the weight is decreased or weight is weakened. The strengthening of weight is called as Long Term Potentiation(LTP) and weakening of weight is called Long Term Depression(LTD).

Following equations are taken from (49) :

$$\delta w = \begin{cases} A * e^{-\frac{|t_{pre}-t_{post}|}{\tau}} & \text{if } t_{pre} - t_{post} \leq 0 \text{ and } A > 0 \\ B * e^{-\frac{|t_{pre}-t_{post}|}{\tau}} & \text{if } t_{pre} - t_{post} > 0 \text{ and } B < 0 \end{cases} \tag{2.2}$$

A and B are constant parameters indicating learning rates, τ is time window for learning. It has also been shown that SNN with STDP shapes neuronal selectivity to the stimulus pattern within SNN i.e response latency of post-syanptic neuron decreases as STDP proceeds, thus neuron responds to a specific input faster than any other.

2.3.2 Supervised Learning

All supervised learning use labels of some kind, it adjusts weights via gradient descent on a cost function comparing observed and desired network operations.

In SNN, supervised learning tries to minimize error between desired and output spike trains.

Before discussing about various supervised SNN learning methods, we would be understanding various issues in SNN with Backpropagation which has been quite clearly explained in (49) :

- In the chain rule for backpropagation(49) i.e $\delta_j^\mu = g\ '(a_j^\mu)\Sigma w_{kj}\delta_k^u\ g\ '(\cdot)$ requires $g(\cdot)$ which is already applied to spiking neuron, it is likely represented by a sum of dirac-delta funtion, which means derivative doesn't exist.
- Weight transport problem(49) i.e $\Sigma w_{kj}\delta_k^\mu$ is using feed forward weights w_{kj}in a feedback fashion. This means that matching symmetric feedback weights must exist and project accurately to the correct neurons, this issue has made some progress and simple problems can be resolved by any feedback while for complex problems it requires symmetric feedback.

2.3.2.1 SpikeProp

It is the first algorithm(9) for supervised learning in SNN by backpropogation errors. It's cost function takes into account spike timing and spike properties to classify non linearly separable data for a temporally encoded XOR problem using a 3-layer architectureFigure 2.10.

SpikeProp has following Limitations :

- Has only 1 neuron input from presynaptic neuron.
- each output unit has been constrained to discharge exaclty one spike i.e it is encoded as spike delays which takes lots of time.

Multi-spikeprop(10) covers the limitations of Spikeprop but uses same neural architecture of spikeprop. It has multiple input from presynaptic neurons.

2.3.2.2 Tempotron

In this method (21) only one neuron has ability to recognize encodings by the precise timing of incoming spikes. The drawbacks of tempotron is that it gives output in terms of 0 or 1 during an interval, due to this the spike timing information is not encoded.

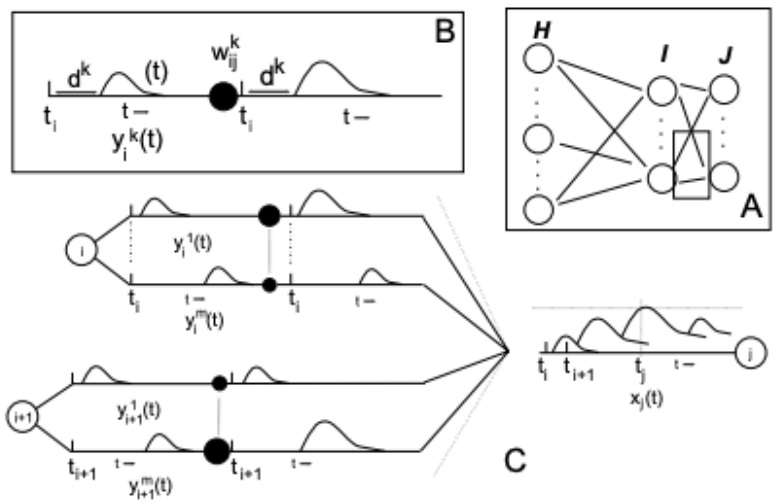

Figure 2.10: Illustration of SpikeProp (A)Multilayer feedforward network (B)a single synaptic connection (C)Two multi synapse connections (9)

2.3.2.3 ReSuMe(Remote Supervised Learning)

This method(44) uses correlation between teacher/desired neuron output and the input neuron but there is no direct physical connection, hence it is called as 'remote'. This method uses a Widrow-Hoff Delta Rule i.e weight change is directly proportional to the difference between desired output and observed output. Mathematically, it can be expressed as :$\delta w = (y^d - y_0) * x = y^d x - y^0 x$ where, y^d is desired output, y^0 is observed output and x is presynaptic input.

Thus, for a SNN equation can be written as :

$\delta w = \delta w^{(STDP)}(S^{in}, s^d + \delta w^{aSTDP}(S^{in}, S^o))$ where δw^{STDP} is the correlation function of presynaptic and desired spike trains, δw^{aSTDP} is the correlation function of presynaptic and observed spike trains.

Only drawback of this method that it is limited within certain STDP window.

2.3.2.4 Chronotron

Motivation of this approach(16) is similar to SpikeProp and its successors . This supervised learning approach is based on a more sophisticated distance measure i.e Victor Purapura (VP) distance metric.

The VP distance is the minimum cost of transforming one spike train into the other by creating,removing or moving spikes. VP was adapted because it can be piecewise differentiated and can perform gradient descent with respect to weights.

2.3.2.5 Spike Pattern Association Neuron(SPAN)

This method(40) is also based on Widrow-Hoff Algorithmm, although its similar to ReSuMe but unlike ReSuMe instead of adapting algorithm to SNN, SPAN modifies SNN for the algorithm.

It uses digital to analog conversion of spike trains using alpha kernels of form $t * e^{\frac{-t}{\tau}}$ thus, $\delta w \propto \int x'_i(y'_d(t) - y'_o(t)) * dt$

where x' and y' indicates analog version of spike train and bounds of integration cover local time interval.

Table 2.1 summarises all supervised learning approaches used in SNN with their key characteristics.

Algorithm	Characteristics
SpikeProp(9)	* Based on the error back-propagation learning algorithm for MLP * Classification of static data, information is encoded in few spikes * Multi-layer feedforward architecture (input, hidden and output layers) * Limited to a single spike per neuron
Tempotron(21)	* Minimizes an error based on the membrane potential * Classification of spatio-temporal spike patterns using single neuron * Spatio-temporal spike pattern classification * Binary output * Batch learning
ReSuMe(44)	* Based on biological interpretation of WidrowHoff rule * STDP and learning window * Mainly for precise time spike sequence generation * Spatio-temporal input and output spike patterns * The learning is local in time and space
Chronotron(16)	* Minimizing an error function based on the VP distance metric * For spike sequence generation and classification * Spatio-temporal input and output spike patterns * Batch learning (E-learning), online learning (I-learning)
SPAN(40)	* Based on WidrowHoff rule and kernel function convolution * For spike sequence generation and classification * Spatio-temporal input and output spike patterns * Batch learning or increamental learning

Table 2.1: Summary of Supervised Learning methods in SNN

2.4 Advantages of SNN

SNN has following advantages over conventional ANNs:

- Efficient modelling of time dependent data(30).
- Efficiently involves different time scales(30).
- Enables Fast and parallel information processing(30).
- Low energy consumption for neuromorphic systems(30).
- Deep learning in SNN is inspired by brain(30).

2.5 SNN Simulators

There are various SNN simulators as shown in available that can be used to simulate various SNN based models :

- NEST (18) , BRIAN(48) , ANNarchy(52) focus on biological simulation, components and reactions.
- NEURON(12) and Genesis(13) are more biological focused simulators usually used for simulating neural science problems.
- NeuCube(29) and Nengo(7) focus on high level user behaviour and Machine learning computations, NeuCube(29) is rate coding based, it maps data in three dimensional SNN models, its not an open source project thus limits its usage for broader purposes. Whereas Nengo(7) is an open source platform based on python and tensorflow(5). It simulates Neural Engineering framework(47) rather than Machine learning framework.
- CARLsim(8) and NeMo(15) focus on high level SNN and thus good for SNN based machine learning implementations however they only use Izhikevich(27)neurons hence can't be used for any other neuron models.
- BRIAN1(19) is the most popular spiking neuron simulator with most of SNN simulations done with this simulator. It can do high level implementation for SNN and has core functionality based on C++ and can be programmed in python. Can be modified and extended very easily.
- Although most work is in BRIAN1(19) and it is most popular but it has been discontinued and replaced by BRIAN2(20), there is no provision to convert BRIAN1(19) to BRIAN2(20) code yet thus making it really difficult to build up on any previous work. BRIAN2(20) is drastically different from BRIAN1 due various new functionalities and features. There is not much work done on BRIAN2(20) yet.
- BindsNet(22) is a very new simulator that can be used on multiple hardware platforms like : ASIC, FPGA, DSP or ARM based platforms, it is still not that popular yet as its still in development phase. BindsNet(22) relies heavily on Pytorch(43) and is good for simulations at high-level behaviour.

- In-house Simulator is a SNN hardware implementation focused simulator that considers R and C value of every neuron, it bridges the gap between SNN modeling and SNN hardware implementation. In this thesis we will be using In-house Simulator for classification using SNN architecture.

2.6 Conclusion

In this thesis we will be using In-house Simulator for classification using Spiking Neural Networks. In-house Simulator as discussed above helps in realistic implementation of the neuromorphic hardware. Also we are going to use LIF neuron model(has less computational complexity and easier to implement on hardware) and STDP(unsupervised) for learning.

There were supervised approaches like ReSuMe and Chronotron were explored for getting the weight matrix but due to complexity in their implementation for architectures with deeper layers, a more traditional approach was used that is discussed in chapter 4.

Deep Learning and Restricted Boltzmann Machines

3

In previous chapter 2 we discussed main features of Spiking Neural Networks work. In order to get a weight matrix we either need an identical spiking neural network or a behavioral equivalent network that can behave similar to the SNN because existing SNN training methodologies are limited only for single layer networks (40)(16)(44) and thus can not be extended to the deeper architectures.

Restricted Boltzmann Machines (RBMs) are one of possible alternatives that can help us in making an equivalent SNN that can be modified to have neurons that have equivalent transfer function of a LIF spiking neuron.

Thus, in this section we will understand the working of RBMs and their training which can then be used to get initial weight matrix.

3.1 Background of Boltzmann Machine

The recent rise in deep learning can be attributed to the progress made on Boltzmann Machines(BM) (24) since 1986. Boltzmann Machines are composed of two layers and connection between these layers is bidirectional thus allowing these layers either to have an external state or generate data from an internal representation. BM encodes the input into its probabilities of input data on which they are trained and modeled on a physical analogy of energy configurations. BM are capable of reproducing the data on which they have been trained hence they are also known as "generative models". For MNIST when a network is trained the BM will produce digits like patterns on the visible part of the system after the training.

When RBMs were first introduced there was not much acceptance as it was not effective enough in supervised training of smaller network architectures with backpropagation. The over-learning of the top layer and under-learning of lower layer also resulted in improper

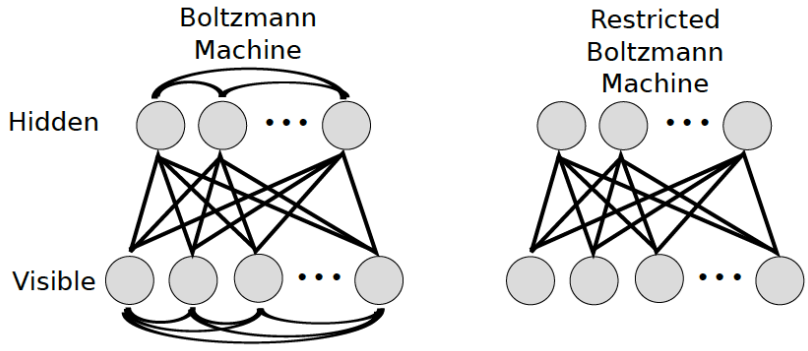

Figure 3.1: Figure illustrating difference between Boltzmann Machine and Restricted Boltzmann Machine(42)

usage of network resources leading to performance worse than a single well trained layer and used to take very long time to train. However, things changed when Hinton's work was published where he introduced a new kind of learning method for RBMs called Contrastive Divergence(23) which gave rise to deep networks.

3.2 Previous work in Spiking DBN

(34) implemented sparse DBNs by using an interleaving contrastive divergence with gradient descent on a sparsity term. An extension of this lead to convolutional sparse DBNs, it used Gibbs sampling to realise proper weight update rules. (41) used a stochastic integrate and fire neuron instead of memoryless stochastic units and also showed that a variant of STDP can approximate Contrastive Divergence.

Another approach is to convert already trained DBNs to spiking ones. The spiking DBN and RBMs are power efficient also enable implementation of low latency hardware with higher accuracy.

3.3 Training a RBM

In binary RBMs units have two states i.e either '0' or '1', considering v_i as state of visible units, h_j as state of hidden units, w_{ij} as weights connecting these units, , the joint encoded probability of an RBM via an energy function as :

$$E(v, h; \theta) = -\Sigma_i \Sigma_j w_{ij} - \Sigma_i b_i^v v_i - \Sigma_j b_j^h v_j \tag{3.1}$$

where $\theta = (w, b^{(v)}, b^{(h)})$. The joint encoded probability then can be given as :

$$p(v, h|\theta) = \frac{exp(-E(v, h; \theta))}{\Sigma_{v\,'} \Sigma_{h\,'} exp(-E(v\,', h\,'; \theta))} \tag{3.2}$$

The training is done in the following way Figure 3.2 :

- Visible units are sampled to given input image.
- The task of learning is to adapt θ such that the distribution becomes similar to observed data, which shows that weights can be approximated by the Gibbs sampling procedure which alternates between hidden and visible layer respectively.
- The RBM learning rule can given as $\delta w_{ij} = \eta(((v_i h_j)_{data}) - ((v_i h_j)_{model})))$, where $(v_i h_j)_{data}$)denotes average over samples with visible units clamped to actual input, $(v_i h_j)_{model}$denotes average over samples when the network is allowed to sample all units freely.
- For a RBM, the Contrastive Divergence(CD) algorithm uses only a single sample for the data and model distribution.
- CD first samples new values for all hidden units in parallel conditioned on the current input which gives (v_{data}, h_{data})
- It then generates the sample for the visible layer conditioned on h_{data} sampled in previous step.

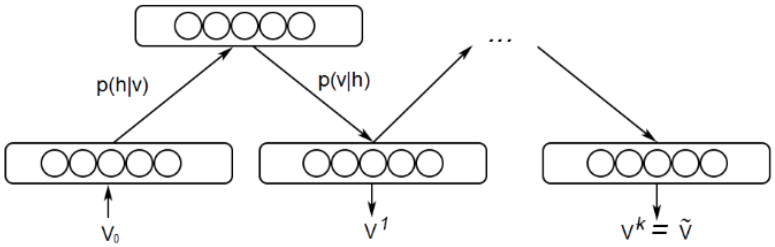

Figure 3.2: Figure illustrating the gibbs sampling if an RBM(gib)

- Hidden layer is sampled again conditioned on this new activity in visible layer, this generates a sample (v_{model}, h_{model}) from model distribution.
- The weight update can be given as : $\delta w_{ij} = \eta(v_{i,data}h_{j,data} - v_{i,model}h_{j,model})$

As CD approach is biased for the recently seen sample, there is an alternate approach called Persistent Contrastive Divergence (50) is used where where the model distribution in randomly initialized. At every iteration of training phase the new samples are created by sampling done on the most recent sampled hidden layer neurons.

In persistent CD the sampling and learning are related on the learning rate, if its too high then the joint probability distribution also changes very fast and states doesn't get to settle in equilibrium. However, higher learning rates have proven to beneficial. According to (51) "fast weights" are added to the network and then decay at each training step. Fast Weights are updated using the following rule : $\delta_{ij}^{fast} = -\alpha(v_i h_j)_{model}$

Figure 3.2 shows steps for the Gibbs sampling, however in practice only 1 Gibbs step is used.

3.4 Constructing Deep Belief Network

By interpreting hidden layer of lower RBM as visible for the next one, it has been shown that by increasing number of hidden layers and applying unsupervised learning for RBM, hidden layers tend to encode more abstract features which are very informative for classification tasks.

Top layer of DBN is trained with supervised methods and whole multi-layer network can be optimised for task through error backpropagation.

3.5 Relation between SNN and SRBM

(42) showed a way to construct a DBN for spiking neurons. A RBM trained on siegert neuron units can be easily converted into equivalent spiking LIF neurons :

- By normalizing the average firing rate relative to maximum firing rate($\frac{1}{t_{ref}}$)
- ρ_{out}(average firing rate) can be converted into activation probabilities as required to sample RBM units during "CD" learning.

- After learning, parameters and weights are retained but instead of sampling at every time step the units generate Poisson spike trains with rates computed by Siegert Formula Equation 3.3.

(31) shows how a siegert neuron can be converted into an equivalent LIF neuron, where k = $\frac{1}{\tau_m} = \frac{1}{RC}$; (ρ_e, ρ_i) firing rates of excitatory and inhibitory inputs; weights(w_e, w_i); $\mu_Q = \tau\Sigma(w_e\rho_e + w_i\rho_i)$; $\sigma_Q^2 = \frac{\tau}{2}\Sigma(w_e^2\rho_e + w_i^2\rho_i)$; $\perp = v_{rest} + \mu_Q$; $\Gamma = \sigma_Q$

$$\rho_{out} = (t_{ref} + \frac{\tau}{\Gamma}\sqrt{\frac{\pi}{2}} \cdot \int_{V_{reset}+k\gamma\Gamma}^{V_{th}+k\gamma\Gamma} exp(\frac{(u-\perp)^2}{2\Gamma^2}) \cdot (1 + erf(\frac{u-\perp}{\Gamma\sqrt{2}}))du)^{-1} \tag{3.3}$$

3.6 Conclusion

The Spiking DBN based pre-training plays a very crucial role in the pre-training methodology that will be presented in this thesis. Spiking DBN is chosen for the following reasons :

- RBMs have greedy, layerwise and unsupervised training
- With each new hidden layer, weights are adjusted until that layer is able to approximate the input from previous layer.
- It requires no labels to improve weights of network.
- After this we can use required SNN to classify images.

The key advantage of this approach is the proper initialisation of weights to facilitate later learning and classification.

Part II

Proposed Methodology

Generalised Training Methodolgy for Spiking Neural Networks

4

As mentioned already mentioned in the Introduction section the objectives of the proposed generalised methodology can be broken into following sub-goals that we are trying to achieve in this thesis :

- Get weight matrix for the SNN.
- Should be able to work with as less as possible samples in the dataset.
- Using the weight matrix the SNN should be able to perform classification with sufficient accuracy.

In chapter 2 and chapter 3 we saw how we can achieve the first two objective of getting the matrix for SNN with a small dataset. Using RBM showed a way to get the weight matrix. Now in this chapter we will discuss about the whole methodology flow to get considerably higher SNN classification accuracy.

Here we propose a flexible approach that can be trained in a greedy, unsupervised, layer-wise fashion. As mentioned in chapter 5 RBMs are very effective in feature extraction/classification of various features for networks with multiple layers and have successfully shown for other ANNs. (42) showed a way of using these RBMs for SNN by using Siegert Neuron which has equivalent transfer function to that of a spiking LIF neuron.

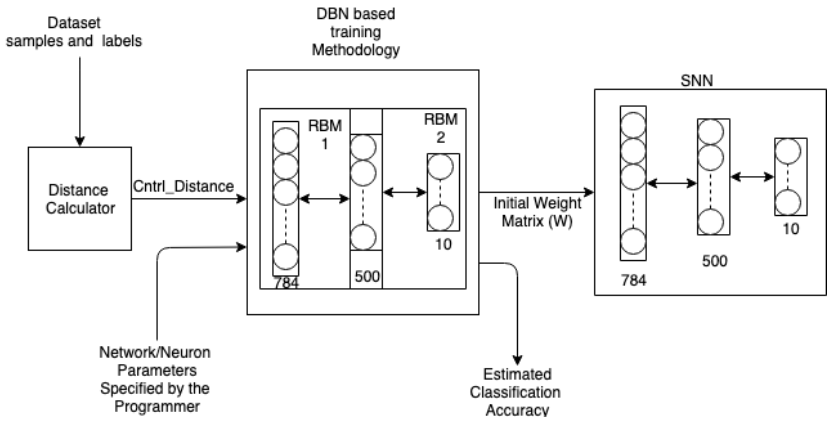

Figure 4.1: Proposed Methodology Pipeline showing 784 input,500 hidden and 10 output layer architecture

In this chapter and subsequent sections we will be discussing about this methodology and each block in much more detail and will show how it achieves objectives of this thesis.

4.1 Methodology Flow

In this section we will be discussing working flow Figure 4.2 of the proposed methodology.

The flow illustrates working of the methodology :

- The training data and labels are given to distance calculator.
- Distance calculator calculates the control distance.
- The control distance(Cntrl_dist) acts as a Hyper Parameter along with other hyper parameters.
- The user specifies the required, control distance network configuration along with the learning and neuron parameters.
- If weights for only a specific layer are required then also a network of atleast 3 layer configuration is required because input layer and output layer acts as default.
- Once the network configuration, neuron and learning parameters are specified the DBN based CD unsupervised learning is done that also acts as a feature extraction.
- Now, at this stage there are two options for the user. If user wants to know the estimated accuracy of the learning then there is an option of passing test dataset through the model. Else, if user just wants the weight matrix that can be ported to SNN based model which can be then used for classification purposes.
- Once the weight matrix is ported to the In-house simulator for SNN classification the weight matrices are partitioned into their corresponding layer, thus providing an option to use weight matrix for a particular layer.
- Once all parameter of SNN based simulator are set then classification is performed and accuracy is reported in In-house Simulator as the output.

4.2 Distance Calculator

For any new dataset this calculator will calculate mean distance between data samples of each class. The distance calculated by this can be used as a "Control Knob" to control the training outputs.

This calculator is to be used whenever we have a new dataset. It is used once to assist the programmer to estimate distance between various classes and then use the same distance in training methodology to influence the learning process.

The notion of distance shows the degree of similarity or dissimilarity between two data samples. If distance value is small then images are similar whereas if distance in larger then images are not similar. We will be discussing later in this chapter on how we are using distance to compare images belonging to same class and how it influences the training.

- Give the dataset as input.
- Use dimension reduction for the dataset to reduce dimensions from 784 to 2 using t-SNE.

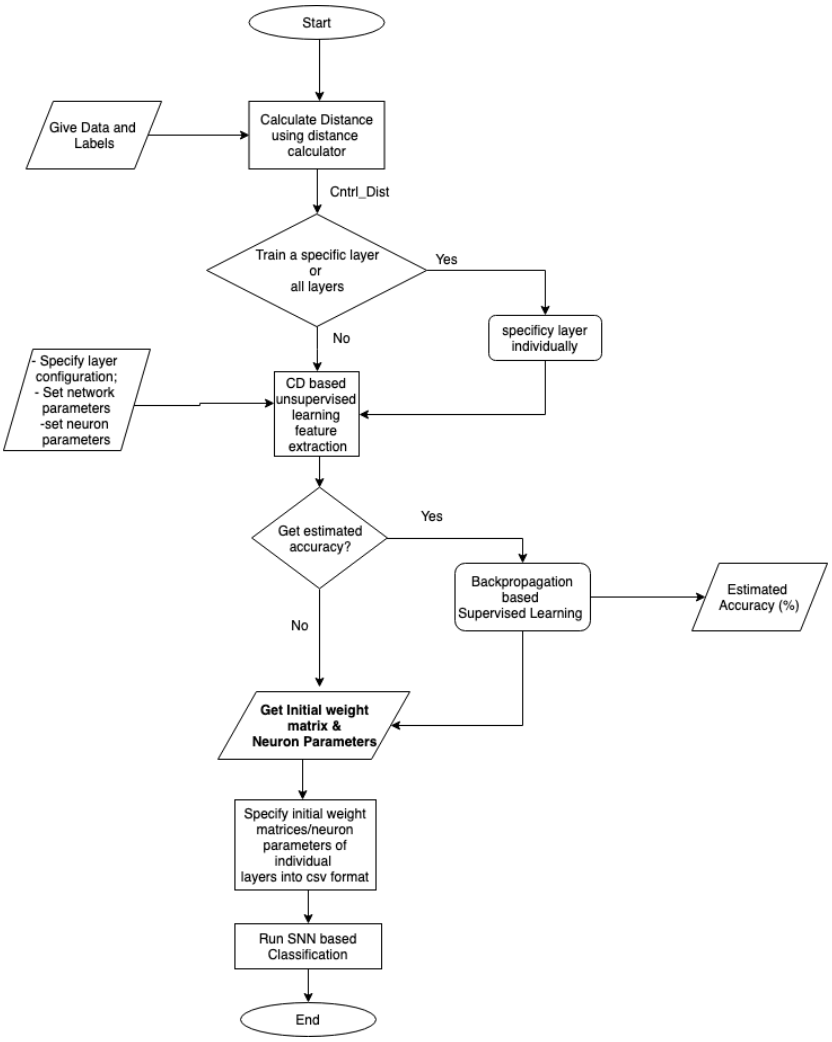

Figure 4.2: Proposed Methodology Flow

- Find mean of each cluster obtained by the t-SNE dimension reduction.
- Find Euclidean distance among each mean values obtained in previous step with other mean values.
- After finding minimum distance value obtained by the previous step. The smallest distance value among clusters will thus act as the control distance.

4.2.1 Dimension Reduction technique selection

There were various dimension reduction however there are three most widely used dimension reduction techniques:

- PCA(Principle Component Analysis)(28) : Doesn't use labels hence unsupervised, deterministic, uses Eigen vector used to find maximum variance

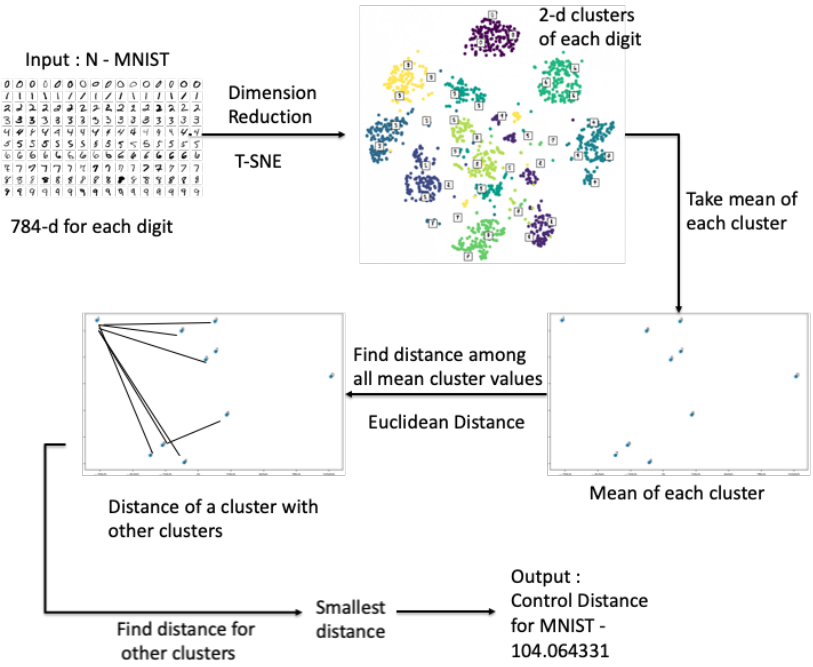

Figure 4.3: Distance Calculator Flow

- LDA(Linear Discriminant Analysis)(53) : Uses labels hence supervised, deterministic, Separates the class with the axis of largest class separation.
- t-SNE (t-Distributed Stochastic Neighbor Embedding)(35) : Probabilistic, t-SNE aims to minimize the divergence between two distributions , very resource-intensive hence in practice dimensions are reduced to 50 using PCA/LDA then t-SNE is used. Performs optimization across two distributions to produce an lower-dimensional embedding so that the points in the lower dimensionality are pairwise representative of how they appear in the higher dimensionality.

Figure 4.4 shows the difference between the choice of hyperplane in PCA and LDA. As it can be seen from the figure that the LDA chooses the hyperplane that ensures maximum separation between two classes whereas PCA can not do separation effectively.

Although t-SNE takes more time in separation of clusters but it separates the clusters most distinctively as compared to other dimension reduction techniques.

For distance calculator we will use t-SNE as a dimension reduction technique. Before understanding the working of the distance calculator it is important to understand working of t-SNE.

4.2.2 T-distribution Stochastic Neighbour Embedding (T-SNE)

This method is used to convert a high dimensional dataset into a low dimension dataset.

The working of T-SNE can be understood by taking a small example of dimension reduction from 2 dimension data to a 1 dimension data. The same priciple can be then extended to bigger dimesnions as well.

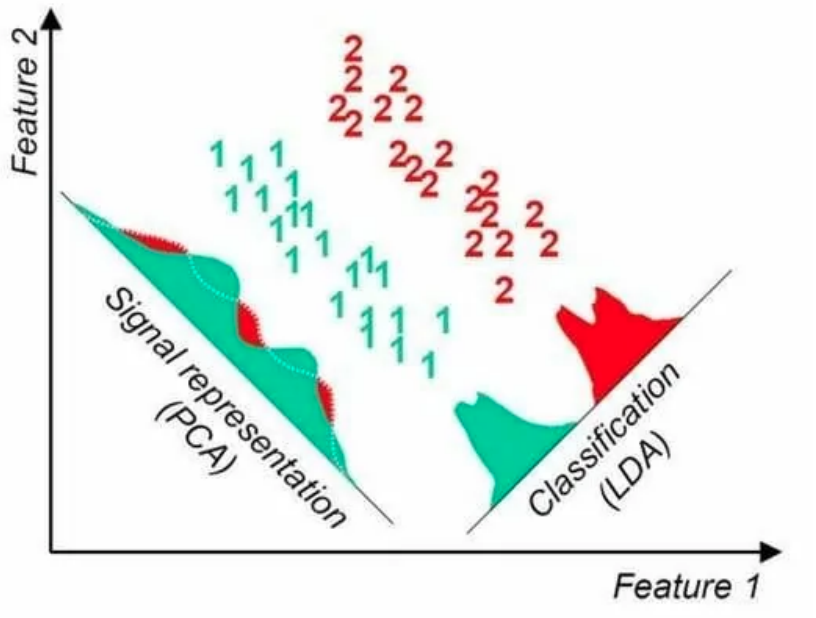

Figure 4.4: Figure showing difference between PCA and LDA Hyperplane selection(pca)

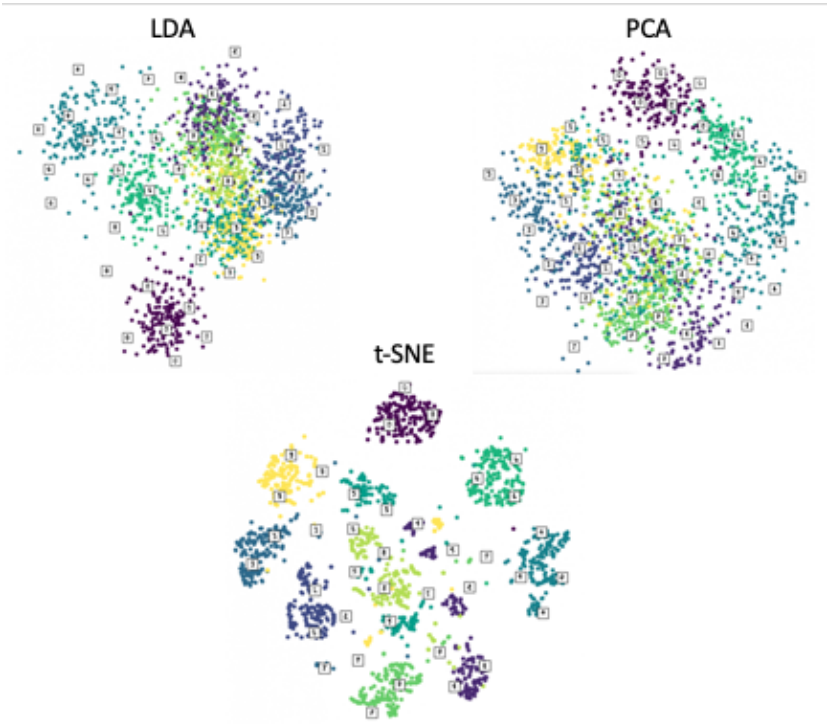

Figure 4.5: Comparison between various dimension reduction techniques(pca)

Figure 4.6 shows that if we do not project our data on a correct axis then the clusters are not distinguishable or are in a "mishmash", which means that when we are reducing the

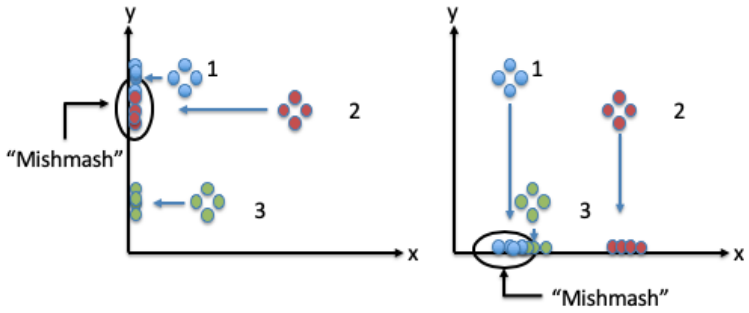

Figure 4.6: Projection of 2D data to 1D data with other dimension reduction techniques

dimensions of the data they are not segregated properly and tend to overlap each other. This is common in other dimension reduction techniques like PCA or LDA, this can also be seen in Figure 4.5

T-SNE at each step a point in the reduced dimension space us attracted to the points it is near in the scatter plot and repelled by the points it is far from.

This is done in following manner :

- Determine the "similarity" of all points in the scatter plot by measuring the distance between two points Figure 4.7.

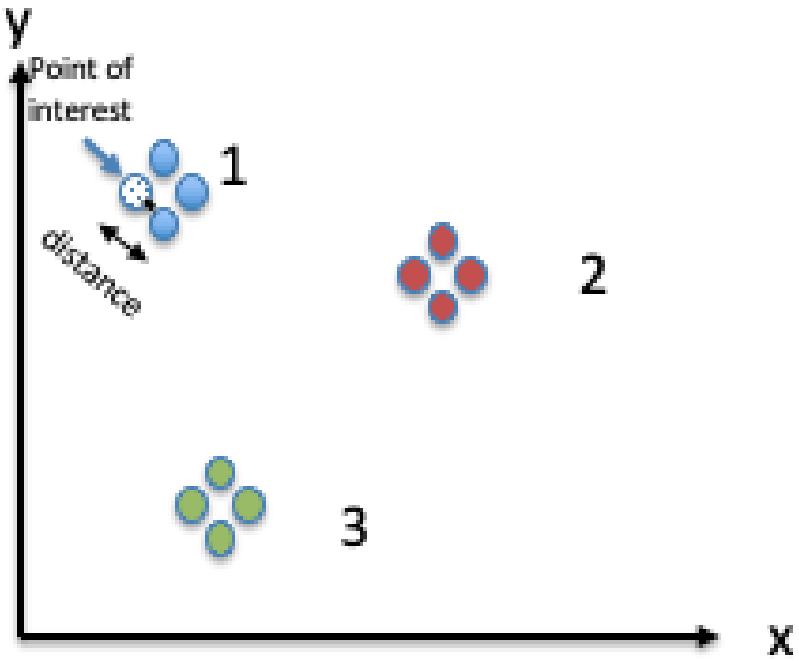

Figure 4.7: Distance between the point of interest and the other point

- Plot that distance on a normal curve which is centered around the point of interest.
- Draw a line from the point of the normal curve the length of that line is the "unscaled similarity" Figure 4.8.

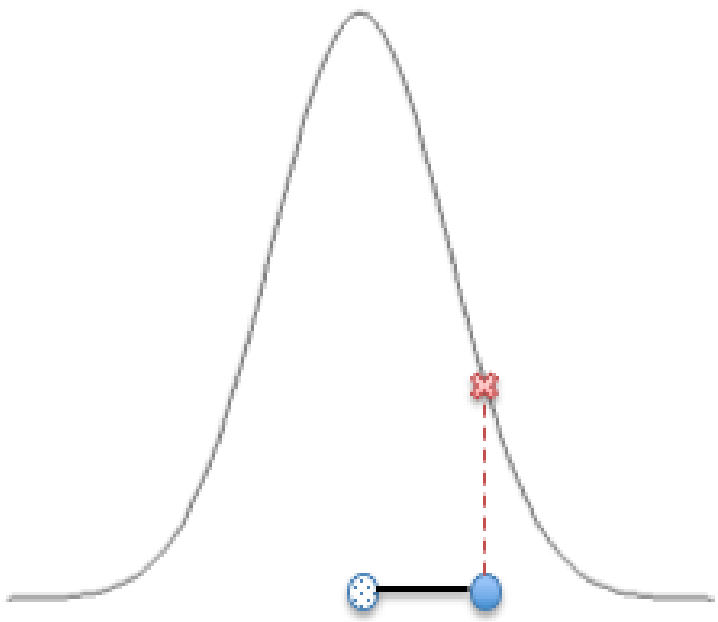

Figure 4.8: Distance between the point of interest and the other point on normal curve

- Now calculate "unscaled similarity" for all pair of pointsFigure 4.9.
- Thus showing far points have high similarity values and close points have low similarity values.
- Ultimately, we measure the distances between other points and the same point of interest using above mentioned step.
- Scale the "unscaled similarity" scores so that they add up to 1, this done because in case of "unscaled similarity" when a normal curve is plotted the the width of the curve depends on the density of the cluster near the point of interest. Thus, after scaling values the similarity scores will be sane for both clusters Figure 4.10.
- Scaled Score = $\frac{Score}{\Sigma Score}$
- Now, we repeat above steps for all points
- Project the data into reduced dimension space (here on a number line) Figure 4.11
- Thus, T-SNE retains most from the original dimension.

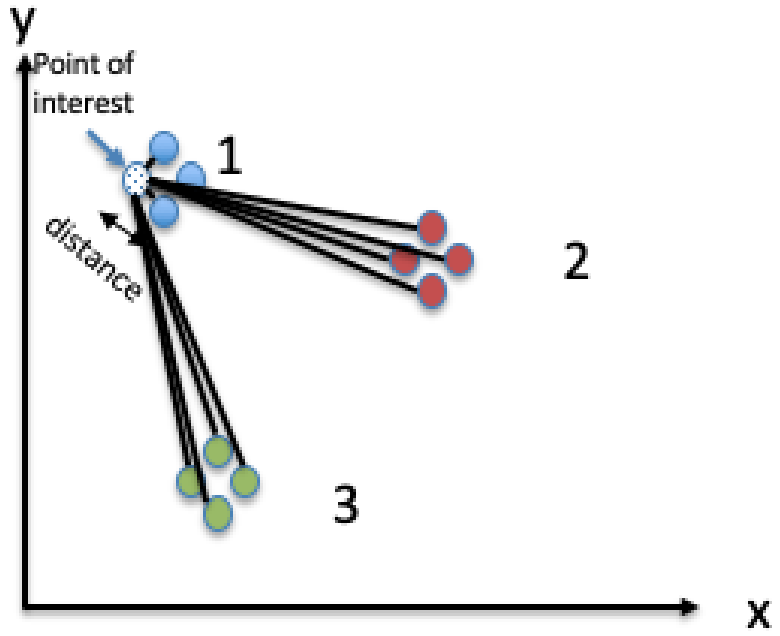

Figure 4.9: Distance between the point of interest and all other points

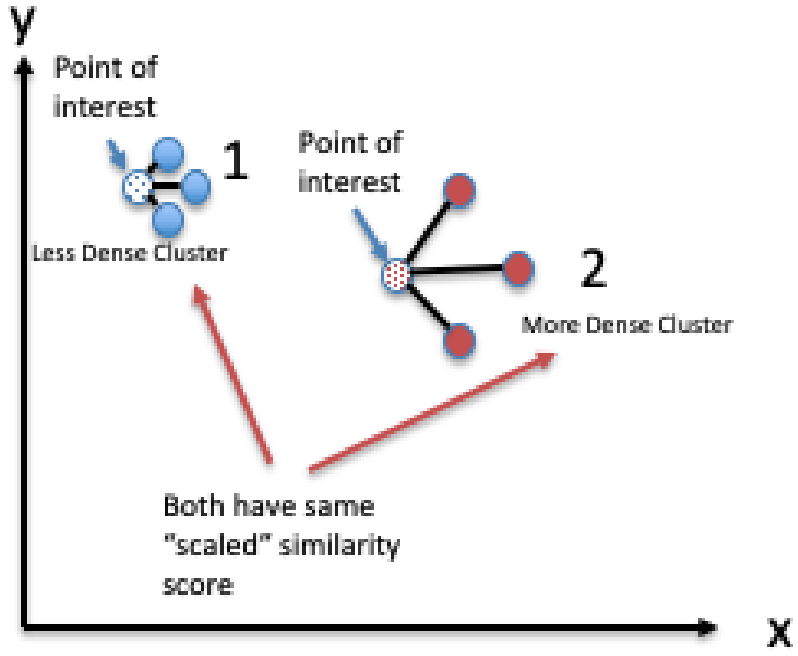

Figure 4.10: Two clusters with same scaled similarity score

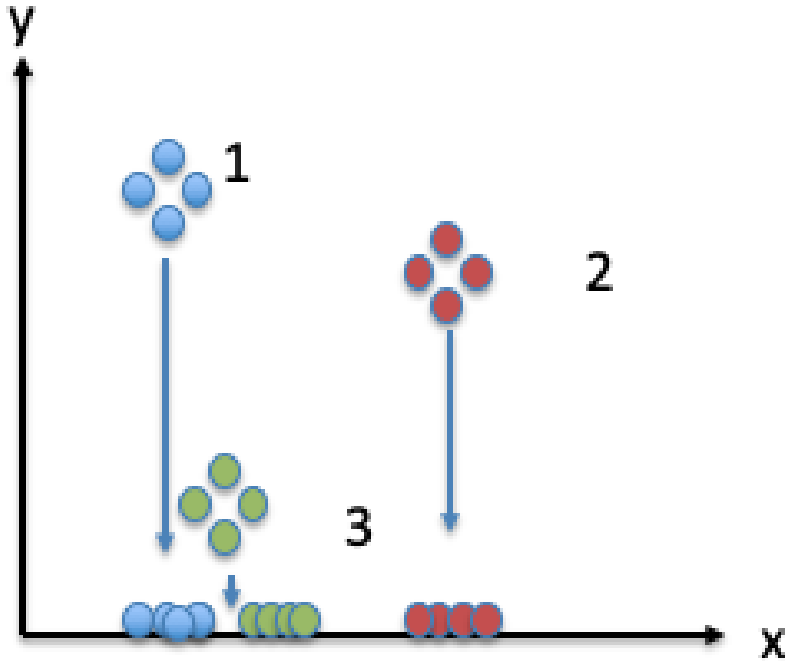

Figure 4.11: Dimension reduction of 2D to 1D using T-SNE

4.2.3 Euclidean Distance

Euclidean Distance is the most simple form of distance matrices, it is calculated just by using Pythagoras theorem for two dimensional space.

$$distance^2 = x^2 + y^2$$

As we have reduced our 784 dimensional space to 2 dimensional space and each class has multiple images, we use pairwise euclidean distance because it finds distance between each rows of the given vectors.

It is also computationally efficient for finding data with sparse data.

4.3 DBN based training

In this phase we take help from (42) that helps in using an equivalent approach to train LIF spiking neurons using equivalent Siegert Neurons that have similar transfer function as discussed in chapter 5.

Although DBN based training phase is very flexible and can be used in multiple ways but here in this thesis we can use it in the following manner :

- Pick 1 layer.
- Apply data sample (here it can be 1/10/50/100/500/1000/10000 samples per class depending upon availability of dataset)

- Contrastive Divergence based training.
- Using above steps train other layers.
- Integrate all layers.
- Weight matrix obtained can be used for SNN model.
- If user wants to know the estimated accuracy then use backpropagation based supervised training on the model to find estimated accuracy.

4.3.1 Getting the weight matrix

While doing the DBN based training the weights keep getting updated for the relevant synapses, we can use the then generated weight matrix in an SNN based classifier with same network configuration. After completing the training methodology we perform the following steps :

- DBN based pre-processing step generates a .mat file as an output, the generated .mat files are transferred into Joahn's Simulator path
- The .mat contains weights of all layers, thus it is then converted into a csv format.
- As Johan's simulator accepts weight values in a csv format, partition the new generated csv file into new csv files for each layer comprising of weight parameters for respective layers.
- Also change the offset values, Neuron index values and other neuronal parameters(from the .mat file in first step) before running the Johans Simulator.
- Change the weight matrix source from random to the file paths containing csv files with weight matrix of each path.
- Run the simulation by running main.m in Johan's simulator.

4.3.2 Learning for lesser number of images per class or N-shot learning

Here in order to make we modify the contrastive divergence used in (42) in such a way that the distance calculated from the distance calculator is used to control the weights of the synapses. If the distance between the reconstructed image and the given image at the visual layer is larger than the control distance fed by the programmer then the weighted value of distance is incorporated in the weight updation rule otherwise if the distance is lesser than the control distance the standard contrastive divergence is used.

The significance of this modification is to push the learning towards making reconstructed image more closer to the sample image and reducing the distance or degree of dissimilarity among them. We will be discussing the accuracy that can be achieved with lesser data samples in next chapter chapter 7

4.4 Conclusion

The Proposed Methodology is in line with the objectives of this thesis and proposes to give an weight matrix for the SNN classification which gives sufficient classification accuracy.

We will be doing a detailed analysis of this approach in chapter 6.

Distance dependent Contrastive Divergence Learning

5

Before going into the proposed algorithm it important to understand the need of this algorithm and RBMs. In this chapter we will be discussing the current limitations with the contrastive divergence algorithm, explore the possible solution and at last propose the distance dependent contastive divergence algorithm.

5.1 Limitations with RBM

In chapter 3 we saw that using spiking RBM can be used to get the weight martix for a SNN of same configuration which is one of the objectives of this work. However, using spiking RBM doesn't address the objective of getting that weight matrix with a very small dataset. This is due to the fact that the learning method in RBM i.e contrastive divergence(CD) has no controllability, which means that the outcome of the learning can not be controlled, this is not a concern when we have huge sums of data available for training but when it comes to a very small dataset we are not able to train the network to a considerably higher classification accuracy, as only selected neurons are able to detect features due to to limited small training set.

Disadvantage of RBM :

- Learning can not be controlled. Thus, there is very limited controllability during the learning process.

Deep learning has shown particularly higher classification accuracy when working with large amount of as compared to other traditional machine learning algorithms (DS).

5.2 Solution

Thus, in order to overcome the inability of CD algorithm of not performing well with small dataset needs to be addressed and make spiking RBM to provide weight matrix from a very small training dataset, we have to find a way to make Contrastive Divergence learning more controlled. So that in particular we make weights for detected features more stronger and reconstructed images and more similar to the given sample images from a small dataset.

One way to solve this problem can be to make reconstructed images that are more similar to the given sample image by calculating the distance between two images and then comparing the obtained distance with a control distance for a dataset found using a distance calculator chapter 4.

As we have already discussed about working of Contrastive Divergence(CD) algorithm in chapter 3, CD tries to reconstruct the given sample image in a RBM by changing the weights of the synapses between a hidden and visible layer. As increasing number of samples pass through the RBM the the reconstructed image becomes similar to the sample image. However,

this can not be said true when we have a very small dataset of just 100 images instead of 60000 images to train the network, but this can be done if we can make the reconstructed image similar to the given sample image. In this chapter we will be discussing about a new modified CD learning which will be dependent on a distance parameter which can be calculated by a distance calculator chapter 4. The distance calculated by distance calculator is then used to control the learning done in the proposed methodology.

Input: input : RBM($v_1 ... v_m$,$H_1 ... H_m$),training batch D, control distance $dist_{cntrl}$
Result: output : Learned weights
w=0;
for $v \epsilon D$ **do**
  Initialise $v^(0) \leftarrow v$; **for** *t=0....k* **do**
    for *i=1.....n* **do**
      $sample : h_i^t = p(h_i|v^t)$
    end
    for *j=1.....m* **do**
      $sample : v_j^(t+1) = p(v_j|h^(t))$
    end
    Use TSNE for $v^t|v_j$ into 2d
    Find $Distance = v^t|v_j$
  end
  if *if* $Distance \leq dist_{cntrl}$ **then**
    $w \leftarrow w + \eta[\sigma(w_{v_d} + c)v_d - \sigma(wv + c)v_d]$;
  else
    $w \leftarrow w + dist_{cntrl}[\eta[\sigma(w_{v_d} + c)v_d - \sigma(w_v + c)v_d]]$;
  end
end

Algorithm 1: Distance dependent CD algorithm

Thus from algorithm 1 it can be seen that we are trying to move those weight values that have higher distance than the control distance value, due to this only those corresponding weights are increased.

Thus especially helps when dataset is small, because in smaller dataset the distance between two visible layer tends to increase then the control distance, which causes bad reconstruction of image whereas, in large dataset this distance is usually within control distance value.

The use of distance in weight update formula is to increase the weight for those specific synapses. So when the image comes in next epoch, the same synapse will have higher weight and the distance with reconstructed image will decrease. This is especially helpful when a small dataset is there and it runs multiple epoch through the network thus becoming very strong over time.

The control distance acts as a "control knob" that drives the learning towards reconstruction of sample image, thus due to this user can control learning by just varying the distance values. This control distance acts as a hyper parameter value that user can directly use to control the learning process.

5.3 Conclusion

This chapter proposes a unique algorithm that helps in achieving one of the objectives of this thesis i.e to get weight matrix with small dataset(N-shot training), we also discussed about the limitation of RBM learning with small dataset and how our proposed distance dependent learning will help in achieving our objectives. Whether proposed algorithm helps in achieving the goal or not is analysed in chapter 6

Part III
Results

6 Results

In this section we will analyse the proposed methodology and show how proposed methodology helps in achieving the the key objectives of this thesis.

6.1 Experiment Setup

6.1.1 Preparing the dataset

The original MNIST dataset consists of 60000 training images and 10000 test images with around 6000 images per class in training set. For our purpose of experimentation we will make reduced MNIST datasets i.e N-MNIST where "N" represents number of images per class. For example : 10-MNIST indicates there are 10 images in each class and there are 100 images in total.

Following steps were followed for making a N-MNIST dataset :

- Shuffle the original training dataset, we don't make any changes to the test dataset.
- Select top-N (can be 1/10/50/100/500/1000 images class)depending on simulation
- Shuffling is done before any new simulation to increase the randomness of the N-MNSIT.
- Repeat the procedure for making 1-MNIST, 10-MNIST, 50-MNIST, 100-MNIST, 500-MNIST and 1000-MNIST.

6.1.2 Distance Calculation

As discussed about the working of distance calculator in chapter 4 Distance Calculator is used to calculate the distance between each cluster whenever a new kind of dataset is used. For MNIST the control distance is : 104.06433184910078

User has to enter the control distance value in the DBN simulator as one of the parameters. Distance Calculator is to be used only once when a new dataset is introduced.

6.1.3 DBN Training

After getting the control distance for the simulations we enter the value in simulator. We also do the one-hot encoding of the dataset. This is due to the reason that many machine learning algorithms like contrastive divergence can not operate on label data directly as they require all input and output variables to be numeric.

Once the dataset in one-hot encoded we enter all parameter values, network configuration.

After entering all parameters we run the simulations and once the DBN training is complete, we transfer the weight matrix to the SNN classification environment.

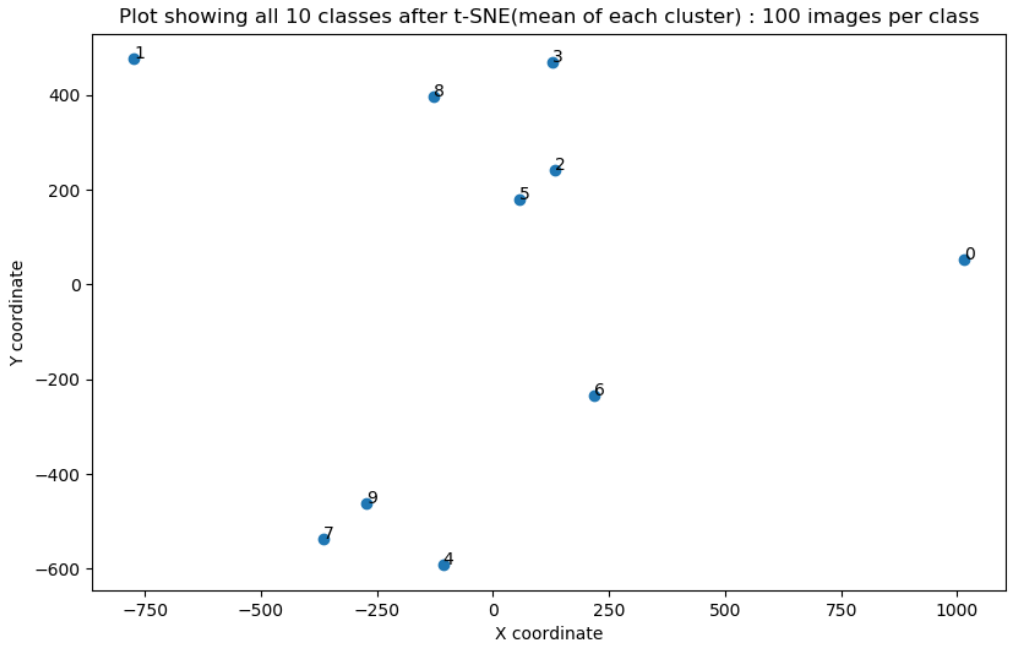

Figure 6.1: Mean of clusters using t-SNE

6.1.4 SNN classification

After porting the weight matrix to the SNN classification environment also known as In-house Simulator we enter the corresponding parameter values and network configuration that were used in the DBN training phase.

Now pass the test set of the dataset to get the classification accuracy.

For any intended simulation with N-MNIST dataset 3 runs are made and average is taken.

6.2 Dimension Reduction

As discussed in chapter 4 t-SNE is the best choice for the class separation of the dataset.

As it can be seen from Figure 6.1 and Figure 6.2 that shows the mean value of the clusters separated using t-SNE and LDA, it can be seen that using t-SNE the separation is more distinguishable as compared to that of LDA for the MNIST dataset. The smallest separation distance among theses clusters is taken as the control distance.

6.3 Comparison

(46) is an existing method to get weight matrix that converts ANN to SNN. However, this method manually converts ANN based network into SNN which has to take following things into consideration during conversion :

- ANN has graded activation of analog neurons where as for SNN has firing rate of spiking neurons.

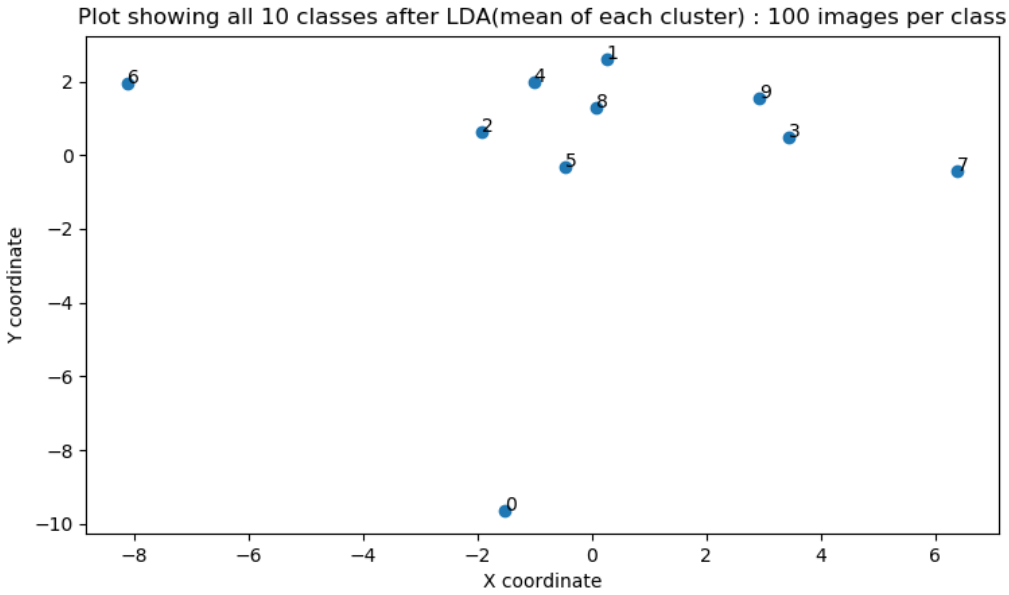

Figure 6.2: Mean of clusters using LDA

- Biases are pretty are standard in ANN, there are no biases involved in SNN thus biases from ANN are adjusted by presenting ANN generated biases can be presented with an external spike input of constant rate proportional to ANN biases.
- Softmax is used on output of deep ANN, In SNNs output neuron corresponding to the class it belongs. However, softmax in ANN is converted to ANN by just SNN accumulating spikes when external generator determines about spike to be generated and then softmax is done.
- The supervised learning in ANN is done by backpropogation whereas in SNN unsupervised method STDP is widely used for learning, but there are now some backpropagation based approaches but still these methods aren't good enough to be deployed into deep SNN architectures yet.
- Time taken in this conversion based approach is very high and for deep networks it can go to one day as well.
- ANN based method doesn't taken Spiking neuron parameters into consideration.

This also shows that the conversion based approach is very time consuming and is just an approximate conversion of ANN into SNN. This method doesn't take neuron behavior into consideration and also doesn't address of training with as less as possible images per class. However, our concern is not about time but aim is to get weight matrix with a very small dataset which is addressed by the proposed methodology.

The proposed methodology is based on siegert neurons that has same transfer function as that of LIF neuron this helps in bridging the gap between ANN to SNN conversion. This also make proposed methodology behavioral equivalent to the SNN.

- Proposed methodology also considers firing rate of siegert neuron which makes it equivalent to a spiking neuron.
- The RBM based Architecture is easily replaceable into SNN by just directly transferring weight matrices, thus has lesser approximation error issues as compared to that conventional approach.
- Proposed methodology assists SNN to get sufficiently higher accuracy with as less as possible data samples.
- ANN based conversion approach for deeper networks can take up to one day time to generate weight matrix for a deep architecture, Whereas proposed methodology can do it in several minutes/hours.
- The proposed methodology provides weight matrix irrespective of learning method to be used in SNN and considers spiking/LIF neuron parameters into consideration.

The proposed methodology has more behavioral equivalence to the SNN as compared to the conventional approach, because proposed methodology take individual neuronal parameters into consideration whereas the conventional approach fails to take such nuances into consideration.

Thus, proposed method seems to be a more natural fit for training deep SNNs that can train with as less as possible data samples or small datasets and gives sufficiently high accuracy during classification.

6.4 N-shot learning

In this section we will be discussing results about the learning with as less as possible data samples.

Images per class(N)	Accuracy standard CD	Batch Size	Epochs	Accuracy New CD	Increase
1000	88.16	50	6	89.06	0.9%
500	86.06	50	6	87.11	1.05%
100	82.54	50	50	85.43	2.89%
50	73.66	50	50	79.24	5.59%
10	63.72	100	1000	71.68	7.96%
1	27.55	10	10000	37.53	9.98%

Table 6.1: N-shot results

New Accuracy is the classification accuracy achieved using the modified CD whereas old accuracy is the classification accuracy with the standard model.

Table 6.1 shows that the modified algorithm is doesn't show much improvement as it was intended because the weighted condition will act only when the distance between sample image and reconstructed image is greater than the control distance. As in 1000 images per class there are enough samples to bring the system to a lower energy state, thus it can reconstruct image pretty well. Similar behaviour can be seen in 500 images per class.

The modified algorithm starts showing improvement with 100 images per class +2.89%, with 50 images per class +5.59% , with 10 images per class +7.96% and 1 image per class

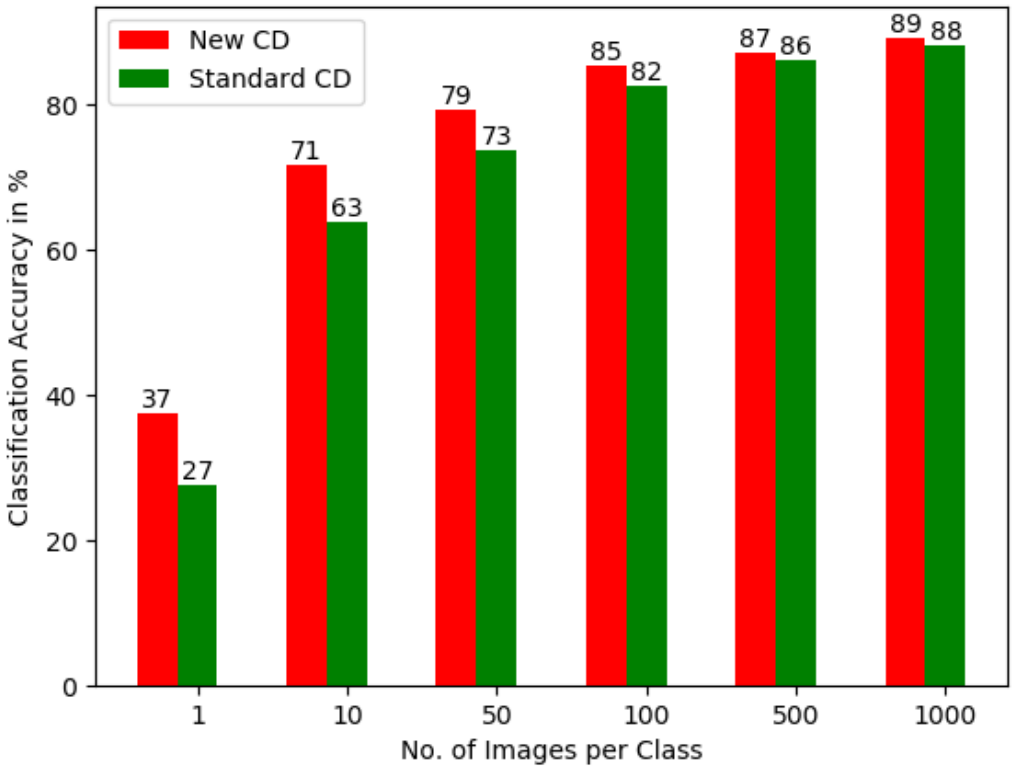

Figure 6.3: Comparison between distance dependent state and standard CD

+9.98%. Thus it clearly demonstrates that the new modification in the learning method helps in improving the accuracy especially with smaller dataset is effective in such scenarios. As the improvement in accuracy increases as the dataset size reduces.

The reason for achieving high accuracy improvement with small dataset can be attributed to the modified CD algorithm. Because in case of small dataset the distance between input sample and reconstructed image is larger than the control distance which causes the algorithm to update the weight of synapses as a product of distance and the change in weight. This weighted distance tends to reduce the distance between reconstructed image and sample image which causes increase in accuracy. This was not possible with the standard CD approach.

Another important observation can be made from the table is about the batch size and epoch. The batch size defines the number of samples that will be propagated through the network, Epoch defines the number of iterations the samples will pass through the network.

While running the simulations a relation between batch size and number of samples was kept in mind. For larger dataset batch size $<$ no. of samples was used.

Advantages of batch size $<$ no. of samples :

- It requires less memory.
- Typically networks train faster with mini-batches.

Disadvantage of batch size $<$ no. of samples:

- The smaller the batch the less accurate the estimate of the gradient will be.

That's why smaller batch size was useful in using larger datasets, however for small dataset like 1-image per class and 10-image per class, batch size couldn't be made much small.

That means they had batch size = no. of samples, because computing the gradient over the entire dataset is expensive, thus the gradient of that sample may take completely the wrong direction. But, the cost of computing the one gradient was quite trivial. As you take steps with regard to just one sample you "wander" around a bit, but on the average you head towards an equally reasonable local minimum as in full batch gradient descent.

Thus there has to be a trade-off to be made between accuracy and the computational costs. But for smaller dataset 10 epochs of 1 image per class will have nearly same computational costs of 1 epoch for 10 images per class thus we can afford to run higher number of epochs for smaller dataset.

Figure 6.3 shows the comparison between standard contarstive divergence and distance dependent contrastive divergence algorithm.

Images per class	Accuracy	Batch Size	Epochs
1	9.8%	1	1
1	10.8%	1	50
1	11.3%	1	1000

Table 6.2: 1-shot results for increasing epochs with batch size =1

Images per class	Accuracy	Batch Size	Epochs
1	15.78%	10	100
1	20.46%	10	500
1	22.55%	10	1000
1	37.53%	10	10000

Table 6.3: 1-shot results for increasing epochs with batch size =10

Table 6.2 batch size = 10 represents complete dataset on 1 image per class. As whole dataset goes in one iteration we can see higher accuracy as compared to batch size = 1 as shown in Table 6.3. During one iteration having multiple data from multiple class helps in adjusting weights, whereas when data from one class is given. weights get adapted selectively as shown in Figure 6.5 and Figure 6.6 respectively.

Another interesting observation can be seen that in Figure 6.5 that due to high number of epochs on just 10 images belonging to only one class each have weights for concerned synapses very well defined but for other it doesn't modify weights that good. However, Figure 6.6 due to larger dataset change in weight among all synapses is there thus resulting in better classification.

Images per class	Accuracy	Batch Size	Epochs
1	37.53%	10	10000
10	71.68%	100	1000

Table 6.4: 10-shot and 1 shot results comparison for same computational cost

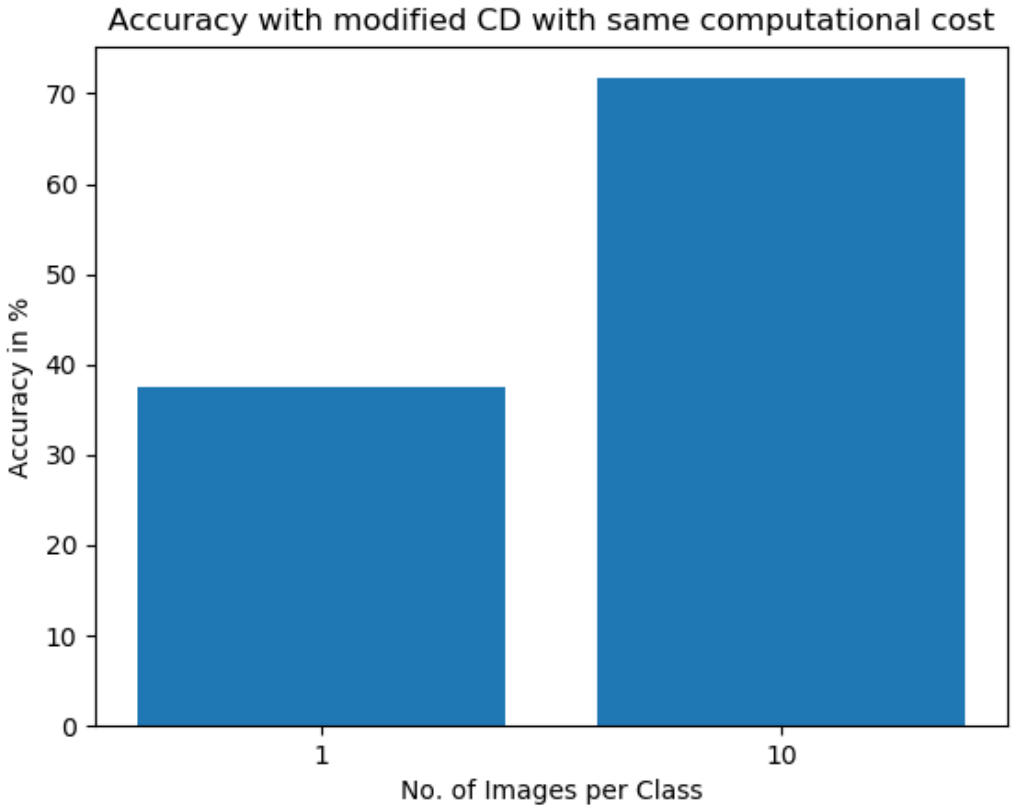

Figure 6.4: 10-shot and 1-shot comparison for same computational costs

Table 6.4 shows that for the almost same computational costs we get better classification accuracy with 10-MNIST dataset this is because of the fact that there is much more variety in the dataset in each class which helps in recognising more features. Thus, this also shows that having much more variety in the dataset can help us getting higher classification accuracy Figure 6.4.

In one shot $batchsize10 = 1$image from class $= 1$ epoch which is repeated for 10000 times thus having 10^5 steps, similarly for ten shot $batchsize100 = 100$ images in one iteration in this case 10images x 100 batches $= 1$ epoch and with epoch $= 1000$ will also lead to 10^5 steps.

6.4.1 Shift in positions of cluster means

Figure 6.7 shows the shift in positions of cluster means with increase in the number of samples. It can be seen that the mean values for 10/100/1000 images per class shifts depending upon the number of samples spread thus resulting in the shift of the cluster means.

6.4.2 Optimal distance

Figure 6.10 shows the variation of accuracy when distance is controlled. This also shows that the training methodology can be controlled by using an external parameter which help in

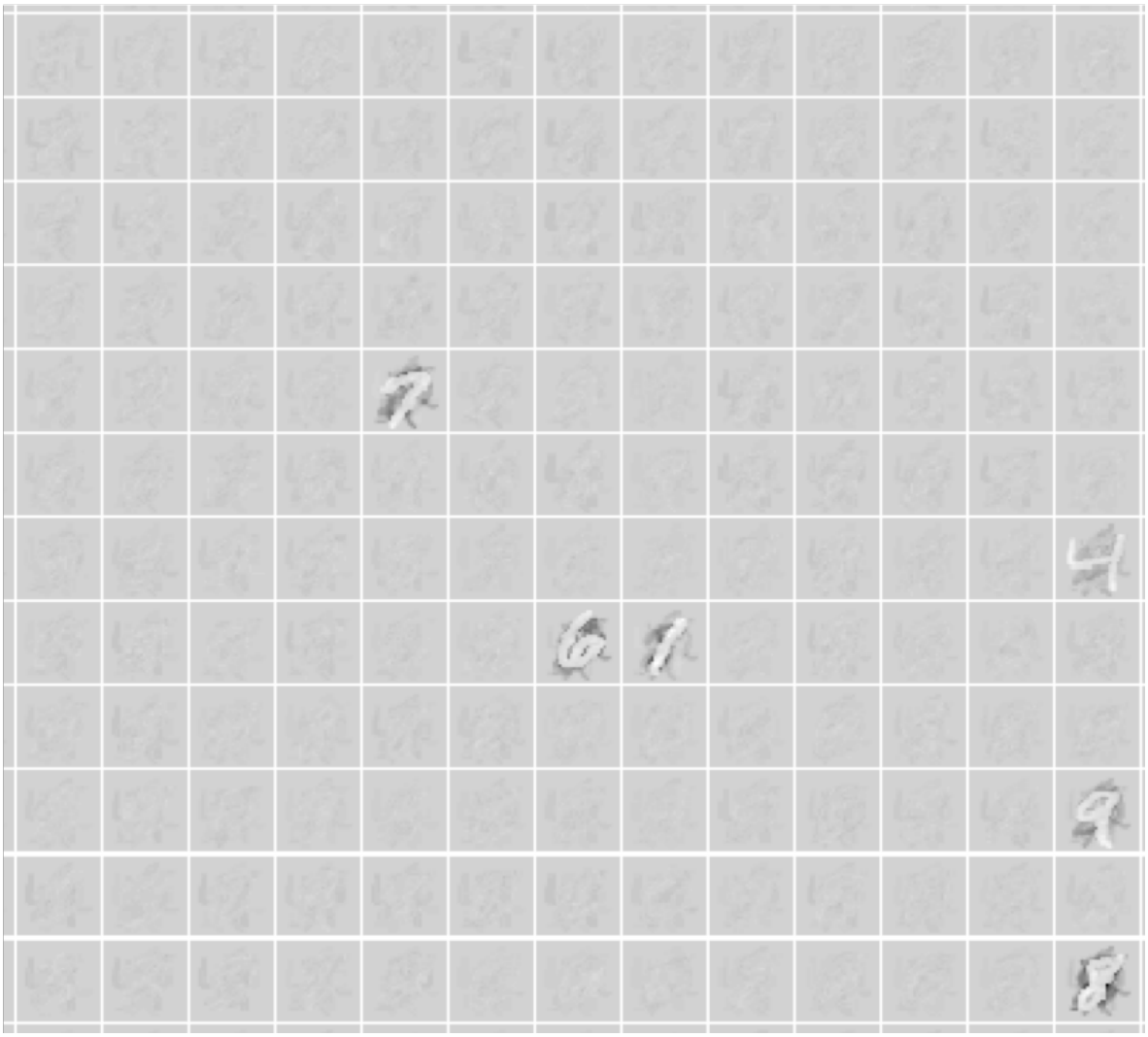

Figure 6.5: 1 Image per class

attaining better accuracy which was not possible in standard CD. The distance calculator yields the optimum distance value i.e for MNIST = 104.0643318.

6.4.3 Comparison distance on each epoch

Figure 6.9 shows the comparison between the standard CD and the proposed modified CD. We can clearly see that due to the new proposed distance dependent algorithm, the distance between the reconstructed and the given input image is reduced below the threshold level i.e control distance for MNIST = 104.0643318 .

6.5 Reliability

In this section we will be discussing about the reliability of the proposed methodology in terms of actual hardware implementation. In this experiment we used actual values of neuron

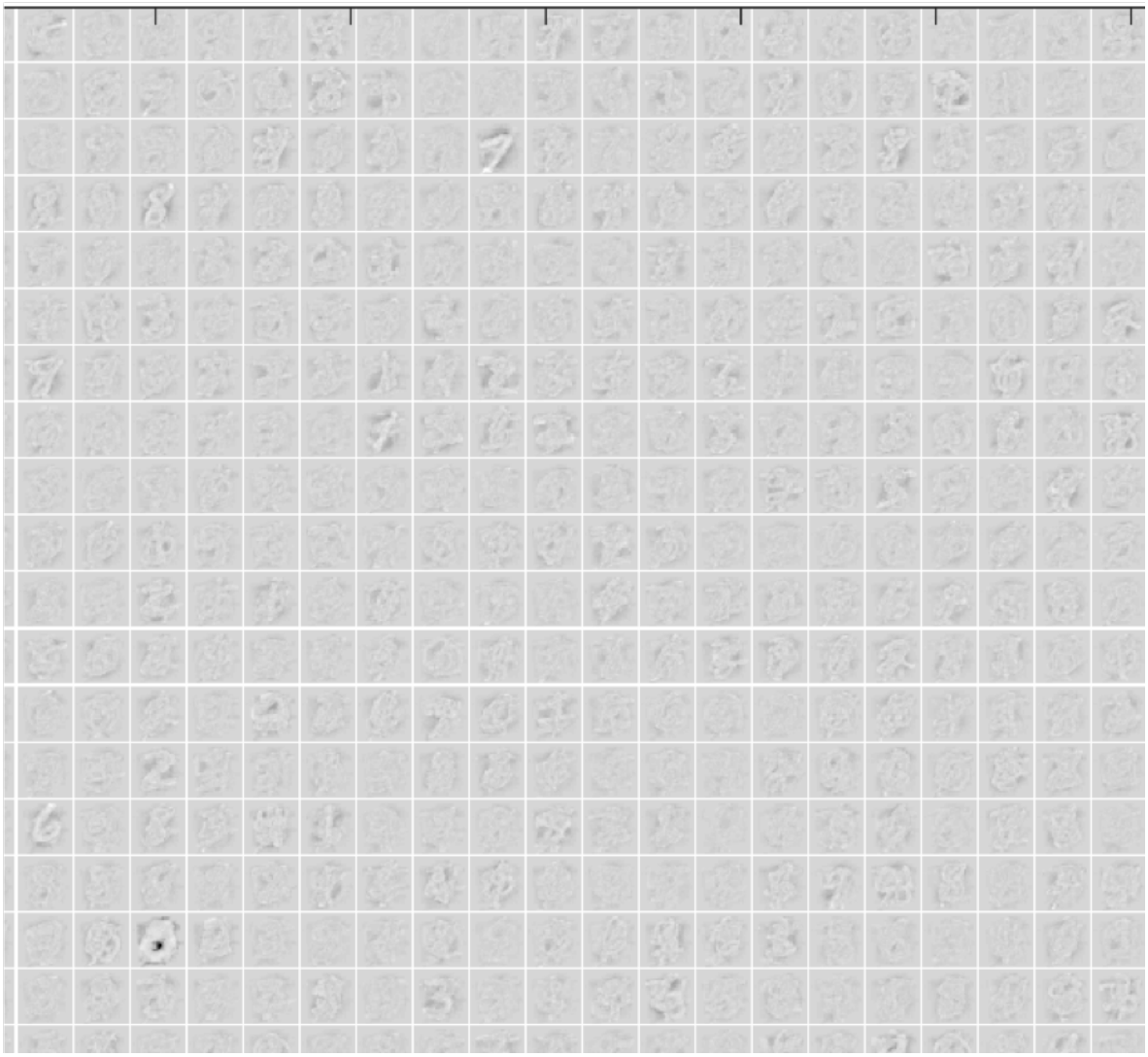

Figure 6.6: 10 images per class

parameters that are eventually used in SNN hardware. The objective of this experiment is to validate that the RBM based methodology performs as an equivalent SNN network and shows that weight matrix generated by this methodology can be realistically deployed as SNN.

Neuron Parameters	Values
Membrane Time Constant τ_m	5msec
Neuron Membrane Threshold v_{thr}	350mV
Refractory Period T_{ref}	1msec
Accuracy	65.75%

Table 6.5: 10-shot MNIST Training with real SNN values

Table 6.6 shows that accuracy with 10 images per class with hardware values is a bit lesser as compared to that of values we got originally, this is due to the issue of scaling. $\tau_m =$

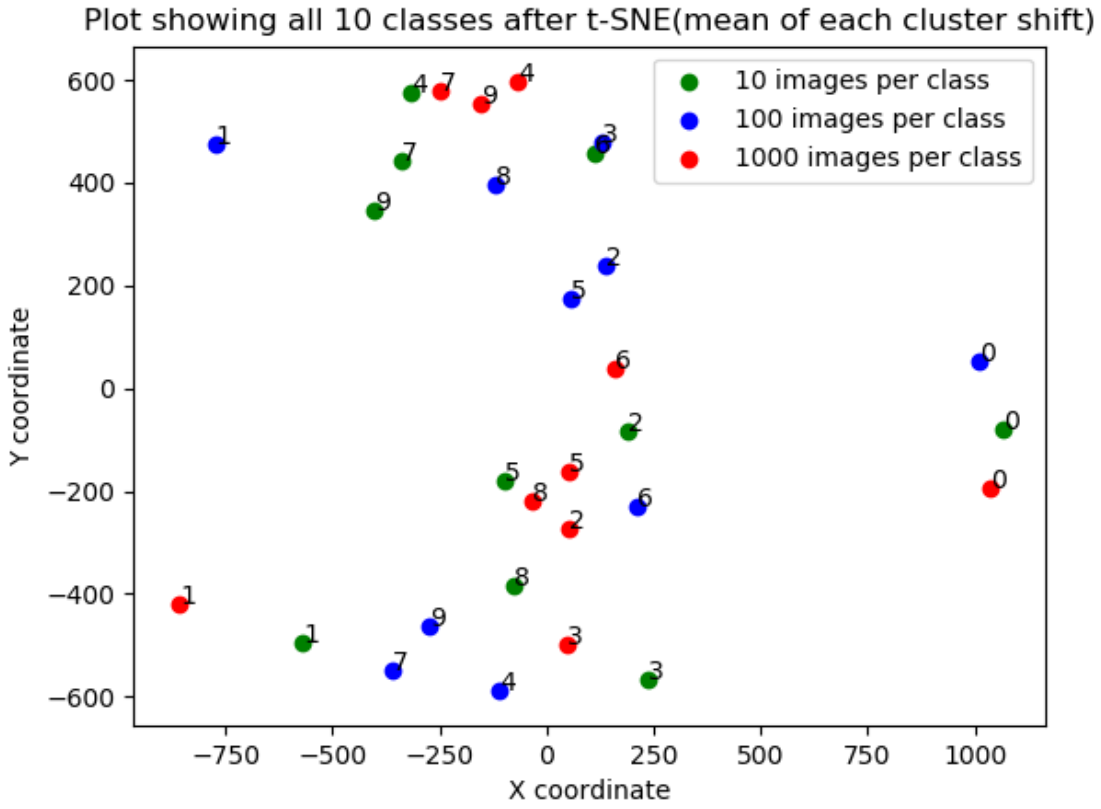

Figure 6.7: t-SNE cluster mean positions for 10/100/1000 images per class

R_m*C_m i.e the membrane time constant is the product of membrane resistance and membrane capacitance of a neuron. The In-house simulator takes R and C values of the neuron as input parameters whereas the proposed methodology takes membrane time constant as input.

6.6 Flexibility

Another aspect of proposed methodology that it should also work on other dataset as well. In order to check its flexibility we used CIFAR-10 dataset which has 32x32 coloured 50,000 training images and 10,000 test images. As CIFAR is a coloured dataset and more complex features as compared to MNIST dataset. There were multiple simulations that we ran in order to check its feasibility.

In this section we will just focus on proposed methodology's compatibility with other dataset in order to demonstrate that methodology is a generalised methodology and can be applied to datasets other than MNIST. Although in this thesis out objective is to develop a methodology that can produce a weight matrix with as less as possible data samples, but it is also important to demonstrate that this methodology can be extended to other datasets as well. Thus, here in this section we have discussed about the same.

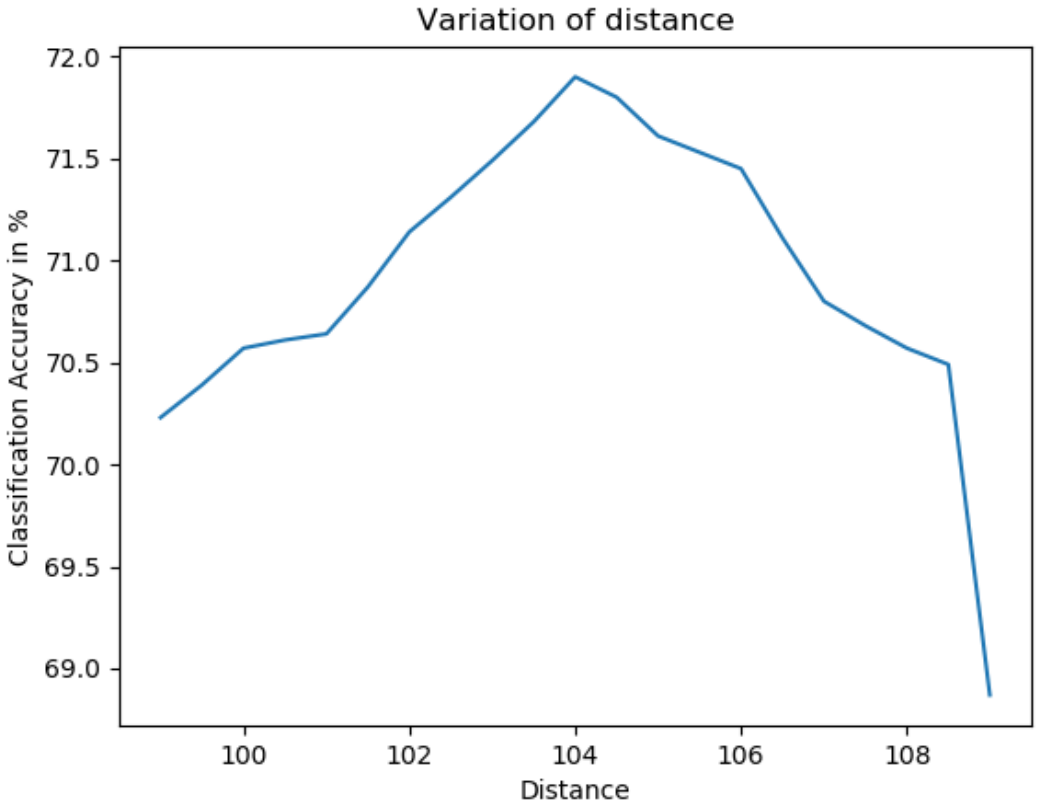

Figure 6.8: Variation of Distance

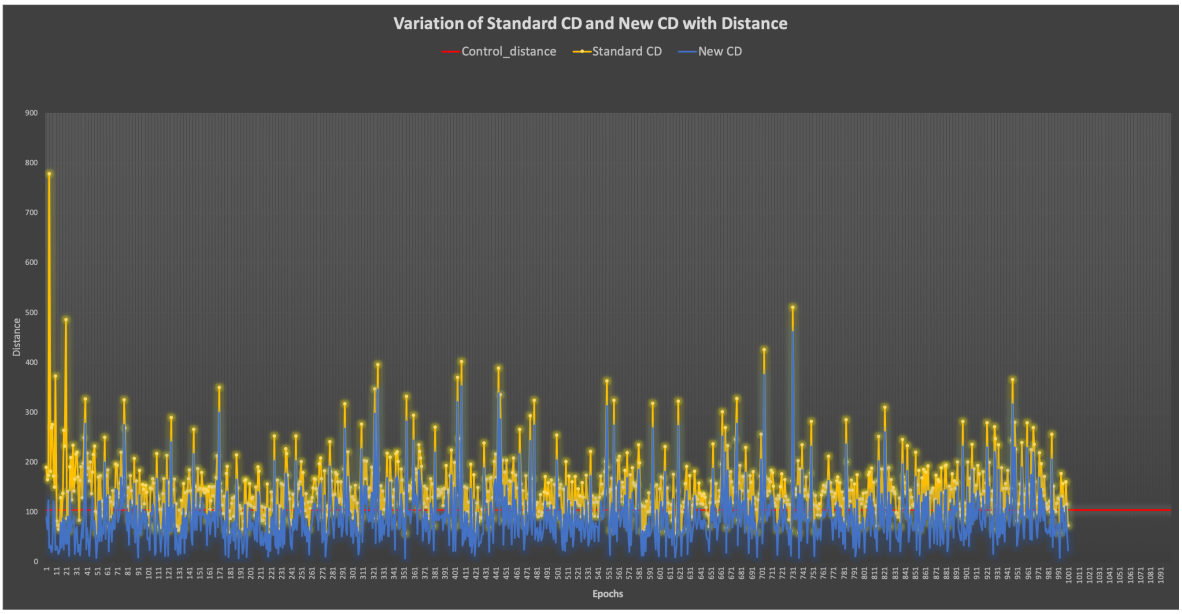

Figure 6.9: Distance v/s Epoch comparison between Standard CD and Modified CD

6.6.1 Converting CIFAR dataset to Grayscale format

CIFAR is a more complex dataset as compared to MNIST which has digit images in a grayscale format with very simple features. CIFAR has separate RBG components for every image pixel making features even more difficult to detect using RBM. However, as we just want to identify features in the image which can also be detected in a grayscale format as well. Thus, CIFAR dataset is converted to grayscale by simply taking a mean of all three components i.e RBG. However, it was also observed that by simply taking the average the Red component was more dominant in order to compensate that weighted average was used where Green component was given more weight, which also helps in detecting features more easily.

Another advantage of grayscale conversion is now instead of 3072 input neuron with RBG now only 1024 input neurons are required.

6.6.2 Simulations with CIFAR dataset

After converting coloured CIFAR dataset into grayscale various simulations were made on N-shot to verify whether proposed methodology is applicable on CIFAR dataset as well and how much it can improve.

Image per Class	Accuracy	Batch Size	Epoch
1000	53.87%	50	6
500	42.14 %	500	50
100	32.04 %	100	70
50	27.66 %	50	100
10	26.82%	100	1000
1	22.08%	10	10000

Table 6.6: N-shot with CIFAR-10 with 1024-400-10 Configuration

(32) showed that state of art ressult for the DBN of same network configuration with CIFAR-10 dataset had an accuracy of 65%. Table 6.6 also showed improved in accuracy with modified CD. Accuracy mentioned in table is the improved accuracy.

The focus in this simulation was to check if the features can be detected by the methodology or not and can be used for other dataset. In these simulations classification was not done on In-house simulator but rather the test set was passed through the same DBN based network on which training was done. Which shows that proposed methodology is flexible enough for more complex dataset.

6.7 Conclusion

By this analysis we can see that our assumed hypothesis for the distant dependent algorithm achieves the objective of getting sufficiently higher accuracy with smaller dataset. The proposed methodology gives the weight matrix with a very small dataset.

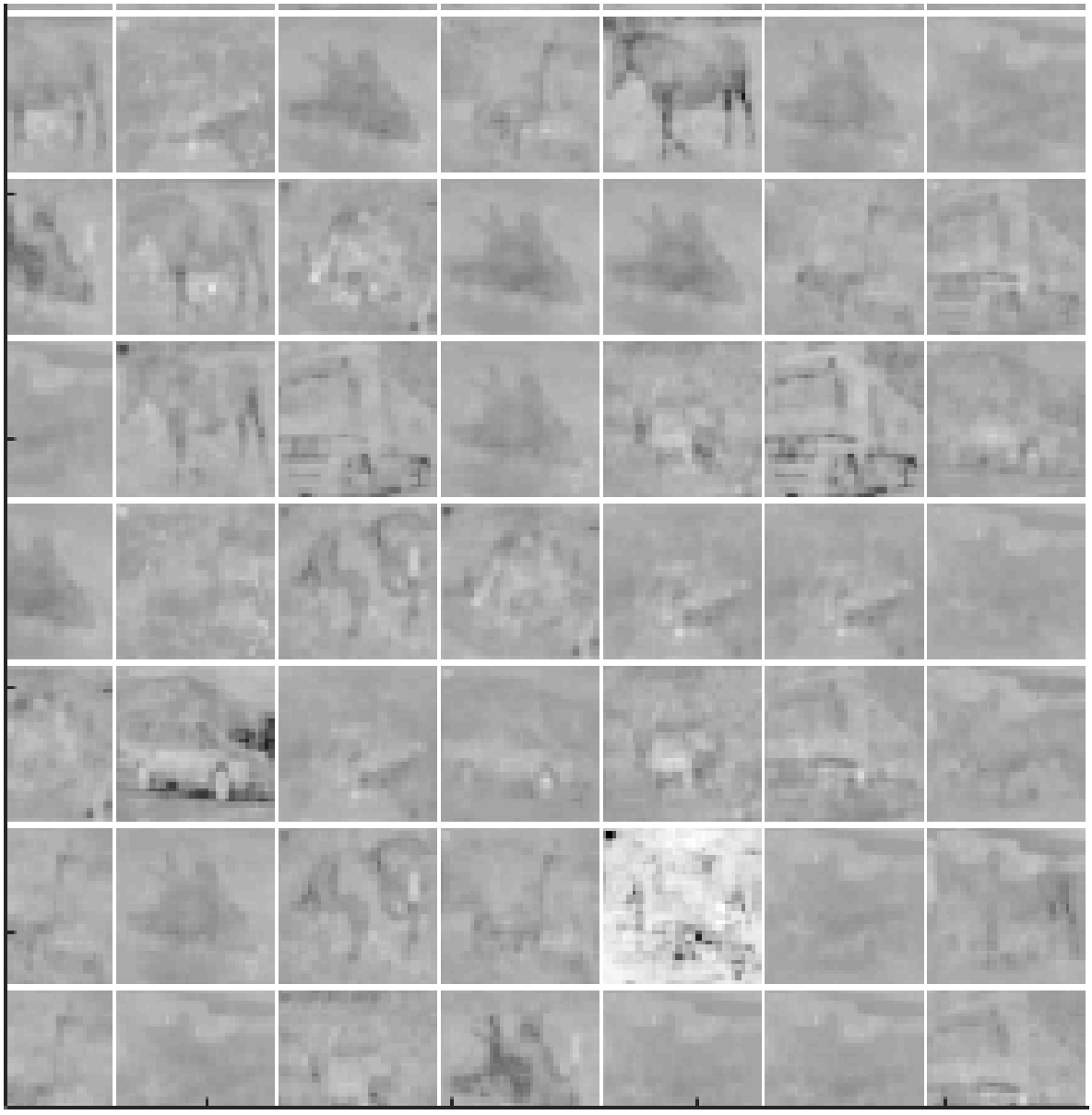

Figure 6.10: 1-shot CIFAR10 weight filter

7 Conclusion and Future Work

As discussed in chapter 3 the key objective for this thesis are :

- Design a methodology to get weight matrix for SNNs.
- Methodology should get this weight matrix with as small possible dataset.
- Methodology should be generalised enough so that irrespective of learning method used(supervised or unsupervised) it gives the weight matrix.

7.1 Conclusion

The methodology proposed in this thesis satisfies the objectives of this report, the proposed state of art methodology is capable of providing weight matrix for SNN classification with sufficiently high accuracy with as less as possible images in a dataset.

- The weight matrix obtained by the methodology is compatible with In-house Simulator, thus weights obtained by proposed methodology can be directly ported to In-house Simulator for SNN classification, however there was a scaling issue for R and C values which can be resolved in future either by scaling the proposed methodology or weight normalisationsection 7.3.
- The state of art proposed methodology flow is much better in comparison to conventional ANN-SNN conversion, as it is based on Siegert Neurons which have same transfer function as that of a LIF neuron thus training the network with more granularity because now programmer can train the network with specific neuron parameters, which was not possible with earlier approaches .
- The proposed methodology incorporates a modified CD learning which is based on the distance(degree of dissimilarity), it showed maximum increase 9.98% for one image per class. The modified CD especially showed increase in accuracy with smaller dataset.
- It also incorporates N-shot learning Table 6.1 i.e can do learning from as small as possible dataset ideally, one image per class, with MNIST which has 10000 images per class as standard : for 100 images per class it showed 85.43%, 50 images it was 79.24%, with 10 images it was 71.68% and with 1 image it was 37.53%.
- When 1 image per class having 10000 epoch(accuracy = 37.62%) and 10 images per class having 1000 epochs (accuracy = 71.68%)having same number of computations were done the 10 images per class gave higher accuracy**??**.
- The proposed methodology is compatible for both supervised and unsupervised learning in SNN i.e it can work irrespective of labeled or unlabeled dataset.

- The Proposed methodology is flexible i.e it also works with other datasets. With CIFAR 10 dataset Table 6.6 having very complex features it is very difficult to get sufficiently higher accuracy, with 1000 images per class(accuracy = 53.87%) and 500 images per class(accuracy = 42.14%) as compared to state of art standard 65% for full CIFAR 10 dataset in ANN.
- The proposed methodology also gives SNN classification accuracy of 65.75% for neuron parameters relevant in hardware implementations for MNIST.

7.2 Contributions

- A generalised training methodology flow for SNN is proposed that gives weight matrix and compatible with in-house simulator.
- A novel one of its kind methodology gives sufficiently high classification accuracy with small dataset(ideally with 1 images per class).
- Methodology is so generalised that irrespective of learning method (Supervised or unsupervised) used in SNN it can provide the weight matrix.
- A unique distance controlled modified CD algorithm, where distance works as a Hyper Parameter and strongly controls the learning.

7.3 Future Work

- Resolve the scaling issue of R and C values, it can be done by using weight normalisation(14) scheme to re-scale weights and avoid other re-scaling errors.
- As of now proposed methodology is a offline training methodology i.e weight matrix is obtained separately and then transferred to SNN, thus make methodology online so that it can be implemented on hardware itself along with SNN if needed.

Bibliography

[gib] Gibbs Sampling. https://towardsdatascience.com/deep-learning-meets-physics-restricted-boltzmann-machines-part-i-6df5c4918c15. Accessed: 2019-09-07.

[DS] ML with small dataset. https://towardsdatascience.com/breaking-the-curse-of-small-datasets-in-machine-learning-part-1-36f28b0c044d. Accessed: 2018-04-19.

[NN] Neural Networks. https://en.wikipedia.org/wiki/Artificial_neural_network. Accessed: 2018-12-09.

[pca] PCA LDA Difference. https://algorithmsdatascience.quora.com/PCA-4-LDA-Linear-Discriminant-Analysis. Accessed: 2019-09-09.

[5] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[6] Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Filho, W. J., Lent, R., and Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541.

[7] Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., Choo, X., Voelker, A., and Eliasmith, C. (2014). Nengo: a python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, 7:48.

[8] Beyeler, M., Carlson, K. D., Chou, T.-S., Dutt, N., and Krichmar, J. L. (2015). Carlsim 3: A user-friendly and highly optimized library for the creation of neurobiologically detailed spiking neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[9] Bohte, S. M., Kok, J. N., and La Poutré, J. A. (2000). Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, pages 419–424.

[10] Booij, O. and tat Nguyen, H. (2005). A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters*, 95(6):552–558.

[11] Campbell, N. A., Mitchell, L. G., Reece, J. B., and Bishop, J. (1997). *Biology: concepts & connections*. Number QH308. 2 C35 1996. Benjamin Cummings Menlo Park, Calif.

[12] Carnevale, N. T. and Hines, M. L. (2006). *The NEURON book*. Cambridge University Press.

[13] Cornelis, H., Rodriguez, A. L., Coop, A. D., and Bower, J. M. (2012). Python as a federation tool for genesis 3.0. *PLoS One*, 7(1):e29018.

[14] Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[15] Fidjeland, A. K., Roesch, E. B., Shanahan, M. P., and Luk, W. (2009). Nemo: a platform for neural modelling of spiking neurons using gpus. In *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, pages 137–144. IEEE.

[16] Florian, R. V. (2012). The chronotron: a neuron that learns to fire temporally precise spike patterns. *PloS one*, 7(8):e40233.

[17] Gerstner, W. and Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

[18] Gewaltig, M.-O. and Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, 2(4):1430.

[19] Goodman, D. F. and Brette, R. (2009). The brian simulator. *Frontiers in neuroscience*, 3:26.

[20] Goodman, D. F., Stimberg, M., Yger, P., and Brette, R. (2014). Brian 2: neural simulations on a variety of computational hardware. *BMC neuroscience*, 15(1):P199.

[21] Gütig, R. and Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience*, 9(3):420.

[22] Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., and Kozma, R. (2018). Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics*, 12:89.

[23] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[24] Hinton, G. E., Sejnowski, T. J., et al. (1986). Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2.

[25] Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544.

[26] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

[27] Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572.

[28] Jolliffe, I. (2011). *Principal component analysis*. Springer.

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

[30] Kasabov, N. K. (2018). *Time-space, spiking neural networks and brain-inspired artificial intelligence*, volume 7. Springer.

[31] Kreutz-Delgado, K. (2015). Mean time-to-fire for the noisy lif neuron-a detailed derivation of the siegert formula. *arXiv preprint arXiv:1501.04032*.

[32] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

[Lapicque] Lapicque, L. Quantitative research on electrical excitation of nerves treated as a polarization. *Journal of Physiology and General Pathology*, 9.

[34] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103.

[35] Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., and Kluger, Y. (2017). Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv preprint arXiv:1712.09005*.

[36] Liu, Q. (2018). *Deep spiking neural networks*. PhD thesis, The University of Manchester (United Kingdom).

[37] Liu, Q., Patterson, C., Furber, S., Huang, Z., Hou, Y., and Zhang, H. (2013). Modeling populations of spiking neurons for fine timing sound localization. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[38] Maass, W. (1998). On the role of time and space in neural computation. In *International Symposium on Mathematical Foundations of Computer Science*, pages 72–83. Springer.

[39] Marieb, E. N. and Hoehn, K. (2007). *Human anatomy & physiology*. Pearson Education.

[40] Mohemmed, A., Schliebs, S., Matsuda, S., and Kasabov, N. (2012). Span: Spike pattern association neuron for learning spatio-temporal spike patterns. *International journal of neural systems*, 22(04):1250012.

[41] Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., and Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience*, 7:272.

[42] O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience*, 7:178.

[43] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[44] Ponulak, F. and Kasiński, A. (2010). Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. *Neural computation*, 22(2):467–510.

[45] R, S. M. (2018). Study of deep learning based perception for autonomous vehicles. Technical report, TU Delft.

[46] Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11.

[47] Stewart, T. C. (2012). A technical overview of the neural engineering framework. *University of Waterloo*.

[48] Stimberg, M., Goodman, D. F., Benichoux, V., and Brette, R. (2014). Equation-oriented specification of neural models for simulations. *Frontiers in neuroinformatics*, 8:6.

[49] Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2018). Deep learning in spiking neural networks. *Neural Networks*.

[50] Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM.

[51] Tieleman, T. and Hinton, G. (2009). Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1033–1040. ACM.

[52] Vitay, J., Dinkelbach, H. Ü., and Hamker, F. H. (2015). Annarchy: a code generation approach to neural simulations on parallel hardware. *Frontiers in neuroinformatics*, 9:19.

[53] Ye, J., Janardan, R., and Li, Q. (2005). Two-dimensional linear discriminant analysis. In *Advances in neural information processing systems*, pages 1569–1576.