NEURO-INSPIRED COMPUTING ENHANCED BY SCALABLE ALGORITHMS AND PHYSICS OF EMERGING NANOSCALE RESISTIVE DEVICES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Parami Wijesinghe

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Prof. Kaushik Roy, Chair

School of Electrical and Computer Engineering

Prof. Anand Raghunathan

School of Electrical and Computer Engineering

Prof. Saeed Mohammadi

School of Electrical and Computer Engineering

Prof. Vijay Raghunathan

School of Electrical and Computer Engineering

Approved by:

Prof. Dimitrios Peroulis

Head of the School Graduate Program

To the three amazing individuals who had significant impacts in my life; my father, my husband, and my PhD advisor

ACKNOWLEDGMENTS

I would like to extend my sincere gratitude towards my PhD advisor, Professor Kaushik Roy "The Legend", for being an amazing mentor, a great visionary and a caring friend. It has always amazed me how he has paid attention to each and every student in a large group of 34. "If it is something interesting, I always have time"; he used to smile and say. I highly appreciate his guidance and immense patience through out the years. It has been a great honor working with him. Even though few lines would not do justice in explaining how amazing a mentor he is, I have to stop here since the instructions state "write a brief statement".

I would also like to thank my PhD committee members, Prof. Anand Raghunathan, Prof. Saeed Mohammadi and Prof. Vijay Raghunthan, for being in my committee and for providing constructive comments and insights. They were very supportive throughout the exam scheduling process and it could not have been any better without them. They were all marvelous teachers and very kind hearted individuals. It has always amazed me how they teach so fast but students still get the content of the subjects, and I would love to learn that magic one day. It was truly an honor having them in my committee.

I would love to thank my colleague, lab mate, and loving husband, Chamika Liyanagedera for always being my shadow. It was just amazing having someone working in the same field, someone who is smarter, under the same roof. Thank you for all the encouragement, valuable discussions and most of all, patience throughout my PhD studies, and beyond.

I would also love to remind my beloved father, the most matured person I have met until today, for all the life lessons he has given, and for believing in me. He has molded me to an independent individual to face the world with confidence. I would take the time to utter his favorite question "So when will you graduate?". *Appachchi*,

if you are watching from somewhere,"I am done finally. Just wish you were here to celebrate".

I do not forget all the co-authors, professor Abhronil Sengupta, Dr. Akhilesh Jaiswal, Gopalakrishnan Srinivasan, Dr. Syed Sarwar, Dr. Kon-Woo Kwon, professor Xuanyao Fong, Dr. Yusung Kim, Aayush Ankit and Priyadarshini Panda. Your ideas and feedback were invaluable for my research work. It is my pleasure to thank my mentors Dr. A. Kanuparthy, H. Khatri at Intel, OR, USA, and Dr. F. Ahmed at Qualcomm, CA, USA for spending your time mentoring me and giving a flavor of the industrial projects. For all the wonderful moments and laughs we shared during my career, I would like to thank Rwitti Roy and the current and past NRL members.

I would like to dedicate few lines to acknowledge the funding support I received for my projects by the Center for Brain Inspired Computing (C-BRIC) - a Joint University Microelectronics program (JUMP) center sponsored by DARPA, the Semiconductor Research Corporation (SRC), the National Science Foundation (NSF), Intel Corporation, the DoD Vannevar Bush Fellowship, Center for Spintronic Materials, Interfaces, and Novel Architectures (C-SPIN), a MARCO and DARPA sponsored StarNet center, the U.S. Army Research Laboratory and the U.K. Ministry of Defense. Furthermore, I am extremely grateful for the support I received from the ECE graduate student office. I would like to especially thank Matt Golden for addressing thousands of concerns I had, with patience.

I am indebted to my mom and sisters for always being by my side while encouraging me to go forward. I would also like to show gratitude to my family and friends who made my life more cheerful and eventful.

Last but not least, I would like to thank 'YOU', who took the trouble to read *this*, when there are so many important things to read in the following pages.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

ABBREVIATIONS

ANN	Artificial Neural Network
AP	Anti Parallel
AP	Approximation Property
CD	Clustered training space division
CeNN	Cellular Neural Network
CMOS	Complementary Metal Oxide Semiconductor
CNF	Conjunctive Normal Form
CNN	Convolutional Neural Network
DR	Discriminant Ratio
ECM	Electrochemical Metallization
FDR	Fisher's Discriminant Ratio
FL	Free Layer
GFLOPS	Giga Floating Point Operations Per Second
GPU	Graphical processing units
GRN	Gene Regulation Network
HM	Heavy Metal
IMA	In Plane Magnetic Anisotropy
LDA	Approximation Property
LLGS	Landau - Lifshitz - Gilbert - Slonczewski
LPCVD	Low pressure chemical vapor deposition
LSM	Liquid State Machine
LSTM	Long Short Term Memory
MIM	Metal Insulator Metal
MLMR	Multiple Liquid-Multiple Readouts

MLSR	Multiple Liquid-Single Readout
MTJ	Magnetic Tunnel Junction
NP	Non-Deterministic Polynomial
P	Parallel
PC	Principal Components
PCA	Principal Component Analysis
PECVD	Plasma-enhanced chemical vapor deposition
PL	Pinned Layer
PMA	Perpendicular Magnetic Anisotropy
RD	Random training space Division
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
SAT	Satisfiability
SHE	Spin Hall Effect
SNN	Spiking Neural Network
SOT	Spin Orbit Torque
SP	Separation Property
STDP	Spike Time Dependent Plasticity
STT	Spin Transfer Torque
TMR	Tunneling Magneto-Resistance

ABSTRACT

Wijesinghe, Parami Ph.D., Purdue University, August 2019. Neuro-inspired computing enhanced by scalable algorithms and physics of emerging nanoscale resistive devices. Major Professor: Roy K. Professor.

Deep 'Analog Artificial Neural Networks' (AANNs) perform complex classification problems with high accuracy. However, they rely on humongous amount of power to perform the calculations, veiling the accuracy benefits. The biological brain on the other hand is significantly more powerful than such networks and consumes orders of magnitude less power, indicating some conceptual mismatch. Given that the biological neurons are locally connected, communicate using energy efficient trains of spikes, and the behavior is non-deterministic, incorporating these effects in Artificial Neural Networks (ANNs) may drive us few steps towards more realistic neural networks.

Emerging devices can offer a plethora of benefits including power efficiency, faster operation, and low area, in a vast array of applications. For example, memristors and Magnetic Tunnel Junctions (MTJs) are suitable for high density, non-volatile Random Access Memories when compared with CMOS implementations. In this work, we analyze the possibility of harnessing the characteristics of such emerging devices, to achieve neuro-inspired solutions to intricate problems.

We propose how the inherent stochasticity of nano-scale resistive devices can be utilized to realize the functionality of spiking neurons and synapses, that can be incorporated in deep stochastic Spiking Neural Networks (SNN) for image classification problems. While ANNs mainly dwell in the aforementioned classification problem solving domain, they can be adapted for a variety of other applications. One such neuro-inspired solution is the Cellular Neural Network (CeNN) based Boolean satisfiability solver. Boolean satisfiability (k-SAT) is an NP-complete ($k \geq 3$) problem that

constitute one of the hardest classes of constraint satisfaction problems. We provide a proof of concept hardware based analog k-SAT solver that is built using MTJs. The inherent physics of MTJs, enhanced by device level modifications, is harnessed here to emulate the intricate dynamics of an analog, CeNN based, satisfiability (SAT) solver.

Furthermore, in the effort of reaching human level performance in terms of accuracy, increasing the complexity and size of ANNs is crucial. Efficient algorithms for evaluating neural network performance is of significant importance to improve the scalability of networks, in addition to designing hardware accelerators. We propose a scalable approach for evaluating Liquid State Machines: a bio-inspired computing model where the inputs are sparsely connected to a randomly interlinked reservoir (or liquid). It has been shown that biological neurons are more likely to be connected to other neurons in the close proximity, and tend to be disconnected as the neurons are spatially far apart. Inspired by this, we propose a group of locally connected neuron reservoirs, or an ensemble of liquids approach, for LSMs. We analyze how the segmentation of a single large liquid to create an ensemble of multiple smaller liquids affects the latency and accuracy of an LSM. Our results illustrate that the ensemble approach enhances class discrimination (quantified as the ratio between the Separation Property (SP) and Approximation Property (AP)), leading to improved accuracy in speech and image recognition tasks, when compared with a single large liquid. Furthermore, we obtain performance benefits in terms of improved inference time and reduced memory requirements, due to lower number of connections and the freedom to parallelize the liquid evaluation process.

1. INTRODUCTION

The transistor invented about 7 decades ago has been subjected to substantial scaling, allowing more functionality into a given area. It is now possible to integrate enormous amount of functions including audio-visual communication, video recording, graphical processing, web browsing, etc. to a pocket sized area, in to which only a radio with 4 transistor amplifier could fit, in 1950's. Since then over the years, the number of transistors inside a chip approximately doubled every year till now, validating the empirical observation [1] of Gordon E. Moore. The vigorous scaling of transistors will eventually hit the atomic limitations leaving further validity of Moore's law in jeopardy. To continue further scaling, novel devices have emerged as promising candidates for a plethora of applications. For example, memristors and Magnetic Tunnel Junctions (MTJs) are suitable for high density, non-volatile Random Access Memories when compared with CMOS implementations. Representing a complex functionality that requires a significant number of devices with a single or less number of alternative devices is as important as scaling the size of a single device.

Along with the advancements in the device domain, a significant improvement in the algorithms sector is also visible. More research is focused on the brain inspired computing that allows massive parallelism along with the flexibility to 'learn from outcomes in the past', in contrast to the traditional Von-Neuman computing. The brain inspired computing models can be extended to vast array of applications even though it mainly dwells in the classification problem solving domain. Up to date, deep artificial neural networks (ANNs) have given the best performance with respect to classification accuracy. For an example, SENet that won the 2017 ILSVRC, is a deep Convolutional Neural Network (CNN) with the reported lowest top 5 error (the correct class is not within the top 5 selection of classes according to the network output) of 2.251% on ImageNet data set [2]. However, such networks require huge

power and time if the algorithms are implemented in software in a computer. For an instance, SE-ResNet requires $\sim$ 3.2GFLOPS (number of operations per second) [2]. It has been shown experimentally that by conserving energy via spike based operation, the brain has evolved to achieve its prodigious signal-processing capabilities using orders of magnitude less power than the state-of-the-art supercomputers. Therefore, with the intention to pave pathways to low power neuromorphic computing, much consideration is given to realistic artificial brain modeling. Furthermore, the need for scalable algorithms/architectures also arise to improve the speed of computations in large scale ANNs.

Given that the actual communication inside the brain transpires using streams of non-deterministic spikes [3], [4], [5], recently more attention is focused on introducing stochasticity to traditional neuro-inspired computations. However, implementing randomness using CMOS devices is complex and thus requires a significant amount of resources such as area and power [6]. Even though the stochastic switching behavior of emerging resistive devices has shown to be unfavorable in applications such as memory, it can be embraced to efficiently represent features in neuromorphic computing. We propose a stochastic deep spiking neural network, of which both the synaptic and neuron functionality are represented by memristors. It was observed that the proposed all-memristor based hardware design offers lower power consumption, and lower area and delay product, when compared with the CMOS based design, for an image recognition application.

It has also been shown that the biological neurons are arranged in locally connected clusters [7]. Inspired by the above, we propose an ensemble approach for liquid state machines (LSMs). Traditional LSMs have a set of randomly interlinked spiking neurons, which is known as the liquid. In the effort of trying to reach human accuracy levels, increasing the size and complexity of the networks tend to be an essential requirement. Larger and complex networks are sluggish and power hungry. In order to address this, in the ensemble approach, we split a large liquid in to an ensemble of smaller liquids, and evaluate them in parallel. The class discrimination capability

of the liquid improves in the proposed ensemble approach, when compared with the traditional LSM architecture. Owing to the improved class discrimination property, the proposed approach gives better accuracy. Furthermore, the liquid ensemble has less number of connections and hence deliver lower amount of computations and lower memory requirement than the conventional LSM approach with the same number of neurons. The smaller liquids could also be evaluated in parallel, leading to faster results.

Another feature of biological neurons is that they tend to get activated in parallel and asynchronously [8]. Similarly, the cells in a Cellular Neural Network (CeNN) operate in parallel, in contrast to conventional fully connected ANNs, and hence are faster. The fast operation of CeNNs are mainly used for large scale image processing applications. Inspired by the CeNN architecture, an analog Boolean Satisfiability (SAT) solver has been proposed [9]. The approach uses the benefits of seeking a solution in parallel, in contrast to the current state of the art SAT solvers that use sequential variable assigning and backtracking. The inherent physics of magnetic tunnel junctions can effectively be harnessed to emulate such a system. With certain device level modifications, the magnetic tunnel junctions (MTJs) can mimic the dynamics of the proposed analog SAT solver. Implementing the system using CMOS devices alone can be complex and thus potentially take longer delays and consume significant amounts of power. Furthermore, it has been explained that addition of stochasticity will reduce the transient time of convergence to a solution, in the analog system [9]. The inherent stochasticity present in MTJ switching due to thermal noise is thus beneficial when implementing the system.

The next few subsections will give an introduction to the types of neural networks, and the nanoscale resistive devices of interest in this work.

1.1 Neuro-Inspired architectures

1.1.1 Spiking neural networks

Even though the exact mechanisms of communication between biological neurons still remain unknown, it has been shown experimentally that neurons use spikes for communication and the nature of the neuron activation is non-deterministic [3], [4], [5]. Brain uses orders of magnitude less power than the state-of-the-art supercomputers via spike based operation [10]. The inception of the Spiking Neural Networks (SNN) concept is a consequence of the above.

Spiking neural networks consist of an array of (leaky) integrate and fire neurons. The neurons will increase/decrease their membrane potentials depending upon the incoming spikes, strengthened or weakened by the synaptic weights. Once the neuron's membrane potential goes above its threshold, it will generate a spike and then return to its reset potential. A typical Spiking neural network structure is as shown in figure 1.1. The excitatory layer neurons receive input spike trains, and generate output spike trains. The inhibitory layer neurons help in suppressing the spiking activity of the excitatory neurons that did not fire. The work presented here does not use integrate and fire neurons even though the communication is carried out via spikes. The neurons take a stochastic form. *i.e.*, depending upon the strength of the total spikes applied on the neuron during a specific time step, it will fire with a certain probability which is a function of the applied input. Once a neuron fires, it will be reset to receive an input during the next time step. The system is synchronous.

SNNs use both unsupervised (Spike Time Dependent Plasticity (STDP) [11] based) and supervised (backpropagation) training. The work presented here uses a supervised learning algorithm and the network is trained as an ANN using stochastic gradient descent based backward propagation method. The network is then converted to an SNN.

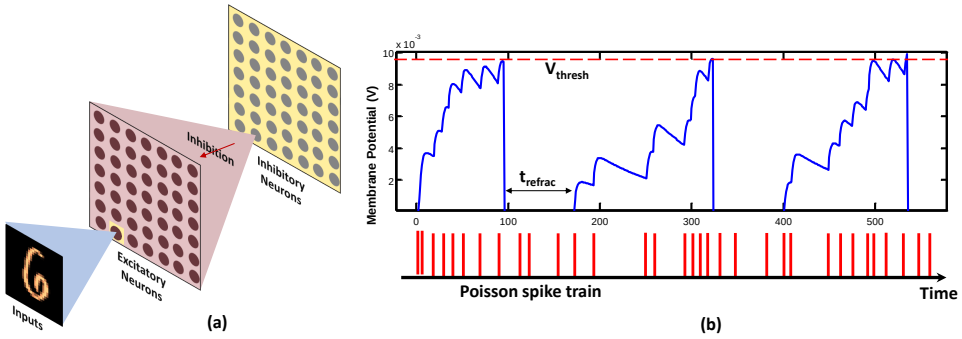

Fig. 1.1. (a) The three layers of a typical spiking neural network; Input layer, Excitatory layer, Inhibitory layer. (b) The dynamics of a single leaky integrate spiking neuron to an incoming spike train (from an excitatory neuron). When the membrane potential goes above V_{thresh}, the neuron generates an output spike and it remains idle for incoming spikes for a duration of t_{refrac}.

1.1.2 Cellular neural networks

The inception of Cellular Neural Networks goes back to 1988 [12]. It is a large scale nonlinear analog computational model that is initially used for image processing applications. It can also be used to solve a set of partial differential equations [13]. Due to the massive parallelism present in CeNN platforms [12], [14] they are faster than conventional digital computation methods that operate in a serial fashion [15]. CeNNs are made of massive aggregate of regularly spaced circuit clones called cells that communicate with each other directly through its neighbors. This 'cell' is the basic element in a CeNN, and it contains linear and non-linear circuit elements such as linear capacitors, linear resistors and linear and non-linear controlled and independent sources. Each cell can be considered as a neuron and is connected to its neighboring cells. Cells that are not directly connected, communicate indirectly via other cells.

Figure 1.2 (a) shows the neighborhood to which a single cell is connected. The neighborhood of r has a total of $(2r+1)^2 - 1$ amount of cells connected to a single cell. Figure 1.2 (b) illustrates the first defined internal circuit architecture of a single cell in a CeNN by L. Chua. It contains one independent voltage source E_{ij}, one

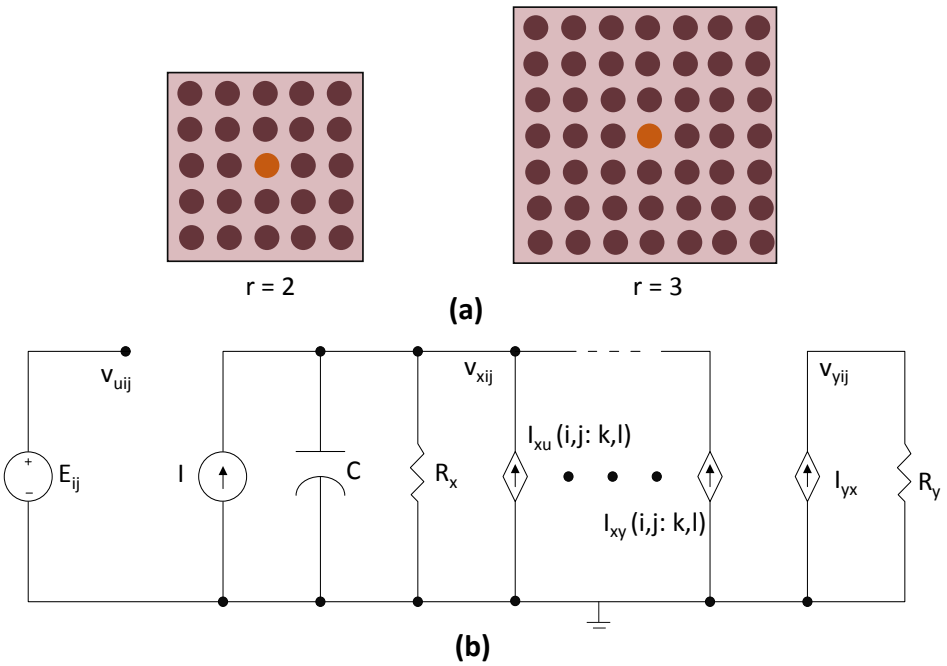

Fig. 1.2. (a) The neighborhood of a cell C_{ij} for $r = 2$ and $r = 3$ in a cellular neural network (b) The internal circuit schematic of a single cell C_{ij} in a cellular neural network. Neighboring cells are C_{kl}.

independent current source I, one capacitor C, two linear resistors R_x, and R_y and current sources, linearly controlled by the neighboring cells' input voltages v_{ukl}, and the feedback from the output voltage v_{ykl}. The cell of interest is denoted as C_{ij}, and the neighboring cells are shown as C_{kl}. The nonlinear element in each cell is a piece-wise-linear voltage-controlled current source $I_{xy} = (\frac{1}{R_y})f(v_{xij})$, where $f(.)$ is the characteristic function.

Implementing CeNN computers is an ongoing research and both analog and digital type CeNNs are available [14] even though the latter is not as beneficial as the analog counterpart in terms of speed. Despite their exceptional performance in image processing applications, CeNNs suffer from noise effects and often lack precision. In this work we present a CeNN implemented using Magnetic Tunnel Junctions, of which the inherent thermal noise is beneficial for a Boolean Satisfiability Solver.

1.1.3 Liquid State Machines (LSM)

Liquid State Machines (LSMs) [16] fall under the spiking neural network category. However, their architecture is different from a traditional SNN since it is a recurrent neural network (RNN) and mainly used for spatio-temporal data related tasks. An LSM consists of a set of inputs sparsely connected to a randomly and recurrently interlinked pool of spiking neurons called the 'liquid'. The liquid is connected to an output classifier, which can be trained using standard methods such as Spike Timing Dependent Plasticity, backpropagation, delta rule *etc.* [17]. LSMs have been used for a variety of applications including robot control [18], sequence generation [19], decoding actual brain activity [20], action recognition [21], speech recognition [16, 22–26], and image recognition [25, 27–29].

LSMs gained their popularity due to two main reasons. First, the LSM architecture is neuro-inspired. As explained previously, biological neurons communicate via trains of spikes [30, 31]. Furthermore, the gene regulation network (GRN) of the Bacterium 'Escherichia Coli' (E-Coli) was experimentally assessed and shown to behave similar to an LSM [32]. The E. Coli has the capacity for perceptual categorization, especially for discrimination between complex temporal patterns of chemical inputs. Second, LSMs have simple structure and lower training complexity among other RNNs. The claim is that, sufficiently large and complex liquids inherently possess large computational power for real-time computing. Therefore, it is not necessary to "construct" circuits to achieve substantial computational power.

1.2 Emerging nano-scale resistive devices

1.2.1 Magnetic Tunnel Junctions

A magnetic tunnel junction (MTJ) is a thin tunneling oxide layer (MgO) sandwiched between two ferromagnetic layers. One layer is called the pinned layer (PL) which is magnetically hardened to have a fixed magnetization direction in order to

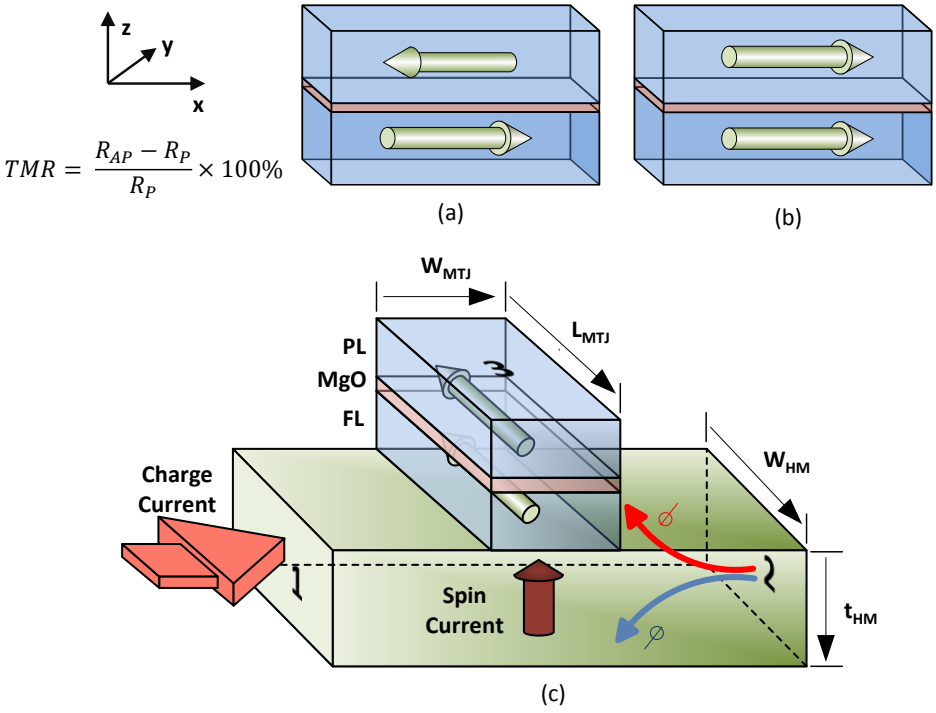

Fig. 1.3. The Heavy Metal-Magnetic Tunnel Junction structure. (a) High resistive (R_{AP}) anti-parallel state of an MTJ (b) Low resistive (R_P) parallel state of an MTJ. The Tunnel Magneto-Resistance (TMR) is a measure of the normalized difference of these resistances. Typical values of the TMR ranges from 150%-600% [33, 34]. (c) An HM-MTJ structure. The charge current through the HM layer underneath the MTJ, gets split into up and down spins, inducing a perpendicular spin current which can reverse the magnetization of the free layer through the Spin Orbit Torque phenomenon.

act as a reference layer. The other ferromagnetic layer is known as the free layer (FL) and the magnetization of this layer can be switched by a charge current going through the device. MTJs can be classified in to two main categories depending upon the stable direction of magnetization of the FL. If the direction is perpendicular to the FL, then the device is known as a Perpendicular magnetic anisotropy (PMA) MTJ. If the stable direction is parallel to the plane of the FL, it is known as an In Plane Magnetic Anisotropy (IMA) MTJ. This work is focused on IMA MTJ device and the term MTJ refers to this type of device throughout this section.

The phenomenon of switching the FL magnetization by sending a current through the device (MTJ) is known as Spin Transfer Torque induced switching. Another mechanism of switching the FL magnetization direction is by means of a charge current passing through a Heavy Metal (HM) underlayer. This phenomenon is known as the spin orbit torque and appeared to be more energy efficient than using STT when switching an IMA nano-magnet. Due to the large spin-orbit-coupling exhibited by heavy metals (like Ta, Pt *etc.* [35]), the charge current flowing through the HM layer gets split into up and down spins, as shown in Figure 1.3. The spin splitting creates a spin gradient and results in flow of a spin current perpendicular to the direction of the charge current. This spin current incident on the free layer of the MTJ, can switch the magnetization of the free layer through the aforementioned STT phenomenon. Thus, a current flowing between terminals 1 and 2 can switch the orientation of the free layer magnet. If the magnetizations of the two layers are oriented in the same direction, the MTJ is said to be in the parallel state and exhibits a low resistance across the device. If the two magnetizations are pointed in the opposite directions (antiparallel state) the MTJ exhibits a high resistance across the device. The current state of the MTJ can be read by sensing the resistance offered by the MTJ measured perpendicular to the plane of the ferromagnets, between terminals 3 and 1/2. The tunnel magneto-resistance (TMR) of the MTJ is a measure of the normalized difference between the resistances of the anti-parallel and the parallel state. This difference in resistance can be easily converted into an output low or high voltage using an inverter configuration for memory applications. Further, due to the nature of the STT phenomenon and the nano-scale sizes of the ferromagnets, thermal noise plays an important role in the switching process of the MTJs. As a result, for a constant input current and for a given switching time, the MTJ switching mechanism exhibits stochastic behavior [36]. The switching probability of the MTJ can be controlled by the amount of current flowing though the HM layer. Following are some favorable features exhibited by the HM-MTJ structures

Table 1.1.
Device simulation parameters

Parameters	Values
Free layer area, A_{MTJ}	20×40 nm^2
Free layer thickness, t_{ss}	1.72nm
Gilbert's damping factor, α	0.02
Saturation Magnetization, M_S	1257.3 KA/m
Interface Anisotropy, K_i	1.3 ergs/cm^2
Spin hall angle, θ_{SH}	0.3
Heavy metal layer thickness	3 nm
Heavy metal layer resistance	50Ω
Oxide thickness, t_{ox}	1.5nm
Tunnel Magneto-Resistance, (TMR)	180%
Switching current, (full switch, $AP \rightarrow P$, 20ns)	17μA

- The three terminal nature of the SHE-MTJ allows independent optimization of the read and write paths resulting in low write currents [37]
- Since the HM resistance remains a constant, (in contrast to the MTJ device itself) the write speed can be controlled by changing the voltage applied across the HM layer
- The efficiency of spin current generated due to the spin-orbit-coupling phenomenon has been shown to be>1 [35] (with proper dimensions), resulting in low write-current requirements.

The magnetization dynamics of the free layer is given by the Landau-Lifshitz-Gilbert-Slonczewski equation and can be written as [38]

$$\frac{(1+\alpha^2)}{|\gamma|}\frac{\partial \widehat{m}}{\partial t} = -\widehat{m} \times \overrightarrow{H}_{EFF} - \alpha \widehat{m} \times \widehat{m} \times \overrightarrow{H}_{EFF} + \frac{1}{|\gamma|}(\alpha \widehat{m} \times \overrightarrow{STT} + \overrightarrow{STT}) \quad (1.1)$$

$$\widehat{m} = [m_x \widehat{x} \quad m_y \widehat{y} \quad m_z \widehat{z}] \quad (1.2)$$

where α is the Gilbert damping constant, γ is the gyro-magnetic ratio, $\widehat{m}$ is the unit vector in the direction of the magnetization, t is the continuous time, and $\overrightarrow{H}_{EFF}$ is the effective magnetic field including the demagnetization field and the interface anisotropy field. $(\overrightarrow{STT})$ is the spin transfer torque term given by

$$\overrightarrow{STT} = |\gamma|\beta\Big(\widehat{m} \times (\epsilon \widehat{m}) \times \widehat{p}\Big), \quad \beta = \frac{\hbar J_s}{2q\mu_0 M_s t_{FL}} \quad (1.3)$$

where q is the charge of an electron, μ_0 is vacuum permeability, $\hbar$ is the modified Planck's constant, t_{FL} is the thickness of FL and M_s is the saturation magnetization. J_s is the spin current density incident on the free layer of the MTJ and ϵ is the spin polarization efficiency. The relationship between this spin current and the charge current density is as shown below

$$\frac{I_S}{I_C} = \frac{A_{MTJ}\theta_{she}}{A_{HM}}\left(1 - sech\left(\frac{t_{HM}}{\lambda_{sf}}\right)\right), \quad (1.4)$$

For this work, all the material parameters were selected according to prior experimental work [39, 40]. These parameters are recorded in Table 1.1.

1.2.2 Memristors

Nanoscale resistive devices have been extensively studied as a leading candidate for non-volatile memory [41], reconfigurable logic [42], and analog circuits [43]. The possibility of using different types of memristors in different types of neural networks has also been explored. For example, the ability to change resistance due to voltage

pulses makes the memristor a better candidate for STDP learning [44]. Some memristors show unstable intermediate resistance states which are suitable as synapses to represent the short-term memory and long-term memory functionality [45].

The typical nano-scale resistive device is based on a metal-insulator-metal (MIM) structure. The resistance change in these devices can be attributed to the formation of a conductive filament inside the insulator (Ag, amorphous Si ($a-Si$), Pt based devices), change in the phase due to Joule heating and cooling (chalcogenide based devices), or field assisted drift/diffusion of ions (TiO_2 based devices). These processes

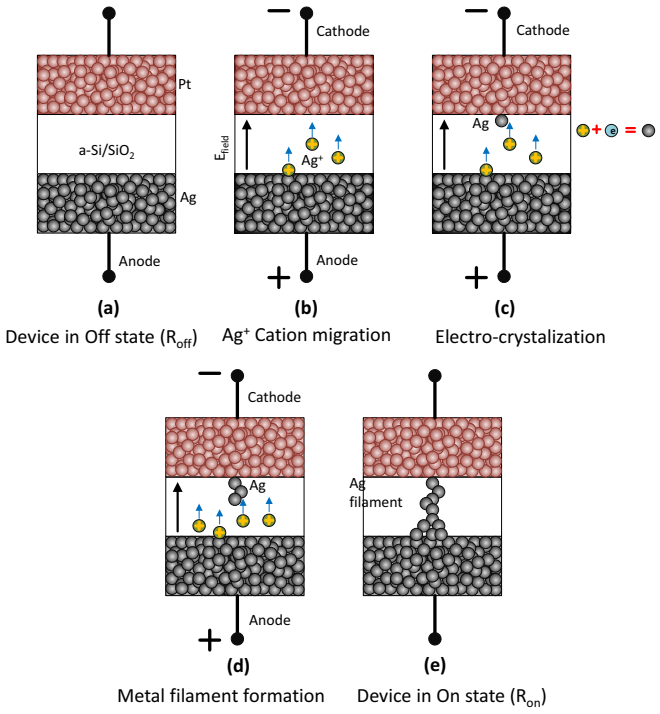

Fig. 1.4. The standard SET operation for an ECM type memristor (a)'Off' state of a memristor has higher resistance due to the sandwiched insulating material. (b) To 'SET' the device, positive voltage must be applied to the active electrode with respect to the inert electrode. Ag cations start traveling towards the inert electrode due to the E_{field}. (c),(d) The Ag^+ ions get combined with electrons and crystalize forming a metal filament. (e) Once a full metal filament is formed between the two electrodes, the resistance of the device lowers.

have shown to be random in nature. For this work, the $a-Si$ based metal filament formation devices (Electrochemical Metallization devices or ECM devices [46, 47]) were considered due to multiple reasons as explained below. However, it must be noted that this particular type of device was selected as an example for a memristor to show the applicability of it for the deep stochastic SNNs. A different type of a memristor can suit better in different contexts (network accuracy, power consumption *etc.*). For example, the HfO_x based devices have higher endurance and lower switching time [48]. $Ag/AgSiO_2$ devices reset after a certain time period eliminating the requirement for resetting [49].

$a-Si$ memristors typically have very high resistance ($\sigma \sim 3 \times 10^{-5}\Omega cm^{-1}$ at 310K). When power consumption is considered, it is better to have high resistances in memristive crossbars. It is also possible to adjust the lower resistance of the device to suit the constraints of the crossbar driving sources. This can be done by tuning the $a-Si$ growth conditions (R_{ON} can be varied from $\sim$ 100MΩ to 10kΩ) during the PECVD or LPCVD deposition processes [50]. The ON-OFF ratio of the device is high ($\sim 10^7$) as well [51]. Having a higher ratio implies the higher reliability of programming a single state under variations in multi-bit configurations [52]. This is also better in terms of sensing if a memristor has switched or not, in the context of a neuron. Unlike some types of memristors, the $a-Si$ memristors require high write voltages. For an example, the nanoporous SiO_x based devices have very low forming voltages ($\sim$ 1.4V) [53]. Having smaller operating voltages in memristive crossbars require sophisticated sensing mechanisms. This also signals a reliability concern for the nanoporous multi-bit SiO_x devices, while operating in a crossbar with larger driving voltages. In contrast, the $a-Si$ based memristor can operate at 2V reading voltages without disturbing the device resistance [50].

The ECM devices consist of an insulating membrane ($a-Si$, SiO_2, Al_2O_3) sandwiched between two active (Ag) and inert electrodes (Pt, Ti). When the device is in its higher resistance (R_{off}) state, it is considered as storing a logic '0' (in memory). When writing a '1' to the device (SET or 'turning on'), a positive voltage must be

applied to the active electrode with respect to the inert electrode. At this point, the active electrode dissolution transpires and cations from the active electrode start migrating towards the inert electrode where it gets electro-crystalized, forming a metal filament (termed as the 'forming process' [47]). Once a full metal filament has grown between the two electrodes, there is a sudden drop in resistance. The aforementioned process is graphically explained in Figure 1.4. The low resistance (R_{on}) stage of the device is assigned to logic '1'.

The possibility of the $a-Si$ (Ag based) devices to act both as a muti-bit storage and as a stochastic switch [51] is beneficial for this work since it can act both as a neuron and a synapse. As explained above, in an $a-Si$ based memristor, a metal filament is formed between the two contacts when going from OFF to ON state. The Ag particles goes to the defect sites inside $a-Si$ and creates this filament. Depending on the number of defects, the I-V characteristics of the device shows multiple abrupt jumps in currents [51]. Devices with lower lengths will have lower number of defects and are much suitable for binary switching applications (which is the neuron in our work). When programming for multi levels, current/voltage must be controlled properly. It has been experimentally shown the possibility to store 8 levels of resistances (3-bit storage) [51] using the same write voltage pulse and different series resistors (R_s) to control the current. Each R_s resulted in different final resistance values of the memristor. Given the fact that a 30nm device can store 3-bit levels, and the resistance of memristors are proportional to the device length, it can be fairly assumed that a 60nm device can store 4-bit levels. It has been shown that nanoporous SiO_x memristors can store up to 9-bits [53] per cell. Such devices can give higher accuracy if used in hardware ANNs to represent the synapses.

Even though the stochasticity is helpful to represent the functionality of a stochastic neuron, it may not be beneficial for the supervised learning scheme being explored in this work for multi-bit synapses. Due to stochasticity, programming using a single voltage pulse with the selected control resistor may not guarantee the device transferring to the expected resistance level. Furthermore, due to variations, the value

of resistance at each level may change. Therefore, the need for better memristor programming schemes arise. It has been experimentally shown that TiO_2 based filament type memristors can be programmed to have a required resistance (within 1% accuracy) using a novel programming scheme, despite the variations [54].

2. CONSTRAINT SATISFACTION ENABLED BY SPIN ORBIT TORQUE MAGNETIC TUNNEL JUNCTIONS

2.1 Boolean Satisfiability Problem

The Boolean satisfiability problem investigates whether there exists an assignment for the input variables that satisfies a given Boolean formula. k-SAT is widely utilized in many practical applications including automated planning [55], test pattern generation [56], hardware model checking [57], software program testing [58] and timing analysis [59]. k-SAT problems are NP-complete ($k \geq 3$) [60, 61]. *i.e.*, there are no known algorithms that can guarantee a solution for a SAT problem in polynomial time, making it extremely difficult to solve most satisfiability problems with reasonable computational resources. Numerous research efforts have been directed towards realizing improved SAT solvers [9, 62–66], since a polynomial time solution to k-SAT implies efficient solutions to a large number of hard optimization problems. The standard conjunctive normal form (CNF) of any Boolean k-SAT problem with N variables can be written as

$$\mathcal{F} = (x_1 \cup x_2 \cup \overline{x}_3) \cap (x_2 \cup \overline{x}_1 \cup x_5) \cap ... (\overline{x}_4 \cup \overline{x}_5 \cup x_3) \tag{2.1}$$

where $x_i \in \{0, 1\}$ is a variable and each clause is the disjunction (OR, $\cup$) of k ($k = 3$ in this case) such variables or their negation ($\bar{x}_i$). The propositional formula $\mathcal{F}$ is a conjunction (AND, $\cap$) of M number of such clauses. The hardness of a SAT problem can be measured as the ratio between the number of clauses and the variables, known as the constraint density α_c (section 2.2).

2.2 Analog Approaches for solving Boolean Satisfiability Problem

Analog computational approaches have recently demonstrated promising results in a diverse array of applications [67] including aforementioned constraint satisfaction [9, 62, 68]. A recent analog formulation of a k-SAT solver has demonstrated its potential on locating a solution for the Boolean satisfiability problem in polynomial continuous-time [62]. However, implementing this set of analog formulae using a digital computer will diminish the polynomial time benefits, due to varying computational complexities between different time steps. Also, a hardware implementation of this analog k-SAT solver [62] is not ideal [69], due to the exponential energy fluctuations in the system, which allowed the polynomial continuous time convergence in the first place. Consequently, a Cellular Neural Network (CeNN) based analog SAT solver with bounded variables was proposed [9], and it is more appealing for hardware implementations. Although this bounded system does not have polynomial time complexity, noise effects in the analog hardware can potentially reduce the long transient times [9] in the system, as we demonstrate in this work. The dynamics of this analog SAT solver, which is also the framework of our work, can be defined by the following set of equations [9].

$$\dot{s}_i(t) = \frac{ds_i(t)}{dt} = -s_i(t) + Af(s_i(t)) + \sum_m c_{mi} g(a_m(t)) \tag{2.2}$$

$$\dot{a}_m(t) = \frac{da_m(t)}{dt} = -\, a_m(t) + Bg(a_m(t)) - \sum_i c_{mi} f(s_i(t)) + 1 - k \tag{2.3}$$

Here the variable s_i represents the state of the i^{th} $(i = 1, 2, ..., N)$ Boolean variable (x_i) and a_m represents the "satisfiedness" of the m^{th} $(m = 1, 2, ..., M)$ clause of the Boolean function. C is the problem specific 'interconnection matrix' of size $M \times N$. The functions $f()$ and $g()$ are the thresholding functions applied on variables s_i and a_m, respectively, as follows (Figure 2.1 shows this graphically)

$$f(s_i) = \frac{1}{2}(|s_i + 1| - |s_i - 1|) \tag{2.4}$$

$$g(a_m) = \frac{1}{2}(1 + |a_m| - |1 - a_m|) \tag{2.5}$$

When the Boolean variable is true ($x_i = 1$), then the value of s_i after thresholding will be equal to 1 (*i.e.* $f(s_i) = 1$). When it is false ($x_i = 0$), then $f(s_i) = -1$. The vector $f(s)$ can be considered as a solution to the k-SAT problem when all the transient variations have stopped and the system is in a stable state. The variable a_m determines whether the m^{th} ($m = 1, 2, ...M$) clause is satisfied at a given moment depending upon the values of the s variables. When the value of variable a_m after thresholding is 0 (*i.e.* $g(a_m) = 0$), then the corresponding m^{th} clause is satisfied. It will not have any impact on the dynamics of the system afterwards. The system converges to a solution of the Boolean function (get stabilized), when vector $g(a)$ is zero. The parameters A and B ('self-coupling parameters') are constants and C is a matrix of size (M, N) that is unique to a given propositional formula $\mathcal{F}$. The elements of the c matrix can be defined as follows. C_m is the m^{th} clause.

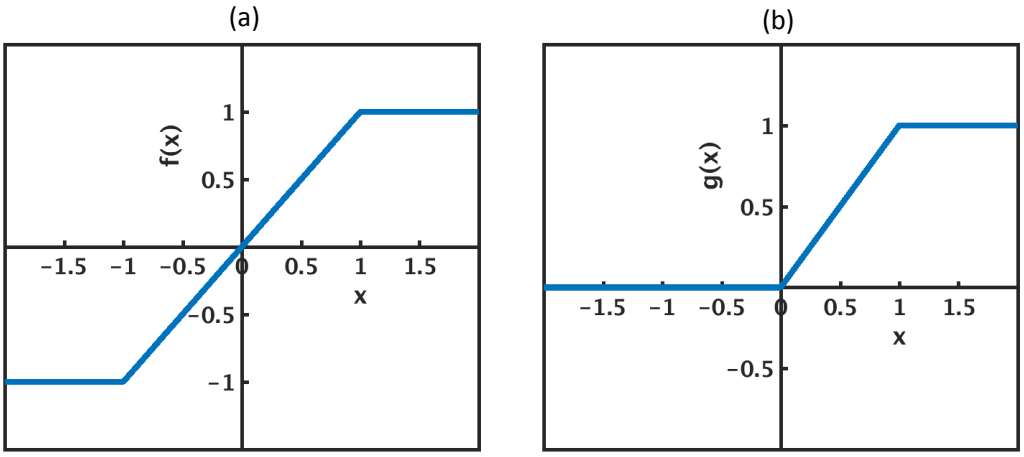

Fig. 2.1. The thresholding functions (a) $f()$ and (b) $g()$ of the analog SAT solver defined by equations (2.4) and (2.5)

$$c_{mi} = \begin{cases} 1 & if\ x_i \in C_m \\ -1 & if\ \overline{x}_i \in C_m \\ 0 & if\ x_i \notin C_m\ and\ \overline{x}_i \notin C_m \end{cases} \tag{2.6}$$

The constraint density (α_c) can be considered as a measure of hardness of a k-SAT problem and is defined as the ratio between the number of clauses and the number of variables

$$\alpha_c = M/N \tag{2.7}$$

In the easy SAT region, only few constraints are there to be satisfied leading to a small α_c. When the number of clauses in the problem is large, deciding whether the propositional formula is unsatisfiable (UNSAT) is easy. There is an intermediate range (hard SAT), where solving the satisfiability problem can be challenging. It has been shown that, for 3-SAT, the hardest problem regime is when the constraint density

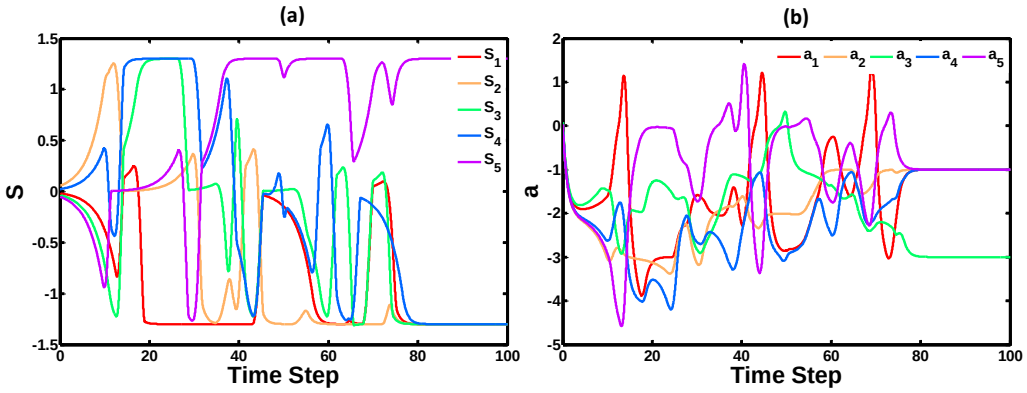

Fig. 2.2. The time evolution of s and a (in (a) and (b) respectively) variables when solving a random 3-SAT problem with 10 variables using the system defined by equations (2.2) and (2.3). The constraint density (α_c) of the problem is 4.25. The self-coupling parameters A and B are set to 1.3 and 2.3 respectively. Only dynamics of five variables are shown here for clarity. The system has converged to a solution at the 80^{th} time step. Note that $g(a)$ is zero after this time.

α_c of the Boolean function is 4.25 and for a 4-SAT problem, the hardest regime is at α_c=9.55 *etc* [9,70–72]. Also, the worst-case complexity of any k-SAT problem depends exponentially on the number of variables, N [61]. The above explained CeNN based system gives solutions to k-SAT problems even in their hardest regime. However, proper tuning of self-coupling parameters A and B is indispensable to achieve better performance. Figure 2.2 depicts the time evolution of s and a variables when solving a 3-sat problem with 10 variables, using proper A, B values for equations (2.2,2.3).

2.3 Magnetic Tunnel Junctions mimicking the behavior of the CeNN based SAT solver

In this work, we present a hardware platform built on nano-scale spintronic devices, which can successfully emulate the behavior of the aforementioned SAT solver. As a matter of fact, recent studies have demonstrated efficient hardware models that utilize the underlying device physics of nano-electronic structures to perform computationally intensive calculations [73–76]. In this work, each differential equation (2.2 and 2.3) is modeled by a single MTJ with an underlying heavy metal (Ta,Pt, etc.) layer. Our numerical results demonstrate that, the MTJ based SAT solver exhibits a polynomial dependency, between the number of variables and the real time for convergence, even for SAT problems that are known to be hardest to solve. We conjecture that, this is due to the non-deterministic nature of our system caused by the random thermal noise, and also due to the added complexities associated with MTJs.

The proposed hardware based SAT-solver is a collection of heavy metal-MTJ (HM-MTJ) structures, interfaced through simple CMOS peripheral circuitry as described in the section 2.4. The device-circuit structure we propose is generic and can be adapted to solve a given k-SAT problem. The HM-MTJ structure is composed of two ferromagnetic layers called the pinned layer (PL) and the free layer (FL), separated by a thin tunneling oxide (MgO) layer and an HM under-layer (figure 1.3 (c)). The

PL magnetization direction ($\hat{p}$) is fixed and acts as a reference. In contrast, the magnetization direction $\hat{m}$ of the FL can be switched by passing a current through the HM under-layer, using the Spin Orbit Torque (SOT) phenomenon. Such a technique has emerged as an energy-efficient mechanism for magnetization reversal [39, 77, 78]. Furthermore, the three terminal structure of this MTJ with a heavy metal under layer is beneficial in this work due to the possibility of simultaneous read and write to an MTJ [79]. This is impossible to achieve in a two terminal MTJ structure that requires the write current to flow through the tunnel junction.

The resistance measured across the MTJ varies with the magnetization of the FL, and shows two stable states; high resistive (R_{AP}) anti-parallel (AP) state, and low resistive (R_P) parallel (P) state. Equations 2.8 and 2.9 depict how the resistance across the MTJ (R_{MTJ}) varies with the direction of the FL magnetization, where θ_{fp} is the angle between the directions of FL and PL magnetizations [80]. The magnetization reversal dynamics of an MTJ with an applied current can be explained using the Landau-Lifshitz-Gilbert-Slonczewski (LLGS) equations [38]. The speed of this magnetization reversal can be controlled by the magnitude of the current passing through the HM layer. However, due to the effect of random thermal noise on nano-scale magnets, the MTJ switching speed follows a Gaussian distribution.

$$R_{MTJ} = \left(\frac{1}{R_P} \left(cos(\frac{\theta_{fp}}{2}) \right)^2 + \frac{1}{R_{AP}} \left(sin(\frac{\theta_{fp}}{2}) \right)^2 \right)^{-1} \tag{2.8}$$

$$\theta_{fp} = cos^{-1}\big(\hat{m}.\hat{p}\big) \tag{2.9}$$

The resistance across an MTJ, R_{MTJ}, can be easily converted into a voltage by using a simple resistor divider circuit. In the proposed hardware implementation of the SAT solver, each s_i and a_m variable from equations 2.2 and 2.3 are represented using a single HM-MTJ structure. The state of these variables at a particular time instant are given by the resistance across the corresponding MTJ device. The couplings between the s and a variables (terms $\sum_m c_{mi} g(a_m(t))$ and $-\sum_i c_{mi} f(s_i(t))$ in equations 2.2 and 2.3) are mapped as currents through the HM layer using the interface circuitry explained in a separate section.

It can be intuitively explained how our MTJ based SAT solver mimics the system elaborated in equations (2.2 - 2.3). One main feature of this system is that, the current values of the variables depend on their previous states as well as some inputs. Similarly, the FL magnetization of an MTJ depends on its previous magnetization as well as the driving current. Another feature of the system in (2.2 - 2.3) is that, when the feedback from a_m towards the dynamics of s_i ($i.e.$, $\sum_m c_{mi} g(a_m(t_0))$) is zero after a particular time t_0, s_i will move towards $+A$ if $s_i(t_0) > 0$, and $-A$ if $s_i(t_0) < 0$, provided $A > 1$. Similarly, in an MTJ, an instantaneous removal of a current through the HM layer, will lead the free layer magnetization to settle down either to the parallel state or to the anti-parallel state. The state to which the magnet settles down is highly dependent upon the resistance it had at the time of removal of the current, in the absence of thermal noise. When the angle between the PL and FL magnetization directions $\theta_{fp} > \frac{\pi}{2}$ ($\theta_{fp} < \frac{\pi}{2}$), and if the drive current is zero, then the final FL magnetization will settle down to the anti-parallel (parallel) state. However, it should be noted that this phenomenon occurs under certain conditions. In this work, we optimized the FL thickness according to the following equation to exhibit the above behavior.

$$t_{ss} = \frac{2K_i}{(N_{zz} - N_{yy})\mu_0 M_s^2} \tag{2.10}$$

where K_i is the energy density constant for interface perpendicular anisotropy and N_{zz}, N_{yy} are the demagnetization factors along y and z directions. The derivation of this FL thickness is explained in detail in section 2.3.1. The new traversal of the magnetization of a device with a thickness of t_{ss} is illustrated in figures 2.3(a),(b). Figures 2.3(c),(d) depicts the magnetization traversal of an MTJ with a thickness larger than t_{ss} for reference. Note the smooth transition of the FL magnetization component along the easy axis (denoted as M_x) in figure 2.3(a) in contrast to the oscillatory transition of that in figure 2.3(c). In the light of this observation, we name t_{ss} as the seamless switching thickness of an MTJ.

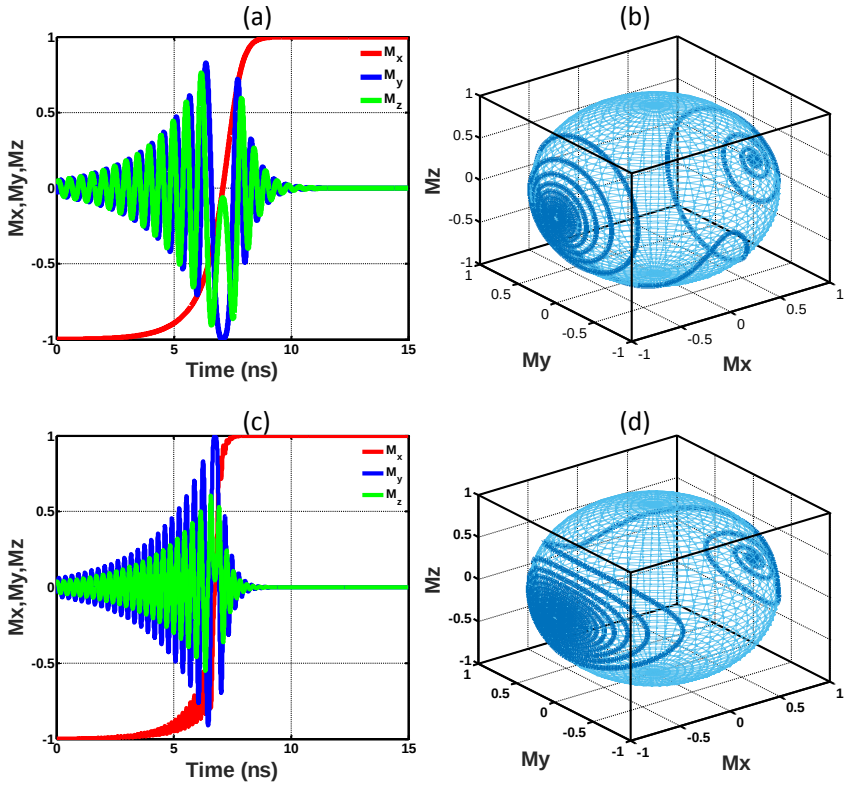

Fig. 2.3. An MTJ changing its state from P to AP due to an applied current. (a) Time evolution of the components of unit magnetization vector in an MTJ with a FL thickness of t_{ss} (b) Unit magnetization traversal (in 3 dimensional space) of an MTJ with a FL thickness of t_{ss} (c) Time evolution of the components of unit magnetization vector in an MTJ with a FL thickness larger than t_{ss} (d) Unit magnetization traversal (in 3 dimensional space) of an MTJ with a FL thickness larger than t_{ss}. Notice the lack of oscillations in M_x in (a), during the magnetization reversal of the FL, in contrast to (c).

2.3.1 Seamless switching Thickness for dominant in plane magnetic anisotropy MTJ devices.

The state of an MTJ at a particular time instant depends on its previous state, and the current that passes through the the HM layer. In order to mimic the CeNN based dynamic system (equation 2.2 - 2.3), it is important that the state of an MTJ (when there is no applied current) goes to the parallel (anti-parallel) state, if the

angle between the FL and the PL magnetization is $\theta_{fp} < \pi/2$ ($\theta_{fp} > \pi/2$). We have optimized the thickness of the FL so that the MTJ exhibits this behavior. Figure 2.4 shows how a typical MTJ with a dominant in-plane magnetic anisotropy field (IMA-MTJ) behaves (here the actual thickness is larger than the value we have derived). The color of each point in the graph refers to the ultimate state to which the free layer magnetization settles down with no current applied. The position of each point shows the initial condition [ϕ,θ] in standard spherical coordinate notations.

We will now explain how we can make the MTJ state go towards the parallel (anti-parallel) state when $\theta_{fp} < \pi/2$ ($\theta_{fp} > \pi/2$) at a particular time instant after which there is no driving current. Let us first consider the fields that affect the FL magnetization. The $\overrightarrow{H}_{EFF}$ in the LLGS equation (1.15) consists of two major types of anisotropy fields; the demagnetization field and interface anisotropy field. For typical choices of materials for MTJs, these anisotropy fields are of the following form.

Demagnetization field [81]

$$H_D = [-M_s N_{xx} m_x \quad - M_s N_{yy} m_y \quad - M_s N_{zz} m_z] \tag{2.11}$$

Interface anisotropy field [82]

$$H_\perp = [0 \quad 0 \quad \frac{2K_i}{\mu_0 M_s t_{FL}} m_z] \tag{2.12}$$

Where N_{xx}, N_{yy} and N_{zz} are the demagnetization factors, and K_i is the energy density constant for the perpendicular interface anisotropy. When the applied current to the MTJ through the HM layer is zero, the current induced spin transfer torque will not be present (i.e. $\overrightarrow{STT} = 0$). The component of the magnetization that contributes to the resistance is the one along the direction of the fixed layer (or easy axis which is $\hat{x}$ in this particular case). The dynamics of that component of magnetization is as follows.

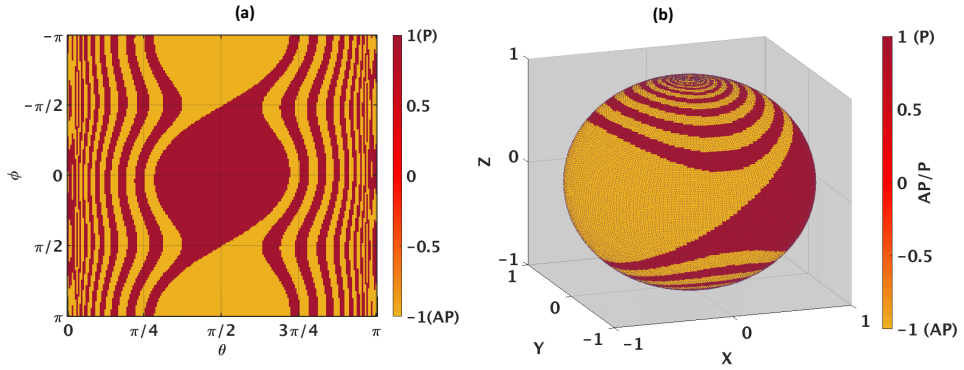

Fig. 2.4. The two colors show the state to which the free layer magnetization settles down in a typical IMA-MTJ (FL thickness $\neq t_{ss}$ in equation (2.3)). There is no current or external field applied, and the initial condition is (ϕ, θ) in standard spherical coordinate notations. Note that thermal noise is not present in this analysis. The pinned layer magnetization is in $+\hat{x}$ direction. It is evident that the angle between the free and pinned layer (θ_{fp}) does not alone decide the final state of the MTJ. (b) shows the same final states in (a) in 3-dimensional space.

$$
\begin{aligned}
\left(\frac{1+\alpha^2}{|\gamma|}\right)\frac{\partial m_x}{\partial t} &= -\Big(\hat{m}\times\overrightarrow{H}_{EFF} + \alpha\hat{m}\times\hat{m}\times\overrightarrow{H}_{EFF}\Big).\hat{x} \\
&= -\Bigg(m_y H^z_{EFF} - m_z H^y_{EFF} - \alpha|\hat{m}|^2 H^x_{EFF} \\
&+ \alpha m_x\big(m_x H^x_{EFF} + m_y H^y_{EFF} + m_z H^z_{EFF}\big)\Bigg) \\
&= -\Bigg(- m_y m_z N_{zz} M_s + m_y m_z N_{yy} M_s + \frac{2K_i}{\mu_0 M_s t_{FL}} m_y m_z \\
&+ \alpha\Big((1-m_x^2)N_{xx}M_s m_x - m_x m_y^2 N_{yy} M_s - m_x m_z^2 N_{zz} M_s \\
&+ \frac{2K_i}{\mu_0 M_s t_{FL}} m_x m_z^2\Big)\Bigg)
\end{aligned}
\tag{2.13}
$$

According to the above equations, it is evident that the final state of the free layer magnetization does not solely depend on the sign of m_x. The existence of terms such as $m_y m_z$ suggests that the final state will depend even on m_y and m_z as well,

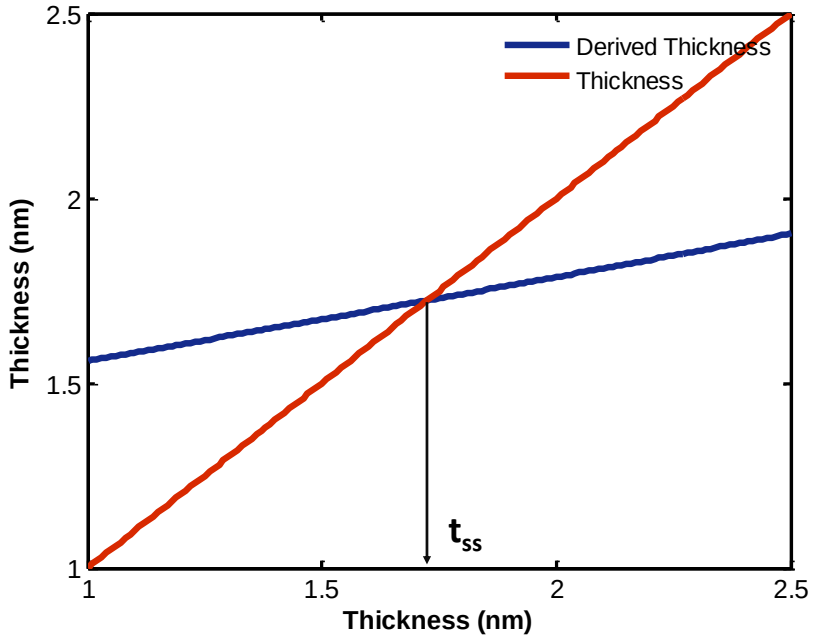

Fig. 2.5. Seamless switching thickness (t_{ss}) solved self consistently for a rectangular IMA-MTJ device. "Derived thickness" (blue) is the t_{ss} obtained from equation 2.3 with demagnetization parameters that correspond to the thickness in x axis. "Thickness" (red) is the 1:1 graph of FL thickness.

justifying the behavior depicted by figure 2.4. To address this, we use a FL thickness as follows.

This thickness t_{ss} will make the $\hat{x}$ component of the precession term $(\hat{m} \times \overrightarrow{H}_{EFF}).\hat{x}$ zero in the LLGS equation. The updated magnetization dynamics is shown in equation 2.14. Note that during the derivation we used the fact that $N_{zz} > N_{yy} > N_{xx}$. This is due to the special dimension requirement that must be followed to allow the magnetization to be stable in the $\hat{x}$ direction. The aspect ratio between the length and the width must be greater than one, as graphically shown in figure 1.3.

$$\left(\frac{1+\alpha^2}{|\gamma|}\right)\frac{\partial m_x}{\partial t} = m_x \alpha M_s (1 - m_x^2)(N_{yy} - N_{xx}) \tag{2.14}$$

Now the dynamics depict a direct relationship with the sign of m_x. When $m_x > 0$, ($m_x < 0$) the final state will be parallel (antiparallel) when no spin current is present, provided that the PL magnetization is directed towards $+\hat{x}$ direction. The effect of noise and process variations is not considered in this section. It is separately discussed in section 2.6.1. Note that the demagnetization factors N_{zz}, N_{yy} (in equation 2.3) in return depends on the dimensions of the FL including the thickness [83]. Therefore, solving for the t_{ss} must be done self consistently. Figure 2.5 shows the evaluation of this t_{ss} (or the 'seamless switching' thickness) for a particular MTJ device.

The new FL final magnetization states of an MTJ with the optimized thickness are graphically shown in figure 2.6. It illustrates how the magnetization converges to a state under zero drive current when the free layer thickness is t_{ss}. Note that the final magnetization depends on just whether m_x is positive or negative, at the time the write current is made zero. The effect of noise will make the final state, a probabilistic function of the magnitude of m_x. *i.e.* higher the magnitude of m_x, more likely it is that the final state will solely depend on the sign of m_x as shown in figure 2.6 (c) and (d).

2.3.2 Theorems to prove the functionality equivalence of the MTJ based system to the CeNN based system

It is mathematically shown [9] that the system explained in section 2.2 satisfies three theorems that demonstrate the properties of the model. The same theorems are used here, to show that our hardware SAT solver demonstrates the same properties. Following are the three theorems.

- Theorem 1: Variables s and a remain bounded.
- Theorem 2: Every k-SAT solution has a corresponding stable fixed point
- Theorem 3: A stable fixed point always corresponds to a solution

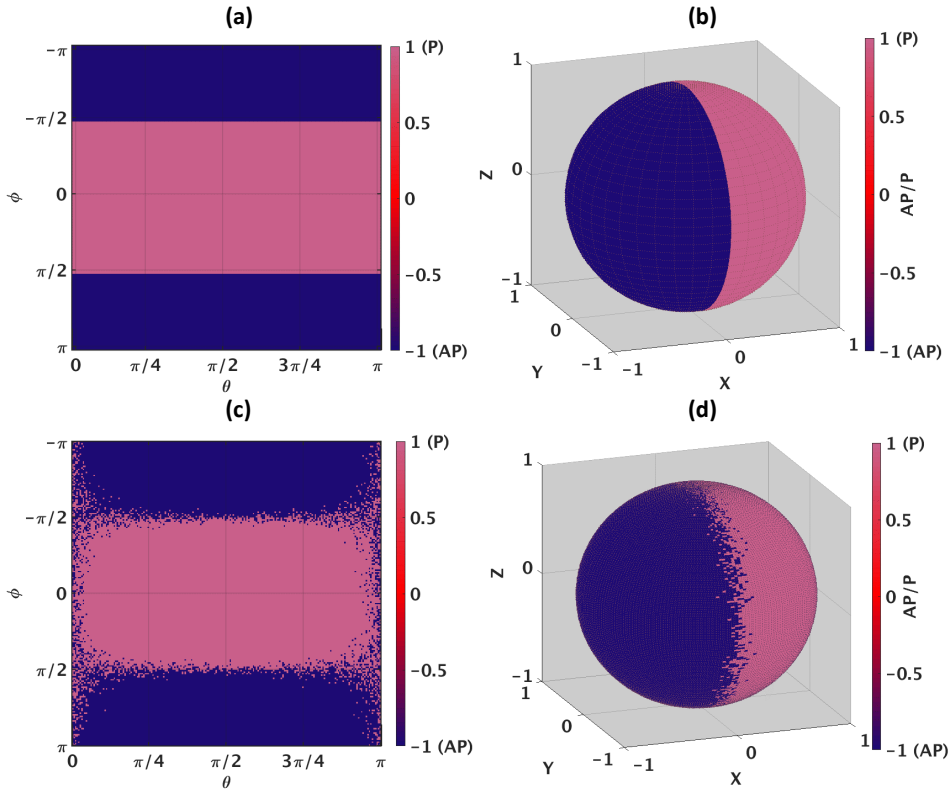

Fig. 2.6. The color shows the state to which the free layer magnetization settles down in our proposed IMA-MTJ device, when there is zero current or external field applied on the MTJ, with an initial condition (ϕ,θ). (a), (b) show the converged final state in absence of thermal noise. (c), (d) show the converged final state when noise is present. (b) and (d) show the same final states in (a) and (c) respectively, in 3 dimensional space.

As explained previously, the state of each MTJ can be measured as a voltage and the input to each MTJ is supplied as a current through the HM layer. However, for simplicity, we will not discuss this resistance to voltage conversions in this section. We will elaborate only using the magnetization dynamics along the easy axis of the MTJs since it gives a direct mapping to the resistance of the MTJs, and thus the output voltage. There are two types of MTJs in our design; a type and s type. The states (resistances) of s type MTJs ($s_i = [s_{i,x} \quad s_{i,y} \quad s_{i,z}]^T, \forall i \leq N$) give the solution to a satisfiability problem $\mathcal{F}$ and the states of a type MTJs ($a_m = [a_{m,x} \quad a_{m,y} \quad a_{m,z}]^T, \forall m \leq M$)

exhibit the "satisfiedness" of the respective clauses. The easy axis of both a and s type MTJs is the $\hat{x}$ axis and the pinned layer magnetization is pointed towards $-\hat{x}$ direction. We define the FL magnetization component along the easy axis of s type MTJs as $-1 \leq s_{i,x} \leq 1$. When the system has converged to a solution, $s_{i,x} = +1$ ($s_{i,x} = -1$) represents $x_i = 1$ ($x_i = 0$) in the Boolean formula $\mathcal{F}$ for $\forall i \leq N$. For the a type MTJs, the FL magnetization component along the easy axis is $a_{m,x}$. When the system has converged to a solution, clause m in $\mathcal{F}$ must be satisfied, and $a_{m,x} = -1$ for $\forall m \leq M$. Further, we apply a rectifying function similar to $g()$ on $a_{m,x}$. The direction of current is defined such that, a positive (negative) current will drive that MTJ towards anti parallel (parallel) state. We will not use the effect of noise and process variations for the proof of the following theorems. Such effects will be analyzed separately in section 2.6.1.

$$\Big(\frac{1+\alpha^2}{|\gamma|}\Big)\frac{\partial s_{i,x}}{\partial t} = \alpha s_{i,x} M_s (1 - s_{i,x}^2)(N_{yy} - N_{xx}) + k_{s,STT}(1 - s_{i,x}^2)\sum_m c_{mi} g(a_{m,x}) \tag{2.15}$$

$$\begin{aligned}\Big(\frac{1+\alpha^2}{|\gamma|}\Big)\frac{\partial a_{m,x}}{\partial t} =& \alpha a_{m,x} M_s (1 - a_{m,x}^2)(N_{yy} - N_{xx}) \\ &+ k_{a,STT}(1 - a_{m,x}^2)(1 - k - \sum_i c_{mi} s_{i,x})\end{aligned} \tag{2.16}$$

$$k_{s,STT} = \frac{\hbar k_{she,s} I_s \epsilon}{2q\mu_0 M_s V_s}, k_{a,STT} = \frac{\hbar k_{she,a} I_a \epsilon}{2q\mu_0 M_s V_a} \tag{2.17}$$

$$k_{she,s} = \frac{A_{MTJ,s}\theta_{she}}{A_{HM,s}}\left(1 - sech\left(\frac{t_{HM,s}}{\lambda_{sf}}\right)\right), k_{she,a} = \frac{A_{MTJ,a}\theta_{she}}{A_{HM,a}}\left(1 - sech\left(\frac{t_{HM,a}}{\lambda_{sf}}\right)\right) \tag{2.18}$$

$$c_{mi} = \begin{cases} 1 & if\ x_i \in C_m \\ -1 & if\ \overline{x}_i \in C_m \\ 0 & if\ x_i \notin C_m\ and\ \overline{x}_i \notin C_m \end{cases} \tag{2.19}$$

The above equation set defines the behavior of our MTJ based SAT solver. The subscripts s and a denotes that the given parameter is related to the MTJs that represent variable s and a, respectively. M_s is the saturation magnetization, I is the charge current through the HM layer, V is the volume of the FL, θ_{she} is the Spin Hall Effect (SHE) angle, λ_{sf} is the spin flip length, t_{HM} is the thickness of the FL, A_{MTJ} is the area of the FL at the HM and FL interface, and A_{HM} is the area perpendicular to the charge current through the HM. C_m is the m^{th} clause. Note that including the field-like torque term (equation 1.3 must be updated in this case to $\overrightarrow{STT} = |\gamma|\beta\Big(\widehat{m} \times \big(\epsilon\widehat{m} \times \widehat{p} + \epsilon^{'}\widehat{p}\big)\Big)$, where ϵ and $\epsilon^{'}$ are dimensionless factors that describe the effectiveness of the spin transfer process) changes the above $k_{s,STT}$ and $k_{a,STT}$ terms (equation 2.15, 2.16). Instead of ϵ in these terms, a new factor of $\epsilon + \alpha\epsilon^{'}$ will be present. This can be viewed as a slight increment to the charge current (I_s, I_a). However, due to the small damping factor α, this effect is smaller and thus not considered for the following analysis.

Theorem 1 : Variables s and a remain bounded

$s_i = [s_{i,x} \quad s_{i,y} \quad s_{i,z}]$ and $a_m = [a_{m,x} \quad a_{m,y} \quad a_{m,z}]$ are unit magnetization vectors. $s_{i,x}$ and $a_{m,x}$ are the components along the easy axis $\hat{x}$, as defined earlier. The highest possible value of these are $+1$ and the lowest is -1.

Therefore all $s_{i,x}$ and $a_{m,x}$ remain bounded for all time instances.

Theorem 2: Every k-SAT solution has a corresponding stable fixed point

Let $s^*_{i,x}$ be a solution to a SAT formula $\mathcal{F}$ for $i = 1, 2, ..., N$. The point (s^*_x, a^*_x) where

$$s^*_{i,x} = \pm 1, \forall i \leq N \quad a^*_{m,x} = -1, \forall m \leq M \tag{2.20}$$

is a stable fixed point of the dynamic system.

According to the equations 2.15 and 2.16, it is evident that both $\frac{\partial s^*_{i,x}}{\partial t}$ and $\frac{\partial a^*_{m,x}}{\partial t}$ are zero. (note that $a^*_{m,x} < 0, g(a^*_{m,x}) = 0$). Therefore the point (s^*_x, a^*_x) is a fixed point. In order to prove stability, we shall give a small perturbation of $|s^*_{i,x}| - |s_{i,x}| = \delta_{i,s}$ and $a_{m,x} - a^*_{m,x} = \delta_{m,a}$. Note that $|s_{i,x}| \leq 1$ and $|a_{m,x}| \leq 1$ since they are components of unit vectors s_i and a_m. Such perturbations will result in positive $(1 - s^2_{i,x})$ and positive $(1 - a^2_{m,x})$. If the variable x_i is present in the m^{th} clause ($c_{mi} \neq 0$), and if the clause is satisfied by $s^*_{i,x}$, then $c_{mi}s^*_{i,x} = 1$. If the clause is not satisfied, then $c_{mi}s^*_{i,x} = -1$. Accordingly the sum $\sum_i c_{mi}s^*_{i,x}$ can take $k + 1$ possible values; $-k$, $-k + 2$,...,$k - 2$, k. The value $-k$ corresponds to the m^{th} clause not being satisfied. Other cases imply that there exists at least one variable satisfying the clause. As per our assumption that $s^*_{i,x}$ is a solution of the SAT problem $\mathcal{F}$,

$$\sum_i c_{mi}s^*_{i,x} \geq -k + 2 \quad \rightarrow \quad 1 - k - \sum_i c_{mi}s^*_{i,x} \leq -1 \tag{2.21}$$

Due to the perturbation, there will be a negative current through HM in the a type MTJs. This will shift $a_{m,x}$ towards $a^*_{m,x}$ (*i.e.* towards -1 or parallel state). As long as $|\delta_{m,a}| < 1$, the $a_{m,x}$ after thresholding $g(a_{m,x})$ is zero, leading to a zero current through the HM of the s type devices. When there is no current through the HM of the s type devices, as long as $|\delta_{i,s}| < 1$, $s_{i,x}$ will reach $s^*_{i,x}$. Therefore, the solution point (s^*_x, a^*_x) is a stable fixed point.

Theorem 3 : A stable fixed point always corresponds to a solution

Let us assume that there exists a stable fixed point (s_x^*, a_x^*) for the Boolean SAT problem with some unsatisfied clauses where $|s_{i,x}^*| = 1$ and $|a_{m,x}^*| = 1$. Since $1 - {s_{i,x}^*}^2 = 0$ and $1 - {a_{m,x}^*}^2 = 0$, automatically

$$\Big(\frac{1+\alpha^2}{|\gamma|}\Big)\frac{\partial s_{i,x}^*}{\partial t} = 0, \quad \Big(\frac{1+\alpha^2}{|\gamma|}\Big)\frac{\partial a_{m,x}^*}{\partial t} = 0 \tag{2.22}$$

By multiplying both sides of the equation 2.15 by $sign(s_{i,x}^*)$, we obtain

$$\begin{aligned}\Big(\frac{1+\alpha^2}{|\gamma|}\Big)\frac{\partial |s_{i,x}^*|}{\partial t} =& \alpha |s_{i,x}^*| M_s (1 - {s_{i,x}^*}^2)(N_{yy} - N_{xx}) \\ &+ k_{s,STT}(1 - {s_{i,x}^*}^2)\sum_m sign(s_{i,x}^*) c_{mi} g(a_{m,x})\end{aligned} \tag{2.23}$$

since (s_x^*, a_x^*) is a fixed point by assumption,

$$\begin{aligned}(1 - {s_{i,x}^*}^2)\Bigg(\alpha |s_{i,x}^*| M_s (N_{yy} - N_{xx}) + k_{s,STT}\Big(\sum_{sign(s_{i,x}^*)c_{mi}=1} g(a_{m,x}^*) - \\ \sum_{sign(s_{i,x}^*)c_{ni}=-1} g(a_{n,x}^*)\Big)\Bigg) = 0\end{aligned} \tag{2.24}$$

Let us assume that there are P number of unsatisfied clauses in which the variable x_i or $\bar{x_i}$ appears.

$$\sum_{sign(s_{i,x}^*)c_{ni}=-1} g(a_{n,x}^*) = P \tag{2.25}$$

For the other clauses that are satisfied at the fixed point, $g(a_{m,x}^*) = 0$. This will lead to,

$$(1 - {s_{i,x}^*}^2)\Big(\alpha |s_{i,x}^*| M_s (N_{yy} - N_{xx}) - k_{s,STT} P\Big) = 0 \tag{2.26}$$

However, since $\frac{k_{s,STT}}{\alpha M_s (N_{yy} - N_{xx})} > 1$ depending upon the typical values of parameters, $|s_{i,x}^*| \neq \frac{k_{s,STT} P}{\alpha M_s (N_{yy} - N_{xx})}$. This leads to $(1 - {s_{i,x}^*}^2) = 0$ become the sole reason for the

existence of the stationary point. Similar situation exists for state dynamics of a-type MTJs. Let us consider an unsatisfied clause m, and the corresponding dynamics. As mentioned previously, for an unsatisfied clause, $\sum_i c_{mi} s^*_{i,x} = -k$. Further, since $a^*_{m,x}$ is a stationary point as per our assumption, $\left(\frac{1+\alpha^2}{|\gamma|}\right)\frac{\partial a^*_{m,x}}{\partial t} = 0$. Therefore

$$(1 - {a^*_{m,x}}^2)\Big(\alpha a^*_{m,x} M_s(N_{yy} - N_{xx}) + k_{a,STT}\Big) = 0 \tag{2.27}$$

However, $a^*_{m,x} \neq -\frac{k_{a,STT}}{\alpha M_s(N_{yy}-N_{xx})}$ since it violates our initial assumption that the clause is not satisfied ($a^*_{m,x} > 0$). Note that $N_{yy} > N_{xx}$. This leads to $(1 - {a^*_{m,x}}^2) = 0$ become the sole reason for the existence of the stationary point. Now let us introduce small perturbations $\delta_{i,s}$ and $\delta_{m,a}$ to the system such that, $|s^*_{i,x}| - |s_{i,x}| = \delta_{i,s}$ and $a_{m,x} - a^*_{m,x} = \delta_{m,a}$. As mentioned in theorem 2, this will lead to

$$(1 - {s_{i,x}}^2) > 0, \quad (1 - {a_{m,x}}^2) > 0 \tag{2.28}$$

Since the perturbation is small, $|s_{i,x}| \cong 1, |a_{m,x}| \cong 1$ is valid as well. When $s_{i,x} \cong +1$, only the clauses where the variable appears as its complement ($c_{mi} = -1$) are not satisfied. Similarly, when $s_{i,x} \cong -1$, the clauses where $c_{mi} = 1$ are not satisfied.

$$\Big(\frac{1+\alpha^2}{|\gamma|}\Big)\frac{\partial s_{i,x}}{\partial t} = \begin{cases} (1 - {s_{i,x}}^2)\Big(\alpha s_{i,x} M_s(N_{yy} - N_{xx}) - k_{s,STT} P_-\Big) < 0, for s_{i,x} > 0 \\ (1 - {s_{i,x}}^2)\Big(\alpha s_{i,x} M_s(N_{yy} - N_{xx}) + k_{s,STT} P_+\Big) > 0, for s_{i,x} < 0 \end{cases} \tag{2.29}$$

Where $P_\pm = \sum_{c_{mi}=\pm 1} g(a_{m,x})$. In the light of this, it can be observed that there exists an unstable direction along which the dynamics can escape from the stationary point (s^*_x, a^*_x). That is, $||s^*_{i,x} - s_{i,x}(t)|| \leq \delta$ for any $t > t_0$ is not valid (with t_0 being the time at which the perturbation is applied). Therefore, this disproves the initial assumption that (s^*_x, a^*_x) is a stable stationary point. Therefore, if there exists a stable stationary point in the defined continuous system, they correspond only to solutions of the SAT problem $\mathcal{F}$.

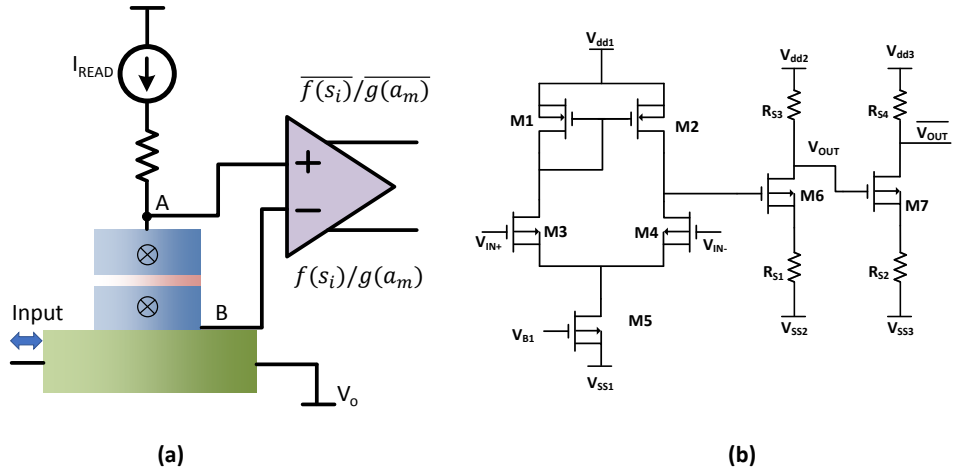

Fig. 2.7. (a) The MTJ circuit for an s/a variable. The input current through the HM layer will change the state of the MTJ and this change will be measured by the amplifiers. The read current is assumed to be constant and its magnitude should be small so that it does not hinder the proper operation of the system. The MTJ resistance changes with the input current. The non-inverting amplifier output generates the 'state' $(f(s_i)/g(a_m))$ and the inverting amplifier output generates the 'inverse-state' $(\overline{f(s_i)/g(a_m)})$ of an MTJ. (b) The reference circuit of the differential amplifier. V_{OUT} is the non-inverting output and $\overline{V_{OUT}}$ is the inverting output.

2.4 Structure of the SAT solver

In this section, it will be elaborated how the system in (2.2-2.3) is mapped to an array of HM-MTJs with a CMOS control interface. Each s and a variable in (2.2-2.3) is represented by the resistance of an MTJ. The resistance of the MTJ can be read as a voltage difference, when a constant read current (I_{READ}) flows through the MTJ. Note that this read current must be sufficiently small ($< 1\mu A$) to not to interfere with the proper operation of the system. The functions $f()$ and $g()$ can be generated efficiently using a differential amplifier shown in figure 2.7(a). Note that the amplifier is connected to the bottom of the magnet (not the heavy metal layer). This is to avoid the small variable voltage (ΔV) induced across the HM layer due to

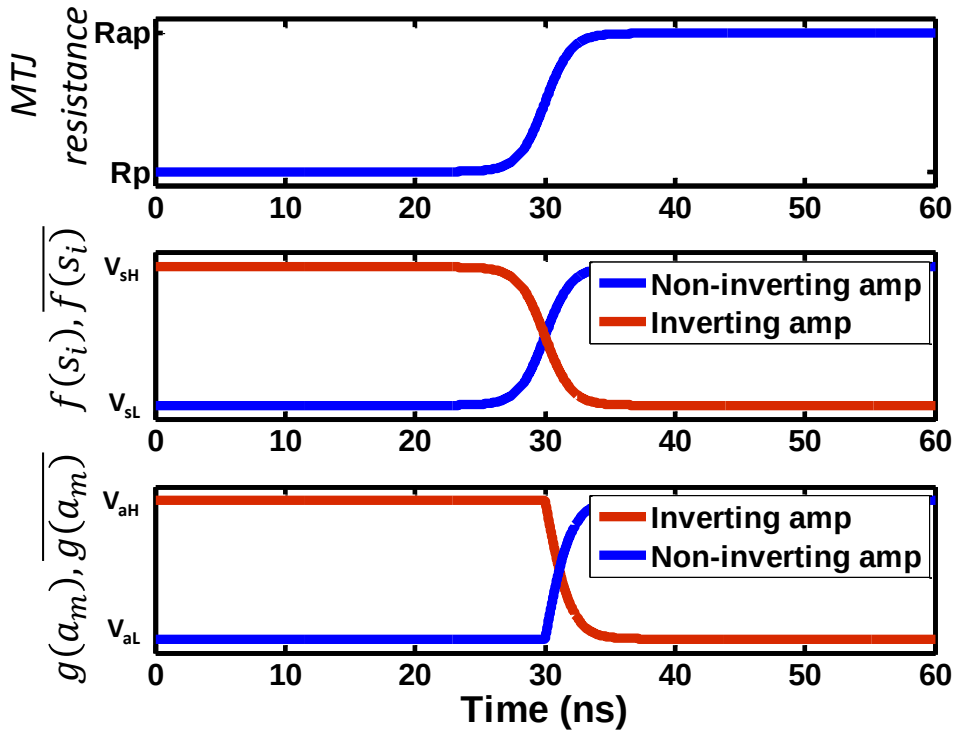

Fig. 2.8. The outputs of the differential amplifiers connected to variable s and a during a state change of an MTJ from parallel to anti parallel state.

the varying current that flows through it. These amplifiers will increase the voltage differences incurred due to the changing resistance of the MTJs. The state AP results in a larger voltage difference between nodes A and B with respect to that resulting from P state. In our structure, the AP and the P states of an MTJ that represents an s variable, gets mapped in to +1 and -1 states in equation (2.2) respectively. In a k-SAT problem, a variable can appear as x_i, or its negation ($\bar{x}_i$) in the m^{th} clause. This information is encoded in the elements of the connection matrix c_{mi}, as explained in section 2.2. In order to account for different values of c_{mi} at the circuit level, we generate the 'state' ($f(s_i)/g(a_m)$) and the 'inverse-state' ($\overline{f(s_i)}/\overline{g(a_m)}$) signals of an MTJ. These signals are produced at the amplification stage outputs, as shown in figure 2.7(b). A differential amplifier is employed to read the voltage difference

across an MTJ. Additionally, a source degenerated common source amplifier is used as a second amplification stage, to boost the voltage to the desired levels. A third amplifier is employed in the design to generate the aforementioned inverse functions ($\overline{f()}$ and $\overline{g()}$). The complete schematic of the amplification stages used in this work is shown in figure 2.7(b). The same amplifier architecture with different control voltages was used for interfacing with MTJs representing both a and s variables. The outputs V_{OUT} ($f(s_i)$ or $g(a_m)$) and $\overline{V_{OUT}}$ ($\overline{f(s_i)}$ or $\overline{g(a_m)}$) are used to drive the MOSFETs controlling the current through the heavy metal layers (figure 2.9).

Figure 2.8 (c) elaborates how the above mentioned 'states' and 'inverse states' vary with the resistance of an MTJ. Each differential amplifier output will vary between a predefined high voltage (V_{sH}, V_{aH}), and a low voltage (V_{sL}, V_{aL}). Therefore, the state -1 and +1 of variable s will be mapped to V_{sL} and V_{sH} at the output of the non-inverting (V_{sH} and V_{sL} at the output of inverting) amplifier. For the variable a, the non-inverting (inverting) amplifier output will be V_{aL} (V_{aH}) when the resistance of the MTJ is less than $(R_{ap} + R_p)/2$ and V_{aH} (V_{aL}) when the resistance is R_{ap}.

The term $\sum_m c_{mi} g(a_m(t))$ in equation 2.2, and the term $-\sum_i c_{mi} f(s_i(t)) + 1 - k$ in equation 2.3 (the coupling between variable s_i and a_m) are mapped as currents through the HM layers of MTJs, that represent s variables and a variables, respectively. At a particular time instant when the m^{th} clause is not satisfied, if the connection parameter c_{mi} is positive, the current should drive the MTJ that represents variable s_i towards the AP state. Similarly, when c_{mi} is negative, the current should drive that MTJ towards the P state, and when c_{mi} is zero, the current through the HM should be zero. Figure 2.9 (a, b) graphically explains how this is realized at the circuit level. Figure 2.9 (c, d) shows the circuit realization of the feedback from the s_i variable acting on the a_m variable, depending upon the connection parameter c_{mi}. When c_{mi} is positive (negative), $-c_{mi} f(s_i(t))$ should drive the MTJ that represents a_m towards the P (AP) state. The two transistor structures (heavy metal current controllers) in Figure 2.9(c, d) should provide an output voltage of V_o, when the input $f(s_i(t)) = \overline{f(s_i(t))} = (V_{dd} + V_{SS})/2$. This is to make sure that there is no

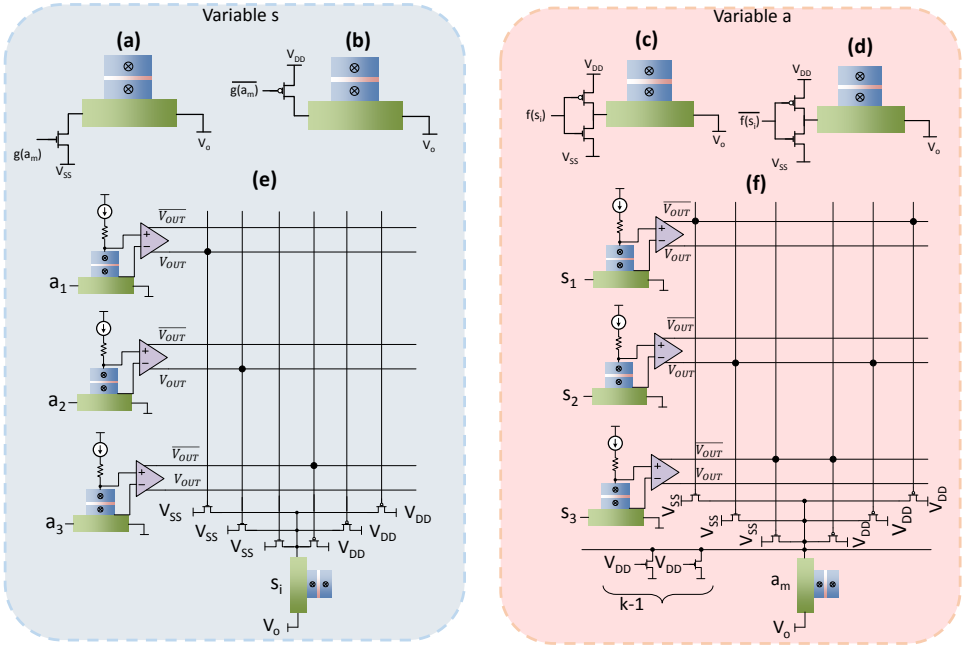

Fig. 2.9. The input connection diagram of the SAT solver. The input connection to an s_i node from a_m, if (a) c_{mi} is negative (b) c_{mi} is positive. The input connection to an a_m node from s_i, if (c) c_{mi} is positive (d)c_{mi} is negative. Here the charge current from left to right through the HM layer drives the MTJ towards the AP state. The value of V_0 is smaller than V_{DD} but larger than V_{ss}. The sizing of the transistors must be done appropriately. (e) Outputs of three a nodes connected to an s_i node. The connection parameters (c_{m1}, c_{m2} and c_{m3}) between s_i and a_1,a_2 and a_3 are -1,-1 and 1, respectively. (f) Outputs of three s nodes connected to an a_m node. The connection parameters (c_{m1}, c_{m2} and c_{m3}) between a_m and s_1,s_2 and s_3 are -1,1 and -1, respectively.

current through the HM layer when $s_i(t) = 0$. For the figure 2.9, we assume that a charge current from left to right through an HM layer, drives the MTJ on top towards the AP state. Figure 2.9(e,f) illustrates the final structure of the SAT solver with all the control logic. The connections in the 'network' depends on the SAT problem to be solved. Therefore, the connecting switches must be initialized depending upon the problem. Note that when the number of clauses of a problem increases, the connections become more complex.

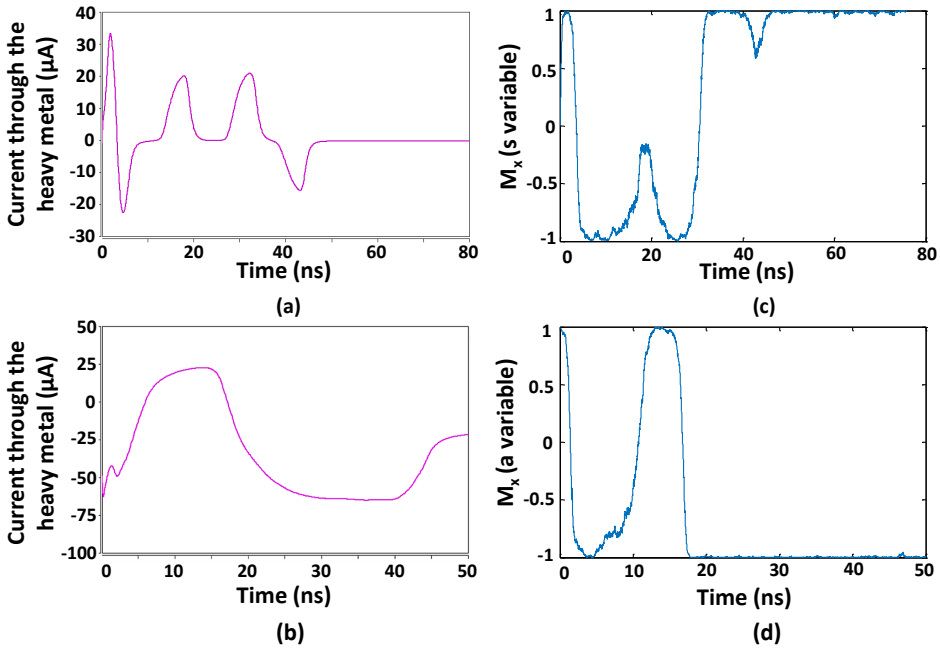

Fig. 2.10. The varying current through the heavy metal layer of two MTJs that represent (a) s variable and (b) a variable of a 10-variable SAT instance ($\alpha_c = 4.25$). The currents were obtained via HSPICE following the circuits explained in figure 2.9, simulated in IBM 45nm technology. The resultant time evolution of the free layer magnetization along the $\hat{x}$ direction is shown on right for (c) s variable and (d) a variable.

2.5 Dynamics of the MTJ based SAT solver

In order to observe the functionality of our SAT solver, circuit level simulations were conducted. Figure 2.10 illustrates the currents through the heavy metal layers of the two MTJs representing s variable and a variable that correspond to a 10-variable hard SAT instance. The results were obtained from HSPICE simulations using IBM 45nm technology node. The resultant evolution of the free layer magnetization along the $\hat{x}$ direction (M_x) is shown on the right (figure 2.10(c) and (d)). Note that a positive current drives the MTJ towards AP (+1) state and a negative current drives the MTJ towards P(-1) state in the figure.

As elaborated in the section 2.2, the constraint density (α_c) is an indicator of the hardness to solve a particular SAT instance. In order to observe the functionality of our solver for SAT instances with different hardness levels, we solved randomly generated 3-SAT problems with different constraint densities, and different number of variables. Figure 2.11 shows the magnetization dynamics of three MTJs that correspond to three variables in two 20 variable 3-SAT problems, each having a constraint density of 4.25 and 3.00, respectively. The color of the trajectories in figure 2.11(c,d) indicates the normalized energy of the system at that particular point. This energy of the system can be defined by the following equations.

$$E(a,s) = \sum_{m=1}^{M} a_m K_m^2 \tag{2.30}$$

$$K_m = 2^{-k} \prod_{i=1}^{N} (1 - c_{mi} s_i) \tag{2.31}$$

where M and N are the number of clauses and the number of variables in the k-SAT problem, respectively. The energy is a function of the number of clauses not satisfied at a particular instant. This can be used as a cost function to determine the "satisfiedness" of a particular problem at a given instant. Notice that the trajectories in figure 2.11 (c), (d) pass through higher energy states as the system tries to converge to a solution. This shows that our system escapes local minimum points naturally, unlike other algorithms [84] where simulated annealing is necessary to escape from such local minimum points.

2.5.1 Approximate polynomial time solution

We also solved randomly generated satisfiable 3-SAT problems in the hard regime ($\alpha_c = 4.25$) for different number of variables (20, 30, 40, 50). We have calculated the number of problems solvable within $10\mu s$ for the purpose of illustration. However, since our system has no limit cycles owing to thermal noise (more details are available in section 2.6), we argue that our proposed method will probably reach a solution

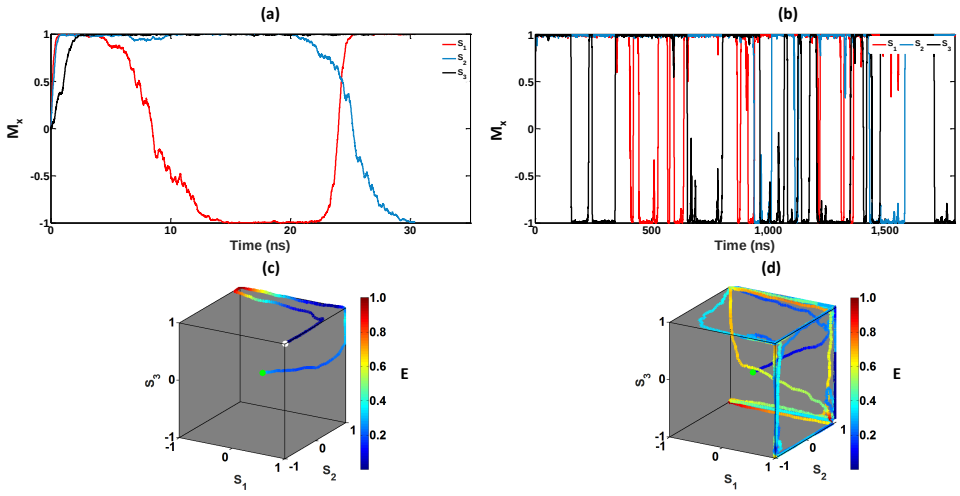

Fig. 2.11. The time evolution of three variables in a 20 variable 3-SAT problem with different constraint densities. (a) and (c) correspond to a SAT problem with a constraint density $\alpha_c = 3$ whereas (b) and (d) correspond to a SAT problem with a constraint density $\alpha_c = 4.25$. (c) and (d) show the trajectories of the same 3 variables in (a) and (b) respectively, while converging to a solution inside a hypercube Q_3. The color presents the energy of the system at a given state. The starting point is a green circle and the end point (solution) is the vertex with a white circle.

if sufficient time has been provided, given that a solution exist. We monitored the fraction of problems not solved by the algorithm at time t and the result is depicted in figure 2.12. It is evident that the fraction of problems not solved $p(t)$, has an exponential decay with time t. The relationship between $p(t)$ and t can be approximated by

$$p(t) = re^{-\lambda(N)t+\gamma} \tag{2.32}$$

where r and γ are constants. The decay rate λ obeys $\lambda(N) = bN^{-\beta}$, with $\beta \approx 1.1$. Therefore the continuous time t needed to solve a $(1-p)$ fraction of problems can be written as

$$t(p, N) = \Big(\gamma + ln(r/p)\Big)b^{-1}N^{\beta} \tag{2.33}$$

This implies that the time to solve a $(1-p)$ fraction from a set of k-SAT problems is of polynomial complexity. This polynomial relationship still holds when the fraction of

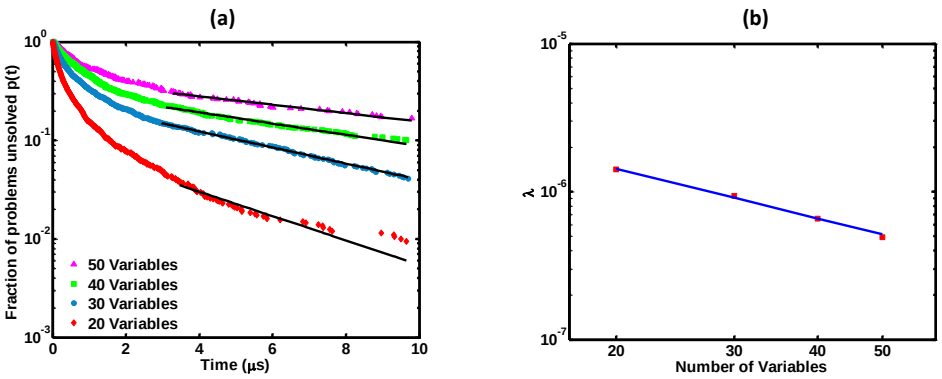

Fig. 2.12. Computational complexity of the SAT solver (a) The fraction of problems $p(t)$ not yet solved at real time t, for 3-SAT problems with $\alpha_c = 4.25$, for N= 20, 30, 40 and 50. Averages were calculated with 10^3 instances for each N. (b) The decay rate λ for different number of variables. λ takes the form $\lambda(N) = bN^{-\beta}$ with β approximately 1.1 (note the log scale).

problems left unsolved is a fixed number (irrespective of the number of variables) [62]. That is, the time taken to solve all possible k-SAT formulas for a given N and α_c ($\Theta(k, N, \alpha_c)$), except for a constant amount of problems c ($p(t) = c/\Theta(k, N, \alpha_c)$), would follow a relationship as shown below when $N \to \infty$ (if we assume that the relationship in equation 2.32 and 2.33 still holds as $N \to \infty$).

$$t(p, N) \sim N^{\beta+1} ln(N) \tag{2.34}$$

A previous proposal [62] shows a similar relationship for the time required to solve k-SAT problems. However, the relationship is not valid in real time if a digital computer is to be used for the calculations. It is because, the complexity of the analog system varies over time, and when solving each step, it would take different real time values depending upon the complexity. Since our method is purely based on hardware, we argue that the above polynomial time relationship is valid in real time for our proposed system. We conjecture that this behavior is due to multiple reasons including the thermal noise associated with the MTJs. It has also been predicted [9]

that, the noise effects may avoid long transient oscillations. Our results too suggest that higher amounts of noise leads to faster convergence (section 2.35). It must be noted that, this proposed MTJ based SAT solver does not behave identical to the cellular neural network based solver in equations (2.2-2.3). Our solver has some added complexities not present in the CeNN based system (refer to the set of equations in section 2.3). We conjecture that such complexities offered by the device physics, act favorable to give faster solutions to SAT problems as well. However, these benefits come at the cost of handling the circuit limitations (fan-out etc.) that can arise when solving problems with larger N.

2.6 Variation Analysis

2.6.1 Effect of Thermal Noise

Thermal noise has significant impact on the switching dynamics of nano-magnets. Equation 2.35 explains the renowned Brown's model [85] that captures the behavior of thermal noise which can be used as a random magnetic field in the LLGS equations.

$$\vec{H}_{Thermal} = \vec{\varsigma}\sqrt{\frac{2\alpha k_B T}{|\gamma| M_s V}} \tag{2.35}$$

$\vec{\varsigma}$ is a vector with components that are zero mean Gaussian random variables with standard deviation of 1. V is the volume of the free layer, T is the temperature, and k_B is the Boltzmann's constant. The time discretization value dt must be included in the numerator when solving the equation numerically. The existence of thermal noise is mandatory for the proper operation of our SAT solver. This is due to the tilt of the FL magnetization with the easy axis, induced by thermal noise without which a magnet cannot be switched. Results indeed show that, under zero thermal noise, the magnetizations evolve to frozen non-solution states.

As implied by equation (2.15) and (2.16) in section 2.3, the state $\hat{m} = [m_x\hat{x} \quad m_y\hat{y} \quad m_z\hat{z}] = [\pm 1 \quad 0 \quad 0]$ of a magnet is stable in the absence of noise. Any magnitude of current

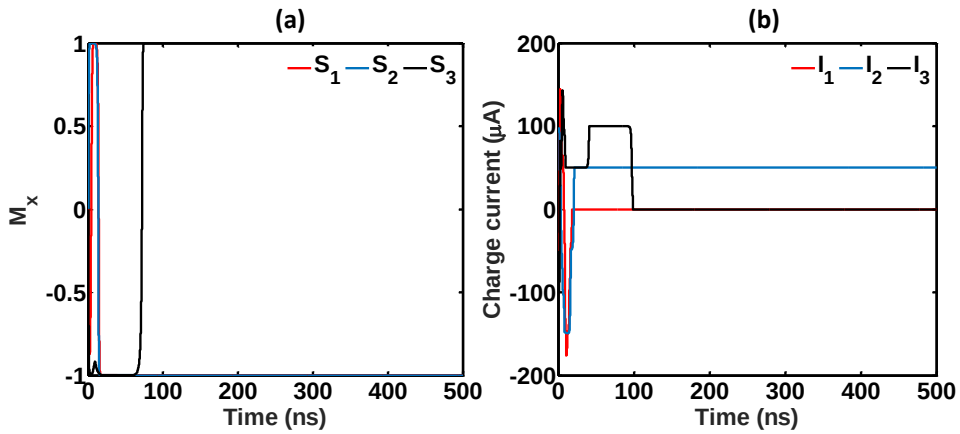

Fig. 2.13. The effect of absence of thermal noise on the operation of the SAT solver. (a) The states of 3 MTJs that represent three s variables (S_1, S_2, S_3) when the thermal noise is not present. The system seems as if it has stabilized even though the solution is not correct. (b) The charge current through the same three MTJs in (a). A positive current drives an MTJ towards the AP state (+1) and a negative current will drive it towards the P state (−1). Note that the positive current through the second device is insufficient to flip the state

(other than ∞) is not capable of switching the magnet as long as the magnetization is perfectly aligned with the positive or negative $\hat{x}$ direction (easy axis). In our proposed system, the switching between the two stable states of a set of MTJs should continuously occur until the system converges to a solution. However, when the thermal noise is not present, a magnetization can end up in the $[m_x\hat{x} \quad m_y\hat{y} \quad m_z\hat{z}] = [\pm 1 \quad 0 \quad 0]$ state. If the state of that MTJ must be changed in the process of converging to a solution, it will be impossible. When thermal noise is present, it will induce a slight tilt in the FL magnetization with the $\hat{x}$ direction. When trying to switch an MTJ using a current, this tilt of the FL magnetization has a significant contribution. The switching time between two states of an MTJ is a strong function of this tilt [86]. When the tilt is zero, the switching time can be explained as infinite. Figure 2.13 illustrates the time evolution of the states of three MTJs, that represent three variables when solving a 3-SAT problem with 20 variables. The effect of thermal noise is ig-

nored for this case. We have forced an initial condition for the system to $M_x = 0$ (*i.e.* the resistance values of all the MTJs that represent the variable s is $(R_{ap} + R_p)/2$). This can be done by passing a current through the heavy metal, perpendicular to the write current direction in figure 1.3 (c). Even though the states show that they have stabilized (note S_1, S_2, and S_3 after 75ns in figure 2.13(a)), there exists a current through the heavy metal layer of the second device (note I_2 is not zero after 75ns in figure 2.13(b)). This current tries to drive the state of the MTJ towards the other direction. Note, a positive current drives an MTJ towards the AP state (+1) and a negative current will drive it towards the P state (−1). Despite the presence of current (I_2) through the second device, the magnet does not flip to [+1 0 0] state. This is because the tilt the FL magnetization has with the easy axis is too small, and the current is not sufficient to drive the state towards [+1 0 0]. This is a 'frozen non-solution state' that occurs only in the absence of thermal noise. Therefore, it is evident that the system fails under zero thermal noise.

In the next two subsections, it will be explained how the thermal noise assists in avoiding limit cycles and the impacts of larger thermal noise on the time to converge to a solution.

Not behaving as a chaotic dynamic system and absence of limit cycles.

We define the states of all the MTJs that represent variable s as $H_N = [-1, 1]^N$. The parallel state is mapped to −1 and the anti-parallel state is mapped to +1. The boundary of H_N is the N hypercube Q_N, with vertices $V_N = \{-1, 1\}^N$. The solution space to a particular problem can be a subset of these V_N. Let us denote such a solution by ${V_N}^{sol} = \{{V_N}^i, {V_N}^j, ...\}$. Due to the effect of thermal noise, the solution to which the system will ultimately converge has minimal dependency with the initial states of the MTJs. For example, let us consider a SAT problem that has multiple solutions V_N^{sol} and solving it in two trials with the same initial states of MTJs. The output solutions in the two trials may not be the same even though the starting

conditions were identical. This implies that our system does not show any chaotic behavior (it is not deterministic) in contrast to the system given by equations 2.2 and 2.3.

If the states of the MTJs that represent variable s continuously change in a periodic manner (it has entered a limit cycle), it is possible that the system never reaches a solution (even if there exists one). That is, $s_i(t) = s_i(t + nT)$ for $\forall n = 1, 2, ...$ and $s_i(t)$ is not a solution of the system. This is known as the system getting trapped in a limit cycle [62]. It is shown that for the system explained in equations 2.2 - 2.3, this can occur for certain coupling parameter (A, B) choices [9]. However, due to the added stochasticity from the thermal noise, we argue that our MTJ based SAT solver does not get trapped in limit cycles. It is highly probable that our system reaches a solution if sufficient amount of time is provided.

Increased temperature resulting in faster solution convergence

Now let us consider how different amounts of thermal noise will affect the operation of our MTJ based SAT solver. Temperature can affect the amount of thermal noise applied on an MTJ device (equation 2.35). Sufficiently higher temperatures on MTJs with smaller switching energy barriers, can cause the state of an MTJ to oscillate over time, even without any input current or magnetic field [63]. We solved randomly generated 20 variable, 3-SAT problems at different temperatures and observed the percentage of problems that can be solved within $10\mu s$. As figure 2.14(a) illustrates, it is evident that in the range of $20°C - 130°C$, there is no significant degradation in the percentage of problems solved within $10\mu s$. However, according to figure 2.14(b), it appears that the average time to solve a k-SAT problem decreases by $\sim 100ns$ with increasing temperature in the range $20°C - 130°C$.

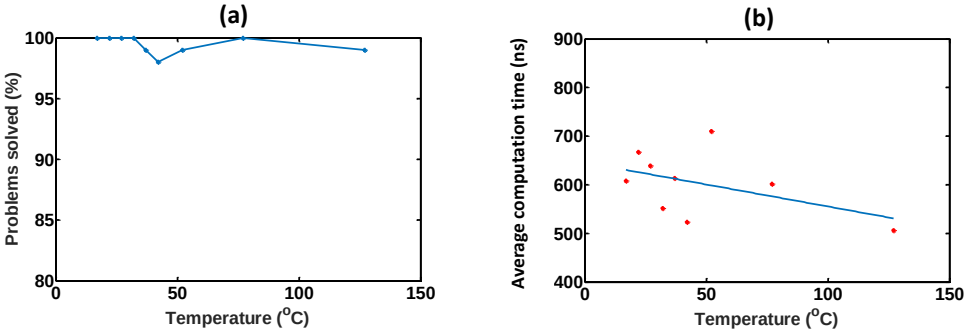

Fig. 2.14. (a) The percentage of randomly generated 20 variable SAT problems ($\alpha_c = 4.25$) solved at different temperatures within $10\mu s$. (b) Average time to solve a 20 variable SAT problem at different temperatures.

2.6.2 Effect of process variations

In our design, we selected a particular thickness (t_{ss}) for the free layers of the MTJs. In reality, it is impossible to achieve the exact thickness due to number of reasons including atomic limitations, process variations, fabrication limitations *etc.* How much precession is present along the easy axis is dependent upon how much the actual thickness has deviated from t_{ss}. In order to observe the effects of thickness variations on the operation of our proposed SAT solver, we consider two scenarios.

1. Global variation of thickness
2. Local variation of thickness.

In the first case, we perturb all the thicknesses of MTJs in our system by some constant percentage from t_{ss}. For a particular percentage global variation in thickness, we solved randomly generated 20 variable, SAT problems. Then the percentage of problems solvable within $10\mu s$, and the average time to solve a single problem was observed. Figure 2.15 (a) illustrates that there is no significant change in the percentage of solvable problems when the global variations in thickness is changed from -5% to $+10\%$. However, as figure 2.15 (b) shows, the average convergence time

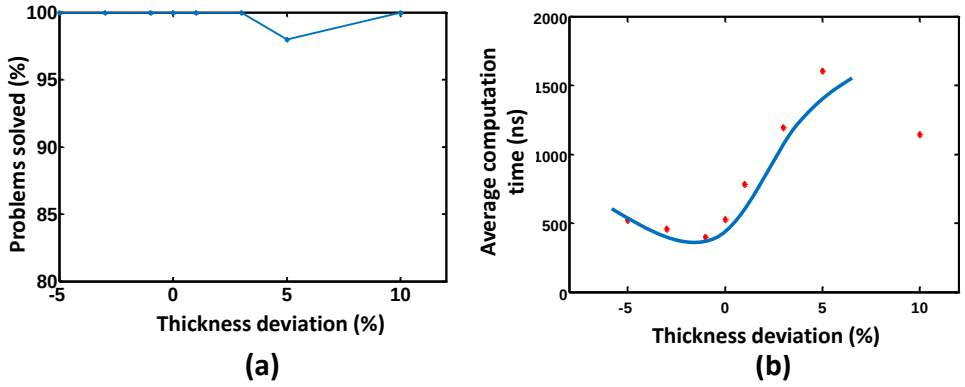

Fig. 2.15. (a) The percentage of randomly generated 20 variable SAT problems ($\alpha_c = 4.25$) solved at different percentage variations in thickness (from the seamless switching thickness, t_{ss}) of the free layer within $10\mu s$. (b) Average time to solve a 20 variable SAT problem at different percentage variations in thickness of the free layer. The variations are considered as global. *i.e.*, all the devices in the SAT solver has the thickness with same deviation from t_{ss}

increases when the deviations in thickness increases. We also observed that the solver no longer works if the thickness is less than a particular value. This is the limit at which the perpendicular magnetic anisotropy (PMA) becomes dominant and the FL magnetization stabilizes in $\hat{z}$ axis instead of $\hat{x}$ axis. We observed this when the actual thickness is $\sim -10\%$ deviated from the t_{ss}, for the choice of materials and dimensions used in this work. In the second case, we change thicknesses of all the MTJs according to a Gaussian distribution with a 3σ value (where σ is the standard deviation) of 10% from the t_{ss}. The solver gave an average convergence time of $588.31ns$ and was able to solve 97% of randomly generated 20 variable, 3-SAT instances within $10\mu s$.

During the fabrication process, non-idealities such as edge damage [87, 88] can be introduced to the MTJs. Such non-idealities can change the parameters associated with the devices and might affect the performance of the proposed SAT solver. We investigate the effect of global variations of three parameters, *viz.* interface anisotropy energy density constant (K_i), width and length.

The changes in K_i due to edge damage can be modeled as a linearly varying K_i over a small length from the edge towards the center of the MTJ [87]. For the experiments, we assumed that the K_i variations near the edge can be represented by an effective K_i (K_i^{eff}) throughout the magnet. Figure 2.16 shows how the computation time changes due to global variations in the K_i^{eff} as a percentage from the nominal K_i.

We observed that the increased/decreased K_i^{eff} results in increased computation time. This is due to the fact that t_{ss} is dependent upon K_i (equation 2.3). Since increased K_i^{eff} results in increased t_{ss} and vice versa, the trend in figure 2.16 is similar to the mirror image (along y axis) of the 'computation time vs thickness' curve in figure 2.15(b). However, figure 2.16 (b) shows that the percentage of problems solved does not have a significant impact from the K_i^{eff} variations.

We further observed how the changes in shape anisotropy (due to changes in dimensions) affect the performance of our MTJ based SAT solver. We induced global variations in the range of $\pm 15\%$ to the widths and lengths of the MTJs and observed the change in computation time by means of 20-variable hard SAT instances. Figure 2.17 summarizes the results obtained. Figure 2.17 (a) and (c) illustrate that, decreased dimensions decreases the computation time and vice versa. This is due to the reduction in the energy barrier with reduced dimensions. We thus conclude that the reduced dimensions improves the performance of our solver.

Intuitively, since the non-ideal effects such as edge damage can potentially reduce the switching current of a magnet [87], the computation time of our system must reduce. If an edge damage free-MTJ switches within t_1 amount of time due to an I_1 current, a magnet with edge damage switches faster than t_1 for the same current, I_1. To view the above effect on computation time of our solver, we increased the applied current on all MTJs by 10%. This resulted in an average computation time reduction of about 36% for 20-variable hard SAT problems.

As explained in prior work [87], the effects of edge damage is minimal on MTJs with larger dimensions. Furthermore, it has been shown [88] that using smaller accelerating voltages during the ion milling fabrication process reduces the edge damage

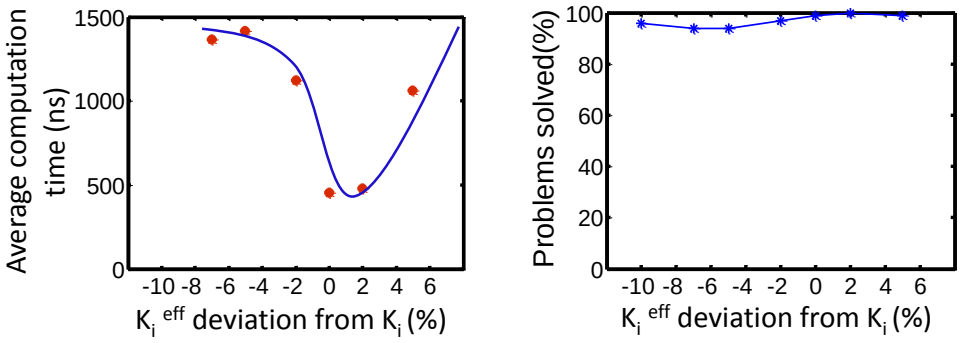

Fig. 2.16. The effect of global variations in interface anisotropy energy density constant K_i, over the (a) computation time of the system, and (b) the percentage of 20-variable SAT instances solved within 10μs

as well. These counter measures can be exploited to avoid consequent performance degradation of the proposed SAT solver if required.

2.7 Power consumption and computation time of the SAT solver

In this section, we will present the power consumption and computation time of our proposed system, and compare with existing methodologies. In order to calculate the power consumption, we used SPICE simulations in IBM 45nm technology. The measured average power consumption over solving $1,000$, 20-variable (our software based LLGS solving framework could not handle bigger benchmarks such as 'ais8' [89]) hard SAT problems (constraint density $\alpha_c = 4.25$) was 1.37mW. The power requirements of each section of the solver are presented in Table 1. We observed that the peripheral circuits such as amplifiers and voltage controlled current drivers consume a significant portion of power. The power consumption of the total MTJ and heavy metal layer structures is approximately about 20% of the total power consumption of the system. Similarly, a recent work of a hardware realization [69] of an analog approach [62], also explains that their overall power consumption to be significant (numerical value not specified), due to the usage of op-amps. In another

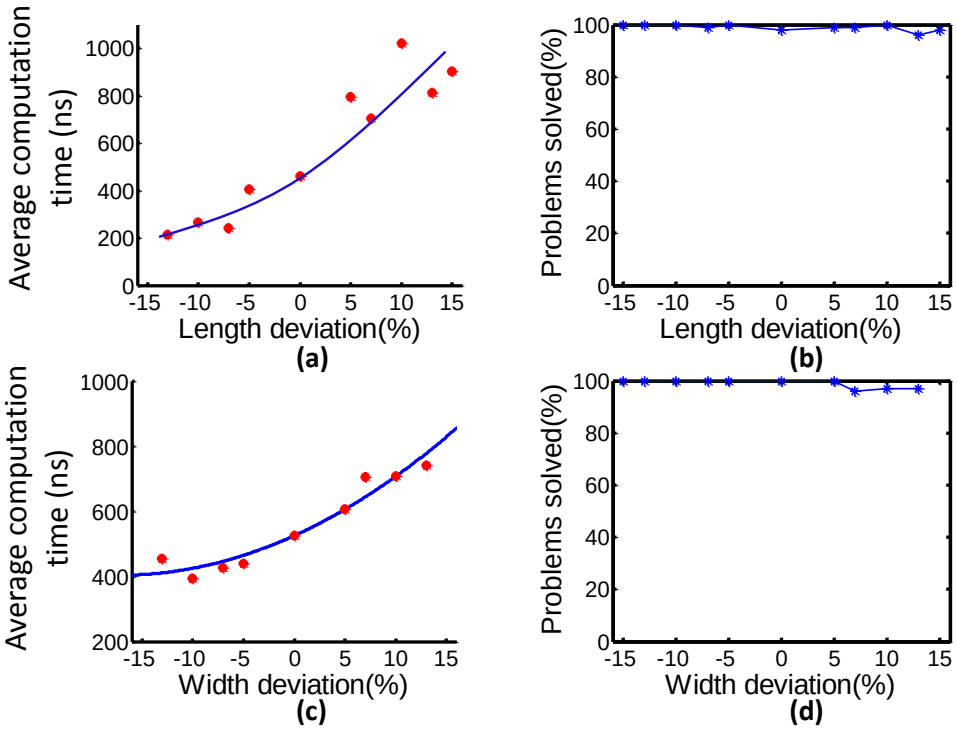

Fig. 2.17. The effect of global variations in lengths and widths of MTJs over the computation time of the system, and the percentage of 20-variable SAT instances solved within 10μs. (a),(b) shows the effects on computation time and percentage of problems solved when the length is varied from -15% to $+15\%$, with respect to the nominal length. (c) and (d) respectively show the effects on computation time and percentage of problems solved, when the width of the free layer is varied

design [68], the CeNN based SAT solving algorithm [9] used in our work was realized using op-amp based integrators. The power consumption was reported as 140μW for a 4-variable, 4-clause problem. For comparison, we evaluated the average power consumption of our MTJ based solver for similar 4-variable 4-clause SAT instances, and witnessed a power consumption of 84μW, which is about 40% smaller than the aforementioned analog hardware design [68].

Digital hardware for realizing typical SAT solving algorithms such as GRASP [64], DPLL [90] *etc.* can be found in literature. [91–93]. A custom IC [91] designed for such an algorithm reported a power consumption of 871μW. This is about 37% lower than

Table 2.1.
Power consumption of the MTJ based SAT solver

Component	Power consumption
Read circuitry	300nW
Amplifiers	315μW
Voltage controlled HM current drivers	827μW
Driving current through HM	235μW
Total	1377μW

our design. However, these digital approaches are slower than analog solvers due to the step-by-step decision making and backtracking required for solving a problem. It is also noteworthy that our proposal is an asynchronous method in contrast to the above synchronous methods. Therefore, the need for a clock signal and the power associated with it is not present in our design. Our system automatically goes to a minimum energy point (which is a solution) and stabilizes there as explained mathematically in section 2.3.

In order to obtain the computation time of our system, we conducted system level simulations using random $1,000$, 20-variable hard SAT instances. The time taken to solve all $1,000$ problems (100%) were evaluated and the average computation time per instance was 553ns. This computation time was then compared with a state-of-the-art software SAT solver, Minisat [95]. The same $1,000$ problems were solved using Minisat in a 3.6GHz processor and the average computation time was evaluated (1.44ms). We observed an average speed-up of 2.6×10^3 with respect to Minisat. Prior hardware SAT solver designs in literature have also presented their speed-up with respect to Minisat, and the values are summarized in Table 2 for comparison. Note that the

Table 2.2.
Speed-up of hardware based SAT solvers with respect to purely software based solvers

Hardware solver	Speed-up with respect to software based solvers
Reconfigurable SAT solver [92]	90 (3.6GHz processor)
BCP accelerator [94]	6.7 (3.6GHz processor)
BCP accelerator [93]	4 (3.3GHz processor)
MTJ based solver	2.6×10^3 (3.6GHz processor)

Boolean Constraint Propagation (BCP) accelerator [94] in Table 2 has reported the speed up with respect to a purely software based algorithm (modified zChaff [63]) which has the same performance for BCP when compared with Minisat. Furthermore, the analog CeNN based system [9] built using op-amp based integrators [68] has an average computation time of 15μs for 10 variable, 2-SAT problems. Our MTJ based proposal has an average run time (over $1,000$ instances) of 186ns for 10 variable 3-sat ($\alpha_c = 4.25$) problems (showing that the MTJ based solver is $\sim 30\times$ faster).

2.8 Conclusion

Boolean satisfiability is an NP-complete problem ($k \geq 3$) that finds utility in vast array of applications [55–59]. Analog solutions to the satisfiability problem has recently appeared attractive [9, 62, 68] due to the massive parallelism available when solving, in contrast to the stepwise search algorithms. In this work, we provide a proof of concept hardware based analog SAT-solver using Magnetic Tunnel Junctions driven by the Spin Orbit Torque Phenomenon. We have mathematically shown how the inherent device physics of MTJs closely mimics an existing analog approach [9] to solving the Boolean satisfiability problem. Device and circuit level simulations were conducted to solve hard satisfiability problems in order to observe the performance and functionality of our proposed system. According to the observations, we witnessed that the proposed SAT solver is capable of finding a solution to a significant fraction (>85%) of hard SAT problems in polynomial time. We conjecture that this is due to the inherent thermal noise present in MTJs and the device complexities added on top of the existing analog approach [9]. The SAT solver automatically comes out of local minimum points and limit cycles due to thermal noise. Therefore, it is highly probable that the system reaches a solution if sufficient time is provided, given that the SAT problem has a solution. Further, the variation analysis illustrates that our proposed solver is robust to variations in the MTJ thickness in the range of -5% to 10%. Larger variations result in higher average convergence time. The proposed

MTJ based SAT solver is 2.6×10^3 times faster than a state-of-the-art software solver, Minisat. The work presented in this section is published in [96]

3. ALL-MEMRISTOR STOCHASTIC DEEP SPIKING NEURAL NETWORKS

Even though the exact mechanisms of communication between biological neurons still remain unknown, it has been shown experimentally that neurons use spikes for communication and the nature of the firing of neurons (spike generation) is non-deterministic [3–5]. By conserving energy via spike based operation [10], the brain has evolved to achieve its prodigious signal-processing capabilities using orders of magnitude less power than the state-of-the-art supercomputers. Therefore, with the intention to pave pathways to low power neuromorphic computing, much consideration is given to more realistic artificial brain modeling [97]. The inception of the Spiking Neural Networks (SNN) concept is a consequence of above. It has recently emerged as an active area of research owing to its resemblance of the "actual human brain" [46].

In a spiking neural network, the communication between neurons take place by means of spikes. The information is typically encoded in the rate of occurrence of spikes. Different learning schemes have been proposed over the past, and Spike Time Dependent Plasticity (STDP) based learning is widely used due to the consistency of the concept with experimental statistics [98]. However, the STDP learning is typically limited to a network with a single layer of excitatory neurons and a single layer of inhibitory neurons [99]. The aptitudes of such a single fully connected layer spiking neural network is limited when compared with the high recognition performances offered by deep ANNs. Up to date, deep ANNs have given the best performance with respect to classification accuracy. For an example, SENet which won the 2017 ILSVRC, is a deep Convolutional Neural Network (CNN) with the reported lowest top 5 error (the correct class is not within the top 5 selection of classes according to the network output) of 2.251% on ImageNet data set [2]. However, such networks require

huge power and time if a von-Neumann computer is to be used for computation. For an instance, SE-ResNet requires power for ∼3.2GFLOPS (number of operations per second) [2].

As an effort of embedding the classification accuracy of such ANNs with the spike based low power operation of SNNs, numerous research efforts have been focused on crafting Deep Spiking ANNs [100]. One of the interesting mechanisms of executing the above is by exploiting certain optimization techniques to convert a fully trained deep ANN to an SNN [101,102]. The work suggested in [101] outperforms all previous SNN architectures to date on MNIST database. Despite the existence of Deep SNNs in the algorithmic level, minimal consideration is dedicated towards devices and realizing such algorithms in hardware level.

With the intention to reduce the energy consumption of the powerful Deep ANNs while preserving the biological plausibility, a non-deterministic, memristive device based hardware architecture for a Deep Stochastic SNN is proposed here. Memristors have been widely used in literature as synapses in neural networks [44,45]. The multi-level storage capability has made the memristor an ideal candidate for the synapses in a neural network. Even though this multi-level behavior of the memristors seems appealing to emulate the behavior of an analog neuron as well (different voltage write pulses (inputs) result in different memristor resistances (output); there is an upper and a lower bound for the resistances; this signals similar functionality of a thresholding function), reliability concerns might arise due to the inherent stochasticity. This stochasticity in memristors has been experimentally shown [49,51] and the statistical measures suggest that the switching probability of these devices can be predicted. For example, the switching times follow a Poisson distribution for Silver/amorphous Silicon/poly Silicon ($Ag/a-Si/p-Si$) based devices.

Memristors offer a variety of favorable features such as higher write-erase cycles (10^{12} [103]), higher yield, CMOS compatibility, lower area *etc.* Despite these benefits, high programming voltages and long pulse durations, [104] or other feedback write mechanisms [105] are mandatory to ensure the switching of the devices, for

applications such as memory, that require very low failure rates. Rather than trying to reduce such non-deterministic effects, in this work we propose an effort to embrace the stochasticity in an efficient way, with the ambition to go towards a more realistic neuron. We propose the memristor as a probabilistic switch to represent the stochastic neuron in a supervised deep stochastic spiking neural network, and memristive crossbar arrays with multi-bit capability to represent the 'inner product' computation between the incoming spikes and the synaptic weights. We introduce this structure as 'All-Memristor' neural network due to the fact that the two main functionalities of a neural network are being taken care of by memristors. We elaborate how the ANNs can be trained, in order to incorporate a stochastic memristor as a neuron. The gradient descent based backward propagation scheme must be modified to account for the probabilistic function which may be different from standard activation functions (ReLU, sigmoid $\left(\frac{1}{1+e^{-x}}\right)$, *etc.*) of a neuron. We will further elaborate certain favorable features accompanied by memristors that makes it suitable to emulate a stochastic neuron. We propose circuit architectures that can be used to realize the proposed All-Memristor network. Then the impact of certain variations towards the accuracy of the network is explored. Finally the energy consumption of the All-Memristor based network is compared with the CMOS counterpart.

Even though the possibility of harnessing the inherent stochasticity of the memristor for neuromorphic computations has been mentioned previously [106–108], the complete analysis of it for deep stochastic SNNs has not yet been studied. Further, stochastic integrate and fire neurons (with a focus on devices) have been proposed in literature [109, 110] for unsupervised learning SNNs and they are different from this work, where we have specifically designed the neuron to suit deep supervised neural networks which are capable of performing complex tasks with better accuracy [101]. A complete analysis of hardware neural network with memristors is provided here, to which four key features of the brain; high accuracy, low power, spike based information transferring, and stochasticity are embedded.

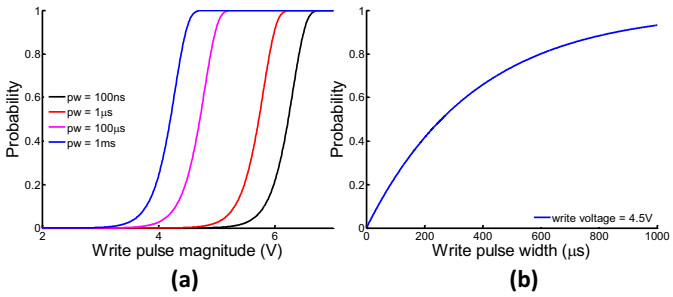

Fig. 3.1. (a) The switching probability of a memristor with varying magnitude of the pulse. (b) The Switching probability of a memristor with varying pulse width

3.1 Stochasticity in memristor devices

Despite the copious favorable features offered by memristive devices, the stochasticity of changing its state has induced reliability concerns. To make the memristor a deterministic device in order to appropriate it for applications such as non-volatile memories, reconfigurable logic *etc.*, significant consideration must be provided to the operating region of the devices. As an example, the TiO_2 based memristive devices have a typical threshold voltage of 1V in order to 'SET' the device [111] for memory applications. This is the magnitude of the voltage write pulse that provides a higher confidence (ex: > 0.99) of writing a logic 1. If a writing pulse of 0.5V is applied instead of 1V, the device may change its state with a certain probability which is less than 1. It is evident that increasing the reliability comes at the cost of high power consumption. Our work is an effort to utilize the stochastic behavior of the nanoscale resistive devices while operating in the low-power non-deterministic regime.

The formation of the filament in a ECM type memristor is a highly voltage bias dependent process. The anode metal particle hopping rate is given by [47]

$$\Gamma = \frac{1}{\tau} = \nu e^{-E'_a(V)/k_B T} \tag{3.1}$$

where k_B is the Boltzmann's constant, T is the temperature, τ is the characteristic switching time, and ν is the attempt frequency. $-E'_a(V)$ is the bias dependent

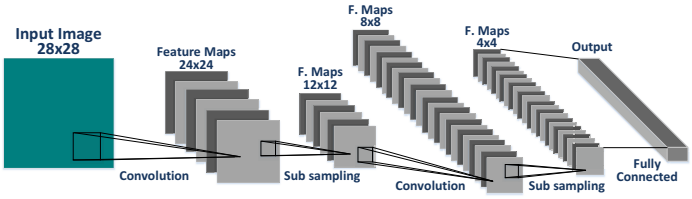

Fig. 3.2. The typical structure of a convolutional neural network. There are two main sections in a CNN in terms of functionality. The convolutional layers (followed by subsampling to reduce the large number of computations) perform the feature extractions from an input (ex: image), and the fully connected layer classifies the inputs depending upon the extracted features.

activation energy. The time required for the formation of the metal filament is shown to follow a Poisson distribution [51]. The probability of switching within the next Δt duration after a t amount of time can be defined as

$$P(t) = \frac{\Delta t}{\tau} e^{-t/\tau} \tag{3.2}$$

The dependency between the characteristic switching time and the voltage of the write pulse is given by

$$\tau(V) = \tau_0 e^{-V/V_0} \tag{3.3}$$

$$\tau_0 = \frac{1}{\nu} e^{E_a/k_B T}, \quad V_0 = 2nk_B T/q \tag{3.4}$$

where E_a is the activation energy at zero voltage bias, n is the number of anode metal particle sites, q is the charge of an electron.

If a particular write voltage pulse is applied on the memristor, according to above equations, it can be noted that the switching probability depends on two key factors.

1. The magnitude of the pulse.

2. The width of the pulse

Fig. 3.1 (a) shows how the magnitude of the write pulse affects the switching probability curve and Fig. 3.1 (b) shows the effect of the width of the write pulse. For a rate based spiking neural network, if a memristor must be incorporated as a spiking neuron, the width of the spikes must ideally be the same (variations might be present and their effect is analyzed in the results section). Therefore, the magnitude of the pulses must be controlled to bring the memristor to its stochastic operating regime. From equation (3.2), the cumulative probability of switching when a voltage V is applied on the memristor for a t amount of time is

$$P = 1 - e^{-t/\tau} = 1 - exp(-\frac{t\nu}{e^{E_a/k_BT}} e^{Vq/2nk_BT}) \tag{3.5}$$

Once the write time t is selected for the network, all the parameters in the above function is fixed (i.e., $P = f(V)$).

3.2 Deep Stochastic Spiking Neural Networks

3.2.1 Convolutional neural networks basics

CNNs have multiple hidden layers of neurons between the input and output layers. For an example, the CNN in Fig. 3.2 has 2 convolution layers, two subsampling layers and one fully connected layer. Each convolution and fully connected layer involves calculating the summation of some weighted inputs and then sending the outcome of it through an activation function. This output is fed as an input to the next layer. Calculating the set of synaptic weight values is called training and stochastic gradient descent method is usually used to back-propagate the error at the output and update the weight values. Typical activation functions for a CNN include sigmoid function, tan hyperbolic function and rectified linear function. The activation function for the stochastic neuron in this work is a probabilistic function as will be described in the next section.

3.2.2 Stochastic neurons

Let us first consider an analog neuron with an activation function f. The input to the neuron is the weighted summation of the set of outputs from the previous layer. The output of the neuron can be given as

$$y = f(x \cdot w) \tag{3.6}$$

where x is the output vector from the previous layer and w is the set of synaptic weights. The output varies between 0 and 1. Therefore, x can be in the form of $x = [0, 1]^N$ with N being the number of fan-in neurons. In contrast, the neurons in spiking neural networks, communicate in terms of Poisson spike trains. *i.e.*, instead of the analog input vector x, we would have $\tilde{x}(t) = \{0, 1\}^N$ where 1 represents a spiking event and 0 represents a non spiking event. In an integrate and fire neuron or leaky integrate and fire neuron, the activities at the inputs are integrated over time until the accumulated value (membrane potential) reaches a certain threshold value. Once this threshold value is crossed, the neuron will produce an output spike (neuron fires) and reset the membrane potential. The stochastic spiking neuron that is being discussed in this work does not temporally accumulate the spiking activities until it reaches a predefined threshold. Instead, it incorporates a probability function that observes the spiking activities at the input from the pre-layer neurons during a time step, and produce a spike with a certain probability that depends on the weighted summation of these activities. Throughout this document, the 'spiking neuron' term refers to the above context.

3.2.3 ANN to stochastic SNN conversion

In this section, we elaborate on the conversion of an ANN to an SNN and its associated error. There are two types of spiking networks in terms of the way the data is encoded in the pulses. One method is encoding the information in the exact time a spike occurs. In this work, we are considering the second method where the

information is included in the rate of the spikes. When converting an ANN to such an SNN, the analog input of an ANN must be rate encoded as a Poisson spike train. The expected value of the input spike events can be elaborated as (for $N = 1$)

$$\langle \tilde{x}(t) \rangle = \frac{\sum_t \tilde{x}(t)}{T} = x \tag{3.7}$$

where T is a sufficiently large number of time steps. Let us assume that we have considered a probabilistic activation function similar to the analog activation function f (or in other words, consider that the ANN was trained with a function similar to the probability curve f of a device). Here the neuron gives an output spike with a certain probability defined by f depending upon the input events (spiking/not spiking). When there is a spike at time t, $\tilde{x}(t) = 1$. The corresponding probability of getting a spike at the output is $\tilde{y}(t) = f(\tilde{x}(t) \cdot w) = f(w)$, where w is the synaptic weight. Similarly, when there is no spike at time t, $\tilde{x}(t) = 0$ and the probability of getting a spike at the output is $\tilde{y}(t) = f(0)$. The expected output can be elaborated as

$$\langle \tilde{y}(t) \rangle = x \cdot f(1 \cdot w) + (1 - x) \cdot f(0 \cdot w) \tag{3.8}$$

As we explained in section 3.1, the probability of switching a memristor with a voltage pulse of constant pulse width and the input $(x \cdot w)$ encoded as the magnitude, takes the form of (for the exact relationship, refer to equation (3.5). We have refrained from using the extra constants for simplicity of elaboration and understanding. These constants does not affect the concepts that are being discussed in this section)

$$f(x \cdot w) = 1 - exp(-e^{x \cdot w}) \tag{3.9}$$

Therefore the expected value of the output is

$$\langle \tilde{y}(t) \rangle = x \cdot \Big(1 - exp\big(-e^{w}\big)\Big) + \big(1 - x\big)\big(1 - e^{-1}\big) \tag{3.10}$$

For an ideal ANN to SNN mapping (one to one mapping), this expected value must be similar to $f(x \cdot w)$. However, as the above equation suggests, $\langle \tilde{y}(t) \rangle$ takes a linear form with input x. As explained in [112], the difference between $f(x \cdot w)$ and

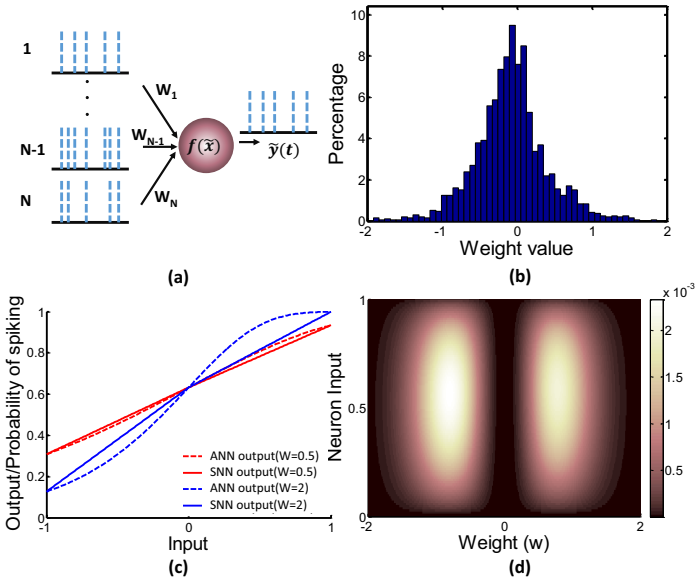

Fig. 3.3. (a)The stochastic spiking neuron structure. Incoming inputs are spikes. Depending upon the weighted summation, of spikes, the occurrence of a spike at the output will be determined according to the probability function f. (b) The distribution of weights of a network when trained according to a device level probabilistic curve assuming an analog behavior. (c) The expected value of the SNN neuron output for different expected values at the input, and ANN neuron output for different inputs. We have considered two weight values; 0.5 and 2. Negative inputs refer to the cases where the synaptic weight value is negative (d) The ANN to SNN conversion error distribution. Maximum error appears close to 0.5 input spike rate and $|w| = 1$. However, the error value is not very significant.

$\langle \tilde{y}(t) \rangle$ grows with the increasing weight value (Fig. 3.3 (c)). However, according to the distribution of weights illustrated in Fig. 3.3 (b) (for a deep ANN trained with the activation function f), all the weight values come under the window of $|w| < 2$. We can now get an estimate for the error when mapping the ANN to an SNN, assuming the probability of having any spiking rate $\langle \tilde{x}(t) \rangle = x = [0, 1]$ is equally likely (uniform distribution). Fig. 3.3 (d) shows the error when we consider having a weight value in the range $|w| < 2$ according to the distribution in Fig. 3.3 (b).

3.2.4 Training the ANN before converting to an SNN

As mentioned in the previous section, we use an activation function similar to the switching probability curve of a memristor given by equation (3.9). The weight update rule should change according to this activation function. The stochastic gradient descent weight update rule is as follows

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} \tag{3.11}$$

$$o_j = f(net_j) \tag{3.12}$$

Where E is the cost function that must be minimized for a given input (preferably the squared error at the output). o_j is the output of the j^{th} neuron, net_j is the weighted summation of inputs coming into the j^{th} neuron and η is the learning rate. The term $\frac{\partial o_j}{\partial net_j}$ changes according to the following equation due to the choice of our device defined activation function.

$$\frac{\partial o_j}{\partial net_j} = (o_j - 1) ln(1 - o_j) \tag{3.13}$$

The bias values in the network are considered to be constant and do not get updated during training. The constant value corresponds to the probability of switching at the output of the neuron during an event of 'no spike'. Any output probability during a no spike event can be selected by properly adjusting this bias value.

3.3 'All memristor' stochastic SNN architecture

In this work, we consider a deep spiking convolutional neural network which is trained offline, using the switching probability curve explained in the previous section. All the synaptic weights are realized by the conductance of multi-level memristors (4-b discretization levels were used for this case [112–114]. A multilevel writing scheme was proposed in [52] for TiO_2 based memristors using the model in [115]. The authors claim that the system can write any number of levels given that the on/off ratio is

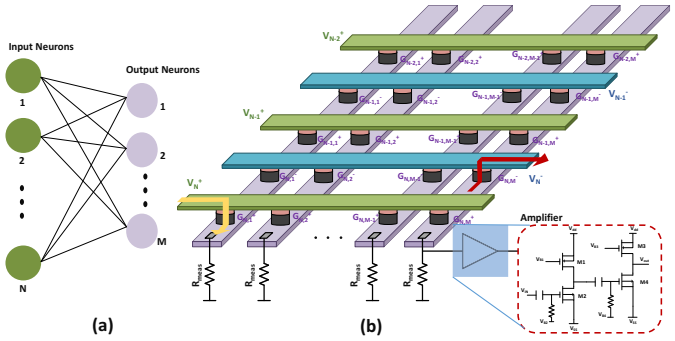

Fig. 3.4. (a) A schematic of a fully connected layer in a neural network with N input neurons and M output neurons. (b) The crossbar structure that represents the inner product functionality between the incoming spikes and the synaptic weights

high). The spike trains are short voltage pulses. The inner product between the incoming spikes and the synaptic weights at time t can be efficiently calculated by using a crossbar structure. Let $V(t) = \{0,1\}^N$ be the incoming voltage spikes from N neurons towards the $N \times M$ crossbar (N pre-layer neurons, M post-layer neurons in a fully connected structure). If a conductance value in the crossbar is $G_{i,j}$, then the inner-product between the voltage pulses, and the conductances of the memristors connected to the j^{th} metal column, is the current that flows through the j^{th} metal column itself ($I_j(t)$).

$$I_j(t) = V(t) \cdot [G_{1,j}, G_{2,j} ... G_{N,j}]^T \tag{3.14}$$

Ideally, the above value must be converted to a proportional voltage that can bring the memristor to the 'stochastic regime' as explained previously. This can be done by sending the above current through a resistor and appropriately amplifying the voltage across it. However, incorporating such measuring resistors (R_{meas}) cause non-ideal inner products [112]. Therefore, the measuring resistance must be made considerably small with respect to the values of other resistors that emulate the synaptic functionality. A crossbar coupled with measuring resistors is shown in Fig. 3.4. Simple low power amplifiers can be incorporated to amplify the voltage across the measuring resistor (Fig. 3.4) as required. The input impedance of the amplifier is

very large. The output impedance is comparatively smaller than the off-resistances of a memristor. The output voltage of the amplifier is biased to give the same probability that the network is trained for (refer to the explanation in section 3.2 D) during an event of no spike. The negative weights are realized by conditionally selecting between positive and negative voltages as shown in Fig. 3.4. For example, if the weight is negative at the (i, j) cross-point in the cross bar, then the memristor between the i^{th} positive metal row and j^{th} metal column is turned off and vice-versa .

Each time step of operation of the SNN architecture, consists of three key tasks; write, read and reset. The write step involves the calculation of weighted addition of the spike events in a given time step using the crossbar, and applying the corresponding voltage to the memristor. In order to observe whether the memristor has switched, a read phase is carried out. This can be done by a resistor divider circuit as shown in Fig. 3.5. If the memristor switched during the write phase, then the inverter output will be high. Else, it will be low.

Due to the variations in the ON,OFF resistances, the voltage at the resistor divider arrangement can vary. This may lead to erroneous identification of an occurrence of a spike, if a properly designed inverter is not present. To account for such potential errors, an inverter with a sharper characteristic curve and a controlled trip point must be used. The inverter will then identify if the resistance of the neuron memristor has gone below a certain threshold resistance (which is a spiking activity). Fig. 3.6 shows the response of the inverter we used in this work. The input voltage variations due to changes in ON and OFF states are shown in red for a standard deviation (σ) of 20%. We conducted $100,000$ Monte-Carlo simulations to check the number of false identifications of a spike for different values of σ. For $\sigma = 100\%$ in ON resistance, we witnessed a false identification of a spike 0.007% of the time for the inverter characteristics in Fig. 3.6.

The spiking events identified by the inverter, can be stored in buffers until the next time step. After the read phase, all the memristors will be reset to be used in the next time step. Resetting 'all' the memristors is necessary since a write pulse

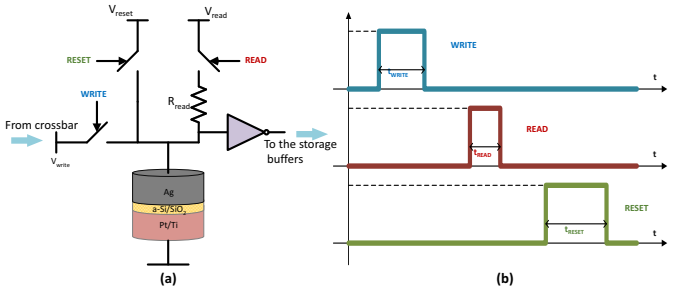

Fig. 3.5. (a)The stochastic memristor neuron (b) The temporal variation of write, read, and reset control signals within a single time step

may contribute to the growth of a conductive filament in a memristor even though it was unable to fully switch the device. Fig. 3.5 (b) shows the aforementioned write, read and reset temporal activities.

3.3.1 Implementation of the network layers using memristive crossbars

In this section, we will discuss how the fully connected, convolutional and subsampling layers are implemented using memristive crossbars. The governing operation in all these layers is the dot product, and a crossbar can be utilized efficiently for this. Fig. 3.4 clearly shows how a fully connected layer with N input neurons and M output neurons can be implemented. The convolutional operation can be implemented as shown in Fig. 3.7. As the figure illustrates, the convolution operation consists of a kernel moving over the image, getting the weighted summation in each location. A single such weighted summation can be implemented by first converting the corresponding section of the image and the kernel in to vectors and then mapping the weight values to memristors in a column in the crossbar as shown in Fig. 3.7. When mapping one entire layer with M_{in} input channels and M_{out} output feature maps, the input must be divided into sections of $k \times k \times M_{in}$ (assuming a kernel size of $k \times k$). The crossbar should also have a similar number of conductances to obtain a single element in the output maps. Multiple such crossbars should be there to account for

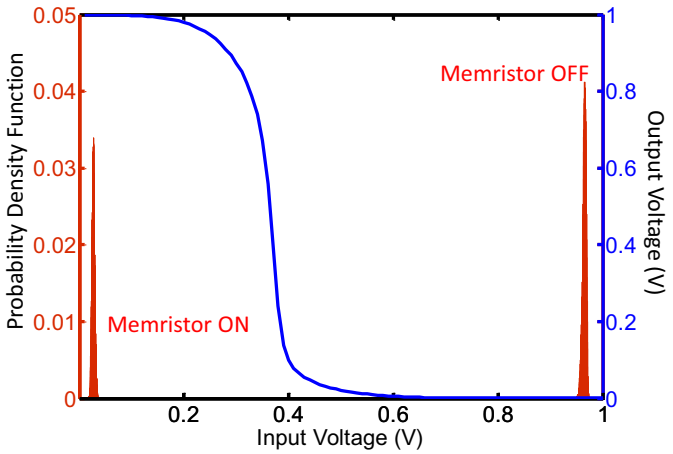

Fig. 3.6. The input and output voltage characteristics of the inverter in the read circuit. The input voltage changes due to variations in ON and OFF resistances (with $\sigma = 20\%$) are shown in red.

all aforementioned input chunks. Subsequently, all the layers in the CNN can be spatially mapped across several crossbars. Consequently, The area consumption of the entire memristive hardware required to execute the CNN can be estimated as the sum of area of all such crossbars and associated peripherals (buffers, amplifiers *etc.*). Additionally, the inference delay is estimated as the time required to sequentially propagate data across the crossbars mapped to the CNN layers. For the calculation of energy consumption, we are doing a circuit level analysis.

At the output of each crossbar, the measuring resistor, the amplifier and read, reset circuitry must be there as shown in Fig. 3.4 and 3.5. In the architectural level, there are multiple ways of dividing the convolution operation into multiple realizable crossbar sizes [116], [117], [118]. When dividing the operation into a number of crossbars, summing amplifiers must be incorporated to add the outputs of multiple crossbars and feed to the memristor neurons.

Subsampling layer (averaging) takes a similar form as above since it performs the convolution operation with a kernel size equal to the scaling factor, and all the kernel elements being equal. Therefore the subsampling layer can be implemented by using

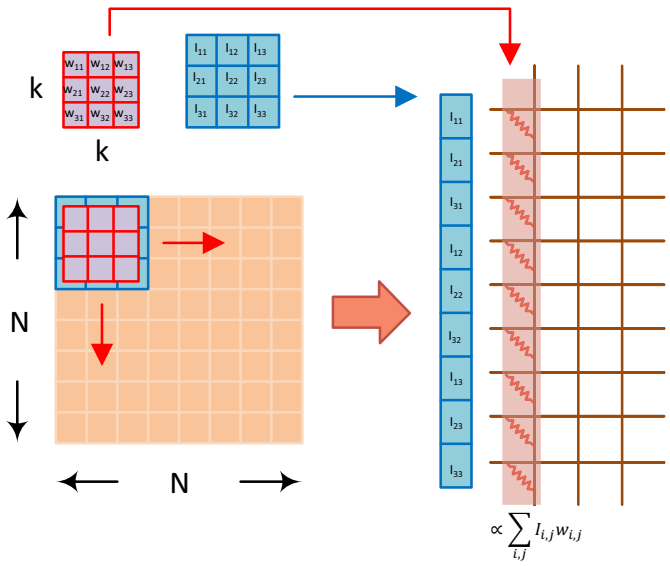

Fig. 3.7. The dot product operation in a convolution layer. Image size is $N \times N$ and the kernel size is $k \times k$

the same convolutional layer architecture. The input channels and the output maps must be selected as equal. The stride in the subsampling layer is equal to the scaling factor. In each filter set, except for the corresponding channel's filter weights, all the other channels' filter weights must be zero. This is due to the fact that there is only one kernel to map each input channel to a single output feature.

3.4 MNIST Data Classification with the Stochastic memristor based neural network

In order to view the functionality of the All-Memristor based deep stochastic SNN, we have created an algorithm-device-circuit framework and tested on a standard digit recognition data set, MNIST. The architecture selected for this work is a convolutional neural network ($28 \times 28 - 6c5 - 2s - 12c5 - 2s - 10o$ [119]). The CNN structure as mentioned in section 3.2 is well known for its high recognition accuracies on complex data sets and we have chosen it for this work to show the applicability of the proposed

devices on state-of-the-art neural networks. It is noteworthy that this proposed device architecture is applicable to any type of ANN (ex: fully connected) since the basic computational blocks (calculating the weighted summation) remain the same. The CNN used in this work has 2 convolutional layers followed by subsampling. Each convolutional kernel is of size 5×5 and there are 6 and 12 feature maps at the output of first and second convolutional layers respectively. The input image is of size 28×28 (an image of a digit in MNIST data set) and the pixel intensity dependent spike activity is fed to the first convolutional layer. The input spikes can also be generated by directly applying voltages to a set of memristors with an amplitude proportional to the intensity of the pixels. The memristors would then generate homogeneous Poisson spike trains proportional to the intensity of the image pixels. After each convolution layer, a subsampling with the kernel size 2x2 is present and this is simply averaging the spiking activity of few neurons. A fully connected layer appears between the second subsampled convolutional layer output and the network output. There are 10 output neurons to account for the 10 digits (classes) in the dataset. The network was trained as an ANN for 60000 images of handwritten digits, with the probabilistic switching curve of a memristor as the activation function of neurons, following the process mentioned in section 3.2. The stochastic memristor neuron model is built according to the set of equations elaborated in section 3.1.

The trained network was then tested on $10,000$ images of handwritten digits as a spiking neural network. Instead of evaluating the outputs of neurons as analog values, the probability of switching is determined according to the voltages applied on the memristors. These voltages depend on the weighted summation of input spikes that goes in to the neuron. We observed the spiking activities at the 10 output neurons over a 100 time steps (each including a write, read, and a reset phase) and the winner is considered as the neuron that gave the highest number of spikes during the total time interval. We obtained a classification accuracy of 97.84% with a write pulse of 10ns, after detecting the spiking activity over 100 time steps. Fig. 3.8 shows the spiking activities at the 10 output neurons over 100 time steps for 5 randomly selected

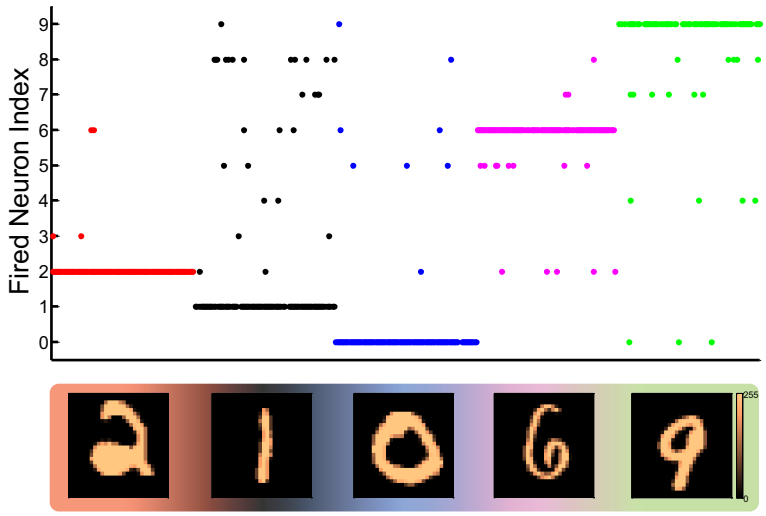

Fig. 3.8. The spiking activities of the 10 output neurons of the All-Memristor neural network over 100 time steps for randomly selected 5 images in the testing data set

images from the testing data set. The accuracy we obtained for this network shows a slight degradation when considered with the baseline ANN with sigmoidal activation functions that provides an accuracy of 98.9%. This degradation is due to the circuit and device related considerations we took in to account while converting the ANN to an SNN. One of the reasons is the fact that we quantized the synaptic weights to suit the currently available multi-level memristors with 4-bit levels. Another reason is the non-ideality due to the inclusion of the measuring resistor described in section 3.2. The ANN to SNN conversion error (section 3.1) has an impact on the accuracy degradation as well.

In the next few subsections, we will discuss about the circuit level implementations and analyze the impact of different types of variations on our All-Memristor deep stochastic SNN.

Providing the accurate voltage to the memristive neuron is important. In order to verify that the correct voltages are being supplied to the neurons, we conducted circuit level simulations for our network. We used IBM 45nm technology node for CMOS

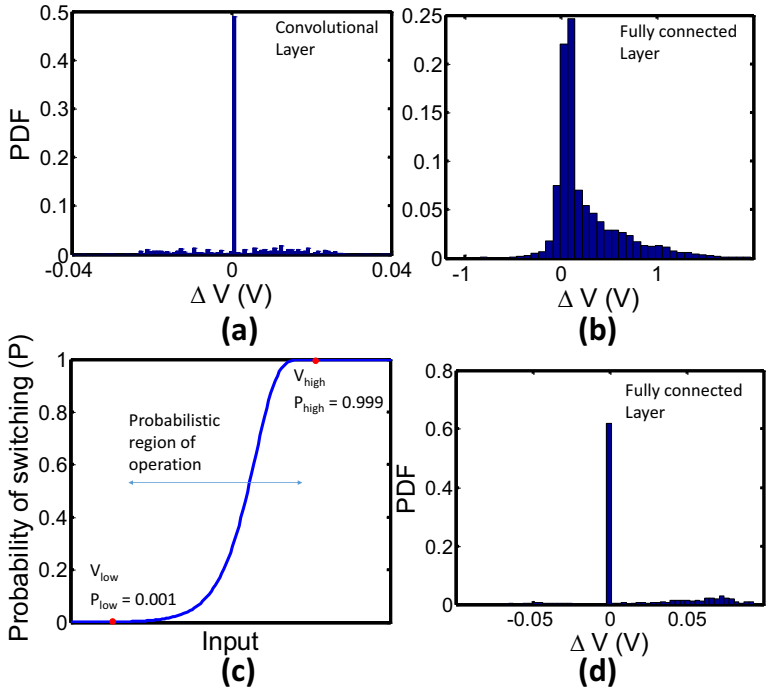

Fig. 3.9. The variations in the input voltages to the memristors. The difference between the actual and the ideal voltage (ΔV) that is being applied to a (a) convolutional layer and a (b) fully connected layer. (c) The probabilistic switching curve and the cut off voltages. (d) The ΔV when the applied voltages to the memristors are limited to V_{high} and V_{low}

devices. We first found the input spiking activities that must be applied to each layer, for 1000 random images in the testing data set. Then the corresponding voltages were applied to the inputs of the layers implemented in circuit level. After that, voltages applied on each memristor neuron was measured. Finally, the difference between these voltages and the actual voltages that must be ideally applied on memristors were calculated.

Fig. 3.9 (a) and (b) show the probability density functions (PDF) of these voltage differences (ΔV) for a convolutional layer and the fully connected layer of our network respectively. As the figure illustrates, for the convolutional layer, the differences in voltages fall well below $\pm$40mV ($\sim 3\sigma$). However, for the fully connected layer, we

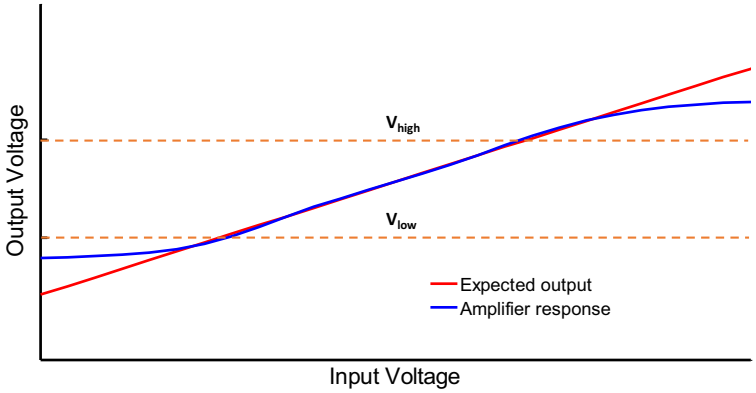

Fig. 3.10. The input output characteristics of the amplifier circuit. Note the non-linearities in the response at considerably high and low voltages

noticed significant ΔV values. It is noteworthy that this does not affect the correct functionality of the network. The reason can be explained as follows.

If a considerably high voltage is applied on the memristor, it is possible to state with higher confidence that the device will switch. For an example, if the applied voltage is larger than V_{high} (Fig. 3.9 (c)), it switches 99.9% of the time. Similarly, when a sufficiently lower voltage is applied ($< V_{low}$), it can be stated with higher confidence that the device does not switch. In contrast to a convolutional layer, the output of the final fully connected layer is forced to be a set of zeros and a one during training. Due to this, the actual mapped voltages to the output neurons will be higher than V_{high} or lower than V_{low}. Therefore, the output neuron memristors will operate on the deterministic region (Fig. 3.9 (c)). However, the designed amplifiers may not work linearly when the inputs are very high or low. *i.e.*, as shown in Fig. 3.10, the amplifier output gets saturated. This behavior in the amplifier is appropriate since applying higher voltages to the memristor leads to higher power consumption and faster device degradation. The higher differences in Fig. 3.9 (b) is due to such non linearities in the amplifier. We limited the applied voltage range from V_{low} to V_{high} to the same data we obtained for the Fig. 3.9 (b). These cut off voltages result in 0.001 and 0.999 probability values respectively. This range is a good approximation

given that the final classification accuracy is decided based upon a number of spikes; not just a single spike. The results in Fig. 3.9 (d) shows the updated differences and they fall below $\pm$100mV($\sim 3\sigma$). It will be shown in the next section that the network experiences only about $\sim$ 3% accuracy degradation for variations in the memristor input voltage with $\sigma = 200$mV.

3.5 Variation Analysis

3.5.1 Impact of variations in the input voltages to the neurons

In this section we observe the effect of variations in the neuron input voltages on the classification accuracy of the network. We conduct the experiment by changing the bias voltages of the neurons. As elaborated in section 3.2 D, a bias value must be selected to account for the output probability of the neuron during a non-spiking event. *i.e.*, if no spike appears at the input of the neuron, the bias voltage is the write pulse magnitude that will be applied to the memristor neuron. We perturbed all the neuron bias voltages following a Gaussian distribution with variable standard deviations from 50mV to 300mV. 50 independent Monte-Carlo simulations were conducted on all the 10000 test images. As Fig. 3.11 illustrates, the classification accuracy degrades by $\sim$ 14% when the σ is increased from 50mV to 300mV. The impact on accuracy increases exponentially with the increased amount of variations in the bias voltage. For an example, a 0.2V will result in just 3% degradation in accuracy which is almost three times smaller compared to the 14% degradation for a 300mV variation. We would thus declare that the network is robust to variations in bias voltages less than 200mV. The circuit simulations in the previous section show that the input voltage to the neurons are well below 100mV.

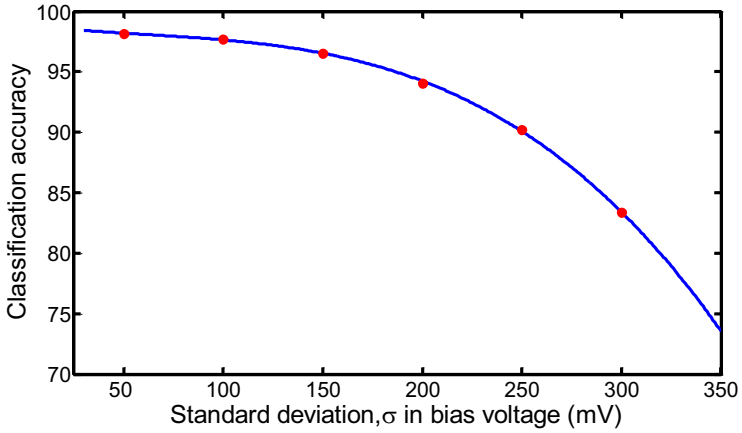

Fig. 3.11. Average classification accuracy with percentage variations in memristor neuron bias voltage. The variations follow a Gaussian distribution and independent Monte-Carlo simulations were conducted over the entire 10,000 testing image set for 50 trials.

3.5.2 The impact of write time on accuracy

As explained in section 3.1, in order to operate in the stochastic regime of a memristor, smaller write pulse widths require larger voltages and vice versa. It is however noteworthy that the switching probability curves for different pulse widths have almost the same sharpness (Fig. 3.1 (a)). The sharpness of the probability curve directly impacts the accuracy. Sharper curves will result in more classification errors. For an example, if the network was trained with a sharper curve, a slight change in a synaptic weight (due to weight quantization according to the multi-level memristors) value will result in a huge deviation at the output of a neuron to which the specific weight is connected. Fig. 3.12 shows how the classification accuracy varies with the number of time steps considered for different write pulse widths. Higher number of time steps will result in higher accuracy. As the figure illustrates, confirming our prior argument about the sharpness, we do not see any significant relationship with respect to accuracy degradation under varying write pulse width. However, it must be noted that the bias voltage in the amplifier must be increased with the reducing

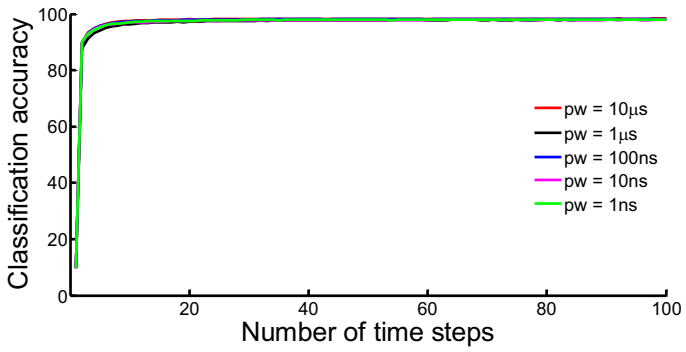

Fig. 3.12. The accuracy variation with increasing number of steps for different write pulse widths for the All-Memristor neural network

write pulse width. Larger voltages might damage the device and also cause in larger power consumptions.

When the network is trained for a given write pulse width (i.e., for a particular probability curve), the variations in this write pulse width when the network is in actual operation, may cause classification accuracy degradations. In order to observe this, we perturbed the write pulse width by a certain percentage and checked the accuracy at the output of the network for all the 10000 images in the testing dataset. For a network that was trained for a 100ns write pulse width, we observed only a 0.64% accuracy degradation for a 20% perturbation (i.e. 20ns perturbation), and a 0.79% degradation in accuracy for a 50% perturbation. The same percentage perturbations were applied to a network which was trained assuming a 20ns write pulse width. The degradation in accuracy we observed was 0.93% and 1.03% for 20% and 50% perturbation in write pulse width respectively. This explains that the network is very robust to the variations in the write pulse width.

3.5.3 The impact of synaptic weight variations

In our network, the synaptic functionality is performed by memristive crossbar arrays. Variations can be present in the memristor resistances in these crossbars due

to multiple reasons including the deviations occurred during programming, the effects of temperature, and temporal drifts in resistance due to the applied small voltages. Since such process variations is a common issue [120], we tested the robustness of our memristor based SNN system to variations in synaptic weights. We perturbed all synaptic weights we obtained from our modified offline training scheme, following a Gaussian distribution with different standard deviation (σ) values. Fig. 3.13 illustrates how the classification accuracy deviated with the increasing standard deviation (it is considered as a percentage of the weight). The accuracy degrades by $\sim 4.5\%$ when the standard deviation is 20%. For smaller σ values around 10%, the accuracy degradation is about 0.5%. Despite the inherent error resiliency associated with neural networks, the accuracy degradation is significant when $\sigma = 50\%$. The work in [121] also shows higher degradation in accuracy when memristors have high variations. However, the experimentally measured variability for a filament based device was as small as $\delta R/R \sim 9\%$ according to [122]. Our network is robust to variations of this magnitude.

As shown in [123], the typical write mechanisms will induce variations in multi level memristors. In order to account for this, high precision write mechanisms must be incorporated. A feedback write scheme would be appropriate to make sure that the proper value has been transferred to the memristors [54], [105]. The work in [54] experimentally shows the possibility to tune the memristive device within 1% accuracy degradation with respect to the desired state, within the dynamic range of the device. Furthermore, the usage of on-chip learning schemes will be helpful to account for these variations [124]

Resistance variations can occur in the neuron memristor as well. However, this does not cause any significant read error at the output since the off to on resistance ratio is in the order of $10^4 - 10^7$ [51] and the resistor divider circuit is capable of detecting this large drop with almost zero error (refer to section 3.3). Further, as long as the amplifier output impedance is low, the write operation does not get affected by the variations in the neuron memristor.

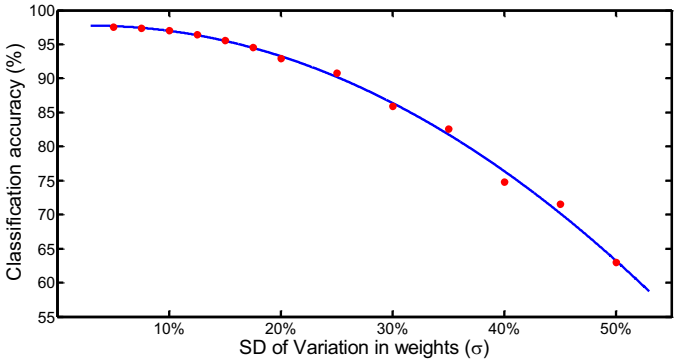

Fig. 3.13. Average classification accuracy with percentage variations in synaptic weights. The variations follow a Gaussian distribution and independent Monte-Carlo simulations were conducted over the entire 10,000 testing image set for 50 trials.

3.5.4 Accuracy degradation of the network due to the measuring resistor R_{meas}

In a memristive crossbar, the weighted summation of a set of input voltages are given in terms of a current. This current is given by the equation (3.14). It must then be converted in to a voltage to feed the neuron memristor. In order to do so, a measuring resistor was incorporated as shown in Fig. 3.4. Due to this measuring resistor, the resultant current witnesses some non-linearities. The actual current flowing through the R_{meas} can be given by the following equation

$$I_j(t) = \frac{V(t) \cdot [G_{1,j}, G_{2,j} ... G_{N,j}]^T}{1 + R_{meas} \sum_{i=1}^{N} G_{i,j}} \tag{3.15}$$

As explained in section IV, having a smaller R_{meas} with respect to the $\sum_{i=1}^{N} G_{i,j}$ will approximately make the current close to the inner product between $V(t)$ and G. In order to view the effect of the magnitude of this R_{meas} towards the accuracy of the network, we evaluated the network with different R_{meas} values. Fig. 3.14 shows how the accuracy changes with the value of R_{meas}. Higher resistances result in higher accuracy degradations.

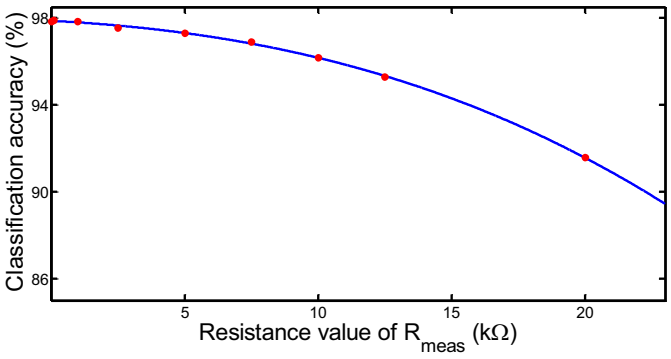

Fig. 3.14. Average classification accuracy with different values of measuring resistors (R_{meas}).

3.5.5 The Impact of variations in neuron memristors

The switching probability of a neuron memristor depends on the two fitting parameters τ_0 and V_0 (equation (3.4)). Due to variations, different memristors in the network can have different τ_0 and V_0 values. In order to view the effect of the variations in these parameters towards the accuracy of the network, Monte- Carlo simulations were conducted. The parameters were perturbed following a normal distribution while considering the experimental values [51] as the mean. As Fig. 3.15 (a) illustrates, the effect of the variations in τ_0 (up to 20%) towards the accuracy of the network is small ($\sim 5\%$). In contrast, the accuracy of the network is sensitive to the variations in V_0. When the percentage variation of V_0 goes above 6%, the accuracy degrades significantly (Fig. 3.15 (b)). When a particular memristor has a higher V_0 value, the neuron corresponding to that memristor has a lower probability of switching than what it was designed for, and vice versa. However, the experimental studies have shown that the thickness can affect the V_0 value and this effect is not significant. For example, it has been shown that when the thickness of the memristor was scaled by a factor of 2 (increased by 100%), the V_0 increases only by $\sim 35\%$ [51], [109]. Therefore, it can be argued that a $5-10\%$ variation in dimensions will not cause significant classification accuracy degradation.

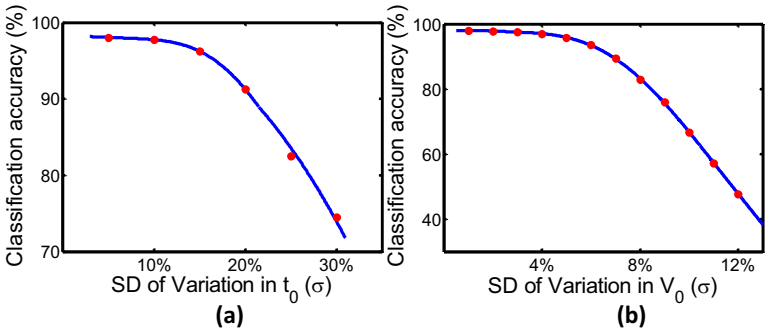

Fig. 3.15. Average classification accuracy with percentage variations in fitting parameter (a) τ_0 and (b) V_0 of the neuron memristors. The variations follow a Gaussian distribution with mean selected according to experimental values. Monte-Carlo simulations were conducted over the entire 10,000 testing images.

Cycle-to-cycle variations in ON and OFF resistances of the neuron memristors can occur as well. However, these variations have already been accounted for while experimentally obtaining the switching probability curve. Furthermore, the memristors degrade after a certain number of set-reset cycles. This can impact the probabilistic switching curve and thus the accuracy of the network. In order to view the impact of these changes, we perturbed the probabilistic switching curve by a small amount (ΔP) and simulated the network on 5000 images for 100 iterations. As Fig. 3.16 illustrates, the network is robust for variations in the probabilistic switching curve. However, sufficient data is not available in literature to exactly represent the effect of memristor degradation on the probabilistic switching curve (for the particular device we selected).

3.6 Delay and energy consumption of the SNN

As we noted in Fig. 3.1, in order to get the same switching probability for a memristor, the lower write pulse widths require higher write pulse magnitudes. Since the relationship between the energy and the write pulse width is not quite intuitive,

Table 3.1.
Device simulation parameters

Parameters	Values
On resistance (R_{on})	500kΩ [50]
On/Off ratio	10^3 [50]
Thickness of the insulation, t_{a-Si}	$60nm$ [51]
Fitting parameter V_0	0.22 [51, 109]
Fitting parameter τ_0	2.85×10^5 [51, 109]
Crossbar operating voltage	$1V$ [50]

we calculated the energy consumption of a single neuron for different write pulse widths. The results are summarized in Fig. 3.17. Here we assumed a spiking activity of 0.5 at the input. The results suggest that larger pulse widths result in larger energy consumption. This is due to the exponential relationship between the voltage and pulse width. For example, if the write pulse width must be reduced from $1\mu s$ to $100ns$ to achieve faster operation, the required voltage increment is just 500mV and the energy consumption would be better than the memristor operating in $1\mu s$ (even though the power consumption reduced).

When considering the energy consumption of the entire system per image classification, the number of time steps (write, read, and reset cycles) plays an important role. The accuracy of the network increases with the number of time steps over which the winning neuron is decided (Fig. 3.12). A reasonable accuracy (above 96%) can be reached within 10 time steps as shown in Fig. 3.12. However, for more complex datasets, the convergence time can be much higher ($\sim$ 50 time steps) [125] due to the fact that the data sets are much bigger, and more number of neurons are required to

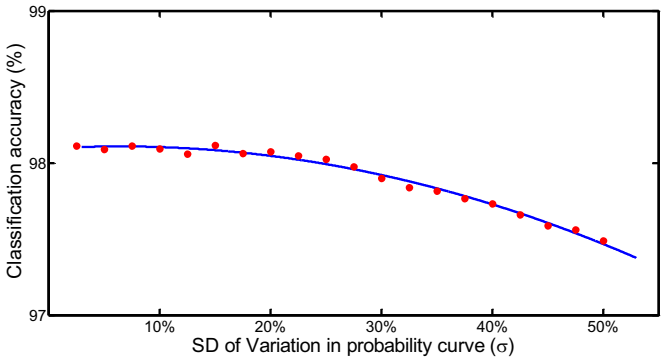

Fig. 3.16. Average classification accuracy with percentage variations in probabilistic switching curve. Each point in the probability curve was perturbed by a ΔP amount following a Gaussian distribution. The accuracy of the network was measured over 5000 images on the testing data set

increase the accuracy [126]. Hence, for our energy comparison with CMOS baseline, we conservatively choose 50 time steps for SNN inference.

In order to calculate the energy consumption of the whole network, we used SPICE simulations in IBM 45nm technology. We considered the average spiking activities for the images in the testing data set. The crossbar voltages must be selected appropriately (depending upon the type of memristors used) so that the drift in the resistance values over time is minimal. All the important parameters involved in this work are included in Table 3.1. The energy required for a single write to the neuron memristor (along with the amplifier) for a switching probability of 50% is 249fJ. A single reset and a single read operation consumes $\sim$ 500fJ and $\sim$ 1.4fJ amount of energy respectively. The average energy per image classification was 115nJ. Note that in our energy estimation for CNN execution (both memristive and CMOS hardware) we assume that input data is available in the form of spikes (two voltage levels). Such Poisson rate-encoded spike streams can be potentially obtained from event-driven sensors [127]. The energy consumption due to the event-sensor operation would be a common component to both the memristor and CMOS implementations. The afore-

mentioned energy number includes the energy associated with the peripheral buffer read and write, crossbars, and the read-write-reset of the neurons.

We observed that the energy of the crossbar is the dominant component and this is justifiable due to the fact that the number of synapses are orders of magnitude larger than the number of neurons. For example, the last fully connected layer of the network has 1920 synapses and the number of neurons are only 10. This is a $\sim \times 200$ difference and thus we state the results are justifiable. The second dominant energy component is from the reset operation. This is because of the fact that the reset must be conducted in the deterministic region of operation of a memristor. That is, a high enough voltage pulse must be used to ensure that the device has turned off. Since the resistor value is now lower, the energy consumption is larger for this step. To address this, feedback reset mechanisms can be incorporated [104]. This will allow the operation in lower voltage stochastic regime with some feedback control circuitry that conditionally gets activated. Furthermore, a novel stochastic volatile memristor has been proposed in [49] as a true random number generator. It has been shown experimentally that once this memristor is turned on, it returns to its off state after a small duration of time ($\sim 100\mu s$) eliminating the requirement to force reset as we propose in this work. The write voltage is also lower (0.5V, for a $300\mu s$ pulse) in this device when compared with $Ag/a - Si/p - Si$ (3.3V for $1ms$ pulse [51]) and TiO_2 devices. This may even eliminate the requirement of high gain amplifiers that consume power. We thus argue that there are other types of memristor devices that can allow energy efficient implementation using the architecture we propose here. Our goal is addressing the applicability of electric field driven memristors in general for deep stochastic SNN. Therefore we conducted the energy calculations without the lack of generality.

Table 3.2.
Area and delay of CMOS and memristor based implementations

Implementation	Area ($\times 10^{-3}\mu m^2$)		Delay	Area×Delay (ns mm^2)
Memristor based	Crossbars	2895	210ns	652
	Neurons	154		
	Buffers	56		
CMOS based	190		28μs	5320

3.6.1 Comparison with CMOS implementation

For the purpose of comparison of our work, we used the CMOS spiking network baseline proposed in [118]. The weight data are stored in SRAM. Subsequently, neurons in the CNN are temporally scheduled on the computation core comprised of FIFOs and neuron units. Each neuron computation involved moving data (input and weight) from SRAM to FIFOs and moving the computed outputs from neuron units back to SRAM when computation is completed. The energy of the CMOS design along with the data fetching energy from the memory, is $\sim 736nJ$ (with 130nJ for memory accessing, 64nJ for buffers, and 542nJ for neurons). The energy number is for iso-number of time steps as our proposed All-Memristor network (50 steps). This is approximately 6.4 times larger than the energy consumption of the proposal. However, we would like to point out that memristor neurons degrade faster over time when compared with CMOS neurons, even though the endurance of memristors is significant (up to 10^{12} set-reset cycles). Larger operating voltages may speed up this degradation process as well [104]. The on-off resistance ratio of a memristor changes after a certain number of write cycles and may have different switching probability curves other than the one used for training the network. This will lead to lowered

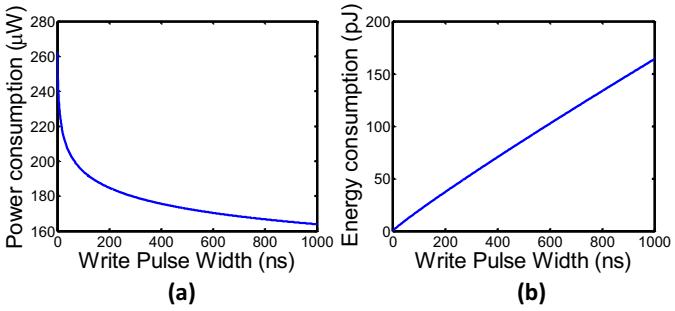

Fig. 3.17. (a) The neuron power consumption and (b) energy consumption for different write pulse widths. The experiment was for an amplifier output voltage that corresponds to a switching probability of 0.5 for the selected write pulse width. Even though the power consumption reduces with the increasing pulse width, the energy consumption grows.

classification accuracies as explained in section 3.5 F. However, retraining might help in regaining some lost accuracy but the feasibility of this is debatable.

In order to find the delay and area of our design, we divided the computation in to 128 × 128 memristive crossbar arrays following the procedure mentioned in section 3.3 A. The total area of the design (including crossbars, neuron memristors, buffers, amplifiers, and inverters) was $\sim 3mm^2$ (Table 3.2). The crossbar cell size was assumed to be $100F^2$ [128]. In our design, for a write time of 10ns [51], a single step takes 42ns (crossbar access time of $\sim$ 10ns [129] [128], read and buffer time $\sim$ 2ns, reset time $\sim$ 20ns). The latency for a single spike is 210ns (the propagation time through the full network). The latency for a single spike for the spiking CMOS architecture [118] takes $28\mu s$ for the same network. In order to fairly compare the two implementations, we are estimating the area × delay product. The product for our proposal is 652nsmm^2. For the CMOS implementation, the value is 5320nsmm^2 (which is $\sim 8\times$ bigger and thus worse).

3.7 Conclusion

Memristive switching devices have shown to be promising candidates for an enormous array of applications including logic, memory and neuromorphic computing. However, their inherent stochasticity has given life to reliability concerns. Numerous mechanisms involving larger write pulse widths, larger operating voltages, or feedback architectures have been proposed to drive these highly stochastic devices to their deterministic operating regime. As a result, we have to pay in terms of larger power consumption. This work is an effort of exploring an avenue where such stochasticity can be embraced rather than eliminating, with the goal of reducing power consumption. The proposal is embedding the functionality of a stochastic neuron to a memristor while representing the synaptic weights by a memristive crossbar to build the"All-Memristor Deep Stochastic SNN".

We tested the functionality of the network using the MNIST handwritten digit data set and witnessed a very low accuracy degradation ($\sim 1\%$) when compared with the deep ANN baseline. The design space of the network was estimated by applying variations and we observed that our proposal is robust to variations in the synaptic weights ($\sigma < 20\%$), neuron bias voltages (< 200mV), probabilistic curve, and the write time durations ($\sim 50\%$ of the pulse widths). The steepness of the activation function of a neural network affects the accuracy of the output and makes it less robust to variations. The constant steepness of the switching probability curve of a memristor (the probabilistic activation function) over different write times gives more flexibility for the memristor to be utilized in platforms with different speed limits without creating any accuracy degradation.

Smaller write pulse widths require larger voltages to bring the memristor to its stochastic region of operation. However, the required increment in voltage magnitude to operate 10 times faster is small (< 500mV) leading to lower energy consumption at the neuron in fast operating platforms. Furthermore, the total energy consumption of the proposal is 6.4 times smaller, and the area$\times$delay product is 8 times smaller when

compared with the digital CMOS counterpart. The work presented in this section is published in [130].

4. LIQUID ENSEMBLES FOR ENHANCING THE PERFORMANCE AND ACCURACY OF LIQUID STATE MACHINES

Over the past few decades, artificial neural algorithms have developed to an extent that they can perform more human-like functions. Recurrent Neural Networks (RNNs) and their variants such as Long Short Term Memory (LSTM) networks have become the state-of-the-art for processing spatio-temporal data. The massive RNNs of today, can describe images in natural language [131], produce handwriting [132], and even make phone calls to book appointments [133]. Such fascinating, human-like capabilities are obtained at the cost of increased structural and training complexity, and thus significant power consumption, storage requirements, and delay.

In this work we focus on a particular type of spiking RNN; the Liquid State Machine (LSM) [16]. An LSM consists of a set of inputs sparsely connected to a randomly and recurrently interlinked pool of spiking neurons called the 'liquid'. The liquid is connected to an output classifier, which can be trained using standard methods such as Spike Timing Dependent Plasticity (STDP), backpropagation, delta rule *etc.* [17]. LSMs have been used for a variety of applications including robot control [18], sequence generation [19], decoding actual brain activity [20], action recognition [21], speech recognition [16, 22–26], and image recognition [25, 27–29].

LSMs gained their popularity due to two main reasons. First, the LSM architecture is neuro-inspired. Spike base communication is used by biological systems, and they are faster and more efficient than state-of-the-art supercomputers [30, 31]. Second, LSMs have simple structure and lower training complexity among other RNNs. The claim is that, sufficiently large and complex liquids inherently possess large computational power for real-time computing. Therefore, it is not necessary to "construct" circuits to achieve substantial computational power. However, such simple

structure of LSMs comes with an accuracy trade-off. A plethora of work in the literature suggests mechanisms for improving the accuracy of LSMs including training the liquid connections [28] and involving multiple layers of liquids to form deep LSMs [134]. Despite the accuracy improvement, these mechanisms found in literature tend to alter the standard simple structure of LSMs. Choosing an LSM for a particular application and improving its accuracy at the cost of added complexity, nonetheless questions the motivation behind choosing an LSM in the first place.

Without deviating from the inherent simplicity of the LSM structure, several basic approaches can be used to improve its accuracy. One such fundamental approach is to increase the number of neurons within the liquid. However, the number of connections within the liquid also increases following a quadratic relationship with the number of neurons. Furthermore, the sensitivity of accuracy to the liquid neuron count decreases with the number of neurons beyond a certain point. In other words, enlarging the liquid introduces scalability challenges, and the accompanied cost tends to veil the accuracy benefits. The percentage connectivity also plays a role in improving the accuracy. Either high or low percentage connectivity results in accuracy degradation, signaling the existence of an optimum connectivity.

Note, there are two key properties that measure the capacity of an LSM: *separation* and *approximation* [16]. Aforementioned basic approaches; changing the number of neurons and connectivity in the liquid, indeed has an impact on the above measures. Based on separation and approximation, we propose an "ensemble of liquids approach" that can improve the classification accuracy (compared to a single large LSM) with reduced connectivity. The approach is scalable and preserves the simplicity of the network structure. In our ensemble of liquids approach, we split a large liquid into multiple smaller liquids. These resultant liquids can be evaluated in parallel since they are independent of each other, which leads to performance benefits. Furthermore, for a given percentage connectivity, the number of connections available in the ensemble approach is less than that of a single liquid with the same number of neurons. This reduces the storage requirement of the LSM as well. We used a

variant of the Fisher's linear discriminant ratio [135, 136] (the ratio between the separation and approximation) to quantify how well the ensemble of liquids represents the spatio-temporal input data. We observed that increasing the liquid count beyond a certain point reduces the accuracy of the LSM. This signals the existence of an optimum number of liquids, which is highly dependent upon the application and the number of neurons in the liquid. We show that dividing the liquid provides both accuracy and performance benefits for spatial and temporal pattern recognition tasks, on standard speech and image data sets.

The 'ensemble' concept has been previously used [137] for echo state networks or ESNs [138], which are similar in architecture to LSMs but use artificial rate-based neurons. Rather than using a single ESN predictor, multiple predictors (component predictors) were used and their predictions were combined together to obtain the final outcome. This approach was proposed to avoid the instability of the output of each individual predictor, since the input and internal connection weights are assigned randomly. The final ensemble outcome was obtained by averaging the predictions of the component predictors. The approach in [137] is different from our work since we design the ensemble of liquids by removing certain connections from a bigger reservoir. Furthermore, only a single classifier is used at the output in our work in contrast to [137]. The authors in [16] conducted a small experiment with two time-varying signals, which shows that using four liquids is better than using a single liquid in terms of enhancing the separation property. However, in their experiments, the four liquids in total have four times the number of neurons as the single liquid case. Therefore, it is not obvious whether the improvement in separation is solely due to having four "separate" liquids. The increased number of neurons itself might have played a role in enhancing the separation. In contrast, we analyze the effects of dividing a large liquid into multiple smaller units, while leaving the total number of neurons the same. Research [29] also shows that multiple liquids perform better than a single liquid, at higher number of neurons. The input to liquid connections

in [29] were trained in an unsupervised manner. Also note that each liquid was fed with distinct parts of an input, and hence is different from this work.

4.1 Liquid State Machine (baseline)

The conventional structure of an LSM consists of an input layer sparsely connected to a randomly interlinked pool of spiking neurons called the liquid. The liquid is then connected to a classifier which has synaptic weights that could be learnt using supervised training algorithms for inference.

4.1.1 Liquid neurons

The neurons within the liquid are leaky integrate-and-fire neurons [139] of two types; excitatory and inhibitory neurons. The number of excitatory (E) neurons and inhibitory (I) neurons were selected according to a 4 : 1 ratio, as observed in the auditory cortex [140]. The membrane potential (V) of a neuron increases/decreases as a pre excitatory/inhibitory neuron connected to it spikes that is described by

$$\tau \frac{dV}{dt} = (E_{rest} - V) + g_e(E_{exc} - V) + g_i(E_{inh} - V) \tag{4.1}$$

where E_{rest} is the resting membrane potential, τ is the membrane potential decay time constant, E_{exc} and E_{inh} are the equilibrium potentials of excitatory and inhibitory synapses, and g_e and g_i are the conductance values of the excitatory and inhibitory synapses, respectively. As the membrane potential reaches a certain threshold, the neuron generates a spike. The membrane potential drops to its reset potential upon generating a spike as shown in Figure 4.1(a), and then enters its refractory period t_{refrac} during which it produces no further spikes. The formulations elaborated in [141] were used for modeling the dynamics of the spiking neurons.

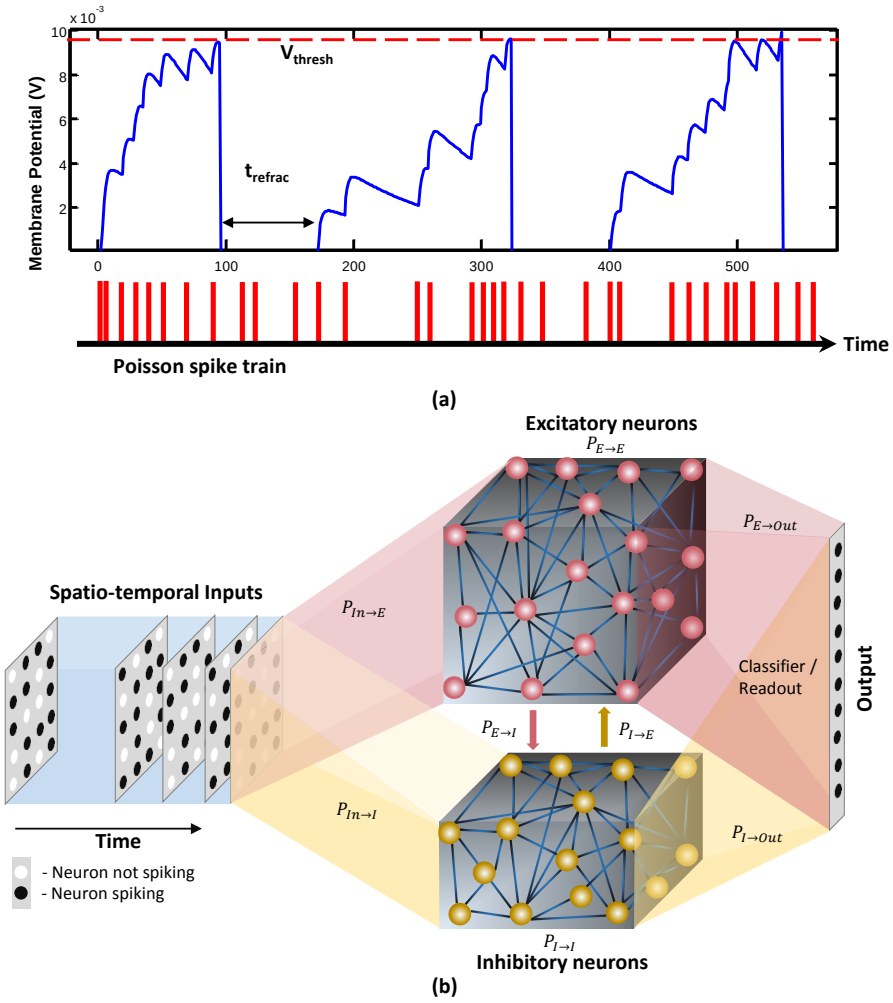

Fig. 4.1. (a) The dynamics of the membrane potential (V) of a spiking neuron. Each spike shown below the graph will increase the membrane potential. When V reaches the threshold V_{thresh}, the neuron will generate a spike and V will drop to the rest potential V_{rest} for a t_{refrac} duration of time. This duration is called the refractory period, and the neuron stays idle within this period. [36] (b) The structure of the liquid state machine. The input is connected to a reservoir with two types of neurons; inhibitory and excitatory. The reservoir is then connected to a classifier which is typically trained using supervised learning methods. The percentage connectivity between different types of *pre* and *post* neurons ($P_{pre \to post}$) are as indicated in the figure.

4.1.2 Liquid connections

The input is sparsely connected to the liquid ($In \rightarrow E$ connections). The percentage input to liquid connectivity ($P_{IN \rightarrow E}$) plays an important role in achieving good accuracy as will be explained in subsection 4.8.2. The liquid is composed of connections from excitatory to excitatory neurons ($E \rightarrow E$), excitatory to inhibitory neurons ($E \rightarrow I$), inhibitory to excitatory neurons ($I \rightarrow E$), and inhibitory to inhibitory neurons ($I \rightarrow I$). In our notation, the first letter indicates the pre-neuron type (PRE) and the second letter denotes the post-neuron type ($POST$). The selected percentage connectivity ($P_{IN \rightarrow E}$, $P_{E \rightarrow E}$, $P_{E \rightarrow I}$, $P_{I \rightarrow E}$, $P_{I \rightarrow I}$) within the liquid are shown in Table 4.1. These percentage connectivity values worked the best in terms of accuracy, for the neuron parameter selections in this work shown in Table 4.2. The strengths of all the connections ($W \in [0,1]^{N_{PRE} \times N_{POST}}$) were selected randomly [16] from a uniform distribution $U(0,1)$ [29,142]. A randomly generated mask $\left(M \in \{0,1\}^{N_{PRE} \times N_{POST}}, m_{ij} \in M\right)$ decides which connections exist to obtain the desired sparsity/percentage connectivity $\left(P_{PRE \rightarrow POST} = \frac{\sum_{(\forall i, \forall j)} m_{ij}}{(N_{PRE} \times N_{POST})} \times 100\%\right)$. Here N_{PRE} and N_{POST} are the number of PRE and $POST$ neurons, respectively. The dynamic conductance change model was used for synapses. *i.e.*, when a pre-synaptic neuron fires, the synaptic conductance instantaneously changes according to their strengths and then decays exponentially with a time constant [141]. Following equation shows the dynamics of a synapse (g_e) with an excitatory pre-neuron. τ_{g_e} is the decay time constant. This is similar to the post-synaptic current model in [143].

$$\tau_{g_e} \frac{dg_e}{dt} = -g_e \tag{4.2}$$

4.1.3 Output classifier

The liquid is connected to a classifier which is trained in a supervised manner. As suggested in [16], a memory-less readout (the readout is not required to retain any memory of previous states) can be used to classify the states of the liquid. The

Table 4.1.
Percentage connectivity within the liquid

Type of connectivity	Percentage connectivity (Speech recognition)		Percentage connectivity (Image recognition)	
	TI-alpha	TI-10	MNIST	E-MNIST
Input – Excitatory	34%	23%	10%	10%
Input – Inhibitory	0%	0%	0%	0%
Excitatory – Excitatory	40%	40%	40%	40%
Excitatory – Inhibitory	40%	40%	40%	40%
Inhibitory – Excitatory	50%	50%	50%	50%
Inhibitory – Inhibitory	0%	0%	0%	0%

liquid state in this work is the normalized spike count of the excitatory neurons [144] within a duration of T, when the input is applied. There is a liquid state vector ($s_i \in [0,1]^{N_E}$, N_E is the number of excitatory neurons) per applied input (i). The collection of all the state vectors were then used to train the classifier using gradient descent error backpropagation algorithm [145], similar to [29]. By doing this, we do discard some temporal information. However, since we do not use "anytime-speech-recognition" (a liquid with a classifier which is capable of recognizing a speech input, before the entire temporal signal is applied to the liquid) proposed in [146], the above classification method is sufficient to achieve reasonable accuracy (as per the accuracy values reported in other LSM works) for the applications we are considering in this work.

Table 4.2.
Spiking neuron parameters of the liquid state machine

Parameter name	Parameter value
Excitatory weight decay time constant, t_{ge}	1ms
Inhibitory weight decay time constant, t_{gi}	2ms
Threshold Inhibitory, $thresh_i$	−40mV
Threshold Excitatory, $thresh_e$	−52mV
Inhibitory rest potential, $v_{rest,i}$	−60mV
Excitatory rest potential, $v_{rest,e}$	−65mV

4.2 Ensemble approach for LSMs

In this section, we explain our proposed ensemble of liquids approach, which improves the scalability of LSMs. The proposed approach is different from the ensemble works available in literature on a variety of network types (feed-forward fully connected spiking and analog neural networks), where multiple classified outputs of independently trained networks are combined together to increase the probability of correct classification [147, 148]. In this work, we analyze the impact of dividing a reservoir, such that all the resultant small reservoirs can potentially be evaluated in parallel, for an applied input. As explained in the previous section, the typical structure of an LSM has an input, a liquid where neurons are sparsely interlinked, and a readout trained using supervised learning methods (Figure 4.1(b)). In our ensemble approach, the same number of liquid neurons (N_{tot}) is divided to create an N_{ens} number of smaller liquids, as shown in Figure 4.2. While dividing the liquid, the number of excitatory neurons (N_E^i) to inhibitory neurons (N_I^i) ratio in the i^{th} ($i = 0, 1, ..., N_{ens}$) liquid is maintained at 4 : 1. The percentage connectivity is also adjusted to suit the new reduced number of neurons ($\frac{N_{tot}}{N_{ens}}$) in a liquid. This is done by

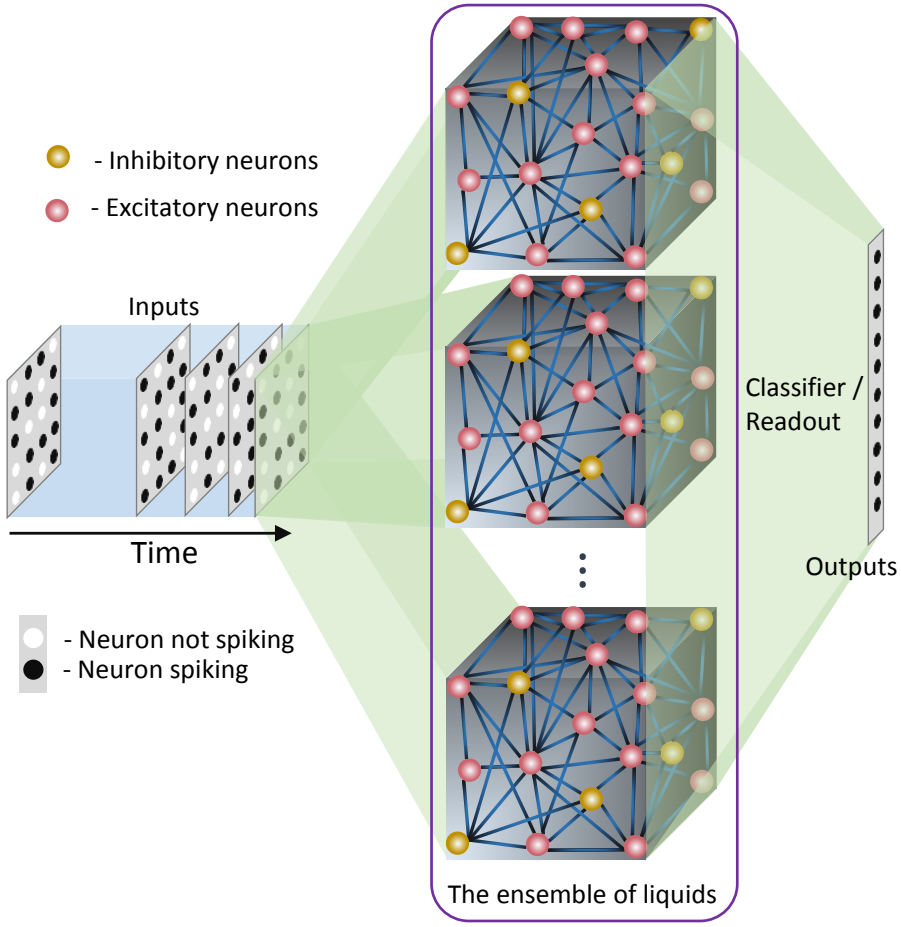

Fig. 4.2. The structure of the ensemble approach. The liquid in the standard LSM is split up to create an ensemble of smaller liquids. The input is sparsely connected to all the liquids. The output of all the liquids are concatenated to form one large liquid state vector, and connected to a single readout that is trained using supervised learning methods.

first creating a standard LSM explained in the previous section with $\frac{N_{tot}}{N_{ens}}$ number of neurons and adjusting all the percentage connectivity values till we get a reasonable accuracy. Then the input to liquid percentage connectivity ($P_{IN \rightarrow E}$) was exhaustively changed until the accuracy peaks for a given application, which is then used for all the experiments reported in this work.

Each small liquid has its own liquid state vector, which is the normalized spike count of all the excitatory neurons in the respective liquid within a duration of T, as explained in the previous section. All the state vectors produced by each individual liquid in the ensemble are concatenated to form one large state vector per input. Note that the length of the concatenated state vectors are the same for both the single liquid case (baseline) and for the ensemble of liquids, since the total number of neurons are held constant for a fair comparison. The concatenated state vector is used to train a single readout using gradient descent backpropagation. This division of one large liquid to form an ensemble of liquids enhances class discrimination associated with LSMs as elaborated in the next section.

4.3 Properties of LSMs

Two macroscopic properties of an LSM, namely, Separation Property (SP) and Approximation Property (AP), can be used to quantify a liquid's ability to provide an improved projection of the input data. With respect to classification applications, SP gives a measure of the liquid's ability to separate input instances that belong to different classes. AP, on the other hand, gives a measure of the closeness of the liquid's representation of two inputs that belong to the same class.

Several methods of quantifying the SP and AP as a measure of the computational power (kernel quality) of an LSM are suggested in [16,149]. Two methods of measuring the SP are *pairwise separation property* and *linear separation property*. The pairwise separation property is the distance between two continuous time states of the liquid ($x_u(t)$ and $x_v(t)$), resulting from two different input spike trains ($u(t)$ and $v(t)$). Here the continuous time states $x(t)$ are defined as the vector of output values of linear filters with exponential decay (with time constant $30ms$ [16]) at time t. The distance can be calculated by the Euclidean norm between $x_u(t_n)$ and $x_v(t_n)$ at sample point t_n. The final pairwise separation property can be evaluated by obtaining the average across all the sampled instances (at t_n), as explained in the following equation

$$SP_{pw} = \frac{1}{N_{samples}} \sum_{n=1(0<t_n<T)}^{N_{samples}} ||x_u(t_n) - x_v(t_n)|| \tag{4.3}$$

where $N_{samples}$ is the number of sample points. The pairwise separation property (SP_{pw}) can be used as a measure of the kernel quality for two given inputs. However, most real life applications deal with more than two input spike trains. To address this, linear separation property is proposed as a more suitable quantitative measure to evaluate the computational power of a liquid in an LSM. The linear separation property (SP_{lin}) is the rank of the $N \times m$ matrix M_s, which contains the continuous time states $(x_{u_1}(t_0), ..., x_{u_m}(t_0))$ of the liquid as its columns. The continuous time state $x_{u_i}(t_0)$ is the liquid response to the input u_i (these inputs are from the training set), sampled at $t = t_0$. If the rank of the matrix is m, it guarantees that any given assignment of target outputs $y_i \in \mathbb{R}$ at time t_0 can be attained by means of a linear readout [149]. The rank of M_s is the degree of freedom the linear readout has, when mapping x_{u_i} to y_i. Even though the rank is $< m$, it can still be used as a measure of kernel quality of the LSM [149].

$$M_s = \left[x_{u_1}(t_0), ..., x_{u_i}(t_0), ..., x_{u_m}(t_0)\right] \tag{4.4}$$

$$SP_{lin} = rank\big(M_s\big)$$

The AP of the LSM can also be measured by the aforementioned rank concept as shown in [149, 150]. Instead of using significantly different examples in the training set, now the continuous time states $x_{u_i^j}(t_0)$ of the liquid are measured by feeding jittered versions of u_i (u_i^j) to the liquid. The rank of the matrix M_a that has m such continuous time states $x_{u_1^j}(t_0), ..., x_{u_m^j}(t_0)$ sampled at t_0 as its columns, is evaluated as a measure of the generalization capability of the liquid for unseen inputs. Unlike SP_{lin}, lower rank of M_a suggests better generalization.

Both AP and SP are important in measuring the computational quality of a liquid. For example, very high quantitative measure for SP and very low measure for AP is ideal. If one liquid has very high SP and a mediocre AP, it is hard to decide whether

the particular liquid is better than another liquid with mediocre SP and a very small AP. Therefore, in order to compare the quality of different liquid configurations, a combined measure that involves both SP and AP is required. To address this, we use some insights from Fisher's Linear Discriminant Analysis (LDA) [135, 136, 151]. LDA is utilized to find a linear combination ($f(.)$) of d features that characterizes or separates two or more classes (ω_i) of objects. The linear combination as shown in the equation below can be used as a classifier, or as a dimensionality reduction method before classification.

$$y_i = f(x_i) = Wx_i \tag{4.5}$$

where y_i is the output vector ($Y = [y_1, ..., y_n] \in \mathbb{R}^{L \times n}$), that corresponds to the set of input features, x_i ($X = [x_1, ..., x_n] \in \mathbb{R}^{d \times n}$, each feature x_i is a column vector), $W \in \mathbb{R}^{L \times d}$ is the weight matrix that describes the linear relationship, L is the number of classes, and n is the number of data samples. The projection from LDA maximizes the separation of instances from different classes, and minimizes the dispersion of data from the same class, simultaneously, to achieve maximum class discrimination. The approximation capability is quantified by the matrix S_w called the 'within class scatter matrix' that is specified by

$$S_w = \sum_{i=1}^{L} P(\omega_i)\hat{\Sigma}_i \tag{4.6}$$

where $P(\omega_i)$ is the probability of class ω_i, $\hat{\Sigma}_i$ is the sample covariance matrix [152] for class ω_i. The separation capability is given by the 'between class scatter matrix' (S_b) that is described by

$$S_b = \sum_{i=1}^{L} P(\omega_i)(\mu_i - \mu_g)(\mu_i - \mu_g)^T \tag{4.7}$$

where μ_i is the sample mean vector (centroid) of class ω_i, and μ_g is the global sample mean vector. In classical LDA, the optimum weight matrix can be found by maximizing the objective function called Fisher's Discriminant Ratio (FDR) [136] that is computed as

$$FDR = tr(S_w^{-1} S_b) \tag{4.8}$$

where $tr(.)$ is the trace operation. For this work, the capability of the liquid to produce a good representation of the input data is quantified by a variant of the above ratio. The FDR is applied on the states of the liquid. However, when the data dimensionality (number of liquid neurons) is large in comparison to the sample size (n), the aforementioned scatter matrices tend to become singular [153] and the classical LDA cannot be applied. Hence, we use a modified discriminant ratio (DR) given by the following function:

$$DR = tr(S_b)tr(S_w)^{-1} \tag{4.9}$$

Note that the trace of S_w measures the closeness of each liquid state to its corresponding class mean as illustrated in Figure 4.3(a), and the trace of S_b measures the distance between each class centroid and the global centroid in multidimensional space as depicted in Figure 4.3(b). High $tr(S_b)$ suggests high SP (hence better) and smaller $tr(S_w)$ suggests better AP.

4.4 Experimental setup

The performance of the ensemble of liquids approach is compared against a single liquid baseline detailed in section 4.1, with the aid of two spatial image recognition applications and two spatio-temporal speech recognition applications. The liquid was modeled in BRIAN [154], a Python-based spiking neural network simulator, and the spiking activities of the neurons were recorded to calculate the liquid state vectors. The state vectors corresponding to the training input instances of each data set were then used to train a single fully-connected classification layer using the stochastic gradient descent algorithm [155,156]. The accuracy of the trained network was calculated on the testing data sets.

4.4.1 Data sets used for illustration

The two spatio-temporal (speech) data sets used in this work are:

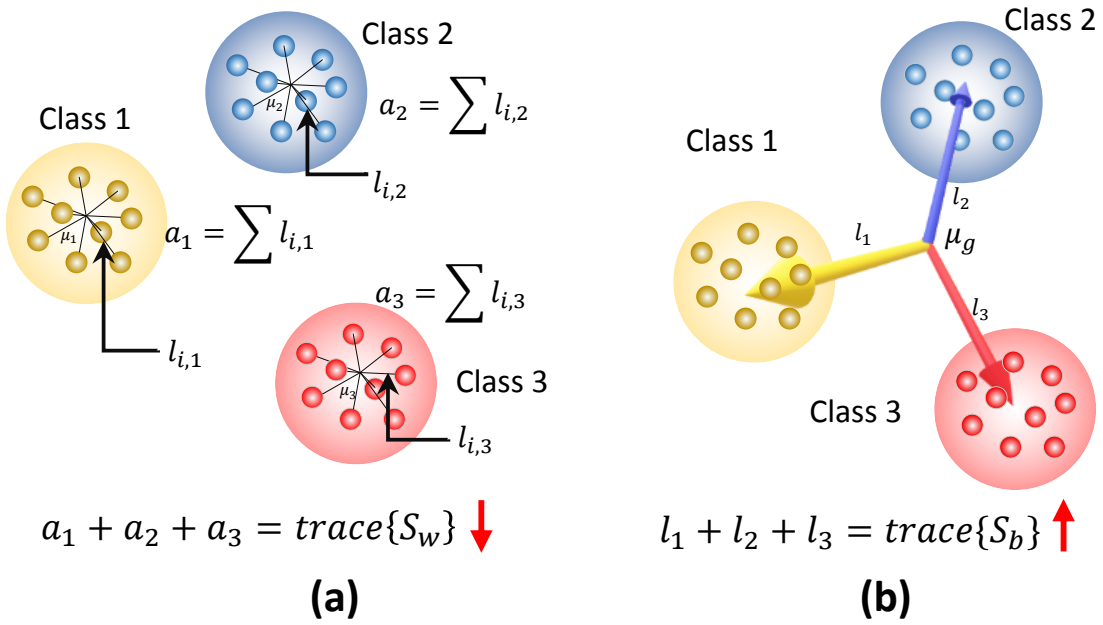

Fig. 4.3. The graphical representation of the components of the discriminant ratio (DR) for a set of two dimensional data points that belongs to three classes. (a) $tr(S_w)$ gives a measure of the addition of all the squared distances from the class means to each data point. This must be lower to have better approximation property. Here $l_{i,j}$ denotes the squared distance between the i^{th} data point in class j to the class centroid, μ_j (b) $tr\{S_b\}$ gives a measure of the addition of the squared distances between the global mean and each class mean. High value for $tr\{S_b\}$ signals higher separation property. Here l_i denotes the squared distance from the global mean μ_g to the centroid of class i.

1. Digit sub-vocabulary of the TI46 speech corpus [157] (TI-10)

2. TI 26-word "alphabet set"; a sub-vocabulary in the TI46 speech corpus [157] (TI-alpha)

TI-10 consists of utterances of the words 'zero' through 'nine' (10 classes) by 16 speakers. There are 1594 instances in the training data set and 2542 instances in the testing data set. TI-alpha, on the other hand, has utterances of the words 'A' through 'Z' (26 classes). There are 4142 and 6628 instances in the training and testing data sets, respectively. For the spatial data sets (images), we used the handwritten digits

from the MNIST [158] data set containing $60,000$ images of digits 0 through 9 in the training set and $10,000$ images in the testing set. In addition, we also created an extended MNIST data set that contains all the images from the original MNIST data set, and the same set of images transformed by rotation, shifting, and noise injection. It has $240,000$ images in the training data set and $40,000$ images in the testing data set.

4.4.2 Input spike generation

The first step is converting the images or the analog speech signals to spike trains to be applied as inputs to the liquid. For spatial data (images), there are p number of input spike trains fed in to the liquid, with p being the number of pixels in an image. The mean firing rate of each spike train is modulated depending upon the corresponding pixel intensity. Each input image pixel (i^{th} pixel) is mapped to a Poisson distributed spike train with the mean firing rate (r_i for the i^{th} image pixel) proportional to the corresponding pixel intensity (I_i) that is specified by

$$r_i = \frac{S_{count,i}}{T} \propto \left(\frac{I_i}{255}\right) \tag{4.10}$$

where $S_{count,i}$ is the number of spikes generated by the i^{th} input neuron within a time period of T. For example, mean firing rate in this work for a white pixel (pixel intensity $I_i = 255$) is selected as 63.75Hz. For a black pixel (pixel intensity $I_i = 0$), the mean firing rate is 0Hz. Each image is presented to the liquid for a duration of 300ms ($= T$).

For the speech data, the audio samples available in wave format were preprocessed based on Lyon's Passive Ear model [159] of the human cochlea, using Slaney's MATLAB auditory toolbox [160]. The model was used to convert each audio sample to temporal variation in the intensity of 39 frequency channels. These intensity values at each time step j ($I_{i,j}$) were then normalized and used as the instantaneous firing probability of an input neuron i ($i = \{1, 2, ..., 39\}$). The time step in this work is 0.5ms.

4.5 The kernel quality improvement due to the ensemble approach

In this section, we will explore the effects of dividing a large liquid, by means of standard measures for SP and AP explained in section 4.3. We involve the same general tasks suggested in [149, 150], to compare the SP and AP. In order to measure the pointwise separation property, we generated 100 input spike trains $u(t)$ and $v(t)$ with different distances $d^{in}_{u,v}$ between them. The distance between two input spike trains is evaluated according to the methodology explained in [149]. The two spike trains were first filtered with a Gaussian kernel $e^{-(t/\tau_{in})^2}$, and then the Euclidean distance between them were measured. τ_{in} was selected as $5ms$ [16].

$$d^{in}_{u,v} = \frac{||u(t) * e^{-(t/\tau_{in})^2} - v(t) * e^{-(t/\tau_{in})^2}||}{T} \tag{4.11}$$

The same 100 $u(t)$ and $v(t)$ signals were fed in to LSMs with different number of liquids ($N_{ens} = 1, 2, 4, 8, 10$), and the pairwise separation property was calculated according to Equation 4.3. The average SP_{pw} was evaluated over 10 different weight initialization trials and the results are shown in Figure 4.4. As the figure illustrates, the SP_{pw} improves with the distance $d^{in}_{u,v}$ between two inputs, and also with the number of liquids in the LSM.

For the linear separation property, we applied 400 randomly generated input signals $u_i(t)$ to LSMs with different number of liquids ($N_{ens} = 1, 2, 4, 8, 10$). The resultant states ($x_{u_i}(t_0)$) were used to create the matrix M_s explained in Equation 4.4. The average SP_{lin} ($= rank(M_s) = r_s$) was evaluated among five different sets of inputs and five different weight initializations (i.e., 25 trials altogether) and the results are finalized in Figure 4.5. As the figure illustrates, the SP_{lin} increases with the number of liquids. However, the rate of increment of SP_{lin} reduces with the increasing number of liquids.

For the generalization property, we conducted same above experiment with a different state matrix M_a. To create this matrix, we involved 400 jittered versions of the input signal $u_i(t)$, ($u^j_i(t)$) as explained in section 4.3. In order to create a

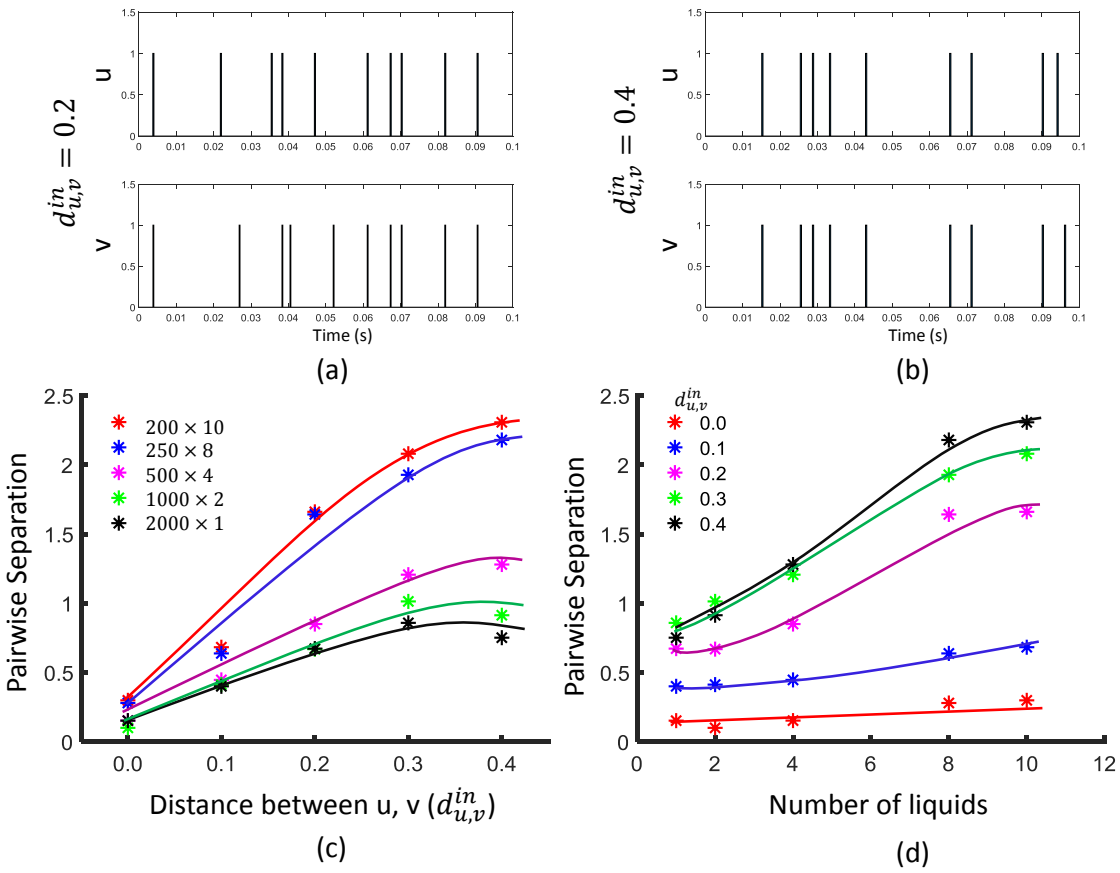

Fig. 4.4. The effect of dividing a large liquid on SP_{pw}, at different distances $d^{in}_{u,v}$ between inputs. Two input spike trains u and v are illustrated at (a)$d^{in}_{u,v} = 0.2$ and (b)$d^{in}_{u,v} = 0.4$. (c) The variation of pairwise separation with the distances between inputs, at different number of liquids (d) The variation of pairwise separation with the number of liquids, at different input distances $d^{in}_{u,v}$

jittered version of $u_i(t)$, we shifted the spike times by a small delay Δt taken from a Gaussian distribution as explained in [16]. The average rank of the matrix M_a is shown in Figure 4.5. A lower rank of M_a (r_a) suggests better approximation of intra-class input examples. According to the figure, r_a increases with the number of liquids. This signals the liquid losing its ability to generalize intra-class inputs.

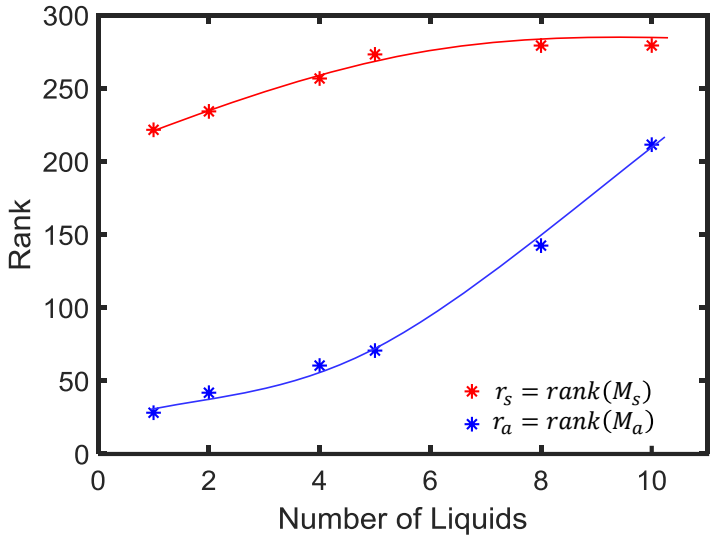

Fig. 4.5. The average rank of state matrix M_s that indicates the inter-class separability (in red) and the average rank of the matrix M_a which is an indication of the intra-class generalization capability (in blue)

We observed that the SP_{pw} improves by 3.06 in the 10-liquid ensemble approach, when comparing with the single liquid baseline. The SP_{lin} improvement is 1.26×. For a similar set of experiments (for $N_{tot} = 135$), the authors in [150] explored the kernel quality of an LSM of which the reservoir connections were trained using a structural plasticity rule [161]. The reported improvement in SP_{pw} is 1.36×, whereas the improvement in SP_{lin} is 2.05× when compared with a randomly generated traditional LSM. It is noteworthy that when training using structural plasticity, the inter-class separation capability can be improved, with respect to a traditional liquid with random connections. Without involving such complex learning techniques, one can obtain improved separation by simply dividing a liquid as shown in our work. However, note that such reservoir connection learning methods can simultaneously preserve the ability of the LSM to approximate intra-class examples, which is not attainable by the ensemble approach, at higher number of liquids. As explained in section 4.3, the ability of a liquid to produce a better representation of an input is

a measure of both SP and AP. In the next section, we will explore this combined measure of SP and AP defined as DR in section 4.3, on real world spatio-temporal data classification tasks.

4.6 Impact of the ensemble approach on accuracy of different applications

Using the experimental setup explained in the previous section 4.4, we initially simulated our baseline single liquid LSM (section 4.1) for the four data sets. We used 500, 2000, 1000, 1000 neurons in total within the liquid for the TI-10, TI-alpha,

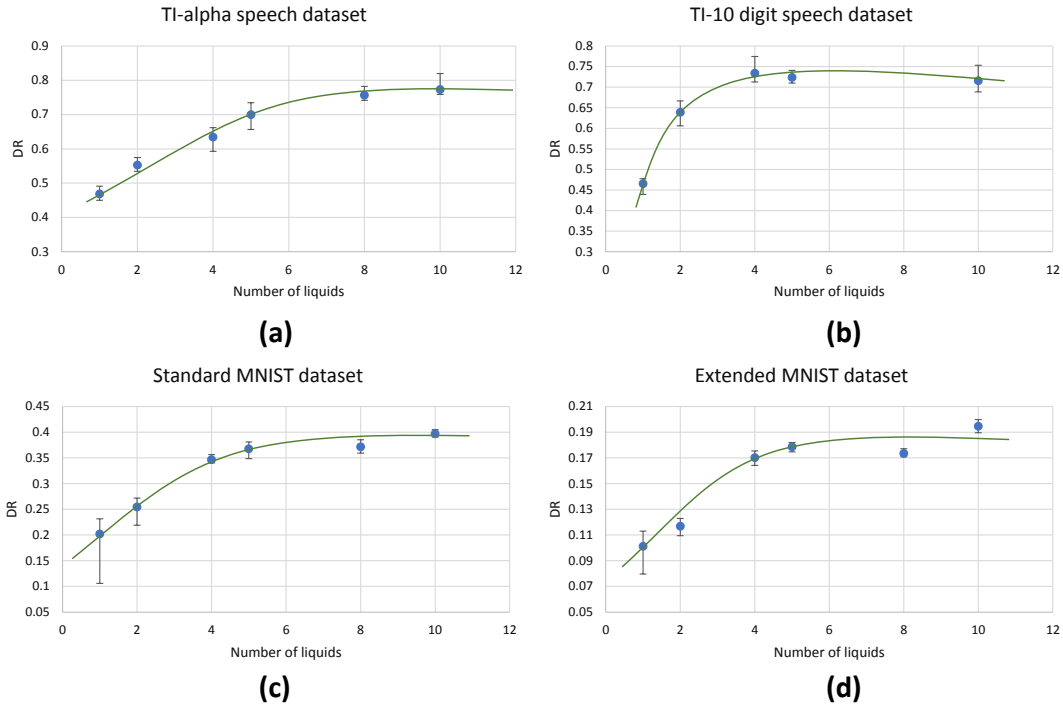

Fig. 4.6. The average (over five trials) discriminant ratio (DR) trends with different number of liquids in an ensemble for two speech recognition tasks; (a) TI-alpha dataset, (b) TI-10 dataset, and two image recognition tasks; (c) standard MNIST dataset, (d) extended MNIST dataset. The total number of neurons in each ensemble of liquids were kept the same. Note that all the DR trends increase with the number of liquids, and saturates after a certain point, that depends on the application

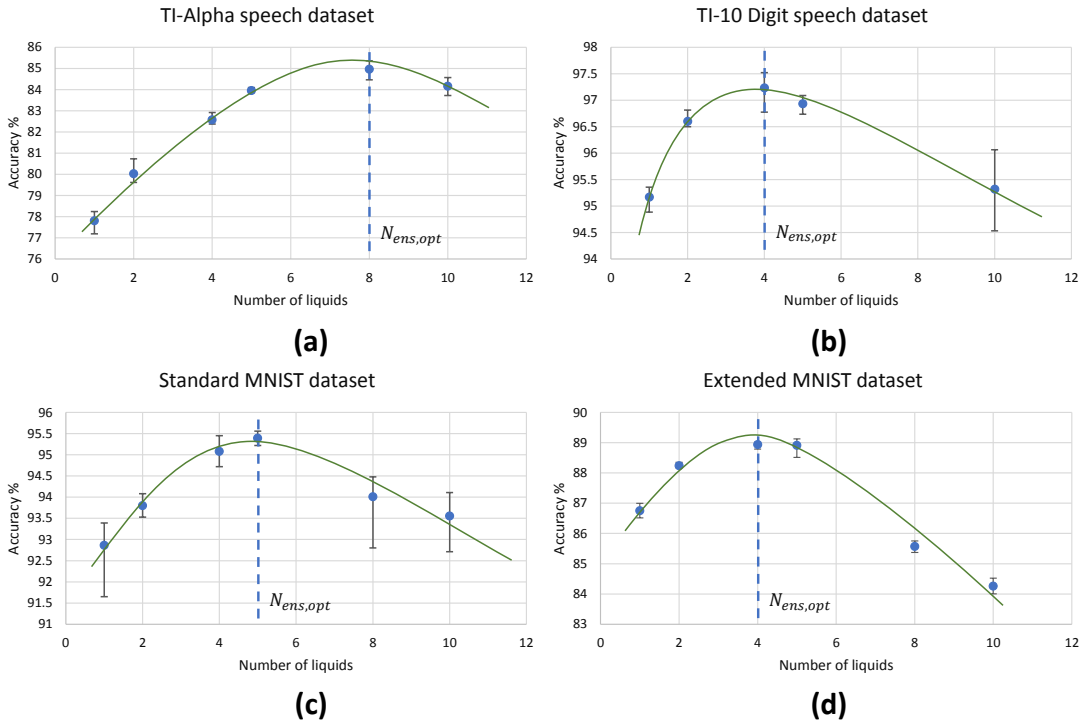

Fig. 4.7. The average (over five trials) accuracy (percentage) trends with different number of liquids in an ensemble, for two speech recognition tasks; (a) TI-alpha test dataset, (b) TI-10 test dataset, and two image recognition tasks; (c) standard MNIST test dataset, (d) extended MNIST test dataset. The total number of neurons in each ensemble of liquids were kept the same. Note that all the accuracy trends peak at a certain point, that depends on the application

standard MNIST, and extended MNIST pattern recognition tasks, respectively. We refined the percentage connectivity for each task as shown in Table 4.1. The classifier was trained using the liquid states corresponding to the training examples, and the classification accuracy of the trained network was obtained for unseen instances from the test data set. For each application, we then created an ensemble of liquids with $\frac{N_{tot}}{N_{ens}}$ number of neurons in each small liquid. For all the four applications, we evaluated the SP and AP for different number of liquids in the ensemble ($N_{ens} = 1, 2, 4, 5, 8, 10$) and quantified how good is the input representation of the ensemble of liquids using

DR (explained in section 4.3). Figure 4.6 shows that the DR increases up to a certain number of liquids in the ensemble and then saturates for the four different applications we have considered. This signals that the ensemble of liquids, in principle, gives a better representation of the input with increasing number of liquids until a certain point. In order to verify whether this improvement in the DR actually implies an improvement in classification accuracy, we evaluated the LSM accuracy for different number of liquids ($N_{ens} = 1, 2, 4, 5, 8, 10$) for the four different classification applications. Figure 4.7 shows that the accuracy indeed improves with the number of liquids until a certain point. Let us denote this point as the 'peak accuracy point' and the corresponding number of liquids for that point as the 'optimum number of liquids' ($N_{ens,opt}$). We noticed that the $N_{ens,opt}$ is a function of the application, and that increasing N_{ens} beyond $N_{ens,opt}$ actually results in accuracy loss. When comparing Figure 4.6 and Figure 4.7, it is evident that the point at which the DR saturates is the same as $N_{ens,opt}$. This explains that dividing a large liquid into multiple smaller liquids enhances the class discrimination capability of the liquid, leading to improved classification accuracy. However, note that after the $N_{ens,opt}$ point, the DR saturates whereas the accuracy degrades. The DR does not offer a direct mapping of the accuracy of an LSM. However, it could still be utilized as a measure of identifying the point at which the accuracy starts to drop ($N_{ens,opt}$). This is the same point at which the DR stops improving.

Figure 4.8 plots the variation of the individual DR components; separation $\big(SP = tr(S_b)\big)$ and approximation $\big(AP = tr(S_w)\big)$ with the number of liquids for the TI-alpha speech recognition task. Figure 4.8 shows that SP improves continuously with the number of liquids. Improved separation suggests larger dispersion among the centroids of the liquid states corresponding to instances from different classes, which renders the input representations provided by the liquid easier to classify. This is illustrated in the cartoon in Figure 4.9(a) for a set of two-dimensional data points from two classes, wherein higher SP while maintaining the same AP results in enhanced class discrimination capability. At the same time, Figure 4.8 indicates that AP also

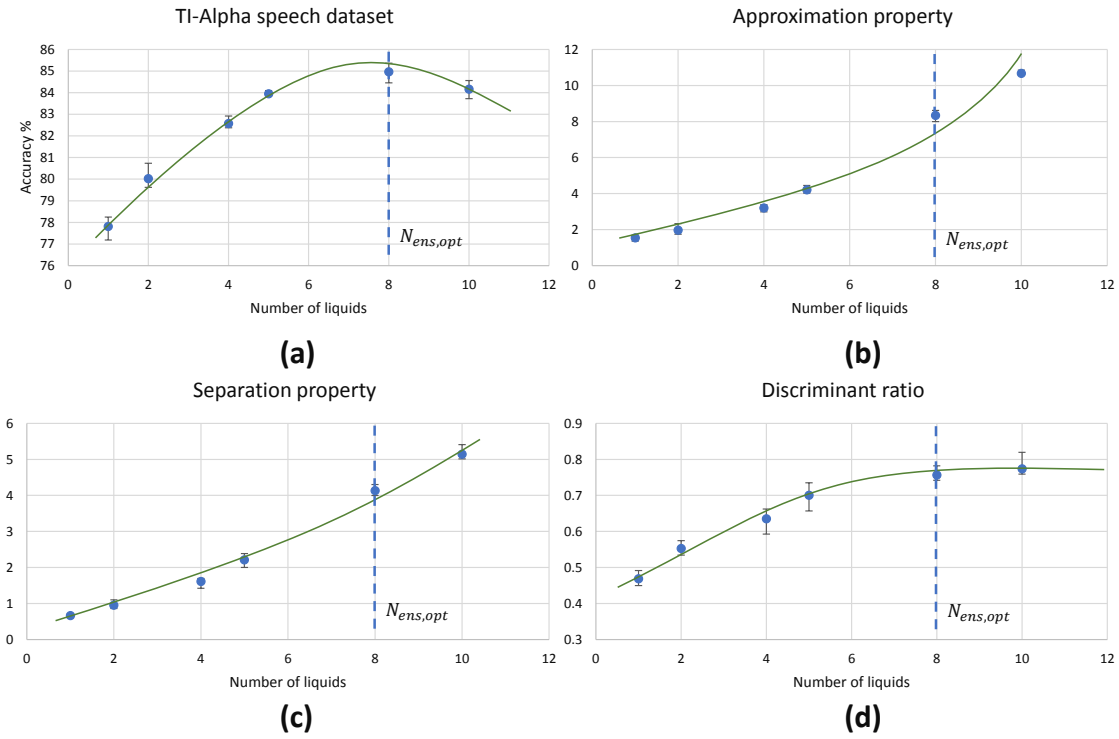

Fig. 4.8. The trends of different measures associated with LSMs, with the increasing number of liquids. (a) Accuracy, (b) Approximation property (AP), (c) Separation property (SP) and (d) Discriminant ratio (DR). The LSM is trained for TI-alpha speech recognition application. Both AP and SP continuously increases with the number of liquids. Note that the increment in AP is more significant than that of SP for larger number of liquids.

increases with the number of liquids, implying that larger number of liquids leads to higher dispersion between projected inputs from the same class. Higher AP for a given SP is not desirable since it could potentially lead to overlap among instances belonging to different classes as depicted in Figure 4.9(b), thereby degrading the class discrimination capability. Since both SP and AP increases, the ratio DR gives a better measure about the overall effect of the proposed ensemble approach on the classification accuracy of the LSM rather than the individual components per se. As shown in Figure 4.8, the DR increases until a certain number of liquids, signaling

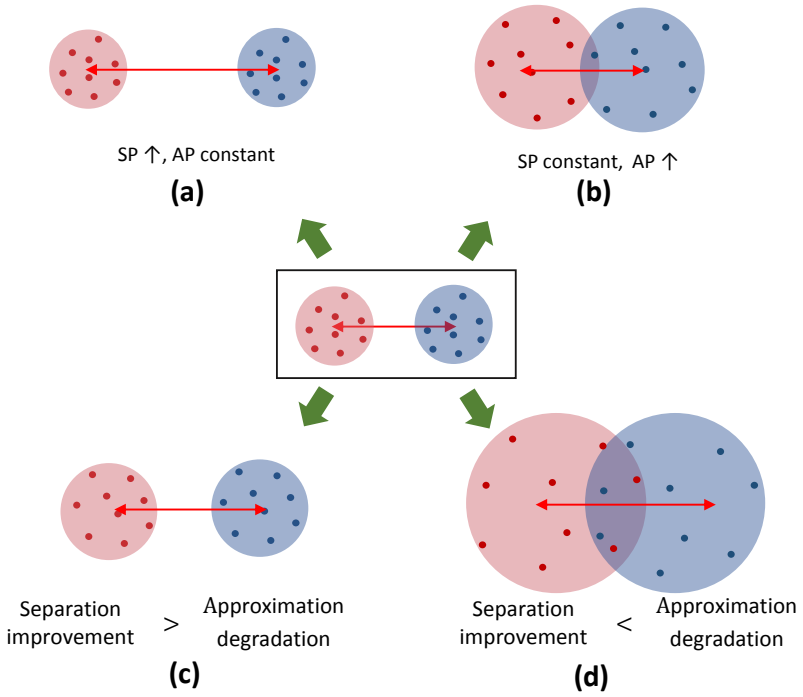

Fig. 4.9. A cartoon that shows the distribution of two dimensional data points that belong to two classes under different conditions. (a) A case with increased SP while maintaining the same AP. (b) A case where the AP is increased while maintaining the same SP. Note that the class boundaries can get overlapped leading to classification errors. Hence, increased AP is not desirable. (c) and (d) shows two scenarios where both SP and AP increased from the baseline distribution of data points (figure in the middle). (c) The improvement in SP is larger than the degradation in AP. (d) The improvement in SP is not sufficient to compensate for the AP degradation, leading to overlapped class boundaries

the dominance of the improvement in SP over the degradation in AP as graphically illustrated in Figure 4.9(c). In contrast, as the number of liquids is increased beyond $N_{ens,opt}$, DR saturates since the increment in SP is no longer sufficient to compensate for the degradation in AP as shown in Figure 4.8. When the dispersion between classes (due to increment in SP) is not sufficient to compensate for the dispersion occurring for instances within the same class (due to AP degradation), there can be

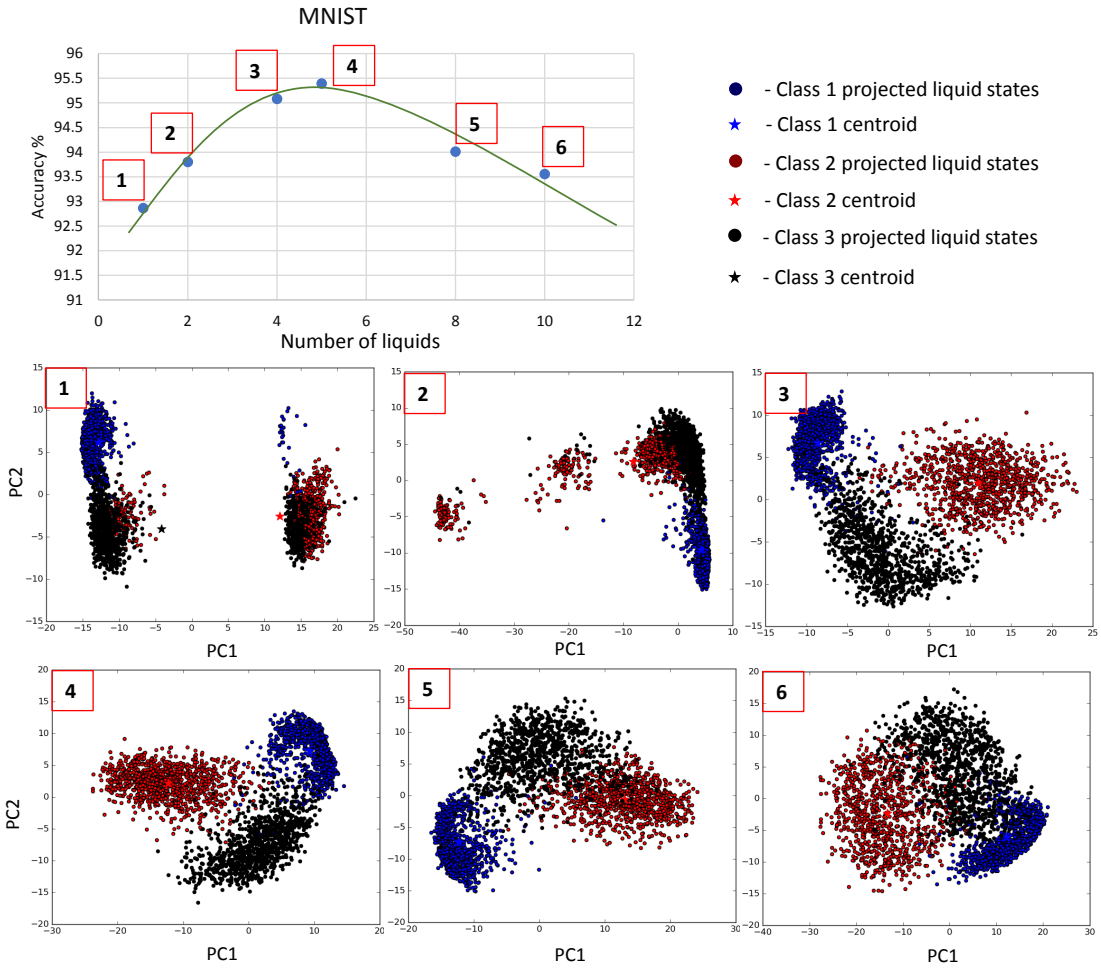

Fig. 4.10. The distribution of the liquid state vectors, as a projection to the first two principal components PC1 and PC2, for different number of liquids. The liquid state vectors (represented as dots) correspond to three classes in the MNIST image data set. Each class has randomly picked 1000 liquid state vectors. Distributions related to point 3 and 4 show less overlapping between classes, and the data points are more concentrated at the class mean points in contrast to 6, which has significant overlapping that caused the accuracy degradation.

overlaps among class boundaries as depicted in Figure 4.9(d), leading to accuracy loss as experimentally validated in Figure 4.7 across different applications.

In order to graphically view the variation in SP and AP with the number of liquids for the applications considered in this work, we used Principal Component Analysis (PCA) to plot the high-dimensional liquid states in a low-dimensional space. Generally, the first few principal components preserves the most variance in a given high-dimensional data set. Hence, the same object in multi-dimensional space can be visualized in low-dimensional space with insignificant changes. To create such a low-dimensional projection of the liquid state vectors for different input patterns, we reduced their dimension using PCA and plotted the two most significant Principal Components (PCs) corresponding to the two largest eigenvalues. Figure 4.10 plots the 800-dimensional liquid state vectors, projected to the two-dimensional space using the first two PCs, for 1000 randomly picked input patterns from three classes in the MNIST data set. Figure 4.10 clearly illustrates why the accuracy improves till the $N_{ens,opt}$ point and degrades beyond that as explained below. The single liquid case shows concentrated (low AP), but overlapped data (low SP). This is where the AP is the lowest due to the concentrated data points. As the number of liquids increases, the classes become clearly separated. Note that the points belonging to the same class also moves away from their respective centroids due to the increased AP. This ultimately results in the aforementioned overlapping between the classes for number of liquids larger than $N_{ens,opt}$, which gives rise to more misclassifications.

4.7 Benefits of the ensemble approach

The ensemble of liquids approach creates smaller liquids where the dynamics of one network does not affect another. When evaluating the spike propagation within the liquids, these smaller liquids can be run independently and in parallel. Since the evaluation time is a higher order polynomial function of the number of neurons, computing few smaller liquids in parallel instead of computing one large liquid is beneficial in terms of reducing the inference time. Note that the evaluation of a large liquid can also be parallelized. The liquid dynamics vary temporally, and for digital

simulations, it can be divided in to multiple time steps. Each evaluated neuron state in the liquid at one time step is temporally correlated to that of the next time step. Therefore, the liquid evaluation process cannot be temporally divided for parallelizing the operation. Furthermore, since all the neurons are connected to each other (with a given sparsity), the dynamics of one neuron is dependent upon that of other neurons connected to it. Therefore, 'fully independent' simulations are also not possible at the neuron level. However, the matrix-vector manipulations involved in each time step *can* be parallelized. Simply put, in finding the pre-synaptic currents of the neurons, the matrix-vector multiplication between the spiking activity and the weight matrix must be evaluated as shown below (with respect to excitatory neurons for example).

$$\Delta g_e(t_i) = WS(t_i) \tag{4.12}$$

where $\Delta g_e(t_i)$ is the instantaneous jump of conductance at time t_i (refer to Equation 4.2), $S(t_i)$ is the spiking activity vector of N number of neurons in the liquid at time t_i, and $W \in \mathbb{R}^{N\times N}$ is the connection matrix that defines the liquid. Consider dividing the above process in to multiple processing cores. The division of the operation in to two cores using row-wise striped matrix decomposition requires the matrix W to be divided in to two parts (Figure 4.11(a)). During each simulation time step (t_i), each core evaluates membrane potentials $S_1(t_{i+1}) = \big[s_1(t_{i+1}), ..., s_{N/2}(t_{i+1})\big]$ and $S_2(t_{i+1}) = \big[s_{N/2+1}(t_{i+1}), ..., s_N(t_{i+1})\big]$. For the next time step, these S_1 and S_2 must be concatenated and requires communication between cores. In contrast, a concatenation is not required until the end of the total simulation duration (T) in our ensemble approach (Figure 4.11 (b)). Due to the lack of communication overhead between processors, the ensemble approach is faster than a parallelized version of the single liquid baseline among N_{ens} number of processors. In fact, due to the aforementioned communication overheads, efficient parallel processing can be hindered even in Graphical processing units (GPUs) [162]. However, in any method of evaluating the liquid dynamics, note that the ensemble approach has less number of connections than a single liquid baseline. Therefore, the ensemble approach has reduced amount

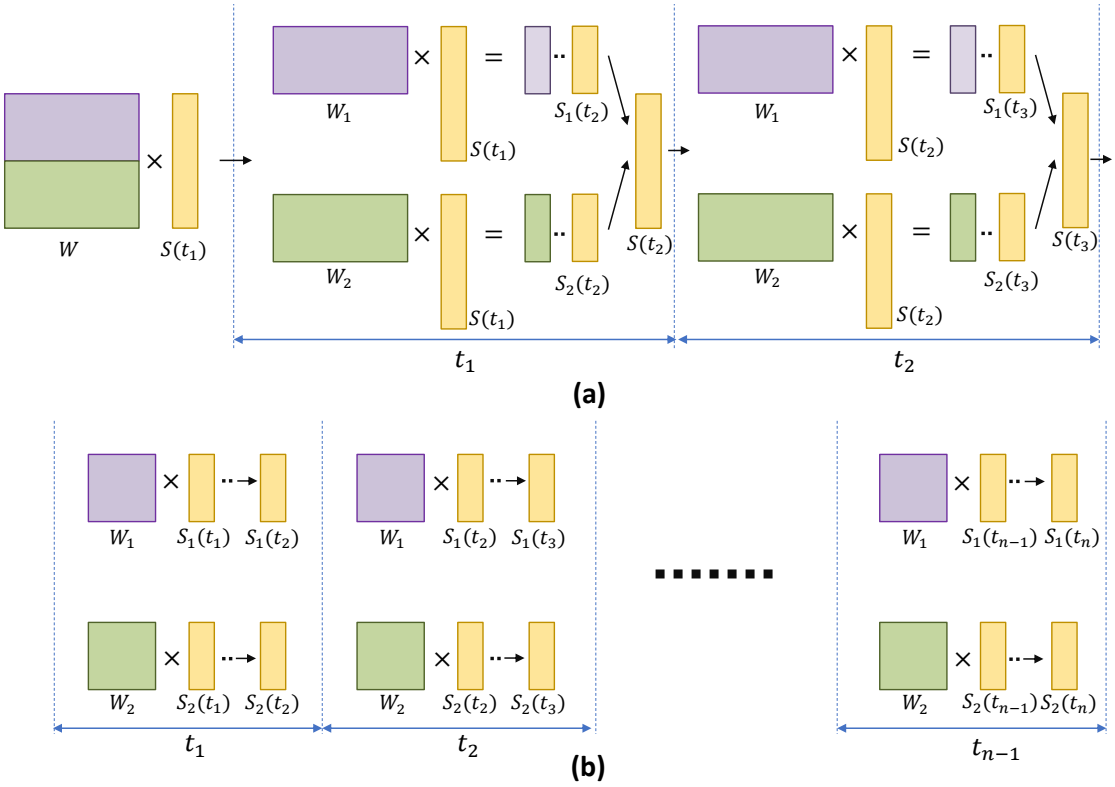

Fig. 4.11. (a) The division of matrix-vector multiplication using row-wise striped matrix decomposition, for the single liquid baseline LSM. Note that during each time step, the generated S_1 and S_2 vectors need to be concatenated to form the S vector (represents the spiking activty of the liquid), which requires communication between cores. (b) The "embarrassingly parallel" nature, and the reduced amount of operations in the ensemble approach allows two small liquids to run in parallel as two independent tasks, until the end of the last simulation time step t_n.

of computation leading to lower evaluation time. Different studies have shown designing hardware accelerators for spiking neural network platforms [163–165]. In the context of reducing the design complexity, above methods could potentially benefit from the low connection complexity, and "embarrassingly parallel" nature [166] of our ensemble approach.

The inference time is the addition of the liquid evaluation time and the classifier evaluation time. The liquid evaluation time was calculated by giving 100 input instances to the LSM model solver and estimating the average liquid computation time per input. The classifier evaluation time is significantly lower than the liquid computation time ($\sim \times 10$). Note that the classifier training time is similar in the baseline (single liquid LSM) and the ensemble approach, since there are equal number of neurons in the liquid and the number of trained weights are the same.

Once an LSM is trained, the connections within the liquid and the classifier weights must be stored. LSMs with large liquids require more space. In the ensemble approach, the number of connections within the liquid are significantly lower than the single liquid baseline. For example, assume dividing a liquid with N_{tot} number of neurons in to N_{ens} number of smaller liquids with $\frac{N_{tot}}{N_{ens}}$ amount of neurons in each of them. The number of connections available within the liquid for the single liquid baseline is $\sim N_{tot}^2$ whereas the number of connections in the multi-liquid case is $\sim (\frac{N_{tot}}{N_{ens}})^2 N_{ens} = \frac{N_{tot}^2}{N_{ens}}$. This shows that the number of connections reduces by a factor of N_{ens} when dividing a large liquid into N_{ens} smaller liquids, given that the percentage connectivity stays the same. Figure 4.12(a) and Figure 4.12(b) illustrate how the memory requirement varies for different number of liquids for the MNIST image recognition and TI-alpha speech recognition applications, respectively. When the optimum accuracy point for the ensemble approach is considered, we witnessed 87% reduction in the amount of memory, 55% reduction in inference time, and a 7.3% improvement in accuracy simultaneously, for the TI-alpha speech recognition application. For the MNIST handwritten digit recognition application, we witnessed 78% reduction in the amount of memory, 72% reduction in the inference time, and 3.9% improvement in classification accuracy.

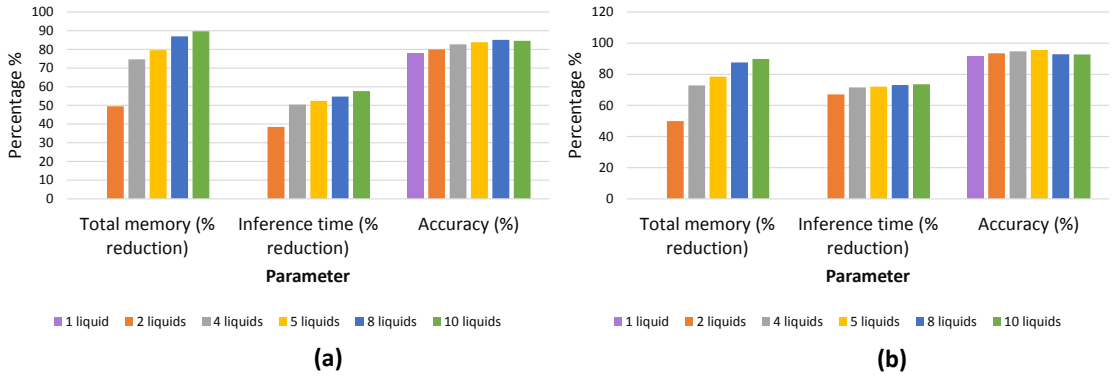

Fig. 4.12. The total memory reduction (%), inference time reduction (%) with respect to the baseline, and accuracy for different number of liquids in the ensemble. Two applications were considered; (a) temporal data classification problem (TI-alpha) (b) spatial data classification problem (MNIST).

4.8 Conventional methods of improving the accuracy versus the ensemble approach

The simple structure and training of LSMs, come with an accuracy trade-off, when compared with other non-reservoir computing techniques such as LSTM networks [167]. Different mechanisms have been studied in the literature such as training the connections in the reservoir [134], using expensive learning rules (for example, backpropagation through time [167]), and selecting complex architectures [28], in order to improve the accuracy of liquid state machines. However, these methods will increase the complexity of the LSM resulting in poor performance with respect to latency, despite the higher accuracy. Furthermore, a liquid can be considered as a universal computational medium. A single liquid with multiple trained readouts can be used for multiple applications [168]. Above methods such as training the connections within the liquid will make the LSM restricted to one application. In this section, we will explain two basic methods of improving accuracy, while leaving

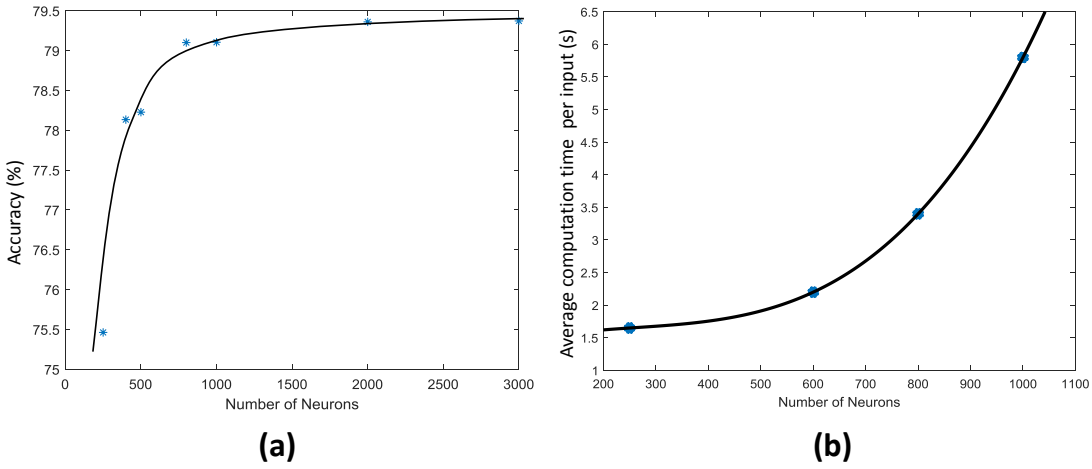

Fig. 4.13. (a) The accuracy of an LSM with a single liquid, measured at different number of neurons, for a speech recognition application (TI-alpha). (b) The average liquid evaluation time of an LSM measured at different number of neurons

the structural and training simplicity of LSMs intact, and compare the results with the ensemble approach.

4.8.1 Increasing the number of neurons in the liquid

As explained in [143], sufficiently large and complex liquids possess large computational power. Increased number of neurons in the liquid will result in increased number of variables for the classifier. Based on 'multiple linear regression' methods of predicting a function, increased number of predictor variables (in this case the number of neurons), will result in better prediction [126, 169]. Therefore, increasing the number of neurons will improve the prediction accuracy of the LSM. Note however that using enormous number of predictor variables/neurons will make the network suffer from overfitting. Figure 4.13(a) shows how the accuracy of an LSM varies with the number of neurons in the reservoir for the TI-alpha speech recognition task. As Figure 4.13(a) illustrates, the accuracy initially increases with the number of neu-

rons and then saturates after a certain point. Increased number of neurons implies increased connections within the liquid, given that the percentage connectivity stays the same. The number of connections within the liquid shows a square relationship $\sim \nu N_{tot}^2$ with the number of neurons N_{tot}, where ν is the global percentage connectivity. Due to this, evaluation time of the liquid increases exponentially as shown in Figure 4.13(b). Therefore, when the number of neurons are already high, the accuracy improvement we obtain by further increasing the number of neurons is not worth the resultant performance and storage requirement penalty. Furthermore, the accuracy saturates around $\sim 79.2\%$ for the TI-alpha application (for $N_{tot} \geq 800$). Note that we have also adjusted the percentage connectivity at each point in the graph, to get the best accuracy for a given number of neurons. However, the ensemble approach for $N_{tot} = 1000$ and $N_{ens} = 4$ gives $\sim 83\%$ accuracy, which is larger than the accuracy achievable by increasing the number of neurons in a single liquid.

4.8.2 Percentage Connectivity within the liquid

The percentage connectivity within the LSM is an important measure of the spiking activity of a liquid. The spiking activity of the liquid could show two negative behaviors which could drastically reduce the accuracy of the network, *viz.* pathological synchrony and over-stratification [170]. Pathological synchrony occurs when the neurons get caught in infinite positive feedback loops resulting in heavy continuous spiking activity. Over-stratification can be defined as the opposite extreme of the above. Here, the neurons do not propagate an input signal properly, resulting in reduced spiking activity. Both the above behaviors result in similar outcomes for input instances of different classes (hence poor separation between classes), making classification tasks hard. We noticed that lower connectivity ($P_{In \rightarrow E}$) results in over-stratification (Figure 4.14(a)) whereas higher connectivity results in pathological synchrony (Figure 4.14(b)).

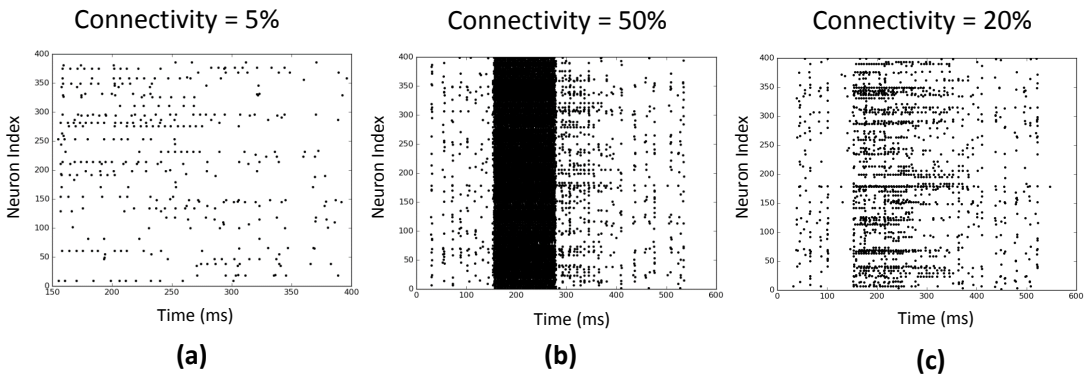

Fig. 4.14. Illustration of two negative behaviors of an LSM at different input to liquid percentage connectivity values. Each raster plot shows the spiking activity of the liquid neurons over time. The application is a speech recognition task (TI-alpha). (a) Over-stratification at low percentage connectivity. (b) Pathological synchrony at higher percentage connectivity. (c) An instance that shows clear differences between spiking activity of the liquid neurons in contrast to (a) and (b)

We changed the percentage connectivity between different combinations of pre- and post-neuron types ($E-E, I-E, E-I$) till we obtain good accuracy avoiding pathological synchrony and over-stratification (Figure 4.14 (c)). After that, we refined the input-liquid connectivity for further accuracy improvement. Figure 4.15(a) shows how the accuracy changes with the percentage connectivity of the input to liquid connections. Liquids with different number of neurons have different optimum connectivity values as shown in Figure 4.15(b). The application is recognizing spoken letters in TI-alpha speech corpus. The maximum accuracy achievable by changing the percentage connectivity ($P_{IN\rightarrow E}$) for $N_{tot} = 1000$ is $\sim 79\%$ (refer to the green colored trend in Figure 4.15(a)). This is smaller than that achievable ($\sim 83\%$) by our ensemble approach with $N_{tot} = 1000$ and $N_{ens} = 4$.

Furthermore, we simultaneously changed the $P_{E\rightarrow E}$ and $P_{IN\rightarrow E}$ percentage connectivity values of LSMs with different number of liquids, and evaluated the accuracy. Four $P_{IN\rightarrow E}$ values $(0.1, 0.2, 0.4, 0.6)$ and three $P_{E\rightarrow E}$ values $(0.2, 0.4, 0.6)$ were se-

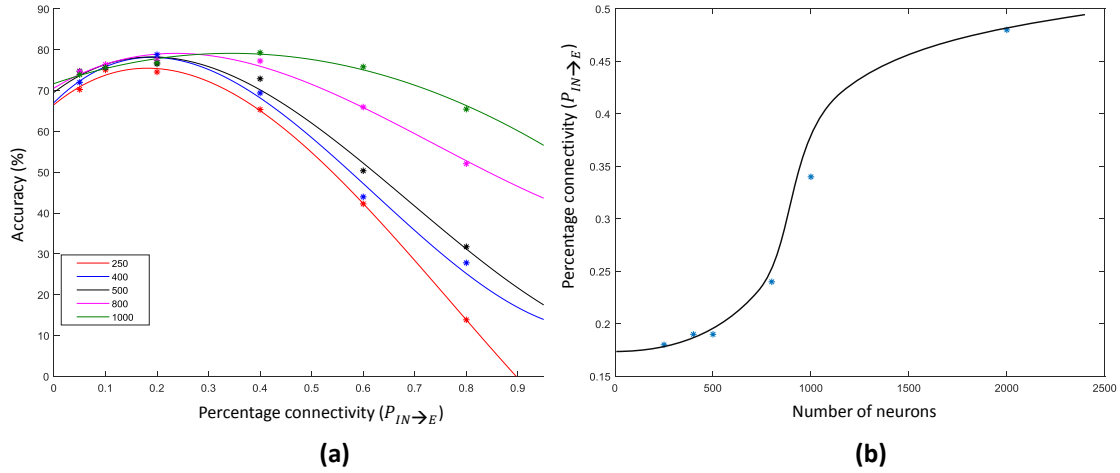

Fig. 4.15. (a) The accuracy trend with varying input – liquid percentage connectivity, for different number of liquid neurons. The experiment is done on a single liquid LSM conducting a speech classification task (TI-alpha). (b) The percentage connectivity that gives the best accuracy at different number of neurons.

lected for the experiment. The summarized results are illustrated in the 3D plot in Figure 4.16. The color code of the figure gives the accuracy of a particular combination of connectivity values. Across all LSM configurations with different number of liquids, we witnessed that higher $P_{IN\rightarrow E}$ and higher $P_{E\rightarrow E}$ results in accuracy degradation. Sparser connectivity gives better results. As the figure illustrates, at sparser connectivity values, a single liquid LSM offers lower accuracy than an LSM with N_{ens} liquids (refer to the upper left corner of the 3D plots). The 'maximum capacity' of each LSM configuration (for a given number of liquids) is plotted in Figure 4.17(a). The 'maximum capacity' is the best accuracy attainable from a particular liquid configuration, after optimizing the percentage connectivity values (in the selected range). As Figure 4.17(a) illustrates, maximum accuracy obtained from the single liquid configuration is smaller than that of other configurations. We also plotted the average accuracy of a given LSM configuration across all percentage connectivity values (Figure 4.17(b)). The average accuracy to some extent could be thought of as the outcome

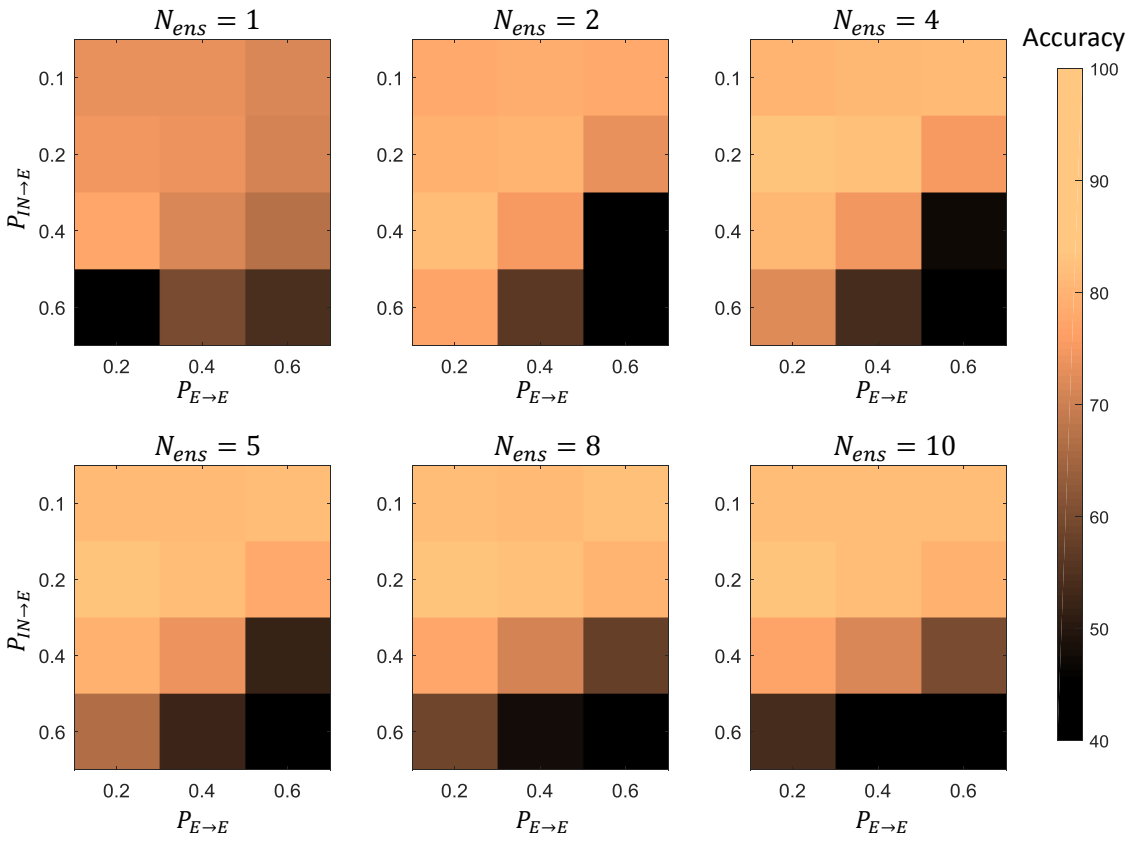

Fig. 4.16. The accuracy of LSMs with $N_{ens} = 1, 2, 4, 5, 8, 10$ at different percentage connectivity ($P_{E\to E}$ and $P_{IN\to E}$) values, for the TI46-alpha classification task

one would witness in a given LSM configuration for an arbitrarily selected connectivity value (within the specified sparse connectivity region of the experiment). The average accuracy of the single liquid LSM configuration is lower than that of multiple liquids.

In section 4.7, we explored the benefits of the ensemble approach due to reduced number of connections in the liquid. A single liquid LSM configuration has N_{ens} times more number of connections as the LSM with N_{ens} number of liquids as explained in section 4.7. In order to view if a single liquid with sparser connectivity offers better accuracy than an LSM with N_{ens} number of liquids and higher percentage connec-

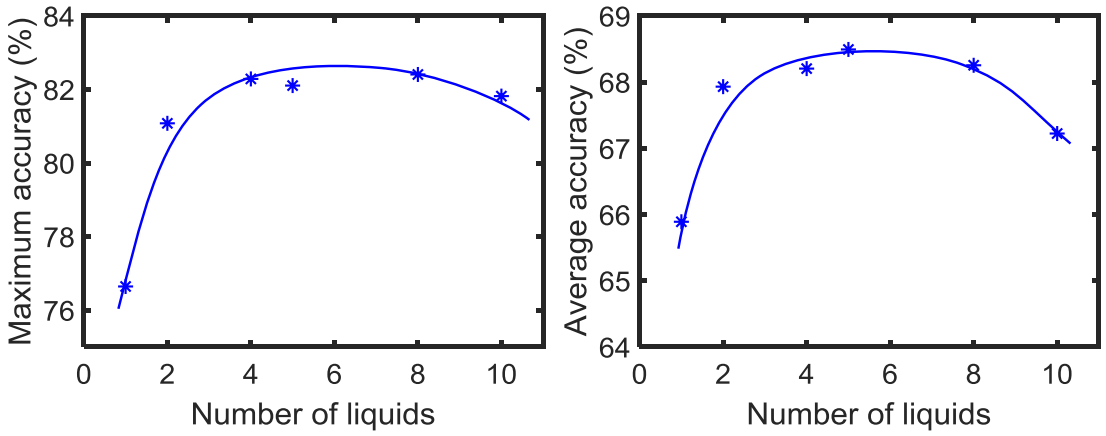

Fig. 4.17. (a)The maximum accuracy among all the LSM configurations with different $P_{IN\to E}$ and $P_{E\to E}$ (b)The average accuracy across all the LSM configurations with different $P_{IN\to E}$ and $P_{E\to E}$

tivity, we conducted an experiment. In other words, the goal of the experiment is to view the accuracy of two LSM configurations with same number of connections. The dominant component of the number of connections in an LSM is the connections between the excitatory neurons. Therefore, we varied the $P_{E\to E}$ for two LSM configurations ($N_{ens} = 1$ and $N_{ens} = 4$) and observed the accuracy for the TI46-alpha application. Figure 4.18 illustrates that for $P_{E\to E} < 0.57$, the multiple liquid configuration ($N_{ens} = 4$) provides better accuracy, suggesting that the ensemble approach gives better results even under same number of connections in comparison to the single liquid baseline. For example, the ensemble approach gives $\sim 83\%$ accuracy at $P_{E\to E} = 0.4$ and for the same number of connections, (i.e. at $P_{E\to E} = 0.1$), the single liquid LSM configuration gives lower accuracy ($\sim 76\%$). However, for $P_{E\to E} > 0.57$, single liquid LSM seems to perform better. Hence we conclude that at higher degrees of sparsity, the ensemble approach performs better than a single liquid baseline with the same number of connections.

Apart from the percentage connectivity, different connectivity patterns within the liquid were also considered in literature. For example, a probabilistic local connectiv-

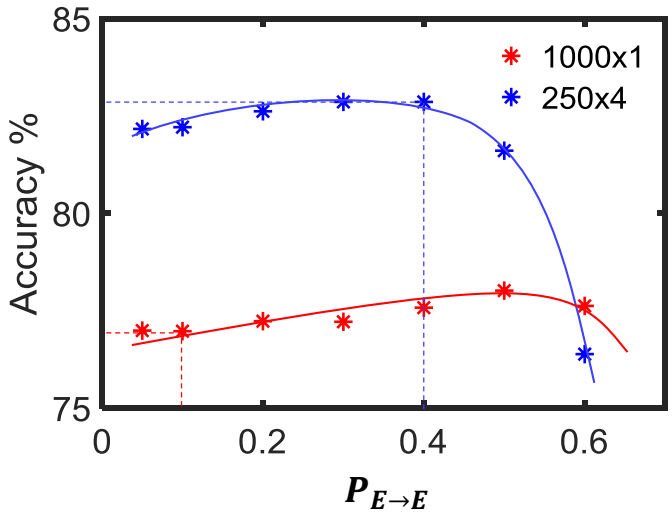

Fig. 4.18. The accuracy variation of the single liquid LSM baseline and ensemble approach ($N_{ens} = 4$) at different percentage connectivity values ($P_{E\to E}$). The accuracy of the ensemble approach at $P_{E\to E} = 0.4$ is higher than the accuracy of the single liquid LSM at $P_{E\to E} = 0.1$. Note that the number of connections in both the cases considered are the same. The accuracy was evaluated on the TI46-alpha classification task

ity within the liquid, inspired by the connectivity in biological neurons is suggested in [143]. As explained in [16], [171], and [172], the further apart two neurons are, more 'likely' it is that they are not connected to each other. The existence of a connection between two neurons (all the neurons were arranged in a 3D lattice initially) is hence modeled as a probabilistic function (P_{con}) of the Euclidean distance ($D^2(a, b)$) between neuron a and b. λ is a parameter which controls both the average number of connections and the average distance between two neurons. C is the highest probability that two neurons are connected (occurs at $\lambda \to \infty$). Multiple values of C ($i.e., C_{E\to E}, C_{E\to I}, C_{I\to E}$) can be used depending upon the type of pre- and post-neurons being connected. The probabilistic function P_{con} is as described by the following equation:

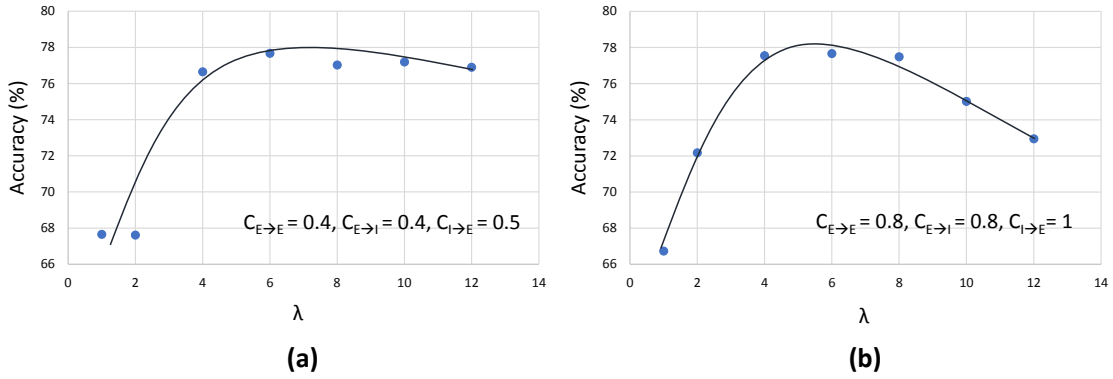

Fig. 4.19. The accuracy varying with λ, for two different sets of C parameter selections, (a) $C_{E\to E} = 0.4, C_{E\to I} = 0.4, C_{I\to E} = 0.5$, and (b) $C_{E\to E} = 0.8, C_{E\to I} = 0.8, C_{I\to E} = 1.0$. This is a single liquid-LSM, which classifies speech patterns in the TI-alpha dataset. There are 1008 neurons in the liquid which are arranged in a $6 \times 6 \times 28$ sized 3D column.

$$P_{con} = Ce^{-\frac{D^2(a,b)}{\lambda^2}} \tag{4.13}$$

Figure 4.19 shows how the accuracy varies with λ for two different sets of C values in a liquid that has 1008 neurons. The application is recognizing the utterances of the letters in the English alphabet (TI-alpha). The neurons were arranged in a 3D column ($6\times6\times28$) approximately preserving the dimensional ratios of the neural microcircuit column proposed in [143]. As Figure 4.19 illustrates, the highest accuracy attained was $\sim 78\%$. The accuracy of an LSM (1000 neurons, refined percentage connectivity values) with randomly pruned connections (without considering the probabilistic local connectivity model) was 77.6% for the same application.

The ensemble approach, to a certain extent, is similar to the aforementioned notion of 'local connectivity', since we split up a large liquid into multiple smaller liquids which avoids long connections. In the probabilistic local connectivity model, the dynamics of each neuron in the liquid are dependent upon that of other neurons, since they are all directly or indirectly connected. In contrast, the ensemble approach

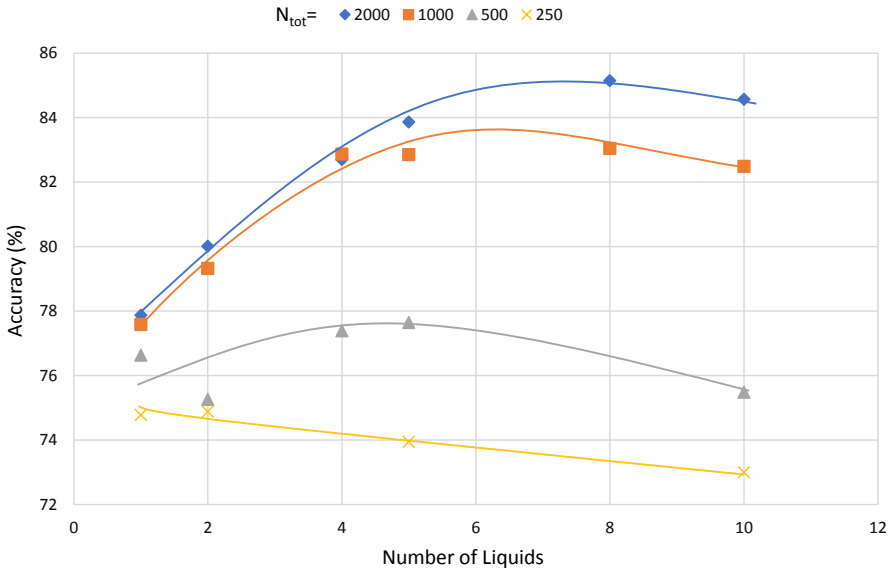

Fig. 4.20. The accuracy varying with the number of liquids in the ensemble approach, for different total number of neurons (N_{tot}). The LSM classifies speech data in the TI-alpha dataset.

allows multiple small networks to be evaluated in parallel since they are independent, making the evaluation time smaller. Furthermore, the accuracy of four liquids with 250 neurons in each is 83%, which is considerably higher than the highest accuracy attainable (for the ranges of parameters we have considered) through probabilistic local connectivity (78% for 1008 neurons) model, for the same speech application.

4.9 Limitations of the ensemble approach

In this section, we analyze whether dividing a liquid with any number of neurons (N_{tot}) would result in similar accuracy improvements. To this effect, we created ensembles of liquids with different total number of neurons (N_{tot}). As Figure 4.20 illustrates, liquids with large number of neurons show clear sign of accuracy improvement when divided into smaller liquids. However, when the number of neurons is smaller, dividing the liquid may result in decreased accuracy. For example, note that

the accuracy reduces continuously when a liquid with 250 neurons is divided. This result is similar to the observation in [29], where the authors have shown that the input and liquid subdivision is beneficial for LSMs with large number of neurons. Similarly, here the ensemble approach makes sense only for LSMs with large number of neurons in them. In conclusion, we state the following with respect to the applicability of the ensemble approach for LSMs. In order to improve the accuracy of an LSM, the number of neurons can be increased. However, beyond a certain point, accuracy does not improve further. In such a case, the ensemble approach can be utilized to further increase the accuracy. Such accuracy improvements are not attainable by means of other simple methods that preserve the structural and training simplicity of the standard LSM, such as changing the connectivity.

4.10 Multiple liquid-multiple readouts (MLMR) approach

When moving from the single liquid approach to the ensemble of liquids approach, any benefit in terms of classifier training time was not observed. This is due to the fact that the number of total liquid neurons is the same, and we are using a single classifier. In this section, we analyze, if including a readout at the end of each small liquid is beneficial than having a single readout for all the liquids. The basic structure of this multiple liquid-multiple readouts (MLMR) approach is shown in Figure 4.21.

In contrast to our previous approach, this structure could be viewed as a collection of small LSMs (s-LSMs). Each s-LSM is trained individually, and the final classification is done by considering either the maximum outcome, or the majority vote among all the local classifiers. During training, we do not use all the training data points for each local s-LSM classifier. Instead, we divide the training space among the ensemble of s-LSMs based on the following two criteria:

1. Random training space division (RD)

2. Clustered training space division (CD)

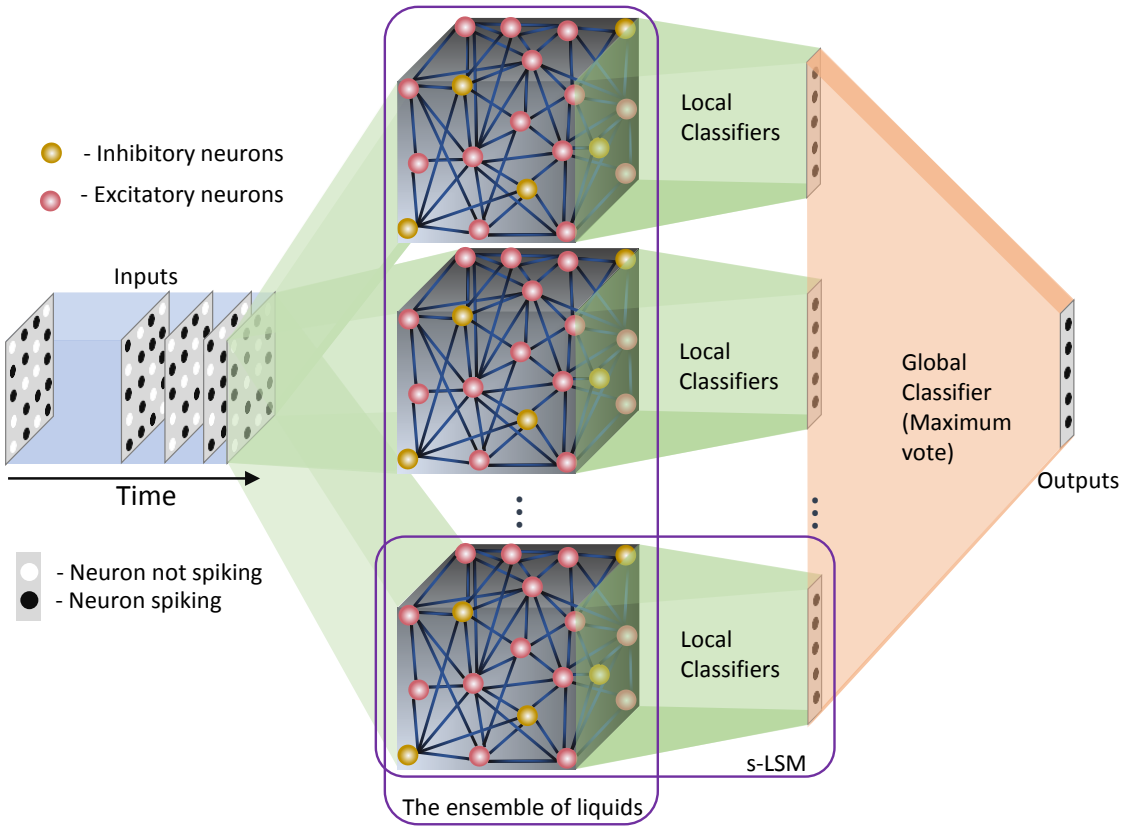

Fig. 4.21. The structure of the multiple liquid-multiple readout (MLMR) approach. There are multiple small liquids, and individual classifiers at the end of each liquid. These are defined as small LSMs or s-LSMs. Final outcome (global output) is calculated by considering the maximum vote among all the local classified outputs from the s-LSMs (local outputs)

In random training space division (RD) method, we randomly divide the training data space among the ensemble of s-LSMs, and feed them to obtain the corresponding liquid state vectors at the output of each liquid. These state vectors were then used to train the local classifiers attached to each s-LSM in the ensemble using gradient descent error backpropagation. For example, if there are N_{ens} number of s-LSMs and N_{train} number of examples in the training set, each s-LSM will be trained with $\frac{N_{train}}{N_{ens}}$ number of randomly picked training examples. On the other hand, in the clustered input space division (CD) method, we divide the training instances into certain clusters depending upon their features, (for instance, we have selected 'original', 'rotated',

'shifted' and 'noisy' images from the extended MNIST dataset as clusters) and used them to train each readout. Here, an s-LSM has specific knowledge about the cluster of examples that it is trained with, and zero knowledge about other clusters. Therefore, an s-LSM may not correctly identify an input that belongs to a different cluster, apart from what it was trained with, leading to large accuracy degradation at the global classifier. For example, if a rotated image of digit '1' is given as the input, the s-LSM that was trained with rotated images will correctly recognize the given image. *i.e.*, the output neuron-1 gives the highest outcome (there are 10 output neurons and they are indexed as neuron-0 through neuron-9 as shown in Figure 4.22). Other s-LSMs may not recognize this input correctly, potentially leading to another neuron apart from neuron-1 to give a high value at the outputs of their corresponding classifiers. When getting the final outcome using 'maximum output' method, the neuron that gives the highest value over all the s-LSMs may not be neuron-1. Instead, it could be some different neuron from an s-LSM that was not trained with rotated images. To address this issue, we use an 'inhibition' criterion to suppress the s-LSMs from giving high outputs for cluster types that they are not trained with. Initially we divide the training space into clusters along with their standard target vectors (vectors of which the length is equal to the number of classes L. If the input belongs to the i^{th} class, then the i^{th} element in the vector will be '1' and the other elements will be '0'. Refer to Figure 4.22). Then, we randomly select 10% of the training instances from each cluster (foreign instances), and add them to the training space of all other clusters. The target vectors of the foreign instances are forced to have all their elements equal to $1/L$ (L is the number of classes) and we name this target vector as 'inhibitory label vector'. This will force each s-LSM outcome to be low, when the presented input does not belong to the cluster with which the s-LSM was trained. This method is explained graphically in Figure 4.22, by means of an example.

We used the handwritten digit recognition application with the extended MNIST data set, to check the accuracy, performance, and training time of the aforementioned methods. The training data set was divided into 4 clusters; original MNIST images,

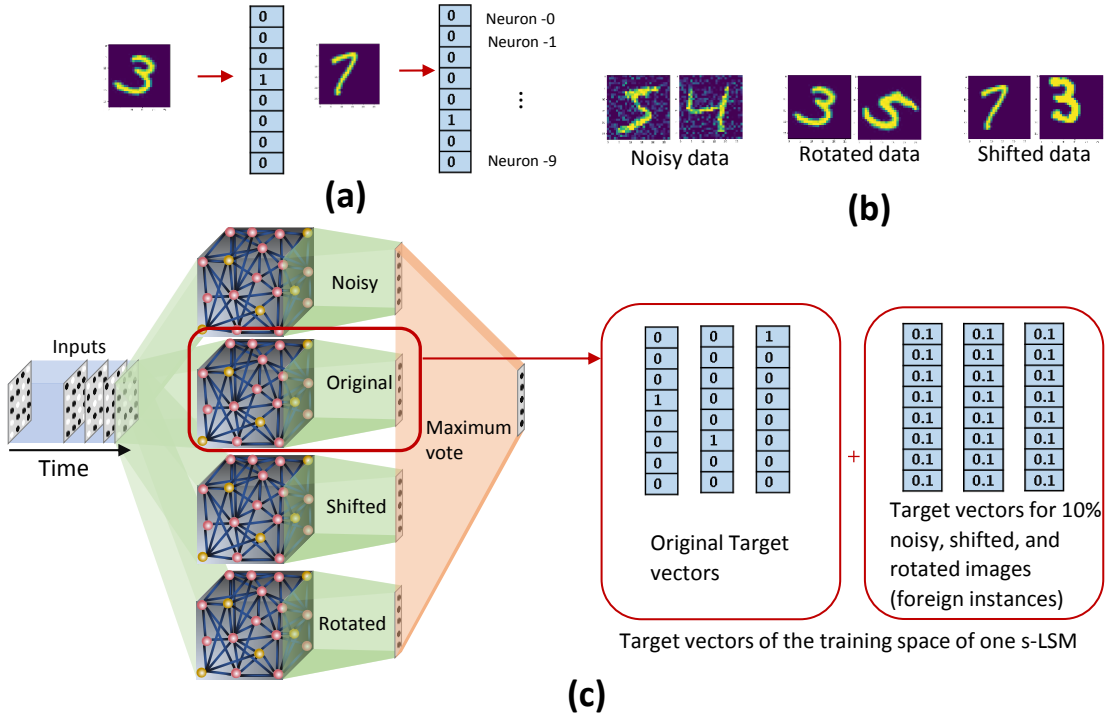

Fig. 4.22. (a) Target vectors that correspond to images in the extended - MNIST data set. (b) Examples from the clustered training data space of the extended MNIST dataset. (c) The clustered training space division method. Each s-LSM is trained with a particular cluster of images, and an additional 10% of the images in the other clusters (foreign data). The target vectors of the foreign data are modified to have each value equal to 0.1

noisy images, rotated images and shifted images. Total number of neurons were 1000 and each s-LSM has 250 neurons. The connectivity is set as indicated in Table 4.1. Table 4.3 reports the accuracy of the above explained two training space division methods (RD and CD) along with the accuracy of the baseline (single liquid with 1000 neurons). The accuracy of the two methods (RD $\rightarrow$ 82.5% and CD $\rightarrow$ 83.1%) are inferior to that of the baseline (86.9%).

When comparing with RD method, CD gives better accuracy for the same number of neurons. The reason for this can be explained as follows. The clusters in the

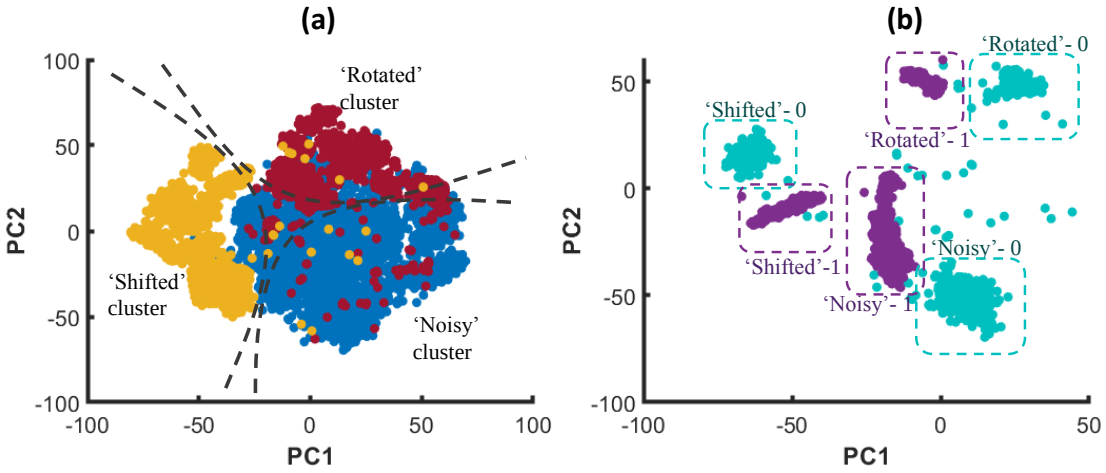

Fig. 4.23. (a) The t-Distributed Stochastic Neighbor Embedding (t-SNE) of the high dimensional input data points, in 2D space for better visualization. The distribution of data points that belong to three clusters ('Noisy', 'Shifted', and 'Rotated') stay spatially separated. (b) The distribution of data points of digit '1' and digit '0' that belongs to three clusters.

Table 4.3.
Accuracy of different ensemble approaches

Approach (total number of neurons= 1000)	Accuracy (%)
Single liquid, single readout (baseline)	86.9
4 ensembles, single readout (MLSR)	89.0
4 ensembles, 4 readouts, (MLMR) random training space division (RD)	82.5
4 ensembles, 4 readouts, (MLMR) clustered training space division (CD)	83.1

training dataset can have different overlapping/non-overlapping distributions. For instance, three clusters ('noisy', 'shifted', 'rotated') in the considered example in

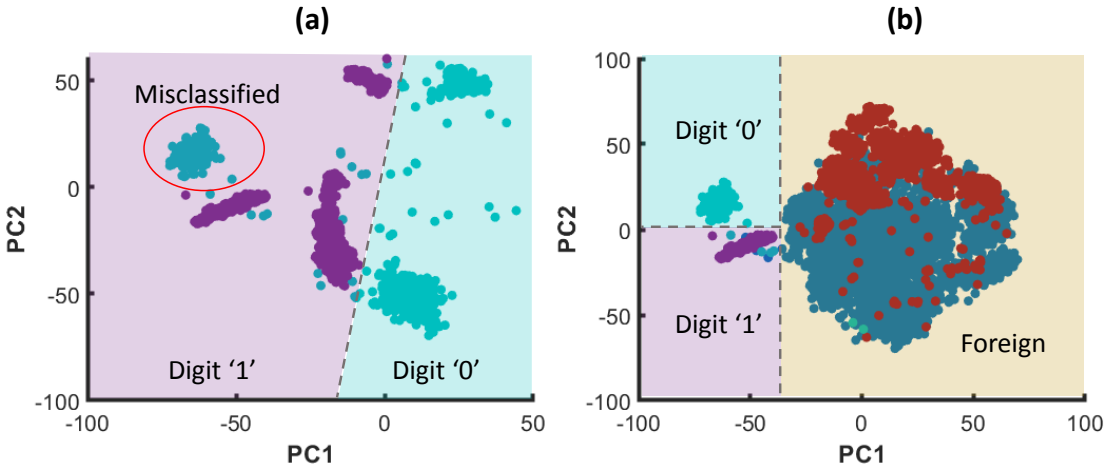

Fig. 4.24. (a) The graphical representation of the RD method trying to classify digit '0' and digit '1' that belong to three clusters using a linear classifier. Note that the digit '0' that belong to the shifted cluster is misclassified as digit '1'. (b)The graphical representation of the CD method trying to classify digit '0' and digit '1'. The particular classifier shown has learned to correctly classify digit '0' and digit '1' that belong to 'shifted' cluster. Furthermore, it recognizes the data points that belong to foreign clusters due to the proposed inhibition criterion. The dashed lines show the classifier decision boundaries.

this work follow three different distributions as shown in Figure 4.23(a). The figure elaborates the t-Distributed Stochastic Neighbor Embedding (t-SNE) [173] of the high dimensional images that belong to the aforementioned three clusters, for better visualization in the lower dimensional space (2D). Due to this separate distributions, examples that belong to the same class but in different clusters may not spatially stay together in the higher dimensional space. For example, Figure 4.23(b) shows the data points that correspond to digit '0' and digit '1' in different clusters, and neither the data points of digit '0' nor '1' stay together. Let us consider the RD method, and how it tries to classify the aforementioned digit '0' and digit '1'. If there are N_{tot} amount of training examples and L classes, the number of examples that belongs to class i each classifier sees is $N_{ex,i} = \frac{N_{tot}}{L \times N_{ens}}$. The $N_{ex,i}$ number of examples a classifier

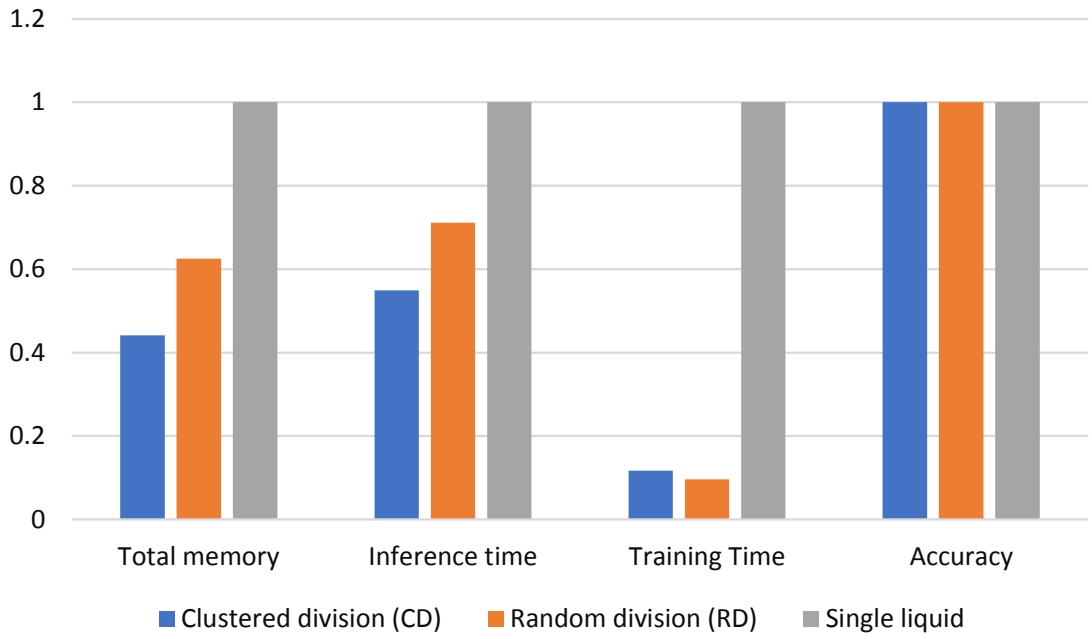

Fig. 4.25. Normalized total memory requirement, inference time, and training time of the clustered training space division method (CD), random training space division method (RD), and the single liquid baseline. The results are under iso-accuracy conditions.

in RD method sees belongs to N_{ens} number of clusters and they are distributed all over as shown in Figure 4.23(b). According to the figure, the two classes are not linearly separable. Therefore, the RD method leads to more misclassifications as elaborated in Figure 4.24(a). In contrast, a classifier trained for 'shifted' data cluster in CD method fits to a decision boundary that classifies digit '0' and digit '1' that *onlybelongs* to 'shifted' data cluster. Owing to the proposed inhibition criterion, the classifier trained for 'shifted' examples in CD tries to put the data points that belong to other clusters into a single category. As the Figure 4.24(b) illustrates, the classes: 'digit 0', 'digit 1', and 'foreign' are more linearly separable by CD method than the RD method, and this leads to higher accuracy in RD method.

Selecting more foreign examples would result in the classifier to concentrate more on fitting the foreign data into 'inhibitory label vectors', instead of classifying data in the corresponding cluster. Consider adding an $x_f\%$ of foreign instances per cluster. This would result in adding $\frac{(N_{ens}-1)x_f\%}{(N_{ens}-1)x_f\%+1}$ overall percentage of extra data that does not belong to the cluster to which the classifier must be trained. The training space of the classifier that must be trained for 'shifted' images consists of the following 'sets': 1)shifted digit '0', 2) shifted digit '1',..., 10)shifted digit '9', randomly selected images from 11)'noisy' cluster, 12) 'original' cluster, 13)'rotated' cluster. We selected a percentage, that will pick approximately equal number of data points from each of the aforementioned 13 sets. In our particular example, to make $\sim 7.7\%(= 100/13)$ of the training space to be attached to each of the above 'sets', $x_f\%$ needs to be selected as 10%. The total percentage of the foreign instances are 23% $\left(= \frac{(N_{ens}-1)x_f\%}{(N_{ens}-1)x_f\%+1}\right)$ per cluster.

In order to see if there is any benefit in the MLMR approach when achieving a 'given' accuracy, we reduced N_{tot} in the baseline to match the accuracy of both the RD and CD methods. The memory requirement, inference time, and training time were calculated for two scenarios. First, N_{tot} in the baseline was selected such that both the baseline and the RD method have the same accuracy (82.5%). Second, N_{tot} in the baseline was selected such that it matches the accuracy of the CD method (83.1%). In each of the above scenarios, the obtained memory requirement, inference time, and training time values were normalized with respect to the baseline. These normalized values for the two cases are shown in a single graph in Figure 4.25. The CD method is better in terms of memory requirement and inference time, in comparison to the single liquid baseline and RD method. We calculated the total number of MAC (multiply and accumulate) operations during training to estimate the training time (it is a function of the number of neurons in a liquid, number of output neurons, and number of training examples). Lowest training time was achieved in the RD method. The CD method offers 56% reduction in memory and 45% reduction in inference time, with respect to the baseline. For a 1000 total number of neurons, the 4 ensemble case

with a single classifier (studied in section 4.6. Let us denote this method as multiple liquids, single readout or MLSR approach) resulted in 78% reduction in memory usage and 72% reduction in inference time along with 2.1% accuracy improvement (hence better than both CD and RD methods under the memory usage and inference time metrics). However, in terms of training time, the MLSR approach did not show any improvement, whereas the MLMR showed 88% reduction with respect to the baseline.

4.11 Conclusion

We have presented an ensemble approach for Liquid State Machines (LSMs) that enhances separation and approximation properties, leading to accuracy improvements. The separation property in LSMs measures the dispersion between projected liquid states from different classes, whereas the approximation property indicates the concentration of the liquid states that belong to the same class. The ratio between SP and AP (DR) is a measure of the class discrimination. We witnessed that the DR increases when a large liquid is divided into multiple smaller independent liquids across four speech and image recognition tasks. We observed the existence of an optimal number of liquids ($N_{ens,opt}$) until which the DR increases and saturates thereafter. Owing to the improvement in the DR in our proposed ensemble approach, we noticed an LSM accuracy enhancement with increasing number of liquids. The accuracy peaked at the same $N_{ens,opt}$ point at which each DR saturated, for different recognition tasks. This validated the existence of an optimal number of liquids which gives the best accuracy for the LSM, and this point is highly dependent upon the application and the total number of liquid neurons.

There is plethora of complex approaches that concentrate on improving the accuracy of LSMs, including learning the liquid connections [28, 134]. In contrast to such works, our proposed approach does not change the simple structure and training methods of LSMs. Furthermore, the ensemble approach gives better accuracy when compared with other simple mechanisms of improving the LSM accuracy such as in-

creasing the number of neurons, changing the percentage connectivity, and utilizing the probabilistic local connectivity models. Apart from providing improved accuracy, the proposed ensemble approach comes with other benefits including lower memory requirement and lower inference time. We have shown that creating an ensemble of liquids leads to lower inter-connections in comparison to a single liquid with the same number of neurons. Furthermore, the liquid evaluation can potentially be parallelized in the ensemble approach due to the existence of small independent liquids. This results in reduced LSM inference time. The accuracy improvement with increasing number of liquids in the ensemble becomes less evident when the total number of neurons is small. In fact, creating an ensemble of liquids with a small number of neurons will rather reduce the accuracy. Hence the ensemble approach makes sense for LSMs with large number of neurons [29].

Since there is no benefit in terms of training time between a single-liquid LSM and the proposed ensemble approach (MLSR), we investigated the MLMR approach where a classifier is added to each small liquid in the ensemble. By dividing the training example space to train each small LSM, we were able to attain significant benefits in terms of training time, when compared with MLSR approach. There are multiple classifiers that were trained independently in the MLMR approach, and the final output is the maximum vote of all the local classifiers. The set of multiple liquid-classifier units are in fact a collection of small LSMs (noted as s-LSMs). Despite the performance benefits during training, we noticed an accuracy degradation in the MLMR approach, when compared with both the MLSR approach and the single-liquid baseline LSM with equal number of liquid neurons. The reason for this can be explained as follows. The classifiers in each s-LSM are smaller than that of the baseline and the MLSR approaches. A large classifier (as in the baseline and MLSR approach) has more number of parameters and is capable of fitting in to an unknown function better than a small classifier [169], leading to improved accuracy. The work presented in this section is published in [174].

5. SUMMARY AND FUTURE WORK

In this work, the possibility of enhancing neuro-inspired computing by means of scalable algorithms and inherent device physics of emerging nanoscale resistive devices was observed. Chapter 2 elaborates the realization of a neuro-inspired solution for the constraint satisfiability problem. Magnetic Tunnel Junctions (MTJs) were shown to be mimicking certain complex dynamics of Cellular Neural Networks, and thus were useful in efficiently implementing the SAT solver. Certain device level modifications were involved such as 'seamless switching thickness' to match the functionality of the cellular neural network based Boolean satisfiability solver. The simulation results show faster convergence to a solution in the proposed system and the possibility of finding a solution in polynomial time to a significant fraction of 3-SAT problems ($N \leq 50$). Furthermore, the system is $\sim 2.6 \times 10^3$ times faster than a state of the art software SAT solver, Minisat, and robust to dimension variations in the range of -5% to 10%. The temperature increments further reduces the computation time of the solver and it is more power efficient than CMOS implementation of the cellular neural network based analog SAT solvers.

The proposed SAT solver provides a satisfying assignment for variables in a SAT problem which *has* a solution. If an unsatisfiable problem is assigned to the solver, the MTJ dynamics will keep on oscillating without converging. Hence the proposed hardware SAT solver is not useful in identifying the satisfiability of a certain SAT instance. However, given that the solver converges fast when there is a solution, it can be used in combination with other CMOS based solvers to identify the satisfiability of a SAT instance. One approach would be to use a predefined time duration until which the hardware solver tries to converge to a solution. If the solver converges within the given duration, it will provide a satisfying assignment to the variables in the given SAT instance (i.e, the instance is satisfiable). If the solver does not converge, it does

not imply that the problem is not satisfiable. Other mechanisms must be involved to check the satisfiability of the instance in such a situation. More analysis on the feasibility and potential benefits of the above must be studied.

In chapter 3, the possibility of using the inherent stochasticity of memristors to realize a deep stochastic spiking neural network was analyzed. The functionality of the neuron was replaced by stochastic binary memristors and the synaptic functionality was performed by multi-level memristors, creating the 'all-memristor neural network'. The work is an effort to incorporate three key features of the actual brain; low power operation, spike based communication, and stochasticity, in an ANN. It was shown how the training mechanism can be updated to suite the stochastic activation function of the neuron. Circuit and system level simulations show that the network is robust to voltage (< 200mV) and synaptic ($\sigma < 20\%$) variations. The network is $\sim 6.4\times$ energy efficient, and the area $\times$ delay product was ~ 8 times smaller than the CMOS implementation of the same spiking neural network. The benefits comes at the cost of loosing only an insignificant amount ($\sim 1\%$) of accuracy in a hand written digit recognition application.

In this work, the device of interest was an $a - Si$ based electro-checmical metalization (ECM) memristor. Other types of memristors can also be used to realize the deep SNN architecture. For example, a device that requires lower reset energy will be a good substitute. The reset energy is the second dominant component of the total energy consumption of the network. This is because of the fact that the reset must be conducted in the deterministic region of operation of a memristor, which requires a sufficiently high voltage pulse to ensure that the device has turned off. A volatile memristor has been proposed in [49] as a random number generator. The device switching dynamics are stochastic and the SET state of it is not stable hence it eventually (within $\sim 100\mu s$) goes to the RESET state. If such a device is used in our design, it will completely eliminate the requirement to force reset the memristors. The write voltage of the device is also lower ($0.5V$, for a $300\mu s$ pulse) when compared

with the $a - Si$ based ECM device in this work. Possible improvements with other novel devices must be investigated.

In chapter 4, a neuro-inspired scalable approach for improving the speed of large scale liquid state machines (LSMs) was proposed. The presented ensemble approach that enhances separation and approximation properties (SP and AP), further leads to accuracy improvements. It was witnessed that the DR (The ratio between SP and AP) increases when a large liquid is divided into multiple smaller independent liquids across four speech and image recognition tasks. It was observed that the existence of an optimal number of liquids ($N_{ens,opt}$) until which the DR increases and saturates thereafter. Owing to the improvement in the DR in our proposed ensemble approach, an LSM accuracy enhancement with increasing number of liquids was noticed. The accuracy peaked at the same $N_{ens,opt}$ point at which each DR saturated, for different tasks.

The proposed approach is simple when compared with other approaches that concentrate on improving the accuracy of LSMs, including learning the liquid connections [28, 134]. Apart from providing improved accuracy, the proposed ensemble approach comes with other benefits including lower memory requirement and lower inference time. We have shown that creating an ensemble of liquids leads to lower inter-connections in comparison to a single liquid with the same number of neurons. Furthermore, the liquid evaluation can potentially be parallelized in the ensemble approach due to the existence of small independent liquids. This resulted in reduced LSM inference time.

The proposed multiple liquid - multiple readout (MLMR) approach explores the possibility of assigning inputs from different clusters in the data set to different small liquids, and train each readout with a fraction of the full data set. This will significantly reduce the training time when compared with the single readout LSM architecture. The considered application was classifying digits in the extended MNIST data set, and the clusters were selected as 'Noisy', 'Original', 'Shifted', and 'Rotated' images. It would be beneficial to observe possibilities of using other forms of clusters.

For example, a set of digits in the MNIST data set can be assigned to a single small LSM (s-LSM) in the ensemble.

The separation and approximation properties which were used to quantify better class discrimination can be extended to find better LSM architectures. For example, different forms of multi-layer LSMs can be analyzed to identify whether they improve SP and AP. Having liquid ensembles as first layers (in a multi-layer LSM) will help in extracting subtle features of the inputs (since the ensemble approach enhances the separation property) and would potentially increase the overall network accuracy. Furthermore, in contrast to verifying whether a particular architecture gives better SP and AP, novel learning methods based on the metrics can be explored. We are currently focusing on using the SP and AP metrics on an echo state network (a recurrent neural network which has a pool of analog neurons connected to each other - 'the reservoir', and a readout layer. Simply put, it is the non-spiking counterpart of the liquid state machine) that converts voice commands in to handwriting.

REFERENCES

REFERENCES

[1] I. Present, "Cramming more components onto integrated circuits," *Readings in computer architecture*, vol. 56, 2000.

[2] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, 2017.

[3] E. D. Adrian, "The impulses produced by sensory nerve endings," *The Journal of physiology*, vol. 61, no. 1, pp. 49–72, 1926.

[4] C. A. Anastassiou, R. Perin, H. Markram, and C. Koch, "Ephaptic coupling of cortical neurons," *Nature neuroscience*, vol. 14, no. 2, pp. 217–223, 2011.

[5] M. D. McDonnell and L. M. Ward, "The benefits of noise in neural systems: bridging theory and experiment," *Nature Reviews Neuroscience*, vol. 12, no. 7, pp. 415–426, 2011.

[6] S.-h. Zhou, W. Zhang, and N.-J. Wu, "An ultra-low power cmos random number generator," *Solid-State Electronics*, vol. 52, no. 2, pp. 233–238, 2008.

[7] B. Hellwig, "A quantitative analysis of the local connectivity between pyramidal neurons in layers 2/3 of the rat visual cortex," *Biological cybernetics*, vol. 82, no. 2, pp. 111–121, 2000.

[8] K. Meier, "The brain as computer: Bad at math, good at everything else," *IEEE Spectrum*, vol. 31, 2017.

[9] B. Molnár and M. Ercsey-Ravasz, "Asymmetric continuous-time neural networks without local traps for solving constraint satisfaction problems," *PloS one*, vol. 8, no. 9, p. e73400, 2013.

[10] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and stdp integrated circuits," *IEEE transactions on biomedical circuits and systems*, vol. 6, no. 3, pp. 246–256, 2012.

[11] Y. Dan and M.-M. Poo, "Spike timing-dependent plasticity: from synapse to perception," *Physiological reviews*, vol. 86, no. 3, pp. 1033–1048, 2006.

[12] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transactions on circuits and systems*, vol. 35, no. 10, pp. 1257–1272, 1988.

[13] ——, "Cellular neural networks: Applications," *IEEE Transactions on circuits and systems*, vol. 35, no. 10, pp. 1273–1290, 1988.

[14] L. O. Chua and T. Roska, *Cellular neural networks and visual computing: foundations and applications*. Cambridge University Press, 2002.

[15] S. Xavier-de Souza, M. Van Dyck, J. A. Suykens, and J. Vandewalle, "Fast an robust face tracking for cnn chips: application to wheelchair driving," in *Cellular Neural Networks and Their Applications, 2006. CNNA'06. 10th International Workshop on*. IEEE, 2006, pp. 1–6.

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

[17] R. Kötter, *Neuroscience databases: a practical guide*. Springer Science & Business Media, 2012.

[18] G. Urbain, J. Degrave, B. Carette, J. Dambre, and F. Wyffels, "Morphological properties of mass–spring networks for optimal locomotion learning," *Frontiers in neurorobotics*, vol. 11, p. 16, 2017.

[19] P. Panda and K. Roy, "Learning to generate sequences with combination of hebbian and non-hebbian plasticity in recurrent spiking neural networks," *Frontiers in neuroscience*, vol. 11, p. 693, 2017.

[20] D. Nikolić, S. Häusler, W. Singer, and W. Maass, "Distributed fading memory for stimulus properties in the primary visual cortex," *PLoS biology*, vol. 7, no. 12, p. e1000260, 2009.

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

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

[23] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification," *Frontiers in neuroscience*, vol. 12, 2018.

[24] E. Goodman and D. Ventura, "Spatiotemporal pattern recognition via liquid state machines," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on*. IEEE, 2006, pp. 3848–3853.

[25] W. Zhang and P. Li, "Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks," *Frontiers in Neuroscience*, vol. 13, p. 31, 2019.

[26] D. Verstraeten, B. Schrauwen, and D. Stroobandt, "Isolated word recognition using a liquid state machine." in *ESANN*, vol. 5. Citeseer, 2005, pp. 435–440.

[27] B. J. Grzyb, E. Chinellato, G. M. Wojcik, and W. A. Kaminski, "Facial expression recognition based on liquid state machines built of alternative neuron models," 2009.

[28] Q. Wang and P. Li, "D-lsm: Deep liquid state machine with unsupervised recurrent reservoir tuning," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, pp. 2652–2657.

[29] G. Srinivasan, P. Panda, and K. Roy, "Spilinc: Spiking liquid-ensemble computing for unsupervised speech and image recognition," *Frontiers in neuroscience*, vol. 12, 2018.

[30] C. A. Anastassiou, R. Perin, H. Markram, and C. Koch, "Ephaptic coupling of cortical neurons," *Nature neuroscience*, vol. 14, no. 2, p. 217, 2011.

[31] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and stdp integrated circuits," *IEEE transactions on biomedical circuits and systems*, vol. 6, no. 3, pp. 246–256, 2012.

[32] B. Jones, D. Stekel, J. Rowe, and C. Fernando, "Is there a liquid state machine in the bacterium escherichia coli?" in *Artificial Life, 2007. ALIFE'07. IEEE Symposium on*. Ieee, 2007, pp. 187–191.

[33] Y. Sakuraba *et al.*, "Co-concentration dependence of half-metallic properties in co–mn–si epitaxial films," *Applied Physics Letters*, vol. 96, no. 9, p. 092511, 2010.

[34] S. Yuasa and D. Djayaprawira, "Giant tunnel magnetoresistance in magnetic tunnel junctions with a crystalline mgo (0 0 1) barrier," *Journal of Physics D: Applied Physics*, vol. 40, no. 21, p. R337, 2007.

[35] S. Manipatruni, D. E. Nikonov, and I. A. Young, "Energy-delay performance of giant spin hall effect switching for dense magnetic memory," *Applied Physics Express*, vol. 7, no. 10, p. 103001, 2014.

[36] P. Wijesinghe, C. M. Liyanagedera, A. Jaiswal, and K. Roy, "Image segmentation with stochastic magnetic tunnel junctions and spiking neurons," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 2460–2468.

[37] A. Jaiswal, X. Fong, and K. Roy, "Comprehensive scaling analysis of current induced switching in magnetic memories based on in-plane and perpendicular anisotropies," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 120–133, 2016.

[38] J. C. Slonczewski, "Conductance and exchange coupling of two ferromagnets separated by a tunneling barrier," *Physical Review B*, vol. 39, no. 10, p. 6995, 1989.

[39] C.-F. Pai *et al.*, "Spin transfer torque devices utilizing the giant spin hall effect of tungsten," *Applied Physics Letters*, vol. 101, no. 12, p. 122404, 2012.

[40] S. Ikeda *et al.*, "A perpendicular-anisotropy cofeb–mgo magnetic tunnel junction," *Nature materials*, vol. 9, no. 9, pp. 721–724, 2010.

[41] M. Zidan, H. Omran, R. Naous, A. Sultan, H. Fahmy, W. Lu, and K. N. Salama, "Single-readout high-density memristor crossbar," *Scientific reports*, vol. 6, p. 18863, 2016.

[42] A. K. Maan, D. A. Jayadevi, and A. P. James, "A survey of memristive threshold logic circuits," *IEEE transactions on neural networks and learning systems*, 2016.

[43] Y. V. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 8, pp. 1857–1864, 2010.

[44] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, M. Hu, N. Ge, J. P. Strachan, Z. Li *et al.*, "Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing," *Nature materials*, vol. 16, no. 1, pp. 101–108, 2017.

[45] T. Ohno, T. Hasegawa, T. Tsuruoka, K. Terabe, J. K. Gimzewski, and M. Aono, "Short-term plasticity and long-term potentiation mimicked in single inorganic synapses," *Nature materials*, vol. 10, no. 8, pp. 591–595, 2011.

[46] I. Valov, R. Waser, J. R. Jameson, and M. N. Kozicki, "Electrochemical metallization memories—fundamentals, applications, prospects," *Nanotechnology*, vol. 22, no. 25, p. 254003, 2011.

[47] Y. Yang, P. Gao, S. Gaba, T. Chang, X. Pan, and W. Lu, "Observation of conducting filament growth in nanoscale resistive memories," *Nature communications*, vol. 3, p. ncomms1737, 2012.

[48] H. Lee, Y. Chen, P. Chen, P. Gu, Y. Hsu, S. Wang, W. Liu, C. Tsai, S. Sheu, P. Chiang *et al.*, "Evidence and solution of over-reset problem for hfo x based resistive memory with sub-ns switching speed and high endurance," in *Electron Devices Meeting (IEDM), 2010 IEEE International*. IEEE, 2010, pp. 19–7.

[49] H. Jiang, D. Belkin, S. E. Savel'ev, S. Lin, Z. Wang, Y. Li, S. Joshi, R. Midya, C. Li, M. Rao *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nature Communications*, vol. 8, no. 1, p. 882, 2017.

[50] S. H. Jo, K.-H. Kim, T. Chang, S. Gaba, and W. Lu, "Si memristive devices applied to memory and neuromorphic circuits," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 13–16.

[51] S. H. Jo, K.-H. Kim, and W. Lu, "Programmable resistance switching in nanoscale two-terminal devices," *Nano letters*, vol. 9, no. 1, pp. 496–500, 2008.

[52] P. Rabbani, R. Dehghani, and N. Shahpari, "A multilevel memristor–cmos memory cell as a reram," *Microelectronics Journal*, vol. 46, no. 12, pp. 1283–1290, 2015.

[53] G. Wang, Y. Yang, J.-H. Lee, V. Abramova, H. Fei, G. Ruan, E. L. Thomas, and J. M. Tour, "Nanoporous silicon oxide memory," *Nano letters*, vol. 14, no. 8, pp. 4694–4699, 2014.

[54] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.

[55] F. S. Bao *et al.*, "Accelerating boolean satisfiability (sat) solving by common subclause elimination," *Artificial Intelligence Review*, pp. 1–15, 2017.

[56] S. Saha *et al.*, "Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 577–596.

[57] Y. Vizel, G. Weissenbacher, and S. Malik, "Boolean satisfiability solvers and their applications in model checking," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2021–2035, 2015.

[58] J. Stanley, H. Liao, and S. Lafortune, "Sat-based control of concurrent software for deadlock avoidance," *IEEE Transactions on Automatic Control*, vol. 60, no. 12, pp. 3269–3274, 2015.

[59] Y.-T. Chung and J.-H. Jiang, "Functional timing analysis method for circuit timing verification," Mar. 11 2014, uS Patent 8,671,375.

[60] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[61] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*. ACM, 1971, pp. 151–158.

[62] M. Ercsey-Ravasz and Z. Toroczkai, "Optimization hardness as transient chaos in an analog approach to constraint satisfaction," *Nature Physics*, vol. 7, pp. 966–970, 2011.

[63] M. W. Moskewicz *et al.*, "Chaff: Engineering an efficient sat solver," in *Proceedings of the 38th annual Design Automation Conference*. ACM, 2001, pp. 530–535.

[64] J. P. Marques-Silva and K. A. Sakallah, "Grasp: A search algorithm for propositional satisfiability," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 506–521, 1999.

[65] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane, "An improved exponential-time algorithm for k-sat," *Journal of the ACM (JACM)*, vol. 52, no. 3, pp. 337–364, 2005.

[66] L. Guo, Y. Hamadi, S. Jabbour, and L. Sais, "Diversification and intensification in parallel sat solving," *Principles and Practice of Constraint Programming–CP 2010*, pp. 252–265, 2010.

[67] H. T. Siegelmann, "Computation beyond the turing limit," in *Neural Networks and Analog Computation*. Springer, 1999, pp. 153–164.

[68] D. A. Basford *et al.*, "The impact of analog computational error on an analog boolean satisfiability solver," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 2503–2506.

[69] X. Yin, B. Sedighi, M. Varga, M. Ercsey-Ravasz, Z. Toroczkai, and X. S. Hu, "Efficient analog circuits for boolean satisfiability," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 1, pp. 155–167, 2018.

[70] S. Kirkpatrick, B. Selman *et al.*, "Critical behavior in the satisfiability of random boolean expressions," *Science-AAAS-Weekly Paper Edition-including Guide to Scientific Information*, vol. 264, no. 5163, pp. 1297–1300, 1994.

[71] F. Krzakała, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborová, "Gibbs states and the set of solutions of random constraint satisfaction problems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 25, pp. 10 318–10 323, 2007.

[72] F. Krzakała and L. Zdeborová, "Phase transitions and computational difficulty in random constraint satisfaction problems," in *Journal of Physics: Conference Series*, vol. 95, no. 1. IOP Publishing, 2008, p. 012012.

[73] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 9, pp. 1907–1919, 2016.

[74] A. Sengupta *et al.*, "Magnetic tunnel junction mimics stochastic cortical spiking neurons," *Scientific reports*, vol. 6, p. 30039, 2016.

[75] A. Sengupta, Y. Shim, and K. Roy, "Proposal for an all-spin artificial neural network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets," *IEEE transactions on biomedical circuits and systems*, vol. 10, no. 6, pp. 1152–1160, 2016.

[76] B. Behin-Aein, D. Datta, S. Salahuddin, and S. Datta, "Proposal for an all-spin logic device with built-in memory," *Nature nanotechnology*, vol. 5, no. 4, pp. 266–270, 2010.

[77] G. Yu *et al.*, "Switching of perpendicular magnetization by spin-orbit torques in the absence of external magnetic fields," *Nature nanotechnology*, vol. 9, no. 7, pp. 548–554, 2014.

[78] L. Liu, C.-F. Pai, Y. Li, H. Tseng, D. Ralph, and R. Buhrman, "Spin-torque switching with the giant spin hall effect of tantalum," *Science*, vol. 336, no. 6081, pp. 555–558, 2012.

[79] C. M. Liyanagedera, A. Sengupta, A. Jaiswal, and K. Roy, "Stochastic spiking neural networks enabled by magnetic tunnel junctions: From nontelegraphic to telegraphic switching regimes," *Physical Review Applied*, vol. 8, no. 6, p. 064017, 2017.

[80] X. Fong *et al.*, "Knack: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque mram bit-cells," in *Simulation of Semiconductor Processes and Devices (SISPAD), 2011 International Conference on*. IEEE, 2011, pp. 51–54.

[81] Z. Wang, G. Yu, X. Liu, B. Zhang, X. Chen, and W. Lu, "Magnetization characteristic of ferromagnetic thin strip by measuring anisotropic magnetoresistance and ferromagnetic resonance," *Solid State Communications*, vol. 182, pp. 10–13, 2014.

[82] A. Jaiswal and K. Roy, "Mesl: Proposal for a non-volatile cascadable magneto-electric spin logic," *Scientific reports*, vol. 7, 2017.

[83] A. Aharoni, "Demagnetizing factors for rectangular ferromagnetic prisms," *Journal of applied physics*, vol. 83, no. 6, pp. 3432–3434, 1998.

[84] Z. Jonke, S. Habenschuss, and W. Maass, "Solving constraint satisfaction problems with networks of spiking neurons," *Frontiers in neuroscience*, vol. 10, 2016.

[85] W. F. Brown Jr, "Thermal fluctuations of a single-domain particle," *Physical Review*, vol. 130, no. 5, p. 1677, 1963.

[86] Y. Huai, "Spin-transfer torque mram (stt-mram): Challenges and prospects," *AAPPS bulletin*, vol. 18, no. 6, pp. 33–40, 2008.

[87] K. Song and K.-J. Lee, "Spin-transfer-torque efficiency enhanced by edge-damage of perpendicular magnetic random access memories," *Journal of Applied Physics*, vol. 118, no. 5, p. 053912, 2015.

[88] Z. Sun, S. Retterer, and D. Li, "The influence of ion-milling damage to magnetic properties of co80pt20 patterned perpendicular media," *Journal of Physics D: Applied Physics*, vol. 47, no. 10, p. 105001, 2014.

[89] H. H. Hoos and T. Stützle, "Satlib: An online resource for research on sat," *Sat*, vol. 2000, pp. 283–292, 2000.

[90] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[91] K. Gulati and S. P. Khatri, "Accelerating boolean satisfiability on a custom ic," in *Hardware Acceleration of EDA Algorithms*. Springer, 2010, pp. 33–61.

[92] K. Gulati, S. Paul, S. P. Khatri, S. Patil, and A. Jas, "Fpga-based hardware acceleration for boolean satisfiability," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 2, p. 33, 2009.

[93] J. Thong and N. Nicolici, "Fpga acceleration of enhanced boolean constraint propagation for sat solvers," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2013, pp. 234–241.

[94] J. D. Davis, Z. Tan, F. Yu, and L. Zhang, "A practical reconfigurable hardware accelerator for boolean satisfiability solvers," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*. IEEE, 2008, pp. 780–785.

[95] N. Eén and N. Sörensson, "Minisat 2.2," *ht tp://minisat. se*, 2013.

[96] P. Wijesinghe, C. Liyanagedera, and K. Roy, "Analog approach to constraint satisfaction enabled by spin orbit torque magnetic tunnel junctions," *Scientific reports*, vol. 8, no. 1, p. 6940, 2018.

[97] M. M. Waldrop, "Brain in a box," *Nature*, vol. 482, no. 7386, p. 456, 2012.

[98] G.-q. Bi and M.-m. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.

[99] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, 2015.

[100] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Stdp-based spiking deep neural networks for object recognition," *arXiv preprint arXiv:1611.01421*, 2016.

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

[102] E. Hunsberger and C. Eliasmith, "Spiking deep networks with lif neurons," *arXiv preprint arXiv:1510.08829*, 2015.

[103] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[104] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, no. 13, pp. 5872–5878, 2013.

[105] W. Yi, F. Perner, M. S. Qureshi, H. Abdalla, M. D. Pickett, J. J. Yang, M.-X. M. Zhang, G. Medeiros-Ribeiro, and R. S. Williams, "Feedback write scheme for memristive switching devices," *Applied Physics A: Materials Science & Processing*, vol. 102, no. 4, pp. 973–982, 2011.

[106] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, 2011.

[107] M. Hu, Y. Wang, W. Wen, Y. Wang, and H. Li, "Leveraging stochastic memristor devices in neuromorphic hardware systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 235–246, 2016.

[108] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nature nanotechnology*, vol. 11, no. 8, pp. 693–699, 2016.

[109] M. Al-Shedivat, R. Naous, G. Cauwenberghs, and K. N. Salama, "Memristors empower spiking neurons with stochasticity," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 242–253, 2015.

[110] G. Palma, M. Suri, D. Querlioz, E. Vianello, and B. De Salvo, "Stochastic neuron design using conductive bridge ram," in *Proceedings of the 2013 IEEE/ACM International Symposium on Nanoscale Architectures*. IEEE Press, 2013, pp. 95–100.

[111] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[112] A. Sengupta, M. Parsa, B. Han, and K. Roy, "Probabilistic deep spiking neural systems enabled by magnetic tunnel junction," *IEEE Transactions on Electron Devices*, vol. 63, no. 7, pp. 2963–2970, 2016.

[113] B. Rajendran, Y. Liu, J.-s. Seo, K. Gopalakrishnan, L. Chang, D. J. Friedman, and M. B. Ritter, "Specifications of nanoscale devices and circuits for neuromorphic computational systems," *IEEE Transactions on Electron Devices*, vol. 60, no. 1, pp. 246–253, 2013.

[114] C. M. Liyanagedera, A. Sengupta, A. Jaiswal, and K. Roy, "Stochastic spiking neural networks enabled by magnetic tunnel junctions: From nontelegraphic to telegraphic switching regimes," *Physical Review Applied*, vol. 8, no. 6, p. 064017, 2017.

[115] Z. Biolek, D. Biolek, and V. Biolkova, "Spice model of memristor with nonlinear dopant drift." *Radioengineering*, vol. 18, no. 2, 2009.

[116] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 541–552.

[117] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[118] A. Ankit, A. Sengupta, P. Panda, and K. Roy, "Resparc: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 27.

[119] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," *Technical University of Denmark*, vol. 5, 2012.

[120] S. Yu, P.-Y. Chen, Y. Cao, L. Xia, Y. Wang, and H. Wu, "Scaling-up resistive synaptic arrays for neuro-inspired architecture: Challenges and prospect," in *Electron Devices Meeting (IEDM), 2015 IEEE International*. IEEE, 2015, pp. 17–3.

[121] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE Transactions on Nanotechnology*, vol. 12, no. 3, pp. 288–295, 2013.

[122] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using rram synaptic devices with sub-pj energy and tolerance to variability: Experimental characterization and large-scale modeling," in *Electron Devices Meeting (IEDM), 2012 IEEE International*. IEEE, 2012, pp. 10–4.

[123] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[124] S. Lim, J.-H. Bae, J.-H. Eum, S. Lee, C.-H. Kim, B.-G. Park, and J.-H. Lee, "Adaptive learning rule for hardware-based deep neural networks using electronic synapse devices," *arXiv preprint arXiv:1707.06381*, 2017.

[125] B. Han, A. Ankit, A. Sengupta, and K. Roy, "Cross-layer design exploration for energy-quality tradeoffs in spiking and non-spiking deep artificial neural networks," *IEEE Transactions on Multi-Scale Computing Systems*, 2017.

[126] P. Wijesinghe, C. M. Liyanagedera, and K. Roy, "Fast, low power evaluation of elementary functions using radial basis function networks," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 208–213.

[127] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: bioinspired cameras with spiking output," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014.

[128] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, p. 52, 2018.

[129] D. L. Lewis and H.-H. S. Lee, "Architectural evaluation of 3d stacked rram caches," in *3D System Integration, 2009. 3DIC 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–4.

[130] P. Wijesinghe, A. Ankit, A. Sengupta, and K. Roy, "An all-memristor deep spiking neural computing system: A step toward realizing the low-power stochastic brain," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 5, pp. 345–358, 2018.

[131] Z. Xie, "Neural text generation: A practical guide," *arXiv preprint arXiv:1711.09534*, 2017.

[132] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[133] L. Yaniv and M. Yossi, "Google duplex: An ai system for accomplishing real-world tasks over the phone," 2018.

[134] F. Xue, H. Guan, and X. Li, "Improving liquid state machine with hybrid plasticity," in *Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016 IEEE*. IEEE, 2016, pp. 1955–1959.

[135] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[136] K. Fukunaga and J. Mantock, "Nonparametric discriminant analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 671–678, 1983.

[137] W. Yao, Z. Zeng, C. Lian, and H. Tang, "Ensembles of echo state networks for time series prediction," in *Advanced Computational Intelligence (ICACI), 2013 Sixth International Conference on*. IEEE, 2013, pp. 299–304.

[138] H. Jaeger, "Echo state network," *Scholarpedia*, vol. 2, no. 9, p. 2330, 2007.

[139] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain research bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999.

[140] M. Wehr and A. M. Zador, "Balanced inhibition underlies tuning and sharpens spike timing in auditory cortex," *Nature*, vol. 426, no. 6965, p. 442, 2003.

[141] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.

[142] C. Toledo-Suárez, R. Duarte, and A. Morrison, "Liquid computing on and off the edge of chaos with a striatal microcircuit," *Frontiers in computational neuroscience*, vol. 8, p. 130, 2014.

[143] W. Maass, T. Natschläger, and H. Markram, "A model for real-time computation in generic neural microcircuits," in *Advances in neural information processing systems*, 2003, pp. 229–236.

[144] J. Kaiser, R. Stal, A. Subramoney, A. Roennau, and R. Dillmann, "Scaling up liquid state machines to predict over address events from dynamic vision sensors," *Bioinspiration & biomimetics*, vol. 12, no. 5, p. 055001, 2017.

[145] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[146] W. Maass, T. Natschläger, and H. Markram, "Computational models for generic cortical microcircuits," *Computational neuroscience: A comprehensive approach*, vol. 18, p. 575, 2004.

[147] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[148] Y. Shim, A. Philippides, K. Staras, and P. Husbands, "Unsupervised learning in an ensemble of spiking neural networks mediated by itdp," *PLoS computational biology*, vol. 12, no. 10, p. e1005137, 2016.

[149] W. Maass, R. A. Legenstein, and N. Bertschinger, "Methods for estimating the computational power and generalization capability of neural microcircuits," in *Advances in neural information processing systems*, 2005, pp. 865–872.

[150] S. Roy and A. Basu, "An online structural plasticity rule for generating better reservoirs," *Neural computation*, vol. 28, no. 11, pp. 2557–2584, 2016.

[151] E. Hourdakis and P. Trahanias, "Use of the separation property to derive liquid state machines with enhanced classification performance," *Neurocomputing*, vol. 107, pp. 40–48, 2013.

[152] K. I. Park and Park, *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer, 2018.

[153] S. Ji and J. Ye, "Generalized linear discriminant analysis: a unified framework and efficient model selection," *IEEE Transactions on Neural Networks*, vol. 19, no. 10, pp. 1768–1782, 2008.

[154] D. F. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in python," *Frontiers in neuroinformatics*, vol. 2, p. 5, 2008.

[155] H. Robbins and S. Monro, "A stochastic approximation method," in *Herbert Robbins Selected Papers*. Springer, 1985, pp. 102–109.

[156] S. Mei, A. Montanari, and P.-M. Nguyen, "A mean field view of the landscape of two-layers neural networks," *arXiv preprint arXiv:1804.06561*, 2018.

[157] M. Liberman, R. Amsler, K. Church, E. Fox, C. Hafner, J. Klavans, M. Marcus, B. Mercer, J. Pedersen, P. Roossin *et al.*, "Ti 46-word," *Philadelphia (Pennsylvania): Linguistic Data Consortium*, 1993.

[158] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[159] R. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, vol. 7. IEEE, 1982, pp. 1282–1285.

[160] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, p. 1998, 1998.

[161] S. Roy and A. Basu, "An online unsupervised structural plasticity algorithm for spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 4, pp. 900–910, 2017.

[162] B. Kasap and A. J. van Opstal, "Dynamic parallelism for synaptic updating in gpu-accelerated spiking neural network simulations," *Neurocomputing*, vol. 302, pp. 55–65, 2018.

[163] Q. Wang, Y. Li, B. Shao, S. Dey, and P. Li, "Energy efficient parallel neuromorphic architectures with approximate arithmetic on fpga," *Neurocomputing*, vol. 221, pp. 146–158, 2017.

[164] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, "Reservoir computing using dynamic memristors for temporal information processing," *Nature communications*, vol. 8, no. 1, p. 2204, 2017.

[165] Q. Wang, Y. Kim, and P. Li, "Architectural design exploration for neuromorphic processors with memristive synapses," in *14th IEEE International Conference on Nanotechnology*. IEEE, 2014, pp. 962–966.

[166] M. Herlihy and S. Nir, *The art of multiprocessor programming*. Morgan Kaufmann, 2011.

[167] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," *arXiv preprint arXiv:1803.09574*, 2018.

[168] Q. Wang, Y. Jin, and P. Li, "General-purpose lsm learning processor architecture and theoretically guided design space exploration," in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2015, pp. 1–4.

[169] M. Krzywinski and N. Altman, "Points of significance: Multiple linear regression," 2015.

[170] D. Norton and D. Ventura, "Preparing more effective liquid state machines using hebbian learning," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on*. IEEE, 2006, pp. 4243–4248.

[171] H. Markram, Y. Wang, and M. Tsodyks, "Differential signaling via the same axon of neocortical pyramidal neurons," *Proceedings of the National Academy of Sciences*, vol. 95, no. 9, pp. 5323–5328, 1998.

[172] W. Maass, R. Legenstein, and H. Markram, "A new approach towards vision suggested by biologically realistic neural microcircuit models," in *International Workshop on Biologically Motivated Computer Vision*. Springer, 2002, pp. 282–293.

[173] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

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

VITA

VITA

Parami Wijesinghe received her BSc in Electrical and Electronics Engineering from University of Peradeniya, Sri Lanka, with the gold medal for the best performance in engineering and the prize for the best performance in engineering mathematics. She represented Sri Lanka in the 9^{th} Asian Physics Olympiad competition (Mongolia), and 39^{th} International Physics Olympiad competition (Vietnam). She is currently pursuing her Direct PhD in Electrical and Computer Engineering at Purdue University, West Lafayette, IN, USA, under the guidance of Prof. Kaushik Roy. She was an intern at Intel, Hillsboro, OR, USA in 2016, and an interim engineering intern at Qualcomm, SanDiego, CA, USA in 2017. Parami Wijesinghe is the recipient of Ross Fellowship award from Purdue University. Her major research interests include neuromorphic computing, emerging devices, and memory design.