

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Application Of Memristive Device Arrays For Pattern Recognition

Permalink

<https://escholarship.org/uc/item/6kr3g9nz>

Author

Shao, Yinghao

Publication Date

2020

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-NonCommercial-NoDerivatives License, available at <https://creativecommons.org/licenses/by-nc-nd/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

SANTA CRUZ

**APPLICATION OF MEMRISTIVE DEVICE ARRAYS FOR PATTERN
RECOGNITION**

A thesis submitted in partial satisfaction

of the requirements for the degree of

MASTER OF SCIENCE

In

ELECTRICAL AND COMPUTER ENGINEERING

by

Yinghao Shao

June 2020

The Thesis of Yinghao Shao
is approved:

Professor Sung-Mo Kang, Chair

Professor Shiva Abbaszadeh

Professor Yu Zhang

Quentin Williams
Acting Vice Provost and Dean of Graduate Studies

Copyright © by
Yinghao Shao
2020

Table of Contents

1. Introduction	1
1.1 Motivation	1
1.2 Working environment	2
1.3 Outline of the thesis	2
2. Machine learning	4
2.1 What is machine learning	4
2.2 Machine learning types and strength	6
2.3 Deep learning and artificial neural network (ANN)	7
3 Pattern recognition using ANN	9
3.1 MNIST as training and testing data sets	9
3.2 A fully connected ANN for pattern recognition	11
3.3 Neural network training algorithm	13
4 ANN using memristor device IFG (Ionic floating gate memory)	15
4.1 Memristor, the missing component	15
4.2 IFG as a memristor	16
4.3 Features of IFG Device	18
4.4 IFG memory as ANN synapses	20
4.5 Fully connected ANN using IFG	22

5 Cadence model and simulation	25
5.1 Cadence model for IFG memory	25
5.2 Network building and weight setting	27
5.3 Results and comparison	30
6 Conclusion	34
7 Future work	35
Appendix	38
1. VerilogA model for IFG	38
2. Cadence text-based simulation codes	40
2.1 Python code to generate text-based "Spectre" simulation	40
2.2 Python code to apply the activation function to the signals	43
3. Recognition results of original and reduced sized ANN	43
References	48

List of Figures

Figure 1. How modern emails get filtered.	5
Figure 2. Types of artificial intelligence models, The ANN is the core part of deep learning. Deep learning is a member of the “Supervised learning” and “Machine learning” group. [5].....	6
Figure 3. Artificial neural network compared with the biological neural network. They both can work for pattern recognition [5]......	8
Figure 4. MNIST database samples [11].	9
Figure 5. MNIST number “7”.....	10
Figure 6. Averaging every 16 pixels to one and cropping the MNIST picture. The pixel numbers are randomly filled for better demonstration.	10
Figure 7. A fully connected ANN for pattern recognition in which every input layer neuron I_i is connected through their corresponding weight W_{ij} to X_j . The summation is run through a nonlinear function $f()$, then fed to next layer neuron X_j [12].	12
Figure 8. The backpropagation of the error term δ_j , each weight of the network is adjusted by the gradient descent method. The error can be reduced through training [12]......	14
Figure 9. (a). The resistor defines the relation between Voltage and Current, the capacitor defines the relation between Voltage and Charge, the inductor defines the relation between Current and Flux. (b). the symbol of a memristor.....	15
Figure 10. A polymer-based redox transistor [3]......	17
Figure 11. IFG memory structure with ‘read’ and ‘write’ process [2].	17

Figure 12. Every weight of our ANN is one IFG memory in the 'read' state. A resistor Rlimit is needed to read out the voltage Vout in the Cadence simulator.....	21
Figure 13. "Read" mode IFG as the synapses.....	22
Figure 14. The reduced ANN using 7×7 pixels MNIST data set for training and testing.....	23
Figure 15. ANN using original 28×28 MNIST data set for training and testing.....	24
Figure 16. (a). The equivalent circuit of the IFG device. (b). The Cadence circuit model of IFG, Vw is the "write" voltage applied to the "Gate" node. Vr is the "Read" voltage applied to drain. Inside the instance of IFG is the VerilogA code (attached in Appendix)	25
Figure 17. Weight writing process of the IFG memory. The yellow line is the conductance of the IFG memory. The red line is the "write" voltage applied to the "Gate", the yellow line is the source voltage and the white line is the conductance of the IFG memory.....	26
Figure 18. Normalization of the MNIST pictures. Every greyscale number is divided by 255 and normalized.....	27
Figure 19. Circuit demonstration of the connections between the 49 input layer neurons and one hidden layer neuron N9.	28
Figure 20. Circuit demonstration of the connections between the 10 hidden layer neurons and one output layer neuron "1".	29
Figure 21. An ANN recognition result of number "0" from our reduced network. ...	31
Figure 22. An ANN recognition result of number "7" from our reduced network. ...	31

Figure 23. An ANN recognition result of our full-size network.....	32
Figure 24. Performance for reduced ANN with 2,000 test samples.	33
Figure 25. Performance for reduced ANN with 1800 test samples.	33
Figure 26. ReLU function.....	36
Figure 27. Structure comparisons between (a). fully connected ANN and (b). Convolutional neural networks CNN. The CNN weights can also be implemented with IFG devices.	37
Table of Illustrations	
Table 1. Recognition results from original sized ANN	44
Table 2. Recognition results from reduced sized ANN	46

Abstract

Application Of Memristive Device Arrays For Pattern Recognition

by

Yinghao Shao

Artificial intelligence (AI) technology like deep learning is powering our daily life in many areas such as pattern recognition. The artificial neural network (ANN) is one of the deep learning models to achieve pattern recognition. A well-trained ANN can recognize images with precision over 98%. Although traditional two terminals memristors like phase changing memory (PCM) are already used to build ANNs. But those devices typically suffer from nonlinear, asymmetric conductance tuning problems. The novel memristive device Ionic Floating Gate memory (IFG) could potentially solve those problems. In this paper, a compact Cadence model of fully connected ANN using IFG is presented. The devices are tuned to an optimized state and formed a well-trained network. The recognition accuracy reaches 93.8%. This work demonstrates the IFG device also has the potential to be further utilized into other deep neural networks as synaptic memory.

Acknowledgment

Many people helped me out throughout this thesis project. Here I express my gratitude to them:

Professor Sung-Mo Kang, my thesis advisor, who is so patient, knowledgeable. He guided and instructed my project whenever I encountered problems.

Donguk Choi, my senior colleague, gave me so much help on Cadence tools and programming the device array. He also kindly shared his experience of his career which is a big inspiration and guidance for me.

Xiaoyang Jia, my colleague, has collaborated on providing optimized weights for the arrays.

Mr. Doug Orr has kindly helped me with the neural network working structure and how the network works for pattern recognition.

Dr. Alec Talin and his group in Sandia National Laboratory provided information on IFG devices.

Above all, I would like to thank my parents. They have supported me throughout my academic pursuit. Without their help, I would not have been able to complete this project.

1. Introduction

Machine learning technology is powering our daily life in many areas, from social communication recommendations to a better filter for the search engine, even to better deliver commercial advertisements customized to the users' habits and interests. Nowadays, machine learning is also applied for image pattern recognition, speech recognition, and language translations [1]. Deep learning is the technique to tackle those problems. And among several deep learning techniques, the artificial neural network (ANN) has been used to achieve the best efficiency and accuracy. Although ANNs using two terminals memristor like the phase changing memory (PCM) and the filament-forming metal oxides (FFMO) are already demonstrated. Those devices typically suffer from nonlinear, asymmetric conductance tuning problems. Their scalability is also very low. The novel memristive device Ionic Floating Gate memory (IFG) could potentially solve those problems.

In this work, we use the Cadence VerilogA model of the memristive device Ionic floating gate (IFG) memory [2] which combined a redox transistor [3] and a volatile conductive bridge memory to make a non-volatile synaptic memory. We design a compact Cadence ANN model for numeric pattern recognition.

1.1 Motivation

Artificial intelligence and deep learning have improved our lives in so many areas, from science to business. Among those models, the artificial neural network (ANN) is the most important one. The idea of ANN is inspired by our human brains. It has

shown great potential in image processing and pattern classification. The image recognition using ANN commonly has a very high accuracy of over 90% [4]. Its strength in extracting image features could also be used in enhancing or classifying blurry pictures [5]. The new device IFG has linear and symmetrical conductance tuning properties that could be implemented to ANN. We built our compact Cadence ANN model with IFG devices. We applied pattern recognition tasks to our ANN and successfully recognized the numbers in MNIST data sets.

1.2 Working environment

This thesis work was done under the supervision of Professor Sung-Mo Kang in collaboration with Donguk Choi and Xiaoyang Jia.

All the simulation is run on Windows PC, we used Xming and Putty to connect the UCSC Linux server, and we used the University licensed version of Cadence simulation tools.

1.3 Outline of the thesis

This thesis work started with an introduction to the research project described in chapter 1. Chapter 2 gives an introduction to AI and machine learning. Chapter 3 introduces the ANN architecture and the back-propagation training method. We describe how we can apply the back-propagation method to ANN and get optimized weights. A brief introduction to the MNIST training data is included. Chapter 4 shows the introduction of the memristor. We described the IFG memory structure and its working principles. The ANN structure of our design is also introduced. Chapter 5

shows our IFG Cadence model and ANN recognition results. Chapter 6 provides a brief conclusion of our work. References are provided in the end. All the Cadence IFG VerilogA model and simulation Python code could be found in the Appendix.

2. Machine learning

2.1 What is machine learning

In 1959, Arthur Samuel gave us a general definition of machine learning:

“Machine learning is the field of study that gives the computer the ability to learn without being explicitly programmed”.

Tom Mitchell gave an engineering explanation in 1997.

"A computer program is said to learn from experience E concerning some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E"

The email spam filter is a commonly used machine learning program. It is a well-organized program that can use examples to identify spam emails. The examples include both spam emails (Flagged by the users) and regular emails (Emails that are not flagged as spam) from your email box. The set of examples the computer used to learn is called "training set", each small and simple example inside are called "training instance". In the case of spam emails flagging, according to Tom Mitchell's definition, Task T is flagging spam emails. The experience E is the training set (all the emails you received, both spam and regular emails). The performance measure P is yours to define, like how accurate the program is while doing the spam email identification.

Simply storing the regular emails, and flagged spam emails separately doesn't make the computer smarter. It only means the computer stores two sets of data. One set is the regular emails, the other set is the spam emails. However, the learning program will automatically identify the emails.

When exam the spam emails, we may realize that some specific words are frequently used, such as "credentials satisfied, pre-approved, qualified, etc." Those words always show up in credit card advertisements. For a traditional filter program, we can extract those phrases and write a program that can identify those phrases automatically.

However, if the advertisement company knew its specific advertisement phrase was blocked, it will modify its advertisement emails by changing those words to other similar words. This can lead to the revision of the filter program endlessly.

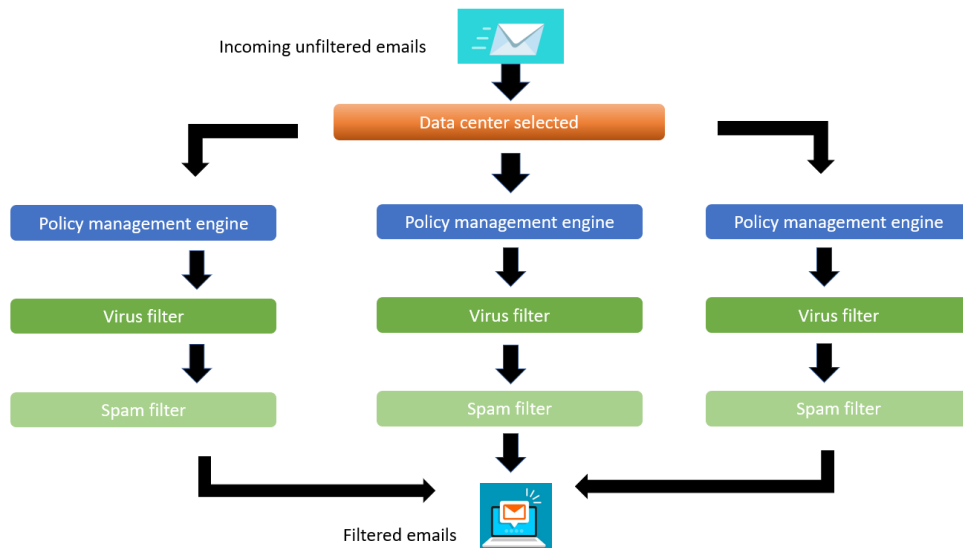


Figure 1. How modern emails get filtered.

To solve those problems, machine learning techniques automatically self-learn the new words that have a high probability to be spam emails. No matter how new phrases appear in spam emails, the program should detect them automatically by monitoring unusually frequent new words that show up among all the emails. And no human effort should be required for maintenance.

2.2 Machine learning types and strength

Nowadays, machine learning is a major part of the artificial intelligence technique, contains many methods. Both supervised learning and unsupervised learning. This work is focused on a supervised learning model for pattern recognition with an artificial neural network (ANN).

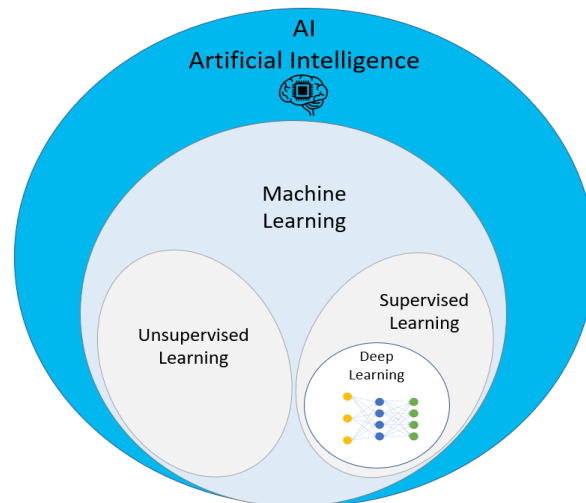


Figure 2. Types of artificial intelligence models, The ANN is the core part of deep learning. Deep learning is the most important member of the “Machine learning” and “Supervised learning” group. [5]

Briefly speaking, the machine learning is good at solving complex problems that are too difficult for traditional programs, data variable environment (Data changes over

time like the spam emails), extremely large scale database problems (image recognition, sound recognition), and optimization of the solutions that already exist [7].

2.3 Deep learning and artificial neural network (ANN)

ANN (Artificial neural network) is one of the machine learning models inspired by the human brain cortex. The neural network was first introduced back in 1943 by McCulloch and Walter Pitts [8]. Their paper showed a very simple computational logic model of how biological neurons works together. That was a primitive neural network model. Since then, the ANN experienced a 50 years' cold winter for several reasons: lack of fundings, computational power, good AI models, etc.

But the Internet explosion has brought us a huge amount of data, which can be used for training the network. The development of CPU and GPU has provided us with sufficient computational power. The algorithm for training has been improved dramatically over the past few decades, and enormous funding is going to AI now.

In 2006, a well-trained neural network that capable of recognizing numerical numbers with state of art precision (over 98%) was demonstrated by Geoffrey Hinton et al [4]. They called it "Deep learning". The deep neural network is also inspired by our brain cortex. It has several layers of artificial neurons. ANN uses neurons to represent the simple unit from the biological neural network. It uses hidden layers to mimic the whole interconnection layers between inputs and outputs.

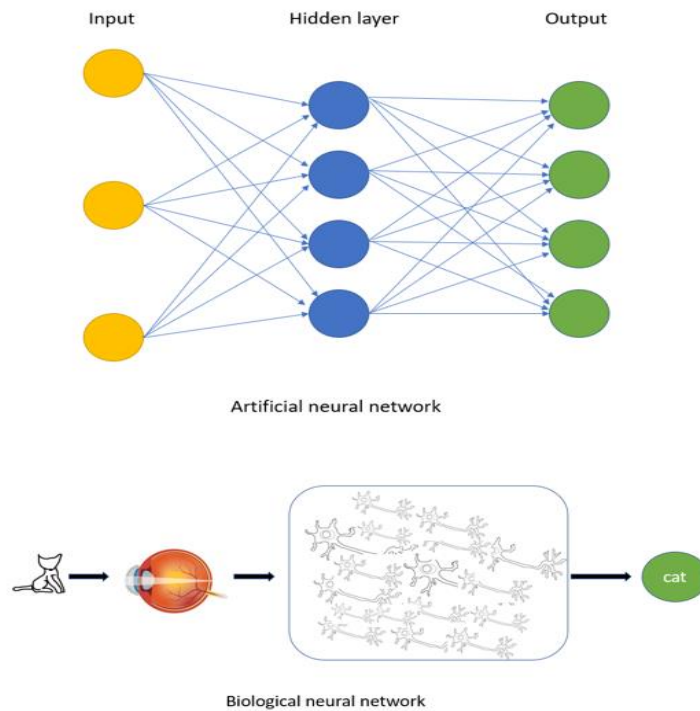


Figure 3. Artificial neural network compared with the biological neural network. They both can work for pattern recognition [5].

ANN has tremendous potential for extracting features from a large amount of raw data. Therefore, it has the potential to help solve the massive data capacity problems that cannot be handled by human or traditional computational algorithms. For example, discovering patterns from “all” human beings’ written data sets, simulate “every” potential drug molecule [8], predict the mutations of “every” DNA sequence and their affection on human diseases [9][10].

In this work, we focus on the number pattern recognition using ANN deep learning algorithms.

3 Pattern recognition using ANN

The pattern recognition we discuss in this work is based on a supervised deep learning method. It uses the artificial neural network (ANN) to automatically extract the features of the input images.

3.1 MNIST as training and testing data sets

To train the ANN to do pattern recognition, first, we collect a large number of patterns. We used the MNIST (Modified National Institute of Standard and Technology database) database [11] to train our Neural Network. It has 70,000 images of data, which contains 60,000 training images and 10,000 testing images. Every picture inside the database has been labeled with their correct digits accordingly. The MNIST database is a commonly used database for nearly every pattern classification algorithm.

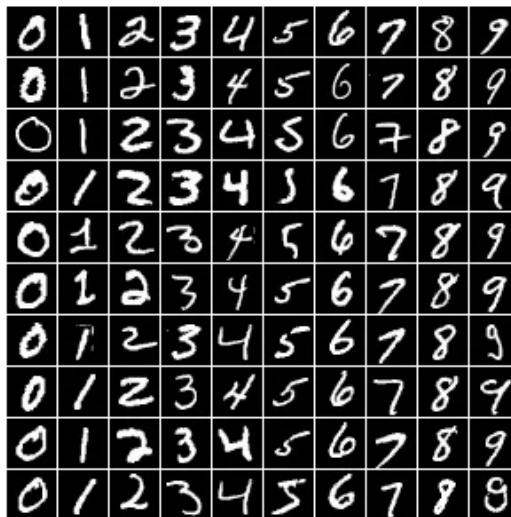


Figure 4. MNIST database samples [11].

Among the 70,000 images data, each one of them is 28 pixels wide and 28 pixels long. Therefore, every picture has 784 pixels. Every pixel's intensity is a greyscale number from 0 (white) to 255 (black). The following Fig 5 is a demonstration for the digit "7".

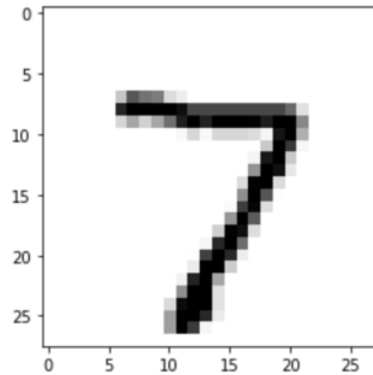


Figure 5. MNIST number "7".

We successfully trained our ANN using this data. To reduce the ANN size for easy implementation, we also cropped the 70000 pictures from 28×28 pixels to 7×7 pixels. We averaged every 4×4 pixels and cropped the pictures.

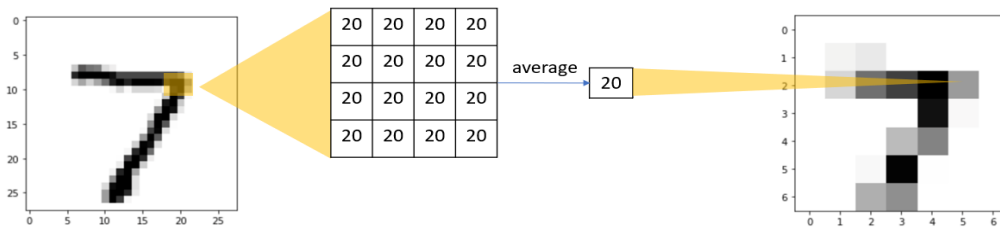


Figure 6. Averaging every 16 pixels to one and cropping the MNIST picture. The pixel numbers are randomly filled for better demonstration.

3.2 A fully connected ANN for pattern recognition

A typical ANN contains three kinds of layers: one input layer, multiple hidden layers, and one output layers. Between the two adjacent layers, every neuron from the first layer is connected through its original synapse (weight) to all the neurons in the next layer, which is also described as fully connected. The strength of the weights indicates how important the connection is between the post-neuron and the pre-neuron [12].

While using ANN for pattern recognition, the input image data would be transferred to a vector also serve as the input neurons. Every input neuron holds a grayscale number from the MNIST pictures. Then the signals will propagate through a layer of weights W_{ij} . Every input layer neuron I_i is connected through its corresponding weight W_{ij} to next layer neuron X_j .

$$X_j = f\left(\sum_{i=1}^n I_i W_{ij}\right) \quad (1)$$

In the receiving end, the weighted sum is run through a nonlinear function $f()$ then fed to next layer neuron X_j . And the nonlinear function we used is the SIGMOID function.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

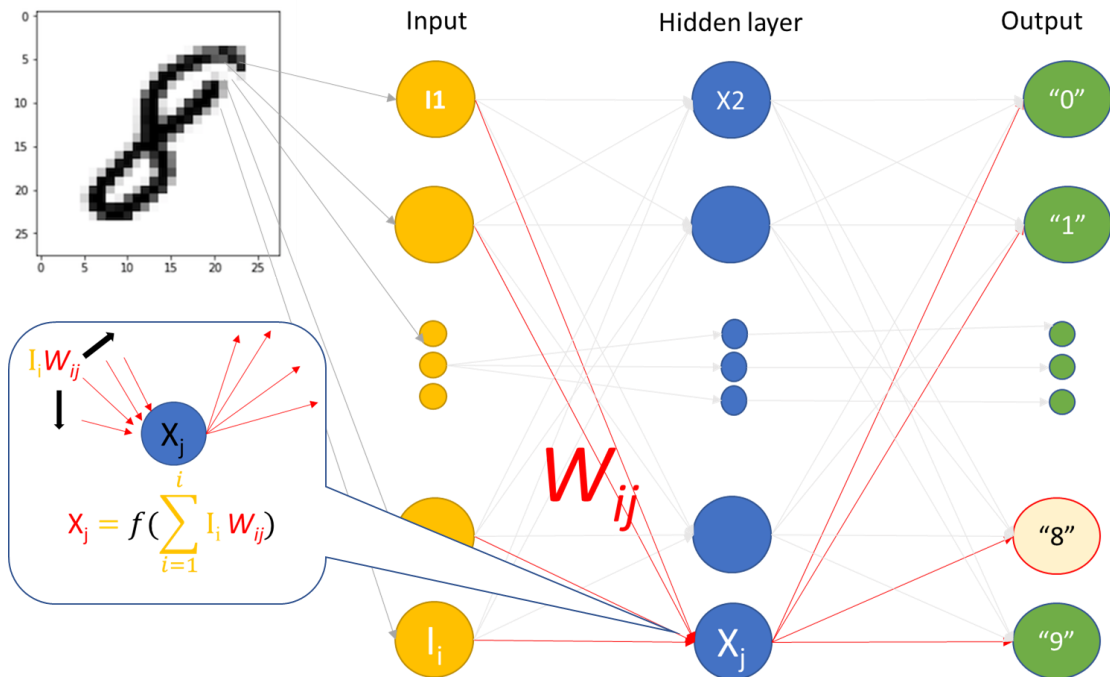


Figure 7. A fully connected ANN for pattern recognition in which every input layer neuron I_i is connected through their corresponding weight W_{ij} to X_j . The summation is run through a nonlinear function $f()$, then fed to next layer neuron X_j [12].

The input neurons receive the grayscale values from the input pictures and through the complete fully connected neuron network, which was then processed and transferred to the 10 outputs neurons. Each one of the output neurons was flagged with "0 to 9". If the value of output neuron flagged with "8" is the largest after processed through the whole neural network, it indicates that the input image is most likely a handwritten number "8".

The successful recognition process can be achieved through training. And a right algorithm is needed to train the weights correctly.

3.3 Neural network training algorithm

The algorithms we used to get the neural network weights are the backpropagation and the gradient descent [1][12][13].

First, we feed the ANN with a labeled picture example. We already know the output neurons vector values will not be the correct (The corresponding neuron has the highest output value indicating the input picture has the highest probability to be that neuron) before training. Next, we calculate the difference between our computational output results and the correct (error term δ_j). Then the difference is feedbacked to our ANN and backpropagated through the network. The weights are adjusted through a procedure called stochastic gradient descent.

$$\Delta W_{ij} = I_i * \eta * \delta_j \quad (3)$$

Training of the network means optimization of each weight into an organized stage: Whenever a new picture input comes in, the neural network will spike the corresponding output neuron to the highest number, indicates the picture is the corresponding number.

We applied those weight values to our Cadence model of ANN and got the simulation results.

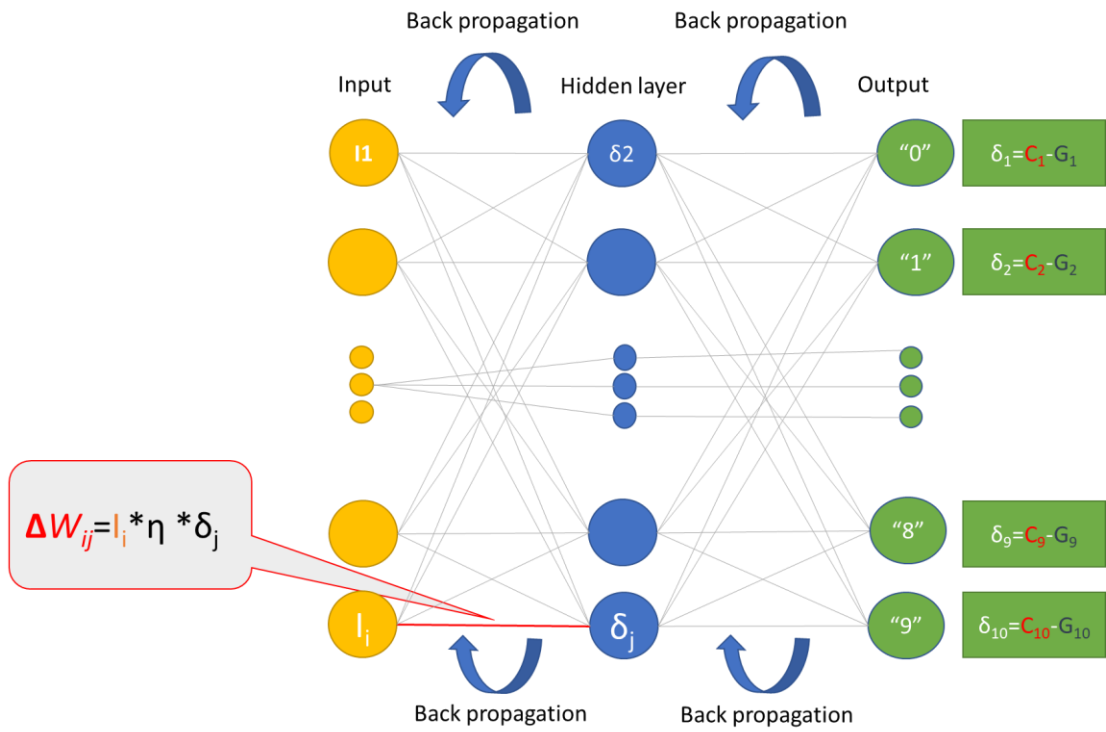


Figure 8. The backpropagation of the error term δ_j , each weight of the network is adjusted by the gradient descent method. The error can be reduced through training [12].

4 ANN using memristor device IFG (Ionic floating gate memory)

4.1 Memristor, the missing component

Memristor is a passive two terminals fundamental electrical component first named by Professor Leon Chua at UC Berkeley in 1971 [14]. It was declared as the fourth fundamental circuit element besides resistor, capacitor, and inductor.

We already know those 3 fundamental electrical components: describe the relations between four circuit variables: Current i , Voltage v , Charge q , and Flux ϕ .

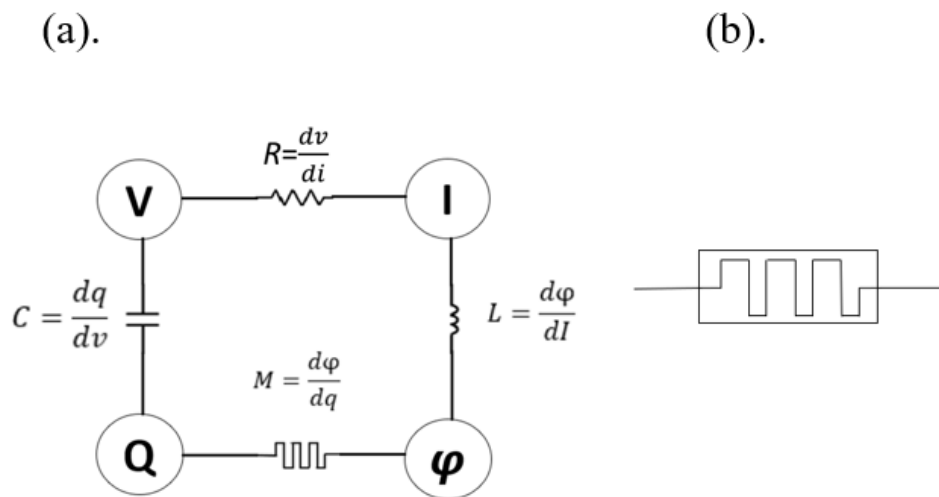


Figure 9. (a). The resistor defines the connection between Voltage and Current, the capacitor defines the connection between Voltage and Charge, the inductor defines the connection between Current and Flux. (b). the symbol of a memristor.

Professor Leon Chua defined the missing component as memristance and it can be measured as memristance M .

$$M(q) = \frac{d\varphi}{dq} \quad (4)$$

The equation could be change to

$$M(q(t)) = \frac{d\varphi}{dt} \cdot \frac{dt}{dq} = \frac{V(t)}{I(t)} \quad (5)$$

From this equation, we can find the memristance also has unit ohm like the resistor.

Memristor can function as a non-volatile memory that could be tuned to multiple states. It could also be operated at very low voltage. These properties could be implemented into the neural network as the synapse [15].

4.2 IFG as a memristor

The IFG device is a three terminals memristive device fabricated by researchers at Sandia National Laboratory [2]. They combined a redox transistor [3] and a volatile conductive bridge memory (CBM) [16]to make a non-volatile addressable synaptic memory (IFG memory).

The redox transistor has three layers. The first layer is the poly (3,4-ethylene-dioxythiophene): polystyrene sulfonate (PEDOT: PSS) film, the second layer is a layer of Nafion, the third layer is a layer of PEDOT: PSS film partially reduced with poly(ethylenimine) (PEI). The electrolyte of the redox is ion conductive but electron blocking. During the “write” process, a positive V_{write} is applied to the Gate. Cations H^+ flows from the PEI/PEDOT: PSS to the electrolyte, resulting in the polarization of the PEI. The ion concentration changing between Drain and Source represents thousands of conductance levels (thousands of weights).

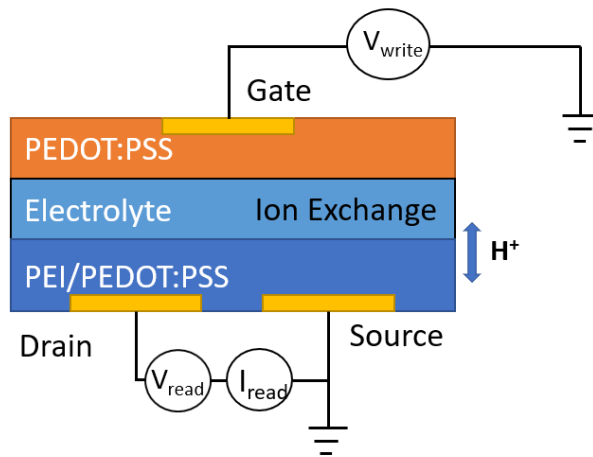


Figure 10. A polymer-based redox transistor [3].

This structure combined with a control bridge memory (CBM) [16] forms the Ionic Floating Gate memory (IFG). The CBM is a layer of silicon oxide doped with Ag and sandwiched by two Pt electrodes.

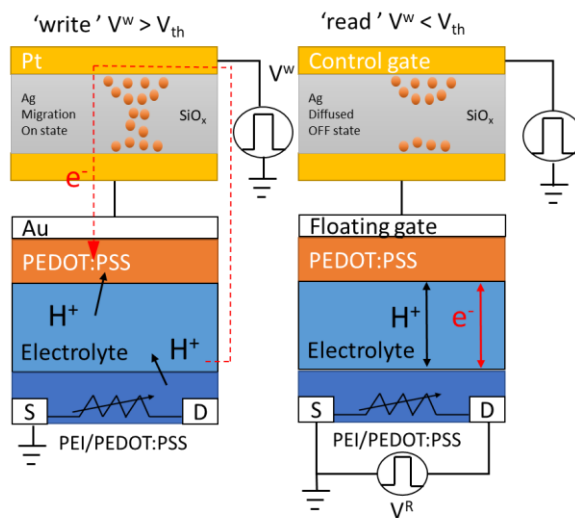


Figure 11. IFG memory structure with 'read' and 'write' process [2].

For the "Write" process, the voltage V^w is greater than the CBM threshold voltage V_{th} . It will result in the formation of the Ag filament. The electrons will insert to the top PEDOT: PSS gate. The channel will be polarized and therefore increase the conductance between source S and drain D nodes.

For the "OFF" state, the voltage V^w is way smaller than the V_{th} . It will result in the splitting of the Ag filament and a high resistance "OFF" state. The "OFF" state of the CBM stops the electron from inserting to the redox transistor and achieves non-volatile memory.

For the "read" process, a voltage V^R is applied to the source and drain node.

Through those states, we can easily manipulate the conductance (weight) of the IFG. Its conductivity is controlled by the charge state. Therefore, it is a memristor device. When built into a fully connected artificial neural network (ANN), the IFG device has the potential to do high-efficiency deep learning. In this work, we applied it for pattern recognition.

4.3 Features of IFG Device

Although there are two terminals memristor using phase changing memory (PCM) and filament-forming metal oxides (FFMO). Those devices typically suffer from several drawbacks. To be implemented as the synaptic unit in learning, they should have the ability to be tuned linearly and symmetrically. However, those devices must be driven out of thermal equilibrium. They typically need to overcome a very large energy barriers to tune the conductance states. Instead of linear and symmetric

changing of the conductance, they typically have an exponential and asymmetric conductance changing behavior [2].

On the other hand, two terminals memristors suffer from the current issue. The traditional two terminals memristors' "read & write" voltages are greater than microampere. As the currents are summed up through the entire array. The total current easily exceed the wire capacity. Large current also causes power dissipation problems which degrades the behavior of the circuit and reduces the learning efficiency. Due to those properties, PCM and FFMO memristors also have scalability problems [2].

However, the IFG device has several appealing features that could tackle those problems.

(1). Near-Linear conductance tuning:

The IFG memory combines a redox transistor [3] and a volatile conductive bridge memory (CBM) [16]. The conductance between Drain and Source are controlled by H^+ ion concentration. The H^+ acts like a dopant of the channel. Through its gradual modulation, thousands of near-linear levels (thousands of weights) are achieved.

(2). Low "read & write" voltages and good scalability

The "read & write" process only requires few millivolts voltages and the read currents are smaller than 10 nano amperes. When we implement the device into a large array, smaller summation current could help to reduce the wire size and the

power dissipation. These properties could solve the traditional two terminals devices' current issues and scalability problems.

(3). Addressable and parallel programmable

The CBM also makes the IFG device addressable and parallel programmable.

(4). Good stability

The IFG could potentially endure over 1 billion read-write operation and over 1 megahertz frequency.

4.4 IFG memory as ANN synapses

For pattern recognition, we built a three-layer artificial neural network: one input layer, one hidden layer, one output layer. The weights of our ANN are already trained and holds optimized values. Therefore, all the weights hold their fixed conductance values which are stored as the IFGs source-drain memristance G . All the IFGs are already written with the corresponding memristance and stay at the “read” mode. Our optimized weights were generated from the software by my colleague Xiaoyang Jia [23].

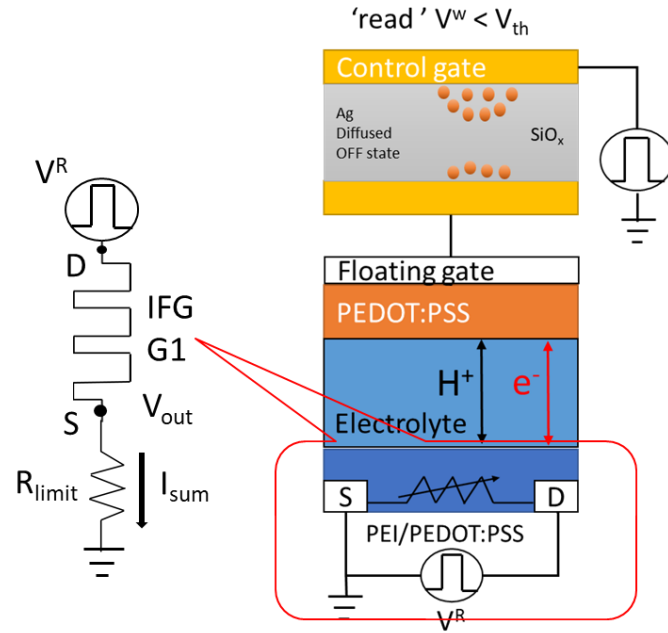


Figure 12. Every weight of our ANN is one IFG memory in the “read” state. A resistor R_{limit} is needed to read out the voltage V_{out} in the Cadence simulator.

Here we demonstrate the circuit implementation of the IFG synapse connection between the input layer and the hidden layer. Input neurons are driven by the MNIST database images. Each input layer neuron is assigned a normalized greyscale number from the MNIST database. We add voltage unit “volt” to the value and apply it to the circuit as the voltage source V_R . If the normalized input neuron value is “0.6”, we will apply 0.6V voltage as the voltage input V_R in our circuit. The IFG memories’ source and drain memristance G_1 represent the weights.

$$V_{\text{out}} = \frac{V_R \cdot G_1 \cdot R_{\text{limit}}}{1 + R_{\text{limit}} \cdot G_1} \quad (6)$$

When we set R_{limit} to 1 milli ohm, $R_{\text{limit}} \cdot G_1 \ll 1$, the equation is

$$V_{out} = VR1 \cdot G1 \cdot R_{limit} \quad (7)$$

If the weight is negative, we set its corresponding voltage input V^R to negative,

Every hidden layer neuron sums up the current from all the weights connections with the input layer. The current value is read out by a resistor R_{limit} as the output voltage V_{out} .

Then the voltage will be fed into Sigmoid function (2) and placed as the next layer neuron value's voltage VR_j

$$V_{out} = R_{limit} \cdot \sum_{i=1}^n VR_i \cdot G_{ij} \quad (8)$$

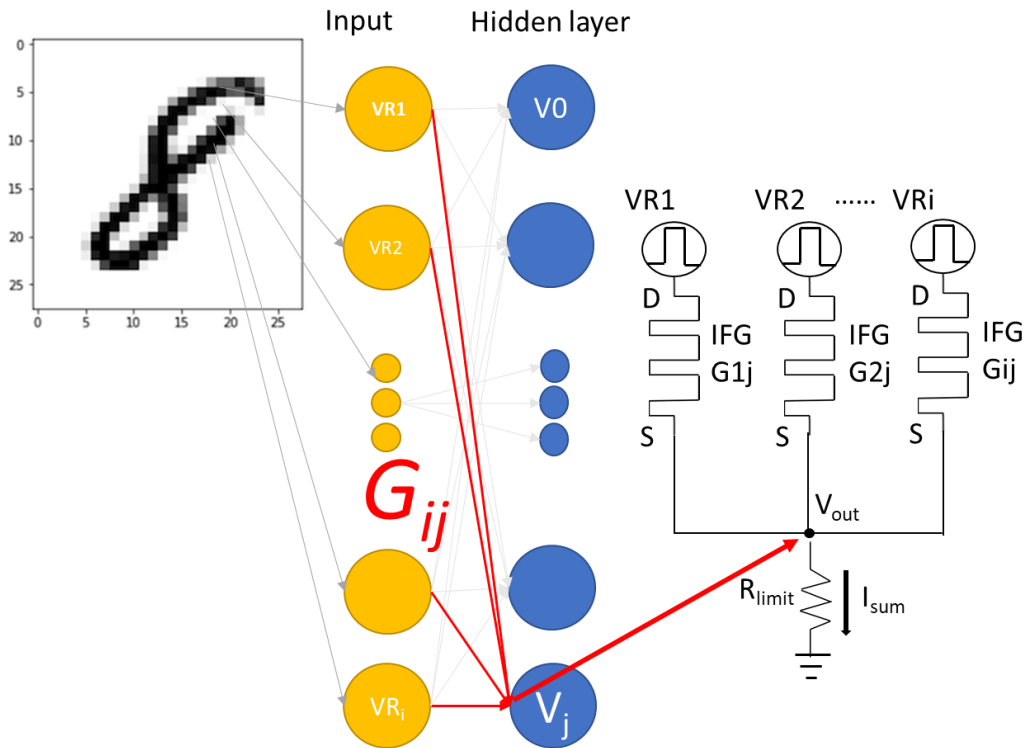


Figure 13. “Read” mode IFG as the synapses.

4.5 Fully connected ANN using IFG

We devise two demonstration networks. To testify the principle of the ANN for pattern recognition, we first designed and tested a smaller and simpler network (49 input neurons and 10 hidden layer neurons). It contains 69 neurons and 590 weights. We cropped the original 70,000 MNIST data set from $28 \times 28 = 784$ pixels resolution to $7 \times 7 = 49$ pixels resolution and done the training and testing. We used successfully optimized weights [23].

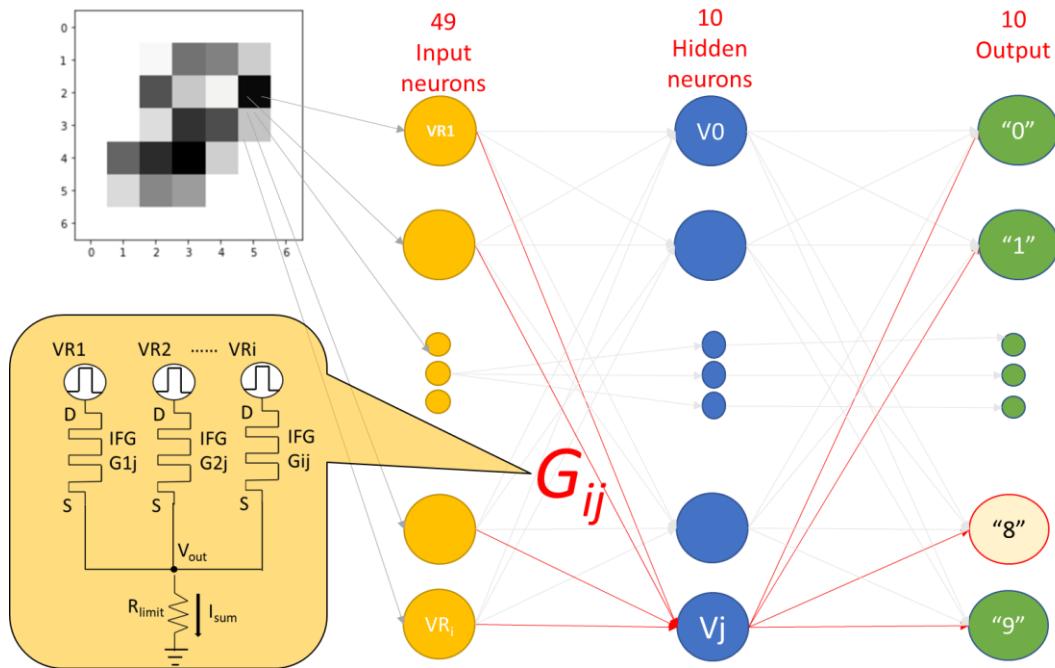


Figure 14. The reduced ANN using 7×7 pixels MNIST data set for training and testing. The structure size is 49 input neurons, 10 hidden layer neurons, and 10 output neurons. Every weight G_{ij} is implemented by an IFG memory. The small circle indicates the neurons in between.

For comparison, the original-sized ANN has 784 input neurons, 100 hidden layer neurons, and 10 output neurons. Those 894 neurons make up to 79400 weights.

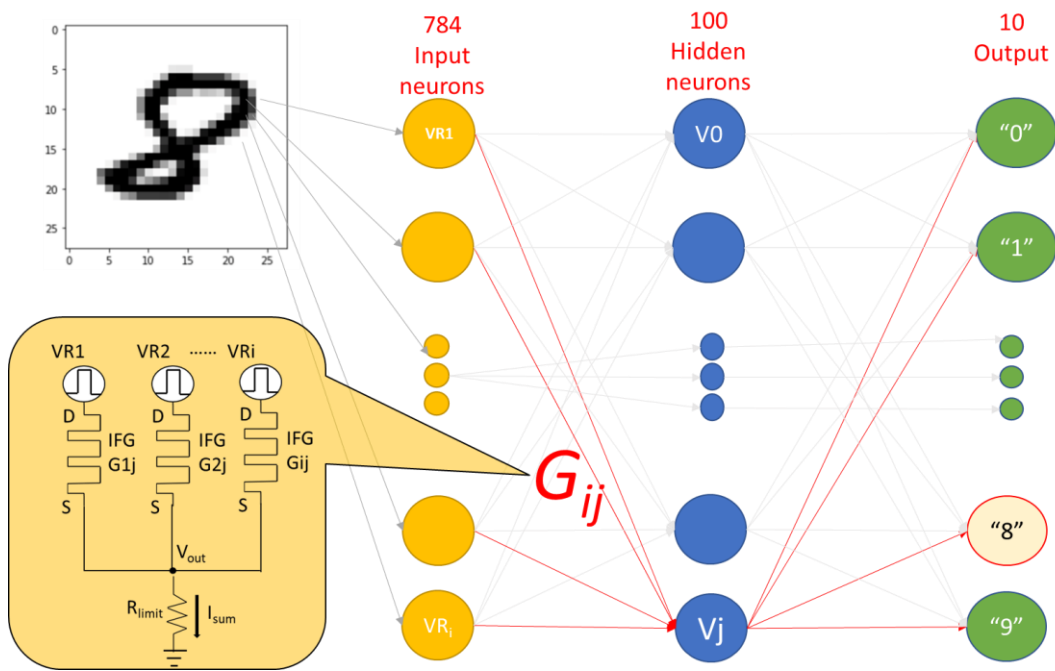


Figure 15. ANN using original 28×28 MNIST data set for training and testing.

The structure size is 784 input neurons, 100 hidden layer neurons, and 10 output neurons. The small circle indicates the neurons in between.

Both the reduced sized ANN and the original sized ANN can successfully recognize the numbers.

5 Cadence model and simulation

5.1 Cadence model for IFG memory

Our Spice IFG model is a three-terminal memristor device based on the Sandia National Laboratory's IFG memory [2]. My colleague Donguk Choi designed our own VerilogA code for the IFG. The VerilogA code is listed in the Appendix [24].

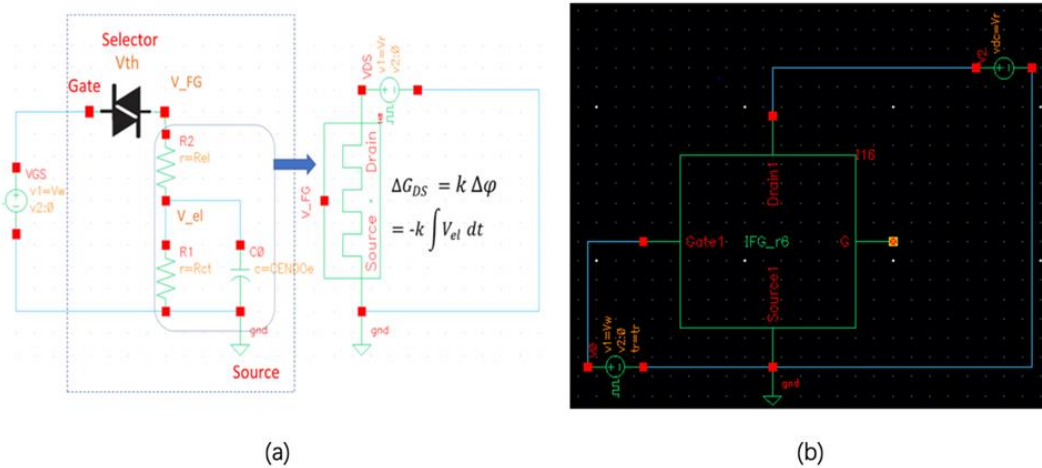


Figure 16. (a). The equivalent circuit of the IFG device. (b). The Cadence circuit model of IFG, V_w is the “write” voltage applied to the “Gate” node. V_r is the “Read” voltage applied to drain. Inside the instance of IFG is the VerilogA code (attached in Appendix)

To set the weight of an IFG memory, we applied voltage to the gate node. For the IFG device [2], the writing can be achieved by applying a voltage greater than the CBM threshold voltage V_{th} at the gate node.

For a demonstration of the weight writing process, we applied the ± 0.95 V voltage to the gate and source node. If the V_{gate} is negative, the conductance increase. If the difference is positive, the conductance drops. If we stop the gate voltage, the

conductance between source and drain holds its value. The threshold voltage has been set to 0.4V according to the device parameter [2].

The conductance of the IFG device is successfully set with the operation above. And after stopping the gate and source node voltages, the written conductance state stays constant, thus we fully control the IFG to achieve state retention.

As we can see in the simulation result, our conductance tuning process is near-linear and very symmetric. Those are the expected properties to improve the ANN learning efficiency.

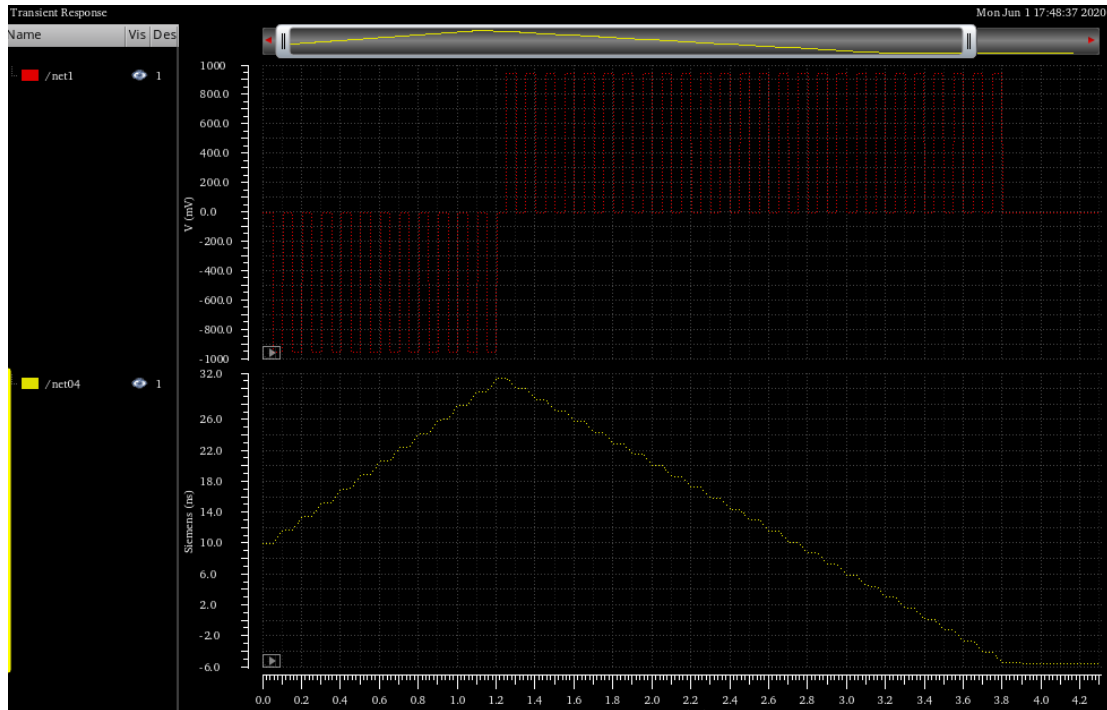


Figure 17. Weight writing process of the IFG memory. The yellow line is the conductance of the IFG memory. The red line is the “write” voltage applied to the “Gate”, the yellow line is the source voltage and the white line is the conductance of the IFG memory.

5.2 Network building and weight setting

After the weight setting the IFG memory serves as a conductor. Its memristance is fixed. To verify the role of ANN for pattern recognition, we used the resistors to build a fully connected ANN. Our optimized weights were generated from the software [23]. The equation 8 will be reshaped to the following equation (9).

$$V_{out} = R_{limt} \cdot \sum_{i=1}^i VR_i \cdot \frac{1}{R_{ij}} \quad (9)$$

The values of R_{ij} are reciprocal numbers of the optimized conductance weights W_{ij} which we get from the software training.

The VR_i in this equation is the input of the entire network. It was normalized from the greyscale numbers of the MNIST pictures pixels. All the greyscale numbers were divided by 255 to normalize it. Then we applied the same number of voltages as the input to VR as the circuit network input. For example, the pixels circled in Figure 17 have the same input value as $VR=1V$.

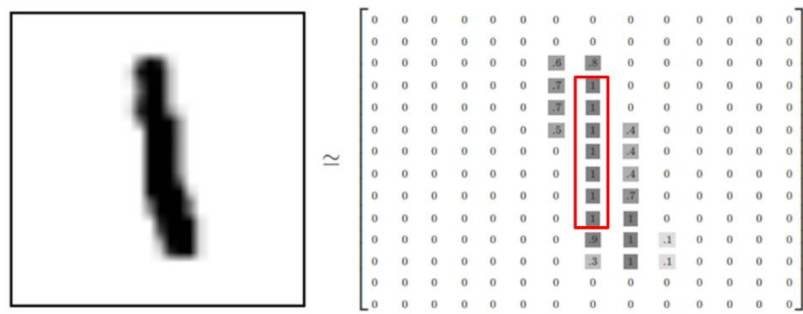


Figure 18. Normalization of the MNIST pictures. Every greyscale number is divided by 255 and normalized.

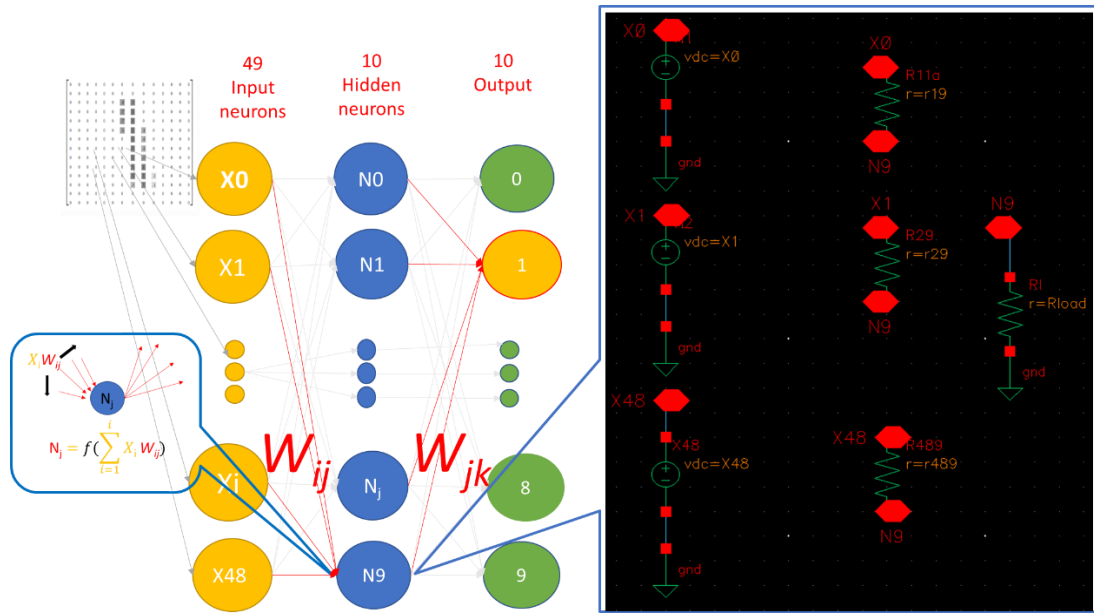


Figure 19. Circuit demonstration of the connections between the 49 input layer neurons and one hidden layer neuron N9.

For easy demonstration, we only show three weight connections. X0 to X48 are the input neurons, and they were applied input voltages $X_i = VR$. VR is the normalized voltage input from the MNIST data set shown in Figure 17. Every resistor represents the optimized weights from software training. The circuit simulation result is the current summation at node X48, therefore the voltage value at the R_{load} . The connections of the other 9 hidden layer neurons N_j are similar, but with their unique weight settings. After the simulation, we feed the summation to the activation function. The results were then used as the next layer input voltages VN_j .

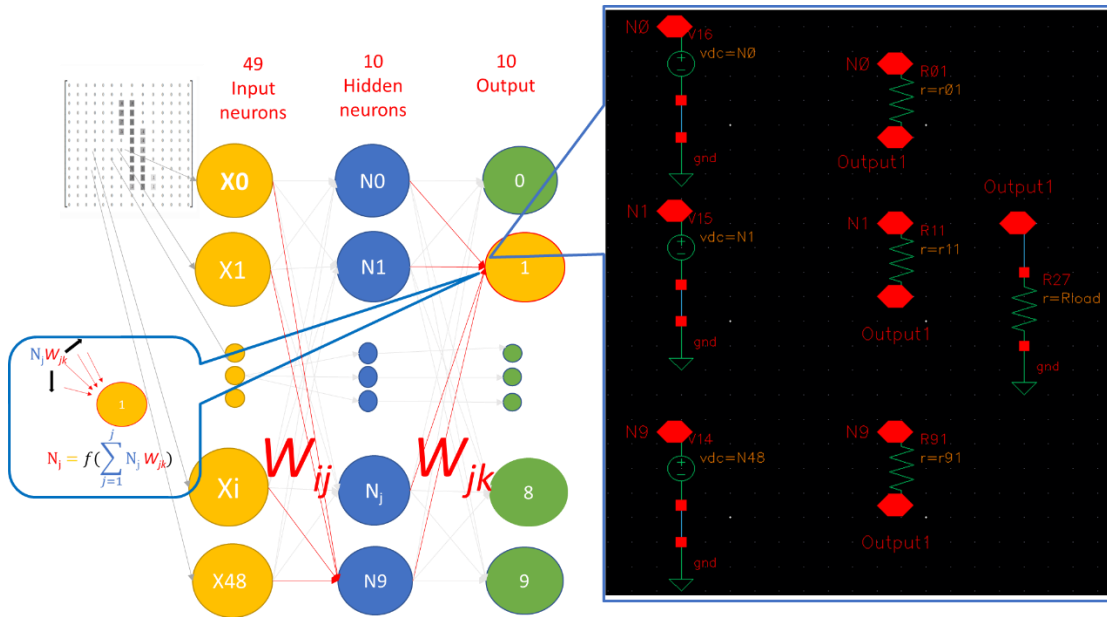


Figure 20. Circuit demonstration of the connections between the 10 hidden layer neurons and the output layer neuron “1”.

For easy demonstration, we only show three connections. N0 to N9 is the hidden layer neurons, and they were applied with output voltages VN_i from Figure 19. Every resistor represents the weights optimized through software training. The circuit simulation result is the current summation at node “1”, therefore the voltage value at the R_{load} . The connections of the other 9 output layer neurons are similar but with their unique weight settings. The summations at output neurons are fed to the activation function. All ten output neurons’ results are ranked in order. The neuron with the highest output voltage value indicates that the corresponding input picture has the highest probability to be the correct number.

Our reduced sized ANN contains $49+10+10=4,900$ neurons while the other original sized ANN contains $784 \times 100 \times 10=784,000$ neurons. For every IFG memory, we need

to control three nodes and apply unique voltage to control the weight. As we can see from these numbers, using a schematic SPICE simulation set up is rather cumbersome. To solve the problem, we used the text-based “Spectre” simulation inside Cadence. It doesn’t require schematic input and for our 784,000 neurons ANN, it runs less than a few seconds. We used Python to generate text-based connections of our conductor ANN circuit. The Python code is listed in Appendix 2.1. The activation function code is also listed in Appendix 2.2.

5.3 Results and comparison

In the software simulation, the reduced ANN performed a recognition accuracy of 86%. The original sized ANN has a recognition accuracy of 95%.

In the hardware simulation. We first ran 10 test MNIST data with our ANN. For our reduced sized ANN, it recognized 6 numbers out of 10, shown below for demonstration are the test results for the correct number “0” and the wrong number “7”. The other 8 numbers of recognition results could be found in Appendix 3.

output	signal	rank	number	image
N0	1.03E-03	1		
N1	-4.60E-03	8		
N2	-3.42E-03	2	0	
N3	-3.89E-03	4		
N4	-6.93E-03	9		
N5	-3.72E-03	3		
N6	-4.45E-03	5		
N7	-4.51E-03	6		
N8	-4.51E-03	7		
N9	-7.55E-03	10		

Figure 21. An ANN recognition result of number “0” from our reduced network.

The N0 to N9 are the voltage values of the output neurons identify digit from “0” to “9”. Rank is the order of that voltage value among ten output signals. For this result, the signal for output neuron N0 ranks the first. It means our ANN recognize this picture as digit “0”, which is correct.

output	signal	rank	number	image
N0	-3.13E-03	5		
N1	-1.74E-03	3	7	
N2	3.69E-04	1		
N3	-2.55E-03	4		
N4	-5.27E-03	9		
N5	-4.82E-03	8		
N6	-3.67E-03	6		
N7	-6.61E-03	10		
N8	-1.56E-03	2		
N9	-4.17E-03	7		

Figure 22. An ANN recognition result of number “7” from our reduced network.

The N0 to N9 are the voltage values of the output neurons identify digit from “0” to “9”. Rank is the order of that voltage value among ten output signals. For this result, the signal for output neuron N2 ranks the first. It means our ANN confused this picture as digit “2”, which is incorrect.

For our original ANN, it recognized all ten numbers. We demonstrate the test results of the number “7” here. Other recognition results could also be found in Appendix 3.

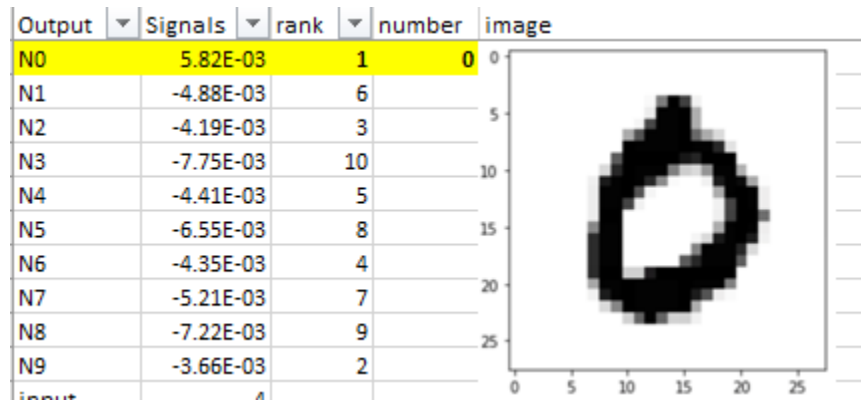


Figure 23. An ANN recognition result of our full-size network.

The N0 to N9 are the output neurons identify digit from “0” to “9”. Rank is the order of output signals. For this result, the signal for output neuron N0 ranks the first. It means our ANN recognize this picture as digit “0”, which is correct.

To better demonstrate the capability of our network, we test more images to calculate the accuracy. We test several images until the accuracy becomes stable. We test our reduced ANN array with 2000 images, it has the accuracy of 78.80%. It’s lower than the software.

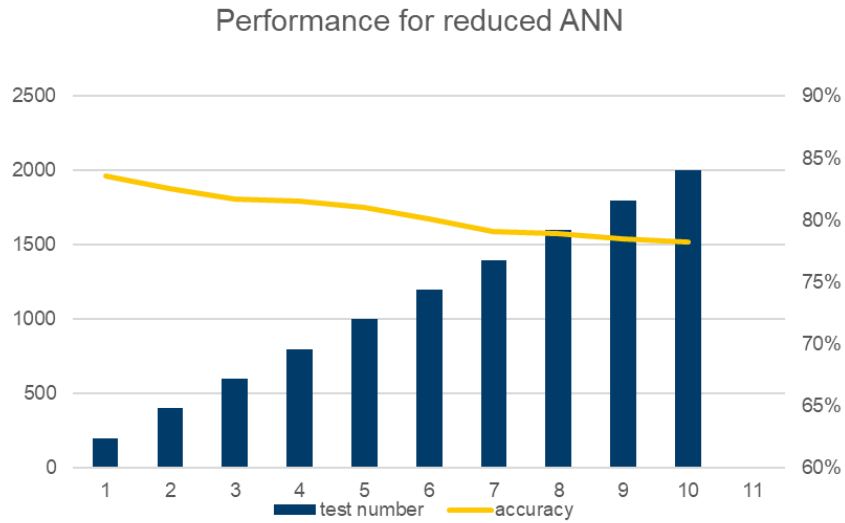


Figure 24. Performance for reduced ANN with 2,000 test samples.

Our network is 17 percentage lower than the software simulation result. We also test our original ANN with 2000 images, it has the accuracy of 93.82%. The 1st recognition error image came out at the 93 sample. The accuracy is lower than the software. The simulation time of the original IFG ANN is 14s per test which limits the simulation time.

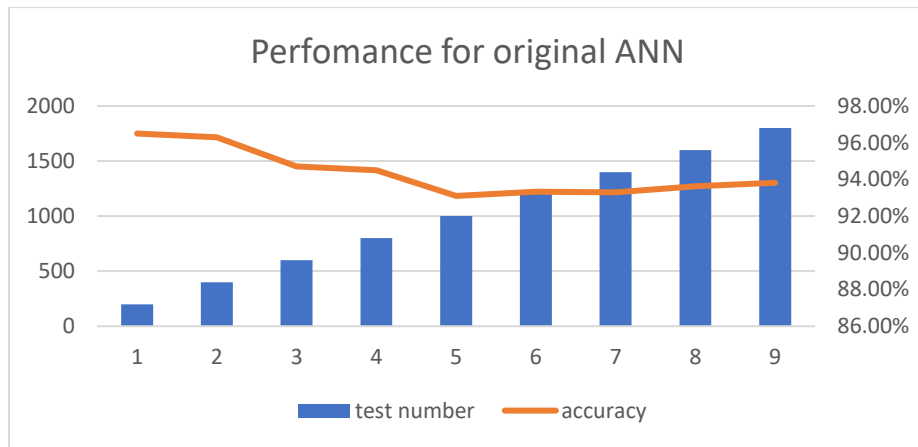


Figure 25. Performance for reduced ANN with 1800 test samples.

6 Conclusion

Artificial neural network forms the core part of the deep learning models. It has a huge potential for improving our daily life. In this thesis report, we successfully applied a Cadence model of the IFG device, designed IFG-based neural networks, and achieved the weight setting properly. By using Cadence tools, we successfully simulated two IFG-based neural networks to demonstrate their capability for pattern recognition. One reduced ANN has 69 neurons 590 weights. The original ANN has 894 neurons and 79400 weights. For the ten testing MNIST pictures. The reduced sized network recognized 6 numbers out of 10. The original sized networks yielded correct recognition results as listed in Appendix Table 1 and Table 2. The accuracy of the reduced ANN is 78.80%. The accuracy of the original ANN is 93.82%. From the comparison between reduced ANN and original ANN, we found the higher complexity of the neural network is, the higher recognition accuracy we will get. And the hardware implementation has lower accuracy than the software. Because we add distortion like the Rlimt into the ANN. We successfully testify that the IFG has the potential to be implemented as the synaptic memory in ANN. The value of our optimized weights ranges from 0.1ms to 10ms. After the implementation of our IFG network, the value of our output neurons ranges from 1mV to 10mV shown in Table 1 and Table 2.

7 Future work

This report provided a compact Cadence model of a fully connected artificial neural network (ANN) using the IFG device. MNIST data set was used to train and test our network. To better investigate and demonstrate the principle of pattern recognition we reduced our network size. Our largest network which consists of 79,400 weights has the highest accuracy of 93.82%. However, Geoffrey E. Hinton and his group achieved 98% state of art recognition accuracy [4]. It was achieved by a very large scale of network and trillions of connecting weights. Their ANN has one input layer consists of 784 neurons, two hidden layers each consists of 500 neurons, one top layer consists of 2000 neurons and 10 output neurons at the output layer. This large neural network builds up to 1,662,000 weights. The higher complexity of the neural network is, the higher the recognition accuracy you will get. Our work is a good starting point on pattern recognition using memristive devices. To improve recognition accuracy, we need to build much larger memristive arrays.

For our network, we applied the SIGMOID function. But the ReLU [7] function is more commonly used nowadays. It was partially inspired by the action potential behavior of the biological neurons. It is easier to train the neural network with the ReLU function. To improve our training efficiency, we could implement the ReLU as the nonlinear function in the future.

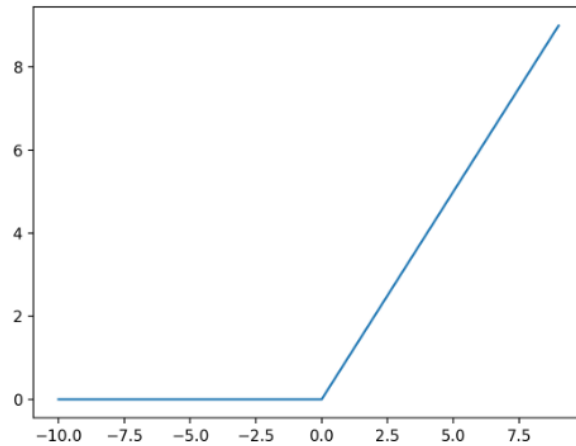


Figure 26. ReLU function

On the other hand, our work demonstrated pattern recognition using MNIST images and fully connected ANN. The MNIST images are all greyscale images. However, a typical colored image contains three 2D input arrays in three RGB color channels. To do colorful image recognition, another deep learning network called convolutional neural network (CNN) is required [1]. Shown in Figure 23. CNN's training algorithms and network structures are different from ANN. But our memristive device array can also be implemented to the CNN structure as the weights with proper training. Therefore, our memristive array can also be used in videos, audios, and language recognition.

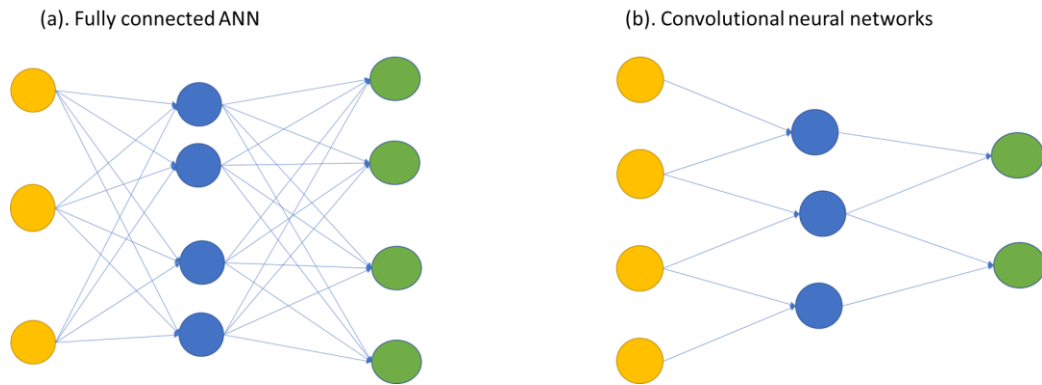


Figure 27. Structure comparisons between (a). fully connected ANN and (b). Convolutional neural networks CNN. The CNN weights can also be implemented with IFG devices.

Above all, our memristive array has the potential to be the future implementation of all kinds of deep neural network systems.

Appendix

1. VerilogA model for IFG

```
// VerilogA for mylib, IFG_r3_yh, veriloga

`include "constants.vams"
`include "disciplines.vams"

nature conductance
  access = Siemens;
  units = "S";
  abstol = 0.01n;
end nature

discipline Conductance
  potential conductance;
enddiscipline

module IFG_r3_yh (Gate1, Drain1, Source1, G);
  inout Gate1, Drain1, Source1, G;

  electrical Gate1, Drain1, Source1, N_el;
  Conductance G;
  real k, q;

  parameter real Rel = 1000 ;
```



```

parameter real Rct = 100M ;
parameter real Cenode = 10u ;
parameter real pulse_w = 50u ;
parameter real pulse_h_e = -0.95;
parameter real pulse_h_p = 1.2;
parameter real n_p= 50;
parameter real Gmax = 100n;
parameter real Gmin = 50n;
parameter real G0 = Gmax ;
parameter real Vth = 0.6;
parameter real k1 = 1.024;
real g_last = G0;
real last_t = 0 ;
real dt ;
real g, delphi;
real delG = Gmax - Gmin ;
real k_p = delG/(pulse_w*pulse_h_p*n_p) ;
real k_e = -delG/(pulse_w*pulse_h_e*n_p) ;
real cur_time, last_time = $abstime ;

analog begin
    q = Cenode * V(N_el,Source1);
    V(Gate1, N_el) <+ Rel * I(Gate1, N_el);
    I(N_el, Source1) <+ V(N_el, Source1)/Rct + ddt( q ) ;
    if (V(N_el, Source1) > 0)
        k = k_p*k1 ;
    else

```

```

        k = k_e*k1 ;

cur_time = $abstime;
dt = cur_time - last_time;
last_time = cur_time;

if (abs(V(N_el, Source1))>Vth)
    g_last = g_last + k*dt*V(Source1, N_el);
    $display("Vext = %7.3f Vint = %7.3f, G = %.3e time = %.2e",
V(Gate1, Source1), V(N_el, Source1), g_last, cur_time);

    I(Source1,Drain1) <+ g_last*V(Source1, Drain1);
    Siemens(G) <+ g_last ;
end
endmodule

```

2. Cadence text-based simulation codes

2.1 Python code to generate text based “Spectre” simulation

```

import numpy as np

row = 28*28
col = 100

def print_header():
    f.write("simulator lang=spectre\n")
    f.write("global 0\n")
    f.write("parameters Rload=1M \\n")

```

```

res = 1/np.loadtxt("wih.txt")
x = np.loadtxt(f"input{inp}.txt")

for i in range(0,row):
    f.write(f"\tx{i}={x[i]} \\n")
    for j in range(0,col):
        res0 = res[i][j]
        if res0>=0:
            f.write(f"\tr{i}{j}={res0} r{i}{j}b=1T \\n")
        else:
            f.write(f"\tr{i}{j}=1T r{i}{j}b={-res0} \\n")
    f.write("\n")

def form_array():

    for i in range(0, row):
        f.write(f"\-Vr{i} (\-X{i} 0) vsource dc=-x{i} type=dc\n")
        f.write(f"Vr{i} (X{i} 0) vsource dc=x{i} type=dc\n")
    f.write("\n")

    for i in range(0, row):
        for j in range(0, col):
            f.write(f"R{i}{j}b (\-X{i} N{j}) resistor r=r{i}{j}b\n")
            f.write(f"R{i}{j} (X{i} N{j}) resistor r=r{i}{j}\n")
        f.write("\n")

    for j in range(0, col):

```

```

f.write(f"R1{j} (N{j} 0) resistor r=Rload\n")

def print_simul():

    f.write("simulatorOptions options psfversion=\"1.1.0\" reltol=1e-3 vabstol=1e-6
    \\n")

    f.write("\tiabstol=1e-12 temp=27 tnom=27 scalem=1.0 scale=1.0 gmin=1e-12
    rforce=1 \\n")

    f.write("\tmaxnotes=5 maxwarns=5 digits=5 cols=80 pivrel=1e-3 \\n")

    f.write("\tsensfile=\"./psf/sens.output\" checklimitdest=psf \n")

    f.write("tran tran stop=1u write=\"spectre.ic\" writefinal=\"spectre.fc\" \\n")

    f.write("\tannotate=status maxiters=5 \n")

    f.write("finalTimeOP info what=oppoint where=rawfile\n")

    f.write("modelParameter info what=models where=rawfile\n")

    f.write("element info what=inst where=rawfile\n")

    f.write("outputParameter info what=output where=rawfile\n")

    f.write("designParamVals info what=parameters where=rawfile\n")

    f.write("primitives info what=primitives where=rawfile\n")

    f.write("subckts info what=subckts where=rawfile\n")

    f.write("\n")

    for j in range(0, col):

        f.write(f"save N{j} \n")

    f.write("saveOptions options save=allpub\n")

#gen_xw()

for inp in range(0,2):

    f = open(f"test{inp}.scs",'w')

    print_header()

```

```
form_array()
print_simul()
f.close()
```

2.2 Python code to apply the activation function to the signals

```
import numpy as np

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

trial = int(input())

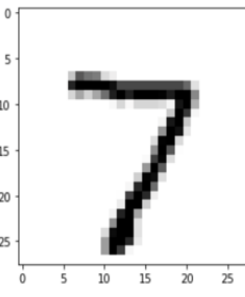
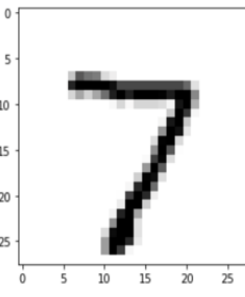
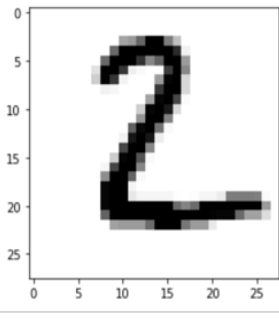
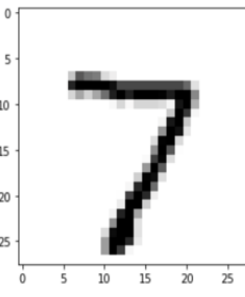
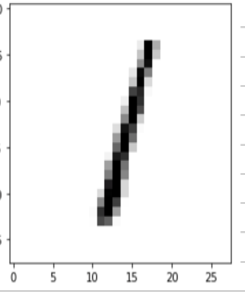
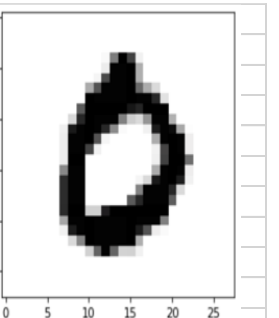
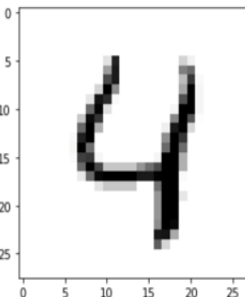
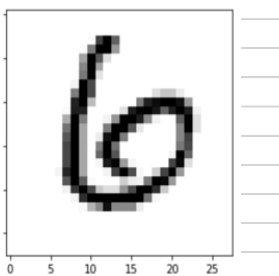
ho = np.loadtxt('ho0.txt')
if trial > 1:
    for i in range(1,trial):
        ho = np.vstack((ho,np.loadtxt(f'ho{i}.txt')))
else:
    ho = np.vstack((ho,np.zeros(np.size(ho))))

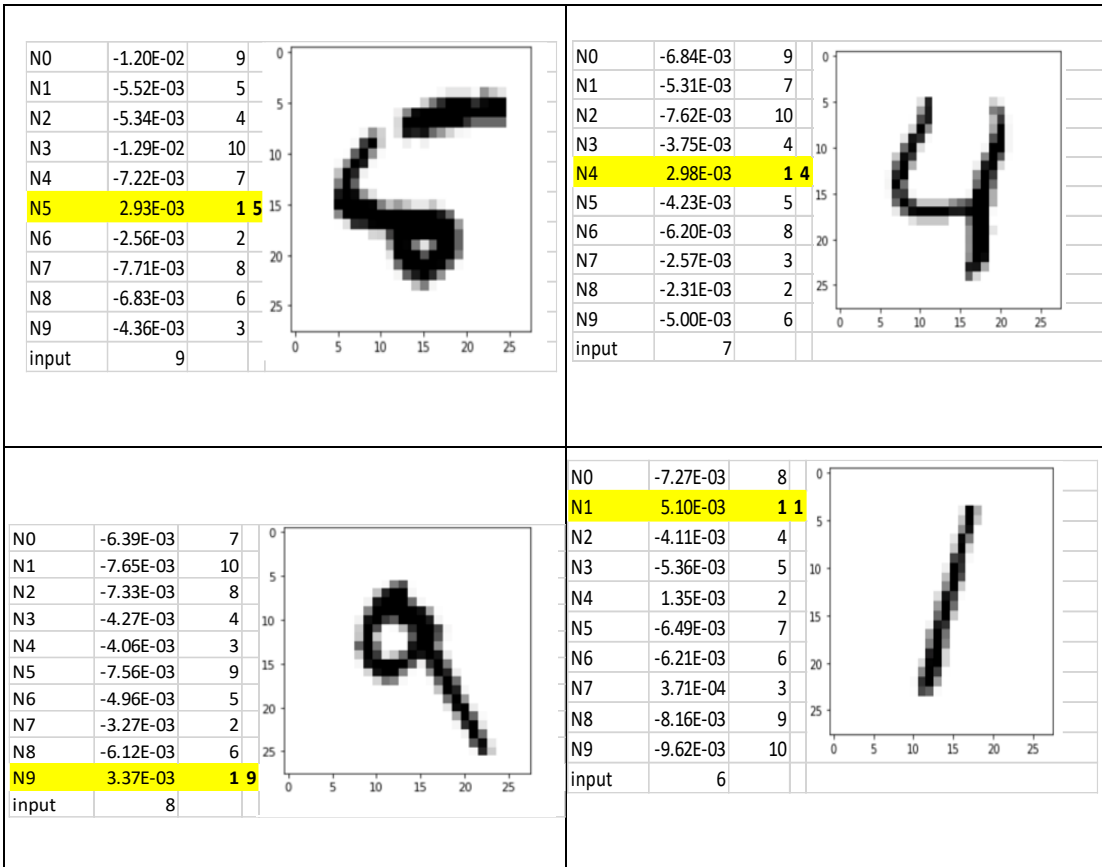
hon = (ho-np.mean(ho))/np.std(ho)
hona = sigmoid(hon)
for i in range(0,trial):
    np.savetxt(f'hon{i}.txt',hona[i,:])

hoa = sigmoid(ho)
for i in range(0,trial):
    np.savetxt(f'hoa{i}.txt',hoa[i,:])
```

3. Recognition results of original and reduced sized ANN

Table 1. Recognition results from original sized ANN

<table border="1"> <thead> <tr> <th>Output</th> <th>Signals</th> <th>rank</th> <th>image</th> </tr> </thead> <tbody> <tr><td>N0</td><td>-5.12E-03</td><td>8</td><td rowspan="10"></td></tr> <tr><td>N1</td><td>-6.44E-03</td><td>9</td></tr> <tr><td>N2</td><td>-4.12E-03</td><td>4</td></tr> <tr><td>N3</td><td>-4.86E-03</td><td>6</td></tr> <tr><td>N4</td><td>-4.65E-03</td><td>5</td></tr> <tr><td>N5</td><td>-3.40E-03</td><td>2</td></tr> <tr><td>N6</td><td>-4.99E-03</td><td>7</td></tr> <tr><td>N7</td><td>5.14E-03</td><td>1 7</td></tr> <tr><td>N8</td><td>-6.50E-03</td><td>10</td></tr> <tr><td>N9</td><td>-3.47E-03</td><td>3</td></tr> <tr><td>input</td><td></td><td></td></tr> </tbody> </table>	Output	Signals	rank	image	N0	-5.12E-03	8		N1	-6.44E-03	9	N2	-4.12E-03	4	N3	-4.86E-03	6	N4	-4.65E-03	5	N5	-3.40E-03	2	N6	-4.99E-03	7	N7	5.14E-03	1 7	N8	-6.50E-03	10	N9	-3.47E-03	3	input			<table border="1"> <tbody> <tr><td>N0</td><td>-2.85E-03</td><td>2</td></tr> <tr><td>N1</td><td>-4.68E-03</td><td>5</td></tr> <tr><td>N2</td><td>4.99E-03</td><td>1 2</td></tr> <tr><td>N3</td><td>-7.79E-03</td><td>10</td></tr> <tr><td>N4</td><td>-6.30E-03</td><td>8</td></tr> <tr><td>N5</td><td>-4.66E-03</td><td>4</td></tr> <tr><td>N6</td><td>-6.08E-03</td><td>7</td></tr> <tr><td>N7</td><td>-5.41E-03</td><td>6</td></tr> <tr><td>N8</td><td>-3.85E-03</td><td>3</td></tr> <tr><td>N9</td><td>-6.74E-03</td><td>9</td></tr> <tr><td>input</td><td>2</td><td></td></tr> </tbody> </table> 	N0	-2.85E-03	2	N1	-4.68E-03	5	N2	4.99E-03	1 2	N3	-7.79E-03	10	N4	-6.30E-03	8	N5	-4.66E-03	4	N6	-6.08E-03	7	N7	-5.41E-03	6	N8	-3.85E-03	3	N9	-6.74E-03	9	input	2	
Output	Signals	rank	image																																																																					
N0	-5.12E-03	8																																																																						
N1	-6.44E-03	9																																																																						
N2	-4.12E-03	4																																																																						
N3	-4.86E-03	6																																																																						
N4	-4.65E-03	5																																																																						
N5	-3.40E-03	2																																																																						
N6	-4.99E-03	7																																																																						
N7	5.14E-03	1 7																																																																						
N8	-6.50E-03	10																																																																						
N9	-3.47E-03	3																																																																						
input																																																																								
N0	-2.85E-03	2																																																																						
N1	-4.68E-03	5																																																																						
N2	4.99E-03	1 2																																																																						
N3	-7.79E-03	10																																																																						
N4	-6.30E-03	8																																																																						
N5	-4.66E-03	4																																																																						
N6	-6.08E-03	7																																																																						
N7	-5.41E-03	6																																																																						
N8	-3.85E-03	3																																																																						
N9	-6.74E-03	9																																																																						
input	2																																																																							
<table border="1"> <tbody> <tr><td>N0</td><td>-7.81E-03</td><td>8</td></tr> <tr><td>N1</td><td>4.22E-03</td><td>1 1</td></tr> <tr><td>N2</td><td>-4.42E-03</td><td>4</td></tr> <tr><td>N3</td><td>-5.43E-03</td><td>5</td></tr> <tr><td>N4</td><td>2.51E-03</td><td>2</td></tr> <tr><td>N5</td><td>-7.57E-03</td><td>7</td></tr> <tr><td>N6</td><td>-5.55E-03</td><td>6</td></tr> <tr><td>N7</td><td>1.93E-03</td><td>3</td></tr> <tr><td>N8</td><td>-8.79E-03</td><td>9</td></tr> <tr><td>N9</td><td>-1.26E-02</td><td>10</td></tr> <tr><td>input</td><td>3</td><td></td></tr> </tbody> </table> 	N0	-7.81E-03	8	N1	4.22E-03	1 1	N2	-4.42E-03	4	N3	-5.43E-03	5	N4	2.51E-03	2	N5	-7.57E-03	7	N6	-5.55E-03	6	N7	1.93E-03	3	N8	-8.79E-03	9	N9	-1.26E-02	10	input	3		<table border="1"> <tbody> <tr><td>N0</td><td>5.82E-03</td><td>1 0</td></tr> <tr><td>N1</td><td>-4.88E-03</td><td>6</td></tr> <tr><td>N2</td><td>-4.19E-03</td><td>3</td></tr> <tr><td>N3</td><td>-7.75E-03</td><td>10</td></tr> <tr><td>N4</td><td>-4.41E-03</td><td>5</td></tr> <tr><td>N5</td><td>-6.55E-03</td><td>8</td></tr> <tr><td>N6</td><td>-4.35E-03</td><td>4</td></tr> <tr><td>N7</td><td>-5.21E-03</td><td>7</td></tr> <tr><td>N8</td><td>-7.22E-03</td><td>9</td></tr> <tr><td>N9</td><td>-3.66E-03</td><td>2</td></tr> <tr><td>input</td><td>4</td><td></td></tr> </tbody> </table> 	N0	5.82E-03	1 0	N1	-4.88E-03	6	N2	-4.19E-03	3	N3	-7.75E-03	10	N4	-4.41E-03	5	N5	-6.55E-03	8	N6	-4.35E-03	4	N7	-5.21E-03	7	N8	-7.22E-03	9	N9	-3.66E-03	2	input	4						
N0	-7.81E-03	8																																																																						
N1	4.22E-03	1 1																																																																						
N2	-4.42E-03	4																																																																						
N3	-5.43E-03	5																																																																						
N4	2.51E-03	2																																																																						
N5	-7.57E-03	7																																																																						
N6	-5.55E-03	6																																																																						
N7	1.93E-03	3																																																																						
N8	-8.79E-03	9																																																																						
N9	-1.26E-02	10																																																																						
input	3																																																																							
N0	5.82E-03	1 0																																																																						
N1	-4.88E-03	6																																																																						
N2	-4.19E-03	3																																																																						
N3	-7.75E-03	10																																																																						
N4	-4.41E-03	5																																																																						
N5	-6.55E-03	8																																																																						
N6	-4.35E-03	4																																																																						
N7	-5.21E-03	7																																																																						
N8	-7.22E-03	9																																																																						
N9	-3.66E-03	2																																																																						
input	4																																																																							
<table border="1"> <tbody> <tr><td>N0</td><td>-6.44E-03</td><td>8</td></tr> <tr><td>N1</td><td>-5.61E-03</td><td>6</td></tr> <tr><td>N2</td><td>-4.73E-03</td><td>5</td></tr> <tr><td>N3</td><td>-8.55E-03</td><td>9</td></tr> <tr><td>N4</td><td>6.96E-03</td><td>1 4</td></tr> <tr><td>N5</td><td>-3.59E-03</td><td>2</td></tr> <tr><td>N6</td><td>-4.12E-03</td><td>3</td></tr> <tr><td>N7</td><td>-6.14E-03</td><td>7</td></tr> <tr><td>N8</td><td>-9.13E-03</td><td>10</td></tr> <tr><td>N9</td><td>-4.51E-03</td><td>4</td></tr> <tr><td>input</td><td>5</td><td></td></tr> </tbody> </table> 	N0	-6.44E-03	8	N1	-5.61E-03	6	N2	-4.73E-03	5	N3	-8.55E-03	9	N4	6.96E-03	1 4	N5	-3.59E-03	2	N6	-4.12E-03	3	N7	-6.14E-03	7	N8	-9.13E-03	10	N9	-4.51E-03	4	input	5		<table border="1"> <tbody> <tr><td>N0</td><td>-4.52E-03</td><td>4</td></tr> <tr><td>N1</td><td>-5.65E-03</td><td>7</td></tr> <tr><td>N2</td><td>-5.90E-03</td><td>9</td></tr> <tr><td>N3</td><td>-5.53E-03</td><td>6</td></tr> <tr><td>N4</td><td>-4.91E-03</td><td>5</td></tr> <tr><td>N5</td><td>-7.15E-03</td><td>10</td></tr> <tr><td>N6</td><td>-4.48E-03</td><td>3 6</td></tr> <tr><td>N7</td><td>-5.85E-03</td><td>8</td></tr> <tr><td>N8</td><td>-4.41E-03</td><td>2</td></tr> <tr><td>N9</td><td>4.32E-03</td><td>1</td></tr> <tr><td>input</td><td>"10"</td><td></td></tr> </tbody> </table> 	N0	-4.52E-03	4	N1	-5.65E-03	7	N2	-5.90E-03	9	N3	-5.53E-03	6	N4	-4.91E-03	5	N5	-7.15E-03	10	N6	-4.48E-03	3 6	N7	-5.85E-03	8	N8	-4.41E-03	2	N9	4.32E-03	1	input	"10"						
N0	-6.44E-03	8																																																																						
N1	-5.61E-03	6																																																																						
N2	-4.73E-03	5																																																																						
N3	-8.55E-03	9																																																																						
N4	6.96E-03	1 4																																																																						
N5	-3.59E-03	2																																																																						
N6	-4.12E-03	3																																																																						
N7	-6.14E-03	7																																																																						
N8	-9.13E-03	10																																																																						
N9	-4.51E-03	4																																																																						
input	5																																																																							
N0	-4.52E-03	4																																																																						
N1	-5.65E-03	7																																																																						
N2	-5.90E-03	9																																																																						
N3	-5.53E-03	6																																																																						
N4	-4.91E-03	5																																																																						
N5	-7.15E-03	10																																																																						
N6	-4.48E-03	3 6																																																																						
N7	-5.85E-03	8																																																																						
N8	-4.41E-03	2																																																																						
N9	4.32E-03	1																																																																						
input	"10"																																																																							



N0	-3.13E-03	5						N0	-6.79E-03	10						
N1	-1.74E-03	3	8					N1	-3.37E-03	4	9					
N2	3.69E-04	1						N2	-1.42E-03	3						
N3	-2.55E-03	4						N3	-3.73E-03	6						
N4	-5.27E-03	9						N4	-3.45E-03	5						
N5	-4.82E-03	8						N5	-5.71E-03	8						
N6	-3.67E-03	6						N6	-4.06E-03	7						
N7	-6.61E-03	10						N7	-8.67E-04	2						
N8	-1.56E-03	2						N8	-6.00E-03	9						
N9	-4.17E-03	7						N9	-1.76E-04	1						

References

- [1] LeCun, Y., Bengio, Y. & Hinton, G. "Deep learning." *Nature* **521**, 436–444 (2015). <https://doi.org/10.1038/nature14539>
- [2] Elliot J. Fuller, Scott T. Keene, Armantas Melianas, Zhongrui Wang, Sapan Agarwall, Yiyang Li, Yaakov Tuchman, Conrad D. James, Matthew J. Marinella, J. Joshua Yang, Alberto Salleo, A. Alec Talin, "Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing" *Science* **364**, 570–574 (2019) <https://doi.org/10.1126/science.aaw5581>
- [3] Yoeri van de Burgt, Y. Lubberman, E. Fuller, et al. "A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing." *Nature Mater* **16**, 414–418 (2017). <https://doi.org/10.1038/nmat4856>
- [4] Geoffrey E. Hinton et al. "A Fast Learning Algorithm for Deep Belief Nets Neural Computation" **18**, 1527–1554 (2006)
- [5] G. Zaharchuk, E. Gong, M. Wintermark, D. Rubin and C.P. Langlotz, "Deep Learning in Neuroradiology" *American Journal of Neuroradiology* October 2018, **39** (10) 1776-1784; DOI: <https://doi.org/10.3174/ajnr.A5543>
- [6] Esteva, A., Kuprel, B., Novoa, R. *et al.* "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* **542**, 115–118 (2017). <https://doi.org/10.1038/nature21056>
- [7] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, Aurélien Géron, O'Reilly - 2019
- [8] McCulloch, W.S., Pitts, W. "A logical calculus of the ideas immanent in nervous activity." *Bulletin of Mathematical Biophysics* **5**, 115–133 (1943). <https://doi.org/10.1007/BF02478259>
- [9] Leung, M. K., Xiong, H. Y., Lee, L. J. & Frey, B. J. "Deep learning of the tissue-regulated splicing code." *Bioinformatics* **30**, i121–i129 (2014).
- [10] Xiong, H. Y. et al. "The human splicing code reveals new insights into the genetic determinants of disease." *Science* **347**, 6218 (2015).
- [11] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," in *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141-142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.
- [12] G. W. Burr *et al.*, "Experimental Demonstration and Tolerancing of a Large-Scale Neural Network (165 000 Synapses) Using Phase-Change Memory as the Synaptic Weight Element," in *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498-3507, Nov. 2015, doi: 10.1109/TED.2015.2439635.

- [13] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
- [14] L. Chua, "Memristor-The missing circuit element," in *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507-519, September 1971, doi: 10.1109/TCT.1971.1083337.
- [15] Adamatzky, A., CHUA, L. Memristor Networks. New York: Springer, 2014.
- [16] Wang, Z., Joshi, S., Savel'ev, S. et al. "Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing." *Nature Mater* 16, 101–108 (2017). <https://doi.org/10.1038/nmat4756>
- [17] Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E. & Svetnik, V. "Deep neural nets as a method for quantitative structure-activity relationships." *J. Chem. Inf. Model.* 55, 263–274 (2015).
- [18] B. Chen, F. Cai, J. Zhou, W. Ma, P. Sheridan and W. D. Lu, "Efficient in-memory computing architecture based on crossbar arrays," 2015 IEEE International Electron Devices Meeting (IEDM), Washington, DC, 2015, pp. 17.5.1-17.5.4, doi: 10.1109/IEDM.2015.7409720.
- [19] Krizhevsky, A., Sutskever, I. & Hinton, G. "ImageNet classification with deep convolutional neural networks." In *Proc. Advances in Neural Information Processing Systems* 25 1090–1098 (2012).
- [20] Mikolov, T., Deoras, A., Povey, D., Burget, L. & Cernocky, J. "Strategies for training large scale neural network language models." In *Proc. Automatic Speech Recognition and Understanding* 196–201 (2011).
- [21] Scott T Keene et al "Optimized pulsed write schemes improve linearity and write speed for low-power organic neuromorphic devices." *Phys. D: Appl. Phys.* **51** (2018) 224002
- [22] L. O. Chua and Sung-Mo Kang, "Memristive devices and systems," in *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209-223, Feb. 1976, doi: 10.1109/PROC.1976.10092.
- [23] Xiaoyang Jia, "Design and Training of Memristor-based Neural Network" MS. Thesis, UC Santa Cruz (2020)
- [24] Donguk Choi, private communication "Compact modeling of IFG device."