UNIVERSIDAD CATÓLICA DEL NORTE

FACULTY OF PHYSICS AND ASTRONOMY

Spin Torque Nano-Oscillator As Nano-Electronic Component For Artificial Intelligence Hardware

A THESIS SUBMITTED FOR THE DEGREE OF BSc IN PHYSICS WITH A MINOR IN ASTRONOMY

Author:
David Rojas Jerez

Tutor:
PhD. Jorge Otálora Arias

Co-Tutor:
PhD. Johan Triana Galvis

August 1st, 2024

1 Acknowledgements

I dedicate this thesis to my beloved wife Paula and to my dog Sunny. Also to my friends and family. Thanks to Dr. Jorge Otalora and Dr. Johan Triana for giving me the opportunity of working on this research project and for the guidance.

2 Abstract

In this research thesis we study the implementation of the Spin Torque Nano Oscillator (STNO) for novel nano-spintronic devices capable of performing Artificial Intelligence (AI) workflows at low energy consumption.

We review a deep learning model based on a STNO hardware circuit for emulating the neural network for speech recognition, with great performance when it was evaluated on the Free Spoken Digit Dataset(FSDD) [1]. We reproduce the analytical models that describe the STNO magnetic oscillation behavior, to study the recognition performance of the algorithm as a function of the STNO parameters, like the radius and thickness of its free layer. We obtain that the performance of the AI became efficient at smaller radii and larger thicknesses, with an abrupt decrease as the radius increases. This abrupt decreasing occurs at smaller sizes for the STNOs with the uniform ground magnetic state.

We also implement the non linear magnetic oscillator model from Abreu, et.al. 2020 [2] for speech recognition directly on the functionality of the STNO neural network hardware. We achieve 90.5 % accuracy on the machine-learning task on a single STNO device with a total of 10.000 trainable parameters, which allows reaching similar accuracies than in those neural networks that implement conventional deep learning architectures such as Multilayer Perceptrons (MLP) or Recurrent Neural Networks (RNN) but with less trainable parameters. The implementation of the STNO for emulating the neural network directly on hardware circuits drastically reduces the energy consumption of the AI model. This showcases the STNO as an optimal nano-component for designing energy-efficient AI hardware.

keywords: Artificial Intelligence, Deep Learning, Hardware, Nano Electronics, Spintronics.

List of Figures

Contents

I Introductory background

1 Introduction

Over the last decade Artificial Intelligence (AI) has been advancing rapidly in terms of the amount of resources devoted to it and also in terms of its outputs [5]. AI is driving current innovation giving rise to novel technologies such as voice command systems [6], self driving cars [7], multi-modal chat bots [8], among many others. The most capable AI models essentially consist of complex artificial neural networks with billions of parameters. The computational and energy requirements to train these models are also growing at an unprecedented speed; for example, the energy required to generate an image with a text-to-image model is approximately half the energy needed to charge a smartphone. [9]. This high energy consumption becomes a big problem for edge and mobile devices. So at the moment, the most probable bottleneck for AI evolution will be the high energy consumption.

To overcome these problems new computing architecture is being developed under the name of AI Accelerators [10]. AI accelerators are specialized hardware designed to process AI tasks such as computer vision, language processing or speech recognition with increased computational efficiency. In recent years, several technological approaches have been proposed for optimizing AI computation, such as Analog In Memory Computing [11] , STT-MRAM, SOT-MRAM [12], Neuromorphic Computing [13] and spintronics [14], among others.

Neuromorphic Computing is a non Von Neumann paradigm heavily inspired by the human brain that mimics the structure and functionalities of the brain on analog or digital circuits such as magnetic-based circuits (spintronic devices), with the advantage of performing AI tasks at lower energy consumption than the usual computational models for Deep Layer Architectures. This is done by designing computing systems where the memory and processing unit are embedded in a single device, avoiding the data transfer between memory and processing unit.

Spintronics is the technological field where nano-electronics devices use the spin properties of electrons to process, store and transfer information more efficient energetically [15]. Over the past years there have been many research in spintronics devices for AI applications [14]. One such device is the Spin Torque Nano Oscillator (STNO), which consists of a Magnetic Tunnel Junction (MTJ) composed of two ferromagnetic layers separated by a non-magnetic intermediate layer (Normal layer). This device produces nonlinear magnetic oscillations when receiving an input Direct Current (DC); the injected electrons are spin-polarized by one magnetic layer, which transfers magnetic momentum to the other magnetic layer when they reach it, hence inducing magnetic oscillations in this layer. The frequency of the magnetization precession ranges between hundreds of megahertz and gigahertz. These magnetic oscillations are measured as voltage oscillations through magneto-resistive phenomena[16]. Spin torque Nano Oscillators offers various characteristics that highlights its potential for neuromorphic computing: its lateral size can be scaled up to 10 nm, its power consumption can get as low as 1 μW, they are compatible with complementary metal-oxide semiconductor (CMOS) technology, operate at room temperature and can be fabricated in large numbers within a single chip (currently up to hundreds of millions). Recently, a single STNO has been used to perform machine learning tasks such as speech recognition on the TI-46 Word Corpus dataset. Fig 1.1 shows a schematic diagram of a STNO [17].

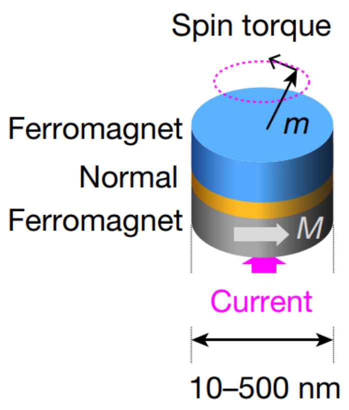


Figure 1.1: Spin Torque Nano Oscillator based on a Magnetic Tunel Junction

1.1 Overview and Scope of the thesis

The interest of this thesis is to study the implementation of the STNO as a spintronic hardware component for AI workloads at low energy consumption.

We implement a deep learning algorithm that utilizes the STNO to emulate the neural network directly on the hardware circuit. The efficacy of our implementation is studied by evaluating its performance on speech recognition on the Free Spoken Digit Dataset(FSDD) [1]. For this purpose, we derive an analytical model of the voltage dynamics of the STNO for two different ground states in the free layer, the vortex state and the uniform state; thus, we integrate the STNO into the AI system to evaluate its recognition performance as a function of different sizes of the free layer. We also evaluate the performance of the AI system using the non linear magnetic oscillator model from [2] (The authors used such model to study the effects of different acoustic transformations on the AI processing). Particularly, we variate the physical parameters of the STNO, such as the radius and thickness of the free layer, to check the performance of the AI model. By training this AI system to perform speech recognition on a new dataset (to date) we study the versatility of the STNO for AI applications and highlights its potential for solving the energy problems of AI.

This work is presented as follows: first, we provide a theoretical framework where we introduce the basic concepts of machine learning, deep learning, neuromorphic computing, and spintronics; then, in the methods section, we explain the AI model implemented. Finally we present and discuss the performance results of the model.

2 Theoretical background

We present the theoretical background divided into two subsections. In the first section, we focus on introducing the foundations of deep learning and the type of algorithm implemented. In the second section, we dive from general concepts of AI hardware to the model of the STNO we implement to perform the computation.

2.1 Artificial Intelligence Algorithms

In this section we present an overview on the state of the art in AI algorithms. We go from the basics of machine learning to the description of the type of neural network model we have implemented. First we describe the mathematical background of machine learning, and second, we define the basic deep learning models, their capabilities, and limitations. Finally, we present reservoir computing as energy efficient approach for AI.

2.1.1 Machine Learning Basics

Machine Learning is a sub field of Artificial Intelligence consisting of statistical models that are able to perform a specific task by learning from data, without any sort of explicit programming .

These models can be categorized mainly by the type of learning technique they perform such as Supervised Learning, Unsupervised Learning and Reinforcement Learning [18]. In this work, we refer only to supervised techniques because the model we implement uses this type of learning.

On the one side, supervised learning consists of a model requiring some form of prior instruction. A dataset associated with a respective label (for classification tasks) or target value (for prediction tasks) is needed to train the model. The objective is that after the training, the model can reproduce the correct output values for each input instance. Examples of the supervised approach are classification and prediction algorithms, such examples are found in the reference [19]. On the other side, unsupervised learning consists of a model that does not require a labeled dataset. The model learns and finds patterns on the dataset without any sort of prior instruction or guide. Examples of the unsupervised approach are clustering and dimensional reduction algorithms, such examples are found in the reference [18].

Besides the differences in both models, there are similarities in how they are trained and process the information to perform with acceptable performance their tasks, whether predictions or classifications. A task is generally done by processing input instances or examples, which are composed of a set of numerically measurable features that describe an object or phenomena of interest. These input instances are described by vectors $\boldsymbol{X} \in \mathbb{R}^n$, with components x_i that represents its features, or matrices $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ with components a_{ij} representing its features. Also, a measurable quantity must be defined to evaluate the model performance. The simplest one is accuracy, defined as the proportion of correct predicted outputs over the total number of predictions, and an error measure that quantifies the difference between prediction and target.

In prediction tasks, the model is trained to reproduce a function $f\colon \mathbb{R}^n \to \mathbb{R}$. that maps input features to a numerical output variable of interest, some examples range from simple regression models to advanced forecasting.

Similarly, in classification tasks, the model is trained to map inputs to a set of q labels which initially can be represented by integer numbers $\mathbb{Q} = \{0, 1, 2, 3, \ldots, q\}$. The model is trained to reproduce the function $f\colon \mathbb{R}^n \to \mathbb{Q}$, that generates such mapping. In order to compute the mapping, these labels are encoded through a process called one hot encoding where each individual label is transformed into a vector $\boldsymbol{Y}$ of length q, and each component y_i will correspond to a 1 or a 0 if the subsequent index is aligned with the number assigned to the label. One hot encoding can be expressed as

$$y_i = \begin{cases} 1 & \text{for } i = label \\ 0 & \text{for } i \neq label \end{cases} \tag{1.1}$$

Examples of classification task range from simple image recognition to advanced object detection systems. Fig 1.2 shows a schematic diagram of a supervised learning algorithm for a classification task. In this diagram the model is trained to identity between apples and pineapples, here each label is identified as a one hot vector following equation 1.1, this labeled data is passed to the model as examples. After training, the model is able to identify between apples and pineapples.

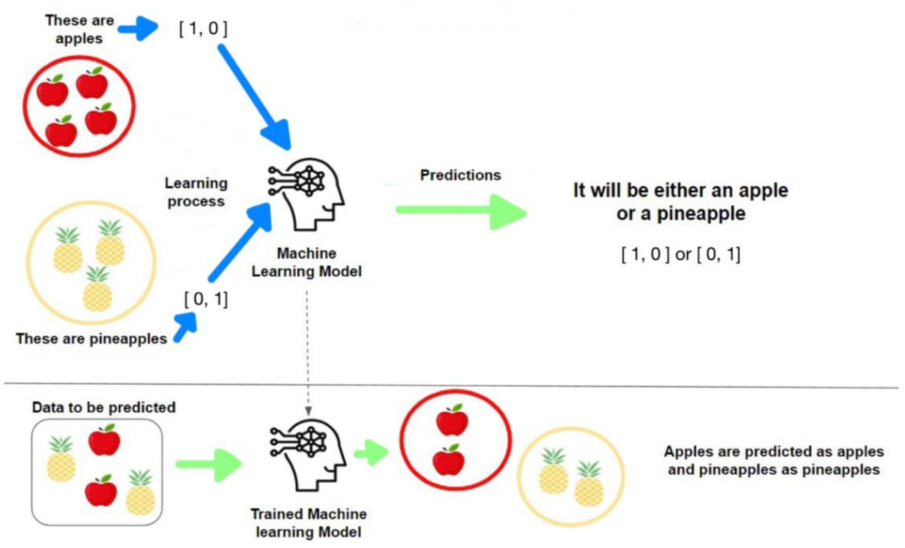


Figure 1.2: Supervised Learning - Clasification Task

To describe the structure and functioning of complex machine learning models, we show the foundational concepts by explaining the two simplest parametric supervised learning models: linear regression [18] and linear classifier (logistic regression)[19].

Linear Regression consist on processing a set of input vectors $\{X_i\}$ whose components x_j represent distintc features that are going to be used to predict a target value $\hat{y_i} \in \mathbb{R}$. By defining a set of optimizable weights w_j the relation between inputs $\{X_i\}$ and predicted outputs y_i is defined as [19]

$$y_i = w_0 + w_1 x_1 + \ldots + w_j x_j \tag{1.2}$$

These weights are optimised and calculated by defining a cost function $\mathcal{L}(w_j)$ that measures the error between the predicted output y_i and the target value $\hat{y_i}$ over the whole dataset

$$\mathcal{L}(w_j) = \frac{1}{m}\sum_{i=1}^{m} L(y_i, \hat{y_i}), \tag{1.3}$$

with m the number of examples or instances in the dataset, and $L(y_i, \hat{y_i})$ the loss function that measures the error between prediction and target of a single instance. There are many types of loss functions adapted to each type of problem; the Mean Square Error (MSE) is commonly used for linear regression

$$L_{\text{MSE}}(w_j) = (\hat{y_i} - y_i)^2. \tag{1.4}$$

The training process consist on finding the optimal value of the weigths w_j that minimizes the cost function; this can be done utilizing different optimiziation algorithms such as gradient descent or cuadratic optimization.

Once training is successful, the cost function is minimized and the model is able to accurately predict target values from unseen input instances.

From this definition, a linear classifier can be defined with some subtle changes. The process is to compute the output by passing the linear combination of the input instance X_i and the weights w_j to an activation function with the range in $[0, 1]$. Such activation function can be, for example, the sigmoid $\sigma(z)$ function defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{1.5}$$

This way the output for each instance is computed as

$$z = w_0 + w_1 x_i + \ldots + w_j x_j, \tag{1.6}$$

$$y = \sigma(z). \tag{1.7}$$

Then in order to identify if the model identifies one class or the other, we define a decision threshold $T \in (0, 1)$ as

$$\text{Output} = \begin{cases} 1 & \text{if } y > T \\ 0 & \text{if } y \leq T \end{cases}, \tag{1.8}$$

where 1 correspond to a given label and 0 correspond to the other one. The output y represents the probability of detecting a label and usually $T = 0.5$, since this is the point where both labels have the same probability, representing a neutral choice without prioritizing one class over the other. From this, we find the optimal weights by defining and minimizing the loss function. For classification task the cross entropy loss is often used [19]

$$H(y, \hat{y}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{i=1}^{q} \hat{y}_i \log(y_i). \tag{1.9}$$

The linear classifier is also called logistic regression. Fig 1.3 (a) shows a diagram of the linear regression and Fig 1.3 (b) the logistic regression.

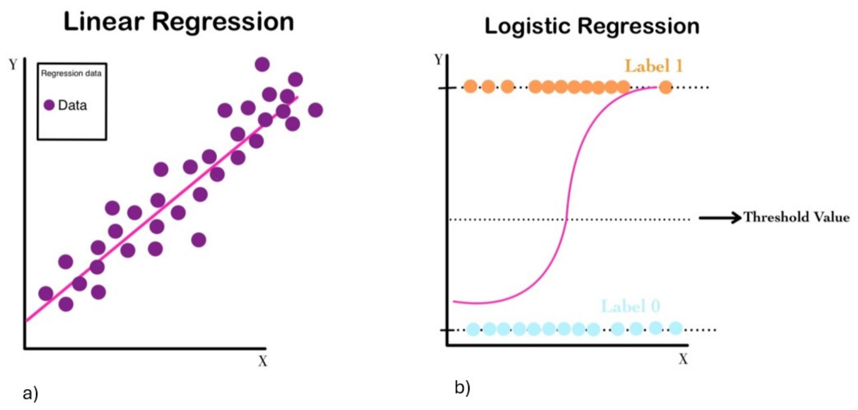


Figure 1.3: (a) Geometric representation of linear regression. (b) Geometric representation of logistic regression. The pink line represents the learned model and the dots represents the targets of the learning process.

Fig. 1.3 (b) illustrates the geometrical and analytical fingerprint underneath the learning process of the linear classifier or logistic regression. Once trained, the linear classifier separates instances by assigning them according to the threshold T as defined in equation 1.8. This aspect can be generalized for linear classifiers that process more complex data sets of instances, allowing us to understand the learning mechanics performed by these machine learning models. In general, the features x_j of a more complex data set $\{X_i\}$ comprehend a feature space in which each example or instance becomes a point. After the training, the linear classifier defines a hyperplane that separates instances corresponding to a given label from the others. A data set that is separated by a hyperplane is called linearly separable data. As shown in Fig 1.4 and Fig. 1.5, for a synthetic dataset of 3 features and 2 classes, a trained linear classifier defines a plane in the feature space that accurately separates instances according to their respective classes.

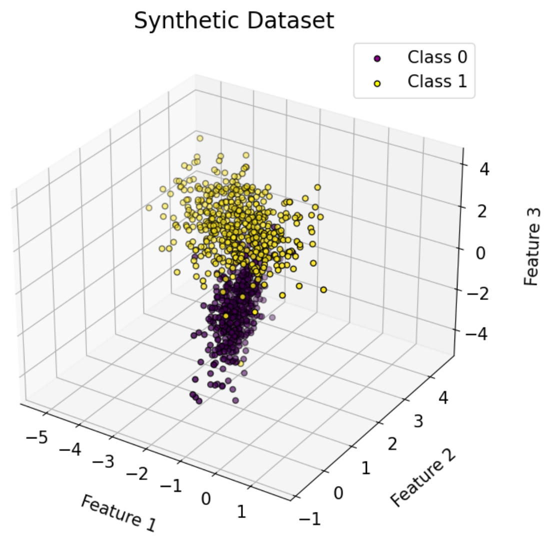


Figure 1.4: Synthetic Dataset of 2 Classes and 3 Features.

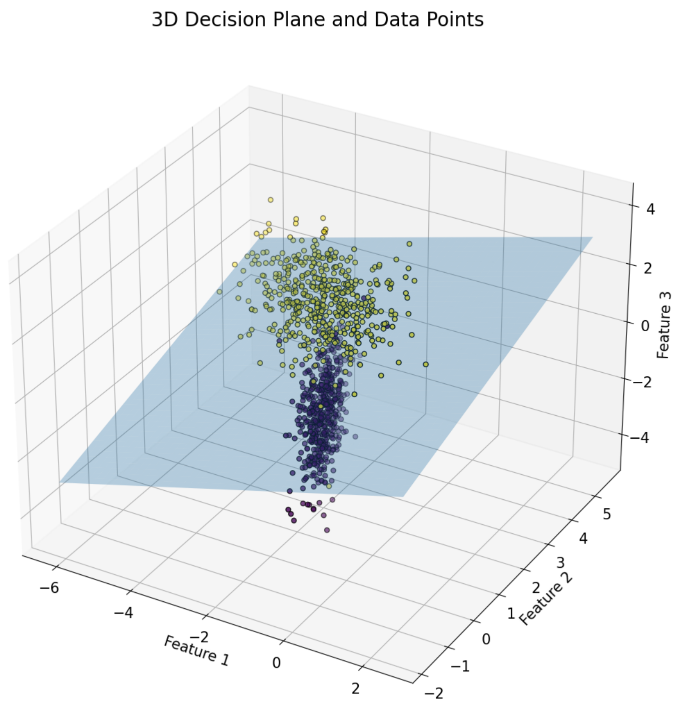


Figure 1.5: Plane defined by a trained linear classifier that accurately distinct between the two classes.

However, most complex datasets are non linearly separable i.e., types of datasets data cannot be there is not such a simple hyperplane that separates them by its labels, as shown in Fig 1.6 [20].

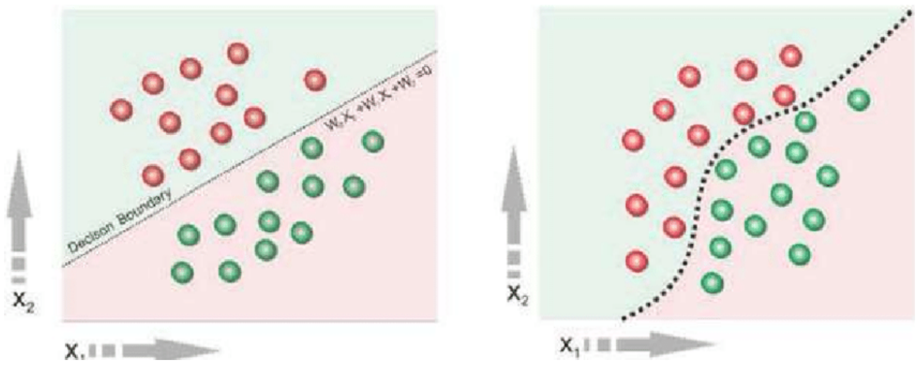


Figure 1.6: Linearly and non linearly separable data.

Indeed, nonlinear separable data can be accurately modeled by nonlinear classifiers. This machine learning models define hyper surfaces that model the correct decision boundary. Deep neural networks are one of the most prominent types of nonlinear classifiers models [?]. Deep neural networks are a set of diverse algorithmic architectures based on the human brain and are able to perform the most complex tasks in artificial intelligence. This models as a wholes constitute the subfield of machine learning called Deep Learning. Fig 1.7 shows a diagram of the subfields of AI.

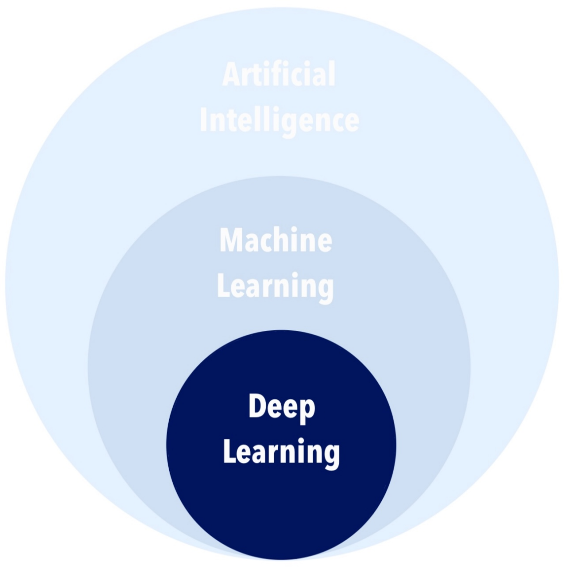


Figure 1.7: Sub fields of AI.

2.1.2 Deep Learning Architectures: from MLP to RNNs

Deep learning models are based on artificial neural networks (ANN) which consist on layers of interconnected artificial neurons where the connections are trained to reproduce a desired output. Fig 1.8 shows a simplified diagram of a ANN.

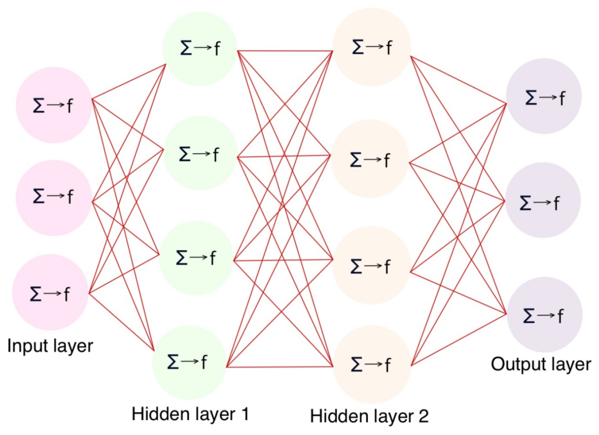


Figure 1.8: Artificial Neural Network

To define an ANN, we will start defining its building block, the so-named artificial neuron, which is also named perceptron. The perceptron is defined as [19]

$$y = f(\mathbf{w} \cdot \mathbf{x} + b), \tag{1.10}$$

where y is the output, the activation function f, being $\mathbf{w}$ as the weight vector containing the weights w_i for each feature, $\mathbf{x}$ as the input vector with features x_i and b as the bias. The weights in this model emulate the connections of the perceptron with other perceptrons. This can be graphically represented as shown in Fig 1.9.

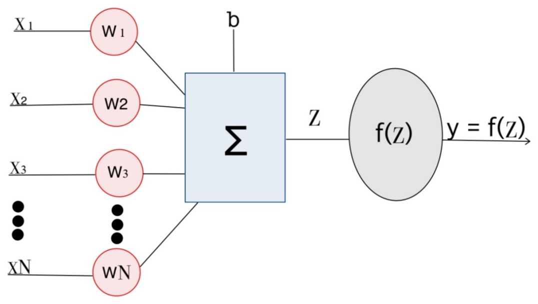


Figure 1.9: Perceptron.

Then, in order to interconnect multiple perceptrons (or neurons) to form a layer, we define a weight matrix $\mathbf{W}$ with components w_{ij} where each column represents the target neuron h_i and each row the input feature x_j, $\mathbf{b}$ now is a bias vector with components b_i that represent the bias of each target neuron, $\mathbf{h}$ is a vector containing the outputs of each neuron also called hidden units. So the data processing operation becomes [18]

$$\mathbf{z} = \mathbb{W}\mathbf{x} + \mathbf{b}, \tag{1.11}$$

$$\mathbf{h} = f(\mathbf{z}). \tag{1.12}$$

Graphically this model is represented as in Fig 1.10 (a) [21]. Now we can stack multiple layers $\mathbf{h}$ (as summarized in Fig. 1.11), and using the same operation developing the notion of depth [18]

$$\mathbf{z^{(l)}} = \mathbb{W}^{(l)}\mathbf{h^{(l-1)}} + \mathbf{b^{(l)}}, \tag{1.13}$$

$$\mathbf{h^{(l)}} = f(\mathbf{z^{(l)}}), \tag{1.14}$$

where l denotes the layer number. As shown in Fig 1.10(b), by stacking multiple layers we develop a deep neural network, this whole model receives the name of Multi Layer Perceptron (MLP).

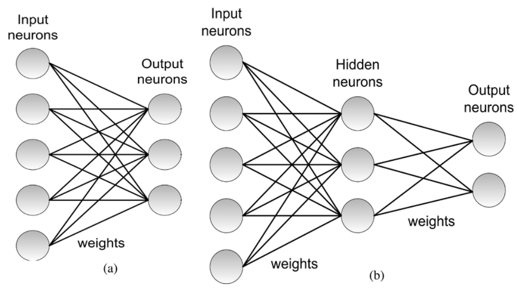


Figure 1.10: (a) Interconected Neurons . (b) Single Layer MLP

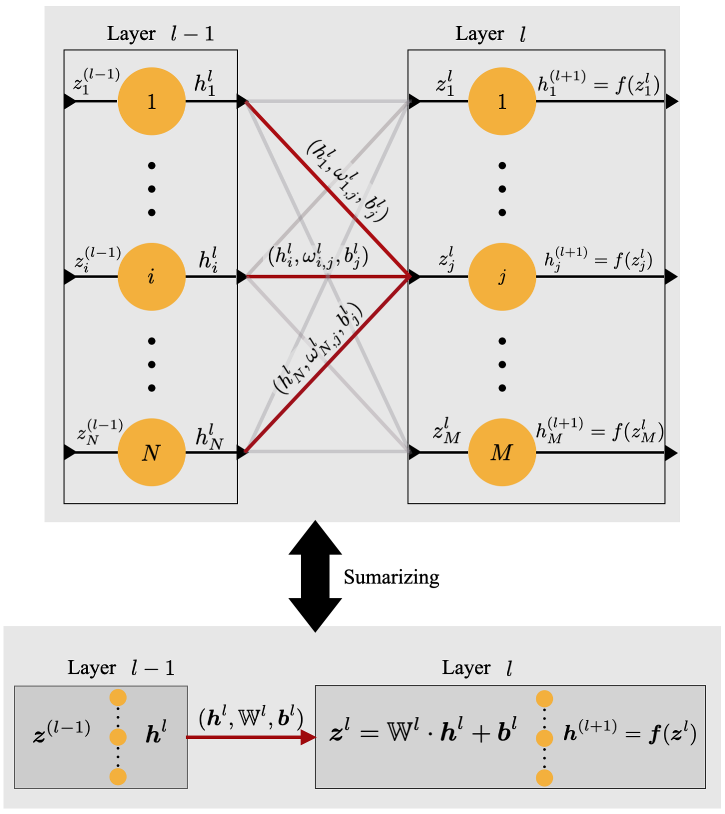

Layer $l-1$
Layer l
$z_1^{(l-1)}$
h_1^l
$z_i^{(l-1)}$
h_i^l
$z_N^{(l-1)}$
h_N^l
$(h_1^l, \omega_{1,j}^l, b_j^l)$
$(h_i^l, \omega_{i,j}^l, b_j^l)$
$(h_N^l, \omega_{N,j}^l, b_j^l)$
z_1^l
$h_1^{(l+1)} = f(z_1^l)$
z_j^l
$h_j^{(l+1)} = f(z_j^l)$
z_M^l
$h_M^{(l+1)} = f(z_M^l)$
Sumarizing
$\boldsymbol{z}^{(l-1)}$
$\boldsymbol{h}^l$
$(\boldsymbol{h}^l, \mathbb{W}^l, \boldsymbol{b}^l)$
$\boldsymbol{z}^l = \mathbb{W}^l \cdot \boldsymbol{h}^l + \boldsymbol{b}^l$
$\boldsymbol{h}^{(l+1)} = \boldsymbol{f}(\boldsymbol{z}^l)$

Figure 1.11: Stack of two consecutive layers, the $(l-1)$'th and the l'th.

In order to train these weights, we define a loss function; for example, for the classification tasks, we use cross-entropy loss. Then, to minimize this loss function, we use the backpropagation algorithm defined as [19]

$$\Delta\mathbb{W}^{(l)} = -\eta \frac{\partial \mathcal{L}}{\partial \mathbb{W}^{(l)}}, \tag{1.15}$$

$$\mathbb{W}^{(l)} = \mathbb{W}^{(l)} + \Delta\mathbb{W}^{(l)}, \tag{1.16}$$

where $\Delta\mathbb{W}^{(l)}$ is the change applied to the weights matrix $\mathbb{W}^{(l)}$ containing the weights, of the layer l , $\mathcal{L}$ is the loss function and η is the learning rate. Now in order to compute the gradient $\frac{\partial \mathcal{L}}{\partial \mathbb{W}^{(l)}}$, the algorithm takes advantage of the chain rule by

$$\frac{\partial \mathcal{L}}{\partial \mathbb{W}^{(l+1)}} = \frac{\partial \mathcal{L}}{\partial h^{(l+1)}} \cdot \frac{\partial h^{(l+1)}}{\partial z^{(l+1)}} \cdot \frac{\partial z^{(l+1)}}{\partial h^{(l)}} \cdot \frac{\partial h^{(l)}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial \mathbb{W}^{(l)}}. \tag{1.17}$$

Running this process for multiple steps allows for a full optimization of the weights minimizing the loss function and allowing the model to successfully perform tasks such as image recognition on diverse complex datasets.

However MLP cannot model temporal data, since it has no connections that match information from past inputs to current inputs. So a new architecture called Recurrent Neural Networks was developed to model or classify time series such as speech. Here the hidden layers have connections to previous inputs. A recurrent layer is defined as [18]

$$a^{(t)} = b + \mathbb{W}h^{(t-1)} + \mathbb{U}x^{(t)}, \tag{1.18}$$
$$h^{(t)} = f(a^{(t)}), \tag{1.19}$$
$$o^{(t)} = c + \mathbb{V}h^{(t)}, \tag{1.20}$$
$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}), \tag{1.21}$$

with $\mathbb{W}$, $\mathbb{U}$ and $\mathbb{V}$ the weights corresponding to hidden-hidden, input-hidden and hidden-output connections, b and c are bias vectors, f and softmax are activation functions for the hidden and output layers respectively, $x^{(t)}$ is the input vector a time step t from the time series and $\hat{y}^{(t)}$ is the output of the hidden node. Softmax is a especial type of the activation function that computes the output probabilities of the model for multiple classes and f can be an arbitrary activation function. Now this configuration can be varied since recurrent neural networks can be defined with or without output nodes at each hidden node or just an output node at the final hidden node in time as shown in Fig1.12 and Fig 1.13 [18]. These recurrent networks are trained with a variation of the backpropagation algorithm called backpropagation through time (BPTT) [22].

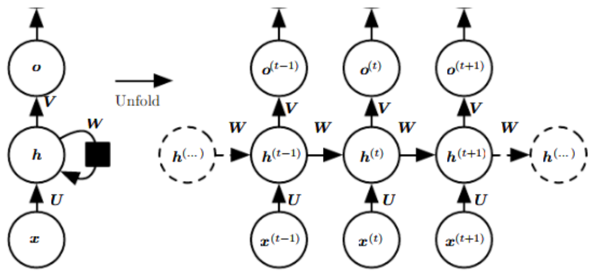


Figure 1.12: Recurrent Neural Network with output nodes at each time step.

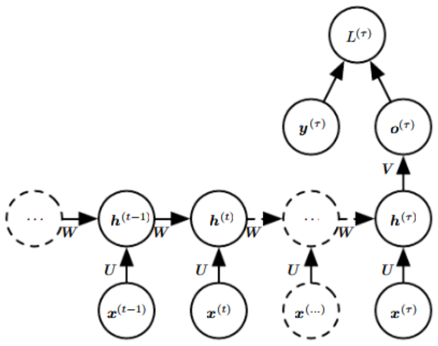


Figure 1.13: Recurrent Neural Network with output nodes at the final time step.

The main inconvenience with (RNNs) is their high computational complexity, high resource demand and high number of trainable parameters. One option to overcome these limitations is to reduce the number of trainable parameters by fixing and randomly initializing the hidden layers and training only the output weights, this paradigm is called reservoir computing or reservoir neural networks.

2.1.3 Reservoir Computing

Reservoir computing or reservoir neural networks are a type of recurrent neural networks (RNN) consisting of an input layer, a reservoir layer and an output layer. Only the output weights are trained whereas the other are randomly initialized and fixed. A reservoir computing neural network is defined as [23]

$$\mathbf{h}(t) = f(\mathbb{W}^{in}\mathbf{x}(t) + \mathbb{W}^{res}\mathbf{h}(t-1)), \tag{1.22}$$

$$\mathbf{y}(t) = \mathbb{W}^{out}\mathbf{h}(t), \tag{1.23}$$

with $\mathbf{h}(t)$ the hidden or reservoir state at time t, $\mathbb{W}^{in}$ the weight matrix from input to reservoir, $\mathbf{x}(t)$ the input time series at time t, $\mathbb{W}^{res}$ the reservoir weights matrix containing the connections between the neurons modeled in the reservoir, $\mathbf{b}$ the bias vector, $\mathbb{W}^{out}$ the weight matrix between reservoir output and $\mathbf{y}(t)$ the output of the reservoir. Fig 1.14 shows a diagram of a reservoir computing architecture [23].

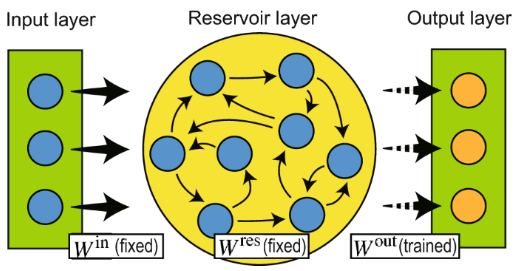


Figure 1.14: Reservoir Computing

The reservoir layer, in essence, behaves as a nonlinear dynamical system that is used to perform the computation. Now this model has to fulfill a condition in order to model a dynamical system that is able to be used for machine learning tasks: the spectral radius (smallest eigenvalue) of the reservoir weight matrix has to be less than 1, this is called the Echo State Property [24].

Since the reservoir layer behaves like a nonlinear dynamical system for computation, multiple proposal of nonlinear physical systems have been implemented to emulate an on hardware neural network such as photonic nano devices, opto-electronic nano devices and spintronic nano oscillators [25]. Theses nano devices have been implemented as neuromorphic computing hardware since they present the nonlinearlity and memory required for such type of computing systems. Most specifically the Echo State property discussed above is equivalent to the dynamical system having short term memory: recent input have impact on the current computation while most past ones do not. With these characteristics nonlinear physical systems can effectively be exploited to perform several machine learning tasks.

2.2 Artificial Intelligence Hardware

In this section, we provide an overview of the description, characteristics, and modeling of the nano-electronic component selected for AI hardware. Initially, We provide an initial introduction to AI accelerators and Neuromorphic Computing, then we continue with a description of the Spin Torque Nano-oscillator as the reservoir computing hardware, and finally, we describe the analytical model that we use for simulating the STNO.

2.2.1 AI Accelerators and Neuromorphic Computing

The Deep learning model's scalability inconveniently also scales with its operational energy consumption. This energetic consumption has increased to unviable levels. AI developers are demanding and looking for the so-called AI Accelerators, which consist of specific purpose computing architectures for handling machine learning tasks with less energy demands and lower latency. Due to their reduced energy consumption, AI accelerators are particularly interested in developing massive AI models. Examples of commercial AI accelerators are Apple Neural Processing Units (NPU), Google Tensor Processing units (TPU), Groq Language Processing units (LPU), and, of course, the well-known NVIDIA's Graphic Processing Units (GPUs).

Despite the success of such technologies for accelerating AI workloads, several experimental technologies and new computing paradigms are in the research stage, intending to bring a revolution in AI computing and energy efficiency. The Von Newman bottleneck and the advent of the end of Moore's law are two main limitations of modern computing systems.

The first limitation mentioned is because modern computing systems are based on the Von Neumann architecture. In such cases, the processing unit (CPU) is separated from the memory (RAM) generating a data transfer between them. This data transfer generates an energy cost and a bottleneck in the computation known as the von Newman bottleneck [26].

The second limitation is related to the end of Moore's law. This statement predicts that the number of transistors in an integrated circuit doubles every two years with minimal cost [27]. However current CMOS technology is expected to hit a wall, since shrinking transistors will eventually reach physical limits [28].

To overcome this limitations several new technologies for novel logic devices based on new nanomaterials are being researched. Examples of such emerging technologies are Photonics [29], Magnonics [30], Spintronics [15], among others.

Neuromorphic computing has emerged over the last years as an alternate paradigm to the Von Newman architecture,it aims to mimic the human brain by merging the memory and processing unit in a single computing node. This configuration reduces the energy needed to transfer information between them. Key components of neuromorphic computing are nonlinearity and memory. AI systems based on this architecture are expected to be more energy efficient. Successful implementations have been reported in recent years. However this technology is at research stage away from mainstream commercial use. Due to the fast development of AI algorithms and its increasing energy consumption, neuromorphic computing has gained a lot of interest over the last few years as an energy-efficient solution. Figure 1.15 shows the differences between the Von Newman architecture and the neuromorphic architecture [31]. Reservoir computing can be implemented using a neuromorphic architecture.

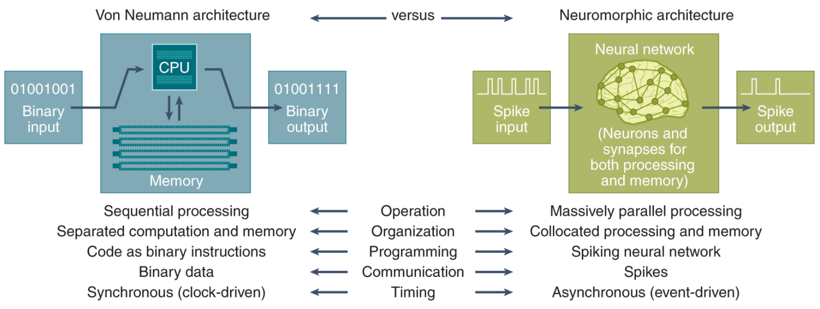

Von Neumann architecture
versus
Neuromorphic architecture
CPU
01001001
Binary input
01001111
Binary output
Memory
Neural network
Spike input
Spike output
(Neurons and synapses for both processing and memory)
Sequential processing
Separated computation and memory
Code as binary instructions
Binary data
Synchronous (clock-driven)
Operation
Organization
Programming
Communication
Timing
Massively parallel processing
Collocated processing and memory
Spiking neural network
Spikes
Asynchronous (event-driven)

Figure 1.15: Von Newnman Architecture vs Neuromorphic Architecture

2.2.2 Spintronics Hardware and Spin Torque Nano Oscillator

Spintronics is an electronics paradigm where the devices use the spin of the electrons to process, store and transfer information. Over the recent years spintronics systems have been proposed for developing new computing architecture for energy efficient AI systems [14]. Some spintronic devices are based on two main effects: magnetoresistance and spin transfer torque.

Magnetoresistance is the effect when the resistance of a material changes upon the application of a magnetic field. In magnetic multi-layers several types of magneto-resistance effects are observed, such as Giant Magneto Resistance or Tunnel Magneto Resistance [15].

Spin transfer torque is a quantum effect where a spin-polarized of an electric current (there is a majority of electrons with a preferred spin orientation[32]) transfers angular momentum to the atoms of a magnetic material, generating a net torque on the magnetization either reversing it or inducing a sustained precession.[32].

Several types of magnetic multi-layer structures can be used to design spintronic devices. Here we refeer only to the magnetic tunnel junction (MTJ) which is a nanodevice consisting of two ferromagnetic layers separated by an insulating layer. One ferromagnetic layer is called the fixed or pinned layer, has a strong saturation magnetization, and its role is polarizing the electrons spin's of the injected electric curent to the MTJ. The other ferromagnetic layer is called the free layer and has a lower saturation magnetization. Therefore, when a current is injected into the pinned layer it becomes spin-polarized. Such spin-polarized current interacts with the the free layer and exert a torque on the magnetization. If the current induces magnetization reversal the MTJ can be used to encode binary information based on the relative orientation of the ferromagnetic layers: if they are parallel the MTJ reproduces a 1 and if they are antiparalell the MTJ reproduces a 0 (Fig 1.16 shows a diagram of this configuration). Now if the spin-polarized current induces a sustained precession on the magnetization the MTJ generates voltage oscillations though magneto-resistance. These oscillations are in the range of tens of MHz to GHz. So the MTJ operates as microwave oscillator called the Spin Torque Nano Oscillator (STNO). Fig 1.17 (a) shows a driagram of the STNO and Fig 1.17(b) shows the voltage oscillations generated [3]. This nanoscale oscillator has properties that can be exploited for designing novel energy-efficient AI architectures, because its dynamics intrinsically presents the nonlinearity and memory required for neuromorphic computing at lower orders of magnitude of operational energy costs. In recent years, several experimental implementations of STNO-based machine learning systems have been reported [17].

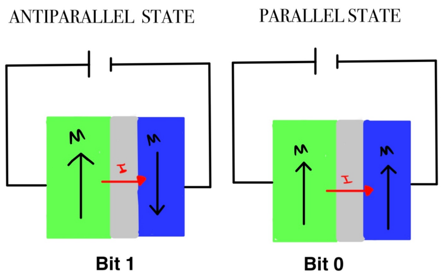


Figure 1.16: MTJ and Magnetization Reversal.

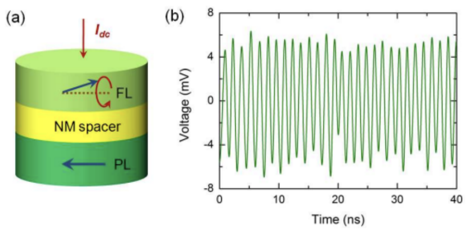


Figure 1.17: (a) Diagram of a STNO. (b) Voltage Oscilationes generated by the STNO [3].

2.2.3 Modeling of Spin Torque Nano Oscillator (STNO)

In this section, we analytically model, under the micromagnetism theory, the operation of a STNO under Direct Current (DC) injection. This model will be then used for performing simulations of the AI hardware and studying its properties.

2.2.3.1 Landau Lifshitz Gilbert Equation

The dynamics of the magnetization $\boldsymbol{M}(\boldsymbol{r}, t)$ of the free layer under the effects of spin polarized current is described by the Landau-Lifshitz-Gilbert equation with an aditional Slonezewski-Berger spin transfer term [33]

$$\frac{\partial \boldsymbol{M}}{\partial t} = \gamma \left[\boldsymbol{H}_{\text{eff}} \times \boldsymbol{M}\right] + \boldsymbol{T}_{\text{G}} + \boldsymbol{T}_{\text{S}}. \tag{1.24}$$

This equation essentially defines the magnetic dynamics of the sample as being influenced by three main terms:

Firstly, the conservative torque term $\gamma \left[\boldsymbol{H}_{\text{eff}} \times \boldsymbol{M}\right]$ that describes the conservative precession of the magnetization vector $\boldsymbol{M}$ around the effective magnetic field $\boldsymbol{H}_{\text{eff}}$. The modulus of the gyromagnetic ratio γ is given by $\gamma = \frac{g\mu_B}{\hbar}$, where g is the spectropic landé factor, μ_B is the Bohr magneton and $\hbar$ is the reduced Planck constant. In most magnetic systems the main contributions of the effective field are the aplied field $\boldsymbol{H}_0$, the magneto dipolar field $\boldsymbol{H}_{dip}$, the anisotropy field $\boldsymbol{H}_A$, the exchange field $\boldsymbol{H}_{ex}$ and the Oersted Magnetic Field $\boldsymbol{H}_I$ created by the induced current I.

Secondly the damping torque $\boldsymbol{T}_\mathrm{G}$ accounts for the dissipative processes in magnetic systems, wich is responsable for carryng the dynamic behavior to an equilibrium state The Gilbert damping torque is given by

$$\boldsymbol{T}_\mathrm{G} = \frac{\alpha(\xi)}{M_0}\left[\mathbf{M} \times \frac{\partial \mathbf{M}}{\partial t}\right], \tag{1.25}$$

where

$$\xi = \frac{\left|\frac{\partial \mathbf{M}}{\partial t}\right|^2}{\omega_M^2 M_0^2}. \tag{1.26}$$

With M_0 the saturation magnetization at zero temperature of the free layer and $\omega_M = 4\pi\gamma M_0$.

For finite and small angles of magnetic precession the function $\alpha(\xi)$ can be expanded in a Taylor series

$$\alpha(\xi) \simeq \alpha_G(1 + q_1\xi + q_2\xi^2 + \ldots). \tag{1.27}$$

With α_G the Gilbert damping constant and q_i the nonlinear damping parameters. For most of the magnetic materials, it is sufficient to consider

$$\alpha(\xi) \simeq \alpha_G. \tag{1.28}$$

And finally, the Spin Transfer Torque $\boldsymbol{T_S}$ which is given by

$$\boldsymbol{T_S} = \frac{\sigma_0 I f(\mathbf{r})}{M_0}\left[\mathbf{M} \times [\mathbf{M} \times \hat{\mathbf{e}}_p]\right], \tag{1.29}$$

where σ_0 is a coefficient that account for spin polarization parameters, I is the current flowing through the free layer, $f(\mathbf{r})$ is the spatial distribution of the current and $\mathbf{r}$ is the unit vector in the direction of spin polarization. The coefficient σ_0 is given by

$$\sigma_0 = \frac{\epsilon g \mu_B}{2eM_0LS}. \tag{1.30}$$

The spin polarization efficiency ϵ is asumed to be constant for most cases, e is the electron charge, L is the thickness of the free layer and S is the area of the region where current flows.

2.2.3.2 Auto-oscillator theory of STNO

It has been shown in Ref [33] that the magnetization dynamics of the free layer of an STNO can describe auto-oscillations. In this section, we follow such derivation in order to describe the oscillation power and its dynamic amplitude to simulate and study the AI hardware.

Lets consider a STNO based with an out-of-plane homogeneous magnetization. Let assume that a spin-polarized current is uniformly distributed across the area of the free layer. The effective magnetic field $\mathbf{H}_\text{eff}$ is then reduced to the contributions of the applied field $\mathbf{H_0} = H_0\hat{\mathbf{e}}_z$ and the demagnetizing field $\boldsymbol{H}_{dip} = -4\pi M_z\hat{\mathbf{e}}_z$ (see Fig 1.18). The static effective field is then expressed as

$$\mathbf{H}_\text{eff} = (H_0 - 4\pi M_z)\,\hat{\mathbf{e}}_z. \tag{1.31}$$

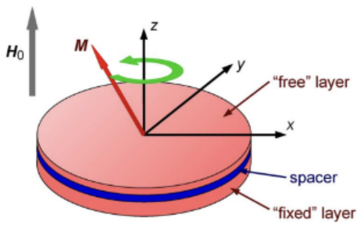


Figure 1.18: STNO

and the magnetization at equilibrium is given by

$$\mathbf{M_{eq}} = M_0\hat{\mathbf{e}}_z. \tag{1.32}$$

Now, the dynamic oscillations of the STNO will be studied by considering a small perturbation around such equilibrium state, therefore, the magnetization $\mathbf{M}$ is then given by

$$\mathbf{M} = \mathbf{M_{eq}} + \mathbf{M}_p, \tag{1.33}$$

where we are assuming perturbations i.e., $\mathbf{M}_p << \mathbf{M_{eq}}$. So the z component of the magnetization M_z can be taken as constant and expressed as

$$M_z = M_0. \tag{1.34}$$

The in-plane magnetic components M_x and M_y of the magnetization are the perturbative components and are described by the following differential equations

$$\left(\frac{dM_x}{dt}\right)_{\text{cons}} = -\omega_0 M_y, \quad \left(\frac{dM_y}{dt}\right)_{\text{cons}} = +\omega_0 M_x, \tag{1.35}$$

$$\omega_0 = \gamma(H_0 - 4\pi M_0), \tag{1.36}$$

where ω_0 is called the ferromagnetic resonance frequency. This equations can be written as one complex differential equation

$$\left(\frac{d\tilde{c}}{dt}\right)_{\text{cons}} = -i\omega_0\tilde{c}, \tag{1.37}$$

where

$$\tilde{c} = \frac{M_x - iM_y}{2M_0}, \tag{1.38}$$

which is known as the complex spin wave magnetic variable.

Now for studying the nonlinear regime of oscillations we redefine the complex spin wave amplitude as

$$c = \frac{M_x - iM_y}{\sqrt{2M_0(M_0 + M_z)}}, \tag{1.39}$$

which is useful to redefine the magnetization vector as

$$\mathbf{M} = M_0\left(1 - 2|c|^2\right)\hat{\mathbf{e}}_z + M_0\sqrt{1 - |c|^2}\left[(\hat{\mathbf{e}}_x + i\hat{\mathbf{e}}_y)c + (\hat{\mathbf{e}}_x - i\hat{\mathbf{e}}_y)c^*\right]. \tag{1.40}$$

Inserting this expression on the Conservative Landau-Lifshitz equation (i.e., the LLG equation without the Gilbert torque and spin-transfer torque contributions), we derive the following expressions

$$\left(\frac{dc}{dt}\right)_{\text{cons}} = -i\omega\left(|c|^2\right)c, \tag{1.41}$$

$$\omega\left(|c|^2\right) = \omega_0 + N|c|^2, \tag{1.42}$$

$$N = 2\omega_M, \tag{1.43}$$

$$\omega_M = 4\pi\gamma M_0, \tag{1.44}$$

where $\omega\left(|c|^2\right)$ is the nonlinear frequency, N is the nonlinear frequency shift and $p = |c|^2$ is the auto-oscilation power. The general solution of this differential equation is

$$c(t) = \sqrt{p}\exp\left[-i\omega(p)t + i\phi_0\right], \tag{1.45}$$

with p the auto-oscillation power and ϕ_0 an arbitrary phase of oscillation.

Now, lets consider the corrections to the equations of the system by introducing the Gilbert (damping) torque and the spin-transfer torque. For the Gilbert term, we assume that the damping constant is equivalent to the Gilbert Damping i.e ($\alpha(\xi) = \alpha_G = \text{const}$), with this we obtain the damping torque in the complex variable description

$$\left(\frac{dc}{dt}\right)_{\text{damp}} = -\Gamma_+\left(|c|^2\right)c, \tag{1.46}$$

$$\Gamma_+\left(|c|^2\right) = \Gamma_G\left(1 + Q|c|^2 + Q'|c|^4\right), \tag{1.47}$$

where

$$\Gamma_G = \alpha_G\omega_0, \quad Q = \frac{2\omega_M}{\omega_0} - 1, \quad Q' = -\frac{2\omega_M}{\omega_0}, \tag{1.48}$$

where Γ_+ is the positive damping rate.

For the spin torque correction, we get

$$\left(\frac{dc}{dt}\right)_{\text{spin}} = +\Gamma_-\left(|c|^2\right)c, \tag{1.49}$$

$$\Gamma_-\left(|c|^2\right) = \sigma_0 I\left(1 - |c|^2\right), \tag{1.50}$$

where Γ_- is the negative damping rate.

Collecting all the contributions, we finally get the proceeding differential equation

$$\frac{dc}{dt} = \left(\frac{dc}{dt}\right)_{\text{cons}} + \left(\frac{dc}{dt}\right)_{\text{damp}} + \left(\frac{dc}{dt}\right)_{\text{spin}}, \tag{1.51}$$

$$\frac{dc}{dt} + i\omega\left(|c|^2\right)c + \Gamma_+\left(|c|^2\right)c - \Gamma_-\left(|c|^2\right)c = 0. \tag{1.52}$$

Equation 1.52 is the universal model for auto-oscillations [33]. Auto-oscillations are generated and sustained oscillations in a nonlinear system by means of a non periodic energy source, where the complex amplitude $c(t)$ describes the dynamics of the oscillation and is directly related to the power $p = |c|^2$ and phase $\phi = arg(c)$.

The resonance frequency of oscillations $\omega(p)$, the positive damping rate $\Gamma_+(p)$ and the negative damping rate $\Gamma_-(p)$ define the evolution of the system. These terms are usually nonlinear functions of the oscillation power p.

The aforementioned approach of the STNO auto-oscillations is valid if the following conditions are fullfiled:

1. Only one mode of oscillation is exited.
2. The possible range of variations of amplitude during an oscillation period is small.
3. Self sustained oscillations start with zero amplitude and smoothly increase with the increase of a control parameter.

In most cases, the physical regime of the STNO auto-oscillation employed for IA is restricted to weak oscillations amplitude, which in terms of our STNO model means ($p << 1$). In such a regime, the mathematical complexity of the analytical model can be reduced by linearizing the frequency and damping torques on p. This linearization looks like

$$\omega(p) \approx \omega_0 + Np, \tag{1.53}$$

$$\Gamma_+(p) \approx \Gamma_G(1 + Qp), \tag{1.54}$$

$$\Gamma_-(p) \approx \sigma I(1 - p). \tag{1.55}$$

This linear regime ($p << 1$) will be taken next for describing the STNO auto-oscillations in two complementary scenarios that have been used as the physical building blocks for AI accelerators

2.2.3.3 Autonomous dynamics of STNO with Uniform State

In this section, we present the analytical description of the sustained dynamic oscillations of the free layer of the STNO when it is in an equilibrium magnetization state [33]. An analogous procedure will also be applied in the following subsection where the free layer is assumed in a vortex equilibrium state.

The complex differential equation 1.52 can be rewritten as as system of two real differential equations for the power and phase of oscillations

$$\frac{dp}{dt} = -2\left[\Gamma_+(p) - \Gamma_-(p)\right]p, \tag{1.56}$$

$$\frac{d\phi}{dt} = -\omega(p), \tag{1.57}$$

which has two stationary solutions ($\frac{dp}{dt} = 0$). The first one correspond to the trivial solution where $p = 0$. Evaluating equation (1.56) around $p \to 0$ we get

$$\frac{dp}{dt} = -2\left[\Gamma_+(0) - \Gamma_-(0)\right]p. \tag{1.58}$$

The zero power solution is physically realizable when $\Gamma_+(0) > \Gamma_-(0)$. Thus from the condition $\Gamma_+(0) = \Gamma_-(0)$ we can define the threshold current I_{th} for auto oscillations to occur. Using equation 1.54 and equation 1.55 we get

$$I_{\text{th}} = \frac{\Gamma_G}{\sigma}. \tag{1.59}$$

The second stationary solution correspond to $\Gamma_+(p_0) = \Gamma_-(p_0)$. This represent the state of the system where the energy losses due to the natural disipation (Damping torque) are exacty compensated by the external energy source (Spin Transfer Torque). Using equations 1.54 and 1.55 we can obtain the expression for the stationary oscillation power p_0

$$p_0 = \frac{\zeta - 1}{\zeta + Q}, \tag{1.60}$$

$$\zeta = \frac{I}{I_{\text{th}}}. \tag{1.61}$$

Now under small power deviations $\delta p = p - p_0 << 1$ we can consider $p \approx p_0$. Expanding a taylor series around $p = p_o$ the damping terms in equation 1.56 become

$$\Gamma_+(p) = \Gamma_+(p_0) + (p - p_0)\left.\frac{d\Gamma_+(p)}{dp}\right|_{p_0}, \tag{1.62}$$

$$\Gamma_-(p) = \Gamma_-(p_0) + (p - p_0)\left.\frac{d\Gamma_-(p)}{dp}\right|_{p_0}. \tag{1.63}$$

Inserting this values in the oscillation power differential equation (Eq. 1.56) we have

$$\frac{d\delta p}{dt} = -2\left[\Gamma_+(p_0) + (p - p_0)\left.\frac{d\Gamma_+(p)}{dp}\right|_{p_0} - \left(\Gamma_-(p_0) + (p - p_0)\left.\frac{d\Gamma_-(p)}{dp}\right|_{p_0}\right)\right]p_0. \tag{1.64}$$

Reducing this expression we get:

$$\frac{d\delta p}{dt} = -2\left[\left.\frac{d\Gamma_+(p)}{dp}\right|_{p_0} - \left.\frac{d\Gamma_-(p)}{dp}\right|_{p_0}\right](\delta p)p_0, \tag{1.65}$$

where the derivatives are evaluated for the weakly nonlinear expression of the damping rates

$$\left.\frac{d\Gamma_+(p)}{dp}\right|_{p_0} = \Gamma_G Q p, \tag{1.66}$$

$$\left.\frac{d\Gamma_+(p)}{dp}\right|_{p_0} = \sigma I = \zeta\Gamma_G, \tag{1.67}$$

with this we get

$$\frac{d\delta p}{dt} = -2\left[\Gamma_G Q p - \zeta\Gamma_G\right](\delta p)p_0, \tag{1.68}$$

rearranging this equation and expanding δp we get

$$\frac{d\delta p}{dt} = \frac{dp}{dt} - \frac{dp_0}{dt} = \frac{dp}{dt} = 2\left[\zeta - 1\right]\Gamma_G(p_0 - p). \tag{1.69}$$

Now, we define the relaxation time τ as

$$\tau = \frac{1}{2\left[\zeta - 1\right]\Gamma_G}, \tag{1.70}$$

which we use in equation 1.69 obtaining the following expression

$$\frac{dp}{dt} = \frac{(p_0 - p)}{\tau}. \tag{1.71}$$

This equation describes the power evolution between stationary states under small power deviations as an exponential evolution toward the stationary state p_0. A solution of this equation has the form of

$$p(t) = p_o(1 - e^{-\frac{t}{\tau}}) + p_i e^{-\frac{t}{\tau}}, \tag{1.72}$$

where p_i is the initial stable power which evolves towards the next stable power p_0. This description provides insight into the applicability of the STNO dynamics for neuromorphic computing and AI accelerators since the system showcases properties such as nonlinearity (nonlinear relation between p_o and I) and memory (exponential decay of the initial stable state) that can be exploited for low energy computation.

Under this model the parameters of interest that affect the functioning of the STNO are the applied field H_0, the saturation magnetization of the free layer M_s, the spin transfer torque efficiency ϵ, the thickness of the free layer L and the area of injected current S. The a-dimensional power p is proportional to the experimentally measured power P.

2.2.3.4 Autonomous Dynamics of Vortex Based STNO

In this section, we present the analytical description of the sustained dynamic oscillations of the free layer of the STNO when it is in a vortex magnetization state [34], in which we derive the measured voltage dynamics generated by the auto-oscillations.

The magnetization dynamics of the free layer of a vortex based STNO can be modeled with use of the dynamics of the vortex core. The vortex core dynamics induced by spin transfer torque is described by the evolution of its orbit radius and phase. The dynamics of the frequency and amplitude can be modeled through the polar coordinates of the vortex core $s(t)$ and $\theta(t)$ in the normalized disk [35] [34]. Fig 1.19 (a) [36] shows a diagram of the vortex magnetization state and Fig 1.19 (b) shows a diagram of the vortex core polar coordinates.

The dynamics of the vortex core polar coordinates in the normalized disk are described by the following differential equations

$$\frac{d\theta}{dt} = \frac{\kappa}{G}(1 + \zeta s^2), \tag{1.73}$$

$$\frac{ds}{dt} = \frac{D\kappa}{G^2} s \left[\frac{a_j I G}{D\kappa\pi R^2} - 1 + (\zeta + \xi)s^2\right], \tag{1.74}$$

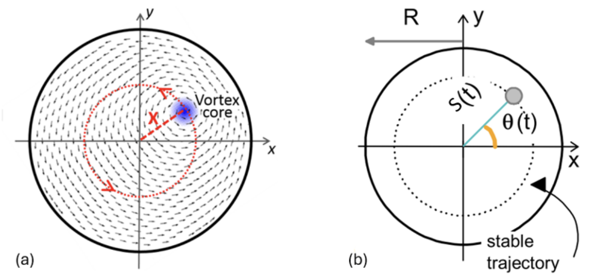


Figure 1.19: (a) Vortex Magnetization. (b) Polar Coordinates of the vortex core.

with R the radius of the ferromagnetic disk, G the gyrovector magnitude, $D(1+\xi s^2)$ the damping coefficients with D its linear part and ι its nonlinearlity factor, a_j the spin transfer torque efficiency, I the current, and $\kappa(1+\zeta s^2)$ the confinement stiffness with κ its linear part and ζ its nonlinearity factor.

The confinement stiffness is given by

$$\kappa = \kappa_{ms} + \frac{\kappa_{oe} I}{\pi R^2}. \tag{1.75}$$

The confinment stiffnes nonlinearlity factor ζ is given by

$$\zeta = \frac{\left(k'_{ms} + k'_{oe}\frac{I}{\pi R^2}\right)}{\left(k_{ms} + k_{oe}\frac{I}{\pi R^2}\right)}. \tag{1.76}$$

The angle θ between the free layer magnetization M_S^{free} and the perpendicular applied field $H_\perp$.

$$\theta_0 = \arccos\left(\frac{H_\perp}{4\pi M_S^{\text{free}}}\right). \tag{1.77}$$

The parameters defining the confinement stiffness are given by.

$$\kappa_{ms} = \frac{10}{9}\mu_0 (M_S^{\text{free}})^2 \frac{L^2}{R}\sin^2\theta_0, \tag{1.78}$$

$$\kappa'_{ms} = 0.25\kappa_{ms}, \tag{1.79}$$

$$\kappa_{oe} = 0.85 C\mu_0 M_S^{\text{free}} LR\sin\theta_0, \tag{1.80}$$

$$\kappa'_{oe} = -0.42 C\mu_0 M_S^{\text{free}} LR\sin\theta_0. \tag{1.81}$$

The gyro-vector norm is given by

$$G = 2\pi\frac{LM_S^{\text{free}}}{\gamma}(1-\cos\theta_0). \tag{1.82}$$

The damping coefficient is given by

$$D = 2\pi\frac{\alpha\eta LM_S^{\text{free}}}{\gamma}, \tag{1.83}$$

with

$$\eta = \left(\ln(R/2b) - \frac{1}{4}\right)\sin^2\theta_0. \tag{1.84}$$

Finally, the spin transfer torque efficiency is given by

$$a_j = \pi\frac{\hbar P_{\text{spin}}}{2|e|}p_z\sin^2\theta_0, \tag{1.85}$$

where

$$p_z = \frac{H_\perp}{4\pi M_S^{\text{SAF}}}. \tag{1.86}$$

Here μ_0 is the vacuum permeability, b is the radius of the vortex nucleus, P_{spin} Spin polarization in the hard magnetic layer, C is the vortex chirality, M_S^{SAF} is the magnetic saturation at the hard magnetic layer.

This model is equivalent to the auto oscillation equation if $c(t) = s(t)e^{-i\theta(t)}$. With this the stable regime of nonlinear oscillation can be analyzed. By identification the damping rates related to the vortex dynamics are deduced

$$\Gamma_+(p) = \frac{Dk}{G^2}\left[1 + (\zeta + \xi)p\right] \tag{1.87}$$

$$\Gamma_-(p) = \frac{a_j I}{G\pi R^2}. \tag{1.88}$$

The stable solution with p_0 where $\Gamma_+(p_0) = \Gamma_-(p_0)$ is a perfect circular trajectory defined as

$$p_0 = s_0^2 = \frac{\frac{a_j IG}{Dk\pi R^2} - 1}{\zeta + \xi}. \tag{1.89}$$

Using the same approach as in the dynamics of auto-oscillation power p [33] (Condition $\Gamma_+(0) = \Gamma_-(0)$), we derive the threshold current I_{th} of this model as

$$I_{th} = \frac{Dk\pi R^2}{a_j G}. \tag{1.90}$$

Now, in order to get an expression for measure amplitude voltage under stable orbits the STNO is considered as an oscillating resistance $R_s(t)$ where

$$R_s(t) = R_0 + \Delta R(t), \tag{1.91}$$

with R_0 the resistance mean value and $\Delta R(t)$ the oscillating part. The STNO is equivalent to the resistance R_0 in series with a voltage generator $e(t) = \Delta R(t)I$ (with I the injected dc current). The generator voltage dependence on the vortex core position leads to the following expressions

$$V_0 = \frac{R_{\text{load}}}{R_{\text{load}} + R_0}\lambda s_0, \tag{1.92}$$

with

$$\lambda = \frac{I\Delta R_{p-ap}(I)}{2}\beta\sqrt{\left[1 - \left(\frac{H_\perp}{4\pi M_S^{\text{SAF}}}\right)^2\right]\left[1 - \left(\frac{H_\perp}{4\pi M_S^{\text{free}}}\right)^2\right]}, \tag{1.93}$$

λ is a magnetoresitive factor, $H_\perp$ the perpendicular magnetic field $H_\perp$, I the electrical current, $\Delta R_{p-ap}(I)$ the difference between anti parallel state and the parallel state at a given I, M_s^{free} the free layer saturation magnetization, M_s^{SAF} the SAF layer saturation magnetization, $\beta = \frac{2}{3}$ the conversion factor of the vortex core displacement into magnetization change, valid for displacements up to 60% of the dot radius, R_{load} the resistance input of the measurement device. This equations describe the relation of the measured voltage amplitude and the vortex core normalized radius.

Now since the normalized radius is related to the a-dimensional oscillation power p as

$$p(t) = s(t)^2. \tag{1.94}$$

Or equivalently

$$s(t) = \sqrt{p(t)}, \tag{1.95}$$

thus, we can derive an expression for the dynamics of the orbit radius $s(t)$ under small auto oscillation power deviations.

Considering equations 1.87, 1.88 and 1.94 we get

$$\frac{dp(t)}{dt} = 2s(t)\frac{ds(t)}{dt} \tag{1.96}$$

$$\frac{d\Gamma_+(p)}{dp} = \frac{Dk}{G^2}(\zeta + \xi), \tag{1.97}$$

$$\frac{d\Gamma_+(p)}{dp} = 0. \tag{1.98}$$

Inserting this into equation 1.65 we derive the differential equation describing the dynamics of s(t) under small auto-oscillation power deviations

$$\frac{ds}{dt} = \frac{Dk}{G^2}(\zeta + \xi)(s_0 - s). \tag{1.99}$$

Or equivalently

$$\frac{ds}{dt} = \frac{(s_0 - s)}{\tau}, \tag{1.100}$$

with the relaxation time in this model expressed as

$$\tau = \frac{G^2}{(\zeta + \xi)Dk}. \tag{1.101}$$

The solution of this differential equation is in the form of

$$s(t) = s_o(1 - e^{-\frac{t}{\tau}}) + s_i e^{-\frac{t}{\tau}}, \tag{1.102}$$

with s_i the initial orbit radius from which the system evolves towards s_0.

Now considering the relation of the orbit radius and the measured voltage we get an expression of the dynamics of the measured voltage amplitude for small auto-oscillation power deviations

$$V_0(t) = \frac{R_{\text{load}}}{R_{\text{load}} + R_0}\lambda\sqrt{\frac{\frac{I}{I_{th}} - 1}{\zeta + \xi}}(1 - e^{-\frac{t}{\tau}}) + V_i e^{-\frac{t}{\tau}}, \tag{1.103}$$

with V_i the initial voltage amplitude from which the system evolves towards V_0

$$V_i = \frac{R_{\text{load}}}{R_{\text{load}} + R_0}\lambda s_i. \tag{1.104}$$

Here the parameters that define the dynamical state of the model are the resistance of the measurement device R_{load}, the mean resistance of the sample R_0, the resistance between parallel and antiparalell states $\Delta R_{p-ap}(I)$, the vortex core displacement β, the radius of the ferromagnetic disk R, the nonlinearity factor of the damping coefficient ξ, the perpendicular magnetic field $H_{\perp}$, the thickens of the free layer L , the saturation magnetization of the free layer M_S^{free},the vortex core radius b , the vortex chirality C, the spin polarization $P_{\mathbf{spin}}$ and the saturation of the hard magnetic layer M_S^{SAF}, the last two define the spin torque efficiency.

II Methods

1 Speech Recognition with Spin Torque Nano Oscilator (STNO)

In this section, we describe and detail the AI system we implement using the STNO (Simulated) as hardware component. We train a machine learning model for Speech Recognition using the STNO voltage dynamics for emulating the neural network directly on the hardware circuit. Specifically we apply a reservoir computing approach where the hardware circuit implements a neuromorphic architecture. The dataset we train the model on is the Free Spoken Digit Dataset [1], which consist of 3000 recordings of spoken digits ranging from 0 to 9 from 6 different speakers recorded at 8 kHz.

The computing procedure we implement consist of 3 main stages as described in Fig 2.1. First, we clean the data and pre-process it with standard speech recognition techniques (fig 2.1 (1)). Secondly, we encode the pre-processed data as input signal and inject it into the hardware circuit that emulates the neural network then we sample the output signal generated by the STNO with a defined time step (fig 2.1 (2). And finally, We read out the sampled output signal by passing it to an fully connected neural network output layer (fig 2.1 (3)), which we train with a simple learning algorithm that finds the closed solution for minimizing the Mean Square Error (MSE) loss function [37]. The simulations are performed using a Asus Tuf Dash F15 with a Intel i7 11th gen (4 cores).

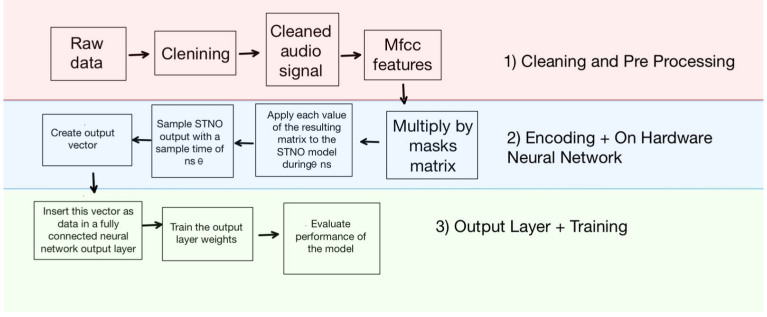

Raw data
Clenining
Cleaned audio signal
Mfcc features
1) Cleaning and Pre Processing
Multiply by masks matrix
Apply each value of the resulting matrix to the STNO model duringθ ns
Sample STNO output with a sample time of ns θ
Create output vector
2) Encoding + On Hardware Neural Network
Insert this vector as data in a fully connected neural network output layer
Train the output layer weights
Evaluate performance of the model
3) Output Layer + Training

Figure 2.1: Diagram of the AI system.

.

We present this section as as follow: first, we describe the data cleaning process; then, we describe the data preprocessing and encoding; following, we describe the hardware model implemented for emulating the neural network; and finally, we describe the output layer and the training algorithm.

1.1 Data Cleaning

We apply outliers analysis and cleaning on the length of the recordings in order to avoid possible biases in the model training. The process consists of analyzing the distribution of the length of each audio recording on the dataset and calculating the z score (Number of standard deviations a data point is away from the mean value of the dataset). From this any data with a z-score higher than 2 is extracted from the dataset. We extracted 89 data points in total leaving the dataset with 2911 data points. We present the resulting distribution of the length of the remaining data points with a histogram and a box plot in Fig 2.2 (before outlier cleaning) and Fig 2.3 (after outlier cleaning).

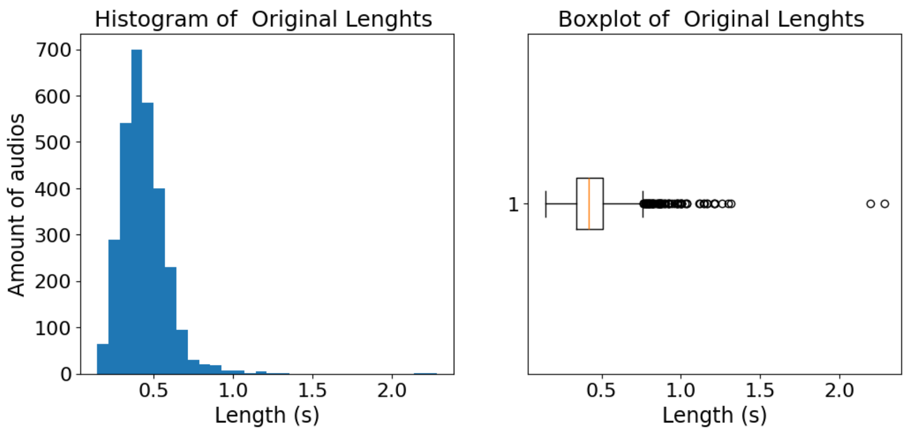


Figure 2.2: Distribution of audio lengths in the dataset before data cleaning.

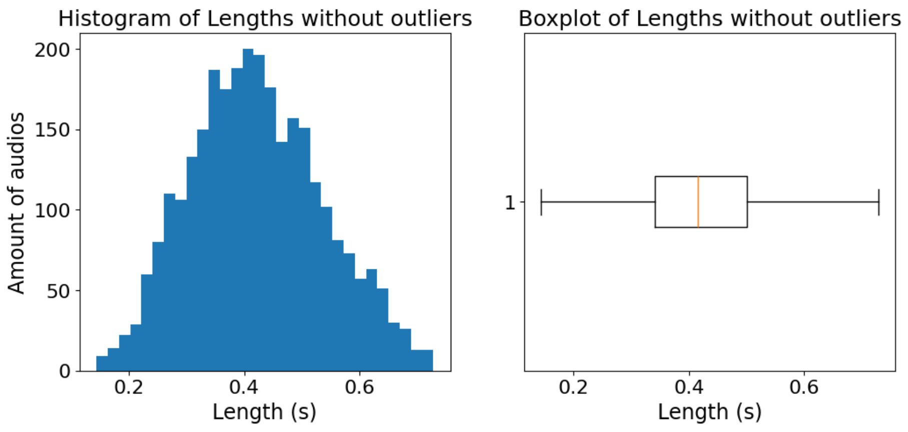


Figure 2.3: Distribution of audio length in the dataset after data cleaning.

We perform noise cleaning on the dataset by implementing the Wiener Filter [38]. Fig 2.4 shows the effect on an audio signal from the dataset.

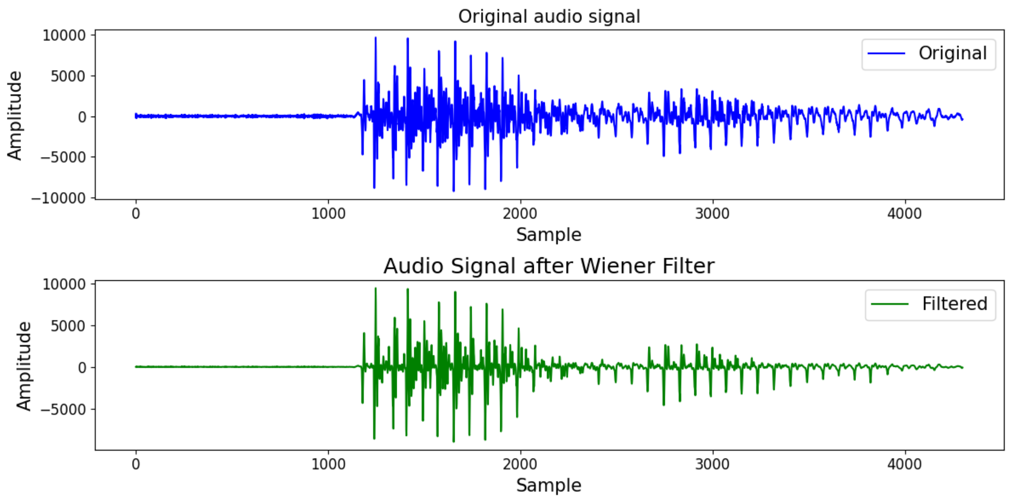


Figure 2.4: Effect of the Wiener Filter for Noise Cleaning.

Then by subtracting the filtered signal to the original signal we calculate the Signal to Noise Ratio (SNR) for each data point through the following procedure

$$noise = orignial - filtered, \tag{2.1}$$

where ”noise” refers to the noise signal, ”original” refers to the original signal and ”filtered” refers to the filtered signal. Then we compute the following expressions

$$P_{\text{original}} = \frac{1}{N}\sum_{i=1}^{N} a_i^2, \tag{2.2}$$

$$P_{\text{noise}} = \frac{1}{N}\sum_{i=1}^{N} b_i^2, \tag{2.3}$$

$$\text{SNR}_{\text{dB}} = 10\log_{10}\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right), \tag{2.4}$$

with a_i the values of the original signal and b_i the values of the noise signal.

From the obtained signal to noise ratios (SNR) we select the top 5 cleanest speakers among the whole dataset and the top 40 cleanest data points for each distinct spoken digit and for each selected speaker. We end with a dataset of 2000 data points. With this, the data points considered not only are the cleanest but also the dataset has the labels balanced. In a balanced dataset there is an equal amount of data points for each label, this highly enhances model performance. Fig 2.5 shows the labels distribution on the dataset.

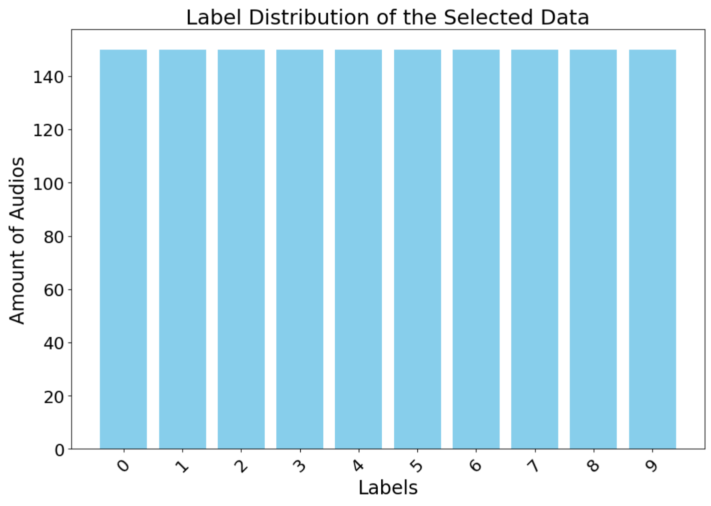


Figure 2.5: Balanced Data Set

1.2 Data Pre-Processing

In order to pass the audio data to the model, the audio waveform must be transformed into insightful features that the model can learn and identify. In Speech Recognition, the feature extraction method involves applying filters to the raw audio signal and projecting them into a frequency and time domain. The process generally consists of extracting the frequency spectrum via applying Fast Fourier Transforms (FFTs) over several time intervals across the audio signal. For each time interval, the frequency spectra obtained is divided into frequency channels that cover a specific range of the spectrum, with this a $N_\tau \times N_f$ matrix $\mathbb{F}_\sigma$ is constructed, where N_f correspons to frequency Chanel's covering a range of the frequency spectrum, N_τ correspons to the time intervals in which the signal was discretized and the index σ indicates the audio file processed from the dataset. By plotting this obtained matrix a spectrogram of the signal is obtained, in which the y-axis displays the frequency content and the x-axis displays the time intervals, showcasing the insightful features of the signal.

The exact process and definition of the frequency channels and time intervals varies with the type of filter applied. In this work we implement a non linear feature extraction filter called the Mel Frequency Cepstrum Coefficients (MFCC) [39].

The number of coefficients or frequency channels selected is 13 ($N_f = 13$), which constitute the first hyper parameter of the algorithm. Each filtered audio $\mathbb{F}_\sigma$ has a different amount of intervals N_τ, for our selected dataset the number of time interval obtained ranges from 3 to 36 as shown in Fig 2.6.

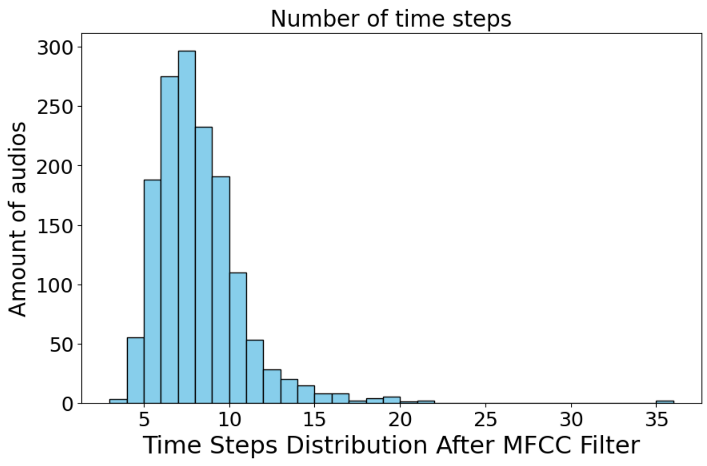


Figure 2.6: Time interval Distribution after Applying MFCC

We perform stratified Train Test Split selecting 10 % of the dataset for testing $\{\mathbb{F}_\sigma^{(\text{test})}\}$ and the remaining 90 % for training $\{\mathbb{F}_\sigma^{(\text{train})}\}$. This train test split is performed so the dataset remains balanced (There is an equal amount of examples for each class).

We perform normalization on the dataset by selecting the max value on the train set and utilizing it to divide the test set and the train set by it, with this the data is on the range [-1,1]

$$\mathbb{F}_\sigma^{(\text{train})} = \frac{\mathbb{F}_\sigma^{(\text{train})}}{\max(\mathbb{F}^{(\text{train})})}, \tag{2.5}$$

$$\mathbb{F}_\sigma^{(\text{test})} = \frac{\mathbb{F}_\sigma^{(\text{test})}}{\max(\mathbb{F}^{(\text{train})})}. \tag{2.6}$$

1.3 Data Encoding

After obtaining the filtered features, performing train test split and normalizing the data, we multiply each feature matrix $\mathbb{F}_\sigma$ with dimensions $N_\tau \times N_f$ by a random mask matrix $\mathbb{M}$ of 1's and -1's, randomly selected. The mask matrix $\mathbb{M}$ has dimensions $N_f \times N_\theta$ with N_θ the number of virtual neurons the hardware circuit is going to emulate

$$\mathbb{U}_\sigma = \mathbb{F}_\sigma \mathbb{M}, \tag{2.7}$$

with $\mathbb{U}_\sigma$ the input matrix for each sample containing the inputs to the on hardware neural network model with dimensions $N_\tau \times N_\theta$.

Finally we scale the input matrices $\mathbb{U}_\sigma$ to the range $[Q_{\max}, Q_{\min}]$ by performing the following data transformation

$$\mathbb{X}_\sigma^{(\text{train})} = \frac{\mathbb{U}_\sigma^{(\text{train})} - \min(\mathbb{F}^{(\text{train})})}{\max(\mathbb{F}^{(\text{train})}) - \min(\mathbb{F}^{(\text{train})})}(Q_{\max} - Q_{\min}) + Q_{\min}, \tag{2.8}$$

$$\mathbb{X}_\sigma^{(\text{test})} = \frac{\mathbb{U}_\sigma^{(\text{test})} - \min(\mathbb{F}^{(\text{train})})}{\max(\mathbb{F}^{(\text{train})}) - \min(\mathbb{F}^{(\text{train})})}(Q_{\max} - Q_{\min}) + Q_{\min}. \tag{2.9}$$

1.4 Hardware Neural Network

After performing data cleaning, pre-processing, and encoding, we feed the neural network with the data. As we describe above, under this approach, we emulate the main portion of the neural network directly on the hardware circuit using the voltage dynamics of the output of the STNO.

In experimental realizations of this approach, a DC is injected into the STNO, as described in Chap. 2, DC induces the precession on the magnetization of the free layer, and the magnetic oscillations generated are converted to voltage oscillation due to magneto-resistance. On top of the Direct Current (DC) that sets the STNO in steady oscillation state a waveform which contains the encoded data is injected as current to the STNO. The response of the STNO produced by this waveform is measured by sampling the output voltage of the STNO using a microwave diode. Finally the amplitude of the output voltage signal is used for the computation. Fig 2.7 shows a diagram of the experimental setup [4].

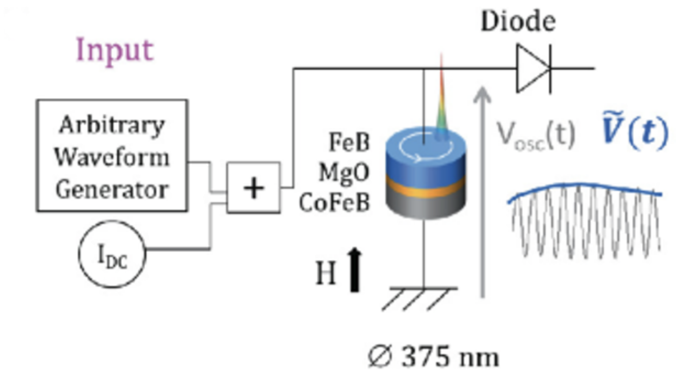


Figure 2.7: Experimental Setup for measuring voltage oscillation in a STNO.[4]

We consider a discredited model of the circuit implemented in experimental realization based on the voltage dynamics of the STNO as derived in Chapter 2. This models showcase the nonlinearity and memory of the STNO, which is suitable for neuromorphic computing. We define two models for the measured voltage amplitude of the STNO. One model accounts for a free layer with vortex ground state as obtained in equation 1.103 and the other one accounts for a free layer with uniform ground state as obtained in equation 1.72 .

For performing the computation we consider the evolution of this models across a time step of Δt. For the model with a free layer in vortex ground state we define the time step of the simulation as

$$v_i^{vortex} = \frac{R_{\text{load}}}{R_{\text{load}} + R_0} \lambda \sqrt{\frac{\frac{I_{\mathbf{DC}}+I_i}{I_{th}} - 1}{\zeta + \xi}} \left(1 - e^{-\Delta t / T_{\text{relax}}}\right) + v_{i-1}^{vortex} \cdot e^{-\Delta t / T_{\text{relax}}}, \qquad (2.10)$$

with v_i^{vortex} the voltage amplitude at time step i, Δt is time length of each time step, I_{DC} is the direct current injected, I_{th} is the threshold current defined by equation 1.90, T_{relax} is the relaxation time of the oscillator defined by equation 1.101, R_{load} is the resistance of the measurement device, R_0 is the mean resistance of the sample, ζ is the non linear damping coeficient, ξ is the non linear stiffnes coeficient and λ is the magnetoresitive factor defined by equation 1.93.

Similarly for the uniform ground state we define the time step of the simulation as

$$v_i^{\text{uniform}} = c \sqrt{\frac{\frac{I_{\mathbf{DC}}+I_i}{I_{th}} - 1}{\frac{I_{\mathbf{DC}}+I_i}{I_{th}} + Q}} \left(1 - e^{-\Delta t / T_{\text{relax}}}\right) + v_{i-1}^{\text{uniform}} \cdot e^{-\Delta t / T_{\text{relax}}}, \qquad (2.11)$$

where v_i^{uniform} is the voltage amplitude at time step i, Δt is time length of each time step, I_{DC} is the direct current injected, I_{th} is the threshold current defined by equation 1.59, T_{relax} is the relaxation time of the oscillator defined by equation 1.70, Q is the nonlinear damping coefficient defined by equation 1.48 and c is a constant fitted to match the the relation between the emitted voltage and the injected DC current in the appropriate range of experimental values. This fitting is possible because a-dimensional power p is proportional to the real emitted power P [33]. We fit c as 0.2 this sets the output voltage to be in the range of mV for injected currents in the range of mA.

We also evaluate a third model for the dynamics of the amplitude voltage of the STNO. This model was introduced in [2]. The time step for a simulation using this model is defined as

$$v_i = c\sqrt{I_{\mathbf{DC}} + I_i - I_{th}} \left(1 - e^{-\Delta t / T_{\text{relax}}}\right) + v_{i-1} \cdot e^{-\Delta t / T_{\text{relax}}}. \qquad (2.12)$$

All of this 3 models approximates the amplitude envelope of the measured voltage oscillations of experimental STNOs and with this the data mapping performed by the STNO is efficiently simulated.

Now, in order to feed the data to the neural network emulated by the STNO model, we form a time series data by concatenating the scaled features $x_{\tau\theta}$, which are the components of the input matrices $\mathbb{X}_\sigma$. The concatenation is performed by the following operation

$$\mathbf{x}_\sigma = flatten(\mathbb{X}_\sigma), \tag{2.13}$$

where $flatten()$ takes a $m \times n$ matrix and outputs a vector of length mn.

In experimental realizations, each value of the obtained time series is applied to the oscillator as a constant Direct Current (DC) during a characteristic time θ. This characteristic time is set to be a portion of the relaxation time of the oscillator. After the injection the response of the oscillator is sampled in a time interval equal to θ.

In our simulation we have time step Δt so each value of the input time series is going to be repeated $\frac{\theta}{\Delta t}$ times. We choose Δt and θ so the values of the time series are repeated 5 times.

$$\mathbf{r}_{\tau\theta} = \underbrace{[x_{\tau\theta}, x_{\tau\theta}, x_{\tau\theta}, x_{\tau\theta}, x_{\tau\theta}]}_{\text{5 times}}, \tag{2.14}$$

$$\mathbf{x}_\sigma = [\mathbf{r}_{11}, \mathbf{r}_{12}, \dots, \mathbf{r}_{21}, \mathbf{r}_{22}, \dots, \mathbf{r}_{\tau\theta}]. \tag{2.15}$$

This concatenated time series is applied to the simulation as input current $I_{in} = x_{\tau\theta}$ and then the output v_i^{osc} in the simulation is sampled every 5 points so the output time series $\mathbf{v}_\sigma$ has $N_\theta \times N_\tau$ points

$$\mathbf{v}_\sigma = STNO(\mathbf{x}_\sigma). \tag{2.16}$$

Then this time series are reshaped to become matrices $\mathbf{V}_k^{(\mathrm{out})}$ of dimensions $N_\theta \times N_\tau$, so a fully connected output layer can be trained and optimized from this matrices.

$$\mathbb{V}_\sigma = reshape(\mathbf{v}_\sigma), \tag{2.17}$$

where $reshape()$ is the reverse operation of $flatten()$, it takes a vector of length mn and outputs a mn matrix .

1.5 Readout Layer

From the output matrices obtained from the emulated neural network $\mathbb{V}_\sigma$, we optimize an output layer with trainable weight matrix $\mathbb{W}^{(\text{out})}$ with dimensions $N_y \times N_\theta$, where N_y is the number of output labels. The dataset selected contains 10 labels so $N_y = 10$. Once the output layer is trained, in order to perform inference, we multiply each instance's output from the emulated neural network $\mathbb{V}_\sigma$ with the output weight $W^{(\text{out})}$ matrix in order to compute the predictions for each interval . We obtain a matrix $\hat{\mathbb{T}}_\sigma$ of dimension $N_y \times N_\tau$

$$\hat{\mathbb{T}}_\sigma = \mathbb{W}^{(\text{out})}\mathbb{V}_\sigma. \tag{2.18}$$

Now in order to compute the overall output of the model for each instance σ, we take the mean of the predictions $\hat{T}_\sigma$ over the intervals τ obtaining the vector $\hat{\mathbb{T}}_\sigma$ with components

$$\hat{t}_y = \frac{1}{N_\tau}\sum_{\tau=1}^{N_\tau}\hat{T}_{y\tau}. \tag{2.19}$$

Finally we implement a winner takes all strategy so we choose the biggest value $\hat{t}_y$ as the predicted class.

$$\text{Predicted Class} = \underset{y}{\text{argmax}}\ \hat{\mathbf{t}}_{\mathbf{y}}. \tag{2.20}$$

1.6 Training

In order to optimize the weights $\mathbb{W}^{(\text{out})}$, we define target matrix $\mathbb{T}_\sigma$ with dimensions $N_y \times N_\tau$ constructed column wise, with each column equivalent to the target vector $\mathbf{t}_\sigma$ with the components t_y defined as

$$t_y = \begin{cases} 1 & \text{if } y = \textit{label} \\ 0 & \text{if } y \neq \textit{label} \end{cases}. \tag{2.21}$$

The training algorithm we implement uses the Moore Penrose Pseudo Inverse [37] defined as

$$\mathbb{A}^+ = (\mathbb{A}^T\mathbb{A})^{-1}\mathbb{A}^T. \tag{2.22}$$

We stack row wise all the emulated neural network outputs $\mathbb{V}_\sigma^{(\text{train})}$ forming the matrix $\mathbb{V} = [\mathbb{V}_1, \mathbb{V}_2, \dots, \mathbb{V}_{N_{\text{train}}}]$. Similar we stack row wise all the target matrices $\mathbb{T}_\sigma^{(\text{train})}$ forming the matrix $\mathbb{T} = [\mathbb{T}_1, \mathbb{T}_2, \dots, \mathbb{T}_{N_{\text{train}}}]$.

So, we calculate the optimal weights by

$$\mathbb{W} = [\mathbb{T}_1, \mathbb{T}_2, \dots, \mathbb{T}_{N_{\text{train}}}]\,[\mathbb{V}_1, \mathbb{V}_2, \dots, \mathbb{V}_{N_{\text{train}}}]^+, \tag{2.23}$$

$$\mathbb{W}^{(\text{out})} = TV^+. \tag{2.24}$$

This is the closed for solution for obtaining the weight matrix $\mathbb{A}^{(\text{out})}$ that minimizes the MSE cost function [19] [37].

1.7 Parameters of the simulation

To select the order of magnitude of the simulated STNO's physical parameters, we surveyed the experimental values. We show the obtained values from the survey in Table II.1 for a free layer with a vortex ground state and in II.2 for a free layer with a uniform ground state.

Parameter	**Grimaldi.et al [34]**	**Mathieu.et al [4]**	**Torr.et al [17]**
Threshold Current (mA)	3.3	3	3.5
Relaxation time (ns)	-	200	500
Applied Field (mT)	440	200-600	430
I DC (mA)	3.3-4.1	1-9	1-7
Radius (nm)	250	187.5	187.5
Thickness (nm)	-	6	-
M_s Free Layer (A/m)	6.5 x 10^5	-	-
M_s Pinned Layer (A/m)	10.1 x 10^5	-	-
Free Layer Material	NiFe	FeB	FeB

Table II.1: Parameter for STNO with free layer with vortex state.

Parameter	**Tomohiro.et al [40]**	**R.Lehndorff.et al [41]**
Threshold Current (mA)	1.6-1.8	27
Relaxation time (ns)	6.3	-
Applied Field (mT)	200-350	(-150)-(+150)
I DC (mA)	2,5	1-39
Radius (nm)	60	115
Thickness (nm)	2	20
M_s Free Layer (T)	1.8	2.15
Free Layer Material	-	Fe

Table II.2: Parameter for STNO with a free layer with the uniform ground state.

From this survey, we select the order of magnitude of the physical parameters of the STNO model we will consider when performing the computation across different simulations. For the applied field, we select the range from 100 to 500 mT; for the radius, we select the range 100 to 250 nm; and for the thickness, the range 2 to 20 nm.

III

Results and Discussion

1 Physical parameters of the simulated STNOs

First, in order to check the physical realization of the simulations, we compute the values of the threshold current and the relaxation time for different sizes of the free layer. We compute these parameters for the vortex ground state model and for the uniform ground state model. These parameters govern the dynamics of the STNO and subsequently affect the behavior of the emulated neural network.

We consider an applied field $H_0 = 440$ mT for a STNO with vortex ground state in the free layer. We assume a free layer magnetization saturation $\mu_0 M_s^{free} = 0.8$ T, a pinned layer magnetization saturation $\mu_0 M_s^{SAF} = 1.2$ T, a spin polarization $P_{\text{spin}} = 0.5$ and a vortex core radius $b = 33$ nm. Figure 3.1 (a) shows $I_{\mathbf{th}}$ as a function of radius R at different thicknesses L of the free layer. Fig 3.1 (b) shows the relaxation time τ as a function of radius R at different thicknesses L. We observe that τ diverges for $R < 70nm$, and $I_{\mathbf{th}}$ becomes negative. This is because for smaller radius a different magnetic ground state and dynamical conditions appear, as a consequence the approximation of the auto oscillator breaks down. Both, the threshold current I_{th} and the relaxation time τ calculated with the STNO model are physically realistic by comparing them with table II.1.

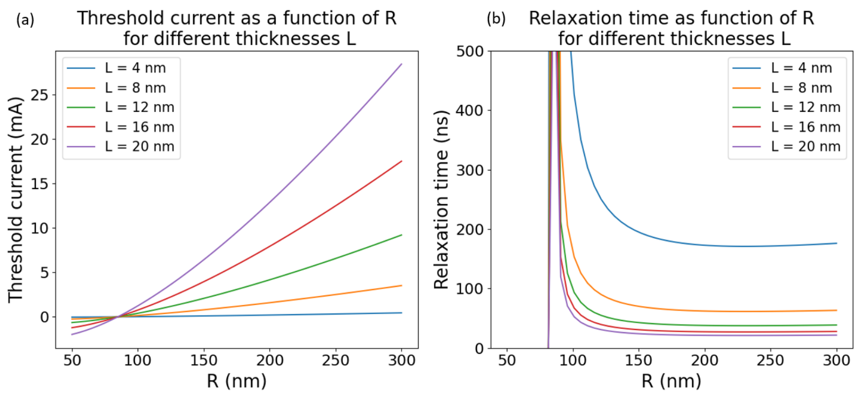


Figure 3.1: (a) Threshold current as a function of radius R for different thickness of the free layer. (b) Relaxation time τ as a function of radius R for different thickness of the free layer. Free layer is in the vortex ground state.

Fig 3.1 (b) shows the relaxation time τ as a function of radius R at different thicknesses L. We observe that τ diverges for $R < 70$ nm. This is because for smaller radius a different magnetic ground state and dynamical conditions appear, as a consequence the approximation of the auto oscillator breaks down.

For a STNO with uniform ground state we consider an applied field $H_0 = 440$ mT, a free layer magnetization saturation $\mu_0 M_s^{\text{free}} = 2.2$ T, a DC current $I_{DC} = 3$ mA and a spin polarization efficency $\epsilon = 0.5$. Fig 3.2 (a) shows the variation for the threshold current and Fig 3.2 (b) shows the relaxation time τ as function of R for different thickness L of the free layer. We observe that the relaxation time diverges above certain radius for different thicknesses. This is because for bigger radius a different magnetic ground state and dynamical conditions appear, as a consequence the approximation of the auto oscillator breaks down.

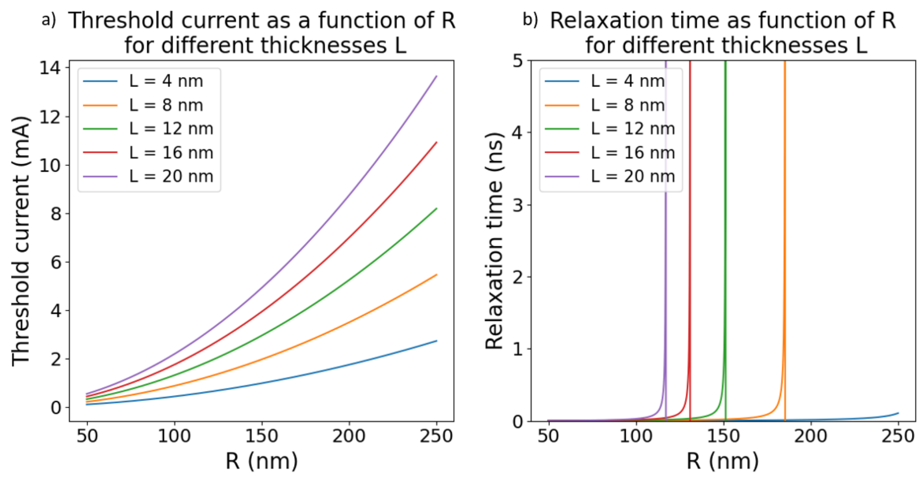


Figure 3.2: (a) Threshold current as a function of radius R for different thickness of the free layer. (b) Relaxation time τ as a function of radius R for different thickness of the free layer. Free layer is in the uniform ground state.

2 Speech Recognition Task Using Spin Torque Nano Oscillator

2.1 Grid Search: Performance v/s Radius and Thickness

The performance of the proposed AI system in Chapter 2 is evaluated by training a model with different parameters and computing the accuracy of defections. We vary the radius and thickness of the free layer for 3 different applied fields. The radius R varies from 100 to 250 nm, the thickness L varies from 3 to 20 nm and the applied fields considered are 440 mT, 350 mT and 250 mT. For simulations we consider that input signals injected to the STNO have a peak amplitude variation of 6 mA. The minimum value of such input signals is set to be just above the threshold current. The neural network is emulated with 2000 neurons. The total processing time of the grid search was of 36 hours.

First we consider STNOs with a free layer with vortex ground state (Eq. 2.10). In this set of simulations we consider a Direct Current (DC) I_{DC} of 10mA, a device mean resistance R_0 of 100 ohms, a load resistance R_{load} of 50 ohms, a free layer magnetization saturation $\mu_0 M_s^{\text{free}}$ of 0.81 T, a pined layer magnetization saturation $\mu_0 M_s^{\text{SAF}}$ of 1.2 T, a spin polarization P_{spin} of 0.5 and a vortex core radius b of 33 nm. The results of the performance of the AI model in the test set are shown in Fig 3.3 (a) - (c). The performance results in the train set are shown in Fig 3.4 (a) - (c). Each figure correspond to a distinct applied field.

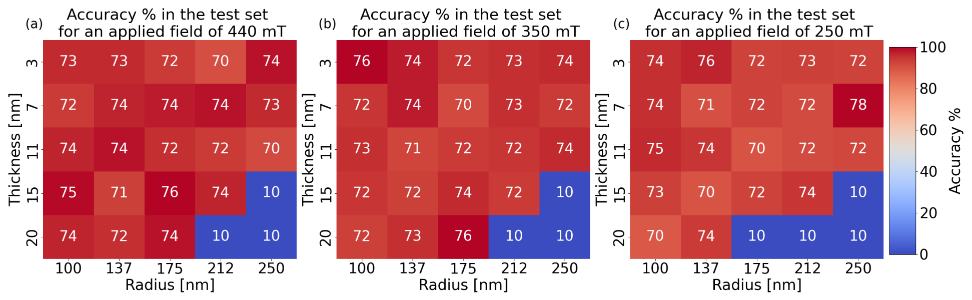


Figure 3.3: Performance of the AI model in the test set (The STNOs considered have a vortex ground state in the magnetization of the free layer). For an applied field of (a) 440 mT, (b) 350 mT and (c) 250 mT.

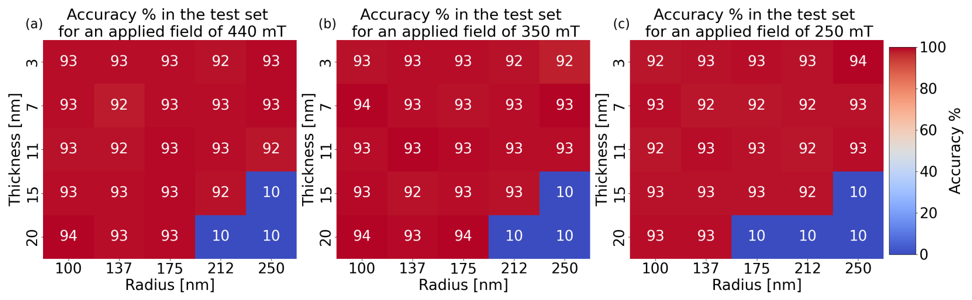
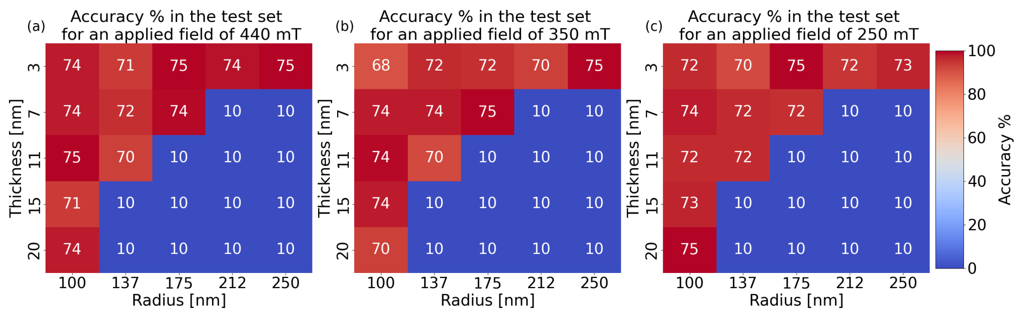


Figure 3.4: Performance of the AI model in the train set (The STNOs considered have a vortex ground state in the magnetization of the free layer). For an applied field of (a) 440 mT, (b) 350 mT and (c) 250 mT.

For STNOs with a uniform ground state in the free layer (Eq. 2.11) we consider an injected DC current of 7 mA a free layer magnetization saturation $\mu_0 M_s^{\text{free}}$ of 0.81 T and a spin polarization efficiency ϵ of 0.5. The results of the the results of the performance of the AI models in the test set are shown in Fig 3.5 (a) - (c). The results of the performance of the AI model in the train set are shown in Fig 3.6 (a) - (c). Each figure correspond to a distinct applied field.

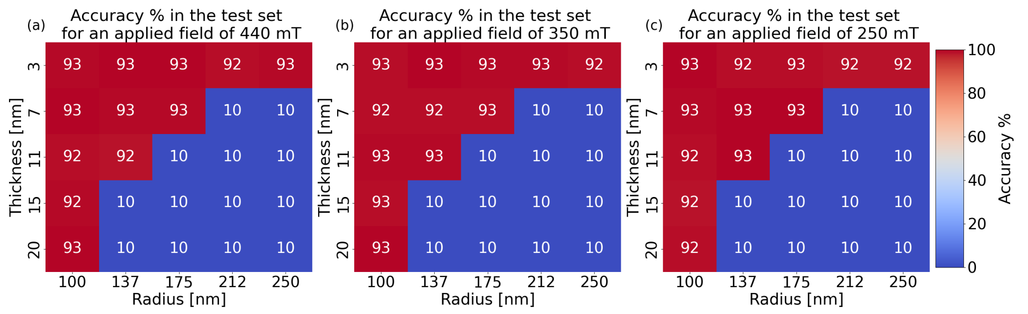


(a) 440 mT: Thickness [nm] \ Radius [nm]	100	137	175	212	250
3	74	71	75	74	75
7	74	72	74	10	10
11	75	70	10	10	10
15	71	10	10	10	10
20	74	10	10	10	10

(b) 350 mT: Thickness [nm] \ Radius [nm]	100	137	175	212	250
3	68	72	72	70	75
7	74	74	75	10	10
11	74	70	10	10	10
15	74	10	10	10	10
20	70	10	10	10	10

(c) 250 mT: Thickness [nm] \ Radius [nm]	100	137	175	212	250
3	72	70	75	72	73
7	74	72	72	10	10
11	72	72	10	10	10
15	73	10	10	10	10
20	75	10	10	10	10

Figure 3.5: Performance of the AI model in the test set (The STNOs considered have a uniform ground state in the magnetization of the free layer). For an applied field of (a) 440 mT, (b) 350 mT and (c) 250 mT.



(a) 440 mT: Thickness [nm] \ Radius [nm]	100	137	175	212	250
3	93	93	93	92	93
7	93	93	93	10	10
11	92	92	10	10	10
15	92	10	10	10	10
20	93	10	10	10	10

(b) 350 mT: Thickness [nm] \ Radius [nm]	100	137	175	212	250
3	93	93	93	93	92
7	92	92	93	10	10
11	93	93	10	10	10
15	93	10	10	10	10
20	93	10	10	10	10

(c) 250 mT: Thickness [nm] \ Radius [nm]	100	137	175	212	250
3	93	92	93	92	92
7	93	93	93	10	10
11	92	93	10	10	10
15	92	10	10	10	10
20	92	10	10	10	10

Figure 3.6: Performance of the AI model in the train set (The STNOs considered have a uniform ground state in the magnetization of the free layer). For an applied field of (a) 440 mT, (b) 350 mT and (c) 250 mT.

The trend observed from the simulations is that increasing the size of the free layer abruptly decreases the performance of the algorithm implemented. This trend is greater for a STNO with uniform magnetization as magnetic ground state in the free layer. For a STNO with vortex ground state the decrease of the performance starts at a radius of 212 nm and a thickness of 15 nm. Meanwhile for a STNO with uniform ground state the decrease of the performance starts at a radius of 137 and a thickness of 7 nm. Changes on the applied field have minimal effect on the performance of the AI model.

2.2 Non linear magnetic oscillator model

Among all the models utilized for simulating the STNO the one that has accuracy is the model from equation 2.12 [2]. We optimize the processing by following [4]. The input signal amplitude peak to peak should be optimized such that it covers a good range of the non linear variation of the $V(I)$ curve. We consider similar parameters to the ones mentioned in [2]. The parameters chosen for the simulation are $\Delta t = 20$ ns, $I_{in} = \pm 3$ mA, $I_{\text{DC}} = 7$ mA and $T_{\text{relax}} = 410$ ns. Fig 3.7 shows the function V(I) for the selected STNO simulation and the peak to peak range of variations we select (6 mA). For this training instance we emulate 1000 neurons ($N_\theta = 1000$) and consider the sample time $\theta = 100$ ns. Under the latter set of parameters, we achieve an accuracy of 90.5 % on the test set with 10.000 trainable weights (The amount of trainable weights is equivalent to the number of weights in the output weight matrix $[\mathbb{W}^{(\text{out})}]_{N_y \times N_\theta}$).

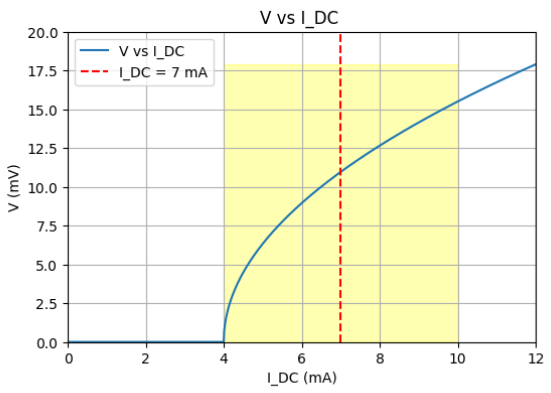
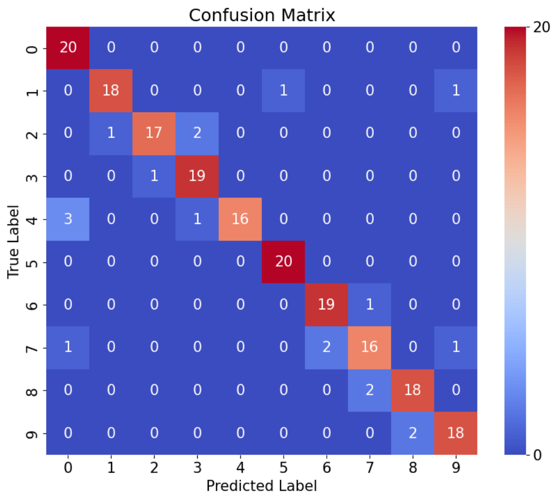


Figure 3.7: V(I) curve for the selected STNO.

Fig 3.8 shows the confusion matrix of the model on the test set, which represent on the y axis the true labels and on the x axis the predicted labels by the model. We can see that the model accurately identifies correct digits with some small errors mainly miss confusing the digits 4 and 7.



True Label \ Predicted Label	0	1	2	3	4	5	6	7	8	9
0	20	0	0	0	0	0	0	0	0	0
1	0	18	0	0	0	1	0	0	0	1
2	0	1	17	2	0	0	0	0	0	0
3	0	0	1	19	0	0	0	0	0	0
4	3	0	0	1	16	0	0	0	0	0
5	0	0	0	0	0	20	0	0	0	0
6	0	0	0	0	0	0	19	1	0	0
7	1	0	0	0	0	0	2	16	0	1
8	0	0	0	0	0	0	0	2	18	0
9	0	0	0	0	0	0	0	0	2	18

Figure 3.8: Confusion matrix of the emulated neural network with the STNO

Without the emulated neural network by the STNO (The extracted features F_σ are directly passed to the output layer) we achieved a accuracy of 53 % on the test set with 130 trainable weights (The output weight matrix $\mathbb{W}^{(\mathrm{out})}$ in this case has dimensions $N_y \times N_f$, specifically $N_y = 10$ and $N_f = 13$, so we have 130 trainable weights). For comparison we train a Multiplayer Perceptron (MLP) on the extracted features $\mathbb{F}_\sigma$ and achieve 91 % on the test set but with approximate 20.000.000 trainable weights (For this MLP the network structure selected is: input layer with 13 features, 1st hidden layer with 256 neurons, 2nd hidden layer with 128 neurons, 3rd hidden layer with 64 neurons and output layer with 10 neurons).

Considering simulation times, the reservoir simulation takes on average 0.3 s to process a single data sample for 1000 emulated neurons. It takes 6.2 min for processing the whole training set with 1800 data samples and the training algorithm takes 2.5 s. Now, by considering the simulations performed with the non linear magnetic oscillator model and by despising the time between DC injections we estimate that in an experimental realization the processing of an input signal takes 3.6 ms for 1000 emulated neurons and 36 time intervals (Which is the max value among our dataset). So processing the whole training set would take about 6.4 s if considering that all the data samples have 36 time intervals (This is an upper limit for our dataset since as seen in fig 2.6 the majority of samples have less than 36 time intervals). Now by considering the training algorithm we estimate that the whole learning process takes 8.9 s. In our system training the MLP takes 20 min. So the reservoir computing processing with the STNO is also faster to train that the MLP. Fig 3.9 shows the comparison between MLP, the model without the emulated neural network by the STNO and the model with the emulated neural network by the STNO trained on the speech recognition task. This comparison highlights the efficiency of the proposed model for efficient AI processing since it achieves similar performance metrics as more complex models while being 2000 times smaller in terms of weights.

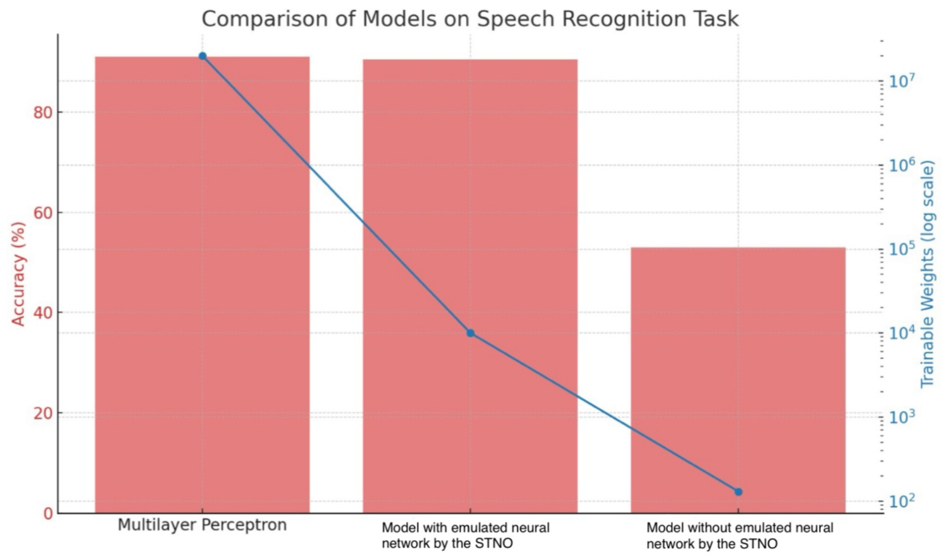


Figure 3.9: Comparison of performance.

IV Summary

8 Conclusions

We evaluate the efficiency of a Spin Torque Nano Oscillator as an energy-efficient AI hardware. We test its performance with a speech recognition AI system that directly used the STNO as an emulated neural network on the hardware circuit. We derive an analytical expression for operating a Spin Torque Nano Oscillator for 2 different magnetic ground states in the free layer: the vortex and the uniform. In a neuromorphic computing approach, these approximated models contain all the necessary ingredients for performing artificial intelligence computations directly on the hardware circuit. The models study the performance of the AI systems as a function of the radius and thickness of the free layer. We obtain that the performance using these models abruptly decreased as the size of the free layer increased. This abrupt decrease occurs for $R > 50$ nm of the STNOs with uniform ground state.

We evaluate the nonlinear magnetic oscillator model from [2], which reaches similar performance as more robust and bigger models with less trainable parameters and a simpler learning algorithm. The model reduces the neural network size up to 3 orders of magnitude by the implementation of the computation using the STNO for emulating the neural network. Since we have less trainable weight, the latter approach reduces considerably the energy needed for performing the computation. Hence, Spin Torque Nano Oscillator and the Magnetic Tunnel Junction are promising nano electronic devices suited for novel energy efficient artificial intelligence hardware. They have the potential to solve energetic problems and limitations that the modern AI systems struggles with. As more practical implementations are developed at the commercial stage, these technologies get closer and closer to become a great tool for designing next generation hardware and continuing scaling AI systems.

Bibliography

[1] Zohar Jackson, César Souza, Jason Flaks, Yuxin Pan, Hereman Nicolas, and Adhish Thite. Jakobovski/free-spoken-digit-dataset: v1.0.8, August 2018.

[2] Flavio Abreu Araujo, Mathieu Riou, Jacob Torrejon, Sumito Tsunegi, Damien Querlioz, Kay Yakushiji, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Mark D. Stiles, and Julie Grollier. Role of non-linear data processing on speech recognition task in the framework of reservoir computing. *Scientific Reports*, 10:328, 2020.

[3] Zhongming Zeng, Giovanni Finocchio, and Hongwen Jiang. Spin transfer nano-oscillators. *Nanoscale*, 5:2219, 2013.

[4] Mathieu Riou, Jacob Torrejon, Flavio Abreu Araujo, Sumito Tsunegi, Guru Khalsa, Damien Querlioz, Paolo Bortolotti, Nathan Leroux, Danijela Marković, Vincent Cros, Kay Yakushiji, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Mark D. Stiles, and Julie Grollier. *Reservoir Computing Leveraging the Transient Non-linear Dynamics of Spin-Torque Nano-Oscillators*, page 307. Springer Singapore, Singapore, 2021.

[5] Jason Furman and Robert Seamans. Ai and the economy. *Innovation policy and the economy*, 19:161, 2019.

[6] Haris Isyanto, Ajib Setyo Arifin, and Muhammad Suryanegara. Performance of smart personal assistant applications based on speech recognition technology using iot-based voice commands. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, page 640, 2020.

[7] Emil Talpes, Debjit Das Sarma, Ganesh Venkataramanan, Peter Bannon, Bill McGee, Benjamin Floering, Ankit Jalote, Christopher Hsiong, Sahil Arora, Atchyuth Gorti, and Gagandeep S. Sachdev. Compute solution for tesla's full self-driving computer. *IEEE Micro*, 40:25, 2020.

[8] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, and ... Sam Altman. Gpt-4 technical report, 2024.

[9] Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, 2023.

[10] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Ai accelerator survey and trends. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, page 1, 2021.

[11] Patrick Bowen, Guy Regev, Nir Regev, Bruno Pedroni, Edward Hanson, and Yiran Chen. Analog, in-memory compute architectures for artificial intelligence, 2023.

[12] Anni Lu, Junmo Lee, Tae-Hyeon Kim, Muhammed Ahosan Ul Karim, Rebecca Sejung Park, Harsono Simka, and Shimeng Yu. High-speed emerging memories for ai hardware accelerators. *Nature Reviews Electrical Engineering*, 1:24, 2024.

[13] Dennis V Christensen, Regina Dittmann, Bernabe Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, Ilia Valov, Gianluca Milano, Carlo Ricciardi, Shi-Jun Liang, Feng Miao, Mario Lanza, Tyler J Quill, Scott T Keene, Alberto Salleo, Julie Grollier, Danijela Marković, Alice Mizrahi, Peng Yao, J Joshua Yang, Giacomo Indiveri, John Paul Strachan, Suman Datta, Elisa Vianello, Alexandre Valentian, Johannes Feldmann, Xuan Li, Wolfram H P Pernice, Harish Bhaskaran, Steve Furber, Emre Neftci, Franz Scherr, Wolfgang Maass, Srikanth Ramaswamy, Jonathan Tapson, Priyadarshini Panda, Youngeun Kim, Gouhei Tanaka, Simon Thorpe, Chiara Bartolozzi, Thomas A Cleland, Christoph Posch, ShihChii Liu, Gabriella Panuccio, Mufti Mahmud, Arnab Neelim Mazumder, Morteza Hosseini, Tinoosh Mohsenin, Elisa Donati, Silvia Tolu, Roberto Galeazzi, Martin Ejsing Christensen, Sune Holm, Daniele Ielmini, and N Pryds. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2:022501, 2022.

[14] Sumanth Umesh and Sparsh Mittal. A survey of spintronic architectures for processing-in-memory and neural networks. *Journal of Systems Architecture*, 97:349, 2019.

[15] Vinod Kumar Joshi. Spintronics: A contemporary review of emerging electronics devices. *Engineering Science and Technology, an International Journal*, 19:1503, 2016.

[16] Julie Grollier, Damien Querlioz, and Mark Stiles. Spintronic nanodevices for bioinspired computing. *Proceedings of the IEEE*, PP, 2016.

[17] Jacob Torrejon, Mathieu Riou, Flavio Abreu Araujo, Sumito Tsunegi, Guru Khalsa, Damien Querlioz, Paolo Bortolotti, Vincent Cros, Kay Yakushiji, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Mark D. Stiles, and Julie Grollier. Neuromorphic computing with nanoscale spintronic oscillators. *Nature*, 547:428, 2017.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[19] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd edition, 2019.

[20] Gabriele Novembri, Francesco Rossini, and Antonio Fioravanti. *Construction time and cost optimization using A.I. and statistical methods, through Bayes-Point Machines*, page 40. 2017.

[21] Luis Camuñas-Mesa, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Neuromorphic spiking neural networks and their memristor-cmos hardware implementations. *Materials*, 12:2745, 2019.

[22] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550, 1990.

[23] Yusuke Sakemi, Kai Morino, Timoth'ee Leleu, and Kazuyuki Aihara. Model-size reduction for reservoir computing by concatenating internal states through time. *Scientific Reports*, 10, 2020.

[24] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100, 2019.

[25] Kohei Nakajima and Ingo Fischer. *Reservoir computing*. Springer, 2021.

[26] Xingqi Zou, Sheng Xu, Xiaoming Chen, Liang Yan, and Yinhe Han. Breaking the von neumann bottleneck: architecture-level processing-in-memory technology. *Science China Information Sciences*, 64:160404, 2021.

[27] Gordon E Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86:82, 1998.

[28] R. Stanley Williams. What's next? [the end of moore's law]. *Computing in Science Engineering*, 19:7, 2017.

[29] David Thomson, Aaron Zilkie, John E Bowers, Tin Komljenovic, Graham T Reed, Laurent Vivien, Delphine Marris-Morini, Eric Cassan, Léopold Virot, Jean-Marc Fédéli, Jean-Michel Hartmann, Jens H Schmid, Dan-Xia Xu, Frédéric Boeuf, Peter O'Brien, Goran Z Mashanovich, and M Nedeljkovic. Roadmap on silicon photonics. *Journal of Optics*, 18:073003, jun 2016.

[30] Benedetta Flebus, Dirk Grundler, Bivas Rana, YoshiChika Otani, Igor Barsukov, Anjan Barman, Gianluca Gubbiotti, Pedro Landeros, Johan Akerman, Ursula Ebels, Philipp Pirro, Vladislav E Demidov, Katrin Schultheiss, Gyorgy Csaba, Qi Wang, Florin Ciubotaru, Dmitri E Nikonov, Ping Che, Riccardo Hertel, Teruo Ono, Dmytro Afanasiev, Johan Mentink, Theo Rasing, Burkard Hillebrands, Silvia Viola Kusminskiy, Wei Zhang, Chunhui Rita Du, Aurore Finco, Toeno van der Sar, Yunqiu Kelly Luo, Yoichi Shiota, Joseph Sklenar, Tao Yu, and Jinwei Rao. The 2024 magnonics roadmap. *Journal of Physics: Condensed Matter*, 36(36):363501, jun 2024.

[31] Catherine Schuman, Shruti Kulkarni, Maryam Parsa, J. Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2:10, 2022.

[32] Mark D. Stiles and Jacques Miltat. *Spin-Transfer Torque and Dynamics*, page 225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[33] Andrei Slavin and Vasil Tiberkevich. Nonlinear auto-oscillator theory of microwave generation by spin-polarized current. *Magnetics, IEEE Transactions on*, 45:1875, 2009.

[34] Eva Grimaldi, Antoine Dussaux, Paolo Bortolotti, Julie Grollier, Grégoire Pillet, Akio Fukushima, Hitoshi Kubota, Kay Yakushiji, Shinji Yuasa, and Vincent Cros. Response to noise of a vortex based spin transfer nano-oscillator. *Phys. Rev. B*, 89:104404, 2014.

[35] A. Dussaux, A. V. Khvalkovskiy, P. Bortolotti, J. Grollier, V. Cros, and A. Fert. Field dependence of spin-transfer-induced vortex dynamics in the nonlinear regime. *Phys. Rev. B*, 86:014402, 2012.

[36] Nicolas Locatelli, Damir Vodenicarevic, Weisheng Zhao, Jacques-Olivier Klein, Julie Grollier, and Damien Querlioz. Vortex-based spin transfer oscillator compact model for ic design. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, page 589, 2015.

[37] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics. Springer, 2009.

[38] Jingdong Chen, J. Benesty, Yiteng Huang, and S. Doclo. New insights into the noise reduction wiener filter. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1218–1234, 2006.

[39] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

[40] Tomohiro Taniguchi, Takahiro Ito, Sumito Tsunegi, Hitoshi Kubota, and Yasuhiro Utsumi. Relaxation time and critical slowing down of a spin-torque oscillator. *Phys. Rev. B*, 96:024406, 2017.

[41] R. Lehndorff, D. E. Bürgler, S. Gliga, R. Hertel, P. Grünberg, C. M. Schneider, and Z. Celinski. Magnetization dynamics in spin torque nano-oscillators: Vortex state versus uniform state. *Phys. Rev. B*, 80:054412, 2009.