Designing Oscillatory Neural Networks by Machine Learning

By:

Tamás Rudner-Halász

Theses of the PhD dissertation

Supervisor:

Dr. György Csaba PhD

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics
Roska Tamás Doctoral School of Sciences and Technology

Budapest, 2025

1 Introduction

Over the last decades, Moore's law has held for digital computers, stating that the number of transistors on an integrated circuit doubles every year. [1], [2], [3] Even though the physical limits have almost been reached (such as source-to-drain leakage, limited gate metals), there are new technologies mostly involving spintronics, tunnel junctions, and nano-wiring, which are promising. [4], [5], [6]

To address these problems, several proposals have proposed using physics in computing and building architectures that operate in fundamentally different ways than conventional computers [7]. To create such systems, computational problems must be investigated from a fundamentally different perspective. Furthermore, it is necessary to map the usual computational problems to physical ones to exploit physics for computing, similar to quantum computing. [8]

Most computing paradigms that leverage physics, using some variation of energy minimization or ground-state finding, are optimization problems. Naturally, all systems are trying to converge to their own minimal energy configuration, which should be the true solution (or one of the solutions) with appropriate setting. [9]

Additionally, by leveraging physics for computation, computational devices are increasingly shifting toward biologically inspired architectures, as neural systems and mammalian brains appear to use oscillatory signals in their functions to achieve a given capacity.

In this thesis, I will introduce a novel direction in analog neuromorphic computing using oscillatory networks.

In this thesis, I present the foundations of the architectures I have developed and explain why scientists from various fields, namely neuroscience, physics, and computer science, converged.

First, I introduce the relevant fields and the results most influential to my research from these areas, such as the Hodgkin-Huxley model in neuroscience and the Kuramoto model, whose development was also motivated by biological oscillators.

Subsequently, I introduce two electrical circuit representations of oscillators that are easy to use in simulation or software environments.

The thesis and my subsequent research can be divided into three segments. First, I discuss how I developed the ODE system to simulate a ring-oscillator-based neural network and how I trained it to solve specific tasks (Thesis I). Subsequently, I outlined time-independent and time-dependent tasks and solutions using ONNs.

For the time-independent, static problem, I used the MNIST dataset for regular handwritten digit recognition as a binary and a multiclass problem (Theses II, III, and V).

Meanwhile, for the other time-dependent problem, I used audio recognition, specifically vowel recognition. There, I introduce a frequency-based computing architecture using both Kuramoto and ring oscillators for vowel recognition (Theses IV and V).

Furthermore, in the following sections, I will highlight two projects involving several research groups and companies to emphasize the motivation for reconsidering how computers are thought about, rather than as humanity has before.

1.1 The shared interest of Intel, BMW, and IBM for a novel computing paradigm

In the past couple of years, I have had the opportunity to work with several companies and research groups on various projects. In the following paragraphs, I will highlight the ones relevant to this dissertation and the intended applications of the companies.

1.1.1 Hierarchically Interconnected Multi-layered Oscillator Networks (HIMON)

In this project, Intel Corporation sought an energy-efficient, low-cost alternative to conventional computing paradigms to solve computationally hard problems on hardware.

They were motivated to either develop full-scale devices for NP-hard problems or to create an accelerator for various high-computational-cost algorithms, such as in silico learning.

Several research groups were encompassed. Practical research included on-chip design for NP-hard problems [10]; meanwhile, there were more theoretically focused works, such as higher-order Ising machine solutions [11].

1.1.2 Phase Change Materials for Energy Efficient Edge Computing (PHASTRAC)

In this project, the two companies—IBM and BMW—and the research group at TU Eindhoven were interested in two distinct research areas that converged. The primary application was from BMW, which sought a rapid solution to several classification tasks arising within a car. This is mainly due to the fact that cars today have a lot of sensors, even inside the car, and a lot of data to process, but limited options to

On the other hand, IBM was motivated to find a way to use the phase change materials to provide energy efficiency in edge computing using relaxation VO_2 oscillators as a basis for emulating the behavior of neurons and couple them together with ReRAM. [12].

From these two projects, it is evident to me that interest in and need for novel, non-Boolean computing paradigms are on the rise and will continue to grow in the next chapter.

2 Methods and Tools

2.1 Computational models for ONNs

I primarily used ring oscillators and Kuramoto oscillators in this dissertation to address the tasks outlined.

2.1.1 Description of the ring oscillator model

The ring oscillator model I used is shown in Figure 2.1.

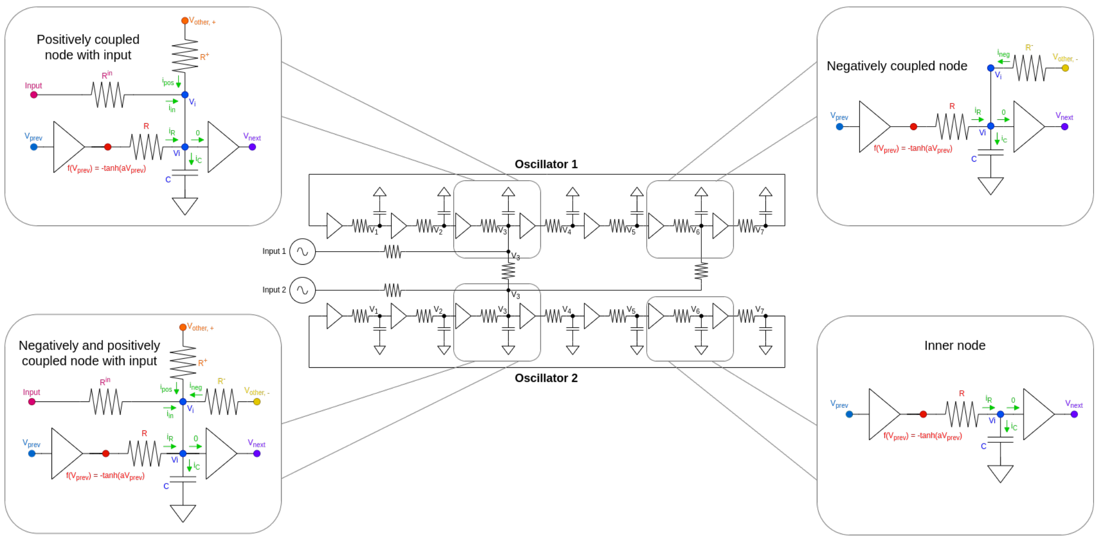

Positively coupled node with input
Negatively coupled node
Oscillator 1
Input 1
Input 2
Oscillator 2
Negatively and positively coupled node with input
Inner node
$f(V_{prev}) = -\tanh(aV_{prev})$

Figure 2.1: **Circuit diagram of a two-oscillator system with both of the coupling schemes.** Here in the middle, a two oscillator system can be seen, coupled together in the two possible ways — positive and negative — with numbered voltage nodes. The oscillators are built using 7 inverters. The zoomed-in regions at the four corners represent the different node types that a given system can have. In those four highlights, nodes of the same color are at the same voltage levels, and v_{i-1} and v_{i+1} denote the nodes preceding and following v_i, respectively, due to the circular design. The green arrows symbolize the agreed-upon directions of the currents. The 0-labeled current denotes $0A$, as by definition the input current to inverters is 0. Also, the v_- and v_+ are symbols for the voltage node of another oscillator, which is coupled to the particular oscillator negatively and positively, respectively. The *Input*-labelled waveform generators can be any source of external input voltages to the system. Note that these input generators can also be current sources; in that case, the connecting resistor is absent.

Quantitatively, the ODEs used for simulation, training, and inference are given by the following equations derived using Kirchhoff's law of currents [13]:

- Voltage generator:

$$C\frac{d\boldsymbol{v}}{dt} = \underbrace{\frac{1}{R}\boldsymbol{A'}(f(\boldsymbol{P_\pi v}) - \boldsymbol{v})}_{\text{Intrinsic dynamics}} + \frac{1}{R_c}\overbrace{\boldsymbol{C'v}}^{\text{Coupling dynamics}} - \underbrace{\frac{1}{R_{in}}\Big(\boldsymbol{B'} \odot (\boldsymbol{v} \ominus \boldsymbol{u}^T)\Big)\boldsymbol{1}}_{\text{External dynamics}} \tag{2.1}$$

- Current generator:

$$C\frac{d\boldsymbol{v}}{dt} = \underbrace{\frac{1}{R}\boldsymbol{A'}(f(\boldsymbol{P_\pi v}) - \boldsymbol{v})}_{\text{Intrinsic dynamics}} + \frac{1}{R_c}\overbrace{\boldsymbol{C'v}}^{\text{Coupling dynamics}} - \frac{1}{R_{in}}\underbrace{\boldsymbol{B'u}}_{\text{External dynamics}} \tag{2.2}$$

where again, $f(x)$ was used as the behavioral model of the inverters:

$$f(x) = -\tanh(ax), \tag{2.3}$$

Here, $a \in \mathbb{R}^+$. Furthermore, π is a permutation, such that

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}, \tag{2.4}$$

and $\boldsymbol{P} \in \{0,1\}^{(7n)\times(7n)}$ is a block matrix for which every 7×7 matrix-block in the main diagonal, there is a permutation matrix corresponding to π. This orders the voltage nodes in the ring oscillator to calculate the voltage differences arising between the two endpoints of the resistors placed in between two inverters.

$\boldsymbol{B'} \in \mathbb{R}_{\geq 0}^{(7n)\times k}$ is the connector matrix for the inputs, assuming that the input $\boldsymbol{u} \in \mathbb{R}^k$, which is typically a sinusoidal voltage or current generator, but in a later chapter a non-coherent signal will be used as voltage levels for a voltage generator. $\boldsymbol{C'} \in \mathbb{R}_{\geq 0}^{(7n)\times(7n)}$ is the modified coupling matrix which is to be constructed from the real, humanly readable coupling matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$. The parameters $R, C \in \mathbb{R}_{\geq 0}$ are fixed for the oscillators, but the actual resistance between the inverters might change — either through learning or manual tuning — to set the frequency of the oscillators. This is done by the introduction of the $\boldsymbol{A'} \in \mathbb{R}_{\geq 0}^{(7n)\times(7n)}$ block diagonal matrix. This is directly related to the oscillators' frequency.

Note that I used $\boldsymbol{A'}, \boldsymbol{B'}$ and $\boldsymbol{C'}$ to train the network to solve various tasks.

2.1.2 Kuramoto model

Later, in developing this thesis, I also used the Kuramoto model as a proof of concept for frequency-based computing. This model is purely mathematical, so its properties and behavior are superior to those of ring oscillators, but it was a valuable first step to assess whether the frequency-based approach is feasible in theory.

This model was introduced in 1975 by Yoshiki Kuramoto and remains fundamental in the mathematical modelling of synchronization in large populations of coupled oscillators with non-homogeneous frequencies [14].

The model I used in the simulations is the following:

$$\frac{1}{2\pi}\frac{d\phi_i}{dt} = \underbrace{f_i}_{\text{Intrinsic frequency}} + \overbrace{\sum_{j=1}^{N} C_{i,j}\sin\left(\phi_j - \phi_i\right)}^{\text{Coupling dynamics}} + \underbrace{\sum_{k=1}^{N} B_{i,k}\sin\left(u_k - \phi_i\right)}_{\text{External dynamics}}$$

This model may differ from the basic model, but the only addition is the external dynamics. There, I represented the external stimuli as a network oscillator with a predefined frequency and a unidirectional connection to the computing oscillators, so that the input oscillators remained constant over time.

2.2 Learning methods

I used both gradient-based and gradient-free learning methods in order to train the networks to solve various tasks.

2.2.1 Backpropagation Through Time

Backpropagation is the de facto standard algorithm for training neural networks [15]. After each neural network inference, the gradient of the aforementioned properly defined loss function is computed efficiently with respect to the system's trainable parameters. Subsequently, the gradient is used to update the system's trainable parameters, thereby driving them to converge to the — possibly local, but ideally global — minimum of the loss function.

BPTT is backpropagation applied to a dynamic system (i.e. an ODE-based description). The ODE can be solved using any standard time-stepping technique—implicit or explicit—using a discrete time step. This discretized solution can be viewed as a many-layer neural network, with one neural layer corresponding to a temporal snapshot of the system dynamics. The BPTT algorithm computes and stores these layers (snapshots) during the forward pass, then calculates the derivatives of the loss function with respect to the trainable parameters in the backward pass.

The code for the simulation and training of various ONNs has been written in *PyTorch*[16]. The autograd feature of PyTorch makes implementing backpropagation and BPTT straightforward. Additionally, *torchdiffeq*[17] package was used for implementing backward-differentiable ODE solvers in some cases, but sometimes a self-implemented solver was used for controlled voltage extraction. *torchdiffeq* is an external, third-party library built on *PyTorch* that provides various differentiable ODE solvers implemented in *PyTorch*. A useful feature is that it can apply the adjoint method to the backward step [18] and compute gradients with a constant memory cost.

It should be noted that BPTT is computationally demanding for complex dynamic systems such as ONNs. The time-domain solution of an oscillating architecture typically consists of thousands of time steps. As BPTT unwraps the time-domain solution of an ODE into a many-layer neural network, it must handle a network with many thousands of layers, which may lead to memory bottlenecks during training.

Backpropagation through many layers also inevitably suffers from the vanishing or exploding gradient problem [19]. This arises from the inherent structure of backpropagation, which uses the chain rule of differentiation. When traversing many layers during the backward pass, the gradients may become nearly zero, preventing meaningful learning, or they may converge to infinity quickly, which again makes learning impossible. I found that my algorithm produces functional gradients up to a few thousand time steps (layers). The ONNs introduced later are designed to converge safely within this time frame.

2.2.2 Nevergrad: a gradient-free library

Gradient-free optimization, also known as derivative-free optimization, refers to a class of algorithms that optimize objective functions without relying on gradient information. These methods are particularly useful when gradients are unavailable, unreliable, or prohibitively expensive to compute, such as in simulations, black-box functions, or highly non-differentiable landscapes. Instead of gradient descent, gradient-free optimization relies on direct function evaluations to iteratively improve candidate solutions.

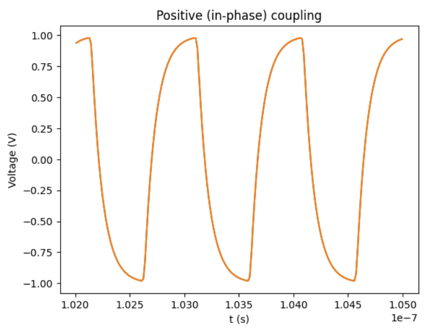
Nevergrad is an open-source Python package developed by Facebook AI Research for gradient-free optimization [20]. It provides a wide range of optimization algorithms, including evolutionary strategies, CMA-ES, PSO, and bandit-based methods, under a unified API. Nevergrad is designed to handle continuous, discrete, and mixed-variable optimization problems, making it highly flexible for both academic research and industrial applications. Additionally, it integrates well with machine-learning workflows and provides benchmarking tools for comparing optimizer performance. Its accessibility and versatility have made it a widely adopted tool in black-box optimization tasks.

I used this library to solve the vowel formant recognition problem; however, it should be noted that this small architecture could also have used BPTT. Having said that, I wanted to experiment with a potentially useful approach for the future, as BPTT has limitations if the number of oscillators gets to a point where it would consume too much memory.

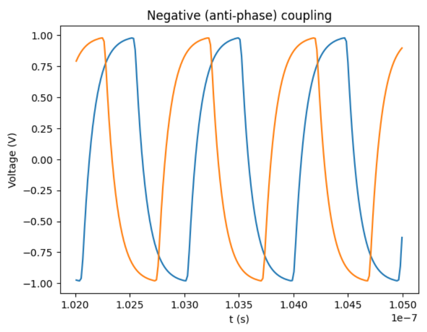
3 New Scientific Results

Thesis I.

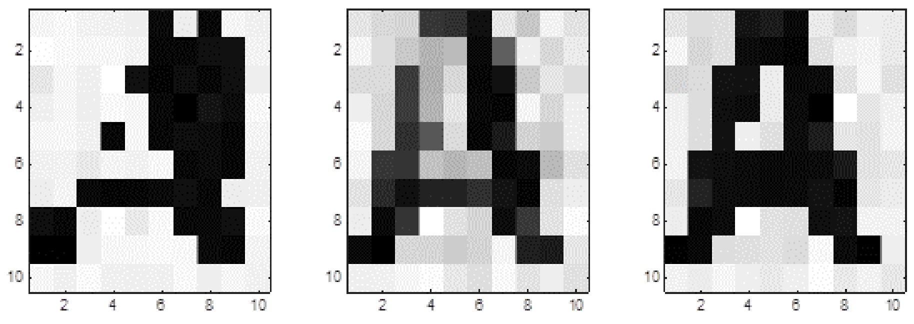
I have developed a computational framework for designing oscillatory neural networks composed of electrical-circuit components and for training them using Backpropagation Through Time (BPTT). I have created two simple architectures: one for wave generation and one mimicking a perceptron, to demonstrate that Oscillatory Neural Networks (ONNs) can implement such devices.



(a) In-phase oscillators



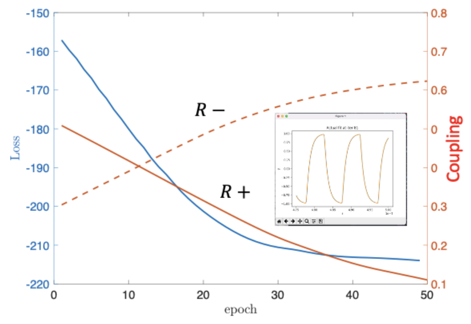
(b) Anti-phase oscillators



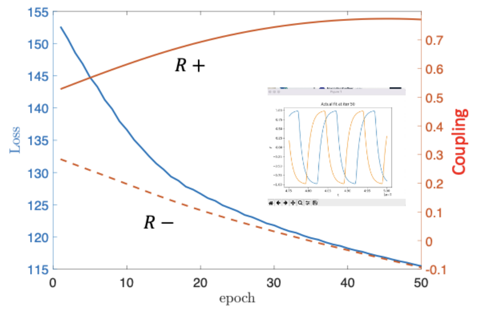
(c) Computing with coupled oscillators to retrieve the written letter ‘A‘

Figure 3.1: **Phase-based computing by two ring oscillators**. On (a) an in-phase configuration is realized, because the two oscillators were connected through the same voltage, i.e., V_3 and V_3, see Fig 2.1. However, on (b) an anti-phase configuration is depicted, where the two oscillators were connected by their different voltage nodes, e.g., V_3 and V_6, see Fig 2.1-it does not have to be necessarily these two nodes, but because of the delay of inverters it has to be different ones and the bigger the gap between the nodes, the bigger will be the phase difference. The oscillators are coupled via resistors, which drive them to the desired phase configurations. If phases correspond to pixels of a grayscale image, the phase dynamics may be used to converge to predefined patterns [21]. The illustrations of the convergence to ‘A’, shown on (c), are taken from [22]

This part of the work was implemented in *PyTorch* to simulate the dynamics of the oscillatory neural network circuit and to train the machine learning algorithm to infer the physical values of the circuit components. Due to the limitations of BPTT — namely that it represents the incremental, numeric solution of the dynamics described by the set of Ordinatry Differential Equation (ODE) of the network as a deep neural network having hundreds or thousands of layers — I simplified the implementation of inverters to a mathematical function, $f(v) = -\tanh(a \cdot v)$, to facilitate learning. The Figures 3.1,2.1,3.2 show how this procedure works in practice.



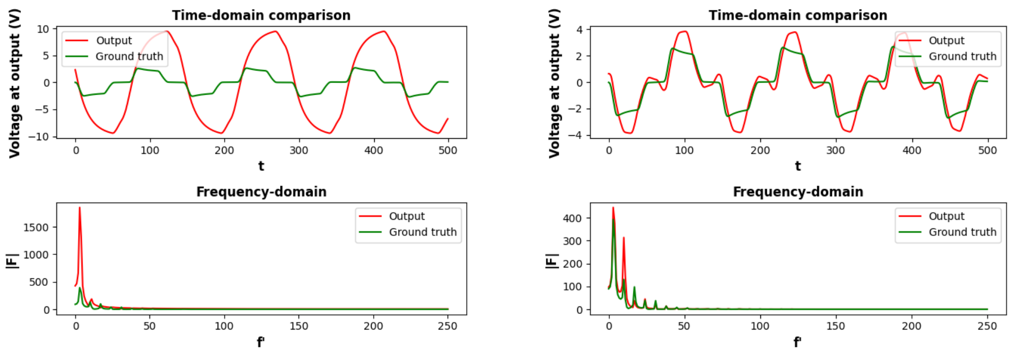
(a) In-phase oscillators



(b) Anti-phase oscillators

Figure 3.2: **Learning the in-phase and anti-phase couplings for a two-oscillator system.** On (a), the simulation's result can be seen for the positively coupled oscillators, meanwhile on (b), there is the same for the negatively coupled 2-oscillator system. The loss changed in both cases from high to low. Additionally, the orange curves indicate the learning parameters in **C** rather than the actual resistor values. Note that in (b) the parameter value corresponding to $R-$ is going below 0, which would mean a negative resistance because of the connection of the parameters in **C** to the physical parameters in this experiment, but this is only the mathematical solution; for a given simulation, the parameters were clamped to be non-negative. If they hit zero, the connection was removed.

Using this framework, I formulated a wave-generation task for 5 oscillators, employing MSE as the loss function. This resulted in near-perfect convergence, and although the problem was simple, this has never been achieved before. This is presented in Figure 3.3.



(a) Start of learning (b) End of learning

Figure 3.3: **Learning a given wave pattern with 5 coupled oscillators.** On (a), the initial output of the sum of the oscillator and the ground truth voltage in the time domain and frequency domain can be seen. In (b), it is evident that convergence occurred, and the algorithm produced a waveform that is qualitatively much closer in both the time and frequency domains than at the start of learning.

Additionally, to test whether conventional neural network components can be reproduced using Oscillatory Neural Networks, I created a simple two-dimensional problem in which I had to separate two blobs of data. To that end, I used 4 oscillators to mimic the behavior of a standard software-based perceptron. The 4-oscillator system I created performed well, achieving nearly 100% accuracy on both the training and test sets. The results are shown on Figures 3.4-3.5..

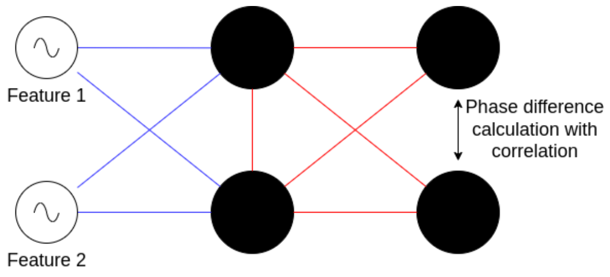


Figure 3.4: **Architecture for binary classification with ring oscillators as perceptrons.** The black-filled circles represent oscillators, while the generators are on the left-hand side, connected to the first two oscillators.

The framework can simulate and train not only ring-oscillator-based neural networks but also other oscillator types, such as the Kuramoto oscillator.

Publications related to this thesis are: [J1], [C1]

Thesis II.

I have designed an Oscillatory Neural Network-based associative memory trained using the Backpropagation Through Time method with both nearest neighbor and fully connected coupling schemes. Both achieved lower Mean Squared Error (MSE) on the given set than a network trained by Hebbian Rule.

This part of the work was conducted to compare ONNs trained with BPTT to ONNs trained with the Hebbian learning rule. One of the most limiting factors for new neuromorphic chips is

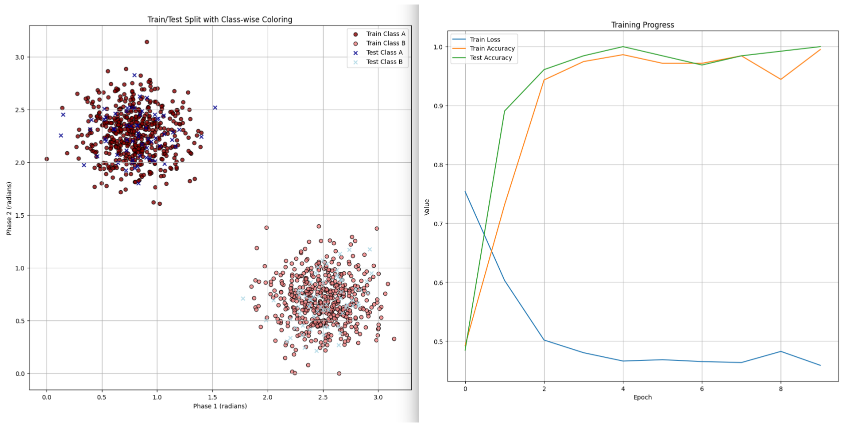


Figure 3.5: **The generated synthetic blobs dataset and the result of training** On the left side, the dataset can be observed with the training and test set having different colors. On the right, the training results across epochs are shown. The network achieved almost 100% accuracy in separating the two blobs, which is a strong result, as it demonstrates that ring oscillator networks can simulate a perceptron for this binary classification task.

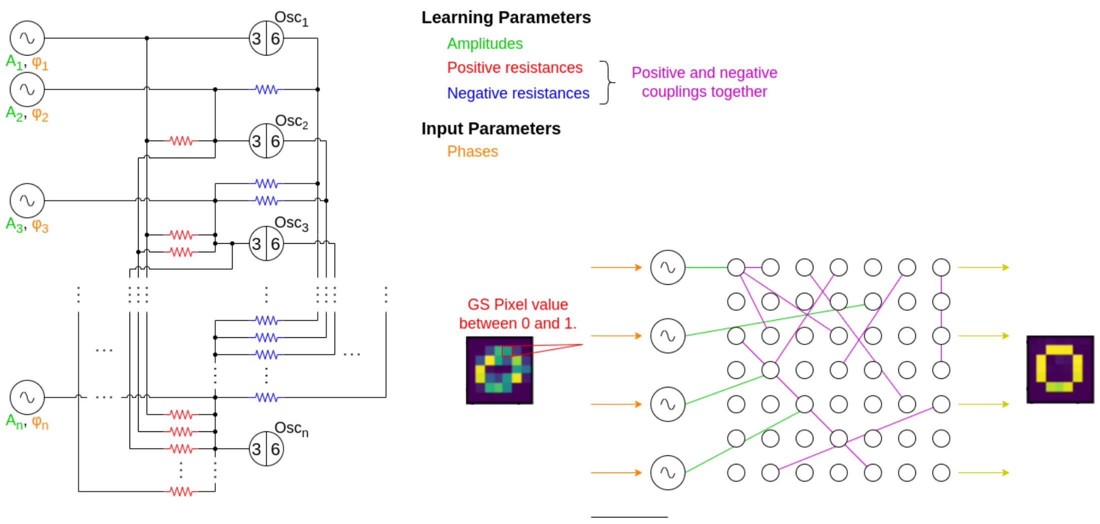


Figure 3.6: **The circuit diagram and logic of the entire computational layer with the input generators**. Input signal generators generate sinusoidal signals whose phases correspond to an input pattern, such as the pixels of an image. These generators are connected to the computing oscillators, whose phase pattern provides the solution to the problem. The $3-6$ marks on the oscillators indicate the input and output nodes in the ring oscillators' circuit. The values of the green, red, and blue-colored circuit elements are learned during the learning process, and the phases indicated in orange are the inputs. In the schematic figure, the purple connections indicate both positive (red) and negative (blue) couplings, as both can occur for optimal convergence. The grayscale pixel value is read from the image, converted into phase information, and then the sinusoidal current generators are connected to the oscillators in a one-to-one mapping. The yellow arrows show that the output is read from the oscillators and an image is formed.

connectivity. Most such architectures use an all-to-all coupling scheme, which is infeasible for physical implementation beyond a certain number of oscillators. To circumvent this limitation, I created not only a fully connected ONN but also a nearest-neighbor connected one by arranging the oscillators on an $N \times N$ grid matching the size of the MNIST images and connecting each oscillator only to its nearest neighbors. In Figure 3.6, this particular setup is visualized.

For this problem, I used a subset of the handwritten digits from the MNIST database and scaled them to either 14×14 or 7×7 images, as the BPTT method can have memory issues with large networks.

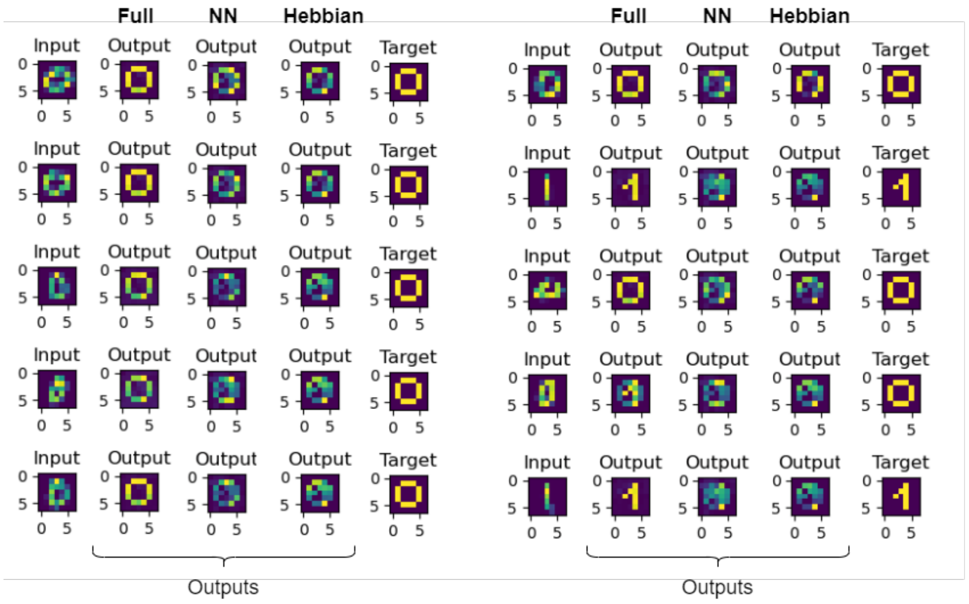


Figure 3.7: **Output association comparison of the differently trained ONNs.** Here, the comparison of the results of the fully connected, nearest neighbor connected, and the Hebbian-rule-based networks can be seen. There are two blocks of 5 inputs, shown side by side. The first column in each block corresponds to the input digit; the next three columns are the outputs of the systems—ordered from left to right: fully coupled, nearest-neighbor coupled, and Hebbian-based. The last column in both blocks shows the target digit. It is apparent that the fully connected system performed best, but even our other proposed nearest-neighbor connected topology outperformed the Hebbian-based architecture.

The results were promising: both the fully connected and nearest-neighbor connected BPTT-trained ONNs outperformed the Hebbian-rule-trained ONN, demonstrating that it is not necessary to use an all-to-all coupled system. Not only did the BPTT-trained networks outperform in terms of quantitative accuracy, meaning that the resulting patterns from the BPTT-trained networks were closer to the real patterns in terms of MSE, but also in terms of parameter count, as the nearest neighbor connected ONN used much fewer parameters than the fully connected, Hebbian-rule-based network. This is mainly because the Hebbian rule assumes all-to-all connectivity. Quantitative comparisons of the results are presented in Table 3.1, and qualitative comparisons are shown in Figure 3.7.

Method	Hebbian	Proposed fully connected	Proposed NN-connected
#Params	1176	2352	312
MSE	0.068	0.020	0.047

Table 3.1: **Parameter count and MSE performance comparison for the different training methods on ONNs.** The MSEs of all the elements from the set and their respective ground truths for the different methods in the case of associative learning. It is apparent that the fully connected network performed the best, but even the nearest neighbor connected layer is good enough to beat the Hebbian learning in terms of quantitative association.

Publications related to this thesis point are: [J1], [C1]

Thesis III.

I have designed an ONN-based architecture for binary and multiclass classification on the MNIST dataset. I have created a two-layer ONN for binary classification of the 0 *and* 1 *classes, with all-to-all and nearest-neighbor couplings in the hidden layer, and compared the results. Managed to reach, on average,* 99% *accuracy and observed no consistent patterns in the hidden layer, confirming that the algorithm learned structures that are not necessarily recognizable to the human eye. I have also developed several purely ONN architectures for multiclass classification on the MNIST dataset and compared their performance with a standard neural network. In terms of accuracy, it is behind State-of-the-Art (SOTA) solutions, but in terms of energy efficiency, ONN-based approaches pull ahead.*

Although state-of-the-art solutions for MNIST classification achieve over 99% accuracy, they often have hundreds of thousands, or even millions, of parameters. To reduce this number, it is possible to leverage ONN dynamics.

Classification is a standard task in artificial intelligence. Dynamic systems, such as oscillatory neural networks, may not be the best choice for classifying static data. This is mainly because, in that case, the oscillators' inherent dynamics are not fully utilized. That said, classification remains possible using ONNs.

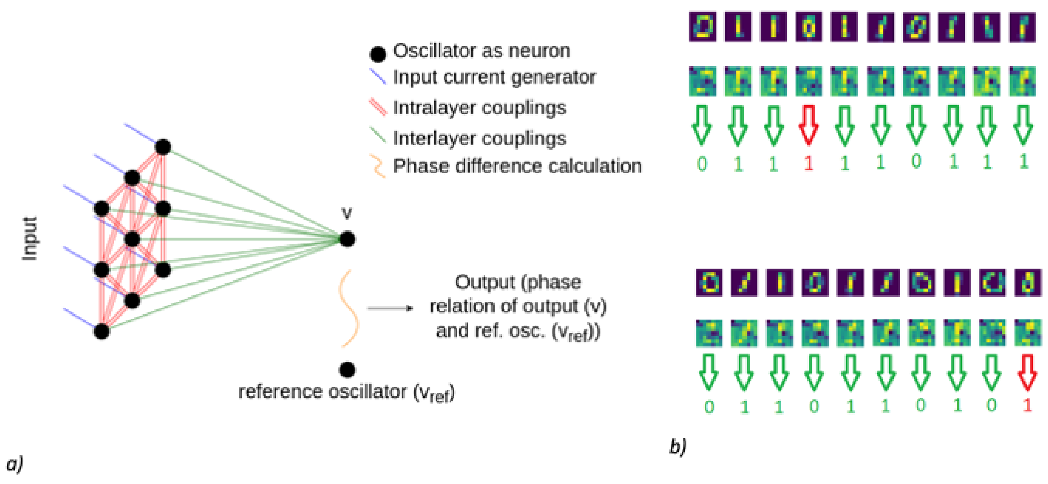


Figure 3.8: **A simple two-layer classifier showing also the patterns forming in the hidden layer.** On the left, the architecture can be seen that I used for binary classification. It might appear to be in a Feedforward-like arrangement, but this is merely a clearer visual representation. In reality, all the oscillators were connected bilaterally. On the right side, samples from the binary classification of 0 and 1 are shown, along with input, hidden-layer pattern, and output-prediction triplets. The two red numbers indicate the network's prediction errors. I want to highlight that, in some cases, there is an apparent structure in the hidden layer, as some phase patterns there resemble either 0 or 1, but this is not universal; at other times, there is no apparent, humanly visible structure at this stage of the architecture.

In Figure 3.8, it can be seen that a network that resembles regular artificial neural networks but assumes bilateral flow of information can be used to distinguish between binary classes of MNIST. On the figure, I also showcased the ability of the network to classify 0 and 1. It was not always true, but sometimes there existed a humanly recognizable structure inside the hidden layer. The quantitative performance of the net was excellent, achieving 99% prediction accuracy on the test set.

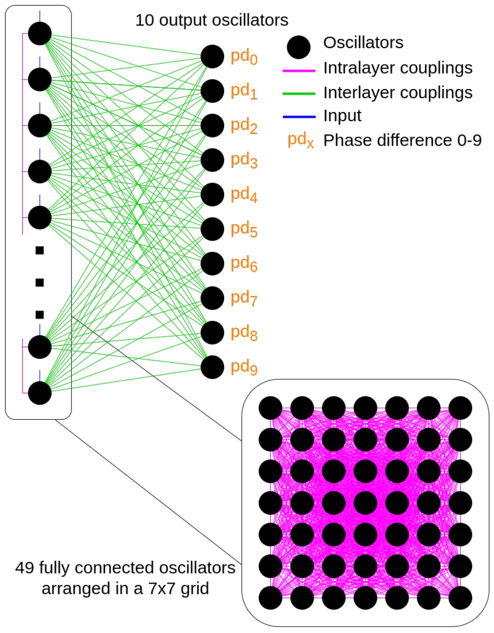


Figure 3.9: **Multilayer network for multiclass classification of MNIST.** The structure is similar to the one shown in Figure 3.8, but the output layer has 10 oscillators instead of just one. Additionally, the hidden layer is flattened here to improve overall network clarity, while its gridlike structure is depicted in the bottom-right corner. The pd_x values represent the phase difference of the output oscillators and a reference oscillator. These pd_x values will be the input of the cross-entropy loss function alongside the real class values.

For multiclass prediction, I tried two approaches: a regular ANN-like architecture, as shown in Figure 3.9, and a distributed, binary-classifier-based solution with ten competing networks. The logic of the latter can be seen in Figure 3.10.

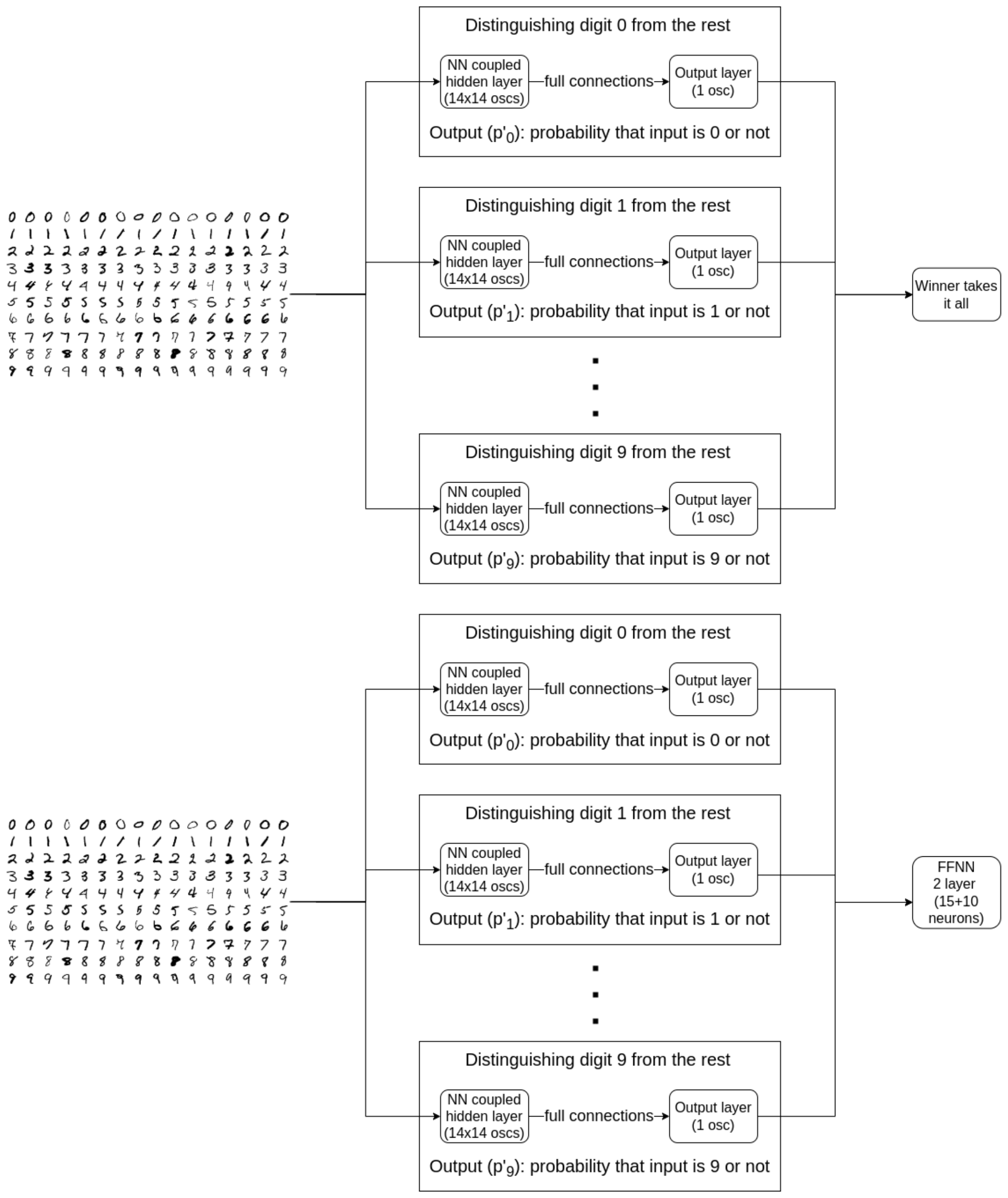

Distinguishing digit 0 from the rest
NN coupled hidden layer (14x14 oscs)
full connections
Output layer (1 osc)
Output (p'$_0$): probability that input is 0 or not
Distinguishing digit 1 from the rest
Output (p'$_1$): probability that input is 1 or not
Distinguishing digit 9 from the rest
Output (p'$_9$): probability that input is 9 or not
Winner takes it all
FFNN 2 layer (15+10 neurons)

Figure 3.10: **Two of the three tested architectures for the time-independent MNIST classification.** Both consist of individually trained, nearest-neighbor-connected subnetworks, which were designed to distinguish between a single class and the rest of the classes using a binary cross-entropy loss function. The top block diagram illustrates the algorithm: to select the prediction, we take the maximum of the individual network outputs. The more sophisticated version is shown in the bottom block diagram. Here, I used the individual classifiers' output probabilities as inputs to a small, regular FFNN and trained it as a 10-class classification problem using cross-entropy.

Unfortunately, the performance of the purely ONN-based networks did not reach the desired levels, so I moved on with a combination of ONN and Artificial Neural Network. The comparison of such networks can be seen in Table 3.2.

	ONN	**WTIA**	**Augmented**	**MLP**
#Param	40180	16000	16325	16363
Perf. (%)	72.3 (70–75)	66.7 (65–70)	95.1 (93–97)	94.4 (93–95)

Table 3.2: **Quantitative comparison of multi-class classifiers.** 'Perf. (%)' is in the format of 'mean (range)'. The Augmented network uses a two-layer MLP as the final decision function. Note that the reported parameter count for the pure ONN version refers to the version with a 14×14 sized hidden layer, because the 7×7 layer performed extremely poorly.

Publications related to this thesis point are: [J1], [C2], [C3]

Thesis IV.

I have created a frequency-based oscillatory neural network using both the Kuramoto and ring-oscillator-based models. Managed to tune the parameters of the networks either manually or with the utilization of a gradient-free optimization method. My network was capable of solving the task at hand with 100% *accuracy using both models and parameter tuning schemes.*

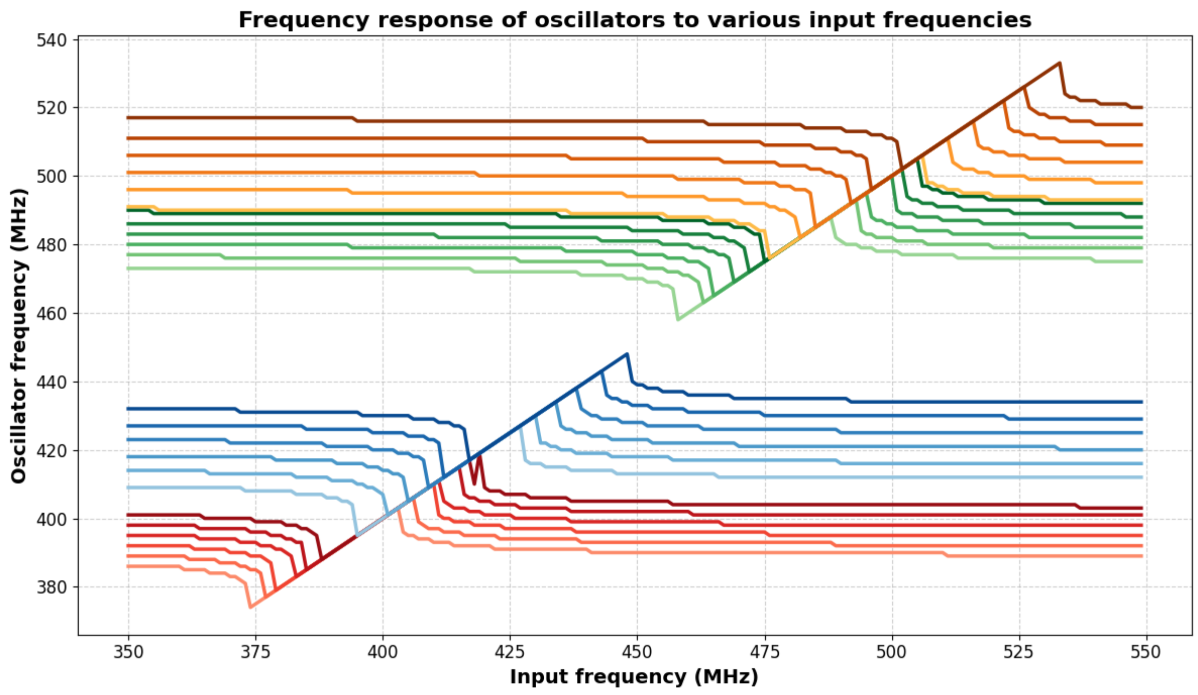


Figure 3.11: **Frequency synchronization of oscillators to input sinusoid with certain frequencies.** A frequency sweep over a range of frequencies for a single input sinusoid voltage generator connected to a 24-uncoupled oscillator network can be seen. The four oscillator groups either synchronize to the input signal or do not, depending on the input frequency. In each group, the oscillators synchronize to the input signal's frequency one-by-one, until all of them are synchronized, then they run together for a specific frequency range, and then they depart in frequency from the rest of the group one-by-one again.

In literature, oscillators are mainly used as phase-based computers. That said, they can be used as frequency-based architectures that exploit their frequency-synchronization property, as depicted in Figure 3.11.

Building upon this principle, it is possible to design an architecture, depicted in Figure 3.12, with a limited number of parameters, as can be seen in Table 3.3, and it can be used to distinguish between two vowels represented by their two most dominant formant frequencies.

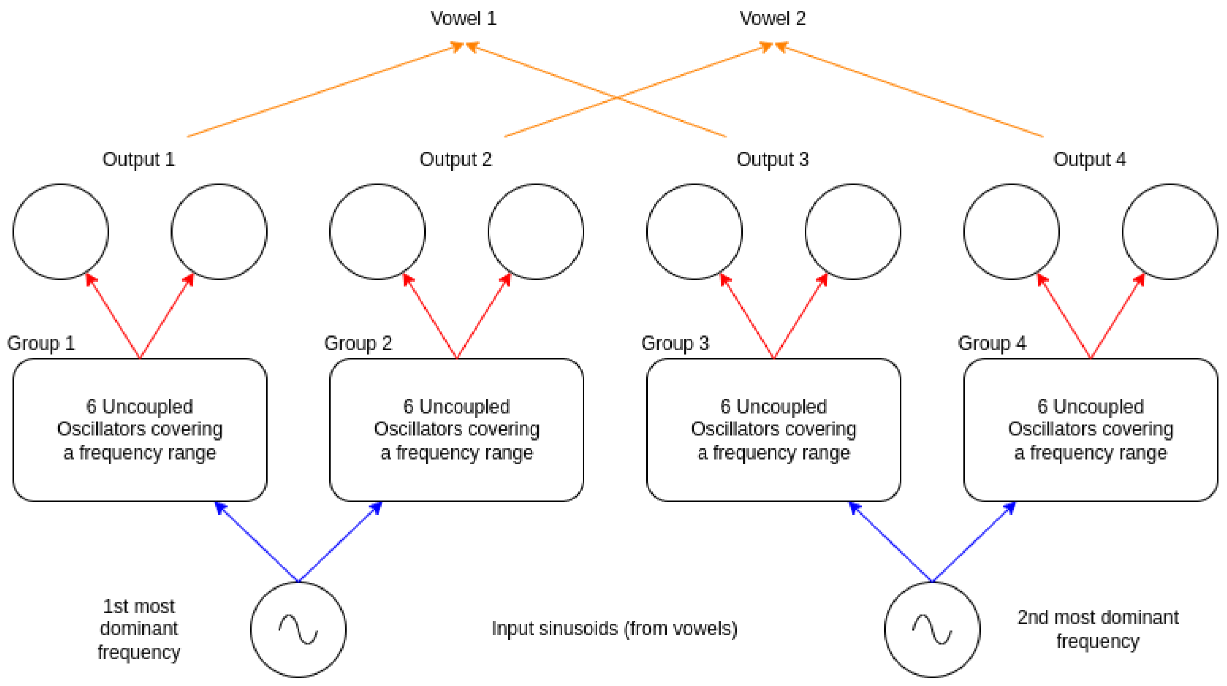

Vowel 1
Vowel 2
Output 1
Output 2
Output 3
Output 4
Group 1
Group 2
Group 3
Group 4
6 Uncoupled Oscillators covering a frequency range
1st most dominant frequency
Input sinusoids (from vowels)
2nd most dominant frequency

Figure 3.12: **My proposed architecture.** The input signals are represented as sinusoid voltage generators. The frequency of these generators is set by the first and second most dominant frequency components of a given input vowel. These are connected to a single layer of unconnected oscillators, grouped into 4 distinct groups to highlight their collective responsibility for detecting a specific frequency in the input signal. The groups are synchronized in frequency if the input sinusoid lies within the group's frequency range. These groups are then connected to a pair of oscillators covering the same frequency range. If the 6 oscillators in either group are synchronized in frequency, then the pair of oscillators in the output layer will be synchronized in frequency as well. This way, certain frequencies can be signalled by different groups found in the input sinusoids.

Table 3.3: **Neuron, coupling, and parameter counts for the proposed multilayer oscillatory network.** The network is lightweight with only 72 couplings — including the inputs' couplings to the oscillators. Note that the network's real parameter count is the sum of the number of oscillators and couplings, as the former sets the oscillators' frequencies and the latter sets the coupling strengths between connected oscillators.

	Input	***Layer #1***	***Layer #2***	**All**
# of neurons	2	24	8	34
# of frequencies	0	24	8	32
# of couplings to next	24	48	0	72
# of parameters	24	72	8	104

I successfully distinguished between the vowels "ah" and "it" using this network with 100% accuracy, further demonstrating that a frequency-based architecture is feasible.

Additionally, it provided a useful parallel to the phase-based perceptron I introduced earlier, as this network is somewhat equivalent to that architecture.

The tuning of the parameters, as shown in Table 3.4, is performed using both manual tuning and a gradient-free optimization method to find the optimal parameters of the oscillatory neural network. This method was employed to test its capabilities because the BPTT method can pose significant challenges when the architecture reaches a point at which gradient-based methods would consume an enormous amount of memory.

Table 3.4: **Physical parameters of the ring oscillator network for formant recognition**. All these parameters were set by trial-and-error and used for the simulations.

Parameter	**Physical Value**
C	$1.0 \cdot 10^{-13}$ F
R	$2.0 - 2.7 \cdot 10^{3}\ \Omega$
R_{in}	$1.0 \cdot 10^{4}\ \Omega$
R_c	$1.2 - 2.2 \cdot 10^{6}\ \Omega$
V_{in}	1 V

Publications related to this thesis point are: [J2], [C4].

Thesis group V.

I have successfully demonstrated, in two tasks—MNIST and vowel recognition—and with both phase- and frequency-based computing, that ONNs can be used effectively as preprocessing layers when combined with standard neural networks. This emphasizes the power of ONNs in terms of energy efficiency and the ability to handle large-scale problems.

Unfortunately, pure oscillator-based networks did not perform well enough. However, combining the ONN with a standard ANN can achieve performance comparable to standard neural networks.

Thesis V.1: *The best ONN network I designed for the MNIST classification task was combined* *with a standard ANN, yielding a hybrid ONN-ANN architecture that achieved an average accuracy of* 95.1% *on a 10-class classification task. Comparing this to a regular, an FFNN with the same parameter size achieved an average* 94.4% *accuracy on the same problem.*

Due to the purely oscillator-based network's inability to reach a high percentage on the MNIST dataset, even with a distributed, competing networks and the winner takes it all algorithm, I decided to move forward with an approach borrowed from Convolutional Neural Networks, meaning that after a couple of layers, there is a fully-connected multi-layered perceptron to take care of the classification.

The architecture I used is shown in Figure 3.13, and a comparison of all multiclass classification networks is presented in Table 3.2.

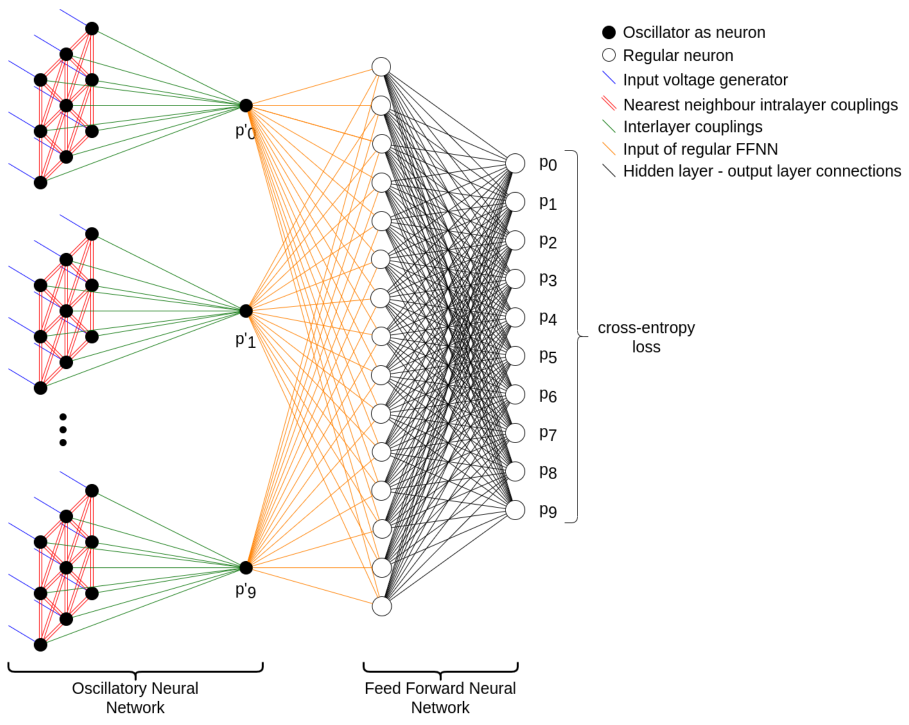


Figure 3.13: **A network with ONN layers as pre-processors and a traditional neural network working tandem**. The easy-to-train output layer significantly improves classification accuracy while keeping the network simpler than state-of-the-art methods. Moreover, the bulk of the work is still performed by the ONNs as pre-processors rather than by the small neural network at the end.

I was able to reach at most 96.7% classification accuracy with this combined network. In comparison, a similarly sized, standard neural network achieved at most 95% accuracy, which is a strong result given that the network was trained for only a few epochs due to computational constraints and was very small in terms of the number of parameters.

Publications related to this thesis point are: [J1], [C2], [C3]

Thesis V.2: *I have created a mixed network using ONNs as preprocessors and a Multi-layered Perceptron (MLP) as the output layer to recognize spoken vowels from their original time-dependent waveforms. On the problem set, the algorithm achieved* 95% *accuracy, and* *the main advantage of this solution is that the network is low-complexity and computationally lightweight. This is because the ONN layer and the transforming layer that converted the ONN's output signal to static data were built using only simple electrical components.*

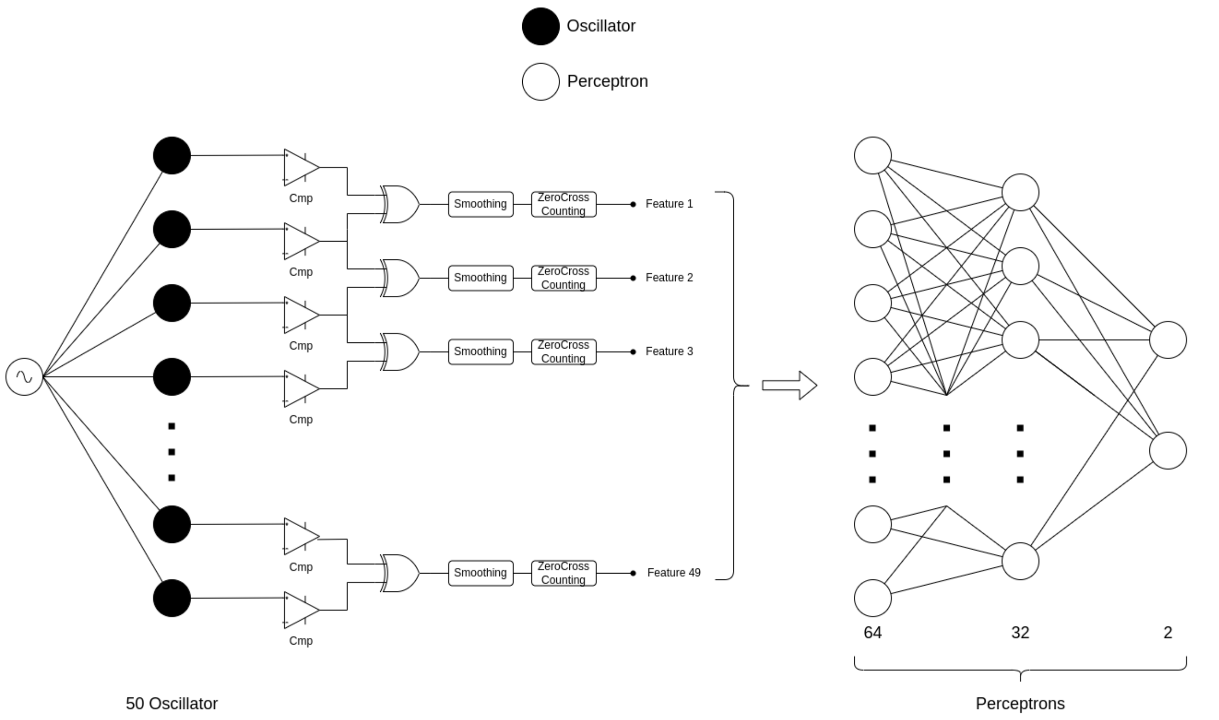
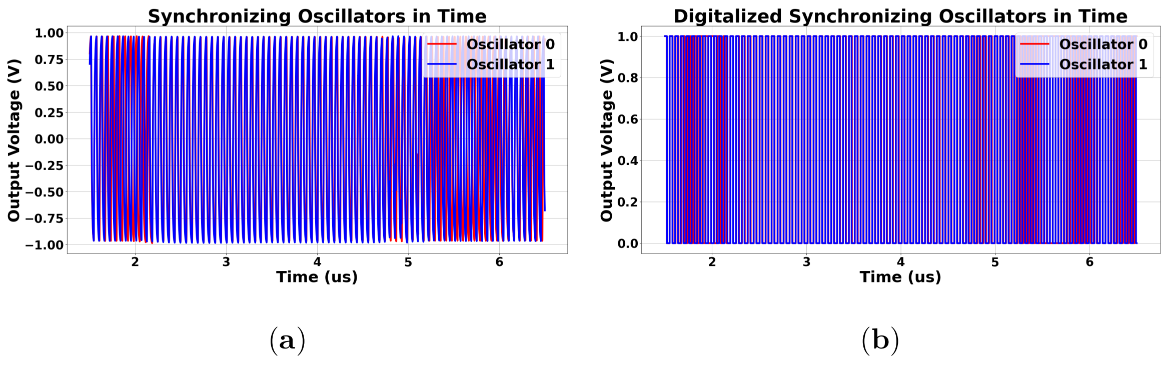


Figure 3.14: **Mixed architecture for time-domain binary vowel recognition.** The first part of the architecture extracts the frequency features of the time-domain vowels by measuring the frequency synchronization strength of neighboring oscillators. The second part uses this feature set as input data to perform the actual binary classification with a simple multi-layer perceptron architecture.

Building on the success of the combined ONN and ANN, I created a network to classify vowels. For this task, I used time-dependent vowel waveforms rather than the dominant formants. The first, oscillatory layer was responsible for extracting frequency information from the vowels by creating a filter-like layer.



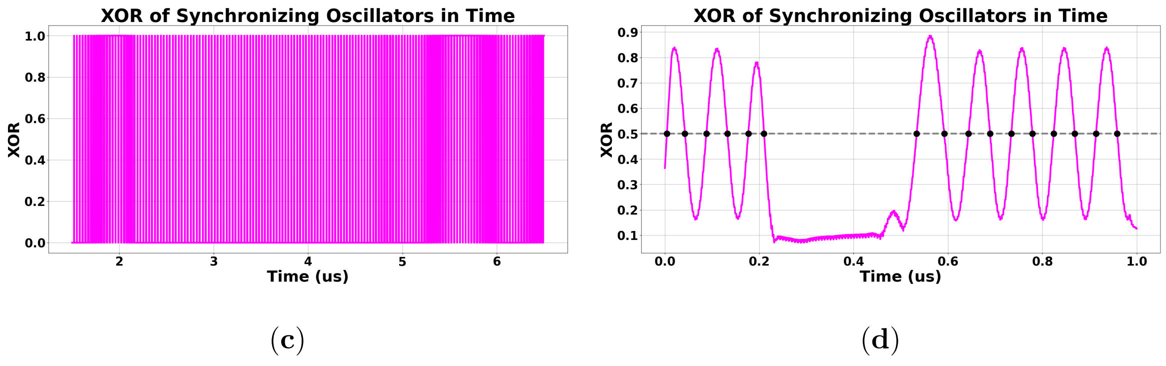


Figure 3.15: **Transformations on the output signals of two neighboring synchronizing oscillators.** The transformations conducted on the two oscillator signals via the small circuit after the oscillators can be seen on (**a**) to (**d**): (**a**) output voltages of synchronizing oscillators near the synchronization time frame, and it can be observed that in the synchronizing region the two oscillators frequencies are not changing relative to other, but before and after that oscillator 1 is constantly moving over oscillator 0 as the higher frequency oscillator; (**b**) after digitizing the output signals, two square signals can be observed; (**c**) the XOR of the synchronizing signals shows similar behavior than (**a**) as the XOR is changing with constant duty signal in the synchronization region but outside of it the duty signal's length is constantly changing; (**d**) after applying a mean filter for the whole XOR signal, the synchronization region can be seen clearly and if zero-crossing are counted on this signal it will result in a lower value than if this synchronization had not been present as it can be seen that outside of this region the smoothed XOR signal is sinusoid-like. Note that I inverted this value and transformed it to the $[0, 1]$ range to obtain the relative signal strength. In this way, it is more intuitive to refer to it as synchronization strength.

This frequency representation is a downsampled version of the signal's FFT, and the features associated with a single vowel have been extracted using only simple electrical circuit components, as shown in Figure 3.14. The feature extraction process is shown in Figure 3.15, and the resulting features are demonstrated in Figure 3.16.

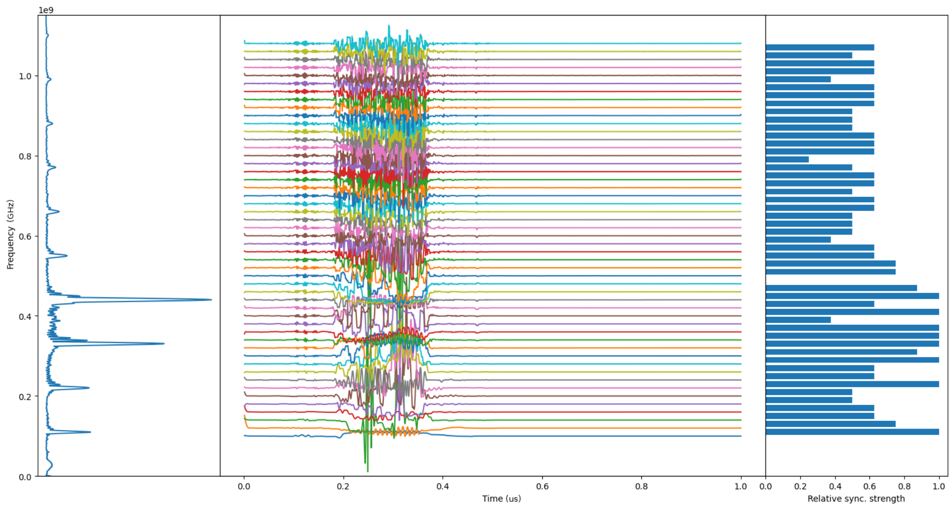


Figure 3.16: Example of feature extraction with oscillatory layer. The feature vector element values are high when synchronization is present between two oscillators and low when it is absent, as desired. The frequency spectra of the vowel are on the left, rotated 90 degrees for clarity. The colored curves are the instantaneous frequencies for each oscillator.

I tested the same vowels, "ah" and "it", and achieved 95% classification accuracy. This is a good result, as most of the literature treats time-dependent audio signals as either one-dimensional sequential data or converts them into representations such as Mel spectrograms and then uses image recognition networks.

The approach I presented offers a new way to reduce the complexity of such classifiers and paves the way for architectures that can receive data directly from sensors.

Publications related to this thesis are : [J2], [C4]

Acknowledgements

I would first like to express my deepest gratitude to my supervisor, Professor György Csaba, for his invaluable guidance, encouragement, and unwavering support over the past 8 years. His insights and expertise have not only shaped this work but also played a crucial role in the researcher I have become.

I am also grateful to Professor Wolfgang Porod at the University of Notre Dame for the opportunity to collaborate with him on parts of this research. His extensive knowledge and the experiences he shared during our collaboration were inspiring.

My sincere thanks also go to the Intel team, in particular Narayan Srinivasa, Dmitri Nikonov, and Amir Khosrowshahi, for their regular guidance and discussions. They helped me shape my vision of the practical applicability of the proposed solution and provided valuable perspectives on the current state of the art.

I would also like to acknowledge the PHASTRAC consortium for our biweekly meetings and the constructive feedback I received. In particular, I am grateful to the team at TU Eindhoven, led by Professor Aida Todri-Sanial, for their close collaboration on time-dependent signal processing in electric circuits, as well as to the IBM and BMW teams for their insights and comments.

I am very grateful for the sense of community I experienced among my fellow PhD students at the university. In particular, I would like to thank Gábor Dániel Balogh, András Attila Sulyok, Bálint Siklósi, and Mihály András Vághy, who were always ready for a chat—whether about research or about life beyond it.

I owe special thanks to the Faculty of Information Technology and Bionics at Pázmány Péter Catholic University for the opportunity to take part in the doctoral program. I am especially indebted to Professor Gábor Szederkényi and Professor Árpád Csurgay for their guidance and regular feedback during the preparation of this dissertation. Likewise, I am also profoundly grateful to all my colleagues at Pázmány Péter Catholic University, whose support and friendship made these years far more enjoyable. In particular, I would like to thank Dr. Tivadarné Vida Katinka for her kindness and constant willingness to help. Her smile and encouragement were a source of strength, even when the challenges seemed overwhelming.

I am profoundly grateful to my mother, whose love, encouragement, and unwavering support have shaped the person I am today. Finally, and most importantly, I wish to express my deepest thanks to Lenke Rudner-Halász, who has stood by me throughout these years, offering her love, patience, and encouragement. Her presence repeatedly reminded me that I could not have accomplished this without her.

List of author publications

List of Journal Publications

[J1] T. Rudner, W. Porod, and G. Csaba, “Design of oscillatory neural networks by machine learning,” *Frontiers in Neuroscience*, vol. 18, p. 1 307 525, Mar. 2024. DOI: `10.3389/fnins.2024.1307525` (cit. on pp. 10, 13, 17, 20).

[J2] T. Rudner-Halász, W. Porod, and G. Csaba, “Oscillator-based processing unit for formant recognition,” *Information*, vol. 16, no. 7, 2025, ISSN: 2078-2489. DOI: `10.3390/info16070611`. [Online]. Available: `https://www.mdpi.com/2078-2489/16/7/611` (cit. on pp. 19, 23).

List of Conference Posters and Presentations

[C1] T. Rudner, G. Csaba, and W. Porod, “Design of oscillatory neural networks by machine learning algorithms,” in *International Workshop on Computational Nanotechnology*, Conference presentation, 2023 (cit. on pp. 10, 13).

[C2] T. Rudner, W. Porod, and G. Csaba, *Design of oscillatory neural networks by machine learning techniques*, Poster presented at the Workshop “Frontiers of Neuromorphic Computing”, 2023 (cit. on pp. 17, 20).

[C3] T. Rudner, W. Porod, and G. Csaba, *Design of oscillatory neural networks by machine learning*, Poster presented at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2023 (cit. on pp. 17, 20).

[C4] T. Rudner, W. Porod, and G. Csaba, *Training oscillator networks for neuromorphic computing*, Poster presented at the International Conference on Neuromorphic Computing and Engineering, 2024 (cit. on pp. 19, 23).

References

[1] G. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998. DOI: 10.1109/jproc.1998.658762 (cit. on p. 2).

[2] G. E. Moore, "Progress in digital integrated electronics [technical literature, copyright 1975 ieee. reprinted, with permission. technical digest. international electron devices meeting, ieee, 1975, pp. 11-13.]," *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 36–37, Sep. 2006, ISSN: 1098-4232. DOI: 10.1109/N-SSC.2006.4804410 (cit. on p. 2).

[3] C. A. Mack, "Fifty years of moore's law," *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 2, pp. 202–207, May 2011, ISSN: 1558-2345. DOI: 10.1109/TSM.2010.2096437 (cit. on p. 2).

[4] D. E. Nikonov and I. A. Young, "Overview of beyond-cmos devices and a uniform methodology for their benchmarking," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2498–2533, Dec. 2013, ISSN: 1558-2256. DOI: 10.1109/JPROC.2013.2252317 (cit. on p. 2).

[5] S. Manipatruni, D. E. Nikonov, and I. A. Young, "Material targets for scaling all spin logic," *ArXiv*, vol. abs/1212.3362, 2012 (cit. on p. 2).

[6] B. Behin-Aein, D. Datta, S. Salahuddin, and S. Datta, "Proposal for an all-spin logic device with built-in memory," *Nature nanotechnology*, vol. 5, pp. 266–70, Feb. 2010. DOI: 10.1038/nnano.2010.31 (cit. on p. 2).

[7] K. Zuse, "The computing universe," *Int. J. Theor. Phys*, vol. 21, pp. 589–600, 6-7 1982. DOI: 10.1007/BF02650187 (cit. on p. 2).

[8] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines," *Journal of Statistical Physics*, vol. 22, no. 5, pp. 563–591, May 1980, ISSN: 1572-9613. DOI: 10.1007/BF01011339. [Online]. Available: https://doi.org/10.1007/BF01011339 (cit. on p. 2).

[9] C. Kim, "Coupled oscillator based computing: Using nature to solve difficult problems," *IBM Unconventional Computing Paradigm Workshop*, 2021 (cit. on p. 2).

[10] H. Cılasun et al., "3sat on an all-to-all-connected cmos ising solver chip," *Scientific Reports*, vol. 14, no. 1, p. 10 757, 2024. DOI: 10.1038/s41598-024-60316-y. [Online]. Available: https://doi.org/10.1038/s41598-024-60316-y (cit. on p. 3).

[11] C. Bybee, D. Kleyko, D. E. Nikonov, A. Khosrowshahi, B. A. Olshausen, and F. T. Sommer, "Efficient optimization with higher-order ising machines," *Nature Communications*, vol. 14, no. 1, p. 6033, 2023. DOI: 10.1038/s41467-023-41214-9. [Online]. Available: https://doi.org/10.1038/s41467-023-41214-9 (cit. on p. 3).

[12] V. Clerico et al., "Edge training and inference with analog reram technology for hand gesture recognition," *arXiv preprint arXiv:2502.18152*, 2025. DOI: 10.48550/arXiv.2502.18152. [Online]. Available: https://doi.org/10.48550/arXiv.2502.18152 (cit. on p. 3).

[13] X. Lai and J. Roychowdhury, "Analytical equations for predicting injection locking in lc and ring oscillators," in *Proceedings of the IEEE 2005 Custom Integrated Circuits Conference, 2005.*, San Jose, CA, USA: IEEE, 2005, pp. 454–457, ISBN: 9780780390232. DOI: 10.1109/CICC.2005.1568706. Accessed: Oct. 2, 2023. [Online]. Available: http://ieeexplore.ieee.org/document/1568706/ (cit. on p. 4).

[14] Y. Kuramoto, "Self-entrainment of a population of coupled non-linear oscillators," *International Journal of Engineering Science*, vol. 16, no. 11, pp. 1187–1206, 1975 (cit. on p. 6).

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14539. Accessed: Oct. 2, 2023. [Online]. Available: https://www.nature.com/articles/nature14539 (cit. on p. 6).

[16] A. Paszke et al., *Automatic differentiation in pytorch*, Long Beach, CA, USA, 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ (cit. on p. 6).

[17] R. T. Q. Chen, *Torchdiffeq*, 2018. [Online]. Available: https://github.com/rtqichen/torchdiffeq (cit. on p. 6).

[18] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems*, 2018 (cit. on p. 6).

[19] T. P. Lillicrap and A. Santoro, "Backpropagation through time and the brain," *Current opinion in neurobiology*, vol. 55, pp. 82–89, 2019 (cit. on p. 7).

[20] J. Rapin and O. Teytaud, *Nevergrad - A gradient-free optimization platform*, https://GitHub.com/FacebookResearch/Nevergrad, 2018 (cit. on p. 7).

[21] G. Csaba and W. Porod, "Noise immunity of oscillatory computing devices," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, no. 2, pp. 164–169, Dec. 2020, ISSN: 2329-9231. DOI: 10.1109/JXCDC.2020.3046558. Accessed: Oct. 2, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9302732/ (cit. on p. 8).

[22] G. Csaba, T. Ytterdal, and W. Porod, "Neural network based on parametrically-pumped oscillators," in *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Monte Carlo, Monaco: IEEE, Dec. 2016, pp. 45–48, ISBN: 9781509061136. DOI: 10.1109/ICECS.2016.7841128. Accessed: Oct. 2, 2023. [Online]. Available: http://ieeexplore.ieee.org/document/7841128/ (cit. on p. 8).