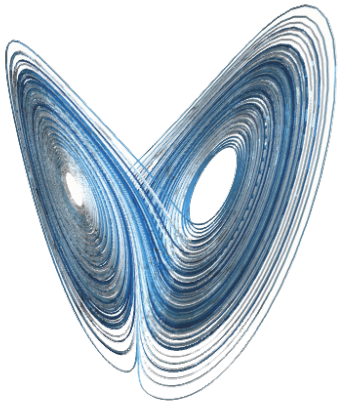
TÉCNICO LISBOA



Modeling Dynamic Systems using Machine Learning Techniques

Eduardo de Azevedo Soares Mineiro Rodrigues

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Luís Miguel Teixeira D'Avila Pinto da Silveira
Dr. Ruxandra Georgeta Barbulescu

Examination Committee

Chairperson: Prof. João Manuel de Freitas Xavier
Supervisor: Dr. Ruxandra Georgeta Barbulescu
Member of the Committee: Prof. Alexandra Sofia Martins de Carvalho

November 2023

Dedicated to my grandfather José Eduardo Pereira Rodrigues

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First and foremost, words cannot adequately express my deep gratitude to my advisor, Prof. Luís Miguel d'Ávila Silveira. His profound technical knowledge, unwavering patience, and invaluable assistance throughout this work have been instrumental in its success.

I would also like to extend my heartfelt appreciation to Ruxandra Barbulescu, who generously provided me with key articles and implementation scripts that proved indispensable for this thesis. Additionally, I am grateful to my classmates for their constant presence, help and support throughout these years.

Last but not least, I would be remiss if I didn't acknowledge the support of the university, Instituto Superior Técnico (IST), and INES-ID. Their facilities and computing capacity enabled me to conduct the necessary simulations to pursue this thesis.

Resumo

Esta investigação teve como objetivo modelar sistemas dinâmicos utilizando técnicas de aprendizagem automática e investigar a sua aplicabilidade na captura do comportamento complexo de organismos, processos e sistemas do mundo real. Para simular as respostas dinâmicas destes sistemas a vários estímulos, foram utilizados circuitos eléctricos como *proxies*. Estes circuitos forneceram um meio para compreender o comportamento dinâmico subjacente, prever reacções futuras, e fazer a ponte entre as complexidades dos fenómenos do mundo real e a modelação computacional. Os modelos *white-box*, que permitem previsões precisas e oferecem uma visão dos fenómenos internos que internos que determinam as respostas do sistema, foram inicialmente explorados. No entanto, à medida que os sistemas se tornavam cada vez mais complexos complexos ou exigiam pormenores excessivos, estes modelos tornaram-se impraticáveis. Consequentemente, os modelos *black-box*, que dão prioridade à previsão exacta em detrimento da compreensão interna, surgiram como uma alternativa mais adequada. As técnicas de aprendizagem automática destacaram-se na construção de modelos *black-box* através da observação de dados de entrada-saída e inferindo o comportamento do sistema. Este trabalho baseou-se numa tese de doutoramento dos anos 90 que investigou a aplicabilidade de utilizar redes neuronais para modelar dispositivos e subcircuitos eletrónicos na simulação de circuitos. Várias arquiteturas de redes neurais, incluindo redes neurais padrão, redes neurais dinâmicas, redes neurais recorrentes e redes neurais recorrentes de tempo contínuo. Esses modelos foram testados em diferentes circuitos conhecidos pelo seu comportamento dinâmico, com o objetivo de explorar a fiabilidade do uso desses modelos para capturar comportamento não-linear exibido por organismos, processos e sistemas do mundo real. Ao longo da tese, foi efectuada uma análise comparativa dos diferentes modelos de redes neuronais, destacando-se os respectivas vantagens e desvantagens. Os conhecimentos adquiridos com esta comparação permitiram extrair conclusões valiosas sobre o desempenho e a adequação de cada modelo para captar o comportamento dinâmico dos sistemas estudados. No geral, esta investigação contribui para a compreensão da modelação de sistemas dinâmicos através de técnicas de aprendizagem de máquina. Ao utilizar circuitos eléctricos como proxies e ao partir do trabalho de doutoramento do Peter Meyer, esta tese demonstra o potencial das abordagens de modelação *black-box* para modelar com precisão o comportamento complexo de organismos, processos e sistemas do mundo real, abrindo portas para as mais diversas aplicações em domínios científicos e de engenharia.

Palavras-chave: Sistemas dinâmicos, aprendizagem automática, modelação de circuitos elétricos, arquitecturas de redes neuronais, previsão sequência-a-sequência, modelação entrada-saída, modelação *black-box*

Abstract

This research aimed to model dynamic systems using machine learning techniques and investigate their applicability in capturing the complex behavior of real-world organisms, processes, and systems. To simulate the dynamic responses of these systems to various stimuli, electric circuits were employed as proxies. These circuits provided an avenue to comprehend the underlying behavior, predict future reactions, and bridge the gap between the intricacies of real-world phenomena and computational modeling.

White-box models, which enable accurate predictions and offer insights into the internal phenomena driving system responses, were initially explored. However, as systems grew increasingly complex or demanded excessive detail, these models became impractical. Consequently, black-box models, which prioritize accurate prediction over internal understanding, emerged as a more suitable alternative. Machine learning techniques excelled in constructing black-box models by observing input-output data and inferring the system's behavior.

This work built upon a PhD thesis from the 1990s that investigated neural network applications for device and subcircuit modeling in circuit simulation. Various neural network architectures, including standard neural networks, dynamic neural networks, recurrent neural networks, and continuous-time recurrent neural networks, were employed. These models were tested on different circuits known for their dynamic behavior, aiming to exploit the practicality of using such models to capture the nonlinear behavior exhibited by real-world organisms, processes, and systems.

Throughout the thesis, a comparative analysis of the different neural network models was conducted, highlighting their respective strengths and weaknesses. Insights gained from this comparison enabled the extraction of valuable conclusions regarding the performance and suitability of each model in capturing the dynamic behavior of the studied systems.

Overall, this research contributes to the understanding of modeling dynamic systems through machine learning techniques. By leveraging electric circuits as proxies and drawing upon the advancements made since the earlier thesis, this work demonstrates the potential of black-box modeling approaches in accurately mimicking the complex behavior of real-world organisms, processes, and systems, opening doors for diverse applications in scientific and engineering fields.

Keywords: Dynamic Systems, Machine Learning, Circuit Modeling, Neural Networks Architectures, Sequence-to-sequence prediction, Input-to-output modeling, Black-box modeling

Contents

List of Figures

Chapter 1

Introduction

Although the term *machine learning* and some of its algorithms and techniques has been around since the 1960's, the practical applicability and therefore the perceived usefulness of this type of artificial intelligence in real life, has remained mostly hidden from the mainstream public. While research in the area has continued to develop steadily, with some important results being reported over the years by pockets of dedicated researchers, it was only recently that major breakthroughs were able to revolutionize the general perception, the importance and the visibility that the area has in our daily lives. In the 90's, machine learning started to reorganize into a subset of the artificial intelligence field and since then, there has been a significant upwards shift on people's understanding of the potentialities that the area as a whole has to offer. This led to an exponential increase in scientific interest and research in related topics coming from a multitude of areas and application fields. This in turn led to the start of incorporation of machine learning in the widest possible human- and nonhuman-related areas with surprising results, which seem to be even more promising and astonishing for the future. The success of such endeavors and the potential that they unveil is now the catalyst for a seemingly global effort to try to use and apply this types of techniques to perform tasks that were previously thought to be impossible for a computer to tackle and solve.

At the heart of all machine learning tasks is the notion of acquiring knowledge and understanding regarding the behavior of systems and processes and be able to interpret, respond, control or mimic such systems. Having said that, since one of main goals of machine learning is to provide automation in work typically performed by humans, and since the majority of systems that describe real-world behavior are dynamic, it is imperative that machine learning is capable of handling faultlessly and with the maximum performance possible, these dynamic systems.

1.1 Context and Problem Definition

Quite often, many real world organisms, processes or systems exhibit complex behavior that changes depending on multiple internal and external conditions. Many such systems exhibit rich dynamics in response to various stimuli, and this behavior can often be encapsulated in a model that allows us to comprehend the underlying behavior and predict its future reactions. In some cases this behavior can be encapsulated in a dynamic model, perhaps consisting of various sets of nonlinear and multidimensional equations that describe down to the appropriate detail the intrinsic phenomena that governs such behavior. Such models are often called white-box models since besides allowing accurate prediction of the system's response to new stimuli, they enable visibility into the internal phenomena that causes those responses. However if either the system is too complex or the phenomena require excessive detail to model, the resulting model can become intractable for a machine to be able to comprehend and potentially mimic such behavior. In such cases, traditional approaches to modeling, that require a physical understanding of the system, become extremely difficult and time-consuming. If, on the other hand, the goal is simply to obtain a model that accurately mimics the original system and allows for accurate and efficient prediction of its input-output relationship, then perhaps it is justified to sacrifice the detail, transparency and accountability that a white-box type of model would provide, and follow a *black box* approach. In black-box models one forfeits a full understanding and knowledge of its internal workings, for the ability to efficiently and accurately being able to predict the system's response to a variety of stimuli. Machine learning techniques are quite adept at this type of modeling, as they are based on observing data corresponding to system input and output behavior and being able to infer from it, the underlying behavior. Given our goal in this work to be able to model systems that exhibit dynamic responses, machine learning techniques not only fulfill this purpose but also can seemingly do it with promising results for accurate and efficient models, as it will become evident in the following sections.

1.2 Objectives

With this context in mind, the objective of this thesis is to study the possible applications of different types of machine learning techniques to learn how to both represent and predict the behavior of dynamic systems and be able to generate a model that can potentially be used as a proxy for such a system. The description above encapsulates the two complementary goals to be attained by the resulting model. On one hand it must be able to justify the underlying causal dynamic relationships that one assumes to be implicit in the available input-output data. On the other hand it should have accurate predictive abilities to infer the system behavior when excited by different stimuli, perhaps unseen in existing data. This is often referred to as the ability to generalize. These are in fact, in and of itself, the main characteristics sought in any machine learning algorithm or machine-learning based system. For that reason, this thesis will mainly focus on dynamic neural networks, that correspond to xan emerging research branch in deep learning, but also are capable of adapting their structures or parameters to different inputs, as described

in [1], which is precisely what is required to modeling dynamic systems.

1.2.1 Electronic Circuit Simulation

Since the ever-increasing complexity of circuits has resulted in simulation times beyond what is practical - the complete design cycle of a circuit and subsequently circuit and sub-circuit simulations may take up to a year - circuit designers and model developers must consider the adoption of alternative methods for modeling their dynamic systems. Therefore, while not restricted to, part of the motivation for this work is to be able to either model electronic systems, or systems whose behavior can be emulated through an electronic proxy. This includes neuronal systems which are also going to be the subject of our analysis and source for the datasets that we will be using during the course of the proposed work. For that reason, electrical and electronic circuits provide an ideal scenario for testing the usefulness of applying machine learning algorithms for modeling dynamic systems.

1.3 Thesis Outline

This thesis is organized into the subsequent sections:

- **Background:** overview and summarized explanation of the core concepts contemplated throughout this thesis, and an investigation of Peter Meyer's PhD thesis alongside a bibliographic review of the state of the art related to different types of neural networks applied to dynamic system modeling;
- **Modeling Dynamic Behavior:** explores sequence-to-sequence prediction and showcases initial results that highlight the necessity for sophisticated machine learning methods, like neural networks, to effectively model dynamic behaviors.
- **Implementation:** this section is dedicated to demonstrating the implementation processes for different neural network architectures, offering a comprehensive view of the configuration and parameterization techniques applied to effectively model dynamic behaviors;
- **Results:** contain the preliminary results of a simple feedforward neural network to showcase the poor job that it does in trying to predict the output response of a dynamic network
- **Conclusions:** serves as the section for consolidating the key findings and conclusions derived from the extensive research presented in this thesis. It also provides a comprehensive display of the experimental results, highlighting the optimal modeling techniques employed. Additionally, this segment outlines future research directions and architectural possibilities that hold the potential to advance the quest for improved outcomes in the modeling of dynamic systems.

Chapter 2

Background

In this section, the objective is twofold: first, to introduce the fundamental theories necessary for understanding the rest of this report and second, to provide readers with a collection of references from scholarly articles and scientific papers that support the core ideas in this thesis, and enhancing the academic rigor of the thesis.

Additionally, this section aims to explore the early work present in Peter Meyer's PhD thesis, which laid the groundwork for the dynamic neural networks used here, based on differential equations. By delving into this historical context, not only is the development of dynamic modeling highlighted but also essential background information is offered for the subsequent analysis of more recent and advanced modeling techniques.

2.1 Modeling via Regression

Modeling in general refers to a process of determining a set of rules, often described in the form of mathematical equations, that are capable of relating and capturing a real physical system's input-output relationship and thus representing its behavior and evolution through time.

From a black-box standpoint, where one deals with or only has access to information from such an input-output relation, a very common modeling technique is to perform regression. Through regression one attempts to estimate the relationships between one or more dependent variables (often seen as the output or outcome variables) and one or more independent variables (simply referred to as input, or excitation, stimuli, etc, but often also called features). Regarding regression, a very common and simple way to model the relationship between an independent variable x and the dependent variable y is to perform a polynomial regression, attempting to fit the available data to a prescribed polynomial function and determining its parameters in other to minimize some error metric between the available outcomes and the predicted values. Using for instance a least squares method, polynomial regression can fit a

nonlinear model of the appropriate form to the available data.

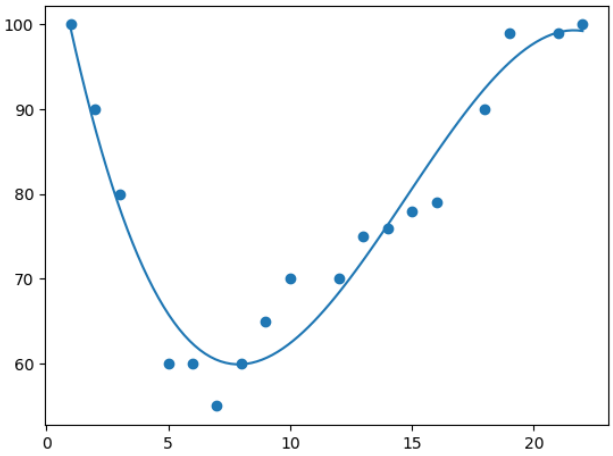


Figure 2.1: Example of a Polynomial Regression using Gradient Descent using Python.

An alternative to using polynomials, is to use neural networks often seen as a very powerful technique for modeling nonlinear behavior. Conceptually the idea is the same, in the sense that in both cases a prescribed form is assumed for the model and optimal parameter assignments are obtained through a process of minimizing some error criteria. It is known that Neural networks can be more effective than polynomials in learning certain classes of functions and hierarchical representations, as shown by [2] and [3]. Neural networks exhibit a particular ability for being able to learn invariances in a dataset, which makes them more effective for generalization in the context of statistical learning when the data has inherent hidden structure. The process of parameter fitting, normally called ”training”, can however be computationally more demanding as it may require solving a non-convex problem. However, there are also some cases where a polynomial approach to modeling might be more appropriate than using neural networks and there are some concerns over their “black box” nature, as explained in [4].

The use of neural networks for this work is justified since the datasets that we expect to encounter, resulting for instance from simulations of electronic or neuronal systems are known to exhibit strong nonlinearities. As previously said, neural networks have shown to be quite adept at modeling such behavior, typically leading to simpler and often more accurate models than those obtained through polynomial regression which may require extremely high order to accurately capture such relations.

2.2 Neural Networks

Artificial neural networks originally date back to 1944 when they were first introduced by Warren McCullough and Walter Pitts, two researchers from University of Chicago, that later moved to MIT to establish what would become the first cognitive science department [5].

The idea was to try to model and replicate the human brain's functioning through a series of algorithms. These algorithms, had the purpose of learning to perform some task by recognizing patterns and analyzing training data, thus making neural networks considered to be part of artificial intelligence.

Nowadays, due to the fact that there has been an exponentially increase in the amount of research and work done in neural networks, neural networks are starting to revolutionize the way businesses operate and everyday life, with applications in a wide range of areas, such as: optimization, electronic simulation ([6] and [7]) and electronic circuit design ([8], robotic control [9], aerospace, medicine and automotive.

2.2.1 Neural Networks Architecture

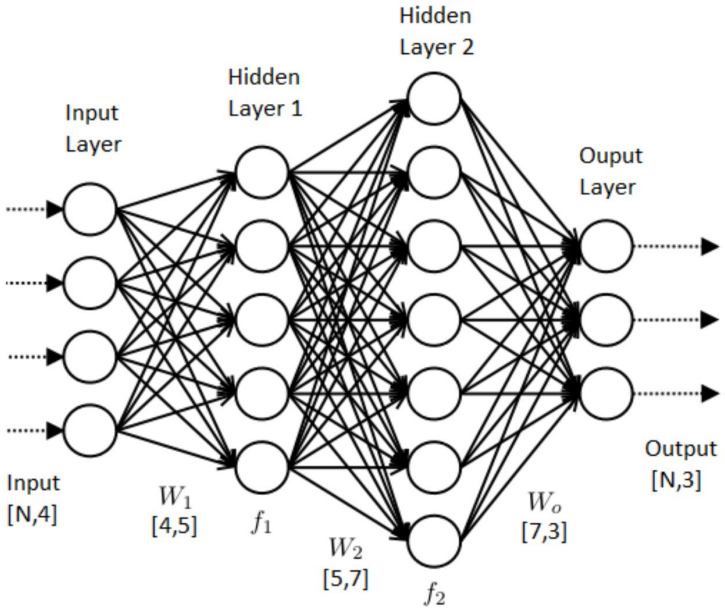


Figure 2.2: Architecture of Neural Networks (Extracted from: *https://dzone.com/articles/the-artificial-neural-networks-handbook-part-1-1)*.

Similarly to the human brain, neural networks may comprise thousands or even millions of interconnected layers of nodes, called neurons. Each one of these neurons might be connected to several neurons in the previous layer, from which it receives data, as well as with several neurons in the next layer, to which it sends data. For each one of these connections there is a weight associated that multiplies the incoming output of the previous neuron, therefore increasing or decreasing the strength of the signal of that connection, as shown in Figure 2.3, with this weighted sum of inputs being:

$$s = [1\ x^T]\ w = \tilde{x}^T\ w \tag{2.1}$$

In addition, each neuron has a (usually nonlinear) activation function through which the sum of the

input connections plus a bias, get passed to compute the output for this neuron. There may also be a threshold for each neuron, such that a signal is sent only if it crosses that threshold. The activation functions can take many forms, but should be continuous and differentiable to allow the evaluation of the influence of weight changes on the network output. The most commonly used activation functions are the *sigmoids* (2.2 and 2.3), *linear* (2.4), *RELU* (2.5) and *softmax* (2.6). The softmax, which is also know as normalized exponential function, is a function that turns a vector of K real values into a vector of K real values that sum to 1. It also transforms the input values between 0 and 1 regardless of being positive, negative or greater than one, so that they can be interpreted as probabilities, hence *softmax* being commonly used as the final layer of the neural network.

$$Logistic\ function : g(s) = \frac{1}{1 + e^{-s}}, \tag{2.2}$$

$$Arctangent\ function : g(s) = arctan(s), \tag{2.3}$$

$$Linearunit : g(s) = s, \tag{2.4}$$

$$RELU : rectified\ linear\ unit : g(s) = max(0, s), \tag{2.5}$$

$$Softmax : \sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},\ for\ i = 1, ..., K\ and\ \mathbf{z} = (z_1, ..., z_K) \in \mathbb{R}^K. \tag{2.6}$$

Even though all of these functions can be found in some applications, the most popular choice for the activation function is the *RELU*, since it does not saturate, which makes the convergence of the gradient algorithm faster. For the output layer, other units like *linear* and *softmax* are also often used.

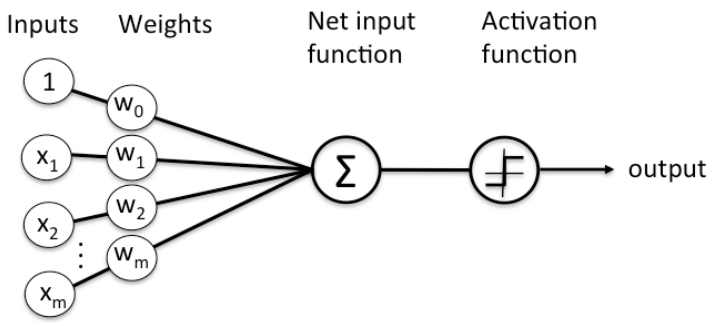


Figure 2.3: Connections in Neural Networks.

Regarding how the architecture of the neural networks is chosen, since there is no generic way to

determine *a priori* the best number of layers and neurons for a neural network, one needs to combine heuristics with prior experience about neural networks used on similar problems to define the architecture. After that, some variations are tried and checked for performance before picking the best one. This is usually done during the training and validation phases of the process.

2.2.2 Learning

In what is called supervised learning, a dataset containing a known input as well as the observed response to that input is fed to the neural network, and the output of the neural network is then stored. Then, one can compute the error in the predicted outputs by comparing how they differ from the actual observed values. That difference, or error, implicitly defines a cost function and a method can then be setup to try and minimize that cost function. A commonly used cost function is the mean-squared error between the computed and observed values of the output, defined in Eqn. (2.7).

$$MSE = \frac{1}{N} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2. \tag{2.7}$$

The network can then adjust its weighted associations and thresholds from that comparison in order to minimize the computed difference. The minimization of the MSE is often achieved by using the gradient algorithm, or a modified version of it.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t). \tag{2.8}$$

With η denoting the learning step and the weight update being:

$$\Delta w_{ij}(t) = -\eta \, \frac{\partial MSE}{\partial w_{ij}}. \tag{2.9}$$

In order to train the network and update the weights w_{ij}, one needs to compute the gradient of the cost function. Therefore, the chain rule of differentiation must be used. An illustrative example of the utilization of such rule goes as follows:

$$\frac{dz}{dx} = \frac{\partial z}{\partial w} \, \frac{dw}{dx} + \frac{\partial z}{\partial v} \, \frac{dv}{dx}. \tag{2.10}$$

There are three ways to update the weights, the first is using the batch mode, where the weight update uses all the training patterns in each iteration; the second is the on-line mode or stochastic gradient, which consists of using only one training pattern to update the weights and finally, the mini-batch mode, that requires a small subset of training patterns to update the neural network weights. This

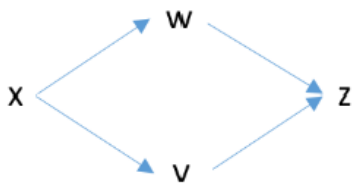


Figure 2.4: Chain rule of differentiation.

minimization of the cost function can be performed with one of many possible optimization schemes, like the *momentum*, *adgrad* and *adam*. However, the stochastic gradient descent method is by far the most common method to optimize neural networks. A comparison between the different optimization techniques for neural networks can be obtained from [10].

The most commonly used algorithm for training and consequently adjusting the network parameters is backpropagation. This algorithm calculates the gradient of the cost function associated with a given state with respect to the weights and by minimizing the computed errors, one can attain a more accurate model. In order to avoid a slow convergence of the gradient algorithm, several acceleration techniques are used, namely: momentum term and adaptive weights, as previously discussed in the optimization of neural networks.

2.3 Dynamic Neural Networks

Most of the neural network structures used presently are static (feedforward) neural networks. The response of such standard neural network can be directly computed from instantaneous knowledge of the inputs of the network using the weights resulting from the training phase. This process is often called inference, but may also be referred to as model evaluation. While this characteristic makes these networks suitable to model functions, they are really not an appropriate choice to model dynamic systems where the same input value, i.e. the same value of x, could lead to different values at the output y, depending on the history of the system and its current state.

In Section **??** a simple example is shown that further supports this claim by illustrating the difficulties encountered when attempting to model dynamic behavior through static neural networks.

Since device and subcircuit modeling for circuit simulation is characterized by having a complex set of mathematical equations to model the electronic components, that are nonlinear, multi-dimensional and dynamic, it is required to have dynamic neural networks instead of standard ones to successfully model the underlying physical behavior of the system. In fact, device and subcircuit modeling is not the only area that require dynamic neural networks for obtaining an accurate model, there are numerous other areas that require deep learning for time series forecasting, as described in [11].

On the other hand, multiple approaches have indeed been reported in the literature on how to apply

neural networks for circuit simulation, e.g. [12] and [13].

An overview of several dynamic neural networks is provided in [14] where a reasoning is also presented on why static feedforward neural networks are well suited for pattern recognition applications, while dynamic neural networks are necessary for time dependent systems.

Because of these difficulties this thesis will mainly focus on dynamic neural networks and in order to provide the network with local memory characteristics, it is necessary to implement some degree of feedback between the neurons of a layer and/or between the layers of the network and to use time delay units.

2.4 Neural Networks for Dynamical System Modeling

The motivation and purpose for the work done in this thesis was originally derived from a PhD thesis carried out by Peter Bartus Leonard Meijer at the Technical University of Eindhoven, The Netherlands, and at Philips Research Laboratories, also in Eindhoven [15]. The title of that thesis is: *Neural Network Applications in Device and Subcircuit Modelling for Circuit Simulation* and serves not only as proof for the advantages of using neural networks for modeling dynamic systems, as it will become evident in the detailed overview of the thesis in the following subsections, but also as the motivational basis for the work performed in this thesis. Interestingly enough it also serves as an historical mark to developments in the field of machine learning and neural networks in particular. In fact, even though recurrent neural networks (RNNs), to be further analyzed in Section 2.5, which are nowadays widely used to model dynamic behavior and for prediction and forecasting tasks, are thought to have first been presented in the late 80's, the truth of the matter is that in the early 90's their usage was limited to a small number of research groups. Therefore Meijer's work, completed in 1996 but obviously started much sooner, can certainly be considered as a pioneering work.

2.4.1 Context

Peter Meijer's PhD thesis explores the possible applications and potentialities of applying artificial neural networks to model the static and dynamic behavior of electronic circuits, hence the focus of the thesis being on developing and creating models for device components and sub-circuits, and also in the possibility of automating that development. Furthermore, the models to be generated were targeted for later inclusion in circuits simulators to be used within Philips Research Laboratories for the development of new circuits, systems and ultimately products. Therefore the requirements of such models clearly included accuracy as a prime target, but also focused on efficiency of computational time and resources required to run said simulations.

2.4.2 Problem and Objective

Since the mathematical models that define electronic circuits, such as semiconductors, are typically characterized by being highly nonlinear, dynamic and multidimensional systems, one could conclude that it is extremely challenging to obtain models that are capable of accurately and precisely simulating the real behavior of a circuit and even more so in an efficient way. Furthermore, as previously mentioned, one is often trapped between requirements for higher precision, that in turn leads to more complicated models and slower simulation speeds, as well as efficiency, hence more simplified models that enables higher simulation speeds but perhaps at the cost of more imprecise models.

2.4.3 Dynamic Feedforward Neural Networks

In Meijer's thesis, instead of using only a nonlinear function of a net input, each neuron also involves a linear differential equation with two internal state variables. This enables each single neuron to represent a second order bandpass filter (leading to very powerful building blocks for modeling).

It was assumed that the most common modeling situation was that the terminal currents of an electrical device or subcircuit are represented by the outcomes of a model that receives a set of independent voltages as its inputs, so that was what was taken into consideration in the work performed in this thesis. However, there are some exceptions that cannot be modeled by this approach such as: representation of combinatorial logic (where the relevant inputs and outputs are often chosen to be voltages). The thesis does not delve into those cases, but clearly some sort of current to voltage conversion would be required.

2.4.4 Notational Conventions and Parameters

The notational conventions and parameters used in [15] can be extrapolated from Figure 2.5.

With the notations and parameters comprising:

- Network inputs: $x^{(0)} \equiv (x_1^{(0)}, ..., x_{N_0}^{(0)})^T$
- Neuron output vector: $y_k \equiv (y_{1,k}, ..., y_{N_k,k})^T$
- Static connection strength (weight parameter): w_{ijk}
- Extra parameter with dynamic behavior (also considered to be a timing parameter): v_{ijk}
- A threshold parameter that it is used to calculate the offset of a static hyperplane: θ_{ik}
- A vector of weight parameters that is used to determine the orientation of the hyperplane: $w_{ik} \equiv (w_{i,1,k}, ..., w_{i,N_{k-1},k})^T$

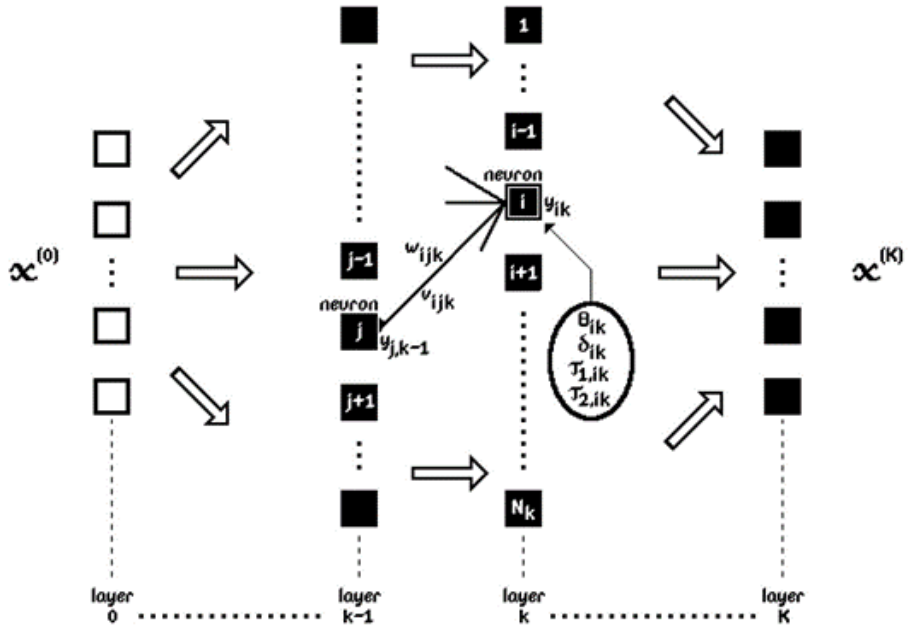


Figure 2.5: Some notations associated with a dynamic feedforward neural network (Extracted from [15] in 2.2.1 - Figure 2.2).

- Additional (transition) parameter: δ_{ik}
- Additional timing parameters: $\tau_{1,ik}$ and $\tau_{2,ik}$

As Figure. 2.5 suggests, the parameters w_{ijk} and v_{ijk} are considered to belong to neuron *i* in layer *k*. It is important to note that, since $dy_{j,k-1}/dt$ is multiplied by the weight parameter v_{ijk}, making the contributions from v_{ijk} amplifying rapid changes in neural signals, while the timing parameters have the opposite effect of making the neural response more gradual.

To guarantee that this parameters lie within some range, they are determined from associate parameter functions:

$$\begin{cases} \tau_{1,ik} = \tau_1(\sigma_{1,ik}, \sigma_{2,ik}) \\ \tau_{2,ik} = \tau_2(\sigma_{1,ik}, \sigma_{2,ik}) \end{cases} \tag{2.11}$$

2.4.5 Neural Network Differential Equations

The differential equation for the output y_{ik} of one neuron is expressed as in Eqn. (2.12) and the weighted sum *s* of the outputs in Eqn. (2.13). Both can be found in Section *2.2.2 Neural Network Differential Equations and Output Scaling*, corresponding to Equations (2.2) and (2.3) of the thesis, respectively.

$$\tau_2\left(\sigma_{1,ik},\sigma_{2,ik}\right)\frac{d^2y_{ik}}{dt^2}+\tau_1\left(\sigma_{1,ik},\sigma_{2,ik}\right)\frac{dy_{ik}}{dt}+y_{ik}=F(s_{ik},\delta_{ik}), \tag{2.12}$$

with *s* being the weighted sum of the outputs from the preceding layer:

$$s_{ik}w_{ik}\cdot y_{k-1}-\theta_{ik}+v_{ik}\cdot\frac{dy_{k-1}}{dt}=\sum_{j=1}^{N_{k-1}}w_{ijk}\,y_{j,k-1}-\theta_{ik}+\sum_{j=1}^{N_{k-1}}v_{ijk}\,\frac{dy_{j,k-1}}{dt}. \tag{2.13}$$

The selection of a proper set of equations for dynamic neural networks requires a careful analysis of the several good choices that may arise. This analysis is partly heuristic and partly based on "circumstantial" evidence.

As section 2.2.3 from [15] unveils, some of the reasons that lead to the choice of Eqns. (2.12) and (2.13) are:

- A nonlinear (typically sigmoid) function $F(\ldots)$ with at least two identifiable operating regions provides a general capability for representing or approximating arbitrary discrete (static) classifications.
- A nonlinear, monotonically increasing and bounded continuous function $F(\ldots)$ also provides a general capability for representing any continuous multidimensional (multivariate) static behavior up to any desired accuracy, using a static feedforward network and requiring not more than one hidden layer.
- It is shown that the ability to represent any multidimensional static behavior almost trivially extends to arbitrary quasistatic behavior, when using Eqs.2.12 and 2.13, while requiring no more than two hidden layers.
- The use of an infinitely differentiable function $F(\ldots)$ makes the whole neural network infinitely differentiable.
- A single neuron can already exactly represent the dynamic behavior of elementary but fundamental linear electronic circuits.
- The term with v_{ijk} provides the capability for time-differentiation of input signals to the neuron, thereby amplifying, or "detecting" rapid changes in the neuron input signals.
- The terms with w_{ijk} and v_{ijk} together provide the capability to represent, in a very natural way, the full complex-valued admittance matrices arising in low-frequency quasistatic modeling.
- The term with $\tau_{1,ik}$ provides the capability for time-integration to the neuron, thereby also time-averaging the net input signal s_{ik}.

- The term with $\tau_{2,ik}$ suppresses the terms with v_{ijk} for very high frequencies. This ensures that the neuron (and neural network) transfer will drop to zero for sufficiently high frequencies, as happens with virtually any physical system.

- If all the $\tau_{1,ik}$ and $\tau_{2,ik}$ in a neural network are constrained to fulfill $\tau_{1,ik}$ ¿ 0 and $\tau_{2,ik}$ ¿ 0, then this neural network is guaranteed to be stable in the sense that the time-varying parts of the neural network outputs vanish for constant network inputs and for times going towards infinity.

- It is also proven how the choice of Eqns. (2.12) and (2.13) avoids the need for a nonlinear solver during DC and transient analysis of the neural networks. Thereby, convergence problems with respect to the dynamic behavior of the neural networks simply do not exist, while the efficiency is greatly improved by always having just one ”iteration” per time step. These are major advantages over general circuit simulation of arbitrary systems having internal nodes for which the behavior is governed by implicit nonlinear equations.

2.4.6 Neuron nonlinearity

In the PhD thesis, three different, but all monotonically increasing and bounded, choices for the neuron nonlinearity $F(\ldots)$ were studied. First, it was considered the special case where all timing parameter in Eqns. (2.12) and (2.13) are zero and the neuron nonlinearity is chosen to be the logistic function, making the dynamic neural network transform into a standard static one. However, generally these parameters were nonzero and instead of applying the logistic function that lacks the common transition between highly nonlinear and weakly nonlinear behavior (that is typical for semiconductor devices and circuits), other infinitely differentiable nonlinear functions were used.

One dynamic alternative to the logistic function that was studied was the function $F(\ldots)$ as described in Eqn. (2.14).

$$F(s_{ik},\delta_{ik})\frac{1}{\delta_{ik}^2}\left[ln\left(cosh\frac{\delta_{ik}^2(s_{ik}+1)}{2}-ln\,(cosh\frac{\delta_{ik}^2(s_{ik}-1}{2}\right)\right] = \frac{1}{\delta_{ik}^2}\,ln\frac{cosh\frac{\delta_{ik}^2(s_{ik}+1)}{2}}{cosh\frac{\delta_{ik}^2(s_{ik}-1)}{2}} \tag{2.14}$$

Analyzing Eqn. (2.14), one can see that one control the sharpness of the transitions between linear and exponential behavior with parameter δ_{ik}. The function $F(\ldots)$ described in Eqn. (2.14), has the following graphic representation:

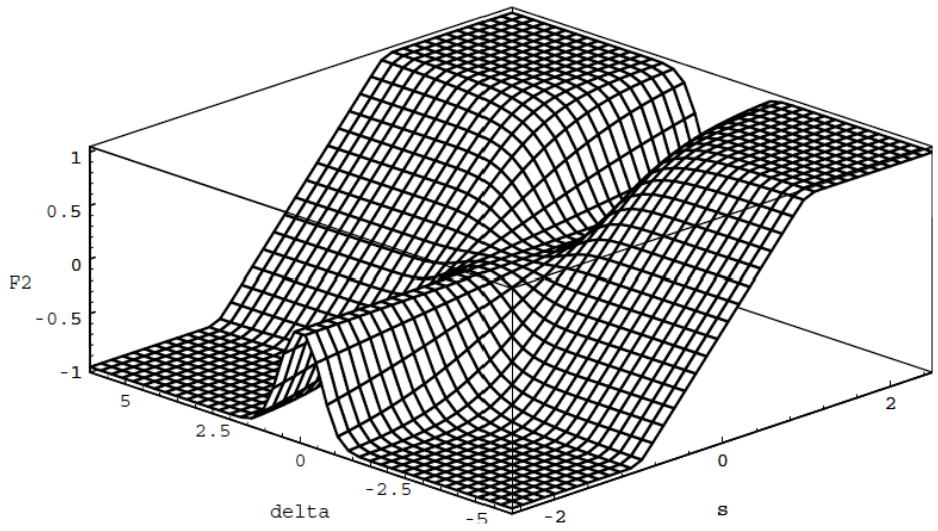


Figure 2.6: Neuron nonlinearity $F(s_{ik}, \delta_{ik})$ (Extracted from [15] in 2.2.4 - Figure 2.5).

2.4.7 Implementation

Although Eqn. (2.12) is an ordinary differential equation whose solution can be computed analytically, a choice was made to solve the differential equations numerically. Therefore, the differential equations are solved using a linear multistep approximation for the derivatives and a nonlinear solver. There are several general algorithms to obtain the numerical solution of ordinary differential equations, such as the Adams-Bashforth, Adams-Moulton, Runge-Kutta, etc. However, the thesis considers only the first order Backward Euler method, a member of the Adams-Moulton family of algorithms, for variable time step size.

2.4.8 Meijer's Conclusions

As Peter Meijer's PhD thesis concludes, electronic circuits can usually be characterized as being complicated nonlinear multidimensional dynamic systems, which makes modeling them not an easy task. However, it is clear how neural networks can lead to practical solutions, especially with the introduction of external feedback into the dynamic neural networks that were developed. Which in turn, would allow for the representation, up to arbitrary accuracy, of a very general class of nonlinear multidimensional implicit differential equations, covering any state equations of the form $f(x, \dot{x}, t) = 0$ as used to express the general time evolution of electronic circuits. Moreover, the introduction of feedback into the existing models makes them universal approximators for arbitrary continuous nonlinear multidimensional dynamic behavior.

2.5 Recurrent Neural Networks

A recurrent neural network (RNN) is a special case of an artificial neural networks that has been investigated intensively in recent years due to their ability to model and predict nonlinear time-variant system dynamics.

This type of network differs from feedforward or convolutional neural networks (CNNs) because they contains loops that allow information to be stored within the network. This in turn allows it to use reasoning from previous experiences to inform upcoming events, similar to having a ”memory”. A common application example for these deep learning algorithms are temporal problems, such as language translation and natural language processing (nlp), as described in [16] and in speech recognition tasks.

Moreover, recurrent neural networks models are not only important for the forecasting of time series but also generally for the control of dynamical systems. Therefore, for completeness, the following subsections include a brief overview of some of the different types of RNN's.

2.5.1 Long short-term Memory

Long short-term memory (LSTM) is a type of artificial recurrent neural network (RNN) used in the field of deep learning with numerous applications in real-world cases that require dynamic system modeling, such as: image processing, speech recognition, autonomous systems or even music composition. It differs from standard feedforward neural networks since LSTM has feedback connections, enabling them to model and predict nonlinear time-variant system dynamics. An extensive overview of LSTM cell derivatives and network architectures for time series prediction is provided in [17].

LSTM networks are capable of learning order dependence in sequence prediction problems, making them well-suited to classifying, processing and making predictions based on time series data (such as speech or video). Usually, LSTM units are composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

The main purpose behind the development of this type of neural networks, is that the computations involved when training a classic recurrent neural network using back-propagation could lead to the gradients ”vanishing” (that is, they can tend to zero) or ”exploding” (that is, they can tend to infinity).

RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged. However, LSTM networks can still suffer from the exploding gradient problem.

2.5.2 Gated Recurrent Units

Recurrent neural networks with gated recurrent units (GRU) emerged in 2014 as a simpler alternative for LSTMs due to the fact that these last networks have a high computational cost. The gain in computational speed sometimes takes a toll in effectiveness. Often, LSTMs are more effective and powerful since they have three gates instead of two. In other cases, however the simplicity of GRUs leads to more efficient solutions.

In Figure. 2.7, extracted from *https://medium.com/@le.oasis/recurrent-neural-networks-report-c70e6f05cc9e*, it can be seen the comparison in structure, namely in terms of gate's quantity, between a LSTM network and a GRU unit.

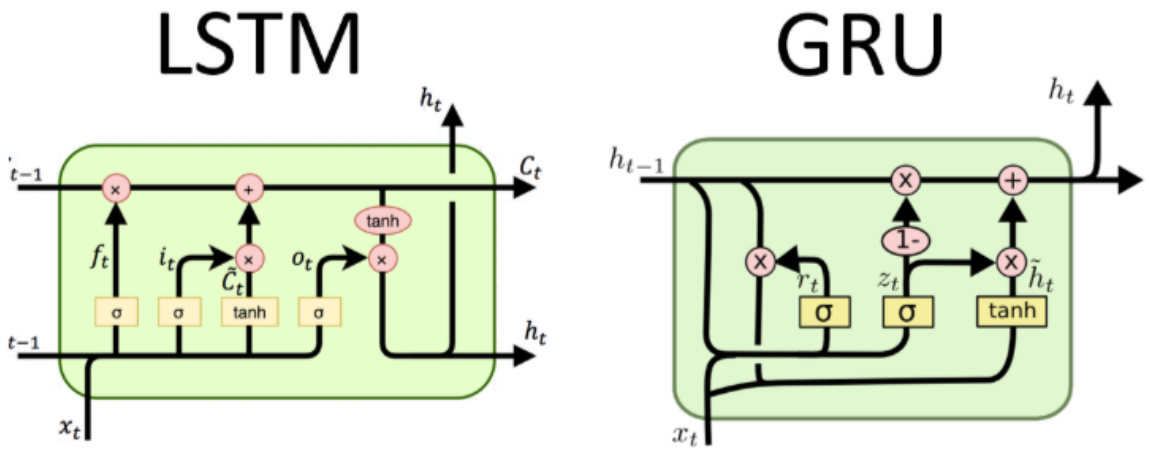


Figure 2.7: LSTM vs GRU Network.

2.5.3 Nonlinear Autoregressive Neural Network with Exogenous Inputs

The nonlinear autoregressive model with exogenous inputs (NARX) is a recurrent dynamic neural network with feedback connections enclosing several layers of the network. It is based on the ARX model, which is a linear representation of a dynamic systems in discrete time, thus being used in time-series modeling.

The NARX model relates the current value of a time series to not only the past values of the same series, but also to the current and past values of another series obtained from external inputs that influence the series of interest, hence being called as exogenous inputs.

2.5.4 Continuous Time Recurrent Neural Network

Continuous time recurrent neural networks (CTRNN) is a type of recurrent neural network that have been successfully applied to a wide range of applications involving dynamic systems [18]. They are

composed of a system of ordinary differential equations, with neuron potentials as the dependent variables.

The rate of change of activation ($\frac{dy_i}{dt}$) of a neuron *i* is modeled by Eqn. (2.15).

$$\tau_i \frac{dy_i}{dt} = -y_i + f_i\Big(\beta_i + \sum_{j \in A_i} w_{ij} y_j\Big), \tag{2.15}$$

where the values and constants correspond to:

- τ_i is the time constant of neuron *i*
- y_i is the potential of neuron *i*
- f_i is the activation function of neuron *i*
- β_i is the bias of neuron *i*
- A_i is the set of indices of neurons that provide input to neuron *i*
- w_{ij} is the weight of the connection from neuron *j* to neuron *i*

Using a time-discretization formula such as the Euler method, it is possible to compute the time evolution of the network:

$$y_i(t + \Delta t) = y_i(t) + \Delta t \frac{dy_i}{dt}. \tag{2.16}$$

The form of Eqn. (2.15) is intriguing as it resembles the form of the model from [15]. Further work on this topic will be carried out to determine the connections between the various models.

2.6 Related Work

In order to ascertain the potentiality of macromodeling non-linear behavior using neural networks and to assess the existing work with regard to this matter, a thorough research was performed.

2.6.1 Dynamic Behavioral Modeling of nonlinear Circuits

As seen in the article [19], applying neural networks techniques to model dynamic behavior of nonlinear circuits, proves to be incredibly effective. Naghibi, Sadrossadat, and Safari proposed an innovative approach to capture dynamic behavior in nonlinear circuits. The technique involved employing a recurrent neural network to model and predict responses. Their work advanced the understanding

and analysis of complex nonlinear circuit dynamics, with potential applications in telecommunications and signal processing. By combining circuit theory and neural networks, the researchers expanded the toolbox for dynamic behavioral modeling in nonlinear circuits. The results demonstrated the efficacy of the recurrent neural network technique in capturing nonlinear circuit dynamics and bridging gaps between traditional methods. Overall, their research contributes to better understanding and prediction of nonlinear circuit systems.

2.6.2 Continuous Time Recurrent Neural Networks

In the study by Heinrich, Alpay, and Wermter, [20], they proposed adaptive and variational CTRNNs, which aimed to enhance the capabilities of traditional CTRNNs by introducing adaptive and variational components. These additions enabled the network to dynamically adjust its internal parameters and variance, leading to improved learning and adaptability in modeling dynamic systems.

On the other hand, Meijer's PhD thesis, [15], focused on the application of neural networks in device and subcircuit modeling for circuit simulation. The research explored how CTRNNs could be employed to model the behavior of electronic devices and subcircuits, allowing for more accurate and efficient circuit simulations. This work demonstrated the potential of CTRNNs in tackling complex modeling tasks within the domain of circuit design.

Both works highlight the importance of CTRNNs in modeling continuous-time dynamics. They showcase the versatility of CTRNNs in various domains, ranging from robotics and adaptive behavior to circuit simulation and electronic device modeling. The research provides valuable insights into the development of adaptive and variational CTRNNs and their potential applications in modeling dynamic systems, fostering advancements in both theoretical and practical aspects of continuous-time neural network modeling.

2.6.3 High-speed non-linear Circuit Macromodeling

In the more recent article entitled *Batch-Normalized Deep Recurrent Neural Network for High-Speed Nonlinear Circuit Macromodeling*, [21], the authors explore a novel approach to macromodeling nonlinear circuit behavior using deep recurrent neural networks. This research extends beyond Meyer's PhD thesis, which primarily focused on neural network applications for device and subcircuit modeling in circuit simulations.

What was achieved in this article is the development and application of a state-of-the-art deep recurrent neural network model, enriched with batch normalization techniques, to tackle the challenge of high-speed nonlinear circuit macromodeling. By incorporating advanced techniques, this study aimed to enhance the precision and efficiency of capturing the intricate dynamics exhibited by real-world circuits.

The significance of this work lies in its pioneering approach to macromodeling nonlinear behavior. The utilization of deep recurrent neural networks, along with batch normalization, represents a progressive step in the field of circuit modeling. It demonstrates the potential of modern neural network architectures in accurately simulating complex nonlinear phenomena. This research has far-reaching implications for scientific and engineering applications, opening new possibilities for the modeling and understanding of nonlinear systems in diverse fields.

2.6.4 Sequence to sequence learning

The article [22] discusses a method for predicting sequence-to-sequence tasks using neural networks, showcasing that Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks.

The authors propose an architecture called the sequence-to-sequence model, which consists of two recurrent neural networks (RNNs): an encoder and a decoder.

The encoder RNN processes the input sequence and generates a fixed-length representation called the context vector. The decoder RNN then takes the context vector as input and generates the output sequence. The encoder and decoder RNNs are trained jointly using a technique called backpropagation through time (BPTT).

The authors demonstrate the effectiveness of the sequence-to-sequence model on various tasks, including machine translation and speech recognition. They show that this approach can handle input and output sequences of different lengths and capture complex dependencies between elements in the sequences.

Overall, the article provides insights into how the sequence-to-sequence model can be used to predict sequence-to-sequence tasks by employing encoder and decoder RNNs trained through BPTT and serves as motivational work for the work implemented in this thesis.

Chapter 3

Modeling Dynamic Behavior

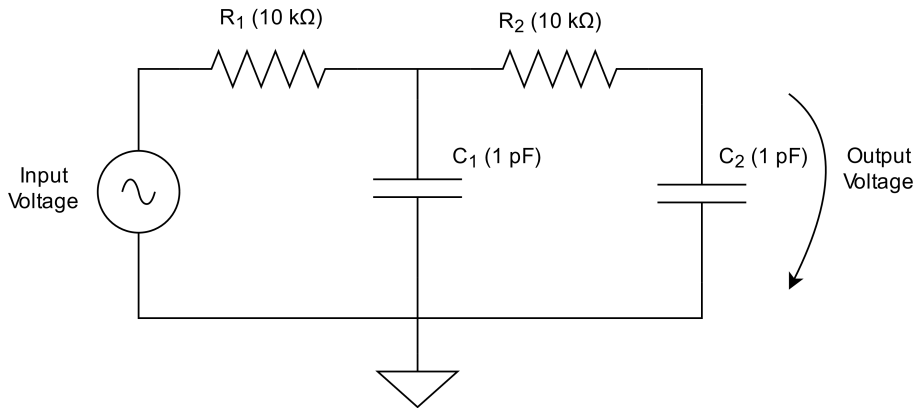
As previously highlighted in influential articles, modeling dynamic behavior is known to be a complex and challenging task. To demonstrate these challenges, a straightforward example was created.

It was initially attempted to be modeled using standard neural networks, which provides a basic reference point. In the following sections, more advanced techniques will be explored to address these complexities.

3.1 Illustrative RC example

The goal here is to be able to predict the output voltage of the simple RC circuit in Figure 3.1. The system was simulated using a standard SPICE-class simulator [23] using a simple piecewise-linear input waveform as excitation.

Figure 3.2 shows three curves corresponding to the input excitation (the blue solid line), the simulated output (the red-dotted line) and the result of a neural network model (the dash-dotted black line).



[H]

Figure 3.1: RC circuit to be modeled.

The neural network used was a standard fully connected neural network, composed of one hidden layer with 10 neurons, having the mean squared error as the performance function and the training algorithm being the Levenberg-Marquardt method.

The output of the neural network shows the learned behavior, meaning that this curve was the best we could obtain from the learned data (i.e. no generalization requested of the model in this case).

It is important to note that, since a function is characterized by having a unique value of x for distinct values of y, the input voltage values (solid blue line) were chosen in order to avoid having the values in the first upward slope of the waveform coincide with the ones from the second upward slope, this is to avoid the case were for two equal values of the input we would be presented with distinct values of the output voltage.

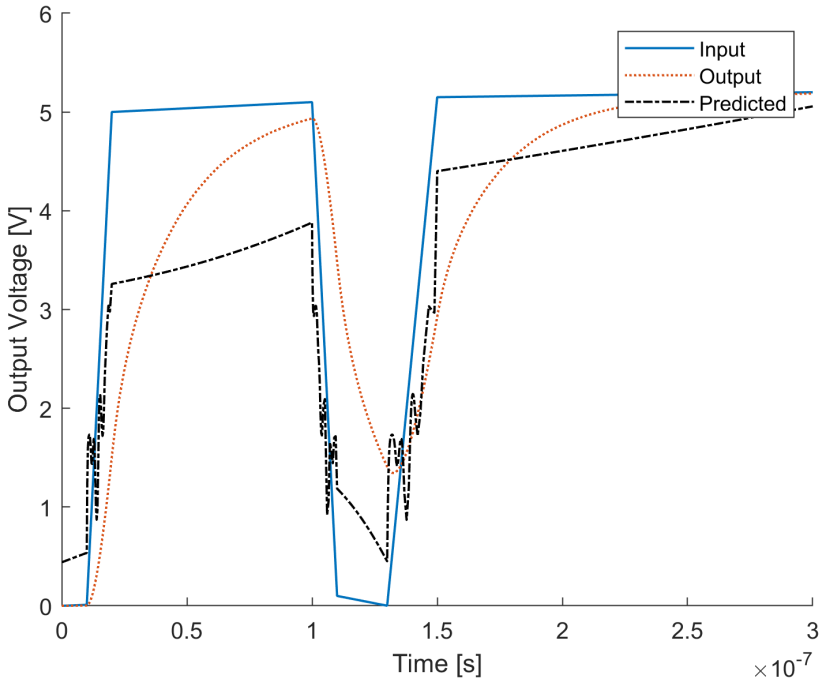


Figure 3.2: Output Voltage for a Standard Neural Network.

Figure 3.2 shows the intended time-dependent response, with time in the horizontal axis. The neural network, however, has no concept of time and for the learning process only the input values and corresponding output voltage values were used.

Figure 3.3 depicts exactly this relationship. While Figure 3.2 shows the evolution of input and output voltage trough time, in this case the graph is constructed by having the output voltage (in volts) as a function of the input voltage (also in volts).

The rationale for representing data in this manner is to showcase how the actual neural network ”sees” the data, y (output) as a function of x (input), and how the neural network is trying to compute a

function that maps, in the most accurately way possible, each input voltage value to an output one.

From the image, it is evident the poor job that the neural network is doing, given the fact that the predicted waveform does not even remotely relates to the target one, even though it seems to be an average of the three phases of the target waveform.

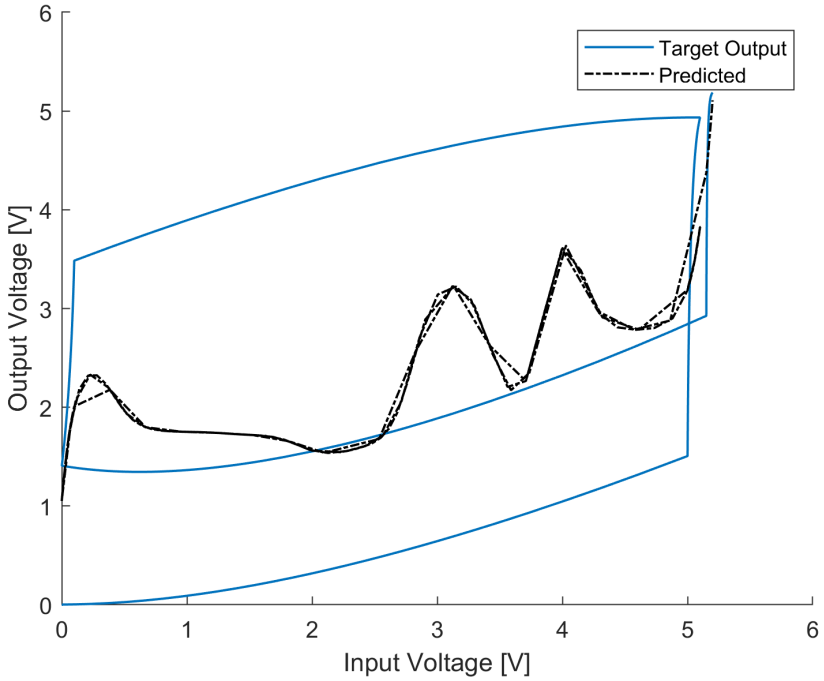


Figure 3.3: Predicted Output Voltage in function of the Input Voltage for a Standard Neural Network.

Having analyzed both figures, it is now clear that the lack of similarity between the waveforms of the output of the RC circuit to an input voltage and that of the predicted output from the model, is proof for the inability of a standard feedforward neural network to predict this types of systems.

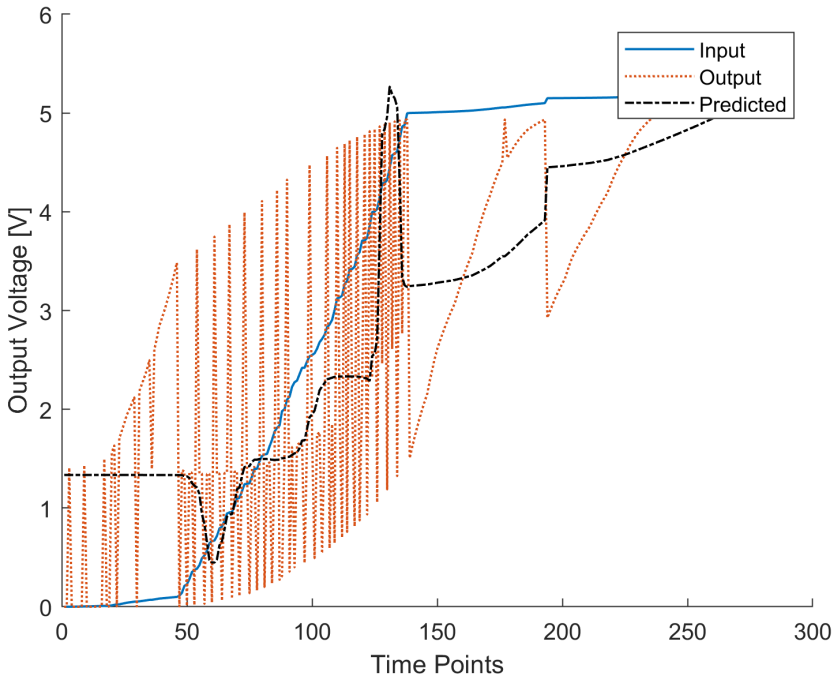


Figure 3.4: Predicted Output Voltage from a Standard Neural Network for an Ordered Input Voltage.

Of course in this case the RC circuit is a first order linear circuit that could be easily modeled by mathematical equations describing its structure and intervening elements.

Its response could be equally easily obtained through the product of the circuit's transfer function with the input's Laplace transform and then applying the reverse Laplace transform to obtain the time domain formula for the output.

However in this case, the neural network did a terrible job in trying to find a suitable regression for the input data, as evidenced from the figures included.

This happens due to the fact that the neural network only "sees" inputs independently from one another, therefore lacking the notion of time required to accurately model dynamic systems, where the output of the current state might depend on the previous state of the system.

Since a standard fully connected neural network does not have the notion of time and the output only refers to each input independently, the neural network fails to capture such dependencies. Figure 3.4, shows the response assuming that the input voltage is ordered so that it is constantly increasing.

This is of course an artificially built response but it does showcase how, from the point of view of the neural network, with its complete absence of time awareness, the system appears to be remarkably oscillatory, since very small changes in the input (from one "timepoint" to the next) lead to enormous changes on the output.

This very simple examples shows how trying to model a system with this behavior is revealed to be

to be a surprisingly complex task for a standard neural network, hence the the need to utilize neural networks that take into consideration the dynamic component of the systems.

3.2 Preliminary Results

In order to ascertain the potential of recurrent neural networks, some experimental results were obtained using some basic architectural units.

Two different types of dynamic neural networks, namely a LSTM neural network and a NARX model, were used to predict the output of the RC circuit from Figure 3.1, showing promising results.

For the time being, no optimization schemes were applied since the purpose of these preliminary results is to showcase the potentialities of such neural networks compared to standard feddforward ones.

The optimization and exhaustive experimentation with LSTMS and NARX neural networks is reserved for the future.

Regarding the input data, ten datasets were generated, each using a different input excitation. Excitations varied between piecewise linear (a sequence of time points and corresponding voltage values), to sine waves with fixed and varying frequency and delay, as well as exponentials with different time constants.

For each example, eight of the ten datasets were randomly chosen for training and the remaining two for testing. At this time, no datasets were used to validation since this is a preliminary experiment.

3.2.1 LSTM

Using the architecture defined in Section 2.5.1, Figures 3.5 and 3.6 were obtained as the result of a LSTM neural network for the electrical circuit in Figure 3.1.

The implemented LSTM neural network comprises only one fully connected hidden layer with 64 neurons followed by one linear layer that provides the outputs.

The optimization algorithm used was the *adam* and the training was performed with a mini-batch with size 20, with the number of epochs being 200.

3.2.2 LSTM Model Structure

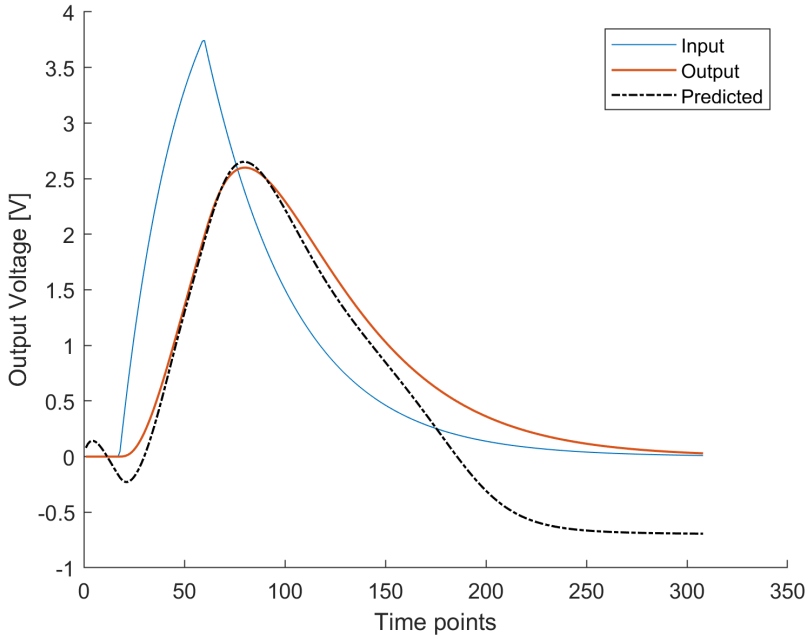


Figure 3.5: LSTM-based modeling of the example RC circuit for a pulse input.

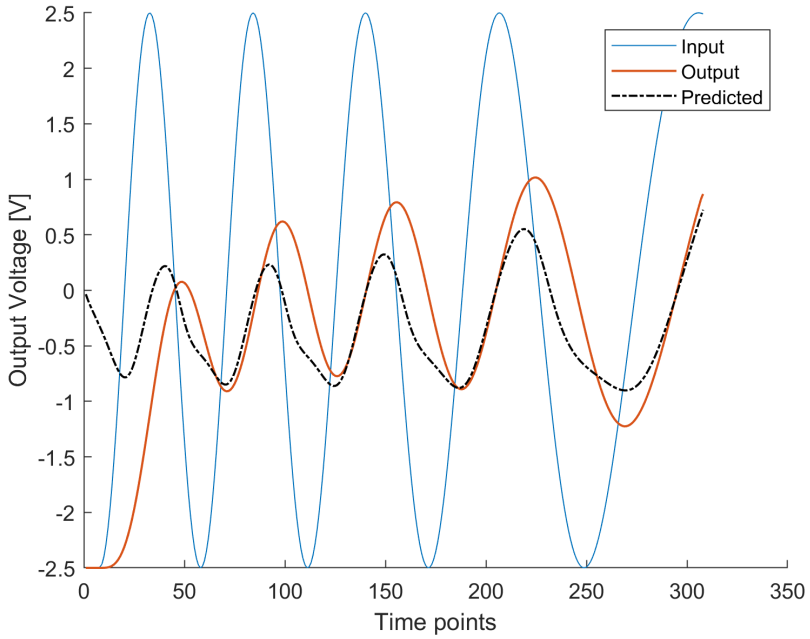


Figure 3.6: LSTM-based modeling of the example RC circuit for a sinusoidal input.

3.2.3 NARX

The implemented NARX neural network is composed of one hidden layer with 64 neurons and the network is using both 1 and 2 input and feedback delays.

Take into consideration that only a 0 input delay is possible, while feedback delays must be positive.

Analogously to the previous section, Figures 3.7 and 3.8 show the results obtained at the output for the described above NARX neural network, which has its architecture defined in Section 2.5.3, for the electrical circuit in Figure 3.1.

3.2.4 Conclusion

Analyzing both figures from the LSTM network as well as from the NARX model, and comparing them with the results from the standard neural network (Figure 3.2), one could conclude that, since for both cases, the predicted output seemingly follows the shape of the waveform corresponding to the target output, these types of neural networks are much more accurate and therefore favored for modeling nonlinear dynamic systems.

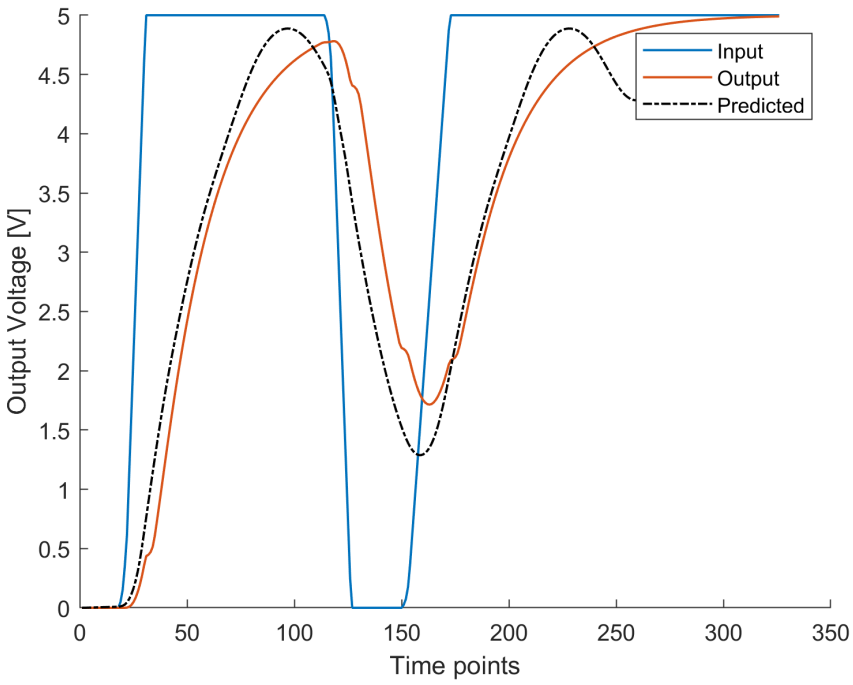


Figure 3.7: NARX-based modeling of the example RC circuit for a squared input.

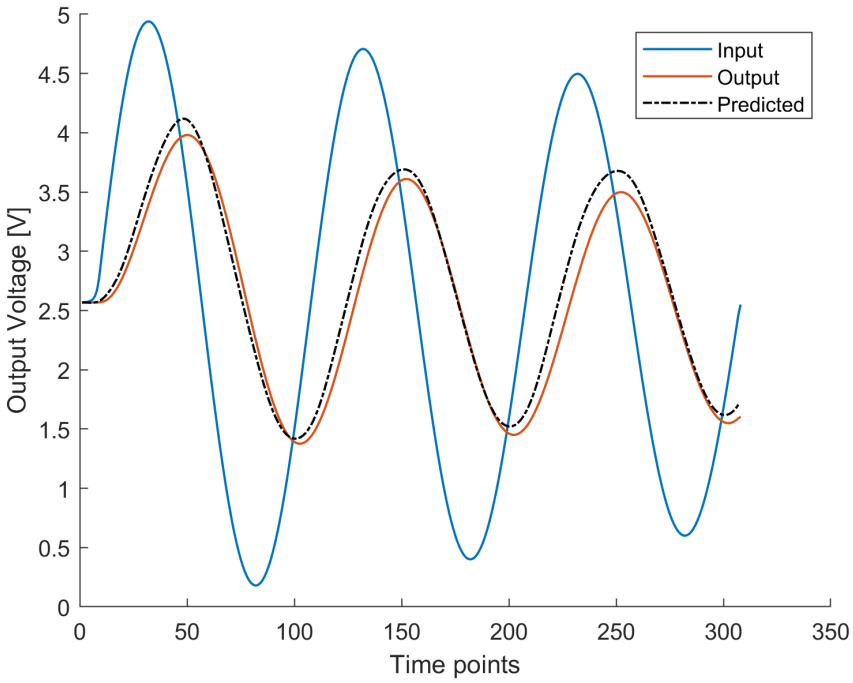


Figure 3.8: NARX-based modeling of the example RC circuit for a sinusoidal input.

3.3 Some Known and Anticipated Modeling Limitations

The dynamic feedforward neural networks as specified by Eqs. (2.2), (2.3) and (2.5), were designed to have a number of attractive numerical and mathematical properties. There is a certain price to be paid, however.

The fact that the neural networks are guaranteed to have a unique dc solution immediately implies that the behaviour of a circuit having multiple dc solutions cannot be completely modeled by a single neural network, indiscriminate of our time domain extensions. An example is the nonlinear resistive ip-op circuit, which has two stable dc solutions and one metastable dc solution that we usually don't (want to) see. Circuits like these are called bistable.

Because the neural networks can represent any (quasi)static behavior up to any required accuracy, multiple solutions can be obtained by interconnecting the neural networks, or their corresponding electrical behavioral models, with other circuit components or other neural networks, and by imposing (some equivalent of) the Kirchhoff Vt current law. After all, in regular circuit simulation, including time domain and frequency domain simulation, all electronic circuits are represented by interconnected (sub)models that are themselves purely quasistatic.

Nevertheless, this solves the problem only in principle, not in practice, because it assumes that one already knows how to properly decompose a circuit and how to characterize the resulting " components

by training data. In general, one does not have that knowledge, which is why a black-box approach was advocated in the first place.

The multiple dc solutions of the bistable ip-op arise from feedback connections. Since there are no feedback connections within the neural networks, modeling limitations will turn up in all cases where feedback is essential for a certain dc behavior. This does definitely not mean that our feedforward neural networks cannot represent devices and subcircuits in which some form of feedback takes place. If the feedback results in unique dc behavior in all situations, or if we want to model only a single dc behaviour among multiple dc solutions, the static neural networks will indeed be able to represent such behavior without needing any feedback, because it is the behaviour that we try to represent, not any underlying structure or cause. Another example in which feedback plays an essential role is a nonlinear oscillator24, for which the amplitude is constrained and kept constant through feedback. Although the neural networks can easily represent oscillatory behavior through resonance of individual neurons, there is no feedback mechanism that allows the use of the amplitude of a neuron oscillation to control and stabilize the oscillation amplitude of that same neuron. The behavior of a nonlinear oscillator may for a finite time interval still be accurately represented by a neural network, because the signal shape can be determined by additional nonlinear neurons, but for times going towards infinity, there seems to be no way to prevent that an initially small deviation from a constant amplitude grows very large. On the other hand, we have to be very careful about what is considered (im)possible, because a number of tricks could be imagined. For instance, we may have one unstable25 neuron of which the oscillation amplitude keeps growing indefinitely. The non-linearity F of a neuron in a next network layer can be used to squash this signal, after an initial oscillator startup phase, into a close approximation of a block wave of virtually constant, and certainly bounded, amplitude. The Fi1's and 2's in this layer and subsequent layers can then be used to integrate the block wave a number of times, which is equivalent to repeated low-pass-filtering, resulting in a close approximation of a sinusoidal signal of constant amplitude. This whole oscillator representation scheme might work adequately in a circuit simulator, until numerical over how problems occur within or due to the unstable hidden neuron with the ever-growing oscillation amplitude. As a final example, we may consider a peak detector circuit. Such a circuit can be as simple as a linear capacitor in series with a diode, and yet its full behavior can probably not be represented by the neural networks belonging to the class as defined by Eqs. (2.2), (2.3) and (2.5).

The fundamental reason seems to be, that the neuron output variable yik can act as a state (memory) variable that affects the behavior of neurons in subsequent layers, but it cannot affect its own future in any nonlinear way. However, in a peak detector circuit, the sign of the divergence between input value and output (state) value determines whether or not a change of the output value is needed, which implies a nonlinear (feedback) operation in which the output variable is involved. It is certainly possible to redefine, at least in an ad-hoc manner,the neuron equations in such a way, that the behavior of a peak detector circuit can be represented. It is not (yet) clear how to do this elegantly, without giving up a number of attractive properties of the present set of definitions. A more general feedback structure may be needed for still other problems, so the solution should not be too specific for this peak detector example.

Feedback applied externally to the neural network could be useful, as was explained in section 2.4.3. However, in general, the problem with the introduction of feedback is, that it tends to create nonlinear equations that can no longer be solved explicitly and that may have multiple solutions even if one doesn't want that, while guarantees for stability and monotonicity are much harder to obtain. With Eqs. (2.2), (2.3) and (2.5), we apparently have created a modeling class that is definitely more general than the complete class of quasistatic models, but most likely not general enough to deal with all circuits in which a state variable directly or indirectly determines its own future via a nonlinear operation.

3.4 RNN Disadvantages

While recurrent neural networks (RNNs) have proven to be powerful tools for modeling dynamic systems, they also possess certain limitations, [24] that need to be considered. Understanding these disadvantages is crucial to employ RNNs effectively in the context of dynamic system modeling.

One notable drawback of RNNs is the computational complexity associated with training RNNs and their susceptibility to the vanishing gradient problem. Due to the nature of their recurrent connections, RNNs can encounter difficulties in propagating error gradients backward through time, especially when dealing with long sequences or extended time dependencies, which is the case with the work that this thesis pertains. This can hinder the model's ability to capture long-term dependencies accurately, resulting in suboptimal performance and limited predictive capabilities.

Additionally, RNNs assume a stationary relationship between past and future inputs, assuming that the underlying dynamics remain unchanged over time. This assumption may not hold in many real-world scenarios where dynamic systems exhibit non-stationary behavior, leading to challenges in accurately capturing and adapting to evolving dynamics, and questioning the extrapolating case posed in this thesis, as the simulations for electric circuits might not be of real-value to real-world organisms.

Chapter 4

Implementation

4.1 Circuits

As aforementioned, electric circuits can be a proxy to emulate the dynamic behavior of real-world organisms, and since limitless simulations can be performed on these circuits, we only focused on two different single-stage amplifiers (extracted from [25] and [26] articles) to simulate steady-state and dynamic behavior and thus being able to extrapolate the results to the generality of dynamic systems. In light of that consideration, the two circuits selected to perform the simulations were single-stage amplifiers with enhanced DC gain without requiring cascode devices or employing positive-feedback or feed-forward techniques. Instead, they employ two voltage-combiners as an alternative to the conventional tail current-source typically used to bias the differential-pair.

4.1.1 VCOTA - Fully-differential amplifier

As a mean to emulate the dynamic behavior of real-world organisms, one single stage amplifier circuit with gain enhancement using voltage combiners was selected and its key performance ranges were studied to test the efficacy and accuracy of the different neural networks in non-linear behavior.

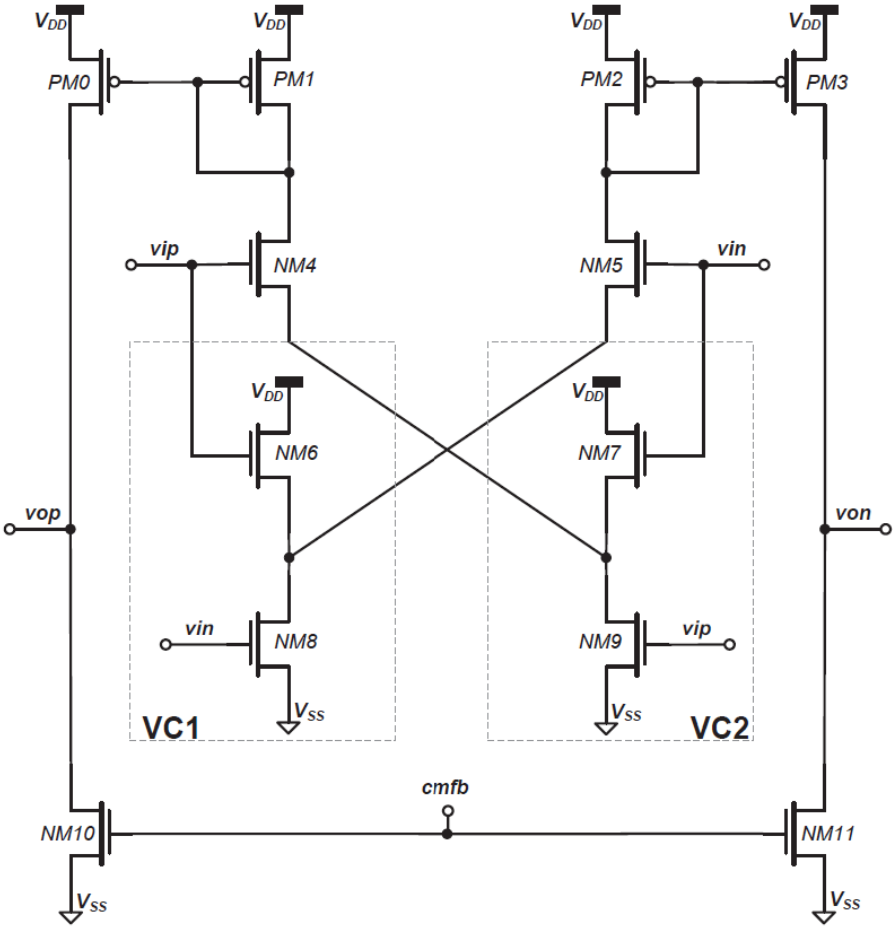


Figure 4.1: Fully-differential amplifier.

4.1.2 FCOTA - Single Stage Amplifier

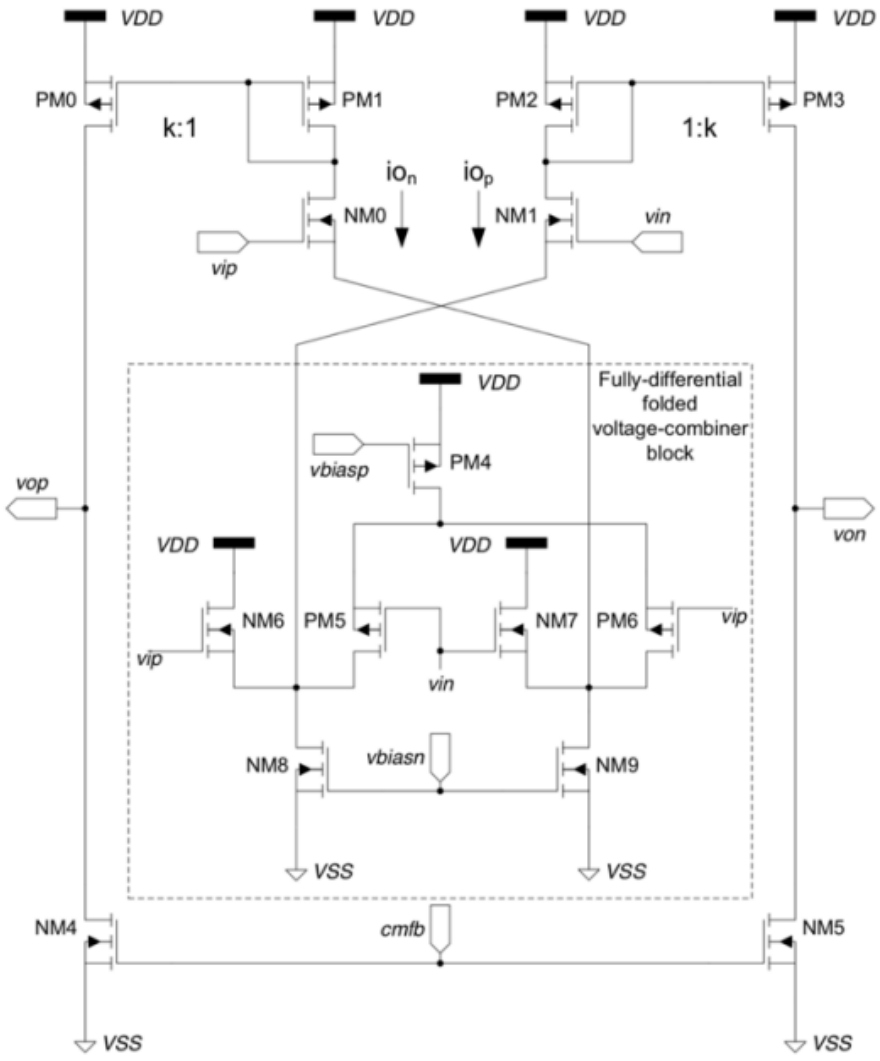


Figure 4.2: Single-stage amplifier.

4.1.3 Operating Conditions

Both circuits, under normal operating conditions as well as extreme scenarios, have been extensively investigated in articles [25] and [26]. The comprehensive findings from these studies are presented in the tables below, providing valuable insights into the expected performance and behavior of these circuits in typical conditions and corner analysis.

	FOM [MHz·pF/mA]	GDC [dB]	IDD [mA]	PM [°]	GBW [MHz]
△FOM	2278.5	50.3	0.218	61	82.79
△GDC	1435.3	54.7	0.215	63.5	51.32

△ GDC	Typical	SS	SNFP	FNSP	FF
FOM	2136.3	1959.3	2173.4	2097.5	2223.1
GBW	72.07	62.1	71.29	71.47	78.84
GDC	50.36	50.42	50.67	50.45	50.25
IDD	0.202	0.19	0.197	0.204	0.213
PM	61.7	64.6	61.7	61.8	60.4
OS[V]	0.616	0.624	0.637	0.604	0.608

Table 4.1: Measures for the corner analysis specifications (1/2).

△ GDC	Typical	SS	SNFP	FNSP	FF
FOM	1538.9	1405.9	1550.7	1528.2	1616.3
GBW	50.28	43.5	49.23	50.47	54.92
GDC	53.12	53.2	53.5	53.3	53.1
IDD	0.196	0.19	0.191	0.198	0.204
PM	65.4	67.4	65.5	65	64.4
OS[V]	0.655	0.663	0.677	0.643	0.648

Table 4.2: Measures for the corner analysis specifications (2/2).

Leveraging on the above tables and by analyzing the images extracted from [26] and [25], and observing the values for the highest FOM (Figure of merit) and for the highest GDC solutions in corner conditions, when the frequency-domain and the time-domain responses are still commonly-shaped waveforms, as seen in 4.4, it is possible to attain key values, such as ac magnitude and phase, to later use as parameters to perform varying simulations in NGSpice and extract several key datasets to train dynamic neural networks and perform simulations.

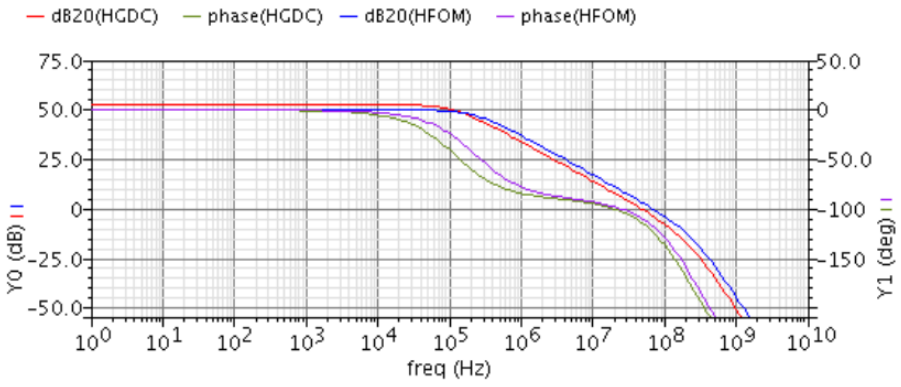


Figure 4.3: Highest FOM and Highest GDC simulation results: AC magnitude and phase.

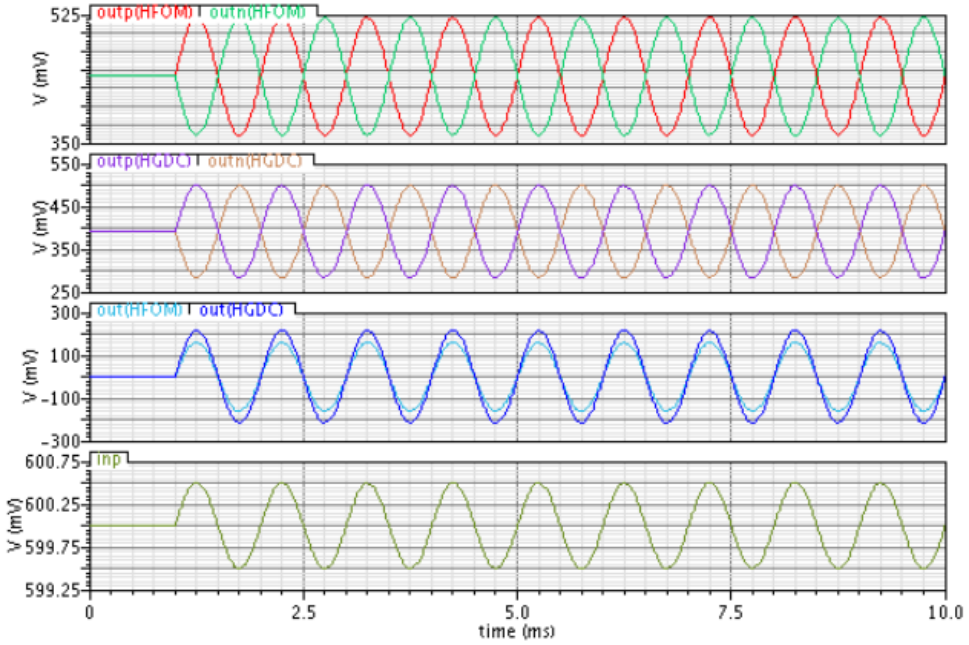


Figure 4.4: Highest FOM and Highest GDC simulation results: Sinusoidal response.

4.1.4 Simulation Datasets

In the course of this thesis, an extensive exploration of the physical properties inherent to the VCOTA and FVCOTA circuits was undertaken. This involved a meticulous examination of the operational boundaries and saturation characteristics of the amplifiers, paving the way for comprehensive investigations into their dynamic behaviors.

To emulate and apprehend the dynamic intricacies, a multitude of simulations were executed. Diverse types of input waveforms, including sinusoidal, pulsed, and piecewise-linear (PWL), were systematically employed, each characterized by varying amplitudes and frequencies. These simulations were meticulously orchestrated through the utilization of NGSpice, offering a robust platform for circuit-level modeling and analysis.

The datasets obtained from these simulations were temporally distributed, encapsulating transient analyses. Throughout these investigations, certain foundational parameters were deliberately pre-defined, such as a constant component temperature (e.g., 25ºC) and a uniform power supply voltage ranging from 1V to 5V.

4.1.5 Transient Analysis

In order to simulate various waveforms, a transient analysis was conducted in the circuit. This analysis incorporated a critical component in the form of a DC bias current source, set at 1.65 amperes,

providing a stable operating point.

The simulation utilized a time step ranging from 1e-6 seconds to 10e-6 seconds, enabling fine-grained monitoring of the circuit's behavior. The choice of this time step range was crucial in capturing the intricate dynamics of the circuit's output response.

The stop time for the transient analysis was established at around 5e-3 seconds, carefully chosen to encompass the desired duration for extracting multiple datasets.

This time span was optimal for capturing the circuit's response to varying input waveforms and would serve as valuable training data for the neural networks, ensuring a comprehensive exploration of the circuit's dynamic behavior.

4.1.6 Output Data

Output data, reflective of the circuit's dynamic response, were logged into text files, as shown for one example in Table 4.3, and then output plots were stored to analyze the different simulations. Four distinct output nodes within the circuit were chosen to provide a comprehensive overview of the dynamic behavior across various stages. These multiple datasets, each emanating from different input waveforms, served as a valuable resource for training neural networks.

The output data was comprised of either 2000 timesteps or 5000 timesteps, with the choice dependent on the frequency of the input waveform and the complexity of the neural network architectures being used. This selection aimed to ensure that the circuit's response was accurately captured. While some early attempts involved simulations with unevenly spaced timesteps, the majority of used datasets were logged with equally spaced timesteps.

Equally spaced timesteps were obtained by linearizing the output of the NGSpice simulation. This choice was motivated by the suitability of Long Short-Term Memory (LSTM) networks for processing time series data, where observations are ordered by time. However, it's important to note that other combinations of unevenly spaced timesteps may yield favorable results, especially in regions of the simulation where the circuit behaves non-linearly.

In contrast, standard behavior regions may require fewer timesteps, as their output can be more easily extrapolated. This flexibility in timestep selection allows for a more tailored approach to capturing the circuit's dynamics across different scenarios.

Timesteps	V(net034)	V(outputp)	V(outputn)	V(output)
0.00000000e+00	0.00000000e+00	1.53208732e+00	5.47699792e-09	1.53208732e+00
2.50000000e-06	-5.20833333e-03	1.53223352e+00	-4.29218236e-03	1.53652570e+00
5.00000000e-06	-1.04166667e-02	1.53238096e+00	-8.60175416e-03	1.54098272e+00
7.50000000e-06	-1.56250000e-02	1.53252841e+00	-1.29121506e-02	1.54544057e+00
1.00000000e-05	-2.08333333e-02	1.53267583e+00	-1.72225792e-02	1.54989841e+00
1.25000000e-05	-2.60416667e-02	1.53282325e+00	-2.15339945e-02	1.55435724e+00
1.50000000e-05	-3.12500000e-02	1.53297063e+00	-2.58455229e-02	1.55881615e+00
1.75000000e-05	-3.64583333e-02	1.53311803e+00	-3.01580896e-02	1.56327612e+00
2.00000000e-05	-4.16666667e-02	1.53326539e+00	-3.44708486e-02	1.56773624e+00
⋮	⋮	⋮	⋮	⋮
4.98000000e-03	1.50000000e+00	1.48533897e+00	1.33823928e+00	1.47099689e-01
4.98250000e-03	1.50000000e+00	1.48533897e+00	1.33823927e+00	1.47099702e-01
4.98500000e-03	1.50000000e+00	1.48533897e+00	1.33823925e+00	1.47099716e-01
4.98750000e-03	1.50000000e+00	1.48533897e+00	1.33823924e+00	1.47099730e-01
4.99000000e-03	1.50000000e+00	1.48533897e+00	1.33823923e+00	1.47099744e-01
4.99250000e-03	1.50000000e+00	1.48533897e+00	1.33823921e+00	1.47099758e-01
4.99500000e-03	1.50000000e+00	1.48533897e+00	1.33823920e+00	1.47099771e-01
4.99750000e-03	1.50000000e+00	1.48533897e+00	1.33823919e+00	1.47099785e-01
5.00000000e-03	1.50000000e+00	1.48533897e+00	1.33823918e+00	1.47099796e-01

Table 4.3: Example of an output dataset from NGSpice.

In the context of this thesis, the predominant focus remained tethered to a single node for simulation and prediction purposes. However, this choice serves as a catalyst for future, more intricate explorations, encouraging a comprehensive analysis of the nuances associated with employing various nodes, each possessing its distinct and intricate behavioral traits.

4.1.7 Noise

One important aspect to take into consideration while extracting datasets from NSGpice is random variations. Even tough SPICE simulations take into account the statistical variations associated with real-world components, components such as resistors, capacitors, and transistors have tolerances, which can introduce random variations in their electrical properties. This inherent variability contributes to the noise in the simulation results.

Altough these factors were not considered in this thesis, understanding them and appropriately modeling them in the simulation can help characterize and manage the noise in the simulation outputs.

4.2 Simulation Datasets

As mentioned earlier, multiple datasets were extracted to facilitate the simulations. These datasets are evenly distributed across time, and for each dataset type, two additional variations were generated, denoted as an *a* and *b* versions (e.g., VCOTA23a.cir). The *a* version featured a slight variation in amplitude compared to the original *.cir* file, while the *b* version had its own distinct characteristics. This approach was employed to augment the quantity of datasets available for training, validation, and subsequent testing phases. The increase in datasets has proven to enhance the overall quality of the results.

4.2.1 Sinusoidal Datasets

Numerous sinusoidal input waveforms were carefully chosen, featuring diverse amplitudes and frequencies. However, it is noteworthy that a predominant portion of these sinusoidal datasets was purposely defined to operate within the standardized operational boundaries of the circuits. This deliberate choice was made to establish a foundational baseline within a linear range, optimizing the efficacy of training the neural networks.

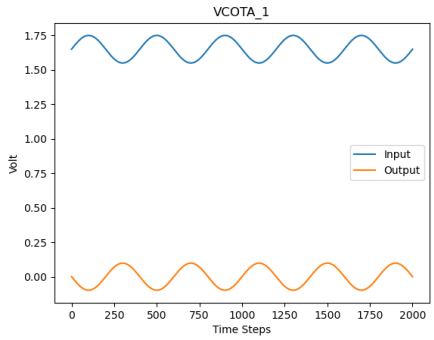


Figure 4.5: Normal Sinusoidal.

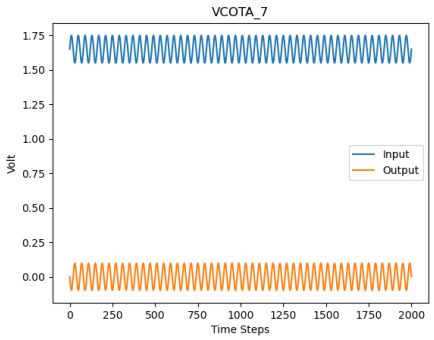


Figure 4.6: High frequency sinusoidal.

4.2.2 Square Wave and Pulse

A selection of square waveforms was thoughtfully created with the specific intent of unveiling nonlinear characteristics inherent within the circuits. These square waves, renowned for their ability to expose nonlinear effects such as distortion, clipping, and saturation, were key in assessing the circuit's response to these instantaneously change in input stimulus. Furthermore, the application of square waveforms allowed for the comprehensive evaluation of transient responses, demonstrating the circuit's behavior during abrupt transitions.

In addition to square waves, a variety of pulse waveforms were carefully selected to serve a dual purpose,timing analysis and model validation before progressing to the simulation of more intricate and nonlinear waveforms. This deliberate sequence of waveform utilization facilitated a systematic and robust evaluation of the circuit's performance characteristics.

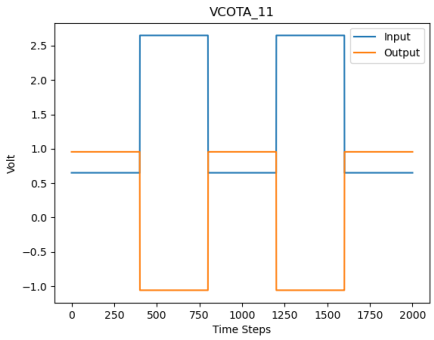


Figure 4.7: Squared Input.

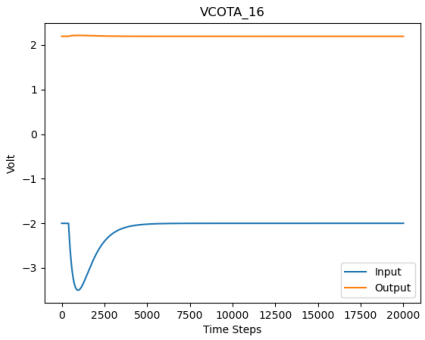


Figure 4.8: Isolated Pulse.

4.2.3 Piecewise Linear (PWL)

The vast majority of datasets employed for simulation purposes predominantly comprised variations of Piecewise Linear (PWL) waveforms. This predominant utilization of PWL waveforms was primarily driven by their exceptional flexibility, a characteristic inherent to their definition. This adaptability facilitated rigorous testing of the circuit under diverse and challenging circumstances, encompassing non-linear and dynamic operational ranges. It is this inherent flexibility that rendered PWL waveforms invaluable in assessing the circuit's performance across a wide spectrum of conditions, thereby enhancing the comprehensive nature of the simulations.

Notably, some illustrative examples of these PWL waveforms are visually presented in the images below, showcasing their versatility in the evaluation of circuit behavior.

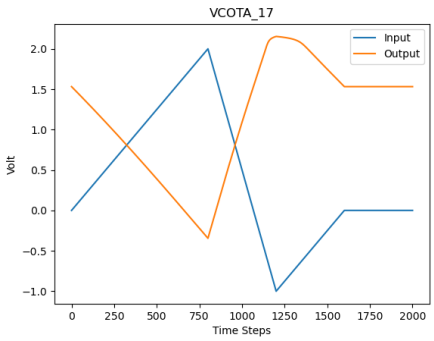


Figure 4.9: PWL 1.

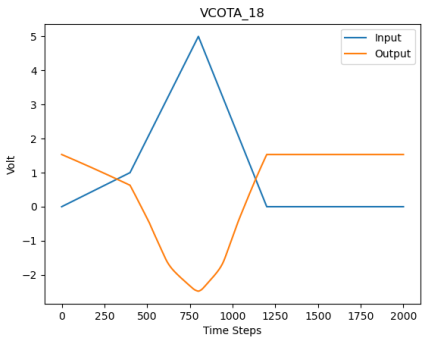


Figure 4.10: PWL 2.

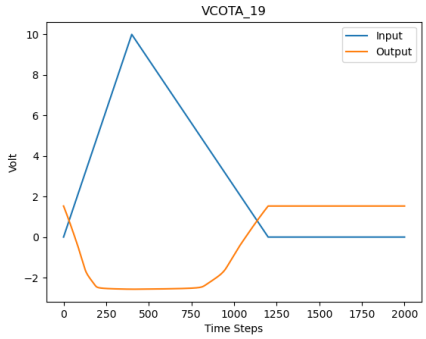


Figure 4.11: PWL 3.

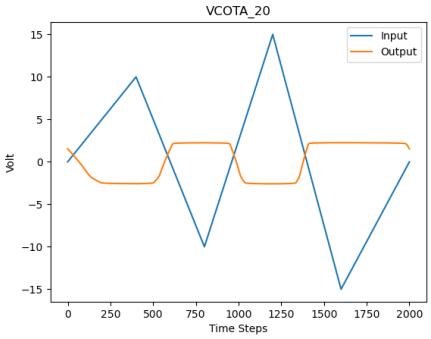


Figure 4.12: PWL 4.

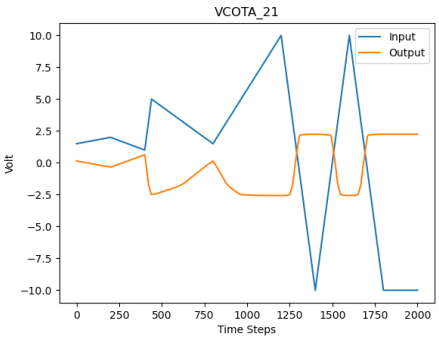


Figure 4.13: PWL 5.

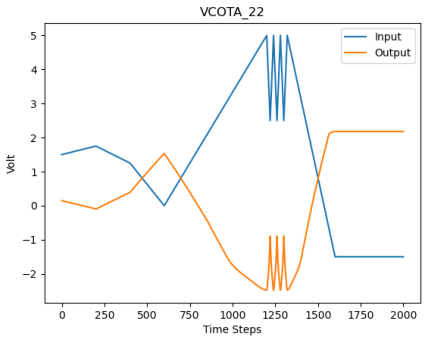


Figure 4.14: PWL 6.

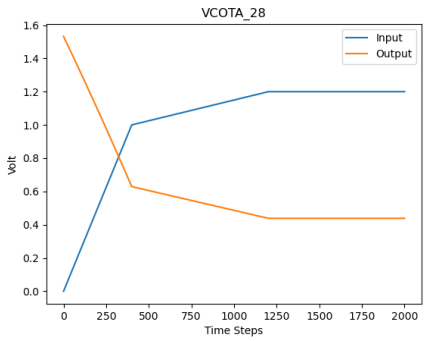


Figure 4.15: PWL 7.

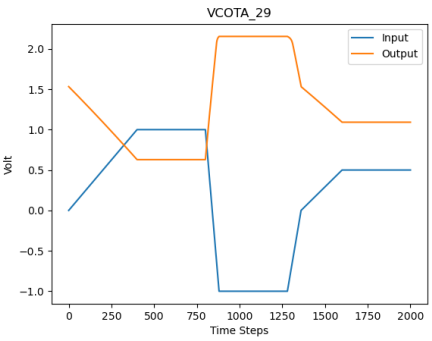


Figure 4.16: PWL 8.

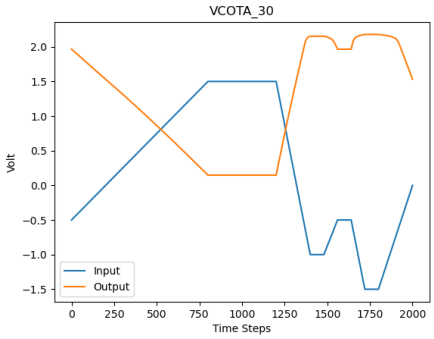


Figure 4.17: PWL 9.

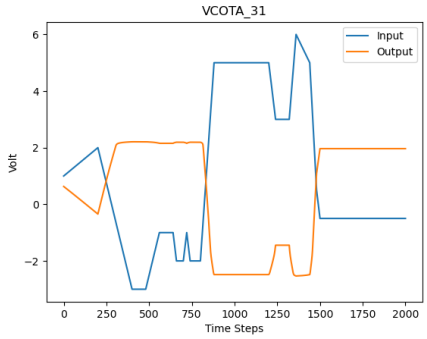


Figure 4.18: PWL 10.

4.3 Differential Neural Network based on Meyer's model

A simplified iteration of Meyer's model was realized through the utilization of the *NeuralODE* module in PyTorch. This approach allowed for the definition of the output function of each neuron as a second-degree ordinary differential equation, effectively encapsulating the dynamic behavior of these neurons.

Furthermore, certain aspects of the model were intentionally left unexplored. The parameters δ_{ik} and the timing parameters $\tau_{1,ik}$ and $\tau_{2,ik}$, which play a role in shaping the behavior of the neurons and were controlled by non-linear equations, were not explored.

4.4 Types of Simulations Performed

As the number of timesteps in the dataset increased, it became evident that the code developed to mimic the model described in Peter Meyer's thesis encountered numerical instability issues during computation. These issues can arise when the values involved in the calculations become too large or too small, leading to overflow or underflow errors.

To address these numerical instability concerns, several measures were taken. Firstly, the input data was subjected to a scaling process, where the values were transformed to a smaller range. This scaling helped to mitigate the impact of large input values and promoted more stable computations throughout the model.

Additionally, a careful selection of the activation function was made. The chosen activation function exhibited stability properties, ensuring that the computations remained well-behaved even with the increased timesteps in the dataset. Functions such as the hyperbolic tangent (tanh) or rectified linear unit (ReLU) were considered as suitable alternatives to the sigmoid function, which is known to be more susceptible to numerical instability.

Another crucial step in addressing numerical instability was the adjustment of the learning rate. By reducing the learning rate, the model ensured that the updates to the weights and biases were more controlled, preventing erratic and unstable changes. This adjustment helped to maintain the stability of the model throughout the training process.

To further monitor and regulate the stability, the gradients during training were carefully observed. This enabled the detection of any instances of vanishing or exploding gradients, which could contribute to numerical instability. By keeping the gradients within reasonable bounds, such as through the application of gradient clipping techniques, the model's stability was safeguarded.

These measures ensured the robustness and reliability of the model's computations, enabling accurate predictions and insights into the dynamic behavior of the systems under investigation.

Chapter 5

Results

Several architectures were tested and the complexity was gradually increased in order to reach the minimum possible complexity that would yield some good results and compare it with the same architecture but with more neurons, epochs and different batch size and also compare it with more complex architectures in general.

5.1 Comparison between models

Dynamic systems are characterized by their time-varying behavior, making them challenging to model accurately. Machine learning techniques have shown promise in capturing the intricate dynamics exhibited by such systems. In this chapter, we explore and compare various machine learning methods to identify the most effective approach for modeling dynamic systems. Specifically, we examine the performance of standard feedforward neural network, differential neural network, CTRNN, LSTM and Transformer.

5.1.1 Baseline Setup

To conduct a fair comparison, as previously mentioned, the simulations used the same datasets of electric circuit behaviors, comprising time-series measurements of electrical variables, such as voltage and current, in response to different input stimuli. These datasets were thoughtfully designed to encompass both linear and non-linear responses from the circuit. Several parameters were established as a baseline for consistency across all models. The training procedure involved the use of the Adam optimization algorithm with a fixed learning rate of 0.05. The loss function used was the mean squared error (MSE), ensuring uniformity in the evaluation process. Additionally, the same set of hyperparameters were selected for each model.

These meticulous measures aimed to ensure a level playing field for model comparison. It was recognized that the datasets, though obtained through simulations in artificial simulators (nmainly NGSpice but also LTSpice), accurately represent real-world electrical circuits and systems. Therefore, these steps were taken to guarantee the validity of the conclusions presented in the following sub-section.

For each model, the loss function was logged during the training process, and the results were systematically compared. The detailed comparative results are provided in the subsections below.

5.1.2 Performance Metrics

While there exist a multitude of performance metrics to assess the efficacy of machine learning techniques, the selection of a consistent and reliable metric is paramount for meaningful comparisons between models. For this purpose, Mean Squared Error (MSE) and its square root, Root Mean Squared Error (RMSE), were chosen. MSE quantifies the average squared difference between predicted and actual values, making it a widely accepted measure. RMSE, on the other hand, is the square root of MSE and provides a more interpretable measure in the same units as the target variable. The choice of these metrics ensures a safe and easily comparable baseline for model evaluation and fosters a clearer understanding of predictive accuracy.

In Figure 5.1, we present an illustrative case demonstrating the training and validation loss progression of an LSTM neural network. This network was trained using the Adam optimizer with a configuration comprising 32 neurons, 150 training epochs, and a batch size of 32. A noteworthy observation is the abrupt increase in the loss metric occurring around the 45th epoch. This phenomenon raises the suspicion that the learning rate employed may have been excessively high, leading to a decision to subsequently lower it in other simulation runs to achieve a more gradual decline in the loss function.

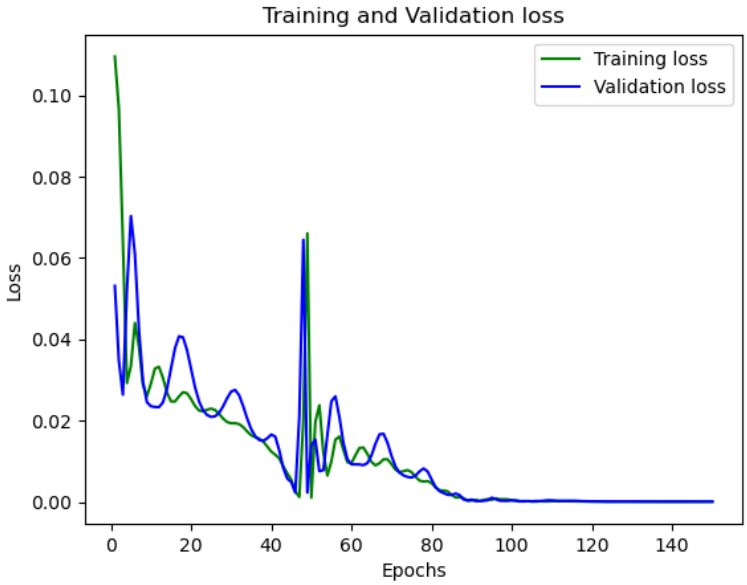


Figure 5.1: Training and validation loss for the LSTM model.

5.2 Differential Neural Network

The differential neural network, inspired by Meyer's PhD model, has shown promise in modeling complex dynamic systems. However, to fully unleash its potential, further fine-tuning and rigorous training are essential. With enhanced parameter optimization and more extensive training data, the model can deliver even more accurate and robust results, solidifying its efficacy in dynamic system modeling.

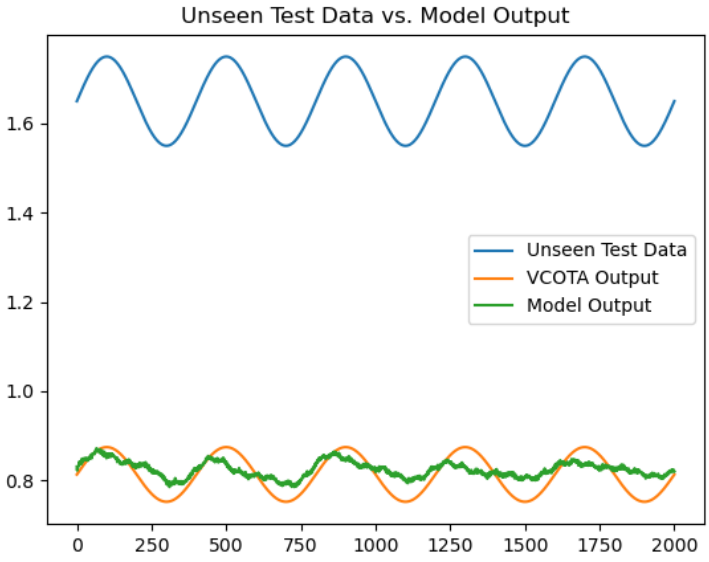


Figure 5.2: Output of the Differential Neural Network.

5.2.1 Noisy Results

Since a second-order differential equation neural network is specifically designed to model systems governed by second-order differential equations, such as electrical circuits. It includes additional layers and parameters to capture the dynamics and behavior of these systems accurately. However, this complexity can sometimes lead to more noisy outputs compared to other neural networks used for modeling input-output relationships.

There are a few reasons why a second-order differential equation neural network may produce more noisy outputs in the context of electrical circuit modeling:

Nonlinear Dynamics: Electrical circuits often exhibit nonlinear behavior, which can introduce complexity and sensitivity to initial conditions. Second-order differential equation neural networks aim to capture this nonlinear dynamics, but it can be challenging to train them effectively. The model may struggle to learn and generalize the intricate dynamics of the circuit accurately, resulting in noisy outputs.

To mitigate the issue of noisy outputs in second-order differential equation neural networks, one might employ regularization techniques during training, adjusting the network architecture, refining the training process, or incorporating noise reduction methods in the post-processing of the network's predictions.

5.3 CTRNN

CTRNNs, operating in continuous time, excel in modeling sinusoidal waves due to their inherent capacity to capture and process continuous, time-varying signals. This continuous-time operation aligns naturally with the smooth, periodic nature of sinusoidal waves, enabling CTRNNs to precisely replicate their oscillatory behavior and maintain accurate phase relationships, making them an ideal choice for sinusoidal wave modeling.

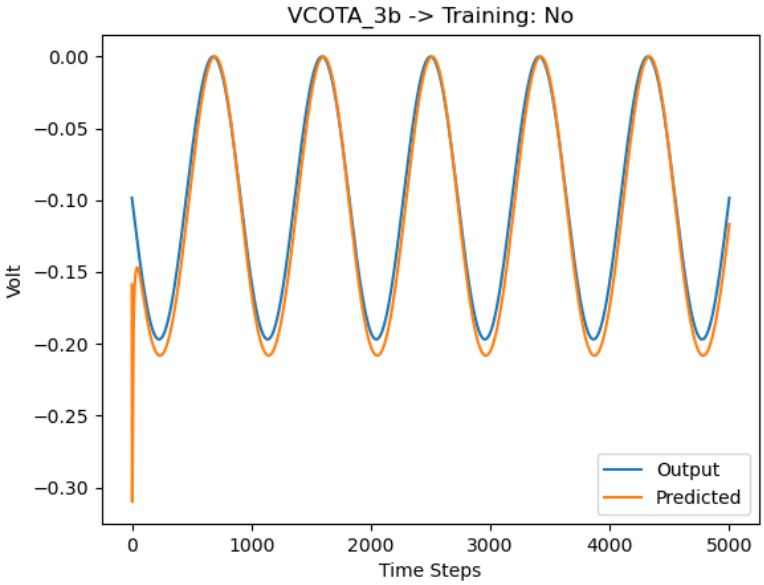


Figure 5.3: Output of the CTRNN.

Throughout the thesis, as simulations followed a sequence-to-sequence approach where each input neuron received a sequence, a recurring challenge emerged. The initial point in the output often exhibited significant inaccuracies, resulting in substantial deviations from the expected values. This issue was particularly pronounced due to the sequential nature of the modeling approach and the way the model's architecture was designed.

5.4 LSTM

As expected, the LSTM model accurately captured the nuances of various stimuli, showcasing its proficiency in encoding and predicting complex temporal patterns. Its inherent ability to handle diverse inputs reinforced its suitability for modeling a wide array of sequential data.

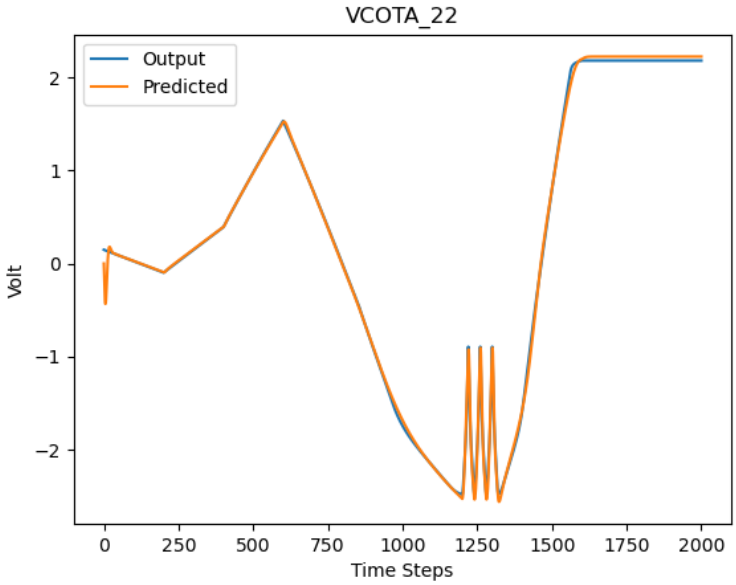


Figure 5.4: Output of the LSTM Neural Network.

5.5 Stacked LSTM

Among the tested neural network architectures, it was evident that the stacked Long Short-Term Memory (LSTM) model consistently outperformed the others. The stacked LSTM configuration demonstrated superior performance in capturing complex temporal dependencies and yielded the best results across various evaluation metrics, underscoring its effectiveness in modeling sequential data.

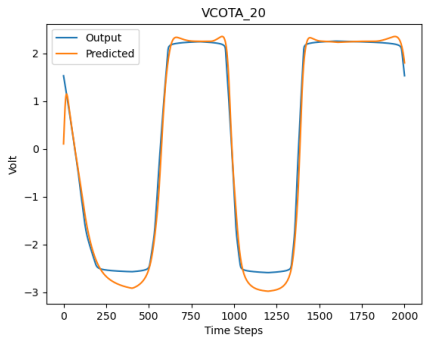


Figure 5.5: Stacked LSTM 1.

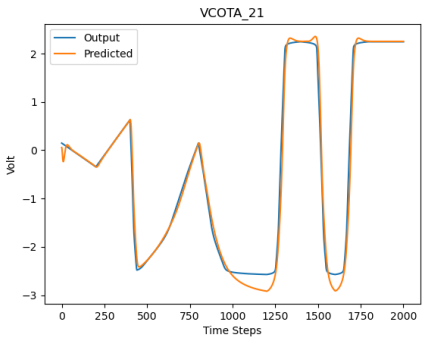


Figure 5.6: Stacked LSTM 2.

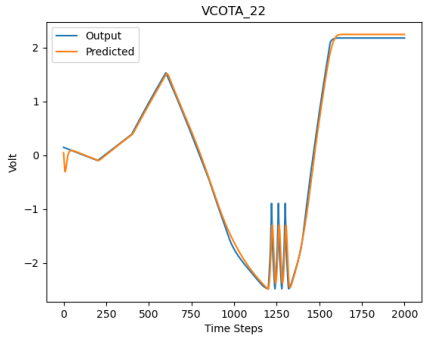


Figure 5.7: Stacked LSTM 3.

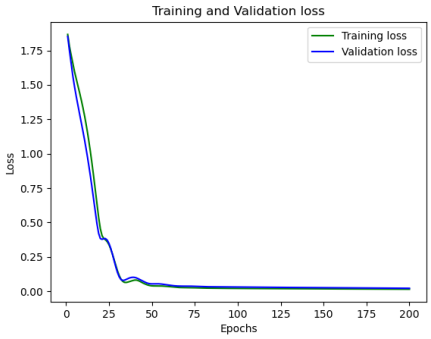


Figure 5.8: Loss for the Stacked LSTM.

5.6 Transformer

Despite being considered state-of-the-art for sequence-to-sequence modeling, our transformer model faced limitations in capturing the full amplitude of high-frequency input signals. This challenge highlighted the need for additional strategies to address the intricacies of signal dynamics and ensure accurate representation in our modeling approach.

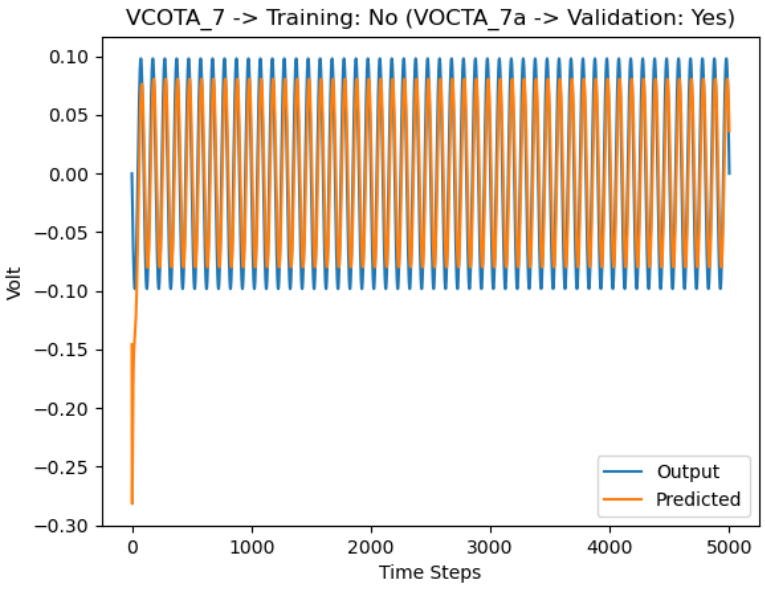


Figure 5.9: Transformer output for a high-frequency input.

5.7 RMSE Evolution Comparison

The table below provides a clear comparison between the performance of the implemented neural networks by showcasing the number of epochs that the model required to ahcieve a value of 0.1 in Root Mean Square Error (RMSE), assuming all networks configured in the baseline configuration. This tabulated data serves as a comprehensive reference for tracking how the performance of these networks evolves throughout the modeling process. By analyzing the RMSE values at different stages, one can gain insights into the networks' predictive accuracy and convergence.

Neural Networks	**Number of Epochs to reach RMSE of 0.1**
Differential NN	—
CTRNN	220
LSTM	61
Stacked LSTM	45
Transformer	150

Table 5.1: Number of epochs to reach the RMSE value of 0.1.

The results obtained for the number of epochs required to reach an RMSE value of 0.1 provide valuable insights into the performance of various neural network architectures in modeling the given simulations. Among the architectures tested, the stacked LSTM architecture exhibited the best performance, demonstrating its capability to effectively approximate the underlying system dynamics within a relatively shorter training period.

Surprisingly, the Transformer architecture did not perform as well in this context. While it has the potential to provide superior results due to its focus on solving sequence-to-sequence tasks, which align with the nature of the input-output relationship in this study, it suggests that careful adjustments in hyperparameters or model architecture may be necessary to harness the full modeling potential of the Transformer.

Notably, the CTRNN implementation, which Peter Meyer's model is a precursor of, showed remarkable efficiency in reaching the RMSE of 0.1 in only 220 epochs. This result is particularly intriguing as it highlights that neural networks that operate around continuous-time, pose the inherent capacity to accurately model complex dynamic systems, provided they are well-adjusted and appropriately parameterized. These findings emphasize the importance of choosing the right architecture and optimizing model parameters for effectively capturing the dynamics of the system under investigation.

5.8 Results and Discussion

Having established the baseline of evaluation metrics, the results for each model were as follows: the models yielded very different outcomes, showcasing a significant variance in their performance. These divergent results underscored the need for in-depth analysis and further investigation to comprehend the factors contributing to the observed disparities in model performance.

5.8.1 Differential Neural Network

The differential neural network exhibits improved performance compared to the standard feedforward neural network. By incorporating recurrent connections, it can capture some temporal dependencies. However, it falls short in accurately capturing the intricate dynamics of the electrical circuits. The performance metrics suggest lower accuracy, higher errors, and weaker model fit than desired. However, the model was still far worse than the other neural network, not reaching the desired RMSE value of 0.1.

5.8.2 CTRNN

CTRNN, as a continuous-time recurrent neural network, holds promise for modeling dynamic systems due to its inherent temporal nature. Nevertheless, the experimental outcomes reveal that CTRNN does not adequately capture the dynamics of the electric circuits. The obtained performance metrics demonstrate limited accuracy for other than sinusoidal modeling and higher errors compared to other methods.

5.8.3 LSTM

In contrast to the aforementioned methods, LSTM demonstrates superior performance in modeling dynamic systems. The LSTM architecture, specifically designed to address the challenge of capturing long-term dependencies, excels at capturing the intricate dynamics of the electric circuits. The experimental results showcase significantly lower MSE, RMSE, and MAE values, indicating better accuracy, precision, and model fit.

LSTM Layer Architecture

When comparing the impact of the architecture on modeling dynamic behavior in electric circuits, it should be noted the layer architecture of the LSTM models played a significant effect on the accuracy of the simulations. Notably, having more LSTM layers (stacked LSTM) but with the same overall number of neurons, such as the case where one model had one-single layer with 64 neurons, while the other had

two layers with 32 neurons, the simulations yielded better results for the architecture with more layers, indicating the advantages of utilizing deeper architectures for this task.

The explanation rests in the case that the addition of more LSTM layers introduces increased depth and complexity to the neural network model. Each layer can learn and represent different levels of abstraction and temporal dependencies in the data. In the case of dynamic behavior in electric circuits, having multiple layers enables the model to capture both low-level and high-level temporal patterns, facilitating a more hierarchical representation of the system's dynamics.

With a single LSTM layer and 64 neurons, the model can capture some temporal dependencies and patterns in the data. However, this architecture may have limitations in capturing more complex and subtle dynamics exhibited by electric circuits. In contrast, employing two LSTM layers with 32 neurons each allows for a deeper and more hierarchical representation. The first layer can focus on learning low-level temporal dependencies, while the second layer can build upon the outputs of the first layer to capture higher-level abstractions and more intricate temporal relationships.

The improved performance observed in simulations using the architecture with more layers can be attributed to several factors. First, the additional layers provide the model with increased representational capacity, enabling it to learn and model more intricate dynamics. This added capacity allows for a more accurate representation of the complex interactions and dependencies present in electric circuits.

Second, the hierarchical nature of the deeper architecture allows for a more efficient and effective extraction of features from the input data. The lower-level layers capture and process fine-grained temporal patterns, while the higher-level layers aggregate and interpret these patterns to form a holistic understanding of the dynamic behavior of the circuits. This hierarchical feature extraction facilitates a more comprehensive and nuanced modeling of the system.

Furthermore, the additional layers in the architecture introduce more non-linear transformations, enhancing the model's ability to capture and represent highly non-linear dynamics exhibited by electric circuits. The increased depth provides the neural network with greater flexibility in mapping the input variables to the desired output, allowing for more accurate predictions and better overall model performance.

In conclusion, the simulations conducted in modeling dynamic behavior in electric circuits demonstrate the advantages of employing architectures with more LSTM layers, even when keeping the overall number of neurons the same. The increased depth and hierarchical nature of these architectures enable the model to capture complex temporal dependencies, extract meaningful features, and accurately represent the intricate dynamics of the circuits. Therefore, it can be inferred that the additional layers provide the neural network with enhanced modeling capabilities, resulting in improved performance for this specific task.

Time Distributed Layer

LSTM networks have proven effective in capturing temporal dependencies and modeling sequential data in deep learning. However, challenges arise in sequence-to-sequence prediction tasks where input and output sequences have varying lengths. To address this, the time distributed layer becomes a crucial architectural component in LSTM networks. This layer allows the application of a specific operation to each time step of the input sequence independently, such as a dense layer or convolutional layer.

The time distributed layer operates on the input sequence, executing the designated operation at each time step and generating a sequence of outputs that captures temporal information. Importantly, this layer ensures that the weights and biases associated with the operation are shared across all time steps. This parameter sharing facilitates the learning process, enabling the model to capture long-term dependencies and generalize effectively to sequences of varying lengths.

In the context of sequence-to-sequence prediction tasks, the time distributed layer is often followed by a fully connected layer or another appropriate layer. This additional layer helps map the generated sequence of outputs to the desired output sequence length, facilitating sequence transformation and accurate predictions based on learned temporal patterns. In summary, the time distributed layer plays a fundamental role in enhancing the capabilities of LSTM networks for sequence-to-sequence prediction, making it an essential tool for deep learning practitioners working with sequential data.

Chapter 6

Conclusions

In this thesis, it was conducted a comparative analysis of machine learning methods for modeling dynamic systems. The experimental findings reveal that LSTM outperforms standard feedforward neural networks, dynamic neural networks, CTRNN, and even the Transformer-based neural network in capturing the dynamic behavior of electric circuits. LSTM achieves superior performance in terms of accuracy, precision, and model fit.

6.1 Achievements

This thesis stands as a testament to the accomplishments achieved through the utilization of machine learning techniques in the modeling of dynamic systems. The journey began with Peter Meyer's pioneering PhD thesis in the 1990s, which introduced a model based on second-order differential equations, acting as a precursor to the Continuous-Time Recurrent Neural Network (CTRNN). Over time, advancements have been made, culminating in the adoption of the highly effective Long Short-Term Memory (LSTM) model as one-of the optimal choice for modeling dynamic systems.

The research focused specifically on the intricate task of modeling non-linear, multidimensional, and multivariable behavior in electric circuitry, a challenge that persists in the broader context of dynamic systems. The findings uncovered certain limitations, including occasional noise interference in the model outputs, difficulties in accurately determining initial conditions, and struggles in capturing abrupt or highly non-linear behavior.

However, the thesis also presented a positive outlook by exploring innovative techniques to address these limitations. Notably, researchers have started incorporating more complex architectures and mechanisms into their models, such as advanced recurrent neural networks, attention mechanisms, and hybrid approaches that integrate various machine learning techniques. These novel methodologies have shown great promise in mitigating the aforementioned challenges and improving model performance.

The primary objective of this thesis was to showcase the immense potential of machine learning techniques in capturing the behavior of dynamic systems. By highlighting the accomplishments achieved and laying the groundwork for future research endeavors, this work contributes significantly to the advancement of this field.

6.2 Future Work

6.2.1 Introduction

This thesis has explored various neural network architectures for modeling dynamic behavior in different systems. The results obtained have shed light on the potential of neural networks in capturing complex temporal dependencies and dynamics. However, this research only scratches the surface of the vast possibilities that lie ahead. In this chapter, we discuss potential avenues for future work that can build upon the findings of this thesis and pave the way for new advancements in the field of dynamic system modeling.

6.2.2 Custom Architectures

The study of custom architectures involving combinations of neural network types has emerged as a promising avenue for modeling the dynamic behavior of circuit simulations and dynamic systems in a comprehensive and efficient manner. Throughout this thesis, we have explored various neural network architectures, including feedforward neural networks (FNNs), recurrent neural networks (RNNs), convolutional neural networks (CNNs), graph neural networks (GNNs), and transformer-based models. The amalgamation of these distinct architectural elements holds significant potential in enhancing the accuracy and versatility of circuit simulation models.

One of such examples of custom or hybrid architectures are showcased in [27], where a pioneering approach to macromodeling nonlinear circuits using deep independently recurrent neural networks (DIRNN) is introduced. DIRNN mitigates issues like vanishing and exploding gradients commonly seen in traditional RNNs. Unlike conventional RNNs, where all neurons in a layer are interconnected, DIRNN ensures independence among neurons by allowing each to solely rely on its prior hidden state. This innovation enhances model simplicity, thus facilitating effective training for extended sequences. The method's effectiveness is validated through modeling two complex nonlinear circuit examples: a multistage driver with multiline interconnect termination and an on-chip voltage generator.

By integrating FNNs, which excel in capturing simple relationships, with RNNs, which specialize in handling sequential and time-dependent data, custom architectures can effectively model the transient and time-varying characteristics of circuits. This integration is particularly valuable when dealing with

dynamic systems involving feedback loops and complex signal dynamics. Furthermore, incorporating CNNs can facilitate the analysis of complex circuit layouts and image-based data, providing additional insights into electric circuit simulations with intricate geometries.

Graph neural networks (GNNs), designed for graph-structured data, have proven invaluable in the analysis of complex circuits, as they can capture the intricate interconnections and dependencies between circuit components. These architectures are instrumental in modeling the behavior of large-scale circuits and optimizing circuit layout designs for improved performance. The introduction of transformer-based models, originally developed for natural language processing tasks, can enable a more holistic understanding of textual or tabular circuit descriptions, aiding in automated circuit design and specification interpretation.

Empirical evidence from recent research articles underscores the efficacy of hybrid neural network architectures in the realm of electric circuit modeling.In the work shown in [28], showcased the potential of combining CNNs and RNNs, illustrating the versatility of hybrid architectures in practical applications.

In conclusion, the synthesis of custom neural network architectures by combining the diverse types explored in this thesis represents a compelling approach to capture the dynamic behavior of circuit simulations and dynamic systems at large. These architectures, tailored to the specific characteristics of the problem at hand, offer the potential to significantly advance the state-of-the-art in circuit modeling, paving the way for innovative solutions in the domain of electrical engineering and beyond.

6.2.3 Other Machine Learning Techniques

While neural networks have shown great promise in modeling dynamic systems, there is an opportunity to leverage the strengths of other machine learning techniques in combination with neural networks. For instance, incorporating reinforcement learning algorithms can allow for the development of adaptive neural network models that can continuously learn and improve their performance in dynamic environments. Hybrid approaches that integrate genetic algorithms, particle swarm optimization, or fuzzy logic with neural networks could also yield novel solutions for capturing dynamic behavior. One such way of combining different models to create a powerful architecture is through the usage of a hybrid architecture between an LSTM and the novel Transformer architecture, which holds great promise for modeling dynamic systems due to the combined strengths of these two approaches.

As previously seen, LSTM networks, [29] excel at capturing temporal dependencies in sequential data, since they maintain an internal memory cell that allows them to retain information over long sequences.

On the other hand, Transformer architectures, [30] have revolutionized tasks like machine translation with the introduction of the ”attention” mechanism. Transformers capture dependencies between different parts of a sequence without relying on recurrence, making them more parallelizable and computa-

tionally efficient. They excel at capturing global dependencies and have shown remarkable performance on various natural language processing tasks.

By combining LSTM and Transformer architectures, we can leverage the strengths of both models to create a more powerful tool for modeling dynamic systems. This integration offers several advantages such as long-range dependencies, better sequential modeling, enhanced flexibility and adaptability and scalability and parallelism.

In conclusion, the combination of LSTM and Transformer architectures holds tremendous potential as a powerful tool for modeling dynamic systems. By leveraging the strengths of both approaches, the model can effectively capture short and long-term dependencies, perform efficient sequential modeling, enhance representation learning, and address a wide range of dynamic system modeling tasks.

6.2.4 Integration of Domain Knowledge

Dynamic systems often possess inherent domain-specific characteristics and constraints. Future work can focus on incorporating domain knowledge into the design of neural network models. By integrating prior knowledge about the system dynamics, the models can achieve better interpretability, generalization, and robustness. Techniques such as transfer learning, domain adaptation, or the inclusion of physical laws and constraints can be explored to enhance the modeling capabilities of neural networks in dynamic systems.

6.2.5 Evaluation Metrics and Performance Analysis

Further research is needed to develop comprehensive evaluation metrics and performance analysis techniques specifically tailored for dynamic system modeling. While metrics like MSE, RMSE and MAE have been employed in this thesis, they might not capture the full complexity and characteristics of dynamic behavior. Future work can focus on the development of novel metrics that consider the time-varying nature, nonlinearity, and temporal dependencies exhibited by dynamic systems.

6.2.6 Real-Time and Online Learning

One direction for future work is the exploration of real-time and online learning techniques for modeling dynamic behavior, as demonstrated in [31]. Traditional machine learning approaches often rely on offline batch learning, which might not be suitable for systems that exhibit rapid changes or require immediate adaptation. Investigating online learning algorithms that can continuously update the model in real-time based on incoming data streams can significantly enhance the modeling capabilities in dynamic environments.

6.2.7 Experimental Validation on Different Dynamic Systems

This thesis primarily focused on modeling dynamic behavior in electric circuits. However, future work should expand the scope and validate the developed neural network models on a diverse range of dynamic systems. Applications could include control systems, robotics, financial markets, biological systems, and climate modeling, among others. Conducting experiments on different dynamic systems will help assess the generalizability and transferability of the developed models.

6.2.8 Conclusion

This thesis has served as an exploratory work that demonstrates the potential of neural networks for modeling dynamic behavior. The future work outlined in this chapter aims to push the boundaries of this research and open up new possibilities for advanced modeling techniques. By combining neural networks with other machine learning techniques, developing new neural network architectures, incorporating domain knowledge, refining evaluation metrics, exploring real-time learning, and validating on diverse systems, we can unlock greater accuracy, interpretability, and robustness in modeling dynamic behavior, leading to advancements in various fields of science and engineering.

Bibliography

[1] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*, 2021.

[2] C. Mitrea, C. Lee, and Z. Wu. A comparison between neural networks and traditional forecasting methods: A case study. *International journal of engineering business management*, 1:11, 2009.

[3] D. J. Livingstone, D. T. Manallack, and I. V. Tetko. Data modelling with neural networks: advantages and limitations. *Journal of computer-aided molecular design*, 11(2):135–142, 1997.

[4] X. Cheng, B. Khomtchouk, N. Matloff, and P. Mohanty. Polynomial regression as an alternative to neural nets. *arXiv preprint arXiv:1806.06850*, 2018.

[5] H. Wang and B. Raj. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.

[6] M. A. Stošović and V. Litovski. Application of artificial neural networks in electronics. *Electronics*, 21(2):87–94, 2017.

[7] M. V. Andrejević and V. B. Litovski. Electronic circuits modeling using artificial neural networks. *Journal Of Automatic Control*, 13(1):31–37, 2003.

[8] M. Dieste-Velasco, M. Diez-Mediavilla, and C. Alonso-Tristán. Regression and ann models for electronic circuit design. *Complexity*, 2018, 2018.

[9] Y. Jiang, C. Yang, J. Na, G. Li, Y. Li, and J. Zhong. A brief review of neural networks based learning and control and their applications for robots. *Complexity*, 2017, 2017.

[10] S. Sun, Z. Cao, H. Zhu, and J. Zhao. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 50(8):3668–3681, 2019.

[11] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso. Deep learning for time series forecasting: A survey. *Big Data*, 9(1):3–21, 2021.

[12] M. Kawaguchi, T. Jimbo, and N. Ishii. Dynamic learning of neural network by analog electronic circuits. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 73–79. Springer, 2011.

[13] F. Djeffal, M. Chahdi, A. Benhaya, and M. Hafiane. An approach based on neural computation to simulate the nanoscale cmos circuits: Application to the simulation of cmos inverter. *Solid-State Electronics*, 51(1):48–56, 2007.

[14] N. Sinha, M. Gupta, and D. Rao. Dynamic neural networks: an overview. In *Proceedings of IEEE International Conference on Industrial Technology 2000 (IEEE Cat. No. 00TH8482)*, volume 1, pages 491–496. IEEE, 2000.

[15] P. Meijer. *Neural Network Applications in Device and Subcircuit Modelling for Circuit Simulation*. PhD thesis, TU Eindhoven, May 1996.

[16] K. M. Tarwani and S. Edem. Survey on recurrent neural network in natural language processing. *Int. J. Eng. Trends Technol*, 48:301–304, 2017.

[17] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 2021.

[18] R. D. Beer. The dynamics of adaptive behavior: A research program. *Robotics and Autonomous Systems*, 20(2-4):257–289, 1997.

[19] Z. Naghibi, S. A. Sadrossadat, and S. Safari. Dynamic behavioral modeling of nonlinear circuits using a novel recurrent neural network technique. *International journal of circuit theory and applications*, 47(7):1071–1085, 2019.

[20] S. Heinrich, T. Alpay, and S. Wermter. Adaptive and variational continuous time recurrent neural networks. In *2018 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 13–18. IEEE, 2018.

[21] A. Faraji, M. Noohi, S. A. Sadrossadat, A. Mirvakili, W. Na, and F. Feng. Batch-normalized deep recurrent neural network for high-speed nonlinear circuit macromodeling. *IEEE Transactions on Microwave Theory and Techniques*, 2022.

[22] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[23] H. Vogt, M. Hendrix, P. Nenzi, and D. Warning. Ngspice user's manual. http://ngspice.sourceforge.net/docs/ngspice-34-manual.pdf, January 2021. version 34 (ngspice release version).

[24] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[25] R. Povoa, N. Lourenço, N. Horta, R. Santos-Tavares, and J. Goes. A cascode-free single-stage amplifier using a fully-differential folded voltage-combiner. In *2014 21st IEEE international conference on Electronics, circuits and systems (ICECS)*, pages 263–266. IEEE, 2014.

[26] R. Povoa, N. Lourenço, N. Horta, R. Santos-Tavares, and J. Goes. Single-stage amplifiers with gain enhancement and improved energy-efficiency employing voltage-combiners. In *2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 19–22. IEEE, 2013.

[27] A. Faraji, S. A. Sadrossadat, A. Moftakharzadeh, M. Nabavi, and Y. Savaria. Deep independent recurrent neural network technique for modeling transient behavior of nonlinear circuits. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 2023.

[28] S. Y. Xiong, E. F. M. Gasim, C. X. Ying, K. K. Wah, and L. M. Ha. A proposed hybrid cnn-rnn architecture for student performance prediction. *International Journal of Intelligent Systems and Applications in Engineering*, 10(3):347–355, 2022.

[29] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[31] N. H. Phong and B. Ribeiro. Offline and online deep learning for image recognition. In *2017 4th Experiment@ International Conference (exp. at'17)*, pages 171–175. IEEE, 2017.