Explorations on chaotic behaviors of Recurrent Neural Networks

by

Bagdat Myrzakhmetov

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Master of Science in Applied Mathematics

at the

NAZARBAYEV UNIVERSITY

Apr 2019

Author .....
Department of Mathematics
Apr 30, 2019

Certified by .....
Zhenisbek Assylbekov
Assistant Professor
Thesis Supervisor

Certified by .....
Rustem Takhanov
Assistant Professor
Thesis Co-supervisor

Accepted by .....
Vassilios D. Tourassis
Dean, School of Science and Technology

Explorations on chaotic behaviors of Recurrent Neural Networks

by

Bagdat Myrzakhmetov

Submitted to the Department of Mathematics
on Apr 29, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Applied Mathematics

Abstract

In this thesis work we analyzed the dynamics of the Recurrent Neural Network architectures. We explored the chaotic nature of state-of-the-art Recurrent Neural Networks: Vanilla Recurrent Network, Recurrent Highway Networks and Structurally Constrained Recurrent Network. Our experiments showed that they exhibit chaotic behavior in the absence of input data. We also proposed a way of removing chaos chaos from Recurrent Neural Networks. Our findings show that initialization of the weight matrices during the training plays an important role, as initialization with the matrices whose norm is smaller than one will lead to the non-chaotic behavior of the Recurrent Neural Networks. The advantage of the non-chaotic cells is stable dynamics. At the end, we tested our chaos-free version of the Recurrent Highway Networks (RHN) in a real-world application.

In a sequence-to-sequence modeling experiments, particularly in the language modeling task, chaos-free version of RHN perform on par with the original version by using the same hyperparameters.

Thesis Supervisor: Zhenisbek Assylbekov
Title: Assistant Professor

Acknowledgments

First of all I would like to thank my supervisor, Professor Zhenisbek Assylbekov who suggested this interesting project, supported and gave direction throughout the year. Special thanks to Rustem Takhanov who was my second supervisor, who gave interesting suggestions. Also, I would like to thank Professor Anastasios Bountis, who is specialist in chaos theory, for his valuable feedback. Finally, I would like to thank my family, friends for supporting me every time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The dynamics of the Neural Networks has been studied in recent papers ([54, 40]). Laurent and Brecht ([27]) proposed to design architecture of a Recurrent Neural Network (RNN) cell in such a way that it is not chaotic. The concept of chaos ([53, 39]) comes from the theory of nonlinear dynamical systems and essentially means that wide divergence in outcomes of a system is due to small differences in initial conditions (such as those due to rounding errors in numerical computation). So, Laurent and Brecht show that the widely-used RNN cells, LSTM ([18]) and GRU ([10]), are chaotic. Depending on the initialization of the weights, LSTM and GRU might show a chaotic behavior. The proposed Chaos Free Network (CFN) architecture is devoid of chaos and is not inferior to LSTM. Recently, there were two main advancements over ubiquitous LSTM architecture: 1) Zoph and Le [60] used LSTM to generate a new RNN cell, which they refer to as a 'Neural Architecture Search' (NAS) cell; 2) Zilly et al. [59] extended the success of Highway networks ([51]) to recurrent networks and suggested a new RNN cell, which they refer to as a 'Recurrent Highway Network' (RHN) cell. Both, NAS and RHN cells significantly outperform the LSTM cell in language modeling tasks when evaluated on a traditional PTB dataset ([31]). Also, Structurally Constrained Recurrent Network (SCRN) proposed by Mikolov et al. [33] showed comparable results to the LSTM cell on the language modeling task. Therefore the following questions arise: Are these new state of the art architectures chaotic? If so, then according to Laurent and Brecht ([27]) there should be non-

chaotic alternatives that do not underperform significantly. And if there are no such analogs, can chaos be necessary after all? We will try to answer these questions in this thesis work.

Chapter two describes the chaotic behaviour of the simple Recurrent Neural Network (Vanilla RNN, Elman et al. [13]) architecture by using the different techniques and the way of the making the neural network non-chaotic.

Chapter three describes the dynamics of the Recurrent Highway Networks ([59]). This section also gives some inside information about the Recurrent Highway Networks and also suggests making the non-chaotic alternatives.

Chapter four describes the Structurally Constrained Recurrent Networks (SCRN by Mikolov et al. [33]) and its dynamics.

Chapter five describes real-world applications and different experiments performed by using the chaotic as well as non-chaotic Recurrent Neural Networks.

1.1 Motivations for exploring the dynamics of the RNN

The success of the Recurrent Neural Networks (RNN) in many sequence-to-sequence tasks attracted the attention of researchers to the analysis of the internal architecture of RNN cells. The idea of exploring the dynamics of Recurrent Neural Networks is motivated by recent studies on the dynamics of recurrent neural networks by Laurent and Brecht ([27]). There are several proposed new RNN architectures, and the aim is to identify whether they are chaotic or not. If these systems are chaotic, then we would propose non-chaotic neural cells. One thing, compared to Laurent and Brecht [27], instead of creating a new architecture, we try to suggest *ways* to make the RNN cells non-chaotic.

Another important motivation was an attempt to understand the internal structure of Recurrent Neural Networks. Even now, Recurrent Neural Networks, can be considered as a "black box", and there are some studies ([40, 54]) that tried to under-

stand the inside structure of recurrent neural networks. Now people want to have a "understandable" neural network. Dynamic Systems are a widely used technique for mathematical modeling of many real-world problems. We try to analyze Recurrent Neural Network from a dynamical point of view, i.e. we consider the RNN system as a dynamical systems.

Taking these ideas, it seems that the exploring of the dynamics of the recent, state of the art Recurrent Neural Network architectures is a decent idea. If we explore the chaos in the dynamics of Recurrent Neural Networks, we will try to suggest non-chaotic alternatives.

In the end, we compare chaotic and non-chaotic architectures in real world applications. We test these architectures in Language Modeling tasks.

1.2 Description of dynamics of the systems

A dynamical system Φ_t is any object or process for which the concept of state space is uniquely defined as a set of certain quantities at a given time t and a rule is given that describes the change (evolution) of the initial state over time [4]. One of the examples of a simple dynamical system described by $x_{t+1} = 5x_t$. Here the variable t stands for time and x_t denotes the state value at time t.

The rule that allows to predict the future state of the dynamic system by the initial state, is called the *rule of evolution*. Dynamical systems are mechanical, physical, chemical, and biological objects, computational processes, and information transformation processes performed in accordance with specific algorithms. The descriptions of dynamic systems for defining the rule of evolution are also varied: using differential equations, discrete mappings, graph theory, Markov chain theory, etc. The choice of one of the description methods sets a specific type of mathematical model with the corresponding system dynamics [3]. There are two types of the evolution rule: if the predicted next state has a unique consequent, then the system is called *deterministic* and if there are several consequent for a given state, then the system is called *stochastic* or *random*.

The definition of a dynamic system includes the state space s and the time-dependent t evolution operator Φ_t (rule), according to which the system from the initial state s_0 comes to the state s_t at the time t. The state of a dynamic system is described by a set of s variables chosen for reasons of their natural interpretation, simplicity of description, symmetry, etc. The set of states (phases) of a dynamic system forms a phase space in which a point corresponds to each state, and evolution is represented by moving a point along phase trajectory - a curve embedded in the phase space. For example, the motion of n particles under the action of attraction forces is described in the phase space by the set of all sets of coordinates and velocities of these particles, and the evolution operator is determined by the solution of the corresponding ODE system [2]. A state space could be continuous or discrete.

The set of instants of time t can be either an interval of the real line $\mathbb{R}$ (then it is said that time is continuous) or a set of integers or natural numbers (discrete time) [55]. In the second case, the "movement" of a phase space point is more like instantaneous "jumps" from one point to another: the trajectory of such a system is not a smooth curve, but simply a set of points, and is usually called an orbit. Nevertheless, despite the external difference, there is a close relationship between systems with continuous and discrete time: many properties are common to these classes of systems or are easily transferred from one to another [55].

Features of the evolution of the system are manifested in the type of phase trajectories. The sequence $s_1 = f(s_0), s_2 = f(f(s_0)) = f^2(s_0) \equiv f \circ f, ...$, i.e., $\{s_k\}_{k=0}^{\infty}$, is called the *forward orbit* (or forward trajectory), this is a time-ordered sequence of states. The equilibrium state of a dynamical system corresponds to a degenerate trajectory - a point in phase space, to periodic motion - a closed curve, to quasiperiodic motion that has base frequencies in the m spectrum âĂŤ a curve on the m-dimensional torus embedded in the phase space [2]. The steady state (steady motion) of a dissipative system corresponds to an *attractor* - a set of trajectories that attract all close trajectories to themselves [2].

1.3 Background information

1.3.1 Historical background

A dynamic approach to the description of systems of various origins has been known since the time of Newton. It is the basis for the analysis of most of the classical phenomena in physics and other natural sciences: first, the corresponding mathematical model is constructed in the form of dynamic equations, and then their solutions are studied in one way or another, which, in principle, can be compared with experimental data [29].

A. Poincaré and A. M. Lyapunov can be considered the founders of the theory of Dynamic Systems. In the late 19th - early 20th centuries, they discovered and investigated a class of problems (in celestial mechanics, in the theory of equilibrium figures of a rotating fluid, etc.), in which it was necessary to know the behavior of not one single solution $x(t)$ of the system of ordinary differential equations (ODE), but all (or very many) solutions corresponding to different initial states of a real (for example, physical) system. In this case, $x(t)$ can be represented as a curve in the space of all possible states (that is, the values of vectors x) and use the geometric properties of this curve to understand and describe the properties of the solution $x(t)$. Such a curve is called a phase trajectory [2].

Although the dynamic system is a kind of mathematical abstraction, this paradigm has proven to be a very productive tool in describing many real phenomena. The greatest success in this direction was obtained in the first third of the twentieth century, when the theory of oscillations of two-dimensional systems was created. Subsequent research efforts were devoted to exploring the possibility of extending this theory to multidimensional systems. However, despite significant discoveries in this area, until the 60s of the twentieth century it was not clear how complex the movements in such systems could be.

The situation radically changed after S. Smale [49] introduced the hyperbolic theory. Research in this direction revealed a wide variety of dynamics of nonlinear systems and led to one of the most important discoveries of the twentieth century -

dynamic chaos. The Y-systems [5] were introduced, bifurcations of separatrix loops, leading to complex behavior [44, 45] were described, famous Lorenz systems [28] were introduced and billiard models, which are simplified models of statistical physics [47, 48], were studied.

In the 2nd half of the 20th century. D. V. Anosov, V. I. Arnold, R. Bowen, R. Manet, Ya. G. Sinai, S. Smale, S. Hayashi, L. P. Shilnikov, and others developed and created a deep and coherent theory of Dynamic Systems, which gives the correct idea of the nature of deterministic processes and allows you to explore the models of real systems.

1.3.2 Chaos theory

Chaos theory is the interdisciplinary subject which describes dynamic systems' sensitivity to the initial conditions.

Currently, there are several possibilities to introduce the concept of chaos. The most common and frequently used definition was proposed in Devaney (1989, [12]). It relies on the property of the system's extreme (exponentially strong) sensitivity to setting initial conditions or to external influences. This seems to be quite natural, since the main manifestation of dynamic chaos is the exponential divergence of close trajectories.

However, to define the concept of chaos one exponential instability is not enough. In addition, the condition of transitivity and the presence of some regularity, called the density of periodic orbits (i.e. cycles), is necessary. Often, transitivity is replaced by the condition of topological mixing, which is stronger [29].

Let M be a metric space. According to Devaney (1989, [12]), a mapping $f : M \to M$ is called chaotic if the following statements hold: (a) f is unstable with respect to the specification of the initial conditions; (b) f is topologically transitive; (c) the periodic points of the mapping f are dense in the space M.

A mapping f is said to be unstable with respect to the initial conditions, if there is some quantity (instability constant) δ such that for some point $x \in M$ and ϵ there is a point $y \in M$ for which $dist(x, y) < \epsilon$ and $dist(f^n(x), f^n(y)) \geq \delta$ for some $n \in N$,

where $dist(\cdot)$ means distance. The geometric interpretation of these relationships is shown in Figure 1-1. It is noteworthy that the constant δ does not depend on x or on ϵ. It is determined only by the properties of the system under consideration.

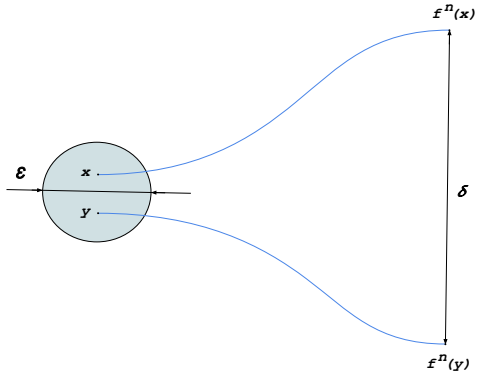


Figure 1-1: Geometric interpretation of instability on initial conditions.

Further, a mapping f is called transitive if, for any two open sets U, V, there is a number n such that $f^n(U) \cap V \neq \varnothing$. The informal sense of the property of transitivity is demonstrated by Fig 1-2. We note that it is known from the theory of metric spaces that transitivity is equivalent to the existence of a dense trajectory.

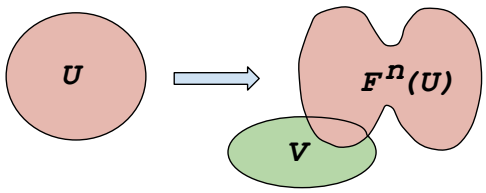


Figure 1-2: Explanation of transitivity.

Finally, the density property of periodic trajectories means that there exists at least one (and, therefore, infinitely many) periodic trajectories in any neighborhood of any point in M [29].

Thus, a chaotic system must have three important properties: unpredictability (exponential instability), indecomposability (transitivity), and a regularity element (density of orbits). However, Banks et al. [7] discovered that, in the above definition,

the condition of sensitive dependence on the initial conditions is redundant. Consequently, if the mapping is continuous, the transitivity property holds, and the orbits are dense, then the system has a sensitive dependence on the initial conditions. A little later, it was shown [6] that neither the transitivity nor the density of orbits follow from the remaining two conditions in the definition of chaos. Apparently, a transformation defined on a compact set can be defined as chaotic if it has a sensitive dependence on the initial conditions and has dense orbits.

More recently, a definition of the chaoticity of a dynamical system was proposed in Kolesov and Rozov [23], which, in addition to the sensitive dependence on the initial conditions, also includes the requirement of trajectory complexity. Here, by complexity, the authors understand in a certain sense the absence of recurrence. It was also shown that the definition of chaos, based on instability with respect to initial conditions, transitivity and density of orbits [12], implies chaos, proposed by the authors in [23].

There are other definitions of chaos. For example, Gulik [17] states that the chaos exists when either there is a substantial dependence on the initial conditions, or the function has a positive Lyapunov exponent at each point of its definition domain and therefore is not ultimately periodic. Thus, shows the importance of the Lyapunov exponent [30] to explore the chaoticity of the systems.

1.3.3 Strange attractor

The term appeared in the XX century and is used both to describe the behavior of nonlinear systems, and as a broad scientific metaphor. If a system falls into a neighborhood of some attractor (it can be a point or a whole region of space), then its various trajectories (variants) of behavior “attract" to it.

The use of the term “attractor" is easy to understand if we look at the pendulum example in a viscous medium. Suppose that the pendulum is in the lower position of a stable equilibrium (at a stable stationary point). If now to perturb him a little, then he will begin to make damped oscillations around this equilibrium position. In this sense, the equilibrium state of the pendulum would seem to be attracting, or

attractor [29]. In this case, obviously, the attractor will have zero measure. In a similar way, one can get an idea of attractors corresponding to periodic (limit cycles) and quasiperiodic (invariant tori) motions. The formalization of these ideas leads to the modern concept of an attractor. There are a significant number of different attractors, but mainly, we can distinguish three types:

1. “Point attractor". This can be a system of swinging pendulum, which, with time, stops the friction force at one point. Here, the system “attracts" to the initial equilibrium point.

2. “Limit cycle". If there is no friction, then the pendulum will fluctuate forever and become a regular periodic system.

3. “Strange attractor." If we randomly change the action affected to the pendulum at regular time intervals, the resulting motion will be different and non-periodic. However, it is limited by the maximum amplitude of the pendulum and the laws of physics. As a result, a movement will be a chaotic, and shows strange attractor.

A strange attractor is some “complexly arranged” phase space to which almost all trajectories from its certain neighborhood attract, and the motion on the set itself has an exponentially unstable character. Such a combination of global compression with local instability leads to the fact that the attractor can no longer be smooth as, for example, a torus; it is stratified in a certain way and is a Cantor set in some section [29]. A strange attractor may appear after several new bifurcations.

The concept of “strange attractor” and the possibility of its existence were first mentioned in [42], and the main idea in [42] was that such subsets of the phase space play a decisive role in solving the problem of turbulence. Although this approach was not fully justified as showed in [26], the work [42] provided the impetus to the development of the theory of chaotic dynamical systems and their applications.

Chaoticity is based on exponential instability. At the same time, in order for the system to have a chaotic behavior (i.e., to have a sensitive dependence on the

initial conditions), only an instability is necessary, which appears in the definition of hyperbolicity. But it is not at all necessary that this instability be the same for all trajectories. Moreover, for different trajectories the number of unstable directions may be different. This is, for example, the Lorenz attractor. Attractors of this type do not collapse under small perturbations, but their geometric structure, in general, may vary. Lorenz attractor or “butterfly effect" is the famous attractor in chaos theory, proposed by Lorenz in 1963 [28]. We also reproduced this attractor, and this is given in Figure 1-3.

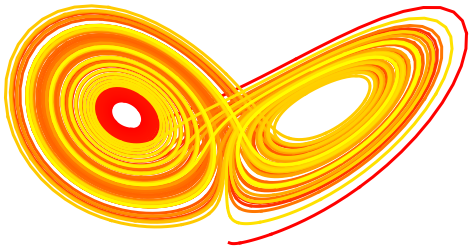


Figure 1-3: Lorenz Attractor

1.3.4 Bifurcation

The term bifurcation is derived from the Latin bifurcus - “forked", “separation" and is used in a broad sense to denote all sorts of qualitative rearrangements of various objects when the parameters on which they depend change [37]. In some cases, if the parameter of the system changes, then the behavior of the system might change smoothly. However, in some cases, when a parameter passes through a certain critical value, the dynamics of the system may change dramatically. The values of the

parameters at which the restructuring of the steady-state modes of motion in the system occurs are called the *bifurcation parameter values* (or the bifurcation point), and this change itself is called the *bifurcation*. As a result of a sequence of bifurcations in a dynamic evolving system, the establishment of a chaotic regime is possible. The bifurcation cascade is one of the typical scenarios of transition from order to chaos. The model of complex system development through the sequence of bifurcations and the idea of chaos is applicable to the phenomena of the most diverse nature: physical, biological, social, economic, i.e. to any systems where there is a sequence of period doubling bifurcations [37]. The establishment of chaotic behavior in a dynamic system as a result of a sequence of bifurcations is usually called a scenario of chaos [29].

The bifurcation point is one of the most significant concepts of the theory of self-organization, dynamics. This is such a period or moment in the history of a system when it turns from one systemic definition to another. Its qualitative characteristics after reaching the bifurcation point are doomed to a fundamental change, leading to a change in the essence of the system itself. The system transformation mechanism that operates at such times is associated with the branching of the system trajectory, which is determined by the presence of competition of attractors [37].

1.4 Recurrent Neural Networks as dynamical systems

Recall that the Recurrent Neural Network models can be considered as a mapping from $\mathbb{R}^d \to \mathbb{R}^d$. Thus, the mapping for Recurrent Neural Networks will be: $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^d$. The general form of the recurrent neural network is given in the following form:

$$u_t = \Phi(u_{t-1}, W_1 x_t, W_2 x_t, ..., W_k x_t) \tag{1.1}$$

where x_t is the t-th input data, and W is the weight matrices. Depending on the structure of Recurrent Neural Networks, there could be several weight matrices. Φ is an evolution rule that transforms the state u_t and the input data at each time point

t. This function Φ can be a sigmoid function, a tanh function or it can have several activation functions depending on the architecture of RNN. t is the current time and u_t is the current state.

For a given mapping, we have the initial state u_0, the initial time t_0 and, according to Laurent and Brecht [27], the mapping Φ: $u_{t+1} = \Phi(u_t)$, $t > t_0$, $u_{t_0} = u_0$ $t = t_0$, defines a *discrete-time dynamical system*, and this is a simple repeated iteration of the mapping Φ.

The set of all visited states $\mathcal{O}^+(u_0) := \{u_{t_0}, u_{t_0+1}, ..., u_{t_0+n}, ...\}$ forms a *forward trajectory* (can also be called a *forward orbit*) through u_0 [27]. This forward trajectory can be used to build an *attractor* for a dynamical system. If the dynamical system is chaotic, then the attractors will have the form of fractal sets, and this is called as *strange attractors*.

According to Laurent and Brecht [27], to gain inside information about the underlying structure of Recurrent Neural Networks, it has been proven that it is better to evaluate dynamical systems, forward orbits when there is no input data is provided from outside. Thus, we don't consider any effects from outside, and dynamical system given in Equation 1.1 will become:

$$u_t = \Phi(u_{t-1}),\ \Phi(u) := \Phi(u, 0, 0, ..., 0). \tag{1.2}$$

If we compare the dynamical systems in 1.1 and 1.2, we can say that the time-invariant system in 1.2 is much more tractable than in 1.1 [27], and it allows to explore inside the architecture of Recurrent Neural Networks. We can play around with the parameters, without any external inputs. By restricting external input data, we can separate external influences from influencing into the model. This 1.2 is called as a *dynamical systems induced* by RNN [27].

Chapter 2

Chaoticity of RNNs

2.1 Vanilla RNN architecture

Recurrent Neural Networks (RNN) are widely used technique to model the sequence-to-sequence models. Simple feed-forward neural networks are unable to memorize sequences by virtue of their architecture, in contrast to the Recurrent Neural Networks. RNN can also be called feed forward networks, but neurons in such a network transmit values not only to the neuron in the next layer, but also to themselves in the next stage. That is, when the data first arrives at the neuron, it processes them in accordance with the activation function, sends it to the next layer and stores some of this information. When data is received for the second time, the neuron, along with this data, also receives the value that it saved at the previous iteration, as input.

A neural network is called recurrent if it contains inverse connections between neurons, i.e. the output of the neuron is transmitted to the input of another neuron located in a layer with a smaller index. The presence of feedback provides information not only from the previous layer, but also from previous training passes, which guarantees the preservation of temporal information. This means that the result will also depend on the sequence in which the training data was transmitted. This allows to model sequences.

The main idea of RNN is to generate a fixed dimension vector from the input sequence of characters using recursion. Suppose that at step t we have the vector

h_{t-1}, which is the history of all the previous input data. RNN will calculate the new vector h_t (that is, its hidden state), which combines all the previous input data $(x_1, x_2, ..., x_{t-1})$, as well as the new symbol x_t with:

$$h_t = \phi_\theta(x_t, h_{t-1})$$

where ϕ_θ is a function parametrized by θ, which takes as input a new input data x_t and a history h_{t-1} to the t-th word. Initially, we can assume that h_0 is a zero vector. The recurrent activation function ϕ is usually implemented as a simple affine transformation, followed by an element-wise nonlinear function

$$h_t = \tanh(Ux_t + Wh_{t-1} + b). \tag{2.1}$$

In this equation, the following parameters are present: input weight matrix U, recurrent weight matrix W and bias vector b. It should be noted that this is not the only option. There are many ways to develop new recurrent activation functions (ReLU, sigmoid, several mixed functions, etc.), usually they are non-linear functions.

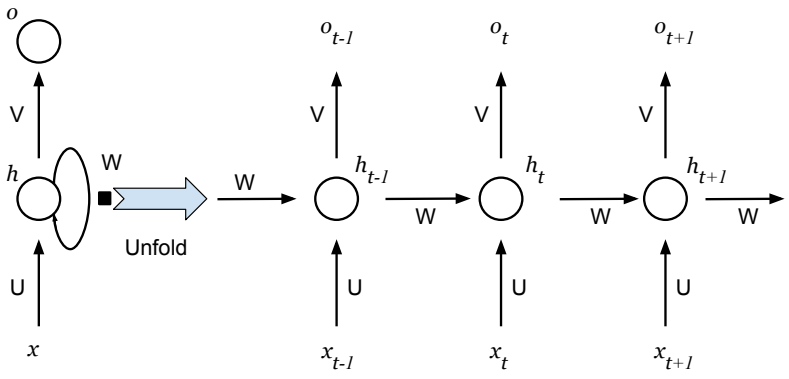


Figure 2-1: Recurrent Neural Network architecture and its unfolding

Figure 2-1 shows the unwrapped internal structure of Recurrent Neural Networks. From here, we can see that the input data x unwrapped into parts, i.e. if our input

data x is a text sentence, then each x_t corresponds to each words. So, each layer corresponds to each word. The network contains three layers, an input layer, a hidden layer and an output layer.

Equation 2.1 shows the calculations of RNN. In the input layer, x_t is an input word, usually each word is modelled using one-hot encoding vectors, or it can take an embedded word vector.

The hidden layer h_t is calculated by using the previous hidden state information h_{t-1} and the new input x_t. For example, suppose we want to calculate the hidden state value of the third word: h_3, using Equation 2.1, $h_3 = \tanh(Ux_3 + Wh_2 + b)$. Here h_2 contains information about previous words sequences, and is calculated as $h_2 = \tanh(Ux_2 + Wh_1 + b)$ and so on. From here we can notice the role of the hidden state values h_t, it contains information about previous word sequences, calculated recursively. For this reason, the hidden state value h_t is refers as the "memory" of the RNN architecture. It is assumed that RNNs can use unlimited history information, and it has current connections on hidden states, so that history information can circulate within a network for an arbitrarily long time.

However, in practice, h_t might not contain very long sequence information, there are problems with the gradient vanishing during training. A number of papers have shown that it is possible to increase the efficiency of learning RNN using more complex activation functions, for example, using neurons with a long short-term memory unit (LSTM), gated recurrent unit (GRU), recurrent highway networks (RHN) and etc.

The output at time t is calculated by $o_t = softmax(Vh_t)$. It takes the value of the hidden state (history) and, multiplying the hidden state value into the matrix V, passes through a non-linear function. Depending on the task (classification or sequence prediction), this function might be softmax or sigmoid. In addition, again, depending on the task (many-to-many, one-to-many, many-to-one), we may not need to predict the output at each time t, in some cases we may need calculate the output only at the end.

This form of Recurrent Neural Networks first proposed by Elman [13] and Jordan [21] in 1990 and can be considered as a simple or "vanilla" Recurrent Neural Network.

2.2 Vanilla RNN in 1D case

In this section we consider the dynamics of the Vanilla Recurrent Neural Network. Before analyzing the complex neural network architectures of Recurrent Highway Network (RHN) and Neural Architecture Search (NAS), we started by analyzing the simple Recurrent Neural Network (RNN), proposed by Elman in 1990 [13], for chaoticity. So, for the Simple RNN architecture we want to discuss the nonlinear map $h_{t+1} = \tanh(Wh_t + Ux_{t+1} + b)$, where W and U are the weight matrices. We assume that there is no input data is provided, and the bias term is zero, so our system will become: $h_{t+1} = \tanh(Wh_t)$.

In this subsection, we consider the simple RNN architecture in 1D case, we assume that our values $h, W \in \mathbb{R}$, i.e. our state and weight are scalars.

Claim 1. *A dynamical system induced by Simple RNN:*

$$h_{t+1} = \tanh(Wh_t),\ \ h_t, W \in \mathbb{R} \tag{2.2}$$

is non-chaotic when $W \in (1, 1)$.

2.2.1 Fixed point and Bifurcation Analysis

First, we would like to analyze the map given in Equation 2.2 for extremum points.

$f'(h) = \frac{\partial}{\partial h} \tanh(Wh)$

$\frac{\partial}{\partial h} \tanh(h) = \frac{(e^h+e^{-h})(e^h+e^{-h})-(e^h-e^{-h})(e^h-e^{-h})}{(e^h+e^{-h})^2} = 1 - \frac{(e^h-e^{-h})^2}{(e^h+e^{-h})^2} = 1 - \tanh^2(h) = \operatorname{sech}^2 h$.

If we also consider the weight parameter W:

$$\frac{\partial}{\partial h}(\tanh(Wh)) = W \operatorname{sech}^2(Wh) = W(1 - \tanh^2(Wh)) \tag{2.3}$$

If we take $f'(h) = 0 \Rightarrow 1 - \tanh^2(h) = 0 \Rightarrow (\tanh(h) - 1)(\tanh(h) + 1) = 0$. No solutions exists. So there is no maximum or minimum points.

For $f'(h) = W(1 - \tanh^2(Wh)) = 0$, only solution exists when $W = 0$.

Finding the the fixed points and the periodic points of maps and then study the region of their stability is important in bifurcation theory [25]. The fixed points can be calculated by solving the equation $f(x) = x$. For our case, $h = \tanh(Wh)$, for $W \leq 1$ there is only one solution: $h = 0$, for other values, there are 3 solutions.

The implicit plot of $h = \tanh(Wh)$ is given in Figure 2-2. Plots of h and $\tanh(Wh)$ for $W = 1, 2$ are provided in Figure 2-3 and 2-4.

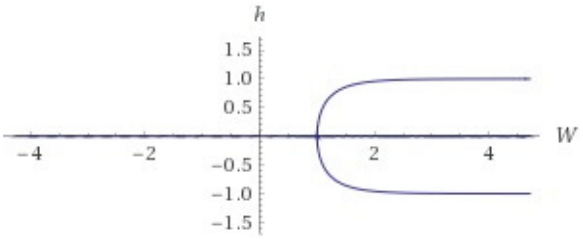


Figure 2-2: Implicit plot of $h = \tanh(Wh)$.

In order to analyze the stability of the fixed points, we use the stability criterion: if $|f'(x)|_{x=x^*} < 1$ then the fixed point $x = x^*$ is stable, otherwise it is unstable [25].

In our case, we have fixed points $h = 0$, $h_1(W)$ and $h_2(W)$. For the first fixed point $h = 0$: $|f'(h)|_{h=0} = W(1 - \tanh^2(Wh)) = W(1 - \tanh^2(0)) = W(1 - 0) = W$. So, by using the above notation, fixed point $h = 0$ is stable when $|W| < 1$, otherwise it is unstable. Hence $h = 0$ remains as a stable fixed point when $-1 < W < 1$.

When W passes through the value 1, the stable fixed point $h = 0$ becomes an unstable one and this shows a qualitative change in the behavior of the fixed point at

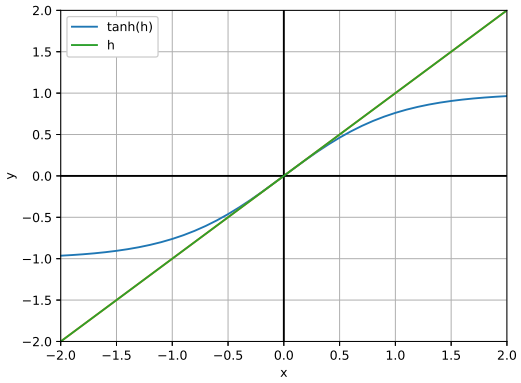


Figure 2-3: One solution of $h = \tanh(h)$.

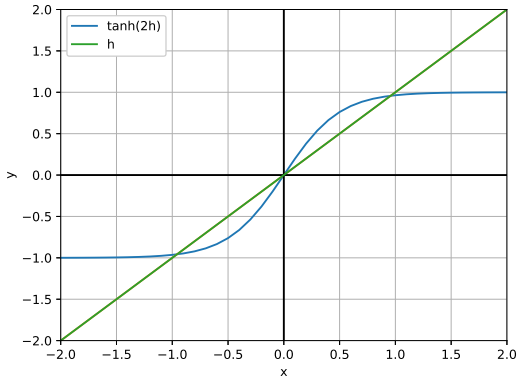


Figure 2-4: Three solutions of $h = \tanh(2h)$.

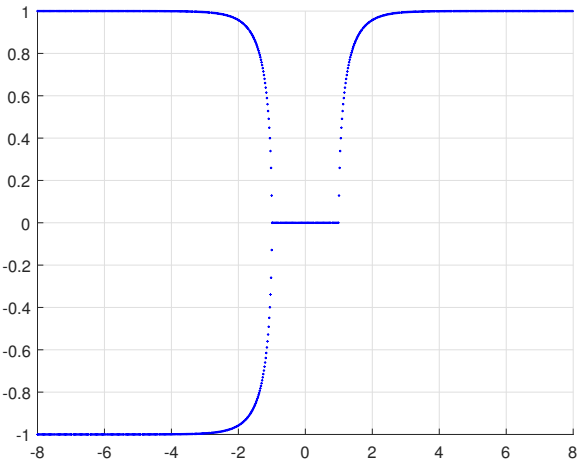


Figure 2-5: Bifurcation diagram of the $1D$ RNN $h_{n+1} = \tanh(Wh_n)$.

$W = 1$. So we consider $W = 1$ as the first bifurcation point. Also at $W = -1$, there also occurs a bifurcation.

To analyze the fixed points $h_1(W)$ and $h_2(W)$, we have to solve the inequality: $|W(1 - \tanh^2(Wh))| < 1$, where $h_1 \in (0, 1)$ and $h_2 \in (-1, 0)$. If this inequality holds, then the fixed points are stable. The solutions of this inequality will be $|W| \operatorname{sech}^2(Wh) < 1$. Most of the values of h_1 and h_2 are close to 1 and -1. If we put these values of h into the inequality $|W| \operatorname{sech}^2(Wh) < 1$, then this inequality holds for all W, therefore we do not consider these fixed points.

The bifurcation diagram of the $1D$ RNN is given in Figure 2-5. We want to study the long term behavior of the map, so, after understanding the behavior of the fixed

points of $f = \tanh(Wh)$, we now consider the periodic points of period 2 and higher and analyze their stability property. The period 2 points are fixed points of the second order iteration of the map. So, let us consider the iterated map $f^2(h)$.

If we draw the graph of $f^2(h) = \tanh(\tanh(h))$ for $W = 1$ with the line $x = y$, there will be only one point of intersection, which is $h = 0$, which is already our first order fixed point of f. This graph is shown in Figure 2-6.

The fixed points of f are also fixed points of f^2 as $f(h) = h \Rightarrow f(f(h)) = f(h) \Rightarrow f^2(h) = h$. The period 2 points of the map are given by the solution of the equation:

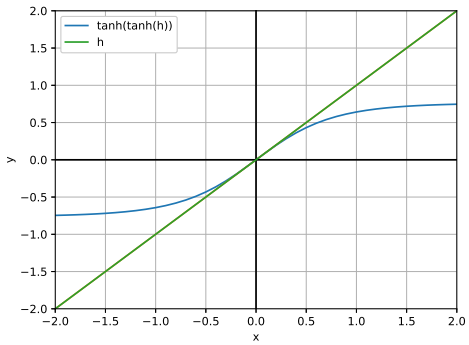


Figure 2-6: Solutions of $h = \tanh(\tanh(h))$.

$f^2(h) = f(f(h)) = \tanh(W(\tanh(Wh))) = h$. We find the solutions $h \approx -1$, $h = 0$ and $h \approx 1$ for other values of W (except $W = 1$).

Let us see how the derivatives of the second iterate function change at the bifurcation value.

$$\frac{\partial}{\partial h}(\tanh(W\tanh(Wh))) = W^2\operatorname{sech}^2(Wh)\operatorname{sech}^2(W\tanh(Wh)).$$

Now let's analyze the bifurcation points:

$|f'(h)| = W^2\operatorname{sech}^2(Wh)\operatorname{sech}^2(W\tanh(Wh)) \Rightarrow$

$|f'(h)|_{h=1} = W^2\operatorname{sech}^2(W)\operatorname{sech}^2(W\tanh(W))$.

The value of the above equation is always less than the absolute values of $|1|$ for any values of W. So all points of W are stable on the second order periods.

So for all values of W, $W^2\operatorname{sech}^2(W)\operatorname{sech}^2(W\tanh(W))$ will be between -1 and 1.

Also for the second fixed point, we have

$$|f'(h)|_{h=-1} = W^2 \operatorname{sech}^2(-W) \operatorname{sech}^2(W \tanh(-W)).$$

Here also $-1 < W^2 \operatorname{sech}^2(-W) \operatorname{sech}^2(W \tanh(-W)) < 1$ is for any values of W. This means that these two fixed points of f^2 are stable fixed points for all values of W and they will not become unstable. Periodic points of period 2 will not occur.

2.2.2 Lyapunov Exponent

There are several ways to diagnose whether the system under investigation is in a chaotic state or not. As Moon [36] states, "chaos in deterministic systems implies a sensitive dependence on initial conditions". This means that two trajectories that are close to each other in the phase space at some initial moment of time diverge exponentially in a short average time [36]. If d_0 is a measure of the initial distance between two reference points, then after a short time t, the distance between the trajectories leaving these points becomes equal to:

$$d(t) = d_0 2^{\lambda t} \tag{2.4}$$

Here, λ is the Lyapunov Exponent.

Wolf et al. [57] give an excellent review on Lyapunov exponent [30] and their use to identify chaoticity of the systems. Also they suggest useful calculations how to measure the Lyapunov exponent.

As Moon staes [36], the exponential divergence of chaotic trajectories can only be local, since if the system is limited, then $d(t)$ cannot grow to infinity. For this reason, in order to determine the measure of the divergence of the trajectories, it is necessary to average the exponential growth over many points along the trajectory [36]. The calculation of the Lyapunov exponent begins in the choice of the reference trajectory [57], points on the adjacent trajectory and measurement of $\frac{d(t)}{d_0}$. When the distance $d(t)$ becomes too large (i.e. its growth deviates from the exponential behavior), the experimenter finds a new "neighboring" trajectory and determines the new initial distance $d_0(t)$ [36]. So, by using the above notion, the Lyaponov exponent

can be calculated as:

$$\lambda = \frac{1}{t_n - t_0} \sum_{i=1}^{n} \ln \frac{d(t_i)}{d_0(t_{i-1})} \tag{2.5}$$

In practice, the calculation of the Lyapunov exponent using Equation 2.5 is difficult. To make the calculation easier, there used “close" enough numerical estimations of the Lyapunov exponent. Consider given one-dimensional mappings: $x_{n+1} = f(x_n)$, where the function f is smooth and differentiable, the distance between adjacent trajectories is measured by the quantity $|\partial f/\partial x|$. To verify this, we introduce two initial conditions: x and $x + \delta$. Then in the relation 2.4:

$$d_0 = \delta \tag{2.6}$$

$$d_1 = f(x_0 + \delta) - f(x_0) = \left.\frac{\partial f}{\partial x}\right|_{x_0} \delta \tag{2.7}$$

Following Equation 2.5, determine the Lyapunov exponent as:

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln\Big|f'(x_i)\Big|. \tag{2.8}$$

The criterion of chaos in terms of the Lyapunov exponent, if λ is positive, then there is a chaotic behavior, as neighboring trajectories separate from each other at large n [46]. If λ is negative, then there is a non-chaotic behavior, as trajectories converge to a fixed point or a limit cycle. We have calculated the Lyapunov exponent for some values of W in case of the Simple RNN map.

In our case, $f(h) = \tanh(Wh)$ and $f'(h) = W(1 - \tanh^2(Wh))$. For Equation 2.8, we have considered iteration size of 100000 (instead of ∞) to get the values of Lyapunov Exponent. We used Python scripts to calculate the values. Full results with the numerical values are given in A.1.

In Figure 2-7 we draw the values of Lyapunov Exponent versus the weight value W. From this Figure we can see that the values of -1.1 and 1.1 of W the Lyapunov Exponents become positive, showing the beginning of a chaotic region. Also this Figure 2-7 further supports the first two bifurcation points as -1.0 and 1.0 where

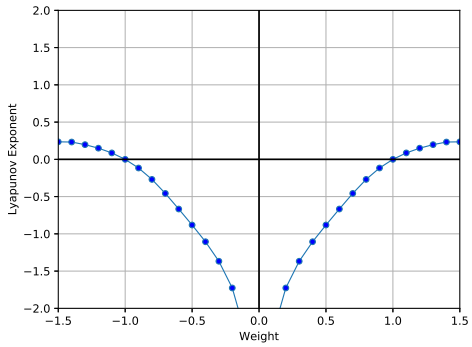


Figure 2-7: Lyapunov coefficient versus W value.

the Lyapunov Exponent is almost zero. Moreover, after attaining the chaotic region at $W = 1.0$ and $W = -1.0$, we see the negative Lyapunov exponent values. They signify that within the chaotic region also, at certain values of the parameter, there are regular behaviors. This is supported also by the bifurcation diagram which we have drawn in Figure 2-5 in the previous section. (After some values of W, it will stabilize).

2.3 Multidimensional case for Vanilla RNN

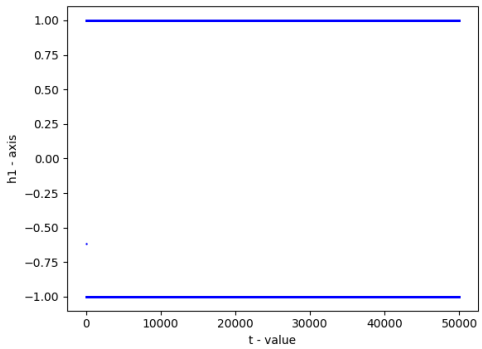
Now, consider the high dimension cases. For vanilla RNN:

$$h_{(t+1)} = \tanh(Wh_t), \ h_t \in R^d \tag{2.9}$$

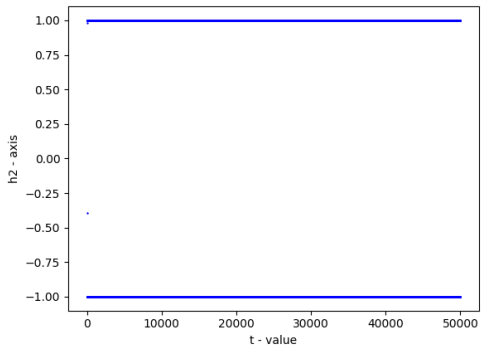
Claim 2. *If $||W|| < 1$, then for (2.9) we have the following statement: for any h_0 we have $lim_{(n\to\infty)} h_t = 0$.*

Proof. $||h_{t+1}|| = ||\tanh(Wh_t)|| \leq ||Wh_t|| \leq ||W||||h_t||$. Therefore, $||h_t|| \leq ||W||^t||h_0|| \to 0$, $t \to \infty$. □

Claim 3. *There exists W with $||W|| > 1$, such that induced dynamical system (2.9) is chaotic (means that there should be at least 1 nontrivial attractor, i.e. attractor which is not a point).*



(a) $h^{(1)}$ vs t



(b) $h^{(2)}$ vs t

Figure 2-8: State vs. time graphs for $2D$ case when the norm is larger than one

When do we have these Claims 2 and 3? Let $||h||$ be a norm on $\mathbb{R}^d$ such that $||\tanh(h)|| \leq ||h||$. Examples of such norm are:

a) $||h||_p = (\sum_{i=1}^{d} h_i^p)^{1/p}$ is a l_p norm. For any such norm let us define corresponding matrix norm as $||W||_p = \max_{h:||h||_p=1} ||Wh||_p$.

Now, we tested the weight matrix, the norm of which is greater than 1. Lets consider the weight matrix $W = \begin{bmatrix} -1 & -4 \\ -3 & -2 \end{bmatrix}$. If we plot the graph of $h^{(1)}$ vs. t and $h^{(2)}$ vs. t, then we get the graphs shown in Figure 2-8.

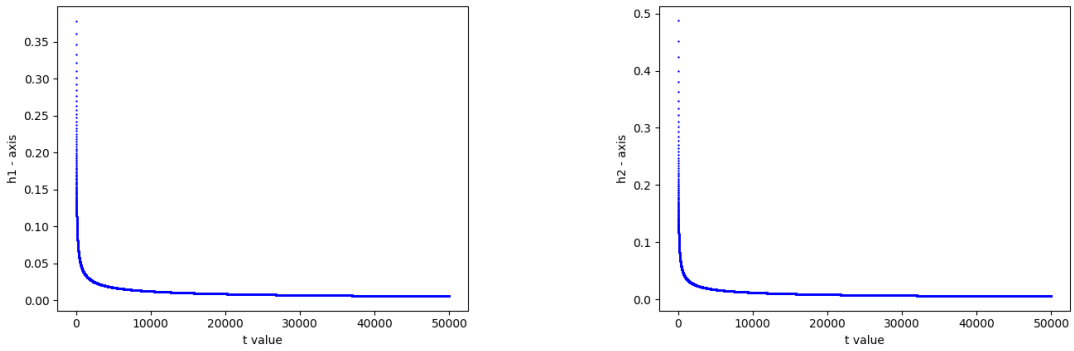
Also for the weight matrix: $W = \begin{bmatrix} 4 & 1 \\ -9 & -7 \end{bmatrix}$ the last three values of h^1 and h^2 will be: [..., 0.99512, 0.99512, 0.99512] and [..., -0.97297, -0.97297, -0.97297].

For the weight matrix: $W = \begin{bmatrix} -2 & 6 \\ 0 & -6 \end{bmatrix}$ the last three values of h^1 and h^2 will be: [..., -0.999990, 0.99999, -0.99999] and [..., 0.99998, -0.99998, 0.99998].

For the weight matrix: $W = \begin{bmatrix} -1 & -6 \\ 6 & -9 \end{bmatrix}$ the last three values of h^1 and h^2 will be: [..., -0.99999, 0.99999, -0.99999] and [..., -0.99454, 0.99454, -0.99454].

In the above example, all norms are larger than one (Frobenius norm, nuclear norm (trace norm), max norm, l_1 norm, l_2 norm).

What happens when $||W|| = 1$? If $W = I$, then we have: $h_{t+1} = \tanh(h_t)$. Then we can get the following graphs: the graph of h^1 vs t and the graph of h^2 vs t is given



(a) $h^{(1)}$ vs t (b) $h^{(2)}$ vs t

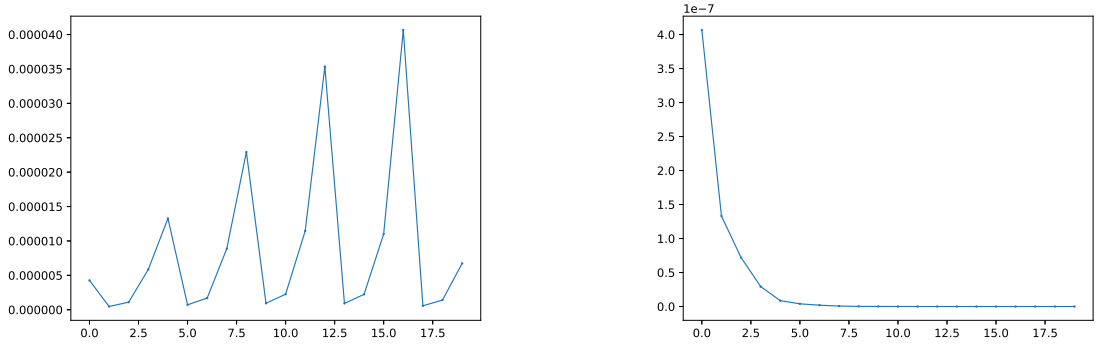
Figure 2-9: t vs. h for vanilla RNN

in Figure 2-9. In both cases h_{inf} eventually goes to zero.

Also if we take the weight matrices whose norm is smaller than 1, then, both h^1 and h^2 goes to the zero for $t \to \infty$ and we will get the same picture as in Figure 2-9. We will have non-chaotic behavior in vanilla RNN.

Now we tested the chaoticity using the Lyapunov Exponent as described in Section 2.2.2. We choose two points that are initially very close to each other. Then we iterate these two points through our map and calculate the Euclidean distance between them. We initialize the weight matrix $W_h = \begin{bmatrix} -7 & -6 \\ 6 & -4 \end{bmatrix}$, whose norm is greater than 1. Then, if we draw the graph of the Euclidean distance, we can get a graph as shown in Figure 2-10a.

Next, we repeat the above experiment, but in this case with the weight matrix $W_h = \begin{bmatrix} -0.1237 & -0.3446 \\ 0.3282 & -0.3723 \end{bmatrix}$, whose norm is smaller than one. Then, if we draw the graph of the Euclidean distance, we can get the graph shown in Figure 2-10b. From here, again, we can see that if we initialize the weight matrix with a matrix whose norm is greater than one, then after some iteration we might notice a divergence. On the other hand, initialization with a matrix whose norm is smaller than one will give a zero difference immediately after the first iterations.



(a) $h^{(1)}$ vs t

(b) $h^{(2)}$ vs t

Figure 2-10: Euclidean distance from the 2 close points for vanilla RNN.

Chapter 3

Chaotic behavior of RHN

3.1 Recurrent Highway Networks

After exploring the vanilla RNN, we considered the dynamics of the state-of-the-art RNN architectures. Recurrent Highway Network (RHN) proposed by [59] introduced a new theoretical analysis based on the Geršgorins circle theorem [15]. This theorem helps to clarify many optimization issues and modeling. Their approach allows transition depths to be larger than one and it improves the LSTM cell. Depending on the depth of the transition, on the Penn Treebank corpus, RHN improves word-level perplexity from 90.6 to 65.4, using the same number of parameters. Also RHN architecture outperform all experiments on larger datasets.

Recurrent Neural Networks can be considered as a powerful tool at representing certain function classes and have credit assignment paths and so deep in time. However, some aspects of RNN do not take an advantage from the depth, mostly because of the vanishing gradient problems. Srivastava et al., (2015 [51]) proposed *Highway Layers* based on the LSTM cell, enabling the training of networks with even hundreds of layers. Also, in this paper, authors made mathematical analysis on the strength sides of the LSTM network.

The main idea behind increasing the depth of the step-to-step recurrent state transition is to allow the RNN tick for several time steps per step of the sequence ([52, 16]). By using this technique we can adapt the recurrence depth to the problem.

3.1.1 Revisiting Gradient Flow in Recurrent Networks

Zilly et al. [59] did revising on the Gradient Flow problem. They showed gradient vanishing and exploding problems with Jacobian matrix, namely the problem in terms of largest singular values. Here, the spectral radius sheds light on boundary conditions for vanishing and exploding gradients yet does not illuminate how the eigenvalues are distributed overall. Then, Zilly et al. applied the Geršgorin circle theorem to provide further insight into this problem.

Geršgorin circle theorem (GCT) (Geršgorin, 1931, [15]): For any square matrix $A \in \mathbb{R}^{n \times n}$,

$$\text{spec}(\mathbf{A}) \subset \bigcup_{i \in \{1,\ldots,n\}} \left\{ \lambda \in \mathbb{C} | \left\| \lambda - a_{ii} \right\|_{\mathbb{C}} \leq \sum_{j=1, j \neq i}^{n} |a_{ij}| \right\}, \tag{3.1}$$

i.e., the eigenvalues of matrix $\mathbf{A}$, comprising the spectrum of $\mathbf{A}$, are located within the union of the complex circles centered around the diagonal values a_{ii} of $\mathbf{A}$ with radius $\sum_{j=1, j \neq i}^{n} |a_{ij}|$ equal to the sum of the absolute values of the non-diagonal entries in each row of $\mathbf{A}$. In Figure 3-1 two Geršgorin circles are centered around their diagonal entries a_{ii}. The corresponding eigenvalues lie within the radius of the sum of absolute values of non-diagonal entries a_{ij}. Circle (1) represents an exemplar Geršgorin circle for an RNN initialized with small random values. Circle (2) represents the same for an RNN with identity initialization of the diagonal entries of the recurrent matrix and small random values otherwise. The dashed circle denotes the unit circle of radius 1. Using GCT we can understand the relationship between the entries of $\mathbf{R}$ and the possible locations of the eigenvalues of the Jacobian.

3.1.2 Recurrent Highway Networks (RHN)

The Highway layer computation is defined as

$$y = h \cdot t + x \cdot c$$

where "$\cdot$" denotes element-wise multiplication.

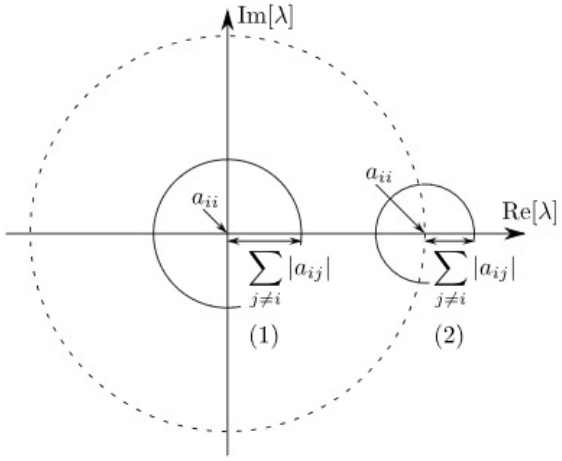


Figure 3-1: Illustration of the Geršgorin circle theorem.

Zilly et al. [59] proposed to construct a Recurrent Highway Network (RHN) layer with one or multiple Highway layers in the recurrent state transition. An RHN layer with a recurrence depth of L is described by

$$s_l^{[t]} = h_l^{[t]} \cdot t_l^{[t]} + s_{l-1}^{[t]} \cdot c_l^{[t]}$$

where

$$\mathbf{h}_l^{[t]} = \tanh(\mathbf{W}_H x^{[t]} \mathbb{I}_{\{l=1\}} + \mathbf{R}_{H_l} s_{l-1}^{[t]} + b_{H_l}), \quad (7)$$

$$\mathbf{t}_l^{[t]} = \sigma(\mathbf{W}_T x^{[t]} \mathbb{I}_{\{l=1\}} + \mathbf{R}_{T_l} s_{l-1}^{[t]} + b_{T_l}), \quad (8)$$

$$\mathbf{c}_l^{[t]} = \sigma(\mathbf{W}_C x^{[t]} \mathbb{I}_{\{l=1\}} + \mathbf{R}_{C_l} s_{l-1}^{[t]} + b_{C_l}), \quad (9)$$

and $\mathbb{I}_{\{\}}$ is the indicator function.

A schematic illustration of the RHN computation graph is shown in Figure 3-2.

It is important to note that an RHN layer with $L = 1$ is essentially a basic variant of an LSTM layer [59].

Zilly et al. did an analysis on RHN layers similar to standard RNNs based on GCT. Omitting the inputs and biases, the temporal Jacobian $A = \frac{\partial y^{[t]}}{\partial y^{[t-1]}}$ for an RHN layer with recurrence depth of 1 (such that $y^{[t]} = h^{[t]} \cdot t[t] + y[t-1] \cdot c[t]$) is given by

$$\mathbf{A} = \text{diag}(\mathbf{c}^{[t]}) + \mathbf{H}' \text{diag}(\mathbf{t}^{[t]}) + \mathbf{C}' \text{diag}(\mathbf{y}^{[t-1]}) + \mathbf{T}' \text{diag}(\mathbf{h}^{[t]}), \quad (3.2)$$

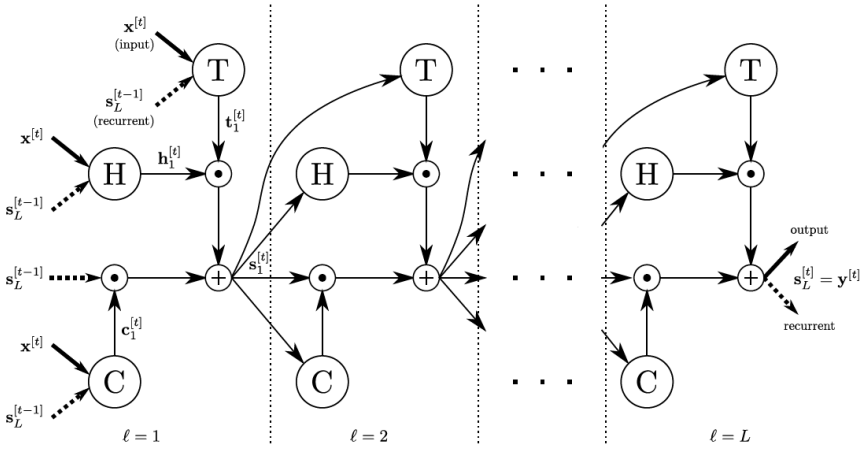


Figure 3-2: RHN layer inside the recurrent loop

where

$$\begin{aligned}\mathbf{H}^{'} &= \mathbf{R}_H^T \text{diag}[tanh'(\mathbf{R}_H \mathbf{y}^{[t-1]})],\\ \mathbf{T}^{'} &= \mathbf{R}_T^T \text{diag}[\sigma'(\mathbf{R}_T \mathbf{y}^{[t-1]})],\\ \mathbf{C}^{'} &= \mathbf{R}_C^T \text{diag}[\sigma'(\mathbf{R}_C \mathbf{y}^{[t-1]})],\end{aligned} \tag{3.3}$$

and has a spectrum of:

$$\begin{aligned}\text{spec(A)} \subset \bigcup_{i\in\{1,\dots,n\}} \Bigg\{ \lambda \in \mathbb{C} \Big| \Big\| \lambda - c_i^{[t]} - \mathbf{H}_{ii}^{'} \mathbf{t}_i^{[t]} - \mathbf{C}_{ii}^{'} \mathbf{y}_i^{[t-1]} - \mathbf{T}_{ii}^{'} \mathbf{h}_i^{[t]} \Big\|_{\mathbb{C}} \leq \\ \sum_{j=1, j\neq i}^{n} \Big| \mathbf{H}_{ij}^{'} \mathbf{t}_i^{[t]} + \mathbf{C}_{ij}^{'} \mathbf{y}_i^{[t-1]} + T_{ij}^{'} \mathbf{h}_i^{[t]} \Big| \Bigg\}.\end{aligned} \tag{3.4}$$

Last equation shows the influence of the gates on the eigenvalues of $\mathbf{A}$. Compared to the standard RNN, we can see that an RHN layer has more flexibility in adjusting the centers and radii of the Geršgorin circles.

3.1.3 Experiments

During the experiments, C values replaced by $C(\cdot) = 1_n - T(\cdot)$, similar to the suggestion for Highway Networks. Like all RNNs, in RHN also regularization can be essential for obtaining good generalization. Zilly et al. used the regularization technique proposed by Gal (2016 [14]), which is an interpretation of dropout based on

Model	Size	Best Val.	Test
RNN-LDA + KN-5 + cache (Mikolov & Zweig, 2012 [35])	9 M	-	92.0
Conv.+Highway+LSTM+dropout (Kim et al., 2015 [22])	19 M	-	78.9
LSTM+dropout (Zaremba et al., 2014 [58])	66 M	82.2	78.4
Variational LSTM (Gal, 2016 [14])	66 M	77.3	75.0
Variational LSTM + WT (Press & Wolf, 2017 [41])	51 M	75.8	73.2
Pointer Sentinel-LSTM (Merity et al., 2016 [32])	21 M	72.4	70.9
Variational LSTM + WT + augmented loss (Inan et al., 2016 [20])	51 M	71.1	68.5
Variational RHN	**32 M**	**71.2**	**68.5**
Neural Architecture Search with base 8 (Zoph & Le, 2016 [60])	32 M	-	67.9
Variational RHN + WT	**23 M**	**67.9**	**65.4**
Neural Architecture Search with base 8 + WT (Zoph & Le, 2016 [60])	25 M	-	64.0
Neural Architecture Search with base 8 + WT (Zoph & Le, 2016 [60])	54 M	-	62.4

Table 3.1: Validation and test set perplexity of recent state of the art word-level language models on the Penn Treebank dataset [59]

.

approximate variational inference.

For Penn Treebank dataset RHN with depths from 1 to 10 and with fixed total parameters (32M) were applied. Also reported the results for each model trained with WT regularization. For the best 10 layer model, reducing the weight decay further improves the results to 67.9/65.4 validation/test perplexity. As the recurrence depth increases from 1 to 10 layers the “width" of the network decreases from 1275 to 830 units since the number of parameters was kept fixed.

In Table 3.1 RHN results were compared with the best published results on this dataset. RHNs outperform most single models as well as all previous ensembles, and also benefit from WT regularization similar to LSTMs. However, the architecture which found by reinforcement learning and hyperparameter search (Zoph & Le, 2016 [60]) gives the best results. We also reproduced the results of Zilly et al. in Tensorflow. We got the perplexity 68.355 on the validation set and the perplexity 65.506 on the test set for Penn Tree Bank datasets. These results are similar to the results in the paper (it was 67.9 and 65.4).

3.2 Dynamics of RHN

RHN architecture is given in the following form: the Highway layer computation is defined as:

$$s_{n+1} = t \odot (h - s_n) + s_n \tag{3.5}$$

where

$$t := \sigma(W_t x + R_t s + b_t);$$

$$h := \tanh(W_h x + R_h s + b_h);$$

$\odot$ denotes Hadamard product.

3.2.1 RHN chaoticity in 1D

In this subsection we analyze the dynamics of the Recurrent Highway Network (RHN) in 1D case. First, we can start with the analysis of fixed points and check the region of their stability.

If we assume that no input is provided, then the induced form of the RHN will become:

$$t := \sigma(R_t s); \tag{3.6}$$

$$h := \tanh(R_h s); \tag{3.7}$$

$$s_{n+1} = t \odot (h - s_n) + s_n. \tag{3.8}$$

If we put everything together, we will get the following equation:

$$s_{n+1} = \sigma(R_t s_n) \odot (\tanh(R_h s_n) - s_n) + s_n. \tag{3.9}$$

For the Recurrent Highway Networks we have the following claim.

Claim 4. *A dynamical system induced by RHN in Equation 3.9 shows non-chaotic behavior when $R \in (1, 1)$, as thus follows from the properties of Vanilla RNN.*

Proof. To find the fixed points, we have to solve the equation: $x = f(x)$, so for the Equation 3.9 we will have:

$$
\begin{aligned}
s &= \sigma(R_t s) \odot (\tanh(R_h s) - s) + s \Rightarrow \\
0 &= \sigma(R_t s) \odot (\tanh(R_h s) - s) \Rightarrow \\
0 &= \sigma(R_t s) \text{ and } 0 = \tanh(R_h s) - s
\end{aligned}
$$

$0 = \sigma(R_t s) \Rightarrow$ no solutions exists, as the values of sigmoid function lies between 0 and 1. So we will consider only the second part.

$$
\begin{aligned}
0 &= \tanh(R_h s) - s \Rightarrow \\
s &= \tanh(R_h s)
\end{aligned}
$$

This equation $s = \tanh(R_h s)$ is the case for the simple RNN. We already considered the fixed points of the simple RNN in section 2.2. So the fixed points of the Recurrent Highway Networks are the same as the vanilla RNN. So, the fixed point analysis of the Vanilla RNN can be applied for the Recurrent Highway Networks (RHN). This proves our Claim 4. □

We tried to plot the bifurcation diagram for 1D RHN. Classic logistic map is given in the form: $x_{n+1} = r * x_n * (1 - x_n)$. What will be if we replace this equation with the $x_{n+1} = \sigma(r * x_n) \odot (\tanh(r * x_n) - x_n)) + x_n$. Then we will get the bifurcation diagram shown in Figure 3-3. From here, we can notice that the system is stable at some values of r. However, this example is very specific case of induced RHN form in 3.9, where two weight parameters are the same.

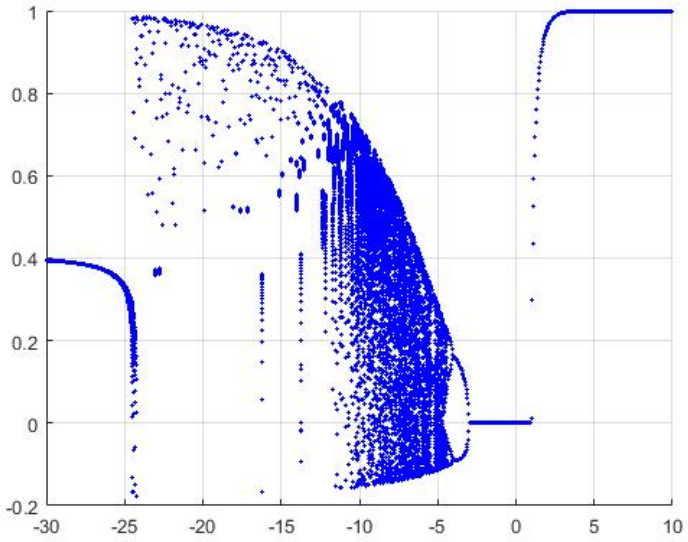


Figure 3-3: Bifurcation diagram of the 1D map $x_{n+1} = \sigma(r * x_n) \odot (tanh(r * x_n) - x_n)) + x_n$

3.2.2 RHN chaoticity in 2D

Claim 5. *There exists R: $||R|| > 1$ such that a dynamical system induced by RHN in Equation 3.9 is chaotic.*

We performed experiments to check the chaotic behavior of the RHN in 2D. We show that in the absence of the input data RHN can lead to dynamical systems $s_{n+1} = \Phi(s_n)$ that are chaotic ([53]). Again, we assume that there is no input data is provided. Then the dynamical system induced by a two-dimensional RHN with weight matrices: $R_t = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $R_h = \begin{bmatrix} -5 & -8 \\ 8 & 5 \end{bmatrix}$ and zero bias for the model. s can be initialized with any values. If we assume that no input data is provided and all bias terms are zero, then the induced RHN architecture will become:

$$t := \sigma(R_t s); \tag{3.10}$$

$$h := \tanh(R_h s); \tag{3.11}$$

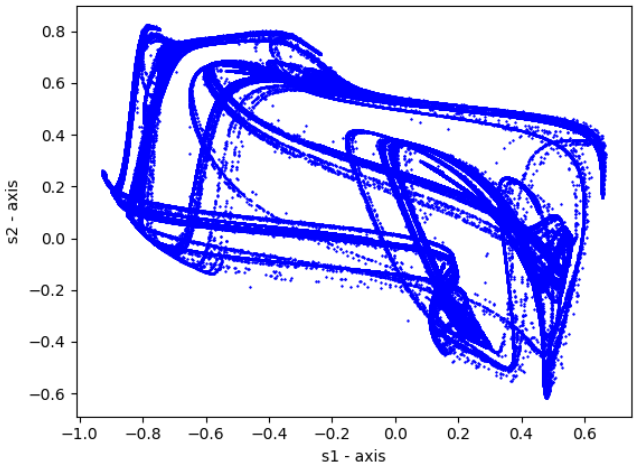


Figure 3-4: Strange attractor of chaotic behavior of RHN for the weight matrices: R_t=[[0, 1],[1, 0]] and R_h=[[-5, -8],[8, 5]]

$$s_{n+1} = t \odot (h - s_n) + s_n. \tag{3.12}$$

Now we plot the RHN state values $s_n^{(1)}$ vs. $s_n^{(2)}$ for $n = 100000$ iterations. The resulting plot is shown in Figure 3-4. Most trajectories converge toward the depicted attractor. We can get above pictures for any initial values of s (we can initialize with zeros or any values) and for any number of highway layers (we tried 1, 5, 10 highway layers). This picture shows the strange attractor as in LSTM and GRU given in Laurent and Brecht [27].

Now we studied time series analysis of this system. If we plot s_1 vs. n we can notice that the values of s_1 will jump from one place to another in the chaotic manner. There is no convergence. This is given in Figure 3-5. This is also true for s_2 vs. n (given in Figure 3-6). Then, if we plot the graph s_1 vs. s_2, we can get the strange attractor as shown in Figure 3-4.

Next we tested chaoticity of the RHN by using the Lyapunov instability of Bernoulli shift [39] as in section 2.2.2. We consider the two points which are initially very close to each other, with δs_0 "infinitesimally small" differences: $\delta s_0 := |s_0' - s_0|$. Then we iterate these two points through our induced RHN map $s_{n+1} = \Phi(s_n)$, in Equation

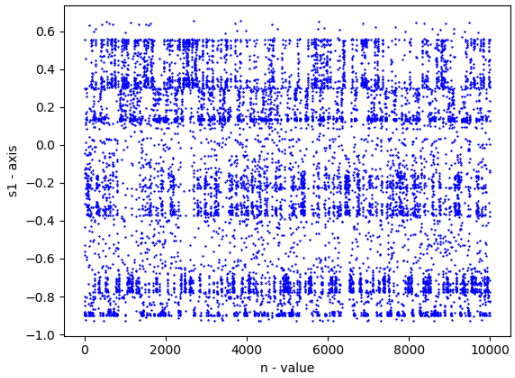


Figure 3-5: $s^{(1)}$ vs. n

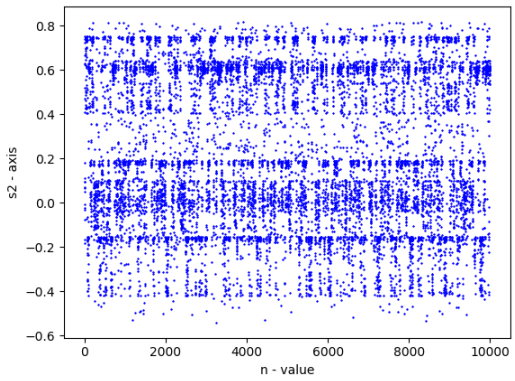


Figure 3-6: $s^{(2)}$ vs. n

3.9, 100 times and calculated the Euclidean distance between $|s'_n - s_n|$ these points. The graph is given in Figure 3-7. From this graph we can see that after some iteration, two trajectories diverge exponentially despite the fact that initially these two points are highly localized, with the distance no more than 10^{-7}. If we have positive

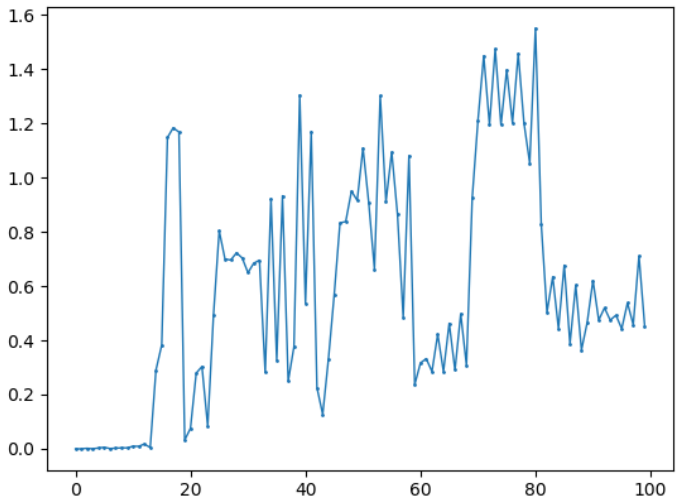


Figure 3-7: Euclidean distance from the 2 close points

λ, then we have a chaos. In Figure 3-8 shown the divergence of two very close points after each iteration.

Also we tested the weight matrices $R_t = \begin{bmatrix} -2 & 6 \\ 0 & -6 \end{bmatrix}$ and $R_h = \begin{bmatrix} -5 & -8 \\ 8 & 5 \end{bmatrix}$. The norm of these matrices are larger than one. If we plot the graph of s_1 vs. s_2 with $n = 100000$, then we again explore the strange attractor as shown in Figure 3-9. Here again, for any initial value of s and for any number of highway layers we will get this picture.

The third example of the chaoticity of RHN can be obtained by using the following parameters. Again, we consider the induced form of RHN as in Equation 3.9. If we initialize the weight matrices with the following matrices, $R_t = \begin{bmatrix} 0 & 3 \\ 2 & 0 \end{bmatrix}$ and $R_h =$

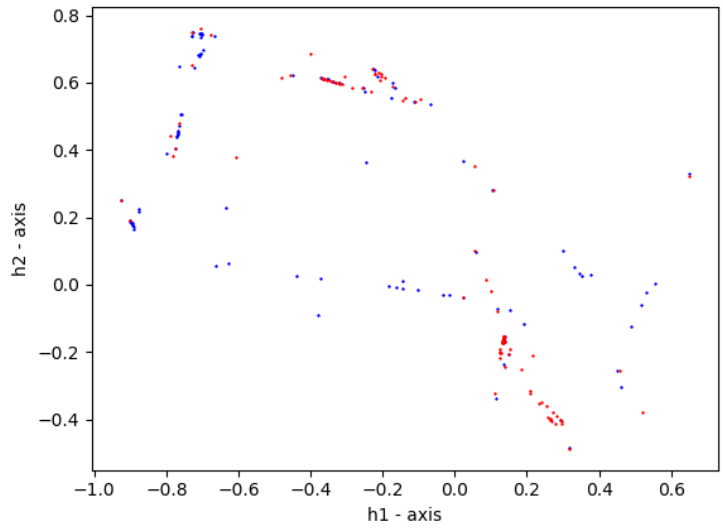


Figure 3-8: Divergence of two close points (red and blue points).

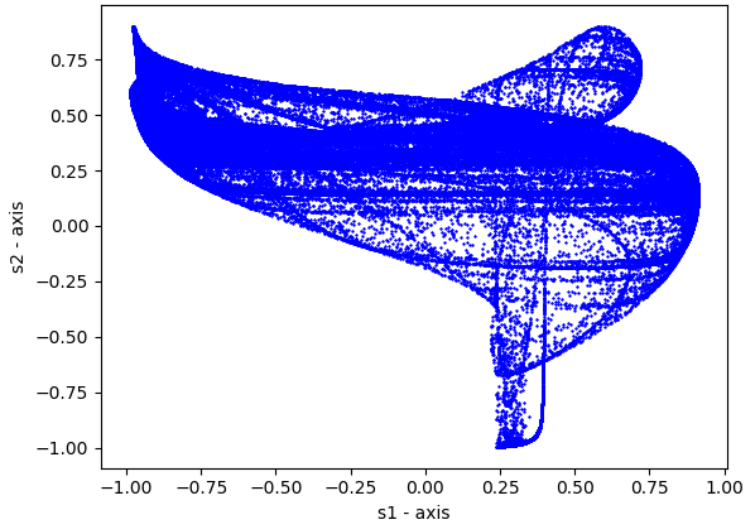


Figure 3-9: Strange attractor of chaotic behavior of RHN for the weight matrices: R_t=[[-2, 6], [0, -6]] and R_h=[[-5, -8], [8, 5]]

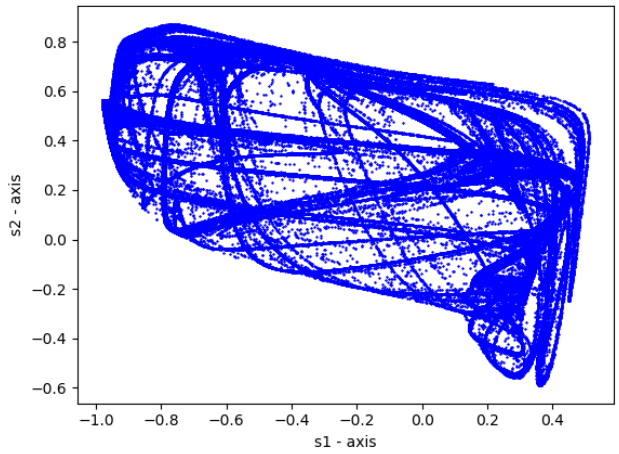


Figure 3-10: Strange attractor of chaotic behavior of RHN for the weight matrices: R_t=[[0, 3], [2, 0]] and R_h=[[-5, -8], [8, 5]]

$\begin{bmatrix} -5 & -8 \\ 8 & 5 \end{bmatrix}$ and if we assume that the bias term is zero, then again we explore the chaotic nature in induced RHN form. If we plot the forward trajectory of the induced RHN, then we can get the strange attractor shown in Figure 3-10. Another example of the chaoticity of RHN is given in B-1. Here again, we see the strange attractor for some parameters.

After exploring the chaotic behavior in RHN, we now tried to build chaos-free Neural Networks. For RHN, we again use our Claim 2 which was applied in vanilla RNN. Here, if we initialize the weights with matrices whose norm is smaller than one, then again we can have the non-chaotic behavior in RHN.

In the above cases, the norm of the weight matrices are larger than one and we explored the chaotic behavior. Now, let's analyze the case when the norm of the matrices are smaller than 1. For example, we can test these weight matrices:

$$R_t = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \text{ and } R_h = \begin{bmatrix} -0.4 & -0.3 \\ 0.3 & 0.2 \end{bmatrix}.$$

The norm of these two matrices are smaller than one. If we explore the values of s_1 and s_2 for $n \to \infty$, then, both values of s will go to zero. We also plotted the graph of s_1 vs. s_2. The plot is given in Figure 3-11. From this, we can see that we can get non-chaotic RHN when we initialize the weight matrices with the values whose norm

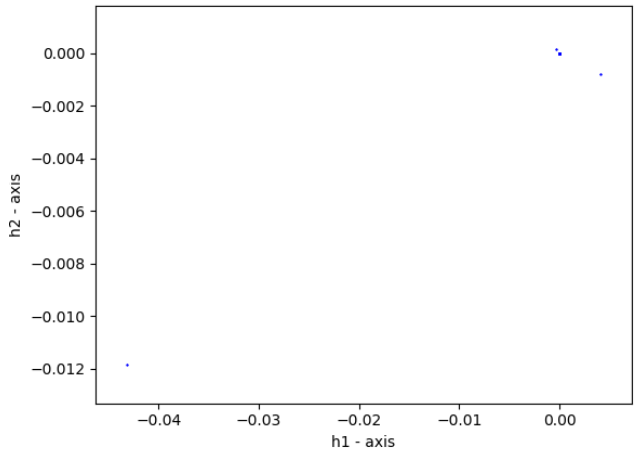


Figure 3-11: Attractor for weight matrices: R_t=[[0, 0.5], [0.5, 0]] and R_h=[[-0.4, -0.3], [0.3, 0.2]]

is smaller than one.

Chapter 4

SCRN chaoticity

4.1 SCRN architecture

Mikolov et al. [33] suggested an alternative RNN structure for the language modeling task. It was one of the attempts on modifying the vanilla Recurrent Neural Network. They called their RNN structure *Structurally constrained recurrent network (SCRN).*

As Mikolov et al. stated, the recurrent neural network is a powerful model that can learn temporal patterns in sequential data. However, for a long time, RNNs are difficult to train by using simple optimizers (e.g. stochastic gradient descent) during the training, because of the *vanishing gradient problem.* Mikolov et all [33] showed that learning longer term patterns in real data, in natural language, is perfectly possible using gradient descent. To achieve this, they do a slight structural modification on the simple Recurrent neural Network architecture. They force some of the hidden units to change their state slowly by making part of the recurrent weight matrix close to identity, and it will form a longer term memory. Their modified RNN structure achieved close results compared to the complex LSTMs networks on Language Modeling task.

For a sequential data, Recurrent Neural Networks models fits well in many applications and it showed the state-of-the-art results in many tasks: Automatic Speech Recognition, Language Modeling, Video Classification and etc. Recurrent Neural Networks represent time recursively, and this recursion allows the model to store

complex sequences for arbitrarily long time periods.

But in practice, training the Recurrent Neural Networks can be considered as hard. Simple recurrent networks suffer from the gradient vanishing problem during the training. For this reason, simple Recurrent Networks (SRN) memory focused only on short term patterns, ignoring longer term dependencies.

There are many architectures were proposed to solve these problems of SRN. Widely used architecture is LSTM models. Mikolov et al. also proposed their methods, which partially solves the vanishing gradient problems.

4.1.1 Simple Recurrent Networks

Mikolov et al., [33] firstly describe simple recurrent network (SRN) in terms of architectures, and then they compare it with their proposed models. In SRN, there are input layer, hidden layer with recurrent units and output layer. This can be seen from the Figure 4-1, (a). In SRN, given the one-hot encoding input x_t of a current

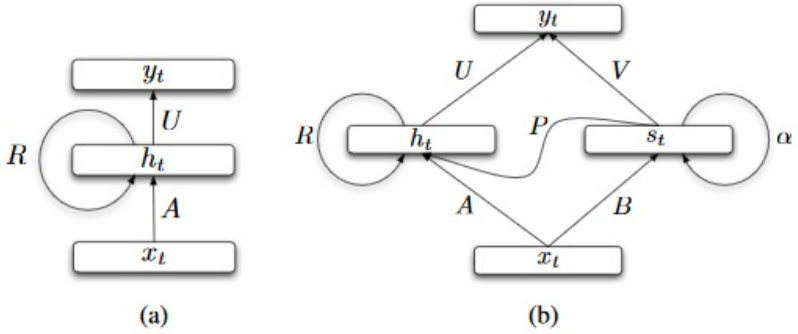


Figure 4-1: (a) Simple recurrent network. (b) RNN architecture suggested by Mikolov et all (2015), Recurrent network with context features.

token, it predicts the probability y_t of next one. Between input and output layers, there are recurrent layer, which has m hidden recurrent units, which allows to store an additional information about the previous tokens.

At each time t hidden state layer h_t is updated based on the previous state h_{t-1} and also the embedded input x_t of the current token, which can be described on the following equation:

$$h_t = \sigma(Ax_t + Rh_{t-1}),$$

where $\sigma(x) = 1/(1 + exp(x))$ is the sigmoid function applied coordinate wise, A is the $d \times m$ token embedding matrix and R is the $m \times m$ matrix of recurrent weights. Given the state of these hidden units, then network outputs the probability vector y_t of the next token, based on the following equation:

$$y_t = f(Uh_t),$$

where f is the soft-max function and U is the $m \times d$ output matrix.

4.1.2 Context features

Mikolov et al., [33] next describes their approach. The main difference from SRN, they add another hidden layer which learns the contextual features using stochastic gradient descent. This state of a hidden layer associated with a diagonal recurrent matrix.

So, in the model of Mikolov et al., they have both: a fully connected recurrent matrix to produce a set of quickly changing hidden units, and a diagonal matrix that encourages the state of the context units to change slowly (in Figure 4-1 (b) can be shown this detail). The first layer, or they call it “fast layer" (will be called *hidden layer*), will behave as the same as Simple Recurrent Networks (SRN), while the slowly changing layer (called *context layer*) can learn topic information, similar to cache models. Let's denote s_t the state of the p context units at time t, the update can be described by the following equations:

$$\begin{aligned} s_t &= (1 - \alpha)Bx_t + \alpha s_{t-1}, \\ h_t &= \sigma(Ps_t + Ax_t + Rh_{t-1}), \\ y_t &= f(Uh_t + Vs_t) \end{aligned}$$

where α is a parameter in (0, 1) and P is a $p \times m$ matrix. There is no nonlinear functions are applied to the state of the context units. This contextual hidden units can be seen as an exponentially decaying bag of words representation of the history.

This *exponential trace memory* has been already proposed in the context of simple recurrent networks [33].

Experiments

Mikolov et al. then tested their model for Language Modeling task in two datasets: Penn Tree Bank corpus and Text8 corpus, which is the first 100 million characters from the Wikipedia corpus. The results can be seen from Figure 4.1. Here, structurally constrained recurrent network (SCRN) model can achieve results comparable to the LSTM models, with relatively small numbers of parameters. Also, compared to Simple Recurrent Networks, SCRN shows better results with even much smaller number of parameters.

On the larger corpora Text8, LSTM showed slightly better results than the Mikolov et al. approach (SCRN) in larger models ($>$ 200 hidden units).

In conclusion, Mikolov et al. proposed architecture outperforms Simple Recurrent Network architecture, but showed slightly lower results compared to LSTM architecture in Large datasets, with large models. But one nice thing about their architecture is, they have less parameters and the architecture is not so complicated as LSTM architecture.

Model	#hidden	#context	Validation Perplexity	Test Perplexity
Ngram	-	-	-	141
Ngram + cache	-	-	-	125
SRN	50	-	153	144
SRN	100	-	137	129
SRN	300	-	133	129
LSTM	50	-	129	123
LSTM	100	-	120	115
LSTM	300	-	123	119
SCRN	40	10	133	127
SCRN	90	10	124	119
SCRN	100	40	120	115
SCRN	300	40	120	115

Table 4.1: Results on Penn Tree Bank corpus: SRN, LSTM and structurally constrained recurrent nets (SCRN).

4.2 SCRN chaoticity

Classic form of the SCRN is given in the following form:

$$s_{t+1} = (1-\alpha)x_{t+1}B + \alpha s_t$$
$$h_{t+1} = \sigma(x_{t+1}A + s_{t+1}P + h_t R)$$

Again, if we assume that no input is provided, then the induced form of SCRN will become:

$$s_{t+1} = \alpha s_t$$
$$h_{t+1} = \sigma(s_{t+1}P + h_t R)$$

Then, if we write them in one equation:

$$h_{t+1} = \sigma(\alpha s_t P + h_t R).. \tag{4.1}$$

Claim 6. *There exists R, P: $||R|| > 1$ and $||P|| > 1$ such that a dynamical system induced by SCRN in Equation 4.1 is chaotic.*

To verify this claim, we performed experiments with two-unit SCRN cells. We again assume that no input data is provided. We initialize weight matrices $P = \begin{bmatrix} -5 & 6 \\ 0 & -6 \end{bmatrix}$ and $R = \begin{bmatrix} -7 & -6 \\ 6 & -4 \end{bmatrix}$. Then we iterate the induced SCRN in Equation 4.1 100000 times. If we draw forward trajectories, $h^{(1)}$ vs $h^{(2)}$, we can get a strange attractor as shown in Figure 4-2. Most trajectories converge to the depicted attractor.

To get the picture shown in Figure 4-2, we initialize the values of h and s as follows: $h = [0.30463, 0.64438]$ and $s = [0.82458, 0.28021]$.

Now, let's study the time-series analyze of the above system. If we draw a graph of t vs. $h^{(1)}$ and t vs. $h^{(2)}$ then we can see the picture shown in Figure 4-3. In Figure 4-3 we can see that the values of $h^{(1)}$ and $h^{(2)}$ jump from one place to another in the chaotic manner. Up to a certain time iteration, $h^{(1)}$ and $h^{(2)}$ will be stable, then after

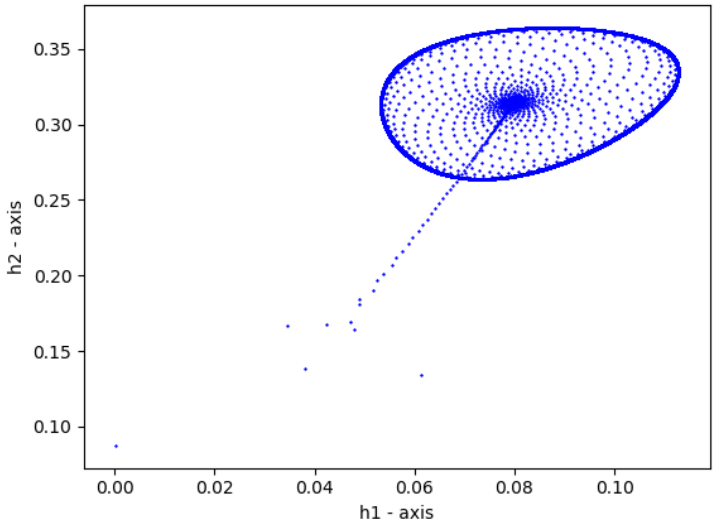
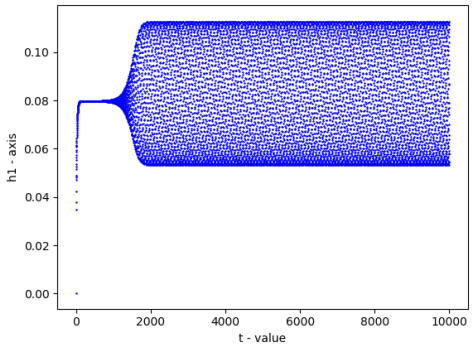
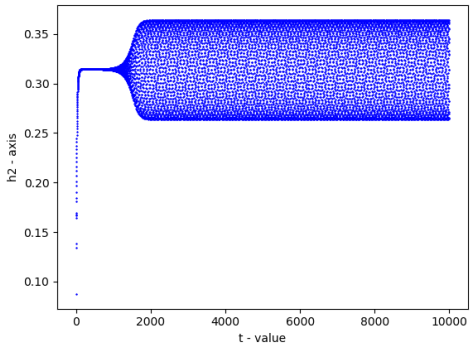


Figure 4-2: $h^{(1)}$ vs $h^{(2)}$ for the SCRN architecture, with weight matrices $P = [[-5, 6], [0, -6]]$ and $R = [[-7, -6], [6, -4]]$.



(a) SCRN, t vs. $h^{(1)}$



(b) SCRN, t vs. $h^{(2)}$

Figure 4-3: t vs. h for SCRN for $t = 10000$

a while they will jump from one place to another. One interesting thing, the values of $h^{(1)}$ and $h^{(2)}$ are bounded. If we enlarge the graph shown in Figure 4-3 in order to see how the values of h change over time t, we can get the graph shown in Figure B-2. Approximately at $t = 1000$ we notice jumps in the values of h.

Also we tested another example. We initialize weight matrices with $P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $R = \begin{bmatrix} -1 & -6 \\ 6 & -9 \end{bmatrix}$, and again assume there is no input data is provided. Then iterate induced SCRN in Equation 4.1 100000 times. If we draw a graph of state values, $h^{(1)}$ vs. $h^{(2)}$, we can get another strange attractor shown in Figure 4-4.

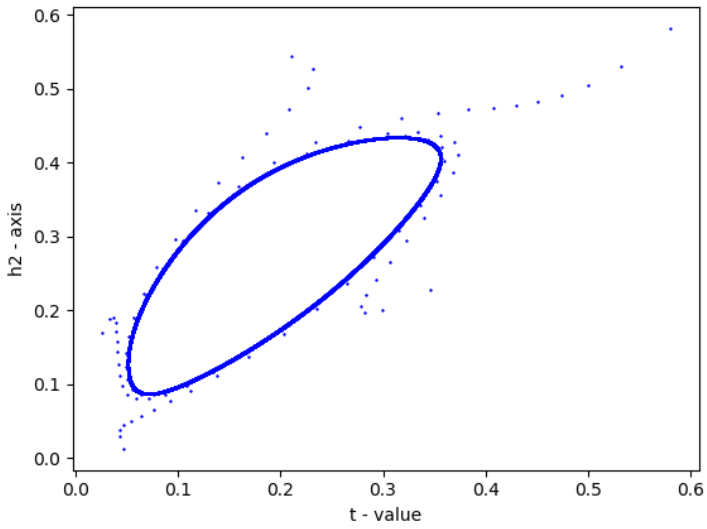


Figure 4-4: $h^{(1)}$ vs $h^{(2)}$ for the SCRN architecture, with weight matrices $P = [[1, 0], [0, 1]]$ and $R = [[-1, -6], [6, -9]]$.

Claim 7. *A dynamical system induced by SCRN in Equation 4.1 shows non-chaotic behavior when the norm of the weight matrices are smaller than one.*

To be convinced of this, we take weight matrices whose norm is smaller than one. Let's initialize weight matrices as follows: $P = \begin{bmatrix} -0.1237 & -0.3446 \\ 0.3282 & -0.3723 \end{bmatrix}$ and $R =$

$\begin{bmatrix} -0.475 & 0.3726 \\ 0 & -0.2363 \end{bmatrix}$. The norms of these matrices are less than one. If we iterate the induced SCRN 100000 times, and plot the graph of the state values $h^{(1)}$ vs. $h^{(2)}$, we can get the graph shown in Figure 4-5. There is no strange attractor.

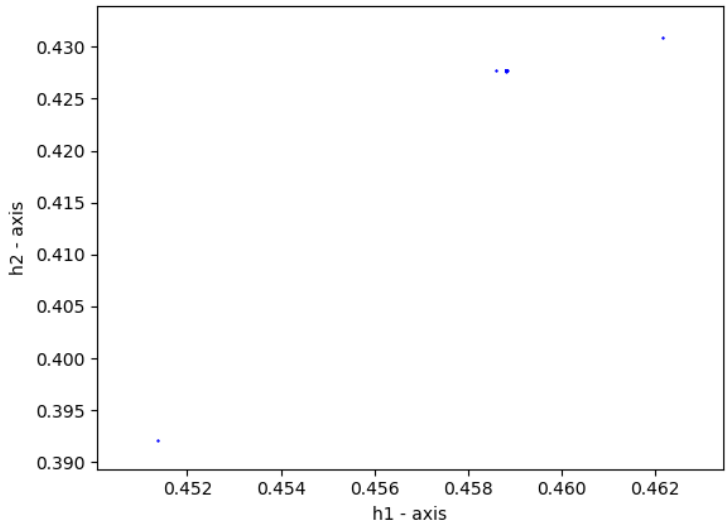
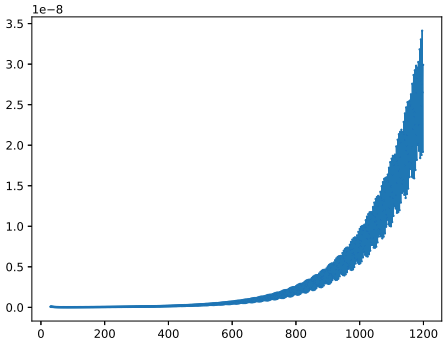


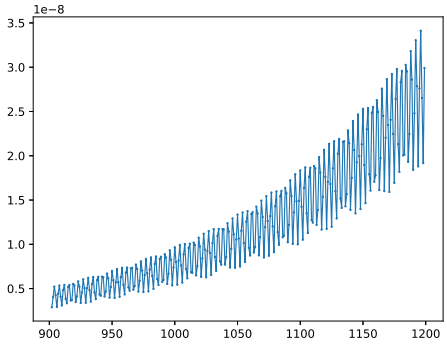
Figure 4-5: $h^{(1)}$ vs $h^{(2)}$ for the non-chaotic SCRN

Now we tested the chaoticity of the induced SCRN using the Lyapunov Exponent, as described in Section 2.2.2. We choose two points that are initially very close to each other. Then we iterate these two points through the SCRN map given in Equation 4.1, and calculate the Euclidean distance between them at each iteration. We initialize the weight matrices with $P = \begin{bmatrix} -7 & -6 \\ 6 & -4 \end{bmatrix}$ and $R = \begin{bmatrix} -5 & 6 \\ 0 & -6 \end{bmatrix}$, whose norms are larger than 1. Then, if we draw a graph of the Euclidean distance, we can get the graph shown in Figure 4-6a. If we enlarge this Figure for a certain period of time, then we can get the graph shown in Figure 4-6b. Here we can see that the difference increases after some iteration and continues to increase.

Next, we repeat above experiment, but, in this case we initialize weight matrices $P = \begin{bmatrix} -0.1237 & -0.3446 \\ 0.3282 & -0.3723 \end{bmatrix}$ and $R = \begin{bmatrix} -0.475 & 0.3726 \\ 0 & -0.2363 \end{bmatrix}$, whose norms are smaller than one. Then, if we draw the graph of the Euclidean distance, we can get the graph



(a) $h^{(1)}$ vs t



(b) $h^{(2)}$ vs t

Figure 4-6: Euclidean distance from the 2 close points for the chaotic SCRN.

shown in Figure 4-7. From here, again, we can see that if we take weight matrices whose norm is smaller than one, then the difference will be equal to zero immediately after the first iteration.

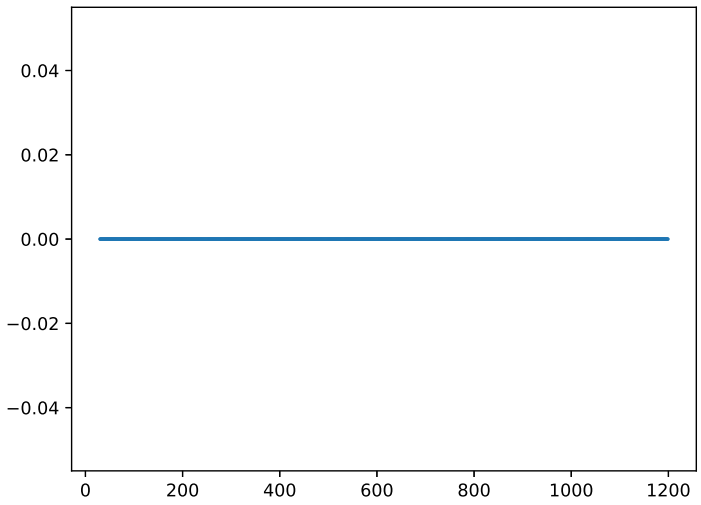


Figure 4-7: Euclidean distance from the 2 close points for the non-chaotic SCRN.

Chapter 5

EXPERIMENTS

5.1 Language Modeling

The main task of the language model is to determine whether the particular sequence of words is appropriate or not in some context, determining whether the sequence is accepted or discarded. It is used in various areas such as speech recognition, machine translation, handwriting recognition [43], spelling correction [24], augmentative communication [38] and Natural Language Processing tasks (part-of-speech tagging, natural language generation, word similarity, machine translation) [11, 9, 19]. Strict rules may be required depending on the task, in which case language models are created by humans and hand constructed networks are used. However, development of the rule-based approaches is difficult and it even requires costly human efforts if large vocabularies are involved. Also usefulness of this approach is limited: in most cases (especially when a large vocabulary used) rules are inflexible and human mostly produces the ungrammatical sequences of words during the speech. One thing, as [56] states, in most cases the task of language modeling is "to predict how likely the sequence of words is", not to reject or accept as in rule-based language modeling. For that reason, statistical probabilistic language models were developed. A large number of word sequences are required to create the language models. Therefore the language model should be able to assign probabilities not only for small amounts of words, but also for the whole sentence. Nowadays it's possible to create large and readable text

corpora consisting of millions of words, and language models can be created by using this corpus.

5.1.1 Neural Language Modeling

Given the sequence of words, we want to predict the probability of each word (in the dictionary). Language models allow us to measure the probability of choice, which is an important contribution to machine translation (since sentences are likely to be correct). A side effect of this ability is the ability to generate new texts by choosing from output probabilities. We can generate other things, depending on what our data is. In language modeling, our input usually represents a sequence of words (for example, encoded as a vector with one hot state (one-hot)), and the output is a sequence of predicted words. When learning the neural network, we feed the previous layer $o_t = x_{t+1}$ to the next layer as we want the result at step t to be the next word.

There are many types of neural architectures, which also applied successfully for the language modeling tasks. Starting from the work of [8] there are many Recurrent Neural Architectures proposed. With Recurrent Neural Networks, it's possible to model the word sequences, as the recurrence allows to remember the previous word history. Recurrent Neural Network can directly model the original conditional probabilities:

$$P(w_1, ..., w_n) = \prod (w_i | w_1 ... w_{i1}) \tag{5.1}$$

To model the sequences, f function constructed via recursion, initial condition is given by $h_0 = 0$ and the recursion will be $h_t = f(x_t, h_{t-1})$. Here, h_t is called *hidden state* or *memory* and it memorizes the history from x_1 up to x_{t1}. Then, the output function is defined by combination of h_t function:

$$P(w_1, ..., w_n) = g_w(h_t) \tag{5.2}$$

f can be any nonlinear function such as tanh, ReLU and g can be a softmax function. In our work, we followed [58] who presented a simple regularization technique

for Recurrent Neural Networks (RNNs) with LSTM [18] units. Srivastava et al. [50] proposed dropout technique for regularizing the neural networks, but this technique does not work well with RNNs. This regularizing technique is tent to have overfitting in many tasks. Zaremba et al. [58] showed that the correctly applied dropout technique to LSTMs might substantially reduce the overfitting in various tasks. They tested their dropout techniques on language modeling, speech recognition, machine translation and image caption generation tasks. We also reproduced the language modeling experiments from Zaremba et al. [58] using Tehsorflow [1] and the code can be found here: `https://github.com/Baghdat/RNN-word-Zaremba`.

5.1.2 Experiments with Chaos Free Network

Laurent & Brecht [27] briefly demonstrated that in the absence of input data, LSTM and GRU can lead to chaotic dynamic systems $u_t = \Phi(u_{t-1})$ (Strogatz, 2014 [53]). The figure shows a strange attractor of a dynamical system, and it can be obtained from: $u_t = \begin{bmatrix} h_t \\ s_t \end{bmatrix}$ $u_t \mapsto \Phi(u) = u_t = \begin{bmatrix} o \odot \tanh(f \odot c + i \odot g) \\ f \odot c + i \odot g \end{bmatrix}$

$i := \sigma(W_i h + b_i)$ $f := \sigma(W_f h + b_f)$ $o := \sigma(W_o h + b_o)$ $g := tanh(W_g h + b_g)$

They used a two-position LSTM with weight matrices to show that LSTM has a strange attractor:

$$W_i = \begin{bmatrix} -1 & -4 \\ -3 & -2 \end{bmatrix} \; W_o = \begin{bmatrix} 4 & 1 \\ -9 & -7 \end{bmatrix} \; W_f = \begin{bmatrix} -2 & 6 \\ 0 & -6 \end{bmatrix} \; W_g = \begin{bmatrix} -1 & -6 \\ 6 & -9 \end{bmatrix}$$

and zero bias for model parameters. These weight matrices were randomly formed from the normal distribution with a standard deviation of 5. The strange attractor in the figure was obtained by choosing in the initial state $u_0 = (h_0; c_0)$ uniformly randomly in $[0; 1]^2 \times [0; 1]^2$. Then they built the h-component of the iterates of $u_t = (h_t; c_t)$ for t between 10^3 and 10^5.

We also conducted this experiment. We wrote a Python script to reproduce the strange attractor. If we gave the weights that the authors provided, then we can get the same picture as the authors.

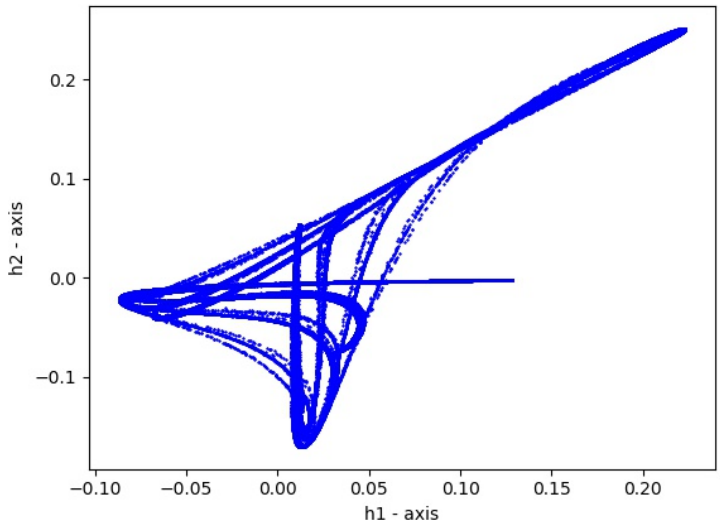


Figure 5-1: Strange attractor in two-unit LSTM

However, if we give different weight values, then we may not get a strange attractor. For example, we simply changed the values of the weight matrices W_i and W_o. (Instead of W_i, weights W_o were used and vice versa). Then we can get the following diagram for h_1 versus h_2 (Figure 5-2).

It can be seen from the figure above that if we take random weights, we might not get the same attractor that was presented in the article for LSTM.

We also reproduced the strange attractor with the GRU (Figure 5-3). Again, if we give the same weight matrices as the authors, then we can get the same picture, and if we change the weight matrices, then we cannot get a picture of the strange attractor.

We also reproduced the results for the language modeling tasks. Implemented chaos-free network can be found here: `https://github.com/Baghdat/Chaos-Free-Network`. We reproduced the small sized and large sized Chaos-Free Networks. For 20 million (in our experiment the model size was 17238900) parameter model with 2 layers, the perplexity was 82.327/78.137 on validation/test sets with dropout and the perplexity was 118.82/113.98 on validation/test sets without dropout. We also made 1 layer CFN model.

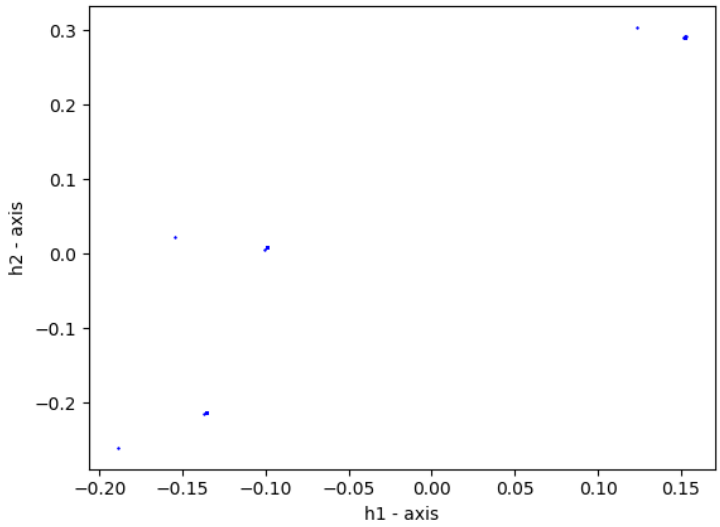


Figure 5-2: h_1 vs. h_2 in two-unit LSTM

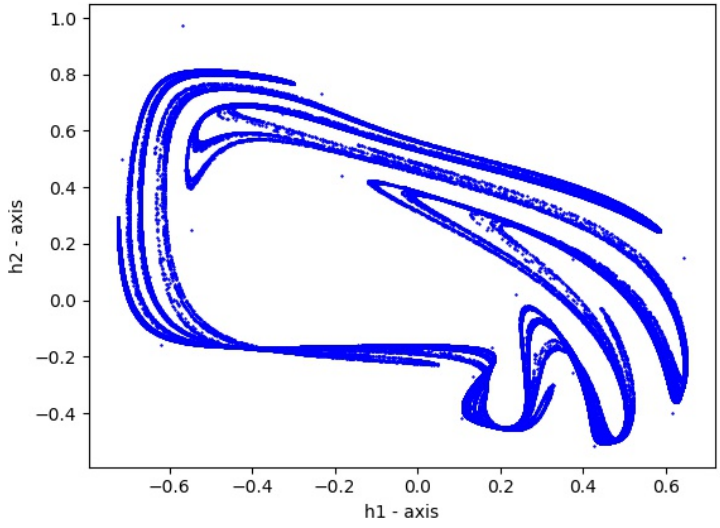


Figure 5-3: Strange attractor in two-unit GRU.

5.2 Non-chaotically initialized RHN

In this section, we tested our non-chaotic neural cells in real-world applications. Our aim is to identify, how non-chaotic version will affect on the performance. Is a non-chaotic behaviour good in a real-world application? Do we need a chaoticity? Or is it good to have a chaotic systems? To answer these questions, we performed experiments.

We examined Recurrent Highway Networks on the language modeling task. We use Penn Tree Bank (PTB) [31] corpus, which was pre-processed by Mikolov et al. [34]. First we reproduced the initial results from Zilly et al. [59] without weighting (WT) of input and output mappings and got the 68.355 perplexity on the validation set and 65.506 perplexity on the test set. These results are similar to the results in the paper (In the paper it was 67.9 and 65.4).

Then we tested our chaos-free version. We initialized the weight matrix in a way, such that their Frobenius norm do not exceed 1. We use TensorFlow ([1]) to perform our experiments. We first created a matrix whose norm is smaller than one and feed it during the initialization. We used the same hyper-parameters as in Zilly et al. [59] during the training. On PTB dataset, our non-chaotic neural cells showed 68.715 perplexity on the validation set and 66.290 perplexity on the test set. Full results and results of Chaos Free Network (CFN) [27] are given in Table 5.1. From this, we can see that the chaos free version of RHN showed similar results as the chaotic version and that chaos-free initialization will not lead to a decrease in performance.

The code for the experiments can be found here: `https://github.com/Baghdat/RHN_init`.

Table 5.1: Perplexity on the PTB set.

Model	Validation Perplexity	Test Perplexity
Variational RHN + WT [59]	68.355	65.506
Non-chaotically initialized RHN	**68.715**	**66.290**
CFN (2 layers)+dropout [27]	79.7	74.9

5.3 Conclusion and future work

In this thesis work we analyzed the dynamics of the Recurrent Neural Networks. Our analyses showed that the vanilla RNN, Structurally Constrained Recurrent Network and the most recent RHN architecture exhibit a chaotic behavior in the absence of input data. We found out that, depending on the initialization of the weight matrices, we can have non-chaotic systems. Our experiments showed that the initialization of the weights with the matrices whose norm is less than one can lead to non-chaotic behavior. The advantage of non-chaotic cells is stable dynamics. We also performed experiments with non-chaotic RHN cells. Our experiments on language modeling with the PTB dataset showed similar results as an RHN cell with chaos by using the same hyper-parameters. In the future, we are going to test non-chaotic RHN cells for other tasks: speech processing, image processing. Also for NAS architecture, at this moment, generating the architecture is an expensive process for us, as there are not enough resources. We will test our chaos-free initialization for NAS architectures again.

Appendix A

Tables

Table A.1: Lyapunov coefficient versus W value

Parameter value (W)	Lyapunov Exponent	Parameter value (W)	Lyapunov exponent
-1	0.00000211	-1.2	0.150067308
1	0.00000101	1.2	0.150055520
1.1	0.08631033	2	-inf
-1.1	0.08630209	-2	-inf
0.9	-0.11629468	0.5	-0.88135679
-0.9	-0.11629206	-0.5	-0.88135706

Appendix B

Figures

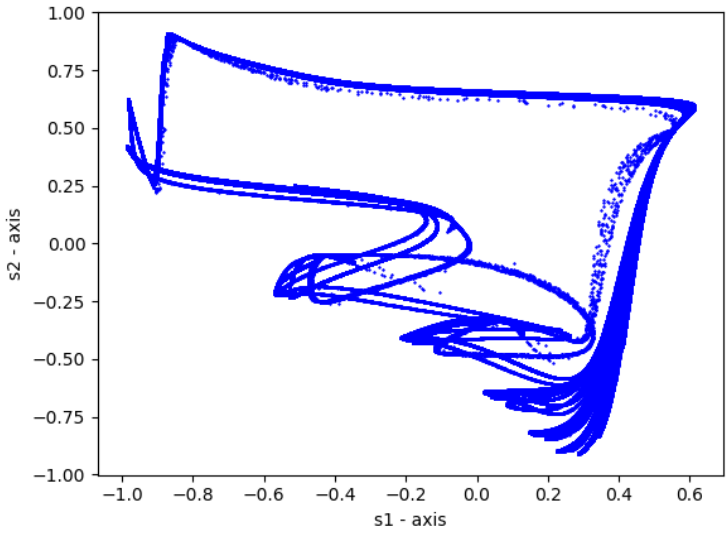
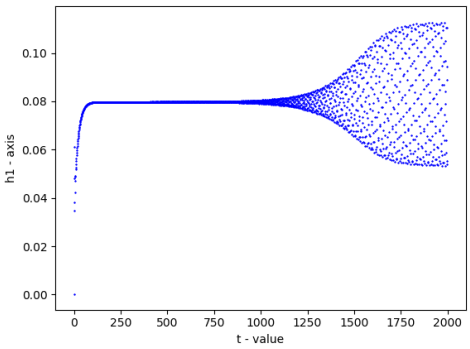
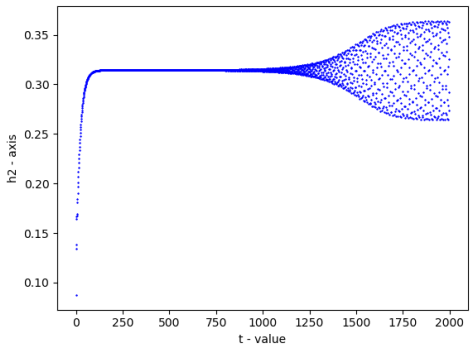


Figure B-1: Strange attractor of chaotic behavior of RHN for the weight matrices: R_t=[[1, 3], [2, 0]] and R_h=[[-5, -12], [9, 4]].



(a) SCRN, t vs. $h^{(1)}$



(b) SCRN, t vs. $h^{(2)}$

Figure B-2: t vs. h for SCRN for $t = 2000$

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[2] VS Afraimovich and Ml I Rabinovich. Dynamic system. https://bigenc.ru/physics/text/1956597. Accessed: 2019-03-30, in Russian.

[3] VS Anishchenko. Complex oscillations in simple systems, 1990.

[4] VS Anishchenko. Dynamic systems. *Soros Educational Journal*, 11:77–84, 1997. (in Russian).

[5] Dmitry Victorovich Anosov. Geodesic flows on closed riemannian manifolds of negative curvature. *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 90:3–210, 1967.

[6] D Assaf and Steve Gadbois. Definition of chaos. *American Mathematical Monthly*, 99(9):865–865, 1992.

[7] J. Banks, J. Brooks, G. Cairns, G. Davis, and P. Stacey. On devaney's definition of chaos. *Am. Math. Monthly*, 99(4):332–334, April 1992.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[9] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, June 1990.

[10] Kyunghyun Cho, Bart van Merriënboer, ÇaÄ§lar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[11] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, ANLC '88, pages 136–143, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics.

[12] Robert L. Devaney. *An introduction to chaotic dynamical systems / Robert L. Devaney*. Addison-Wesley Redwood City, Calif, 2nd ed. edition, 1989.

[13] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[14] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc., 2016.

[15] Semyon Aranovich (S. Gerschgorin) Geršgorin. über die abgrenzung der eigenwerte einer matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 6:749–754, 1931.

[16] Alex Graves. Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983, 2016.

[17] Denny Gulick. *Encounters with chaos and fractals*. Crc Press, 2012.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[19] Jonathon Hull. Combining syntactic knowledge and visual text recognition: A hidden markov model for part of speech tagging in a word recognition algorithm. In *AAAI Symposium: Probabilistic Approaches to Natural Language*, pages 77–83, 1992.

[20] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

[21] Michael I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In Joachim Diederich, editor, *Artificial Neural Networks*, pages 112–127. IEEE Press, Piscataway, NJ, USA, 1990.

[22] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press, 2016.

[23] Andrei Yu Kolesov and Nikolai Kh Rozov. On the definition of 'chaos'. *Russian Mathematical Surveys*, 64(4):701, 2009.

[24] Karen Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439, December 1992.

[25] Yuri A. Kuznetsov. *Elements of Applied Bifurcation Theory (2Nd Ed.)*. Springer-Verlag, Berlin, Heidelberg, 1998.

[26] Polina S. Landa and P.V.E. McClintock. Development of turbulence in subsonic submerged jets. *Physics Reports*, 397(1):1 – 62, 2004.

[27] Thomas Laurent and James von Brecht. A recurrent neural network without chaos. *arXiv preprint arXiv:1612.06212*, 2016.

[28] Edward Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130–148, 1963.

[29] Alexander Loskutov. Mathematical foundations of chaotic dynamical systems. *Successes of physical Sciences*, 177(9):989–1015, 2007. (in Russian).

[30] A. M. LYAPUNOV. The general problem of the stability of motion. *International Journal of Control*, 55(3):531–534, 1992.

[31] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, June 1993.

[32] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[33] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

[34] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, volume 2010, pages 1045–1048. International Speech Communication Association, 2010.

[35] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *SLT*, pages 234–239. IEEE, 2012.

[36] Francis C Moon. *Chaotic and fractal dynamics: introduction for applied scientists and engineers*. John Wiley & Sons, 2008.

[37] Oksana Anatolevna Muzyka. Bifurkatsii v prirode i obshchestve: yestestvenno nauchnyy i sotsiosinergeticheskiy aspect (bifurcations in nature and society: the natural scientific and sociosynergetic aspect). *Sovremennyye naukoyemkiye tekhnologii - Modern high technology*, 1:87–91, 2011. in Russian.

[38] Alan Newell, Stefan Langer, and Marianne Hickey. The rÔle of natural language processing in alternative and augmentative communication. *Nat. Lang. Eng.*, 4(1):1–16, March 1998.

[39] Edward Ott. *Chaos in Dynamical Systems*. Cambridge University Press, 2 edition, 2002.

[40] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[41] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. Association for Computational Linguistics, 2017.

[42] David Ruelle and Floris Takens. On the nature of turbulence. *Les rencontres physiciens-mathématiciens de Strasbourg-RCP25*, 12:1–44, 1971.

[43] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[44] Leonid Pavlovich Shilnikov. Some cases of generation of period motions from singular trajectories. *Matematicheskii Sbornik*, 103(4):443–466, 1963.

[45] Leonid Pavlovich Shilnikov. A case of the existence of a denumerable set of periodic motions. In *Doklady Akademii Nauk*, volume 160, pages 558–561. Russian Academy of Sciences, 1965.

[46] Shodhganga. Bifurcation and lyapunov exponent of a chaotic cubic map. `http://shodhganga.inflibnet.ac.in/bitstream/10603/70172/7/07_chapter202.pdf`. Accessed: 2019-02-10.

[47] Yakov Grigor'evich Sinai. On the foundations of the ergodic hypothesis for a dynamical system of statistical mechanics. In *Doklady Akademii Nauk*, volume 153(6), pages 1261–1264. Russian Academy of Sciences, 1963.

[48] Yakov Grigor'evich Sinai. Dynamical systems with elastic reflections. ergodic properties of dispersing billiards. *Uspekhi Matematicheskikh Nauk*, 25(2):141–192, 1970.

[49] S SMALE. Diffeomorphisms with many periodic points. *differential and combinatorial topology*, pages 63–80, 1963.

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[51] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2377–2385. Curran Associates, Inc., 2015.

[52] Rupesh Kumar Srivastava, Bas R Steunebrink, and Jürgen Schmidhuber. First experiments with powerplay. *Neural Networks*, 41:130–136, 2013.

[53] Steven H Strogatz. *Nonlinear Dynamics and Chaos with Student Solutions Manual: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 2018.

[54] David Sussillo and Omri Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013. PMID: 23272922.

[55] Pavlo Teslenko. Evolutionary theory and synergetics in project management. *Project Management and Production Development*, 4 (36), 2010. in Russian.

[56] Edward William Daniel Whittaker. *Statistical language modelling for automatic speech recognition of Russian and English*. Doctoral dissertation, University of Cambridge, 2000.

[57] Alan Wolf, Jack B. Swift, Harry L. Swinney, and John A. Vastano. Determining lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3):285 – 317, 1985.

[58] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

[59] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4189–4198, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[60] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2017.