

BACHELOR THESIS
ARTIFICIAL INTELLIGENCE

Radboud University



Hyperparameter specialization in the Hierarchical-Task Reservoir

Author:
Mariia Zamyrova
S1038789

First supervisor:
dr. R.C. Farinha Duarte
Donders Institute for
Brain, Cognition and
Behaviour; Radboud
University
renato.duarte@donders.ru.nl

Second supervisor:
dr. M. Shahsavari
Donders Institute for
Brain, Cognition and
Behaviour; Radboud
University
mahyar.shahsavari@ru.nl



June 18, 2022

Abstract

The Reservoir Computing framework presents a more efficient approach to training Recurrent Neural Networks. A classic implementation of this framework is the Echo State Network. In addition to being easier to train, Echo State Networks can display a variety of dynamics, which are controlled by the network hyperparameters. When working with complex multiscale sequential data the capabilities of singular Echo State Networks can prove insufficient. In those cases, Hierarchical Echo State Networks are the better choice. Hierarchical Echo State Networks display richer dynamics by having different timescales on each layer of the model. Recently, a novel Hierarchical Echo State Network, called the Hierarchical-Task Reservoir, has been introduced. Its key feature is abstraction. The model's task is divided into sub-tasks. Every layer in the Hierarchical-Task Reservoir performs a different sub-task as opposed to "vanilla" Hierarchical Echo State Networks, where each layer has the same task. The analysis of the Hierarchical-Task Reservoir's performance showed that the model has higher accuracy with respect to Hierarchical Echo State Networks with no task abstraction. This thesis focuses on investigating how the Hierarchical-Task Reservoir is able to achieve better performance. The novel model is analyzed from the perspective of its hyperparameters. The goal is to find out how the parameters control the model accuracy and which parameters have the most impact. Additionally, the effect of the abstraction feature on the parameter values is evaluated, as well as the weight of the feature's contribution to the model accuracy, in comparison to the contribution of hyperparameters. It is concluded that the model performs best with optimal parameter values and that the abstraction feature has most effect on the model dynamics.

Contents

1	Introduction	2
2	Preliminaries	4
3	Related Work	8
4	Research	10
5	Discussion & Conclusions	17
A	Additional Research Results	23

Chapter 1

Introduction

Recurrent Neural Networks (RNNs) are a frequent choice when working with temporal data like speech. The connections between RNN units can have cycles (recurrent connections) which makes RNNs dynamical systems, whose state changes through time. Those cycles also allow past network inputs and states to have impact on the future states, creating dynamical memory. RNNs, however, are complicated and computationally expensive to train with gradient-descent-based approaches. For larger networks, vanishing gradients and extensive numbers of update cycles can be observed. This can limit the possible network sizes, which can, in turn, reduce the achievable richness of the network dynamics [13].

With the introduction of the RNN-based Reservoir Computing paradigm, specially networks like Liquid State Machines (LSMs) [14] and Echo State Networks (ESNs) [7] this problem has been remedied. What gives Reservoir Computing models the edge is the fact that they have a much simpler training procedure. They typically consist of a layer of input units, a randomly connected, fixed RNN, called a reservoir, and a layer of output units. Through preservation of certain properties in the reservoir the only weights that need to be trained are the final output weights. These weights are tuned with simple linear regression. This significantly lowers the time and effort needed for training, while still letting the network keep the aforementioned advantageous characteristics of classic RNNs [14, 7].

All of the models discussed in this report present variations on the "vanilla" Echo State Networks. Ever since ESNs were introduced there has been a lot of research done both on possible applications of ESNs [10, 18, 17, 6] and on the internal structure and properties of ESNs [17, 2, 8]. An important topic is hierarchical ESNs [2]. These are models that consist of chained ESNs, where each sub-network takes as input the output of its preceding sub-network. Hierarchical ESNs have been proven to exhibit more complex system dynamics in comparison to singular ESNs. They can develop different dynamics on each layer of the model. Therefore, they are a good fit for simulating and predicting multiscale temporal data like speech

signals, where there are both short-frequency components, e.g. phones, and longer frequency ones, e.g. words. Typically, lower layers of the model usually have shorter timescales and higher layers have longer timescales. This diversification is modulated by the model's hyperparameters[3].

This thesis focuses on a novel hierarchical ESN model for speech processing called the Hierarchical-Task Reservoir (HTR). The distinctive quality of HTR is that it uses abstraction. This is done by breaking the main model task down into sub-tasks. Each layer of the model performs a different sub-task, more abstract than its predecessor. In addition to performing different tasks, it was observed that the layers of HTR develop different dynamics. HTR has outperformed a single-layer ESN, but most importantly, a hierarchical ESN that has the same structure as the HTR, the only difference between them being that it does not use abstraction [16].

The aim of this research is to gain a deeper understanding of how HTR achieves its high accuracy and how much the infusion of abstraction contributes to the model's performance. Knowing how effective the use of abstraction is in achieving HTR's high level of precision can serve as a motivation for (not) continuing to use this feature. Given that hyperparameters dictate the dynamics of ESNs and that HTR layers display multiple dynamics, the research is centered around the hyperparameters of HTR. It analyses the effect various parameter value configurations have on the model performance. This is done in order to establish how much of the HTR's superior accuracy, in comparison to other models, comes from having the right hyperparameters and how much is added through abstraction.

The remainder of this report is structured in the following way. Chapter 2 provides an overview of the Echo State Network architecture and properties as well as a more detailed description of the HTR. Chapter 3 gives a survey of the papers that are closely related to the current research. Chapter 4 describes the set up of experiment conducted as part of this research. The report is concluded in Chapter 5 with the analysis of results, discussion of this work's limitations and suggestions for future research.

Chapter 2

Preliminaries

Introduction to Echo State Networks

An Echo State Network (ESN) is a type of RNN implemented according to the Reservoir Computing guidelines. The distinctive quality of an ESN is its learning mechanism. It is significantly simplified with respect to the methods typically used on RNNs and entails only training a set of output weights with linear regression.

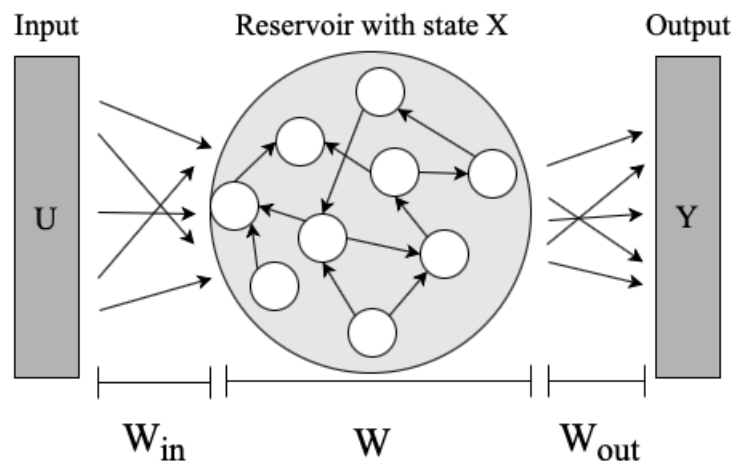


Figure 2.1: Echo State Network architecture

A classic Echo State Network consists of an input neuron layer of size N_u , a fixed randomly connected RNN with N_x units, called a reservoir, and an output layer with N_y units. The parameters of an ESN are the weight matrices $W_{in} \in \mathbb{R}^{N_x \times N_u}$ (modulates the connectivity between the input units and reservoir units), $W \in \mathbb{R}^{N_x \times N_x}$ (the internal connectivity matrix of the reservoir) and $W_{out} \in \mathbb{R}^{N_y \times (N_x + N_u)}$ (modulates the connectivity of the reservoir and input units with the output units) (Figure 2.1). Notice from

the dimensions of W that a single neuron can be connected to every other neuron in the pool. It is also worth mentioning that it is possible to add feedback weights W_{fb} , directed from the output layer back to the reservoir. When feedback weights are used the ESN output acts as teacher input for the reservoir.

The weight matrices W_{in} and W are usually randomly generated from a normal or uniform distributions. The reservoir matrix is suggested to be made sparse for efficiency, but this is not a strict condition for fast performance. The other two matrices W_{in} and W can be dense [11].

The state of the reservoir $x(t)$ at time step t is a vector of dimension N_x , comprised of the individual state of each reservoir unit. It is computed as:

$$x(t) = f(W_{in}u(t) + Wx(t-1)) \quad (2.1)$$

where $u(t) \in \mathbb{R}^{N_u}$ is the input and f is the activation function, e.g. sigmoid or tanh. For the remainder of this report it is assumed that the activation function f is a tanh.

The output of the ESN at time t is computed as:

$$y(t) = W_{out}x(t) \quad (2.2)$$

The output and the reservoir states can be aggregated into matrices Y and X over a time period $t \in \{0, 1, \dots, T\}$:

$$Y = W_{out}X \quad (2.3)$$

The most common method for training W_{out} is linear regression, namely ridge regression:

$$W_{out} = Y_{target}X^T(XX^T + \lambda I)^{-1} \quad (2.4)$$

where Y_{target} is the target output, λ is the regularization parameter and I is the identity matrix [11].

The key hyperparameters of the reservoir that influence the dynamics of the network are the spectral radius ρ and the input norm σ . The spectral radius affects how durable the influence of external inputs, coming from the input layer, is on reservoir's future states, i.e. the memory capacity of the reservoir. It does so by scaling the matrix W , thus controlling the magnitude of the matrix values and the weight of each reservoir unit in the overall state $x(t)$. Larger ρ values are associated with longer memory and higher non-linearity of the reservoir dynamics. The input norm regulates the linearity of reservoir dynamics. It is used to scale the matrix W_{in} and, as such, how much influence current input has on the reservoir state. Larger σ values are associated with higher non-linearity.

An ESN can have an additional parameter - the leaky integrator α , which makes the network a Leaky Integrator ESN (LI-ESN). The leaky integrator modifies the strength of influence a past reservoir state $x(t)$ has on a future state $x(t + 1)$.

For LI-ESNs the state formula is:

$$x(t) = (1 - \alpha)x(t - 1) + \alpha f(W_{in}u(t) + Wx(t - 1)) \quad (2.5)$$

The leaky integrator can be expressed as:

$$\alpha = \frac{\delta t}{\tau} \quad (2.6)$$

The δt is the reservoir state update step-size. In the context of this thesis $\delta t = 1$, i.e. $x(t + \delta t) = x(t + 1)$. The τ is the reservoir dynamic timescale, the duration of a single unit of the signal.

The analysis of the weight training process in RNNs showed that the internal reservoir weights get updated at a slower rate than the output weights. Additionally, W serves a different purpose in comparison to W_{out} . The internal weights represent an expansion of the input data into a high-dimensional space and the output weights linearly combine the points in that space. The ESN training process builds on that clear distinction between the weight matrices and on the fact that, granted specific W properties, it is enough to only train W_{out} [12]. The key required characteristic of W is the Echo State Property (ESP). This property states that the influence of external input on the state $x(t)$ should fade with time to ensure the stability of the reservoir dynamics. ESP is typically enforced by the following restriction: maximal eigenvalue of the recurrent weight matrix $\rho(W)$, a.k.a. the original spectral radius, must be less than 1. However, it has been observed multiple times that ESP can hold for $\rho(W) > 1$ [21].

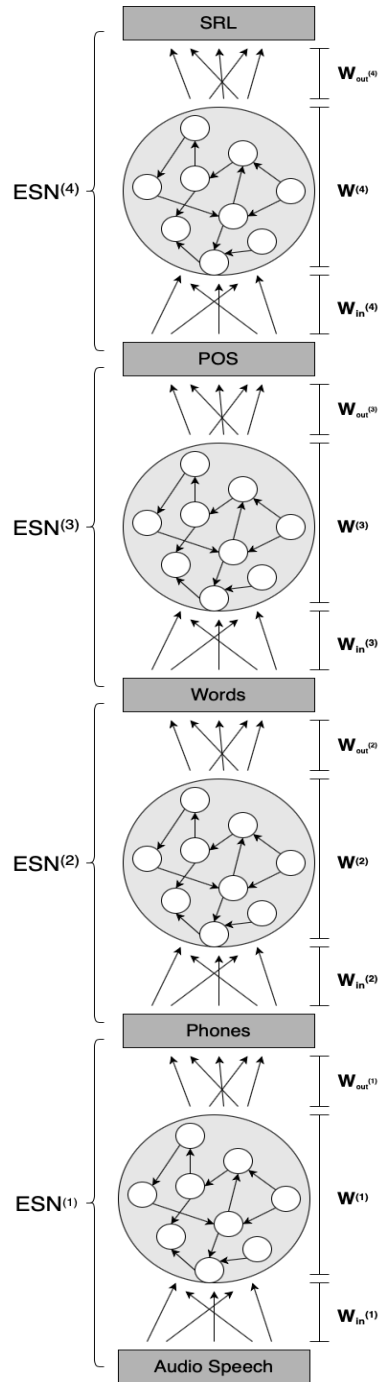


Figure 2.2: HTR architecture

Hierarchical-Task Reservoir

Before diving into the definition of the Hierarchical-Task Reservoir it is important to establish the main characteristics of a hierarchical ESN architecture. It consists of multiple connected ESNs. One of the more simple connection frameworks and the one that the Hierarchical-Task Reservoir uses is chained ESNs. Every consecutive ESN feeds its output as input to the next ESN layer.

The state update function for every layer is defined as:

$$x(t)^{(n)} = (1 - \alpha^{(n)})x(t-1)^{(n)} + \alpha^{(n)}f(W_{in}^{(n)}u(t)^{(n)} + W^{(n)}x(t-1)^{(n)}) \quad (2.7)$$

where n is the number of the layer. The external input $u(t)^n$ is defined as follows:

$$u(t)^{(n)} = \begin{cases} u(t) & \text{if } n = 1 \\ W_{out}^{(n-1)}x(t)^{(n-1)} & \text{otherwise} \end{cases} \quad (2.8)$$

The Hierarchical-Task Reservoir (HTR) is a hierarchical ESN model that is trained to perform the Semantic Role Labeling (SRL) task on audio speech. The goal of the SRL task is to label words in a sentence with the role they play with respect to the other sentence members, e.g. Agent or Predicate. The main aim behind the creation of the Hierarchical-Task Reservoir was increasing the amount of resemblance a model shares with the brain architecture. The most important and novel characteristic of this model is the infusion of abstraction into its structure. This abstraction mimics the processes found in brain. For example, the different areas of the visual cortex that extract visual features of various coarseness. HTR implements abstraction by splitting the SRL task into 4 sub-tasks: processing speech audio into phones (sound representations), phones into words, words into Part-of-Speech (POS) tags and, finally, POS tags into SRL labels. The model has 4 ESN layers, each performing one of the tasks (Figure 2.2). Extracting phones from audio is a more fine-grained task because phones are the smallest features of the recording. Meanwhile, extracting SRL labels from speech is a broader operation, since a single SRL label can take up a large part of a recording that contains several phones. Hence, the layers of HTR perform progressively more abstract tasks. In terms of ESN dynamics, this property manifests itself in the varying timescales of each layer. The lower layers have faster dynamics, because, as has been previously mentioned, phones are a smaller speech component and therefore occur with a higher frequency. Whereas SRLs are longer and as such occur with a lower frequency [16].

Chapter 3

Related Work

In [20] Venayagamoorthy and Shishir perform an investigation of the impact that the spectral radius and the settling time (ST) have on the network performance. Settling time is the number of times z that the reservoir state x is updated for given input u before the reservoir "settles" on the final state $x(z)$. In the paper, the performance of a singular "vanilla" ESN is measured on three tasks: approximating a function, predicting Mackey-Glass chaotic time series and monitoring complex systems. The ρ values were sampled from $[0.2, \dots, 0.8]$ with step 0.2 and ST values 1 and 5 were tested. The results of every task were that higher ρ and lower ST, i.e. $\rho = 0.8$ and $ST=1$, yield the best accuracy. The article looked only at a small partition of parameters, but tested their effect on network performance on a number of tasks of varying complexity. This makes the results of the experiment robust. The relevance of the research, however, is limited to tasks with low variability of timescales, because more complex tasks, like language processing were not discussed. It is also under question whether the results of the analysis are applicable to LI-ESNs, because the leaky integrator has an additional effect on the reservoir timescales and there may be a trade-off between optimal leaky integrator and spectral radius.

In [9] the properties of singular LI-ESNs with output feedback connections are analyzed through the process of parameter optimization. The global parameters $p \in \{\alpha, \rho, \sigma, s^{fb}\}$, where s^{fb} is the scaling coefficient for feedback weights, as well as the output weight matrix W_{out} and state noise s^v are tuned. To update the parameters, the authors use a self-designed gradient-descent-based approach. In comparison to the current report, the article puts a larger emphasis on the effect that different magnitudes of W_{out} and the amount of state noise have on the model performance. The authors observe that some optimal combinations of ρ and α emerge for W_{out} with high magnitude, which, as they point out, is an undesired outcome. Large weights can cause unstable predictive behavior, because every small change in the reservoir state will have a big effect on the final output. To avoid this unwanted outcome and stabilize the model, various amounts of state noise

are used as a way of regularizing the weights. As a result, systems with more noise yield better performance. It is also worth mentioning that during optimization the local minima for the error function have been observed to coincide with $\alpha < 1$, proving that adding the leaky integrator to a classic ESN can improve performance.

In [15], the dynamic characteristics of hierarchical two-layered LI-ESNs are manipulated through the tuning of α and ρ . The paper puts its focus on the effect that timescale differentiation between model layers has on the error rate. It notes that hierarchical models typically develop faster dynamics on earlier layers, which act as high-dimensional transformation layers for the input data. The higher layers have slower dynamics, so they perform more of an input aggregating function. The slowing of dynamics also reflects the need for larger memory in order to successfully perform the task. The amount of memory required is usually dictated by the task at hand. The paper uses the NARMA10 task, where the model needs to output a time-series function given a random number in the range $[0, 0.5]$ as input, and a similar NARMA5 task, where the target time-series output has shorter timescales. When comparing the optimal α values for the two tasks, it is observed that α of the second model layer is larger for NARMA5, making the dynamics of that layer faster and, in that way, showing that α correlates to the timescales required by the task.

Gallichio et al. studied the characteristics of Deep ESNs in [3]. They also paid special attention to α and ρ and their impact on the model dynamics. The value of α was manually varied, while ρ was kept fixed, to assess the individual effect of the leakage rate on layer dynamics. The experiment showed that having progressively smaller α when going up the layer hierarchy further increased the timescale length of consecutive layers. The same experiment was repeated for ρ . Increasing ρ proved to have the same effect as decreasing α .

It is important to note that all of the above mentioned works, except for [20], tune the spectral radius with respect to the Echo State Property, limiting it to a maximal value of 1. This thesis considers ρ values above 1, following the statement in [11] that system stability can still be achieved with $\rho > 1$. Another major aspect of this research that makes it stand out is that it is using a hierarchical ESN with task abstraction whereas all of the above mentioned papers concern "vanilla" DeepESNs. Finally, [15] and [3] pay more attention to the exploration of dynamical properties. While this report acknowledges those properties, the main focus is on the organization of hyperparameters and their contribution to model accuracy.

Chapter 4

Research

Experiment design

The goal of this experiment is to obtain optimal model parameters and discover how different parameters affect the error rate. It also aims at finding out whether the task abstraction has a positive effect on the error rate, in addition to using optimal parameters.

The model is trained on a preprocessed dataset, compiled by Hinaut et al. in [16]. As a base for the dataset the TIMIT corpus [4], containing acoustic speech data, was used. The size of the dataset is 6300 data points. Each point represents a spoken sentence. The sentences were obtained from 630 American speakers that pronounced 10 sentences each. The audio speech signal was processed every 10 ms with the Mel Frequency Cepstral Coefficients algorithm to obtain a representation vector of 39 components for each time frame. The speech was originally labeled every 10 ms by 61 phone and 6012 word classes. As a preprocessing step the number of phone classes was reduced to 51 by merging similar sounds into one label. The number of word classes was reduced to the 50 most frequent ones and the remaining classes were aggregated as a single label "out-of-vocabulary" (OOV). The POS labeling data was not part of the corpus and was obtained by computing POS tags for every word in each sentence. The total number of POS labels is 17. There is an additional label "h" for frames of silence in the speech audio. Each POS label is associated to the frames that fall under the duration of its corresponding word [16].

The SRL task is omitted for the purposes of this experiment, because of the high complexity and large amount of hardware resources and time it would take to train the model for this task.

Also due to hardware limitations, the data size is scaled down to $\frac{1}{5}$ of the original size (1260 data points). This data is split into a training and a testing set with ratio 7:3 (882 and 378 points respectively). The training set is divided into training and validation sets with ratio 7:3 (617 and 265

respectively). For the model optimization the training and validation sets are used. To evaluate the performance of the model after the parameters have been tuned the model is fitted with the whole training set and tested on the test set.

The model is created in Python using the ReservoirPy library [19]. As has been previously mentioned, the SRL task is excluded from the experiment, so the fourth SRL-oriented ESN layer is removed. Thus the model only has three ESN layers, the last one being the POS tagging one. For efficiency purposes, the layers were made with the library's ESN node, rather than manually from Input, Reservoir and Output nodes, because ESN has a parallelization functionality and is therefore faster to train.

The model optimization procedure recreates the one used for the Hierarchical-Task Reservoir in order to have the best comparison baseline. The selected optimization algorithm is the random search approach. This algorithm randomly selects parameter values from the defined search space until the maximum number of evaluations is reached. The number of evaluations for random search is 100 and the number of random models initialized in each evaluation is set to 5. In [16] it was not specified how many neurons did the model layers have during optimization, but in [5] it was suggested to select the minimal possible number of neurons in a reservoir for the random search, in order to reduce the execution time. Following that advice, the selected number of neurons is 100. While only 100 neurons are used to optimize the parameters, for evaluating the model's performance on the testing set 1000 neurons are used as in [16].

The measure of prediction accuracy and the objective function for random search is the Frame Error Rate (FER). The FER value is computed as the ratio of incorrectly predicted frames to the total number of frames in the input sequence.

Optimization steps for each model layer:

1. Tune parameters α and ρ , then σ and lastly λ
2. Train the layer on the whole training sample with obtained parameters
3. Generate layer predictions on the training and testing set
4. Use the training and testing predictions of the current layer as input data for the next layer

Hyperparameter tuning algorithm:

1. Tune α and ρ with random search using the value ranges in Table 4.1. Fix $\sigma = 1$ to avoid interdependencies between parameters. The λ search space was set to loguniform $[10^{-8}, 1]$. The value of λ that is obtained during this round of random search is disregarded following advice from [5], because the number of reservoir units used for random search does not match the number of units in the actual model.

2. Tune σ with random search using the value range in Table 4.1. The search space for λ remains loguniform $[10^{-8}, 1]$. The values of ρ and α are fixed with the best values obtained in step 1. The best values selected are the ones that correspond to minimal loss (FER).
3. Fix the values of ρ , α and σ with those found in steps 1 and 2. Tune λ on the 1000 neuron model with exhaustive search in the range shown in Table 4.1.

The tuning algorithm is a cost-efficient variation of the algorithm suggested in [5]. To perform a single random search a function 'research' from the ReservoirPy library was used. That function utilizes the Hyperopt parameter optimization library [1].

parameter	value sample space
spectral radius ρ	logarithmic distribution $[0.1, 10]$
leaky integrator α	uniform distribution $[0.1, 1]$
input norm σ	logarithmic distribution $[0.1, 10]$
ridge regularization λ	uniform distribution $[10^{-8}, 10^{-7}, \dots, 10^0]$

Table 4.1: Hyperparameter value ranges

Results

Evaluating the success of optimization

Firstly, the parameters obtained after optimization are compared to the original Hierarchical-Task Reservoir parameters provided in [16] (Table 4.2). A clear pattern can be observed in the HTR parameters: the spectral radius and the leaky integrator progressively increase for higher layers of the model. The α grows at an approximately constant pace of 0.2, while ρ increases rapidly by 0.3 for the second layer and then continues to grow by 0.05 for the third layer. No such pattern is observed for the optimized parameters. Instead, a different pattern can be seen: the parameters of the second layer are all smaller than those of the first and third layers. It should also be said that the values of the optimized $\alpha^{(1)}$ and $\alpha^{(3)}$ both deviate from the corresponding $\alpha^{(1)}$ and $\alpha^{(3)}$ of the original HTR by 0.1.

Secondly, the performance of each individual layer and the model as a whole is compared between: hyperparameter values obtained during optimization, hyperparameter values presented in [16], randomly initialized hyperparameters (Table A.1) and uniform parameters for each layer. The random hyperparameter values were sampled from the distributions shown

in Table 4.1. For the uniform parameter case the values are $\alpha = \sigma = 0.5$, $\rho = 0.9$ and $\lambda = 0.001$ for every layer.

		Optimized	HTR
Layer1	$\rho^{(1)}$	2.3865	0.8616
	$\alpha^{(1)}$	0.2084	0.2961
	$\sigma^{(1)}$	0.1113	0.7210
	$\lambda^{(1)}$	10^{-2}	$6.56 \cdot 10^{-3}$
Layer2	$\rho^{(2)}$	0.5671	1.1786
	$\alpha^{(2)}$	0.1288	0.4663
	$\sigma^{(2)}$	1.0137	8.8463
	$\lambda^{(2)}$	10^{-7}	$4.59 \cdot 10^{-5}$
Layer3	$\rho^{(3)}$	0.9293	1.2298
	$\alpha^{(3)}$	0.7673	0.6632
	$\sigma^{(3)}$	0.8596	2.6722
	$\lambda^{(3)}$	10^{-1}	$1.49 \cdot 10^{-5}$

Table 4.2: Hyperparameters obtained from optimization vs in HTR [16]

	FER		
	Layer 1	Layer 2	Layer 3
Optimized	58.17%	26.86%	52.35%
HTR	58.78%	29.8%	52.83%
Random	72.35%	29.99%	56.56%
Uniform	60.85%	30.47%	53.55%

Table 4.3: Frame error rates for different parameter configurations

It can be observed that the optimized parameters yield the lowest FER for every layer. The error rates of the model when it uses the optimized parameters versus when it uses the same parameters as the original HTR are very close for the first and third layers. Another important characteristic is that despite the rather big difference between the optimized parameters and the random and uniform parameters the error of the final third layer for each of the models is in the same range between 50 and 60 percent.

Lastly, as an additional measure of accuracy, the ratio between timescales, $\tau^{(i)} = \frac{1}{\lambda^{(i)}}$, of the model layers i is plotted. In this context, a timescale of a layer is the duration of the token that the layer is predicting (phone, word, POS tag). The timescales describe the speed of dynamics of a network, so

knowing the ratios of the timescales between layers shows how the dynamics of the model change. The ratio is computed for all 4 parameter configurations and compared to the ratios found in the original dataset. In this section only the temporal ratios for original data and for optimized parameters are shown. The remainder of plots for other parameters can be found in Appendix A.

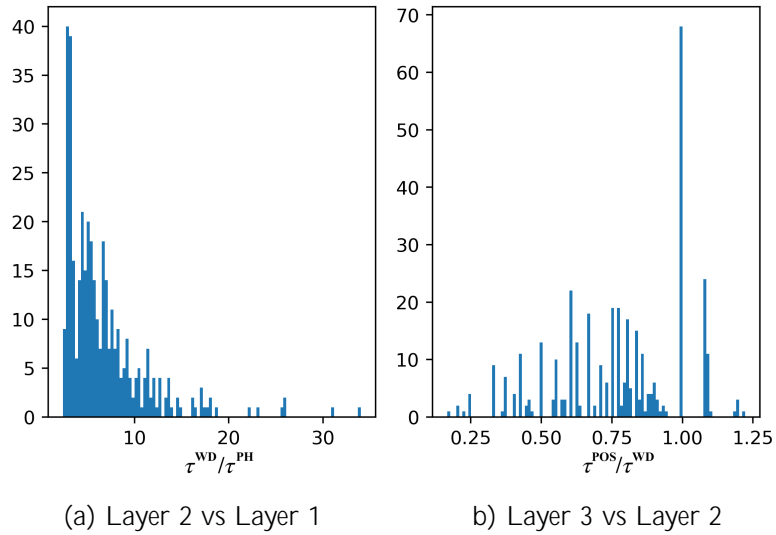


Figure 4.1: Temporal ratios of original data. Labels: PH - phone, WD - word, POS - Part-of-Speech tag.

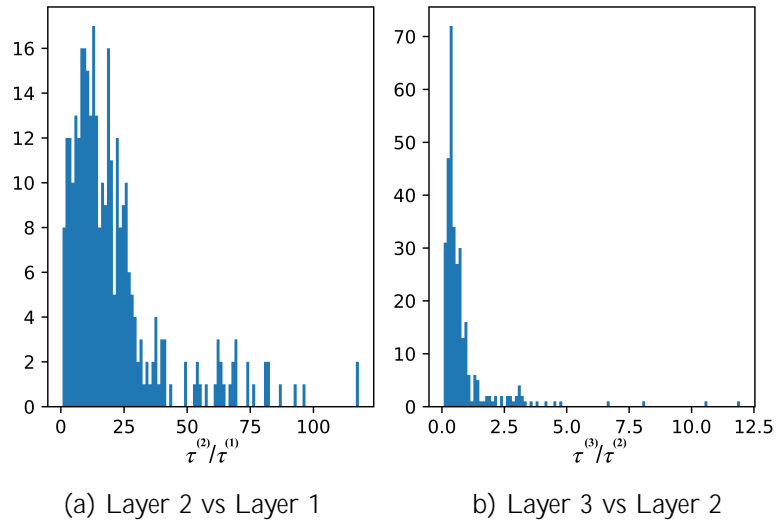


Figure 4.2: Temporal ratios computed with optimized parameters

In Figure 4.1 are the ratios found in the dataset that was used to train and test the model performance. The ratio between the second and first layers is concentrated around 2. By definition the durations of tokens of the second layer - words, are longer than those on the first layer - phones. The ratio between the third and second layers is concentrated around 1, meaning the number of frames taken by the POS tags and words is approximately the same.

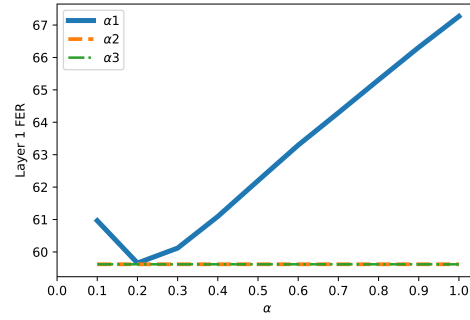
In Figure 4.2 are the ratios of the data predicted by the model with the optimized parameters. The ratio between the second and first layers is concentrated around 10. The ratio between the third and second layers is concentrated around 0.5.

Evaluating individual hyperparameter importance

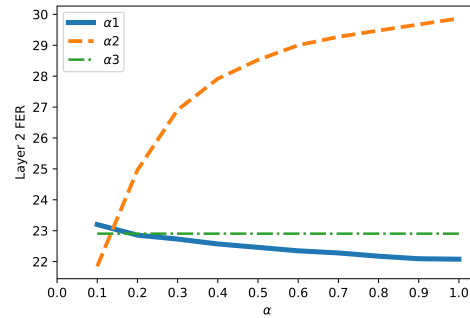
This section of the experiment looks at the effect individual parameters have on the error rate. The primary objective is to discover which parameters contribute the most to the model's performance. What is also tested is the optimality of the tuned parameters. The parameter values were varied within the bounds of the search spaces in Table 4.1. The spectral radius and input scaling were tried in the range from 0.1

to 1 with a stepsize of 0.1 and in the range from 1 to 10 with a stepsize of 1. The range for the leaky integrator was from 0.1 to 1 with a stepsize of 0.1.

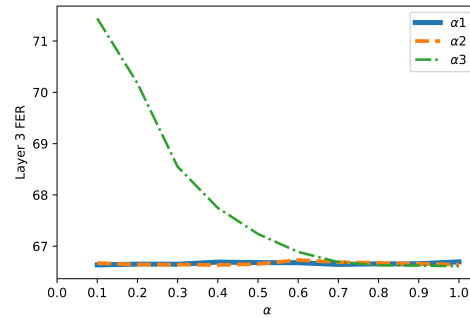
In Figure 4.3 the effect of α on the FER of different layers is presented. What can be observed is that increasing $\alpha^{(1)}$ corresponds to the increasing of the first layer's FER and decreasing of second layer's FER. This creates a trade-off between choosing to have better precision for layer 1 or 2. Additionally, larger values of $\alpha^{(2)}$ correspond to larger FER of layer 2. Increasing



(a) Effect of $\alpha^{(n)}$ on FER of Layer 1



(b) Effect of $\alpha^{(n)}$ on FER of Layer 2



(c) Effect of $\alpha^{(n)}$ on FER of Layer 3

Figure 4.3: Effect of α of FER

$\alpha^{(3)}$ decreases the FER of layer 3, while $\alpha^{(1)}$ and $\alpha^{(2)}$ have no strong impact on the accuracy of layer 3. Judging from the graphs the optimal values are $\alpha^{(1)} = 0.2$, $\alpha^{(2)} = 0.1$ and $\alpha^{(3)} = 1$.

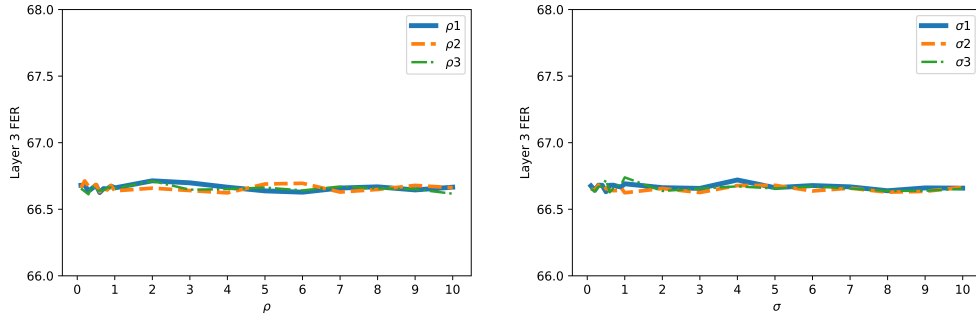


Figure 4.4: Effect of ρ and σ on FER of Layer 3

Varying ρ and σ has shown to have impact only on the third layer's performance. There is no particular pattern to the effect that ρ and σ have on the FER. But one characteristic that ρ and σ share is that their best values are located between 0 and 1. The FER for varying ρ and σ for the first and second layer remained constant and the plots are uninformative, therefore they are not included in this section and can be found in Appendix A instead.

Chapter 5

Discussion & Conclusions

The optimization procedure described in the Results section has successfully produced a set of optimal parameters. From Table 4.3 it can be seen that the model performs best when using the obtained tuned parameters compared to the original HTR, random and uniform parameters. Additionally, exploring the effect of individual parameters on the frame error rate has shown that the optimized α values indeed yield some of the lowest errors.

In [3] it was concluded that the dynamics in hierarchical ESNs get slower for higher layers and that increasing ρ and decreasing α yields slower dynamics. In the original HTR parameters there is a pattern of ρ increasing for higher layers and it could be speculated that the pattern was used to stimulate the slowing of dynamics. The α values, however, also showed an increasing pattern, which means they increase the speed of the layer dynamics. So ρ and α have contradicting effects. This makes the purposefulness of the observed patterns questionable. What makes those patterns even more arbitrary is the fact that the model performance with the optimized parameters, which show no such patterns, was better. Finally, running the model on random and uniform parameters, while rising the error rate, does not worsen the performance by a marginally large percent. These arguments erase the seeming meaningfulness behind the patterns of the HTR parameters, in the context of the current research.

Although, it must be said that the patterns found in the original HTR parameters should not be disregarded completely and rather require further investigation. In [16] those parameters referred to a slightly modified version of the HTR, with a skip-connection from the second to the fourth layer and a word embedding encoding used for the word data instead of one-hot vector encoding. Perhaps the patterns have more purpose with regard to the skip-connection and how they affect the accuracy of the final fourth layer. Then in this experiment their influence is not observed, or is very little, because the used model has only 3 layers.

Reasoning about the optimized parameters from the perspective of dynamics the following is discovered. It can be seen from Figure 4.2 that the

dynamics of the second layer are slower than those of the first layer, because the temporal ratios are above 1. The token durations of the third layer are less than or equal to those of the second layer, because the ratios are at most 1. From equation 2.6 and [3], increasing α increases the frequency of ESN dynamics, i.e. decreases the timescales τ . From the observed temporal ratios, it then follows, that layer 1 should have smaller τ and larger α than layer 2. With the same logic, τ of layer 2 should be more than or equal to that of layer 3 and $\alpha^{(2)} \geq \alpha^{(3)}$. Indeed, the optimized α values fit these characteristics (Table 4.2). It is also important to notice that layers 1 and 3 of the model with the original HTR parameters report FER very close to that of the first and third layer of the optimized model. Coincidentally, $\alpha^{(1)}$ and $\alpha^{(3)}$ of the HTR and of the optimally parametrized model are also very similar in value. Additionally, in the investigation of the impact that individual parameters have on the FER, it was concluded that α has the strongest influence on the error rate. Meanwhile, ρ and σ show no particular effect on the FER. Thus, α is HTR's most important parameter, which not only modulates the model's FER, but also provides information about the model's dynamics.

The analysis of the spectral radius and the input norm concluded that these parameters have little effect on the model's FER. This was a surprising result, knowing that ρ and σ are responsible for important dynamical features, namely the memory capacity and the non-linearity of dynamics. This leads to a future research proposition to carry out a more in-depth investigation of the role that ρ and σ in the Hierarchical-Task Reservoir. Despite the impact of ρ and σ on the FER being underwhelming, there is still one important observation to be made. In Figure 4.4 it can be seen that the parameter values corresponding to the lowest FER are all concentrated between 0 and 1. For the spectral radius this means that, in the context of the HTR, model stability is more likely to be granted when the Echo State Property holds.

The abstraction feature was discovered to modulate the model's dynamics. It is expected that higher layers of a model have slower dynamics. However, what is concluded from the analysis of the temporal ratios is that the third layer has faster dynamics than the second layer. This means that the timescales imposed by the sub-tasks used for abstraction overpower the more typical development of layer dynamics. Knowing that hyperparameter values and dynamics are closely related it can be said that through modulating the dynamics the abstraction feature also influences the hyperparameters.

It is also necessary to discuss the limitations of the research. As has been mentioned in Chapter 4, the size of the dataset that was used for the experiment has been greatly reduced to only 20% of its original size. During parameter optimization, the optimization algorithm aims at minimizing the objective function, in this case the FER. If parameter optimization

was performed with a larger dataset, the achieved errors could have been lower and it is possible that a different set of optimal parameters could have been obtained. Another flaw of the optimization process is that for validating the parameters holdout validation was used instead of cross-validation. This approach was chosen to make optimization more time-efficient. Cross-validation would have been a more suitable choice, because of how small the experiment's dataset is in comparison to the original dataset.

The findings and limitations of the experiment serve as inspiration for some possible future research topics. Firstly, because the model layers were optimized separately one-by-one, it would be interesting to investigate whether optimizing the model as a whole would yield different parameters. Secondly, since in this experiment only the first three layers of the HTR were investigated, the experiment should be repeated for the whole model on more powerful hardware and preferably using the whole dataset. Thirdly, a comparative analysis could be performed for different optimization algorithms. The analysis would test whether each of the algorithms arrives at the same set of optimal parameters. It will also help find a more robust set of optimal parameters. Lastly, in this report it was observed that the model performance with random parameters is not far worse than with the optimized or original HTR parameters. This leads to an assumption that the model accuracy is defined not only by individual parameter values, but also by the relations between parameter values. The interdependencies between model parameters should be explored more in-depth in future research.

The research results provide insight into how the Hierarchical-Task Reservoir parameters affect its performance. There are some discoveries that contradict expectations and require further investigation, like the spectral radius having little effect on the FER and the third layer having faster dynamics in comparison to the second layer. But there are also some promising findings, like the leaky integrator having the strongest effect on the model's accuracy and dynamics. In this way, the experiment lays a foundation for future research on the Hierarchical-Task Reservoir. Overall, the path to understanding the properties of the Hierarchical-Task Reservoir looks very promising.

Bibliography

- [1] James A. Bergstra, Daniel L. K. Yamins, and David Daniel Cox, *Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures*, 2013, <https://dl.acm.org/doi/10.5555/3042817.3042832>.
- [2] Claudio Gallicchio and Alessio Micheli, *Echo state property of deep reservoir computing networks*, 2017, <https://link.springer.com/article/10.1007/s12559-017-9461-9>.
- [3] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli, *Deep reservoir computing: A critical experimental analysis*, 2017, <https://www.sciencedirect.com/science/article/abs/pii/S0925231217307567>.
- [4] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren, and Victor Zue, *Timit acoustic-phonetic continuous speech corpus*, 1993, <https://catalog.ldc.upenn.edu/LDC93s1>.
- [5] Xavier Hinaut and Nathan Trouvain, *Which hype for my new task? hints and random search for reservoir computing hyperparameters*, 2021, <https://hal.inria.fr/hal-03203318/>.
- [6] Herbert Jaeger, *Discovering multiscale dynamical features with hierarchical echo state networks*, 2007, <https://opus.jacobs-universitaet.de/frontdoor/index/index/docId/636>.
- [7] _____, *Echo state network*, 2007, http://www.scholarpedia.org/article/Echo_state_network.
- [8] _____, *Long short-term memory in echo state networks: Details of a simulation study*, 2012, <https://opus.jacobs-universitaet.de/frontdoor/index/index/docId/638>.
- [9] Herbert Jaeger, Mantas Lukosevicius, Dan Popovici, and Udo Siewert, *Optimization and applications of echo state networks with leaky-integrator neurons*, 2007, <https://www.sciencedirect.com/science/article/abs/pii/S089360800700041X>.

- [10] Alexis Juven and Xavier Hinaut, *Cross-situational learning with reservoir computing for language acquisition modelling*, 2020, <https://hal.inria.fr/hal-02594725>.
- [11] Mantas Lukosevicius, *A practical guide to applying echo state networks*, 2012, https://link.springer.com/chapter/10.1007/978-3-642-35289-8_36.
- [12] Mantas Lukosevicius and Herbert Jaeger, *Reservoir computing approaches to recurrent neural network training*, 2009, <https://www.sciencedirect.com/science/article/abs/pii/S1574013709000173>.
- [13] Mantas Lukosevicius, Herbert Jaeger, and Benjamin Schrauwen, *Reservoir computing trends*, 2012, <https://link.springer.com/article/10.1007/s13218-012-0204-5>.
- [14] Wolfgang Maass, Thomas Natschläger, and Henry Markram, *Real-time computing without stable states: A new framework for neural computation based on perturbations*, 2002, <https://ieeexplore.ieee.org/abstract/document/6789852>.
- [15] Luca Manneschi, Matthew O. A. Ellis, Guido Gigante, Andrew C. Lin, Paolo Del Giudice, and Eleni Vasilaki, *Exploiting multiple timescales in hierarchical echo state networks*, 2021, <https://arxiv.org/abs/2101.04223>.
- [16] Luca Pedrelli and Xavier Hinaut, *Hierarchical-task reservoir for online semantic analysis from continuous speech*, 2021, <https://hal.inria.fr/hal-03031413v3>.
- [17] Anthony Strock, Nicolas Rougier, and Xavier Hinaut, *Latent space exploration and functionalization of a gated working memory model using conceptors*, 2020, <https://hal.archives-ouvertes.fr/hal-02494493?lang=en>.
- [18] Fabian Triefenbach, Azarakhsh Jalalvand, Kris Demuyne, and Jean-Pierre Martens, *Acoustic modeling with hierarchical reservoirs*, 2013, <https://ieeexplore.ieee.org/document/6587732>.
- [19] Nathan Trouvain, Luca Pedrelli, Thanh Trung Dinh, and Xavier Hinaut, *Reservoirpy: an efficient and user-friendly library to design echo state networks*, 2020, <https://hal.inria.fr/hal-02595026>.
- [20] Ganesh K. Venayagamoorthy and Bashyal Shishir, *Effects of spectral radius and settling time in the performance of echo state networks*, 2009, <https://pubmed.ncbi.nlm.nih.gov/19423285/>.

- [21] Izzet B. Yildiza, Herbert Jaeger, and Stefan J. Kiebel, *Re-visiting the echo state property*, 2012, <https://pubmed.ncbi.nlm.nih.gov/22885243/>.

Appendix A

Additional Research Results

Temporal ratios between model layers

Here, the temporal ratios of the remaining parameter configurations, not discussed in the main Results section, are presented.

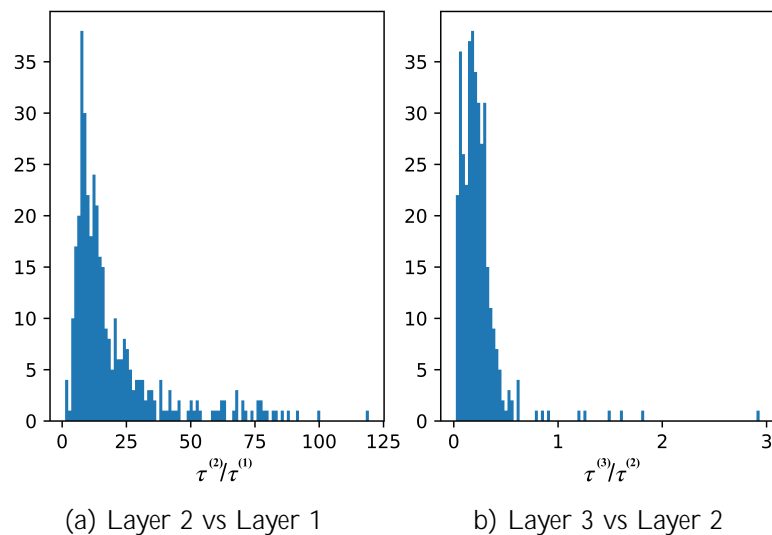


Figure A.1: Temporal ratios computed with the original HTR parameters

In Figure A.1 are the ratios of the data predicted by the model with the original HTR parameters. The ratio between the second and first layers is concentrated around 10, just as for the optimized parameters. The ratio between the third and second layers is concentrated around 0.3.

In Figure A.2 are the ratios of the data predicted by the model with random parameters. The ratio between the second and first layers is concentrated around 5. The ratio between the third and second layers is concentrated around 1.

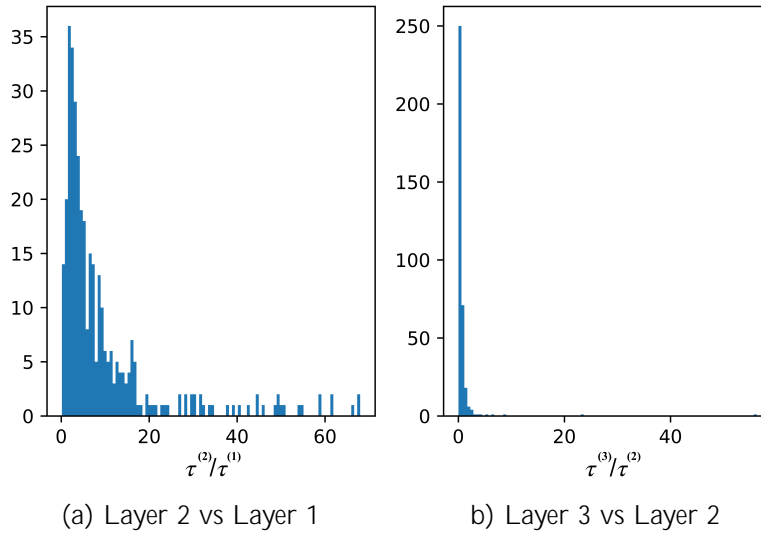


Figure A.2: Temporal ratios computed with random parameters

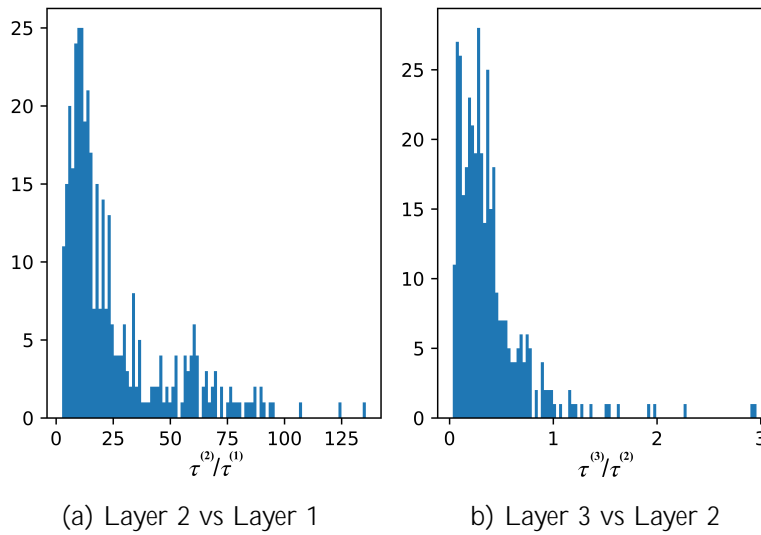
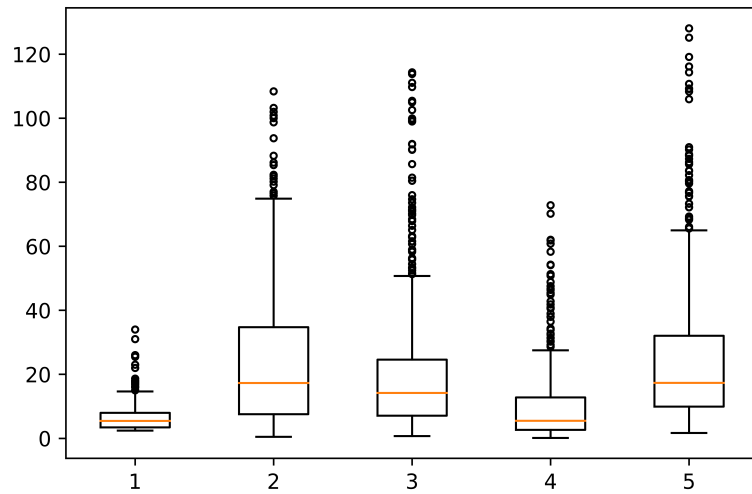


Figure A.3: Temporal ratios computed with uniform parameters

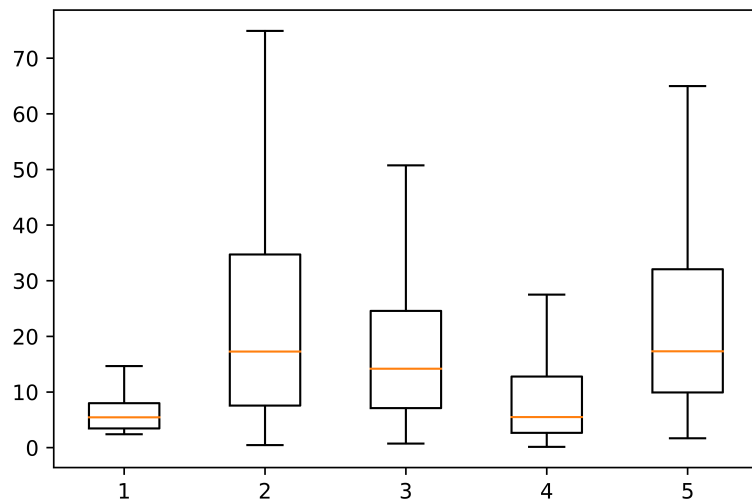
In Figure A.3 are the ratios of the data predicted by the model with uniform parameters. The ratio between the second and first layers is concentrated around 12. The ratio between the third and second layers is concentrated around 0.3.

In Figure A.4 and Figure A.5 additional illustrations of the temporal ratios are provided in the form of boxplots. The plots display the minimum

and maximum, median and first and third quartiles of the temporal ratio distribution.

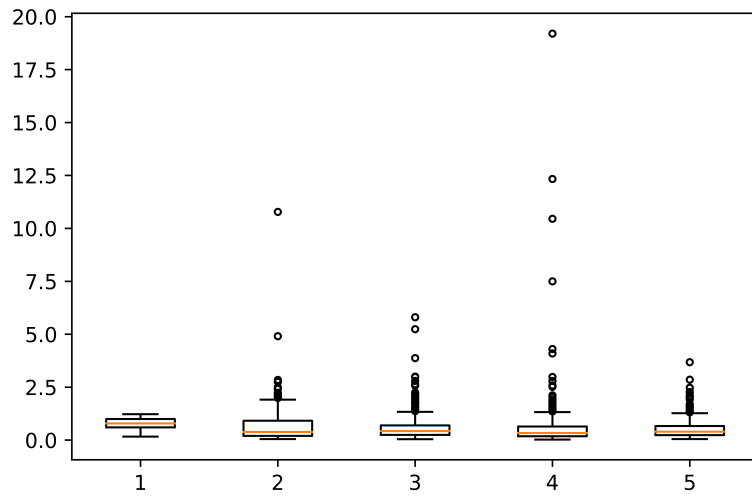


(a) Temporal ratio distributions with outliers

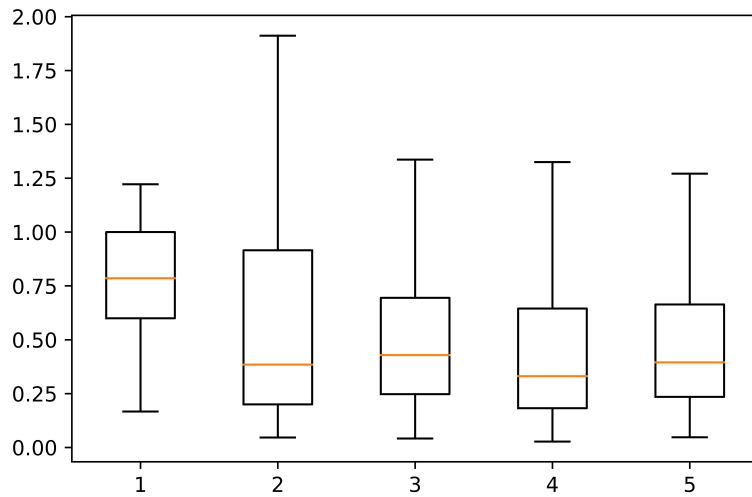


(b) Temporal ratio distributions without outliers

Figure A.4: Temporal ratio distributions between layers 2 and 1. Labels: 1 - original dataset, 2 - model with optimal parameters, 3 - model with original HTR parameters, 4 - model with random parameters, 5 - model with uniform parameters



(a) Temporal ratio distributions with outliers



(b) Temporal ratio distributions without outliers

Figure A.5: Temporal ratio distributions between layers 3 and 2. Labels: 1 - original dataset, 2 - model with optimal parameters, 3 - model with original HTR parameters, 4 - model with random parameters, 5 - model with uniform parameters

Overall, none of the parameter configurations do a good job of recreating the original data's temporal distributions. The ratios between layer 2 and layer 1 are overestimated and ratios between layer 3 and layer 2 are underestimated. Though, interestingly, the random parameters give the closest approximation of the ratios between the second and first layers.

Influence of individual parameters

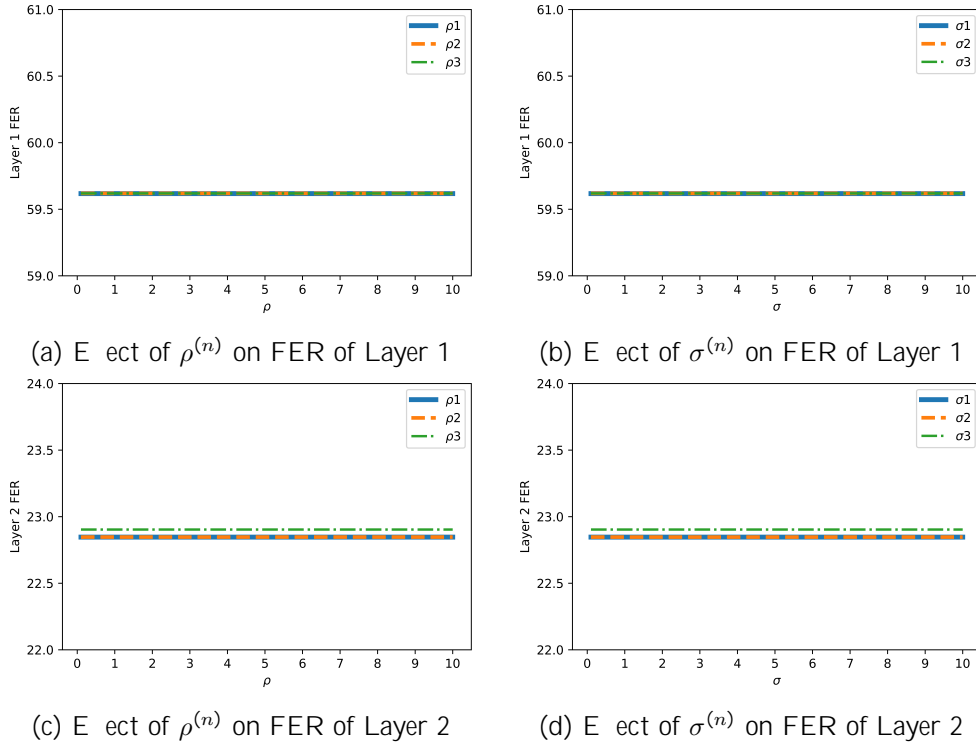


Figure A.6: Effect of ρ and σ on FER of layers 1 and 2

In Figure A.6 are the effects that the spectral radius and input norm have on the first and second layers of the model. It can be seen that the FER stays constant for any ρ or σ value.

Experiment resources

	ρ	α	σ	λ
Layer 1	0.1124	0.0137	0.3317	$2 \cdot 10^{-4}$
Layer 2	0.1169	0.7245	6.0311	$1.33 \cdot 10^{-2}$
Layer 3	0.3444	0.8834	0.1163	$1.45 \cdot 10^{-8}$

Table A.1: Random parameters used for model evaluation

Link to the GitHub repository with the code used for the experiment:
https://github.com/Zamyrova/htr_optimization