Multitasking in RNN: an analysis exploring the combination of simple tasks

Cecilia Jarne

Universidad Nacional de Quilmes, Departamento de Ciencia y Tecnología-CONICET, Argentina

E-mail: cecilia.jarne@unq.edu.ar

Keywords: recurrent neural networks, dynamics, multitasking

Abstract

The brain and artificial neural networks are capable of performing multiple tasks. The mechanisms through which simultaneous tasks are performed by the same set of units in the brain are not yet entirely clear. Such systems can be modular or mixed selective through some variables such as sensory stimulus. Recurrent neural networks can help to a better understanding of those mechanisms. Based on simple tasks studied previously in Jarne 2020 arXiv Preprint 2005.13074, multitasking networks were trained and analyzed. In present work, a simple model that can perform multiple tasks using a contextual signal was studied, trying to illuminate mechanisms similar to those that could occur in biological brains. Backpropagation through time allows training networks with multitasking, but the realizations obtained are not unique. Different realizations for the same set of tasks are possible. Here the analysis of the dynamics and emergent behavior of their units is presented. The goal is to try to describe better the models used to describe different processes in the cortex.

1. Introduction

A task can be defined as the set of computations that a system needs to perform to optimize an objective [2]. Biological brains are capable of performing multiple tasks of different kinds, as well as artificial neural networks can be trained to perform them too [3]. Neural mechanisms through which simultaneous tasks are performed by the same set of units are not entirely clear yet. On one hand, such systems can be modular, on the other, they can be mixed selective through some variable such as sensory stimulus.

Some of the difficulties to experimentally study this field are that laboratory animals are limited to be training on one task at a time. This happens due to the complexity that involves the training. Also, there exists a problem related to the resolution measurements to obtain multitasking neural recordings.

Mainly, for these reasons, the contribution that can be made from the recurrent neural networks (RNNs) to understand task representation at the neural level in the prefrontal cortex is relevant.

RNN is a general framework for theoretical investigation that is related to temporal tasks. Models are used to describe large sets of experimental results observed in different studies mainly related to the cortex and prefrontal cortex [4], or memory [5].

Based on the tasks studied in [1], which consisted of the processing of temporal stimuli with Boolean-like decision making, a simple network that can perform multiple tasks using a context signal was trained and analyzed.

This study focus on networks trained for combining the following tasks related to the processing of stimuli as temporal inputs:

(a) Time reproduction and finite-duration oscillation (one input tasks, with a context signal).
(b) Basic gate operation: AND, OR, XOR (two input tasks, with context signal).
They are relevant in the context of information processing and flow control of information. They are cognitive-type and decision-making tasks. They were chosen to be simple because allow untangling different characteristics of the dynamics previously observed, but now in the multitask paradigm.

The motivation in the selection of these tasks is double. First, to simulate flow control processes that can occur in the cortex when receiving stimuli from subcortical areas [6]. Second, these tasks are the basic and the lowest level for computing in any digital system. In the case of the flip flop, for instance, it is the simplest sequential system that one can build [7].

It has been previously proposed that some sets of neurons in the brain could roughly function as gates [6]. On the other hand, it also is interesting in itself the dynamics of trained networks for the flip flop task, which is generally related to the concept of working memory. It has been previously studied in [8, 9], but in this case with a more complex task referring to a 3-bit register called in the paper a 3-bit flip flop.

Each network was trained to perform a set of tasks. In case (i) to switch between tasks depending on a contextual bi-state signal or the set of tasks. In (ii) depending on a three-state context signal.

The main idea is to use the framework of neural population dynamics [10] to identify the rich structure within the coordinated activity of interconnected neural populations of trained RNN in the multitasking problem.

The rest of the paper is organized as follows. Model and methods are presented in section 2. In section 3 results are presented: the dynamics of the networks, and different aspects that arise from the realizations also are discussed. Finally, in section 4, the remarks and further directions are presented.

2. Model and methods

The dynamics of the interconnected units of the RNN model is described by equation (1) [11], where units have index \( i \), with \( i = 1, 2 \ldots n \).

\[
\frac{dh_i(t)}{dt} = \frac{-h_i(t)}{\tau} + \sigma \left( \sum_{j} W_{ij}^{\text{Rec}} h_j(t) + \sum_{j} W_{ij}^{\text{In}} x_j \right).
\]

In equation (1), \( h_i(t) \) is the activity of the units. \( \tau \) represents the time constant of the system. \( \sigma \) is a non-linear activation function. \( x_j \) are the components of the vector \( \mathbf{X} \) of the input signal. The matrix elements \( W_{ij}^{\text{Rec}} \) are the synaptic connection strengths of the matrix \( \mathbf{W}^{\text{Rec}} \) and \( W_{ij}^{\text{In}} \) the matrix elements of \( \mathbf{W}^{\text{In}} \) from the input units. Network is fully connected, and matrices have recurrent weights initially given from a normal distribution with zero mean and variance \( \frac{1}{N} \).

The network has three layers: one is the input, the second is the recurrent hidden layer, and the last is the output layer. The readout output in terms of the matrix elements \( W_{ij}^{\text{Out}} \) from \( \mathbf{W}^{\text{Out}} \) is described by equation (2).

\[
\mathbf{Z}(t) = \sum_{j} W_{ij}^{\text{Out}} h_j(t).
\]

For this study it was considered \( \sigma() = \tanh() \) and \( \tau = 1 \), without loss of generality. Although other sigmoidal functions could be considered instead. Model is discretized through the Euler method. The value of the time step for the time evolution is \( \delta t = 1 \text{ mS} \). Then the dynamics of the discrete-time RNN can be approximated by 3.

\[
\mathbf{H}(t+\delta t) = \mathbf{H}(t) + (-\mathbf{H}(t) + \sigma(\mathbf{W}^{\text{Rec}}\mathbf{H}(t) + \mathbf{W}^{\text{In}}\mathbf{X}(t)))\delta t.
\]

From equation (3) which is written in vector form, the activity of the recurrent units at the next time step can be written as:

\[
\mathbf{H}(t+1) = \sigma(\mathbf{W}^{\text{Rec}}\mathbf{H}(t) + \mathbf{W}^{\text{In}}\mathbf{X}(t)).
\]

During training, different configurations for parameter fixing or training are possible. For example, the committee machine model [12, 13], where the elements \( W_{ij}^{\text{Out}} \) are all equal to 1. Another alternative is to train only the output weights, which is known as the reservoir computing paradigm with liquid- or echo-state networks [14]. Such cases also will result in a trained network in the considered tasks.

The motivation for all the parameters to be trainable, and not to use more specific configurations such as those presented in the citations comes from specifically exploring the models used in works as [15–18], where the model presented here is used. In this work, the final configurations that are obtained are very close to the initial random configurations, so it is also interesting that some concepts and ideas associated with random networks can be explored.
Figure 1. Model for multitasking set (i). In the training stage, the time series are entered into the network in 2 possible combinations constituting a set with more than 15 000 samples with noise. The training algorithm adjusts the weights, according to the target function, to obtain the trained matrices $W^{in}$ and $W^{Rec}$ of each one and produced the expected output, given the context.

For the network implementation and training, Keras libraries where used [19] and Tensorflow [20] as frameworks, instead of more traditional choices like Matlab [21]. This allows making use of all current algorithms and optimization strategies developed and maintained by a large research community. The reason for this selection is that these new scientific libraries are open source, and their use is rapidly growing.

Supervised learning was the paradigm used with backpropagation through time implementing an adaptive SGD training method provided by Keras framework. Initial recurrent weights were obtained with a random normal distribution with the orthogonal condition on the matrix. For training, noisy square pulse signals were used at the inputs and context. Several training samples were used (> 15 000). The target output was simulated with a time delay answer of 20 mS (figure 1).

Based on the study of the previously described tasks, neuronal networks were trained to perform multitasking depending on the selection made through a contextual input signal. First, it was considered two cognitive-type tasks, with one input and one context signal. That is shown schematically in figure 2. If the context signal is a temporal stimulus with amplitude 1, the output will oscillate during a certain time, only if it receives a stimulus in the input channel. Otherwise, the output remains at the zero level. If the contextual signal has an amplitude equal to $-1$, the network performs OR computation with the stimuli at inputs A and B. If the contextual signal has an amplitude equal to 1, the network performs AND computation with the input stimuli. Finally, if there is no pulse as contextual input, the network performs XOR (exclusive or) computation.

The successfully trained networks were analyzed by stimulating inputs using a noise-free testing set, corresponding to the different possible stimuli combinations at the input, by plotting the activity of some $k$ units was as a function of time ($h_k(t)$).

Dimensional reduction methods are used to study the inner state of the network often [22]. In particular, SVD is the method that was chosen in the present work because it has been widely used in the study of simulations, as well as experimental high dimensional neural space states [23]. A decomposition into singular values was performed with the entire set of the output’s units $h_i(t)$. The behavior of the system was plotted into the three axes of greatest variance for all the different combinations of stimuli and context. An example is
Figure 2. Model for multitasking set (ii). In the training stage, the time series are entered into the network in the four possible combinations constituting a set with more than 15,000 samples with noise. The training algorithm adjusts the weights, according to the target function, to obtain the trained matrices $W_{\text{in}}$ and $W_{\text{Rec}}$ of each context, giving as a result the three different operations (XOR-OR-AND) represented in the figure with the gates' symbols.

Figure 3. Right: input signal, context signal (positive pulse), output, and the activity of 25 different neurons in the RNN. Left: SVD into the three axes of greatest variance corresponding to the activity of all units $h_i(t)$. Triangles represent the starting and ending time for the considered time series.

Presented in figure 3. The left panel shows the input signals, the context, and the output of the trained network along with the activity of 25 units. The right panel of figure 3 shows the singular value decomposition.

Also, for each network, the decomposition of $W_{\text{Rec}}$ in their eigenvectors and eigenvalues ($\lambda_i$) is obtained. An example of one network is presented in figure 4, where the distribution of the eigenvalues in the complex plane is presented. As expected it is observed that leading eigenvalues rules the dynamics.
3. Results and discussion

RNNs are used to understand neural computation, in terms of collective dynamics that is involved in motor control, temporal tasks, decision making, or working memory. It is essential to understand the dynamics behind trained RNN models because they are used to construct different hypotheses about the functioning of the brain areas and to explain the observed experimental results such in [24, 25].

In biological brains, at the computational level, performing a new task correctly, without training, requires composing elementary processes that are already learned, such as the process shown here. This property, called compositionality, has been proposed as a fundamental principle underlying flexible cognitive control [3].

In a neuronal circuit equipped with a compositional code, a new task might be represented as the algebraic sum of representations of the underlying elementary processes. Human studies have suggested that the representation of complex cognitive tasks in the lateral prefrontal cortex could be compositional.

Yang asks in [3] if for a network capable of performing many tasks, should it be clustered? Should its representation be compositional? Conceptually, the answer to either question can be yes or no, independently. A randomly connected network can potentially solve multiple tasks by mixing sensory stimuli and rule inputs in a high-dimensional space. Such a network will have no clustering and show no compositionality across tasks. The trained networks presented here are close to this random condition.

According to Yang a biological network, where different tasks are represented by completely non-overlapping populations, will show clustering but no compositionality. A network can be both clustered and compositional if common cognitive processes across tasks are represented by distinct clusters of neurons. When linearly mixing neuronal activity of a clustered compositional network, the resulting neural activity would still be compositional, but no longer clustered at the single-unit level.

To explore potential solutions, RNNs were trained in 3 simple tasks. It was found that, after training, the tasks simultaneously, the emerging task representations were not organized in the form of clustering of recurrent units under the conditions presented in section 3.

RNN was successfully trained using Keras and Tensorflow framework with context multitasking. All networks were trained with noise, corresponding to 10% of the amplitude in the inputs and the contextual inputs. However, to clearly show the behavior and trajectories obtained in the low-dimensional space corresponding to the three maximum variance axes, rectangular pulses without noise have been used.

The results presented in figures 5 and 6 corresponds to trained networks for multitasking with 200 fully connected units, although the range between 100 and 400 units was explored, not finding large differences in the results. Network’s units presented coherent activity after training.
Figure 5. Example of the response of a multitask RNN with different stimuli. Context signal is followed by a stimulus at the input with a delay of 50 mS. The response follows input after 20 mS. Upper right panel: input is zero, and the context signal is positive. Upper left panel: input signal is a square pulse, and the context signal is negative. Bottom panel: input signal is a squared pulse as well as the context signal.

As it was previously observed in [1], the following situations may occur:

(a) When the stimulus arrives, the $h_i(t)$ activities start a transitory regime that ends in a sustained oscillation, each with a different face and amplitude. The superposition is given by $W_{\text{out}}$ and allows to obtain the expected output.

(b) When the stimulus arrives, the $h_i(t)$ start a transitory regime that leads to a fixed level other than zero for each unit, and whose superposition, given by $W_{\text{out}}$, allows to obtain the output.
The $h_t(t)$, when the stimulus arrives, passes to a transitory regime that attenuates to zero, and the output is zero as a result of the attenuation of the $h_t(t)$. This situation is less often.

It was observed that training on a given task leads to different network implementations. Three main factors contribute to such variability. First is the determination of the network parameters that are trained: the number of units, nonlinear activation function, etc. A second factor is the task parametrization: for instance the considered time delay between context and input signal, noise level, cue time. Another factor is the stochastic nature of the training procedure. Initial conditions of trained parameters are obtained with a random number Gaussian distribution with the orthogonal condition. Different initializations give rise to different outcomes for trained weights that perform the tasks equally well. Here an example of each multitask network considered is shown and discussed.

First, let us consider the task set (i): the time reproduction of a finite-duration oscillation. In figure 5, it is possible to observe the different responses in each panel for the network trained for two tasks in the different contexts in the presence of a stimulus or the absence of it. It is interesting to observe that when the network receives a stimulus (corresponding to the context signal), but the input is zero (upper right panel), the output remains passivated. The units present oscillatory activity, which their superposition results in zero level output. The only case where the units show activity is when there is no stimulus or the input $y$ in the contextual input. Different network realizations show similar behavior.

Second, let us consider the task set (ii) with the binary basic operations. In figure 6, all the different cases for stimuli are shown for one particular trained network, although more realizations are available in the repository and additional realization can be generated using the code provided (see appendix A). In the case shown here, each of the three possible contexts and combinations of stimuli at the inputs is presented. A total of three contexts, with four different input signals. We can see that the only case where the activity of the units is zero is when the context signal is null, and there is no stimulus in the inputs.

It is interesting to observe that the same set of units presents activity in all contexts. On the one hand, there is oscillatory activity against certain stimuli, on the other, there is also fixed-point activity, as shown in the left bottom panel. Each trained network presented a similar response, but realizations are not unique for each task, as previously commented.

It also was found that there are no clusters of units that differentiate the tasks. They are distributed across the network activating all units in the network in different modes, depending on the stimuli at the inputs.
The $W_{\text{Rec}}$ matrices of trained networks are approximately normal when considering the orthogonal initial condition. They do not deviate much from the initial condition after training. They are enough not-normal so that there is a transient amplified effect that leads the system from the initial condition to the long term dynamics observed, which is consistent with estimations made in [26, 27].

The departure from the normal condition of the matrix can be estimated through the parameter Henrici’s departure from normality, obtained as in equation (5) [28].

$$d_F(W_{\text{Rec}}) = \sqrt{\left(\frac{\|W_{\text{Rec}}\|^2 - \sum_{i=1}^{N} |\lambda_i|^2}{\|W_{\text{Rec}}\|^2}\right)}.$$  

(5)

Where, for normalization, it was divided by the norm of the matrix. The number obtained for multitasking trained networks was of the same as for single task trained networks [1].

Another observation made is that more units are necessary to perform well than when considering one-task training, as it was expected. Nevertheless, still a small set of eigenvalues remain outside the circle after training and dominate dynamics [1]. Fix-point and oscillatory states coexist depending on context and input. Oscillatory state remains in a manifold [10].

4. Conclusions

Contextual multitasking systems are interesting mainly for two reasons, first because naturally, such processes occur continuously in different areas of the brain, and second because such processes are necessary for I A. Systems that may process real-time signals.

It was observed that multitasking training with the considered tasks produces a complex high-dimensional system that appears embedded in a low-dimensional system in a manifold, as previously observed for a single task. Backpropagation through time without any regularization term, allows networks to be trained to do the same multitask set, but not univocally. Different realizations for the same task were obtained that generate the same outcome (see appendix A).

On the other hand, the norm of the matrices trained for multi-tasking was estimated and compared with a single task, and the order is the same as previously reported.

Documenting the training process and parameters, as well as knowing the dynamics in these simple task examples, can help to better understanding of models that are used frequently to describe different processes in the cortex and prefrontal cortex.

Further work will include other statistical studies on weights matrices, other studies with excitatory and inhibitory networks, and more complex temporal tasks such as perceptual decision making task [29], parametric working memory task [30] or context-dependent decision-making task [4].

Acknowledgments

Present work was supported by CONICET and UNQ. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. I thanks mainly the researchers who encourage me to keep working in neural networks models. I also like to thank the two anonymous reviewers for their suggestions and comments. Finally, I want to thanks Gabriel Lio, whose support during the Covid-19 quarantined was fundamental.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/katejarne/multitasking_RNN.

Appendix A

Code, simulation, and additional trained networks of this analysis are available at the following Github repository:

https://github.com/katejarne/multitasking_RNN

It will be a public repository from the moment of publication of this article. Additional code is also available upon request.
ORCID iDs

Cecilia Jarne  
https://orcid.org/0000-0002-3917-920X

References

[1] Jarne C 2020 The dynamics of Recurrent Neural Networks trained for temporal tasks and the eigenvalue spectrum  
[arXiv: 2005.13074]
[2] Yang G R, Cole M W and Rajan K 2019 How to study the neural mechanisms of multiple tasks  
29 134–43
[3] Yang G R, Joglekar M R, Song H F, Newsome W T and Wang X-J 2019 Nat. Neurosci. 22 297–306
[4] Mante V, Sussillo D, Shenoy K V and Newsome W T Nature 503 78–84.
[5] Kurikawa T, Barak O and Kaneko K 2020 Phys. Rev. Res. 2 023307
[6] Gisiger T and Boukadoum M 2011 Front. Comput. Neurosci. 5 1
[7] Floyd T 2003 Digital Fundamentals (Englewood Cliffs, NJ: Prentice Hall) https://books.google.com.ar/books?id&tnqx3d;gjAfAQAAIAAJ
[8] Sussillo D and Barak O 2013 Neural Comput. 25 626–49
[9] Sussillo D 2014 Curr. Opin. Neurobiol. 25 156–63
[10] Vyas S, Golub M D, Sussillo D and Shenoy K V 2020 Annu. Rev. Neurosci. 43 249–75
[11] Hopfield J J 1984 Proc. Natl Acad. Sci. 81 3088–92
[12] Saad D and Solla S A 1995 Phys. Rev. E 52 4225–43
[13] Copelli M and Caticha N 1995 J. Phys. A: Math. Gen. 28 1615–25
[14] Maass W, Natschlager T and Markram H 2002 Neural Comput. 14 2531–60
[15] Mante V, Sussillo D, Shenoy K V and Newsome W T 2013 Nature 503 78–84
[16] Sussillo D, Churchland M M, Kaufman M T and Shenoy K V 2014 Nat. Neurosci. 18 1025–33
[17] Wang J, Narain D H, Hossein E A and Jazayeri M 2018 Nat. Neurosci. 21 102–10
[18] Russo A A et al 2018 Neuron 97 953–66
[19] Chollet F et al 2015 Keras https://keras.io
[20] Abadi M et al 2015 TensorFlow: large-scale machine learning on heterogeneous systems software available from  
https://tensorflow.org/
[21] Thompson C and Shure L 1995 Image processing toolbox: for use with MATLAB; [user’s guide ] (MathWorks) https://www.  
mathworks.com/products/matlab.html
[22] Cunningham J P and Yu B M 2014 Nat. Neurosci. 17 1500–9
[23] Balagué-Ballester E, Lapich C C, Seamans J K and Durstewitz D 2011 PLoS Comput. Biol. 7 e1002057
[24] Barak O 2017 Curr. Opin. Neurobiol. 46 1–6
[25] Kao T-C and Hennequin G 2019 Curr. Opin. Neurobiol. 58 122–9
[26] Landau I D and Sompolinsky H 2018 PLoS Comput. Biol. 14 e1006309
[27] García del Molino L C, Pakdaman K, Touboul J and Wainrib G 2013 Phys. Rev. E 88 042824
[28] Aslani M, Lambiase R and Carletti T 2018 Sci. Adv. 4 eaau9403
[29] Raposo D, Kaufman M T and Churchland A K 2014 Nat. Neurosci. 17 1784–92
[30] Barak O, Tsodyks M and Romo R 2010 J. Neurosci. 30 9424–30