Using Tensor Decomposition to Understand the Emergence of Recurrent Neural Networks' Dynamics

Melina Müller

Master of Science
School of Informatics
University of Edinburgh
2024

Abstract

The activity of a neural population can be studied through a dynamic system perspective. Research that has used this approach to gain insight into the emergence of behaviour, usually only considers the final dynamic system. Yet, to fully comprehend how neural dynamics lead to computations, their evolution during learning needs to be understood. This study investigates how the dynamic systems evolve in Recurrent Neural Networks (RNNs) while learning a context-dependent decision making task. By using two tensor decomposition methods (CANDECOMP and Low tensor rank RNN) on the weights of the RNNs, the low-dimensional dynamics of the neural activities over training are captured. These reveal that the evolution of the dynamics consists of two phases which timings differ between RNNs: the integration of 1) stimuli evidence and 2) context. These findings give insight into the steps involved in learning a task as well as the individual differences between network dynamics.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Melina Müller*)

Acknowledgements

I would like to thank the following people for their unwavering support during my dissertation.

My supervisor Dr Angus Chadwick for his continuous guidance and crucial insights throughout the project. My co-supervisor Dr Marino Pagan for providing me with the knowledge and data needed to undertake the project and his infectious passion for it. Also thank you to Arthur Pellegrino, for always taking the time to help me and give me advice.

Thank you to my family for always believing in me and being there for me whenever I need them. And thank you to my friends, especially in the MSc Cognitive Science cohort, who always keep my spirits high, even at the most stressful times.

Table of Contents

Chapter 1

Introduction

One of the main aims of neuroscience is to understand how neural activity leads to computations and behaviours. This is usually accomplished by analysing the neural recordings of animals and humans while they perform tasks. These investigations have revealed that the activity of a single neuron is often more complex than simple correlation with stimuli or internal triggers could explain [40]. Instead, the whole population or network of neurons need to be considered, to understand how their joint activity gives rise to the investigated phenomenon.

One approach to achieve this is by viewing the neural population as a dynamical system and studying its latent dynamics. Since the underlying model for biological networks is usually not known, recurrent neural networks (RNNs) are used in their place to simulate them and form hypotheses about their dynamics. For instance, Mante et al. [24] and Pagan et al. [29] investigated how the same stimuli can have different influences on a decision depending on the context (context-dependent decision-making) using this approach. They found that the dynamic systems consider only relevant stimuli by forming different dynamic characteristics for each context. How exactly these dynamics vary can differ between networks [29].

To be able to understand why specific solutions are found, the evolution of the dynamics during learning needs to be considered. Yet, most research using dynamic system theory for biological and neural networks only studies the final network [40, 43]. For this reason, this project aims to investigate how the dynamics observed in the task-trained RNNs from Pagan et al. [29] come to be. This can be accomplished by considering the evolution of the weights of the RNN since they determine its dynamic system [16]. The change of the weights over training can be analysed and visualised using tensor decomposition. Tensor decomposition offers the possibility to

simultaneously capture changes across all dimensions of a higher-order array under weak uniqueness constraints [19], making it perfectly suited for this aim.

It is hypothesised that by investigating the changes in the weights, different phases in the evolution of dynamics for solving the context-dependent decision making task can be detected. These stages are assumed to be captured by the components of the tensor decomposition and differ between RNNs. Through this, insights can be gained into how learning influences neural activity on the population level and their latent dynamics.

The following section (Chapter 2) introduces the background knowledge about dynamical system theory and context-dependent decision making needed to understand the investigated dynamics. Two different tensor decomposition methods are then described in Chapter 3 and 4 and applied to the RNNs trained by Pagan et al. [29]. The insights gained from these methods and their implications are discussed in Chapter 5.

Chapter 2

Background

2.1 Notation

To aid in understanding the discussed topics, relevant notations and definitions of concepts are given here, which are based on Kolda and Bader [19]. An N^{th}-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is a multidimensional array with N indices, resulting from the tensor product of N vector spaces. The order of a tensor specifies its number of dimensions/modes. A vector is, therefore, a 1^{st}-order tensor, a matrix is a 2^{nd}-order tensor, and anything with a higher order is called a higher-dimensional tensor.

A vector of length I will be referred to as a bold, lowercase letter, $\mathbf{x} \in \mathbb{R}^I$, while a matrix of size $I \times J$ will be notated as a bold, uppercase letter, $\mathbf{X} \in \mathbb{R}^{I \times J}$. Any higher-order tensor will be an uppercase letter in Euler script. For example, a 3-way tensor of size $I \times J \times K$ is $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. Scalars are notated as a lowercase letter: $a \in \mathbb{R}$.

To refer to a single element of a tensor each of its indices needs to be fixed, e.g. x_{ijk} is an element of $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ where $i \in \{1, \ldots, I\}$, $j \in \{1, \ldots, J\}$, and $k \in \{1, \ldots, K\}$. In general, to refer to arrays within a tensor, each index either gets fixed or is denoted as a colon, which indicates that all elements of that mode are accessed. For instance, for the third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, fixing its third mode but referring to all elements in the first two modes gives a matrix $\mathbf{X}_{::k} \in \mathbb{R}^{I \times J}$ which is also called a frontal slice.

The inner product of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ is the sum of the product of each of its elements: $< \mathcal{X}, \mathcal{Y} >= \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N}$.The Frobenius norm of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ is then defined as the square root of the inner product of $\mathcal{X}$ with itself: $\|\mathcal{X}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2}$. In contrast, the elements of the vector outer product $\mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \ldots \circ \mathbf{a}^{(N)} = \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$, for vectors $\mathbf{a}^{(1)} \in \mathbb{R}^{I_1}, \ldots, \mathbf{a}^{(N)} \in \mathbb{R}^{I_N}$, are defined as $x_{i_1 i_2 \ldots i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \ldots a_{i_N}^{(N)}$.

2.2 Computations through dynamics

To understand how neural activity transforms stimuli input into behaviour, it is often not enough to only consider each neuron on its own. Instead how the population carries out computations needs to be studied [40]. One way to do this is by viewing the network of neurons as a dynamical system. The activity in a population of N neurons at a specific time point t can be denoted by $\mathbf{x}(t) \in \mathbb{R}^N$. This vector can be understood as a point in an N dimensional state space where coordinate $n \in \{1, \ldots, N\}$ represents the activity of neuron n [43]. The population's activity over time then carves out a trajectory of this point across the space. The state space's latent dynamics determine these trajectories and thus the computations carried out on the population level. Hence, by determining the dynamic's characteristics through dynamical system theory, insights can be gained into how neural activity elicits behaviour.

In more detail, the change in neural activity over time is determined by the differential equation $\frac{dx}{dt} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$, where $\mathbf{u}(t)$ is external input into the neural population at time t and $\mathbf{f}$ is the change in activity based on the current state and the external input [43]. It is $\mathbf{f}$ which determines the activities' trajectory and thus by "reverse engineering" it, the underlying dynamics can be characterised. Since $\mathbf{f}$ tends to be nonlinear in neural networks, this is usually accomplished by dividing the state space up into different areas which get linearised separately [41]. For instance, common centres of linearisation are fixed points $(\mathbf{x}^*, \mathbf{u}^*)$, which are points where the neural activity does not change, i.e. $\frac{dx}{dt} = 0$. The nonlinear dynamics $\mathbf{f}(\mathbf{x}^* + \Delta\mathbf{x}, \mathbf{u}^* + \Delta\mathbf{u})$ around a fixed point $(\mathbf{x}^*, \mathbf{u}^*)$ can then be linearised using Taylor's theorem as $\mathbf{f}(\mathbf{x}^* + \Delta\mathbf{x}, \mathbf{u}^* + \Delta\mathbf{u}) = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + \frac{\delta \mathbf{f}}{\delta \mathbf{x}}(\mathbf{x}^*, \mathbf{u}^*)\Delta\mathbf{x} + \frac{\delta \mathbf{f}}{\delta \mathbf{u}}(\mathbf{x}^*, \mathbf{u}^*)\Delta\mathbf{u} + \ldots$ [41]. Assuming $\Delta\mathbf{x} = \mathbf{x}(t) - \mathbf{x}^*$ and $\Delta\mathbf{u} = \mathbf{x}(t) - \mathbf{u}^*$ are small, higher-order terms are negligible. Thus, since $\mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) = 0$,

$$\mathbf{f}(\mathbf{x}^* + \Delta\mathbf{x}, \mathbf{u}^* + \Delta\mathbf{u}) \approx \frac{\delta \mathbf{f}}{\delta \mathbf{x}}(\mathbf{x}^*, \mathbf{u}^*)\Delta\mathbf{x} + \frac{\delta \mathbf{f}}{\delta \mathbf{u}}(\mathbf{x}^*, \mathbf{u}^*)\Delta\mathbf{u} \tag{2.1}$$

$$= \mathbf{A}(\mathbf{x}^*, \mathbf{u}^*)\Delta\mathbf{x} + \mathbf{U}(\mathbf{x}^*, \mathbf{u}^*)\Delta\mathbf{u} \tag{2.2}$$

[29]. The trajectories of activity in the linearised system can then be understood by computing the Eigendecomposition on the Jacobian matrix $\mathbf{A}$. The resulting right eigenvectors are the directions in the state space where activity evolves independently of each other, with the corresponding eigenvalue capturing the type of change in activity they exhibit [39]. For instance, if all the eigenvalues have negative real parts values, i.e. the system is stable in all directions, then the fixed point is an attractor. Nearby activity

converges towards the fixed point, which for instance can be understood as a form of memory. Other possible trajectories that can be captured through this, is neural activity being repelled from or orbiting around the fixed point [43].

These fixed points can then be arranged in various ways in the state space to carry out complex computations. For instance, a line attractor is an arrangement of fixed points along a line in the state space. Each of these fixed points has one eigenvalue close to 0 which corresponding right eigenvector pointing in the direction of the line attractor, while all other eigenvalues are negative [29]. Through this surrounding activity is attracted onto the line attractor but only external input can cause movement along it. A line attractor can therefore be used for accumulating evidence (external input) for and against two possibilities (2 ends of the line) [24, 29]. Hence it can be used for making a binary choice [9], highlighting how the underlying dynamics of the neural population lead to computations and behaviour.

One problem of using dynamical system theory in neuroscience is that the exact underlying function $\mathbf{f}$ is typically not known for biological networks [43]. To overcome this drawback, artificial neural networks are commonly employed to model biological networks. In particular, RNNs are often used since they allow for feedback and a time notion, similar to biological networks [2]. The RNNs are typically defined as discrete or continuous models in one of two forms [28] :

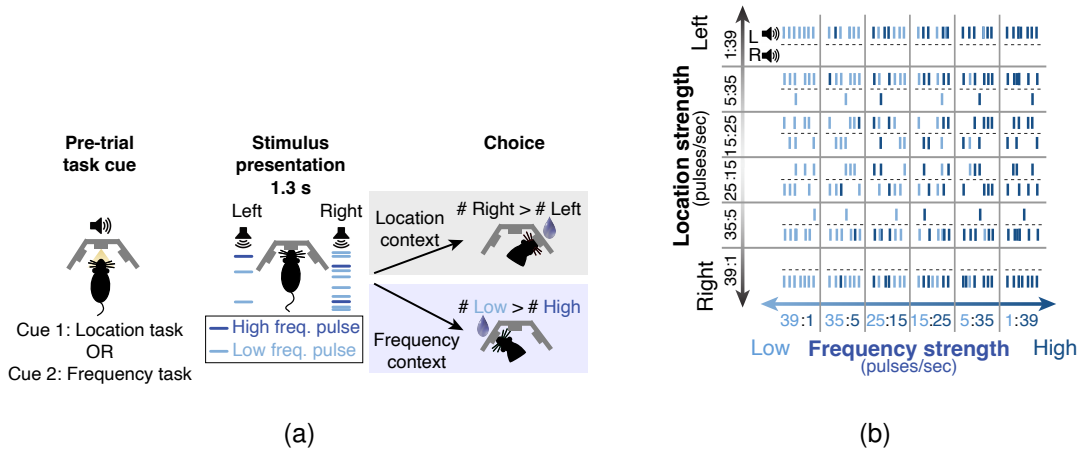
$$\tau\frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x}(t) + \boldsymbol{W}\phi(\boldsymbol{x}(t)) + \mathbf{B}\boldsymbol{u}(t) + \mathbf{b}, \text{or} \tag{2.3}$$

$$\tau\frac{d\mathbf{r}}{dt} = -\mathbf{r}(t) + \phi(\boldsymbol{W}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{u}(t) + \mathbf{b}), \tag{2.4}$$

which will be referred to as the current rate model and the firing rate model respectively. $\boldsymbol{x} \in \mathbb{R}^N$ is considered the synaptic currents in each of the N neurons (called hidden units for RNNs), which can get transformed into the unit's firing rates $\mathbf{r} \in \mathbb{R}^N$ through a non-linear function ϕ, i.e. $r_n = \phi(x_n)$. $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ is the recurrent weight between each unit while $\mathbf{B} \in \mathbb{R}^{N\times M}$ is the input weight for external input $\boldsymbol{u} \in \mathbb{R}^M$. $\mathbf{b} \in \mathbb{R}^N$ is the bias of each unit and τ is the time constant. The activity of the network is read out linearly through the output $\boldsymbol{z} = \boldsymbol{W}\mathbf{r}(t)$.

The two models represent the network's activity and thus the state space in different ways. While this leads to different reversed engineered dynamics, they are related to each other since the two models can be mapped to each other [28, 30].

Both models can be used in two different ways to study biological network activity during a behaviour. The RNN can be trained directly on the recorded neural data so that its activity resembles the biological activity. Otherwise, it can be trained to perform



(a) (b)

Figure 2.1: The context-dependent decision making task in [29] (a) At the start of the trial, the rat was given either the location or the frequency cue. It was then played a series of sound pulses, where each was either low or high frequency and came either from the left or right speaker. Depending on the cue, the rat had to pay attention either to the location or the frequency of the sounds. The predominant pulses for the relevant stimulus type then determined whether it had to turn left or right at the end of the trial. (b) When evaluating the rat's or RNN's performance multiple trials were carried out where the proportion of the inputs was fixed. The proportion between the left:right pulses / low-frequency:high-frequency pulses could fall into one of six bins respectively, representing their evidence strength for turning left vs right. Source: [29].

the same task as the animal, whose activity was recorded [43]. Either way, this leads to networks whose underlying model is known and can be reverse-engineered as described in this section. The results can then be used to form and test hypotheses about the dynamics in biological networks.

2.3 Context-dependent decision making

One area in neuroscience where this approach has been used to understand the emergence of behaviour is context-dependent decision-making. Depending on the context, e.g. one's goals, the same stimuli can elicit completely different behaviour [24]. It has been theorised that the prefrontal cortex's (PFC) activity represents this setting and influences lower cortical areas through top-down signals in how they process the context-relevant and irrelevant stimuli [26, 27]. Yet, there has been found contradicting evidence for [3, 44] and against [34, 36] this early, top-down gating of stimuli based on

their relevance.

To investigate this on the population level, Mante et al. [24] trained two monkeys and some RNNs to make a binary decision, based on either the colour or the direction of motion of some moving dots and the given context (attention to colour vs movement). When using target dimensionality reduction on both the monkey's neural activity in the PFC and the RNNs activity, they observed similar patterns. These were explained by reverse-engineering the RNNs dynamics, which showed two line attractors, one for each context. Colour/Motion inputs caused neural activity to move away from its location on the active line attractor to a similar extent, independent of their relevance. However, if the input was relevant to the context then the activity relaxed back onto the Line Attractor at a distance from its point of origin towards the choice evidenced by the stimuli, while activity perturbed by irrelevant inputs returned to the original location. Hence they concluded that it is the PFC itself that integrates the relevant stimuli using the line attractors instead of them being preselected through early gating.

However, when Pagan et al. [29] trained rats and RNNs on a similar task using auditory cues (see Figure 2.1), they were able to show that there exists a larger solution space that can explain the biological neural activity than what the RNNs discovered. For each trial, a rat was played a series of sound pulses which could come out either from a left or right speaker (location) and be low- or high-frequency (frequency). Depending on an initial context cue, it had to pay attention to one of the two stimuli types (location vs frequency) to make a binary choice. In the location context, it had to turn right if more sounds were coming from the right than the left speaker and other else left. Similarly, in the frequency context, more high-frequency pulses meant turning right as opposed to left. The task was also used to train the RNNs, which used a discrete version of the firing rate model (Equation 2.4) for the RNNs, where $\phi = \tanh()$ and $\mathbf{Bu}(t) + \mathbf{b} = \mathbf{i} = \boldsymbol{w}^{Loc} i^{Loc} + \boldsymbol{w}^{Frq} i^{Frq} + \boldsymbol{W}^{Ctx} \mathbf{i}^{Ctx} + \mathbf{b}$. For each time step during a trial, the location input $i^{Loc} \in \mathbb{R}$ represented the difference in right vs left pulses, $i^{Frq} \in \mathbb{R}$ the difference in low vs high-frequency pulses, and the context input $\mathbf{i}^{Ctx} \in \mathbb{R}^2$ the one-hot encoded context. $\boldsymbol{w}^{Loc} \in \mathbb{R}^N, \boldsymbol{w}^{Frq} \in \mathbb{R}^N$, and $\boldsymbol{W}^{Ctx} \in \mathbb{R}^{N \times 2}$ were then the corresponding input weights. The sign of the readout at the end of the trial indicated the RNN's choices. The linearization of the model at each fixed point $(\mathbf{r}^*, \mathbf{i}^*)$, as described in Equation 2.2 , is

$$\tau \dot{\mathbf{r}}(\mathbf{r}^* + \Delta\mathbf{r}, \mathbf{i}^* + \Delta\mathbf{i}) \approx \mathbf{A}(\mathbf{r}^*, \mathbf{i}^*)\Delta\mathbf{r} + \mathbf{U}(\mathbf{r}^*, \mathbf{i}^*)\Delta\mathbf{i} \tag{2.5}$$

$$= (-\mathbf{I} + \mathbf{D}\boldsymbol{W})(\mathbf{r}^*, \mathbf{i}^*)\Delta\mathbf{r} + \mathbf{D}(\mathbf{r}^*, \mathbf{i}^*)\Delta\mathbf{i}, \tag{2.6}$$

where $\mathbf{D}$ is a diagonal matrix, which diagonal entries are defined as $d_{jj} = g'(\boldsymbol{w}_{j:}\,\mathbf{r}^* + \mathbf{i}^*)$. Assuming that there are no sound pulses at the fixed points, $\mathbf{i}^* = \boldsymbol{W}^{\mathrm{Ctx}}\mathbf{i}^{\mathrm{Ctx}} + \mathbf{b}$ takes one of two values depending on the context. This leads to the two context-dependent line attractors, differing in their dynamics. The net movement along the attractor for a stimuli pulse (location or frequency) is defined by the inner product of the linearised input $\mathbf{D}(\mathbf{r}^*, \mathbf{i}^*)\Delta\mathbf{i}$, which perturbs the activity of the line attractor, and the selection vector $\mathbf{s}$. $\mathbf{s}$ is the left eigenvector of $\mathbf{D}\boldsymbol{W}$ corresponding to the eigenvalue 0 and determines how the activity relaxes back onto the attractor [29].

The taught RNNs in both Mante et al. [24] and Pagan et al. [29] mainly differed in their selection but not in their input vectors between contexts and input types. However, there exist other solutions where the inputs can differ through early gating, which still shows the same low-dimensional activity patterns as the trained RNNs and biological networks [29]. Hence, the contradicting findings about early gating are just different, equally valid solutions. Indeed, the rats in Pagan et al.[29] showed large differences in both their neurological and behavioural data, suggesting that they might use various solutions. This observation raises the question not whether early gating exists or not, but what causes it to be used in one situation/individual but not in another. To be able to understand this, the emergence of the dynamics needs to be considered and not just the final system. Yet, studies in neuroscience using dynamical system theory have only used it on the final trained networks.

2.4 Bifurcations in dynamics during learning

As can be seen in Equation 2.6, the dynamics of the system are defined by the weights,i.e. parameters, of the RNN. For instance, the Jacobian, which determines the direction of the line attractor and selection vector, consists of the recalled columns of the recurrent weight matrix [30]. The parameter space of the model consists of various regions with different dynamical systems, which are separated by bifurcation curves. Bifurcation is a qualitative change in the dynamic system, such as the emergence of fixed points, when changing a parameter [14]. Thus, while a network learns and adapts its parameters, it crosses these bifurcation boundaries, changing its dynamics, until it ends up with its final system. Hence, by studying the changes in the weights during learning insights can be gained into the different dynamic characteristics that emerge.

Most studies which consider bifurcation in RNNs are concerned with how it can hinder optimisation through gradient decent [14, 31, 13, 33]. Haputhanthri et al. [16]

is one of the only studies that have considered bifurcation as a potential to better understand learning in both biological and artificial neural networks and not just as a nuisance. They were able to show that the change in dynamics is reflected in the loss and change in the models' parameters. However, they mainly considered the average across models to understand these relations, instead of comparing the differences between them to better understand how different dynamical systems can be learned.

This study aims to build on this to increase an understanding of individual differences in learned dynamics concerning context-dependent decision making. While the dynamics of the trained RNNs in Pagan et al. [29] only encompassed parts of the possible solution space, they still showed enough variances between them, that it can be assumed that they differ in their learning. Thus, the change in their weights will be studied, to understand how dynamics emerge and their differences and similarities across models.

2.5 Tensor decomposition

One method to study the change in the RNNs' weights during training is tensor decomposition, the higher-order version of matrix decomposition. The recurrent weight matrices of an RNN for different trial points across learning can be stacked along a third dimension, giving a 3$^{\text{rd}}$ order tensor of size $\mathtt{Unit} \times \mathtt{Unit} \times \mathtt{Trial}$. A tensor decomposition method then separates this tensor into three factorisation matrices, each relating to one of the tensor modes (columns, rows, and trials).

Through this, the change of different rows and columns factors of the recurrent weight matrix over training can be captured. This reveals components of the weights that differ in their trajectories during training and thus the dynamics they are involved in. This is the advantage of tensor decomposition as opposed to decomposing each matrix individually: the interaction across all modes is captured simultaneously [45]. Moreover, some tensor decomposition methods, as opposed to matrix decomposition methods, can find unique solutions under weak constraints [19]. This makes it more useful for this task since it helps to ensure that the discovered weight patterns are indeed the ones related to the dynamics and not just one of multiple explanations.

This study will use two different tensor decomposition methods: CANDECOMP and Low Tensor Rank RNN (LtrRNN). Each of them will be introduced in the following two chapters alongside how they were applied to the RNNs and their results.

Chapter 3

CANDECOMP decomposition

3.1 Methodology

One of the most popular tensor decomposition methods is CANDECOMP (CP; also known as canonical decomposition or PARAFAC) [10, 17]. This method decomposes an N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ into the sum of R rank-one N^{th}-order tensors. An N-way tensor is rank-one if it can be expressed as the outer product of N vectors, $\mathcal{X} = \mathbf{x}_1 \circ \mathbf{x}_2 \circ \ldots \mathbf{x}_N$, where $\mathbf{x}_n \in \mathbb{R}^{I_n}$ for $n \in \{1, \ldots N\}$. For example, for the 3^{th}-order recurrent weight tensor $\mathcal{W} \in \mathbb{R}^{N \times N \times K}$

$$\mathcal{W} \approx \widehat{\mathcal{W}} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{3.1}$$

where $\mathbf{a}_r \in \mathbb{R}^N$, $\mathbf{b}_r \in \mathbb{R}^N$, and $\mathbf{c}_r \in \mathbb{R}^K$ for all $r \in \{1, \ldots, R\}$ are the factors of the decomposition (see Figure 3.1). Each variable corresponds to one of the modes of $\mathcal{W}$ and all R factors belonging to the same dimension can be concatenated into a factorisation matrix. For example, the factors $\mathbf{a}_r$ are the column factors of $\mathcal{W}$ and $\mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_1 \ldots \mathbf{a}_R] \in \mathbb{R}^{N \times R}$ is the mode's respective factorisation matrix. Similarly, $\mathbf{b}_r$ are the factors for the rows and $\mathbf{c}_r$ to the trial points of $\mathcal{W}$, each being concatenated to $\mathbf{B} \in \mathbb{R}^{N \times R}$ and $\mathbf{C} \in \mathbb{R}^{K \times R}$ respectively. The rank-one tensor $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ is referred to as a component, e.g. $\mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1$ is component one of the decomposition. R is the rank of the decomposition and $\widehat{\mathcal{W}}$ is of tensor rank R since it is the sum of R rank-one tensors. Hence, $\mathcal{W}$ is rank R if $\mathcal{W} = \widehat{\mathcal{W}}$ for a rank R CP decomposition.

Note that each of the factors can be normalised, with the weights of all factors in the same component captured by λ_r, i.e. $\widehat{\mathcal{W}} = \sum_{r=1}^{R} \lambda_r \ \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$. In general,

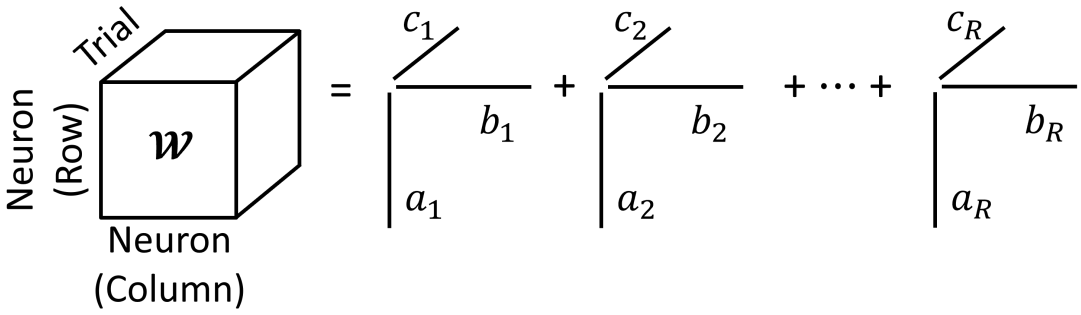


Figure 3.1: CANDECOMP decomposition of the 3rd order weight tensor for rank R.

factors within a component can always be rescaled as long as the overall component value does not change, i.e. $\hat{\mathcal{W}} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = \sum_{r=1}^{R} \alpha_r \mathbf{a}_r \circ \beta_r \mathbf{b}_r \circ \gamma_r \mathbf{c}_r$, where $\alpha_r \beta_r \gamma_r = 1, \ \forall r \in \{1, \ldots, R\}$. Additionally, the order of the R components in the sum can always be changed. Besides these two exceptions, the decomposition of the rank R tensor $\hat{\mathcal{W}} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ is unique [19]. Moreover, a sufficient condition for a CP decomposition of rank R to be unique for a 3-way tensor $\mathcal{X}$ is that $k_{\mathbf{A}} + k_{\mathbf{B}} + k_{\mathbf{C}} \geq 2R + 2$, where $\mathbf{A}, \mathbf{B},$ and $\mathbf{C}$ are the factorisation matrices and $k_{\mathbf{X}}$ is the Kruskal/k-rank of a matrix $\mathbf{X}$ [21]. The k-rank indicates the maximum number of columns in $\mathbf{X}$ such that any combinations of them are linearly independent.

The CP decomposition of the 3rd-order weight tensors was carried out using the Python package Tensorly [20]. The package uses an alternative least squared (ALS) algorithm, where all three of the factorisation matrices are initialised randomly. ALS keeps two of them fixed while optimising the third factorisation matrix and then moves on to the next factorisation matrix. This procedure is carried out until some threshold is met [19].

3.2 Data

The data used in this study are the simulated RNNs from Pagan et al. [29] that were taught by them to perform context-dependent decision making for auditory stimuli. Ten of their trained RNNs were investigated. Five of these had 20 hidden units and the other five had 100, with each having different initial weights. Each is named based on its unit size, e.g. RNN20-1 is the first RNN with 20 hidden units. For the 20 hidden units, 120000 training/batch steps were carried out. After each batch, which consists of 256 trials, the weights were updated using backpropagation with the Adam optimizer. The

100 hidden units were trained over 60000 steps. After every 1000 batches, the recurrent and input weights were sampled for each RNN. These K sample points will be referred to as the trial points $k \in \{1, \dots, K\}$. Hence, the recurrent weight tensor $\mathcal{W} \in \mathbb{R}^{N \times N \times K}$ for an RNN with 20 hidden units was of size $20 \times 20 \times 120$ and for 100 hidden units of size $100 \times 100 \times 60$. This tensor will be referred to as the "Full Tensor".

To gain different insights into how the weights change and account for the fact that the weights were randomly initialised, two additional recurrent weight tensors were formed for each of the RNNs. Firstly, for the "Delta Tensor" $\Delta\mathcal{W} \in \mathbb{R}^{N \times N \times K}$, the first sampled weight matrix was subtracted from all following weight matrices, i.e. $\Delta W_{::k} = W_{::k} - W_{::1}$ for $k \in \{1, \dots, K\}$. Secondly, for the "Tensor of Differences" $\mathcal{W}' \in \mathbb{R}^{N \times N \times K}$, the weight for each trial point was subtracted from the weight of the following trial point: $W'_{::k} = W_{::k+1} - W_{::k}$ for $k = \{1, \dots, K-1\}$ ($W'_{::K} = 0$). Whether a weight tensor is a Full Tensor, Delta Tensor, or Tensor of Differences will be referred to as its tensor type.

3.3 Rank selection

To be able to decompose these weight tensors, the rank of the CP decomposition needs to be selected. Computing directly the rank of a tensor with rational entries is NP-hard and thus usually not possible [18]. Instead, a common method is to decompose the tensor for consecutive ranks from $r = 1, \dots, R$ until some threshold is met. Ideally, this threshold would be that the reconstructed tensor of that rank is the same as the original tensor and thus its tensor rank is found [19]. However, this is often not the case since the original tensor might contain noise or components that are not of interest. Instead other evaluation methods are used to determine this threshold and hence the rank (see [12, 35] for an overview). Since there exists a wide range of them, five of the most common ones were used for evaluating the weight decompositions.

3.3.1 Evaluation methods

Firstly, one of the most common methods to evaluate models is to use cross-validation. Cross-validation helps to ensure that only relevant data instead of noise is captured by the components by withholding entries of the tensor during decomposition [22]. This was accomplished by sorting each tensor element randomly into one of ten bins. Its decomposition was then carried out ten times, masking each of the bins once (alongside

entries around it to account for correlation between neighbouring elements). The masked elements in the bin were then reconstructed based on the decomposition results and compared to the original tensor. However, it has been suggested that cross-validation is not the most suitable method for tensor rank determination [7]. This is because a rank r and rank $r+1$ decomposition do not necessarily have similar components (since ALS computes all of the factors at the same time). While for matrix decomposition methods a higher rank simply means that factors are added and thus cross-validation is therefore able to detect when these additional factors only capture noise, this is not the case for tensor decomposition [6].

A different method that can be used either on the directly decomposed tensor or on the cross-validation results, is to calculate how much variance of the original tensor can the reconstructed tensor explain. One measure which encapsulates this is the fit value which is defined as $\mathtt{fit}(R) = 1 - \frac{\|\mathcal{W}-\widehat{\mathcal{W}}\|_F}{\|\mathcal{W}\|_F}$ for a rank R decomposition [12]. Either a certain fit threshold, e.g. 70% of variance explained, or the elbow in the fit curve is used to determine the rank. However, since that value changes gradually it is often hard to see which exact rank value is the most optimal. For this reason, [42] defined Diffit which considers how the fit value for a specific rank relates to the fit of its neighbouring ranks: $\mathtt{Diffit}(R) = \frac{\mathtt{fit}(R)-\mathtt{fit}(R-1)}{\mathtt{fit}(R+1)-\mathtt{fit}(R)}$. The rank with the highest Diffit value would then be chosen which is a more clear measurement.

Another method that was developed to prevent this gradual change between ranks is core consistency diagnostic (CORCONDIA) [8]. The CP decomposition $\widehat{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ can be understood as a special form of the tensor decomposition method Tucker3 decomposition $\widehat{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$, where the core tensor $\mathbf{G}$ only has entries along its super-diagonal equal to the λ values and $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ being the factorisation matrices ($\times_n$ is the n-mode matrix product). CORCONDIA takes advantage of this by comparing the CP decomposition with a Tucker3 decomposition of the same rank, based on the assumption that if the CP decomposition is appropriate for the tensor then the core tensor of the Tucker3 decomposition will mainly have entries along the super-diagonal. The highest rank for which the decomposition is appropriate would then be the selected rank.

Lastly, [15] developed the method NORMO which determines the similarity between different factors of the same mode. The idea behind it is, that if the decomposition has a higher rank than the actual rank of the tensor then it will split up single components into multiple ones, which have similar factors. Hence, by calculating the similarity between two components based on the correlation of its factors, redundant components can be

identified. The highest rank for which none of the components are redundant is then determined as the tensor rank.

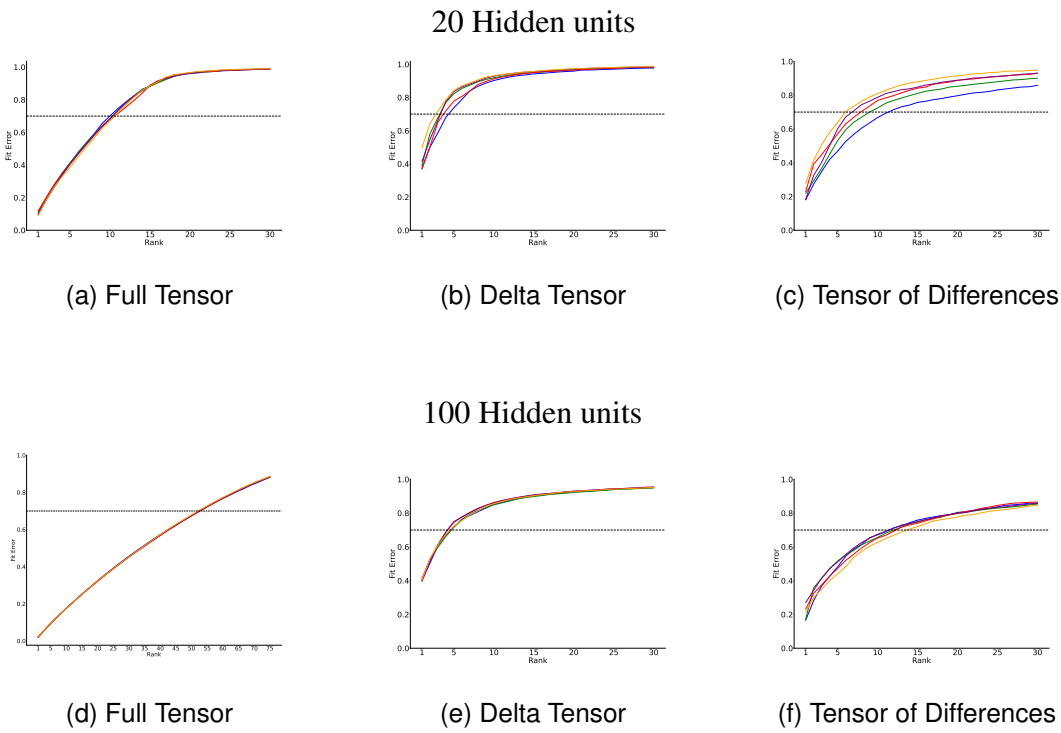
3.3.2 Evaluation results

	20 Hidden units			100 Hidden units		
	Full	Delta	Dif	Full	Delta	Dif
Cross-Validation	13	5	45	67	5	53
Fit Threshold	11	4	9	53	5	13
Fit Elbow	14	8	10	36	9	9
Diffit	41	43	30	97	93	75
CORCONDIA	1	2	2	1	2	2
NORMO	15	3	2	25	4	2

Table 3.1: Best rank for RNNs with the same hidden unit size for different tensor types (Full Tensor, Delta Tensor, Tensor of Differences) and evaluation methods.

CP decomposition was applied to each of the three tensor types for each of the RNNs for a range of different ranks. The tensors of RNNs with 20 hidden units were decomposed using ranks between 1 to 50, while for RNNs with 100 units decompositions with a maximum rank of 100 were computed. To account for the fact that the components are in a random order, which makes it more difficult to compare results, each decomposition was normalised, i.e. $\widehat{\mathcal{W}} = \sum_{r=1}^{R} \lambda_r \ \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$. The components were then reordered by their weights such that $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_R$. Hence, the first component is the most "important" one with the highest weight, followed by the second one, and so on.

Each of the evaluation methods was then applied to the decomposition results of each RNN and tensor type, which can be seen in Appendix A. Since for the same unit size, tensor type, and evaluation method, the RNNs exhibited similar ranks, a representative rank was computed across them, shown in Table 3.1. For the fit value, the best rank was chosen both based on when the decomposition explained 70% of the variance of the original tensor (see Figure 3.2) and based on where the elbow in the plot was (using the python package [37]). The fit threshold of 0.7 was also used to determine the rank for the cross-validation results. The representative rank across RNNs for evaluation methods using the fit value was calculated by computing the average fit at each rank across RNNs and determining the rank based on that. Similarly, for NORMO



(a) Full Tensor (b) Delta Tensor (c) Tensor of Differences

(d) Full Tensor (e) Delta Tensor (f) Tensor of Differences

Figure 3.2: Fit error $\texttt{fit}(\mathrm{R}) = 1 - \frac{\|\mathcal{W} - \widehat{\mathcal{W}}\|_{\mathrm{F}}}{\|\mathcal{W}\|_{\mathrm{F}}}$ of the decompositions for different ranks R. Each subfigure shows the fit error, i.e. the variance of the original tensor explained, for all recurrent weights of a specific tensor type (Full Tensor, Delta Tensor, Tensor of Differences) and hidden unit size (20 in the top row and 100 in the bottom row). Each line depicts one of the five RNNs of that unit size. The horizontal line is when the threshold of 0.7 (70%) is met.

the average similarity scores of the components were used across RNNs. Lastly, for CORCONDIA the mode across RNNs was taken to determine the best rank.

These suggested ranks differ strongly between evaluation methods (Table 3.1). Some methods, such as CORCONDIA, suggest a very low rank, while others, like Diffit, assume a rank double the unit size of the RNNs. This makes it hard to select the best rank for the decompositions. However, some insights can still be gained from these methods, even if the true ranks of the weights remain unclear. Firstly, the fact that RNNs within the same groups showed similar ranks, such as in Figure 3.2, is an indication that their weights exhibit similar patterns, i.e. dynamics. Furthermore, when comparing the Full Tensor type to the other types for both unit sizes, most evaluation methods show a higher rank for the first. The ranks of the Full tensor appear to be similar and thus probably related to the unit size of the RNNs. In contrast, the ranks for the Delta and Differences tensors are low for most evaluation methods (except Diffit

and cross-validation for the Tensor of Differences) and both unit sizes, due to them capturing the change in the weights which seems to be independent of the number of units.

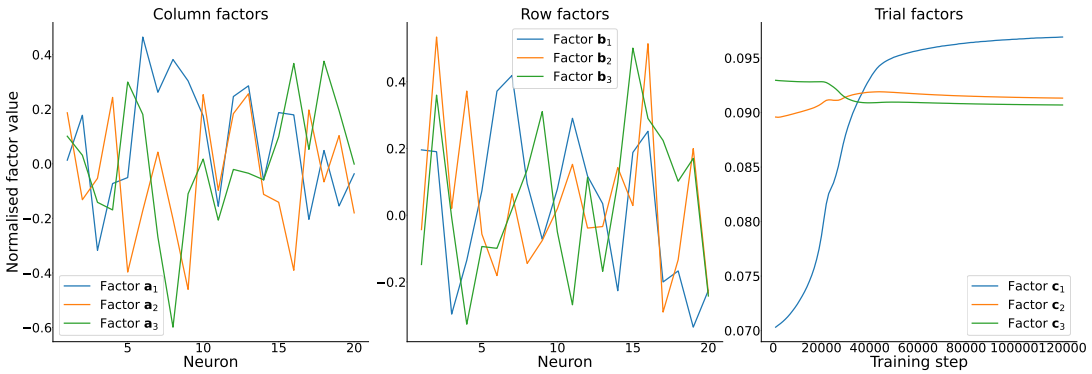
3.4 Factor Visualisation

To form a better understanding of these various decompositions and the change in weights they picked up, the resulting factors were visualised. Since there was no clear best rank, decompositions between ranks of one to five were considered. The lower value makes it easier to understand the main patterns that the components picked up and are in line with some of the suggested ranks of the decomposition results, especially for the Delta Tensors and Tensor of Differences.

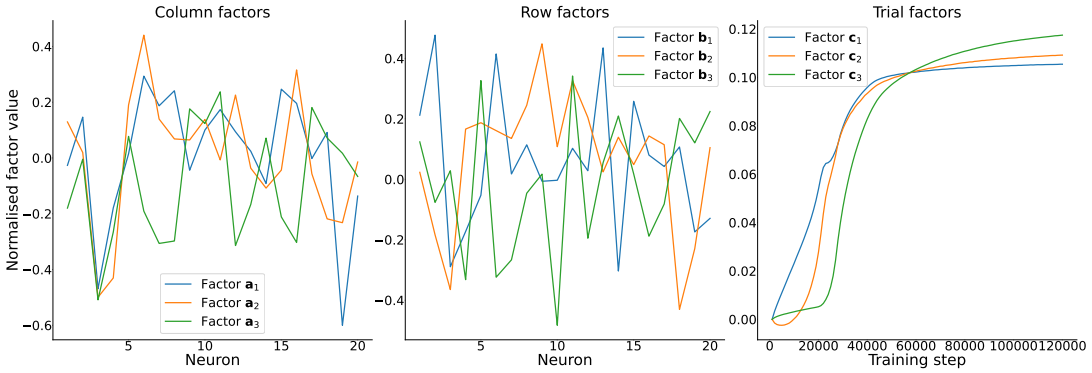
Firstly, an individual decomposition was considered. Figure 3.3 shows the rank three decompositions of RNN20-4 (see the same decomposition for all other RNNs in Appendix B). Each subfigure is the decomposition of one of the tensor types of the RNN's recurrent weights. Each column shows all of the factors corresponding to the columns, rows, and trial points respectively. For example, the first column in Figure 3.3a are the column factors a_1, a_2, a_3 of the rank three decomposition of the full weight tensor (Equation 3.1). When considering the column and row factors on their own, no particular pattern can be observed, since it is random which units are associated with which dynamics. However, the trial factors display how the weights of the units captured by the row and column factors, and thus the dynamics they are involved in, change over time. One of the trial factors for the Full Tensor (Figure 3.3a) and all of the trial factors of the Delta Tensor (Figure 3.3b) show a strong increase of the components over time with changes in their trajectory at various points between training steps 20000 to 40000. Around these same trial points, in particular points 20000 and 27000, the trial factors for the Tensor of Differences (Figure 3.3c) exhibit peaks.

These trajectories indicate that there are specific points during training where the weights of the RNN suddenly change quickly (based on the Tensor of Differences). These changes in the neural weights are permanent, as can be seen by trial factors of the Full and Delta tensor. Hence, it can be assumed that these long-lasting changes are related to the formation of the line attractors.

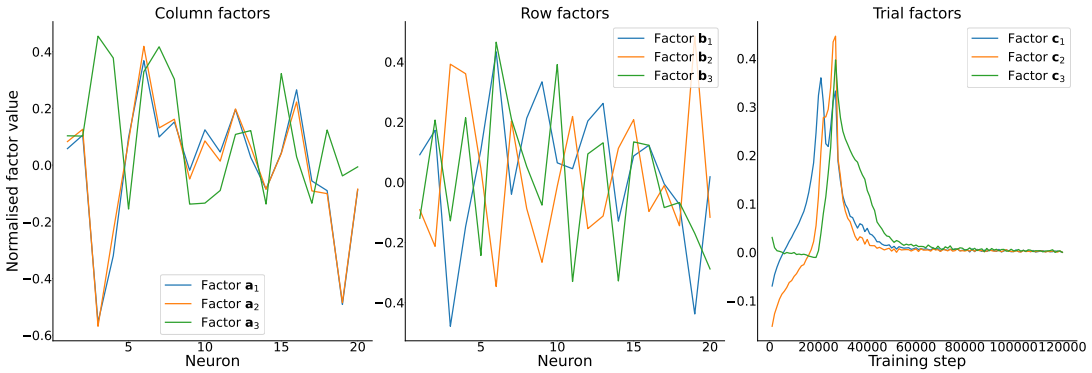
Before trying to further investigate what these changes in trajectory mean and whether they can be observed in all RNNs, other ranks were considered. This was done to make sure that the observed patterns are indeed related to the overall dynamics



(a) Full Tensor



(b) Delta Tensor



(c) Tensor of Differences

Figure 3.3: Rank three decomposition results for RNN20-4. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.

learned by the RNN and not just an individual characteristic of a specific decomposition, especially since the ranks of the tensors are ambiguous. Figure 3.4 shows the trial factors of rank one to five decompositions of all tensor types of RNN20-4. The decompositions

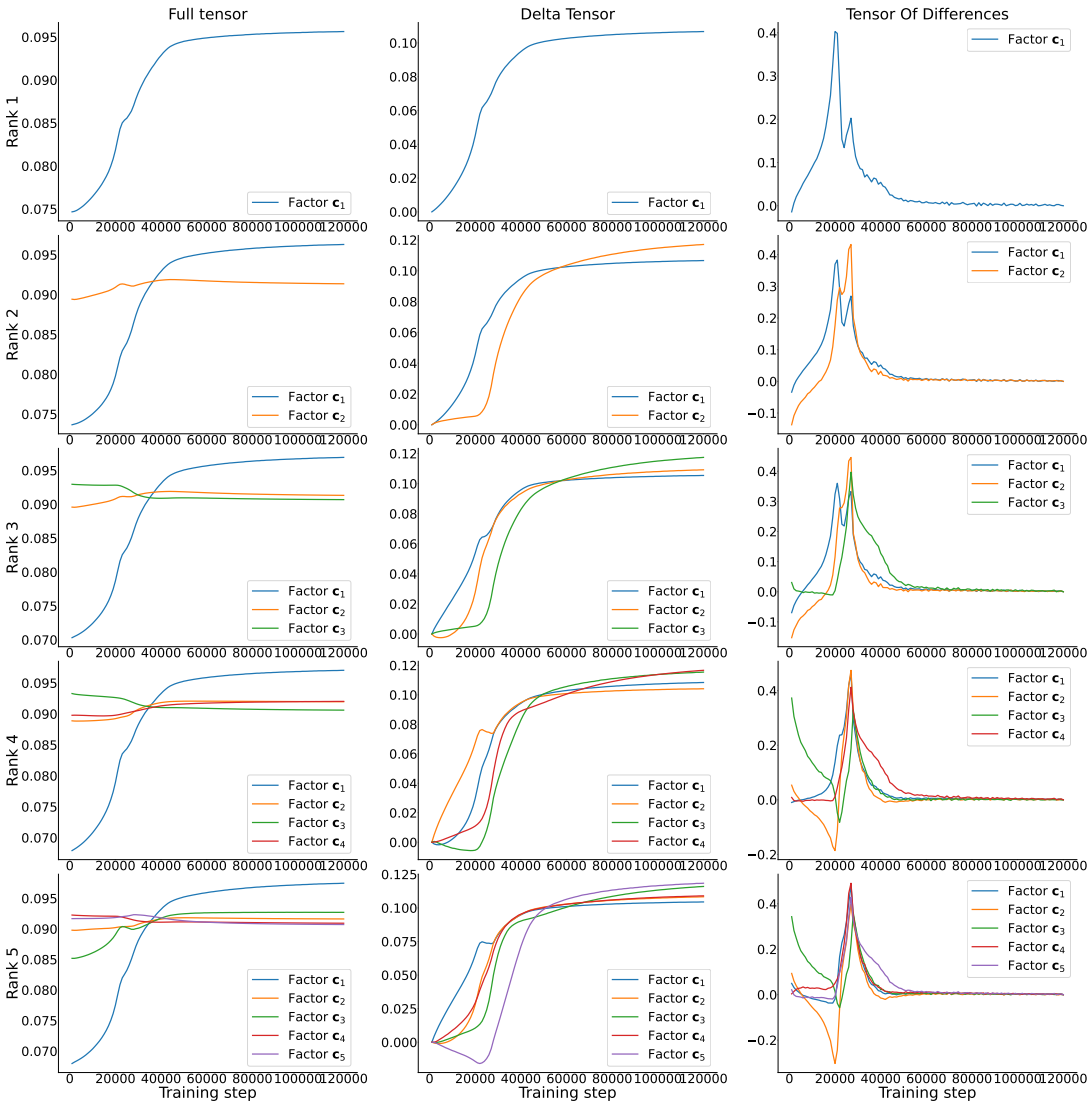


Figure 3.4: Trial factors of RNN20-4 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

of the same tensors for two neighbouring ranks can theoretically exhibit completely different factors since in ALS all factors are identified at the same time. Yet, when considering Figure 3.4, it can be noted that adding a component simply adds another factor with one of the lowest weights, without changing the existing ones too much. Through this, the main points of change observed in Figure 3.3 appear to be preserved across ranks. Higher rank decomposition simply picks up smaller variations of these, but their changes are still in the same time window of 20000 to 40000 training steps. The same observation was also found for the other RNNs which can be seen in Appendix C.

It should be noted that while there appears to be a big change between rank three

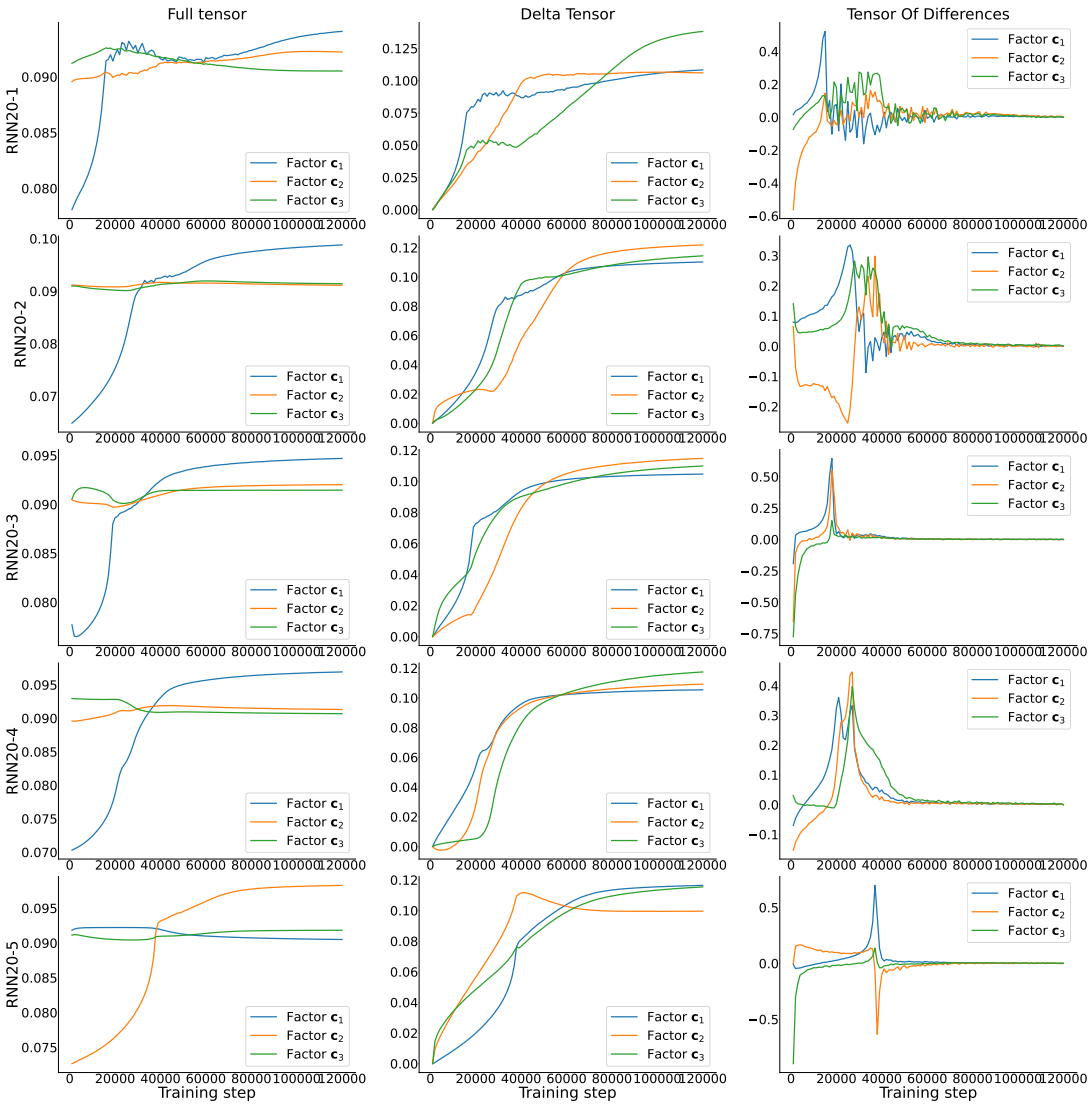


Figure 3.5: Trial factors of the rank three decompositions of the recurrent weights of five different RNNs with 20 hidden units. Each column is a different tensor type of the weights (Full Tensor, Delta Tensor, Tensor of Differences).

and four for the trial factors of the Tensor of Differences when considering the first peak, this can be explained through scaling. As mentioned before, the component $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = \alpha_r \mathbf{a}_r \circ \beta_r \mathbf{b}_r \circ \gamma_r \mathbf{c}_r$ as long as $\alpha_r \beta_r \gamma_r = 1$. Hence, while factors $\mathbf{c}_2$ in the rank four and five decompositions have a negative peak around 20000, multiplying them and either the corresponding row or column factor of that component with -1 would make it positive and thus more similar to the lower rank factors.

As a next step, the decompositions of different RNNs were compared to investigate whether they exhibit similar patterns as RNN20-4. Figure 3.5 shows the trial factors of the rank three decompositions for all weight tensor types for all RNNs with 20 hidden

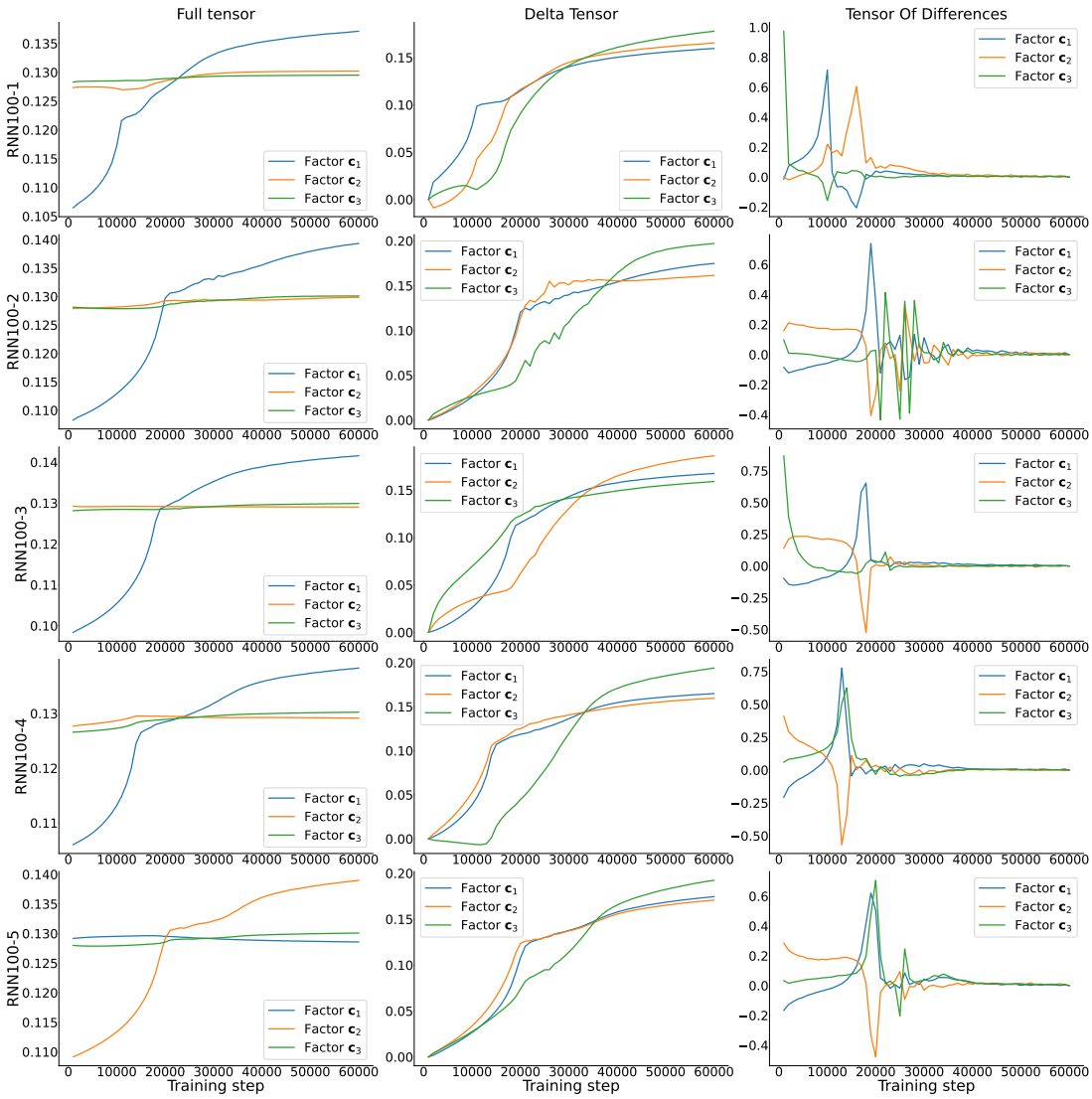


Figure 3.6: Trial factors of the rank three decompositions of the recurrent weights of five different RNNs with 100 hidden units. Each column is a different tensor type of the weights (Full Tensor, Delta Tensor, Tensor of Differences).

units. Figure 3.6 shows the decompositions of the RNNs with 100 hidden units.

The decompositions show similar patterns between RNNs for the same tensor type, independent of unit size. The Full Tensors consist of one trial factor, which normally has the largest or second largest weight, with a strong increase while the other two exhibit smaller changes. Similarly, for the Delta Tensors, all the factors have these increasing trajectories and the factors of Tensor of Differences exhibit peaks at certain time points. The changes in the trajectories of the trial factors for the Full and Delta Tensors align with the peaks of factors for the Tensor of Differences for the same RNN. Hence, as has been observed before, there are certain time points for each RNN where

the weights undergo quick, permanent changes.

However, there are also some notable differences. Firstly, the exact training step when weight changes occur varies between the RNNs. For instance, RNN20-3 main peak is at around 20000 while RNN20-5 has its at 40000 (Figure 3.5). Secondly, the number of peaks differs between the RNNs. While some RNNs, such as RNN20-4 (Figure 3.5) and RNN100-1 (Figure 3.6) have 2 main peaks, most of the RNNs only exhibit one. Lastly, the RNNs differ in the behaviour of the trial factors of the Tensor of Differences after the main peaks. Some show strong oscillations (e.g. RNN20-1) while others simply return to 0 (e.g. RNN100-3).

Overall these findings show that tensor decomposition is able to pick up changes in the recurrent weights of the RNNs, even if it is probably of a lower rank than the true rank of the tensor. The detected trajectories are consistent across ranks and RNNs, suggesting that they are related to the formation of the dynamic system used to solve the task and not to unique, non-relevant characteristics of the RNNs. This indicates that the dynamics form rapidly in discrete steps (peaks in the trial factors). However, when the dynamics form (timing of peaks) and the steps they involve (number of peaks and oscillations) appear to vary across RNNs.

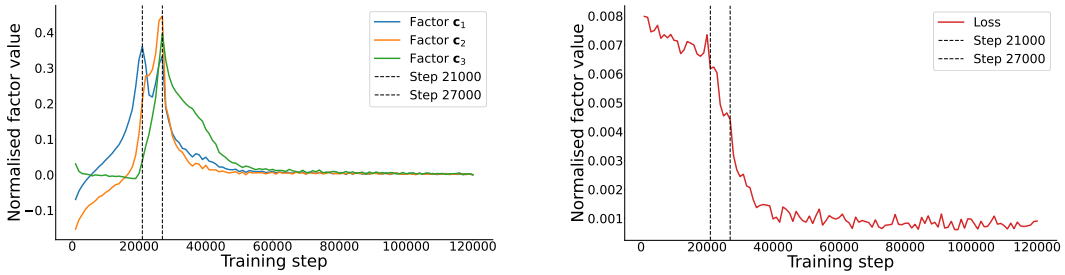
3.5 Comparison to RNNs' performances

To better understand the changes in the dynamics that were picked up by the trial factors, their trajectories were compared to performance metrics of the RNNs. Figure 3.7 shows the trial factors of the rank three decompositions of the Tensor of Differences for RNN20-4 and two different performance measurements of the RNN during training. Firstly, Figure 3.7b is the loss of the RNN during training. The two vertical lines in the factor and loss plots are aligned with the peaks of the trial factors. Secondly, Figure 3.7c plots the performance of the RNN at each trial point for different inputs and contexts. The proportion between left and right location pulses and low- and high-frequency pulses for a trial can take one of six values (see the six different bins in Table 3.2 and Figure 2.1b). For each possible combination of evidence strength of the two stimuli types and context, multiple trials were created for each training step. The percentage of times that the RNN chooses to turn right across the trials was then plotted.

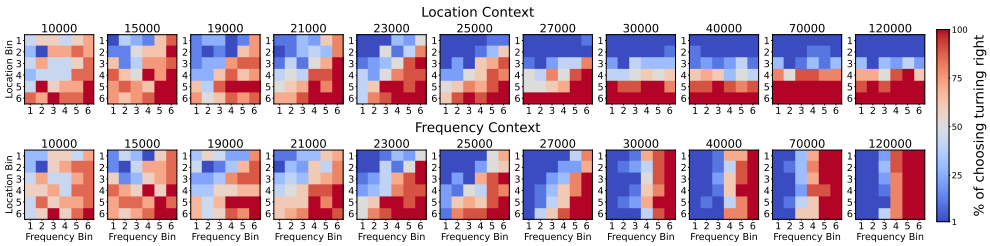
It can be observed that the loss suddenly starts decreasing at the same training step at which the peak of the first trial factor occurs (step 21000). The RNN's performance matrices show that by that point the RNN has learned to correctly identify which

	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6
Evidence strength	Strong left	Medium left	Weak left	Weak right	Medium right	Strong right
Proportion	39:1	35:5	25:15	15:25	5:35	1:39

Table 3.2: Different proportions that frequency and location pulses can have. Each bin determines the exact proportion between the pulses for the two choices for the specific stimuli type and thus how strong the overall evidence for a specific direction is. For location stimuli the proportion is left:right pulses and for the frequency stimuli it is low-frequency:high-frequency pulses.



(a) (b)



(c)

Figure 3.7: Different performance measures of RNN20-4 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

direction to choose based on the evidence. However, it considers and integrates both stimuli types equally and there is no difference between the two contexts. These three plots together suggest that the first peak corresponds with the formation of the line attractors. The line attractors do not differentiate between stimuli relevance yet (the product between the input and the selection vector is the same for both contexts and

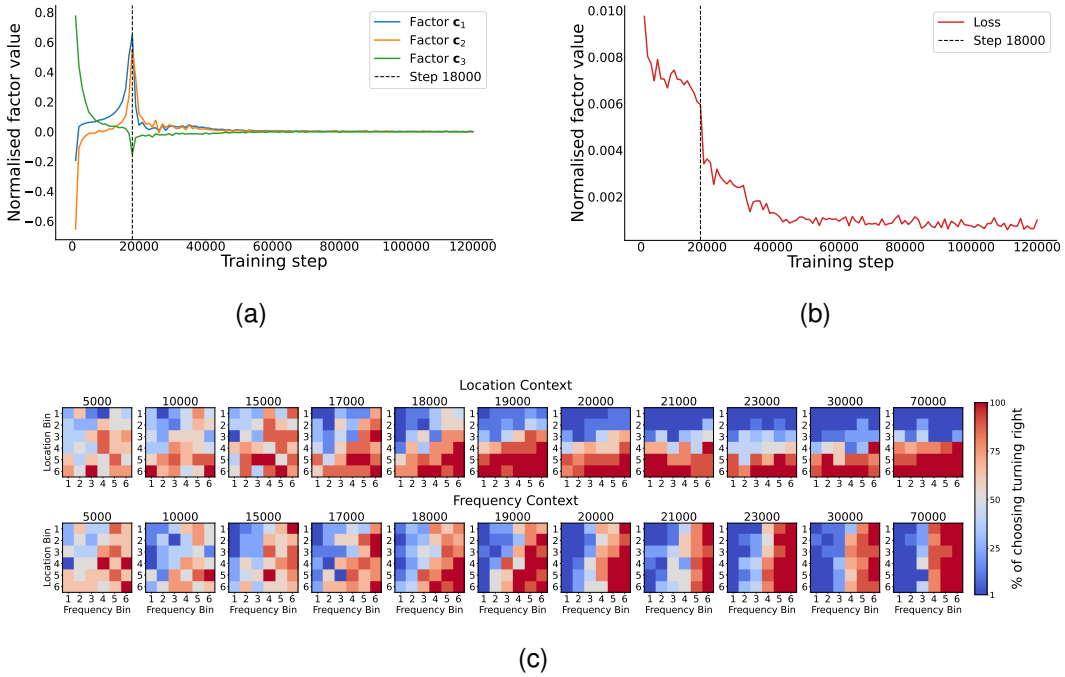


Figure 3.8: Different performance measures of RNN20-3 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

stimuli types). This ability emerges between steps 21000 to 27000, as can be seen in the performance matrices. In fact, by step 27000, which is the peak of the second and third trial factors, context integration has fully formed. Hence, the second change in the weights is when the line attractor dynamics become context-dependent. These weights then keep getting refined until point 40000, which is reflected by the slower decrease in factor $\mathbf{c}_3$ and the continuous reduction in loss.

These observations suggest that the trial factors captured two phases in the formation of the dynamic system: the formation of the line attractor and the integration of context. Yet, many of the RNNs' Tensor of Differences' trial factors only have one peak. Figure 3.8 shows the performance for RNN20-3. As before, the peak is located where the loss shows a steep drop. The performance matrices show at the peak point a similar diagonal pattern of evidence integration as in RNN20-4 for step 21000. However, there is already a light difference in the matrices across contexts with a preference for relevant stimuli and at the next trial point they already switched to full context integration. This suggests that while the RNN still firstly forms the line attractor and then the context integration,

these two phases overlap more. This is also reflected in the trajectories of the trial factors. For RNN20-4 the factors capturing context-integration related dynamics only start changing around when the first factor is peaking, fully separating the two stages. In contrast, for RNN20-3 the second trial factor follows closely after factor 1, as it would be expected if the two phases overlap. If the weights were sampled more often, this timing difference would likely be more clear and perhaps even show two close peaks for the factors.

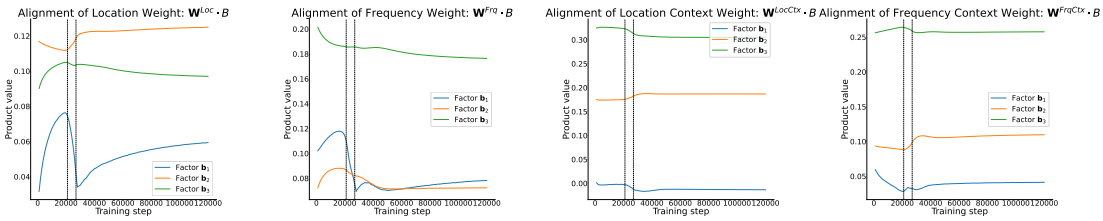
Similar observations can be seen for the other RNNs in Appendix D. In particular, for RNN100-1 (Figure D.4) the timing of the line attractors formation and context integration differs between the two contexts. The trial factors appear to pick up when the phase has been completed for both contexts. Moreover, RNN20-1's (Figure D.1) trial factors showed strong oscillation after the peak. These appear to be associated with the RNN having difficulties fully forming the context integration, leading to constant changes in weights and loss.

Overall, by considering performance matrices alongside the tensor decomposition insights were gained about the dynamics caused by the weight changes. It appears that the trial factors capture two different phases of the formation of the final dynamics: formation of the line attractor and context integration. The timing of these and how much they overlap differs between RNNs which is reflected in their trial factors.

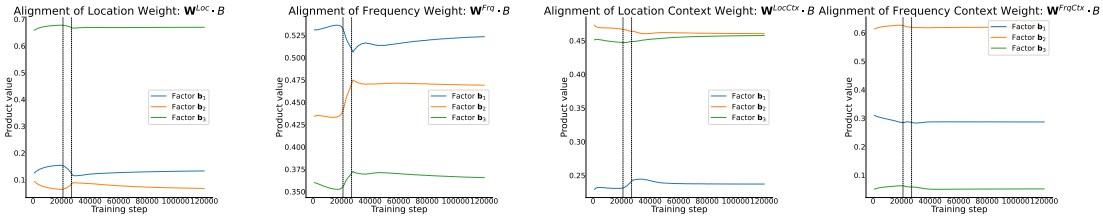
3.6 Factor alignment to input weights

To further investigate whether the assumption of the two-step process in the formation of the dynamics is correct, the input weights were considered. The integration of the evidence along the line attractor depends on the alignment of the selection vector and the linearised input for that context. The selection vector is a left eigenvector of the Jacobian and thus is related to the rows of the recurrent weight matrix. Similarly, the linearised inputs are associated with the input weights. Hence, it was hypothesised that this alignment will be reflected in the inner product of the input weights and the row factors of the decompositions, with different relationships for components capturing line attractor formation compared to context integration.

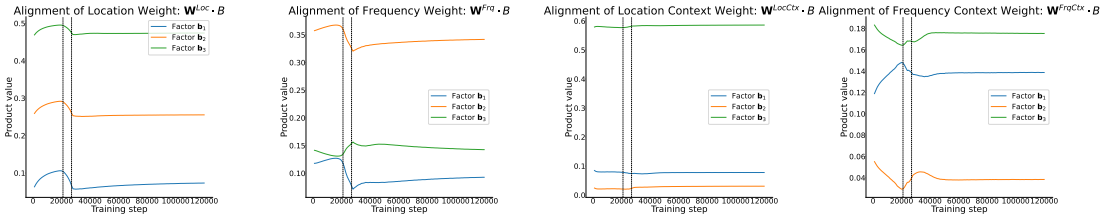
The RNNs have four different input weight vectors: $\boldsymbol{w}^{\mathrm{Loc}}, \boldsymbol{w}^{\mathrm{Frq}}$, and $\boldsymbol{W}^{\mathrm{Ctx}} = [\boldsymbol{w}^{\mathrm{LocCtx}}\ \boldsymbol{w}^{\mathrm{FrqCtx}}]$. Since these changed during learning the sampled, normalised input vectors at each trial point can be concatenated to the matrices $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}$, $\boldsymbol{W}^{\mathrm{FrqCtx}} \in \mathbb{R}^{K \times N}$. Taking the row factorisation matrix $\mathbf{B} \in \mathbb{R}^{N \times R}$ of a decomposition,



(a) Full Tensor



(b) Delta Tensor



(c) Tensor of Differences

Figure 3.9: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN20-4 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences. The vertical lines indicate the peaks of the trial factors of the Tensor of Differences.

the angle between the weights and factors across learning can be calculated by multiplying the corresponding matrices. For instance, the $\mathrm{r}^{\mathrm{th}} \in \{1, \ldots, \mathrm{R}\}$ column of the product $\boldsymbol{W}^{\mathrm{Loc}}\mathbf{B} \in \mathbb{R}^{\mathrm{K} \times \mathrm{R}}$ represents the alignment between the location input weight $\boldsymbol{w}^{\mathrm{Loc}}$ and the r^{th} row factor $\mathbf{b}_{\mathrm{r}}$ for each trial point k.

The results of these computations for the rank-3 decomposition of RNN20-4 for each tensor type can be seen in Figure 3.9 (see the same for the other RNNs in Appendix E). Each column corresponds to one of the input weights (Location, Frequency, Location Context, Frequency Context), with each of its lines being the alignment of that weight with one of the row factors over training. It can be observed that for all tensor types, the alignment between the weights and row factors change at the two peak points of the trial factors of the Tensor of Differences (training steps 21000 and 27000; vertical lines

in the plots). Since the row factors are constant, this indicates that the input weights change at similar points as the recurrent weights, which is likely related to the phases in the evolution of the dynamics.

When considering the alignments for the Delta tensor more closely some interesting patterns can be observed. $\mathbf{b}_3$ is aligned with the location and location context weights but not with the two frequency weights. The opposite is true for the second row factor $\mathbf{b}_2$, but the difference between the two stimuli types is less distinct. Lastly, $\mathbf{b}_1$ shows predominantly an alignment with the frequency input weights. When considering the row factors corresponding trial factors (Figure 3.3b), these alignments make intuitively sense. Component one changes its weights early (see trial factor $\mathbf{c}_1$ in Figure 3.3b) and is thus related to the formation of the line attractor. Hence, as would be expected, the weights for pulse integration show a stronger relationship with its row factor and change their alignment more with it than the context integration weights. In contrast, components two and three appear to be related to context integration (trial factors $\mathbf{c}_2$ and $\mathbf{c}_3$ in Figure 3.3b respectively) and thus have a stronger alignment for both pulse and context input. Moreover, it appears that each factor captures more changes for one of the stimuli types than the other to a certain extent. The strength of these correspondences varies, probably because the decomposition is only of rank three.

Overall these observations show that the two-phase assumption matches with the change in the input weights. While this computation can not capture exactly the angles between inputs and selection vector, since the non-linearised weights are considered and a low rank, overall patterns are observable that align with them. Besides the differences in the alignments of the weights to the components for the two different phases, it also appears as if every component is related more to one than the other stimuli type.

Chapter 4

Low tensor rank RNN

4.1 Methodology

The CP decomposition of the weights has given insights into the evolution of dynamics in the RNNs. However, identifying the correct rank for the decomposition was not straightforward. Consequently, lower ranks than the true ones were probably used. Some important changes in the dynamics might have been missed due to this. Yet using a higher rank would have made it harder to interpret the results and might have split up related dynamic characteristics into multiple components. This problem emerges from the fact that the CP decomposition tries to preserve the high rank. However, not all of the components might be necessary for the dynamics. In fact, it has been observed that low rank recurrent weight matrices capture complex dynamics in neural networks such as line attractors [25]. CP decomposition is not able to identify these dynamic relevant components since it simply aims to explain the variance in the weights.

To overcome this drawback, Pellegrino et al. [32] developed LtrRNN. This RNN is trained on reproducing neural activity, while its weight tensor is restricted to be of low tensor rank. Through this, the weight factors capture only dynamic relevant changes needed to reproduce the activity. The frontal slice of a 3^{rd}-order rank-R tensor $\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \in \mathbb{R}^{N \times N \times K}$ can be expressed as $\mathbf{X}_{::k} = \sum_{r=1}^{R} (\mathbf{a}_r \circ \mathbf{b}_r) c_r^{(k)}$ where $c_r^{(k)}$ is the k^{th} element in $\mathbf{c}_r$. Thus by restricting the recurrent weight to evolve during training in the R dimensional subspace of $\mathbb{R}^{N \times N}$ spanned by $\{vec(\mathbf{a}_r \circ \mathbf{b}_r)\}_{r=1}^{R}$, the resulting weight tensor is of rank R. LtrRNN uses a current rate model (Equation 2.3)

without the bias. The activity dynamics for trial k are hence

$$\tau\dot{\mathbf{x}}^{(k)} = -\mathbf{x}^{(k)} + \boldsymbol{W}_{::k}\phi(\mathbf{x}^{(k)}) + \mathbf{B}\mathbf{u}^{(k)}(t) \tag{4.1}$$

$$= -\mathbf{x}^{(k)} + \sum_{r=1}^{R} (\mathbf{a}_r \circ \mathbf{b}_r) c_r^{(k)} \phi(\mathbf{x}^{(k)}) + \mathbf{B}\mathbf{u}^{(k)}(t), \tag{4.2}$$

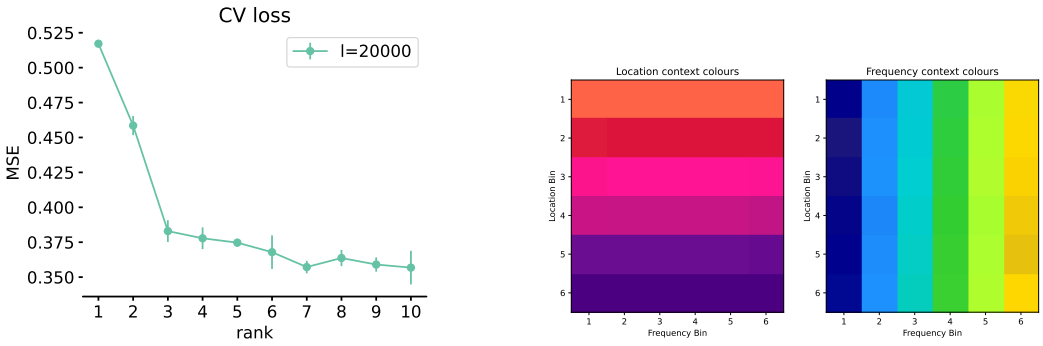
which are fitted to the given activity. $\mathbf{c}_r$ is also restricted to only change smoothly across trials to reflect learning better.

4.2 Data

The data used to train the LtrRNNs were the firing rates of the original RNNs trained by Pagan et al. [29]. For each trial point, the activity of all units at each time point across multiple trials was recorded which can be concatenated across trials to form a 3rd order tensor of size $\mathtt{Time} \times \mathtt{Trial} \times \mathtt{Unit}$. Two types of activity tensors were created, which differed in the inputs used for the simulated trials. Firstly, for the variable input activity tensor, the proportions between right and left/low- and high-frequency pulses were randomly chosen from a wide range of potential values, which resembled the inputs during training. To be able to gain insight into the different influences the proportions and thus their strength of evidence for a direction had on the activity, the proportions for both stimulus types for each trial were classed into one of six, equally sized bins (their strength of evidence reflecting the bins in Table 3.2). To be able to better capture the evidence strength dependent activities, the fixed input activity tensor was generated, where the inputs for each location/frequency bin were always the same with their proportions corresponding to Table 3.2 (see also Figure 2.1b). For both activity tensor types, the condition for each trial was then defined as [Context, Location Bin, Frequency Bin], leading to 72 possible different condition labels.

4.3 Variable Input Results

The LtrRNN was first trained with the variable input activity tensors, which consisted of 120 trials (one for each trial point). The activity of RNN20-4 was used to tune for different hyperparameters using cross-validation (see Appendix F.1). In particular, the mean squared error (MSE) between the LtrRNN's and RNN20-4's activities were computed for different ranks which results are shown in Figure 4.1a. It can be noted



(a) (b)

Figure 4.1: Rank selection and colouring of the conditions for the LtrRNN. (a) MSE between the LtrRNNs activity and the original activity of RNN20-4 for different ranks. The value $\mathrm{l} = 20000$ refers to the smoothness of the trial factor. (b) The colours used for the activities for each trial, based on the trial's condition [Context, Location Bin, Frequency Bin].

that at rank three there is a prominent elbow in the error. Thus, as was expected, the LtrRNN could capture the dynamics with a low rank weight tensor.

The results for the rank three LtrRNN for the activity of RNN20-4 can be seen in Figure 4.2 (the results for all other RNNs with 20 hidden units are in Appendix F.2). The activity $\mathbf{x}$ can be expressed in terms of the column factors $\mathbf{a}$ (see Equation 4.2), making it possible to visualise the low-dimensional activity by projecting $\mathbf{x}$ onto the factors [32]. These projections onto each factor can be seen in Figure 4.2 d and the 3D visualisation for all three of them in Figure 4.2 h. Each line corresponds to the activity for a trial at each time point and its colour represents the condition for that trial (see Figure 4.1b for the exact colour of each condition). When considering the 3D projection, it can be noted that the activities at the end of a trial (darkest colour) are sorted along two lines. One line captures the activities for trials in the location condition and the other for the frequency condition, with the activities sorted along the two lines based on the evidence strength for the relevant stimuli type. Hence, LtrRNN was able to capture the low-dimensional dynamics along the two line attractors. In contrast, using principal component analysis (PCA) (Figure 4.2 c) does not show any interpretable patterns.

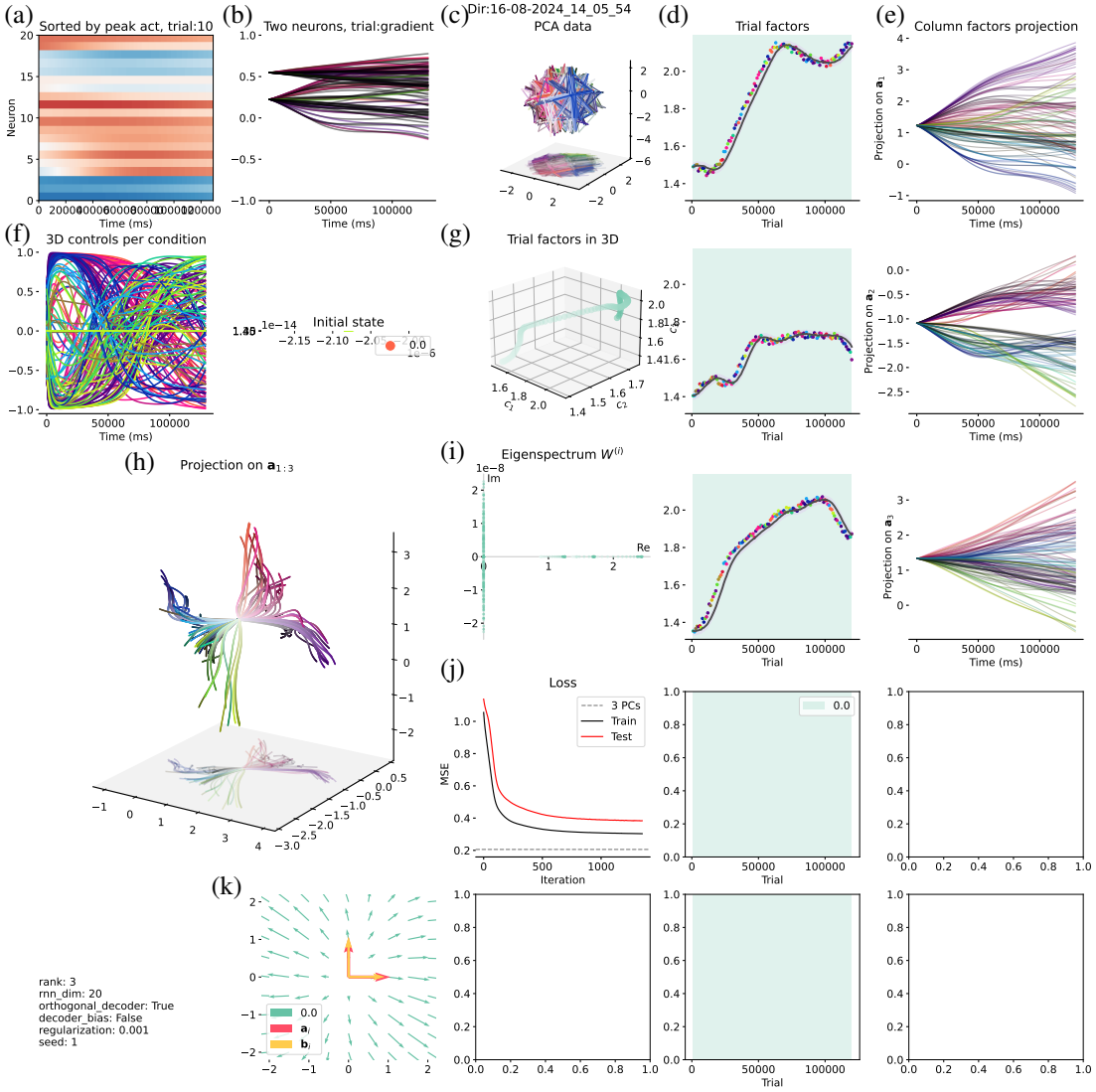


Figure 4.2: Results of using LtrRNN on the flexible input activity of RNN20-4 for rank three. The colours correspond to the conditions of the trials. The darker the colour the later in the trial/training it is. (a) Neurons of the RNN sorted by their peak. (b) The gradient of two neurons over time. (c) 3D visualisation of the activity projected onto the first 3 Principal Components of the weights. (d) Trial factors $\mathbf{c}_r$ of each of the components of the weight tensor. (e) Projection of the activity for each trial over time onto each of the column factors $\mathbf{a}_r$. (f) 3D controls of each condition over time. (g) Visualisation of the trial factors $\mathbf{c}_{1:3}$ in 3D. (h) 3D visualisation of the projection of the activities onto the column factors $\mathbf{a}_{1:3}$. (i) Eigendecomposition of the weight for each trial $\boldsymbol{W}_{::k}$. (j) Training and testing MSE of the activity during training. (k) Vector field of the activity in the subspace spanned by the column factors.

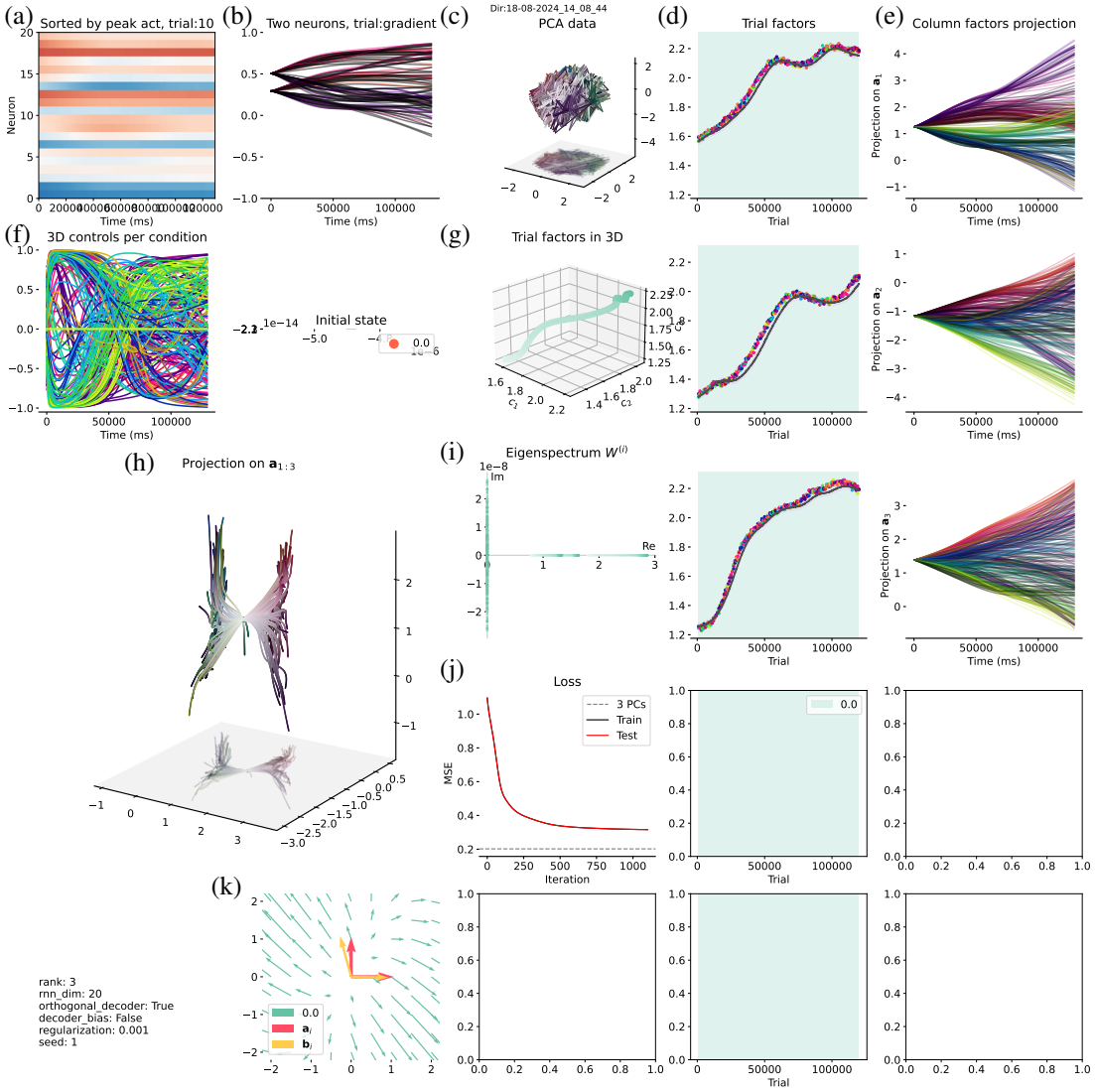


Figure 4.3: Results of using LtrRNN on the fixed input activity of RNN20-4 with rank three. The subfigures are the same as in Figure 4.2.

4.4 Fixed Input Results

LtrRNN assumes that the inputs are fixed for a specific condition. However, this is not the case for the original task where the inputs and their proportion varied from trial to trial. Thus the original RNNs' activity varies a lot for the same condition, which is not accounted for. For this reason, the fixed input activity tensors were created where the inputs for a specific condition were always the same. This allowed to plot more clearly the projections of each condition onto the column factors, especially for a higher trial number (three per trial point in this case).

Figure 4.3 shows the results for the fixed input for RNN20-4. The same dynamics

as before can be observed for the 3D projection of the activity onto the column factors. Considering the projection of each of the column factors (Figure 4.3 e), it appears that the first column factor $\mathbf{a}_1$ captures the two different contexts while the third ($\mathbf{a}_3$) relates to the strength of evidence. Their corresponding trial factors $\mathbf{c}_1$ and $\mathbf{c}_3$ show an earlier and stronger increase of the third component than the first. This aligns with the assumption that the RNN learned to first accumulate evidence (formation of line attractor) and then to integrate context. Hence, the activity dynamics of the RNN at first reflect only the evidence strength, captured by the third component, and then later on also the context, captured by the first component. The second component $\mathbf{a}_2$ appears to be a mixture of the two, also capturing evidence strength but having a similar trial factor $\mathbf{c}_2$ as the first component.

4.5 Comparison to CP decomposition results

Besides not accounting for flexible inputs and the weight tensors having different ranks, the original and LtrRNN models differ in other aspects that could influence the results. LtrRNN uses a continuous current rate model while the original RNNs were defined as discrete firing rate models. These lead to different linearised dynamics, which however are related since the two models can be mapped to each other [30, 28]. Furthermore, while in the LtrRNN the input weights are fixed over trial points and spanned by the column factors, they were learned by the original RNNs.

To better understand how these differences lead to discrepancies in the decomposition of the weights, the trial factors for both decomposition methods were plotted. For this purpose, the trial factors $\mathbf{c}'_r$ of the Tensor of Differences $\mathcal{W}'$ of the LtrRNN were computed based on the factors of the full weight tensor, $\mathcal{W} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$. The frontal slices of $\mathcal{W}'$ can be expressed as $\boldsymbol{W}'_{::k} = \boldsymbol{W}_{::k+1} - \boldsymbol{W}_{::k} = \sum_{r=1}^{R} (\mathbf{a}_r \circ \mathbf{b}_r) c_r^{(k+1)} - \sum_{r=1}^{R} (\mathbf{a}_r \circ \mathbf{b}_r) c_r^{(k)} = \sum_{r=1}^{R} (\mathbf{a}_r \circ \mathbf{b}_r)(c_r^{(k+1)} - c_r^{(k)})$. Hence, $c_r'^{(k)} = c_r^{(k+1)} - c_r^{(k)}$ for $k = \{1, \ldots, K-1\}$ and $c_r'^{(K)} = 0$.

For each of the RNNs with 20 hidden units, the trial factors for the Full and Difference Tensor for the LtrRNN and for the Tensor of Difference for the original RNN are plotted in Figure 4.4. While the factors for the Full Tensor for the LtrRNN do not appear related to the factors and their peaks of the original RNNs, some similarities can be seen between the Tensors of Differences. The LtrRNNs trial factors $\mathbf{c}'_r$ have peaks at similar points as the RNNs (vertical lines). Yet, for the LtrRNNs these are one of multiple peaks. This could be due to the LtrRNN picking up more changes in

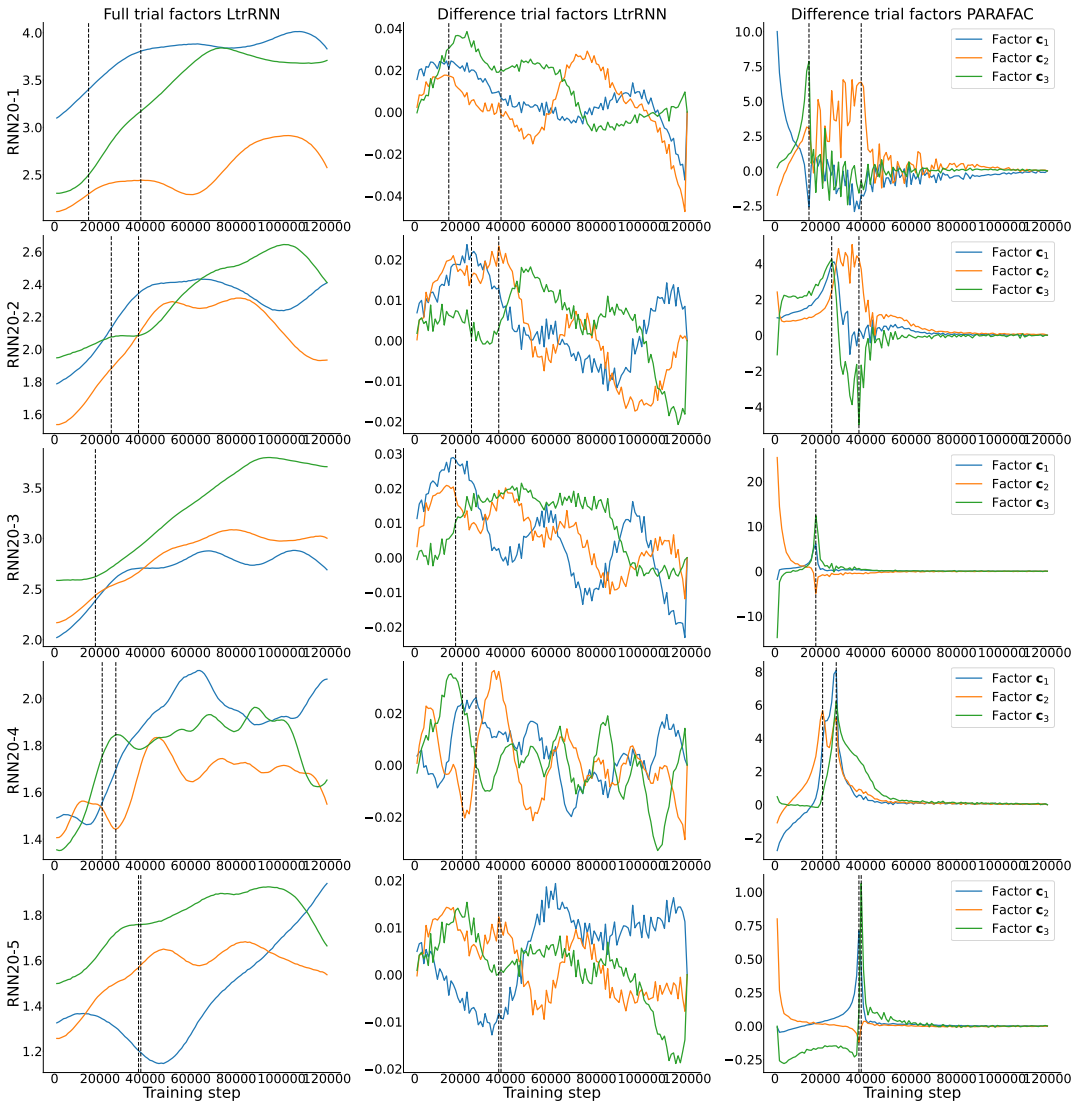


Figure 4.4: Comparison of trial factors for the Full Tensor for LtrRNN, Tensor of Differences for LtrRNN, and Tensor of Differences for the original RNN for all RNNs with 20 hidden units. Both RNNs used a rank three decomposition. For LtrRNN the variable input activity tensor was used with 120 trials.

the weights, that CP was not able to capture without a much higher rank. For example, the change in evidence and context integration dynamics for each context and stimuli type. However, a different reason could be the fixed input vectors. Since the LtrRNN cannot vary these, any differences in the activities due to changes in the input integration during learning must be captured by changes in the recurrent weights. Thus, the weights change more than for the original RNNs.

Chapter 5

Discussion

By applying tensor decomposition to task-trained RNNs, insights were gained into how their dynamics evolve while learning to solve a context-dependent decision-making task. The decomposition of the recurrent weights revealed that the dynamics go through two phases. Firstly, they learn to integrate the evidence strength for both stimuli types. This is accomplished by forming two line attractors, i.e. lines of fixed points which Jacobians have an eigenvalue that is close to 0 and the other ones have negative, real parts. In the second step, they then adapted the dynamics around the attractors so that only context-relevant stimuli cause movement along them, by changing the left eigenvector corresponding to the 0 eigenvalue of the Jacobians.

These dynamics change because the weights cross bifurcation boundaries in the parameter space during learning. A crossing during gradient-based optimisation that influences the current neural dynamics has been associated with explosive and vanishing gradients [13, 14, 31, 33]. Gradient explosion is when the gradient strongly increases, leading to large changes in the weights and loss [5, 31]. This linkage can explain why the weights exhibit sudden prominent changes instead of slow continuous ones to form the dynamics. During learning, there appears to come a point where the small changes in weights cause the neural activity to come within the basin of the fixed points that have started to form. This causes a large change in gradients and weights, which leads to the formation of the full line attractors and thus reduces the loss.

The formation of the line attractors divides the training process into two stages [16]. Firstly, the model is in the exploration stage where it passes different attractor landscapes without a large change in loss until the line attractors are found. It then enters the refinement phase where it adapts the dynamics around the attractors, i.e. learns to integrate the context, which is associated with a decrease in the loss. This

two-step process has been noted both when considering bifurcation [16] as well as when studying the loss in neural networks during stochastic gradient-decent [1]. Achille et al. [1] observed that a network during training crosses through loss landscapes with high curvatures, which they refer to as bottlenecks, until a flat loss basin is found which the model stays in. These bottlenecks could be interpreted from the dynamic system perspective as the attractor formations that the model passes, with the flat minima being the task-relevant dynamic landscape in the parameter space. Further investigations into this relationship might reveal insight into the types of dynamics that RNNs tend to learn due to gradient descent.

It can be noted that RNNs which exhibit the second weight peak for context integration, did not display an associated prominent change in the loss trajectory. This type of change in the dynamics was termed 'hidden bifurcation' by Haputhanthri et al. [16]. They compared the singular values of the weights of task-trained RNNs between time points, observing that the peaks in weight change are associated with bifurcations but not necessarily with a permanent shift in loss trajectory. While in their task the hidden bifurcation was found during the exploration stage, here it was shown that it can also occur during the refinement stage. This highlights the advantage of considering the weights to understand changes in the dynamics since the context-related bifurcation might have not been identified when considering only the loss.

Tensor decomposition has several advantages compared to matrix decomposition methods to investigate these changes in weights [11, 38, 45]. Firstly, tensor decomposition allows for the row and column factors to be the same for each training point, with the trial factor scaling them. In contrast, the changes observed between time points in Haputhanthri et al. [16] when using singular value decomposition (SVD) could also have been due to the singular vectors themselves changing between weights and not just the value. Moreover, tensor decomposition has weak constraints for uniqueness and its factors do not have to be necessarily orthogonal, as opposed to SVD [19].

These properties made it possible for the components of the tensor to represent different bifurcations and to visualise their trajectories over time. This is particularly useful when using LtrRNN since it overcomes the obstacle of the high-rank weight tensor that hinders CP decomposition by forcing the number of components to be low. Through this, all of its components capture different dynamic relevant changes. The role of each component can then be investigated in more detail, by projecting the low-dimensional dynamics onto them, which were found to be more interpretable than using standard dimension reduction methods such as PCA [32]. Even though

LtrRNN used a different model than the original RNN and thus did not have the exact same dynamics, it was still able to capture its main characteristics. This aligns with Maheswaranathan et al. [23] finding that RNNs with different models exhibit similar dynamic structures such as their fixed points and the linearised dynamics around them. Pagan et al. [30] also showed that the linearised dynamics between the discreet versions of the current and firing rate models are related. It is expected that changing the LtrRNN to be the same as the original RNNs, in particular allowing it to account for varying input and learning of input weights, will give even deeper insights into the evolution of the dynamics. Moreover, LtrRNN can also be applied directly to the neural activity of animals. Through this, the emergence of the wider solution space of the dynamics that are assumed for the rats in Pagan et al [29] can be explored directly. This will potentially help overcome the current drawback of gradient descent in task-trained RNNs restricting their found dynamics.

In general, while gradient descent might not be representative of biological learning [4], relevant insights about individual differences in the dynamics of neural networks were still gained. In particular, the timing of the two phases and how much they overlapped, differed between RNNs. This is likely due to where in the parameter space they are initialised and thus how far away the task-relevant dynamics are in the space. These differences are likely to be even greater in animals since biological networks consist of a higher number of neurons and thus potential dynamics. Under this assumption, the initial position in the parameter space probably also accounts for why some animals and RNNs are never able to learn a task.

Yet, the similarities observed in the RNNs dynamics might suggest a potential solution to this problem. The line attractors appear to always form before the context integration. Thus the RNNs might learn more effectively by first being trained to integrate the evidence of the stimuli so that it can find the correct dynamic landscape without any additional input that might distract from it (the context). Once the exploration phase is finished, the context can be added so that it finds the full solution during refinement. This is hypothesised to reduce heterogeneity and help RNNs to learn quickly. This assumption aligns with the success of newer training algorithms that emphasized considering the dynamics instead of just general loss minimisation, such as [16].

Indeed, this procedure of breaking the task down into sub-steps to help learn the task is normally how animals are trained. In fact, Pagan et al. [29] taught the decision-making task to the rats by first having them learn to associate the two stimuli types with the correct directions and then as a second step consider the context alongside them.

Thus, the two dynamic phases that the RNNs underwent to learn the task are the same ones as the ones used to teach the rats. This analogy highlights the theoretical insights that can be gained about learning in biological networks by studying the evolution of RNNs dynamics. By considering how they evolve, the steps required in learning can be better understood through the dynamic system perspective and can potentially be used for teaching more complicated tasks to animals and humans.

Chapter 6

Conclusion

To better understand how neural activity leads to computation and behaviour, the evolution of neural dynamics during learning was considered using tensor decomposition. These revealed that the formation of the dynamic system consists of multiple phases of learning to integrate different, relevant features needed to solve the task, which were captured by the components of the tensor decomposition.

The order of these steps gives a theoretical understanding of training protocols for both artificial and biological networks and how to improve them. Moreover, how much the phases overlap and their timing depend on individual characteristics of the networks. While the RNN only used a sub-sample of the potential solutions, their variations in how the dynamics evolved offer a potential explanation for the difference observed in context-dependent decision-making, both between as well as within the same task.

Overall, these results emphasise that to understand computation through neural activity, the observed dynamic systems have to be considered as one of many, equally valid solutions, instead of a single, exclusive answer. Hence the focus has to shift away from detecting the one final system, and instead concentrate on understanding how the dynamics get shaped and determined during learning.

Bibliography

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

[2] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current opinion in neurobiology*, 46:1–6, 2017.

[3] Joao Barbosa, Remi Proville, Chris C Rodgers, Michael R DeWeese, Srdjan Ostojic, and Yves Boubenec. Flexible selection of task-relevant features through population gating. *bioRxiv*, pages 2022–07, 2022.

[4] Guillaume Bellec, Franz Scherr, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049*, 2019.

[5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[6] Rasmus Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.

[7] Rasmus Bro. Multi-way analysis in the food industry. *Models, Algorithms, and Applications. Academish proefschrift. Dinamarca*, 1998.

[8] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in parafac models. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(5):274–286, 2003.

[9] Carlos D Brody and Timothy D Hanks. Neural underpinnings of the evidence accumulator. *Current opinion in neurobiology*, 37:149–157, 2016.

[10] J Douglas Carroll and JihJie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[11] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE signal processing magazine*, 32(2):145–163, 2015.

[12] Fengyu Cong, Qiu-Hua Lin, Li-Dan Kuang, Xiao-Feng Gong, Piia Astikainen, and Tapani Ristaniemi. Tensor decomposition of eeg signals: a brief review. *Journal of neuroscience methods*, 248:59–69, 2015.

[13] Kenji Doya. Bifurcations of recurrent neural networks in gradient descent learning. *IEEE Transactions on neural networks*, 1(75):218, 1993.

[14] Lukas Eisenmann, Zahra Monfared, Niclas Göring, and Daniel Durstewitz. Bifurcations and loss jumps in rnn training. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Sofia Fernandes, Hadi Fanaee-T, and João Gama. Normo: A new method for estimating the number of components in cp tensor decomposition. *Engineering Applications of Artificial Intelligence*, 96:103926, 2020.

[16] Udith Haputhanthri, Liam Storan, Yiqi Jiang, Adam Shai, Hakki Orhun Akengin, Mark Schnitzer, Fatih Dinc, and Hidenori Tanaka. Why do recurrent neural networks suddenly learn? bifurcation mechanisms in neuro-inspired short-term memory tasks. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

[17] Richard A. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[18] Johan Håstad. Tensor rank is np-complete. *Journal of algorithms*, 11(4):644–654, 1990.

[19] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[20] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *Journal of Machine Learning Research (JMLR)*, 20(26), 2019.

[21] Joseph B Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

[22] DJ Louwerse, Age K Smilde, and Henk AL Kiers. Cross-validation of multiway component models. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 13(5):491–510, 1999.

[23] Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. *Advances in neural information processing systems*, 32, 2019.

[24] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013.

[25] Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018.

[26] Earl K Miller. The prefontral cortex and cognitive control. *Nature reviews neuroscience*, 1(1):59–65, 2000.

[27] Earl K Miller and Jonathan D Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.

[28] Kenneth D Miller and Francesco Fumarola. Mathematical equivalence of two common forms of firing rate models of neural networks. *Neural computation*, 24(1):25–31, 2012.

[29] Marino Pagan, Vincent D Tang, Mikio C Aoi, Jonathan W Pillow, Valerio Mante, David Sussillo, and Carlos D Brody. A new theoretical framework jointly explains behavioral and neural variability across subjects performing flexible decision-making. *bioRxiv*, pages 2022–11, 2022.

[30] Marino Pagan, Adrian Valente, Srdjan Ostojic, and Carlos D Brody. Brief technical note on linearizing recurrent neural networks (rnns) before vs after the pointwise nonlinearity. *arXiv:2309.04030*, 2023.

[31] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[32] Arthur Pellegrino, N Alex Cayco Gajic, and Angus Chadwick. Low tensor rank learning of neural dynamics. *Advances in Neural Information Processing Systems*, 36:11674–11702, 2023.

[33] Alexander Rehmer and Andreas Kroll. The effect of the forget gate on bifurcation boundaries and dynamics in recurrent neural networks and its implications for gradient-based optimization. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2022.

[34] Chris C Rodgers and Michael R DeWeese. Neural correlates of task switching in prefrontal cortex and primary auditory cortex in a novel stimulus selection task for rodents. *Neuron*, 82(5):1157–1170, 2014.

[35] Zuzana Rošt'áková and Roman Rosipal. Determination of the number of components in the parafac model with a nonnegative tensor structure: A simulated eeg data study. *Neural Computing and Applications*, 34(17):14793–14805, 2022.

[36] Ryo Sasaki and Takanori Uka. Dynamic readout of behaviorally relevant signals from area mt during task switching. *Neuron*, 62(1):147–157, 2009.

[37] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[38] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on signal processing*, 65(13):3551–3582, 2017.

[39] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[40] David Sussillo. Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25:156–163, 2014.

[41] 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.

[42] Marieke E Timmerman and Henk AL Kiers. Three-mode principal components analysis: Choosing the numbers of components and sensitivity to local optima. *British journal of mathematical and statistical psychology*, 53(1):1–16, 2000.

[43] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43(1):249–275, 2020.

[44] Ralf D Wimmer, L Ian Schmitt, Thomas J Davidson, Miho Nakajima, Karl Deisseroth, and Michael M Halassa. Thalamic control of sensory selection in divided attention. *Nature*, 526(7575):705–709, 2015.

[45] Ali Zare, Alp Ozdemir, Mark A Iwen, and Selin Aviyente. Extension of pca to higher order data structures: An introduction to tensors, tensor decompositions, and tensor pca. *Proceedings of the IEEE*, 106(8):1341–1358, 2018.

Appendix A

Evaluation results for each RNN and tensor type

	RNN20-1		
	Full	Delta	Differences
Cross Validation	13	5	>50
Fit Threshold	10	5	12
Fit Elbow	14	8	11
Diffit	44	32	36
CORCONDIA	1	2	2
NORMO	17	3	3

Table A.1: Best rank for RNN20-1 for different tensor types and evaluation methods.

	RNN20-2		
	Full	Delta	Differences
Cross Validation	13	4	¿50
Fit Threshold	11	4	9
Fit Elbow	14	6	11
Diffit	39	36	41
CORCONDIA	1	2	2
NORMO	13	5	2

Table A.2: Best rank for RNN20-2 for different tensor types and evaluation methods.

	RNN20-3		
	Full	Delta	Differences
Cross Validation	14	5	42
Fit Threshold	11	4	8
Fit Elbow	15	8	10
Diffit	40	37	32
CORCONDIA	1	2	2
NORMO	12	3	2

Table A.3: Best rank for RNN20-3 for different tensor types and evaluation methods.

	RNN20-4		
	Full	Delta	Differences
Cross Validation	13	4	37
Fit Threshold	11	4	7
Fit Elbow	15	6	8
Diffit	41	45	43
CORCONDIA	1	2	1
NORMO	15	3	1

Table A.4: Best rank for RNN20-4 for different tensor types and evaluation methods.

	RNN20-5		
	Full	Delta	Differences
Cross Validation	14	4	31
Fit Threshold	11	3	6
Fit Elbow	14	9	9
Diffit	24	30	22
CORCONDIA	1	2	3
NORMO	15	2	3

Table A.5: Best rank for RNN20-5 for different tensor types and evaluation methods.

	RNN100-1		
	Full	Delta	Differences
Cross Validation	67	6	58
Fit Threshold	53	5	12
Fit Elbow	37	10	9
Diffit	97	73	93
CORCONDIA	2	2	3
NORMO	5	5	4

Table A.6: Best rank for RNN100-1 for different tensor types and evaluation methods

	RNN100-2		
	Full	Delta	Differences
Cross Validation	67	6	37
Fit Threshold	53	5	13
Fit Elbow	34	8	11
Diffit	95	57	83
CORCONDIA	2	2	2
NORMO	13	4	2

Table A.7: Best rank for RNN100-2 for different tensor types and evaluation methods

	RNN100-3		
	Full	Delta	Differences
Cross Validation	67	5	41
Fit Threshold	53	5	13
Fit Elbow	36	9	13
Diffit	99	64	92
CORCONDIA	1	3	2
NORMO	8	4	2

Table A.8: Best rank for RNN100-3 for different tensor types and evaluation methods

	RNN100-4		
	Full	Delta	Differences
Cross Validation	67	5	46
Fit Threshold	53	5	12
Fit Elbow	39	8	10
Diffit	93	99	82
CORCONDIA	1	2	1
NORMO	16	4	1

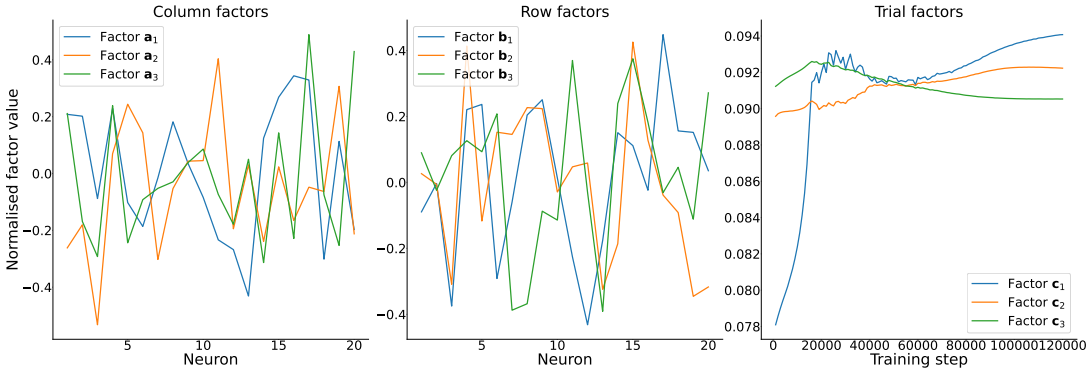
Table A.9: Best rank for RNN100-4 for different tensor types and evaluation methods

	RNN100-5		
	Full	Delta	Differences
Cross Validation	66	6	64
Fit Threshold	53	5	14
Fit Elbow	41	9	11
Diffit	92	46	97
CORCONDIA	1	2	2
Normo	11	4	1

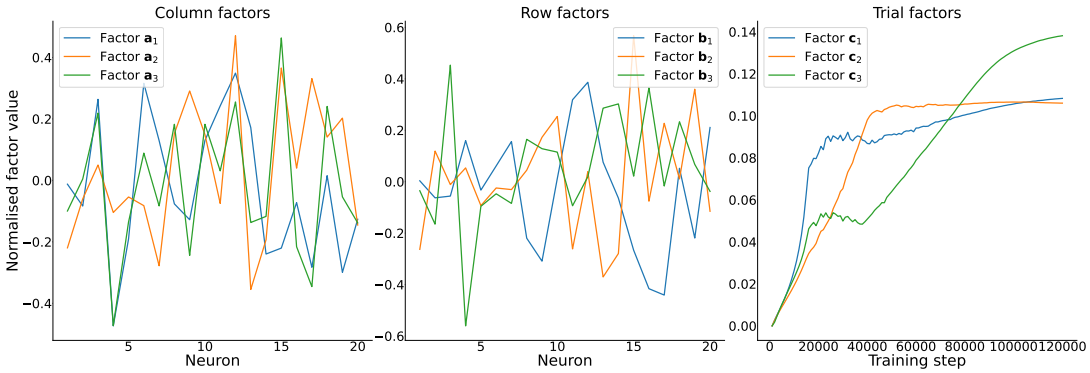
Table A.10: Best rank for RNN100-5 for different tensor types and evaluation methods

Appendix B

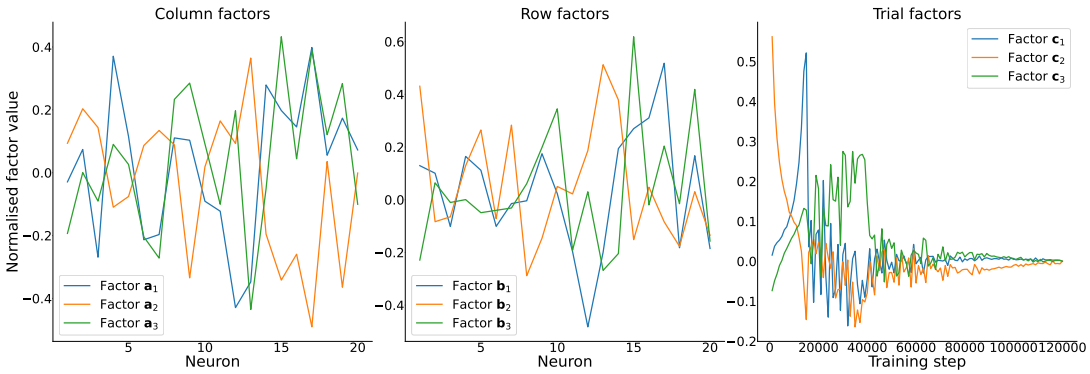
All factors for the rank three decomposition of each RNN



(a) Full Weight Tensor

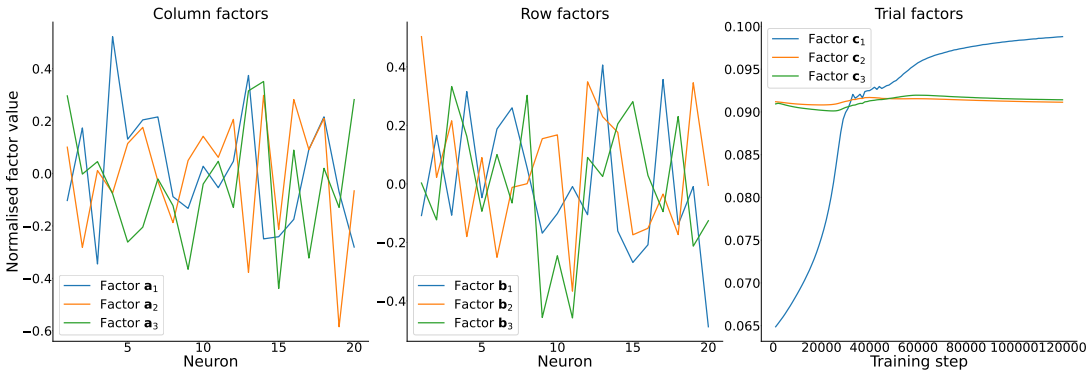


(b) Delta Weight Tensor

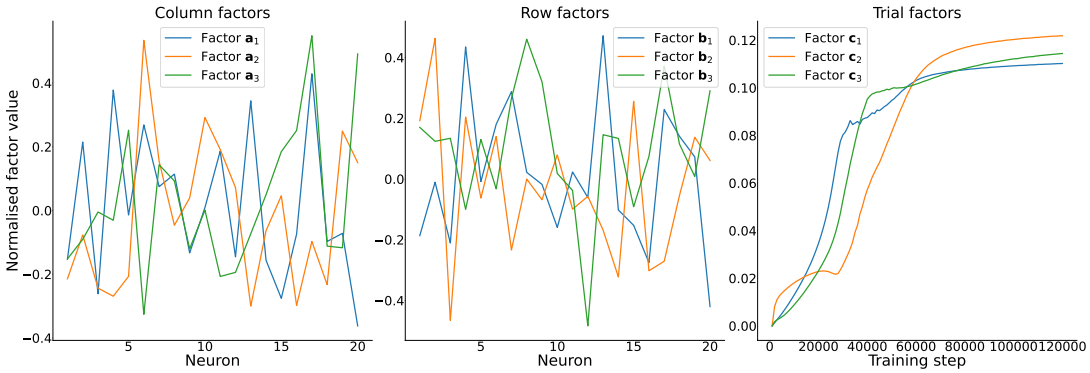


(c) Weight Tensor of Differences

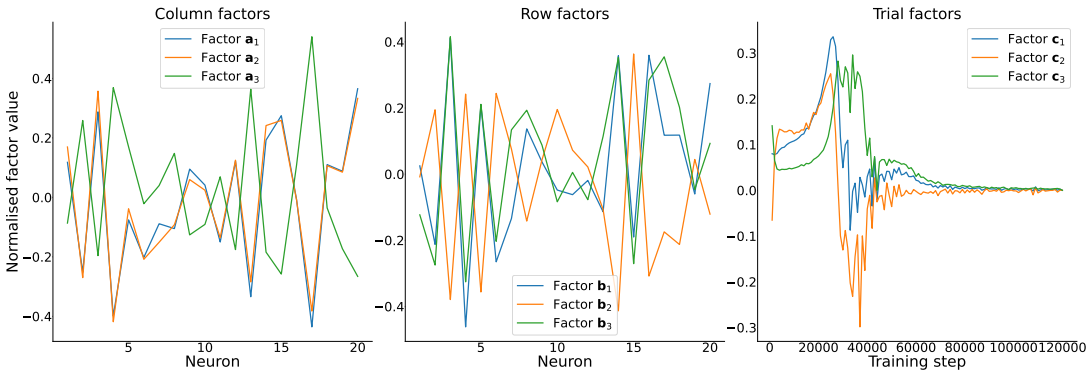
Figure B.1: Rank three decomposition results for RNN20-1. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor

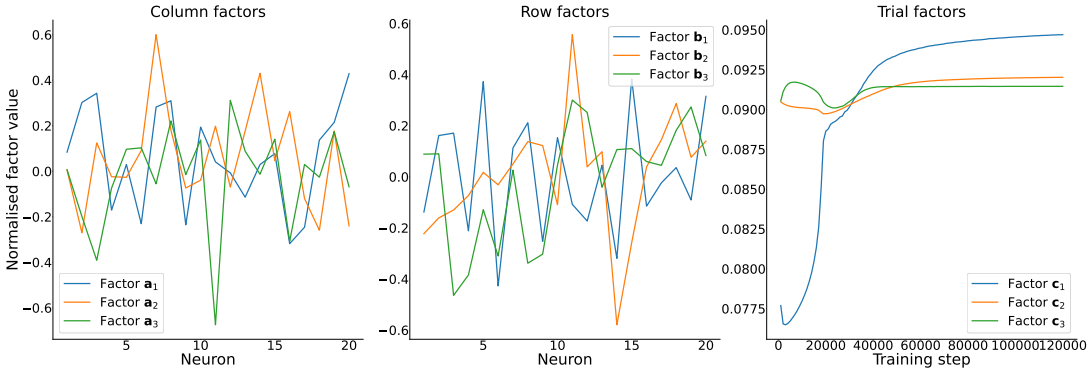


(b) Delta Weight Tensor

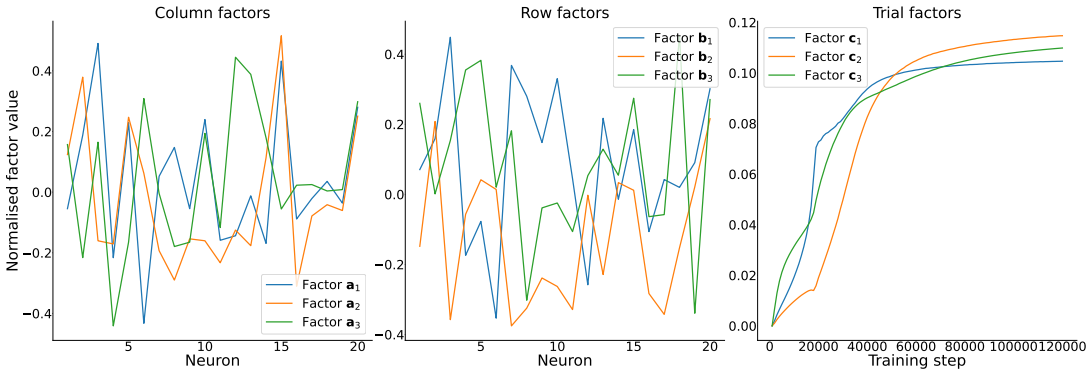


(c) Weight Tensor of Differences

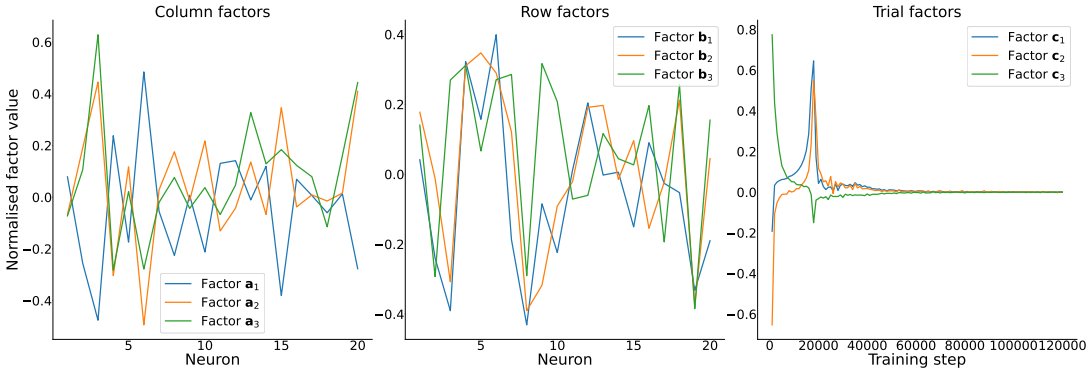
Figure B.2: Rank three decomposition results for RNN20-2. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor

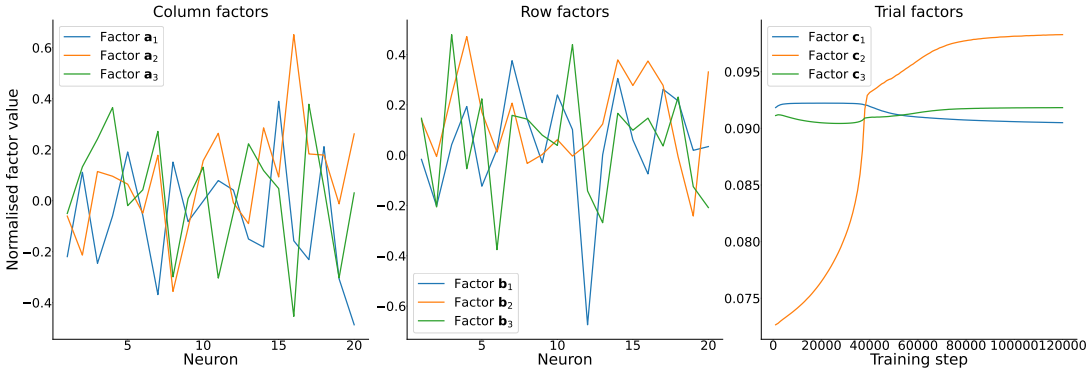


(b) Delta Weight Tensor

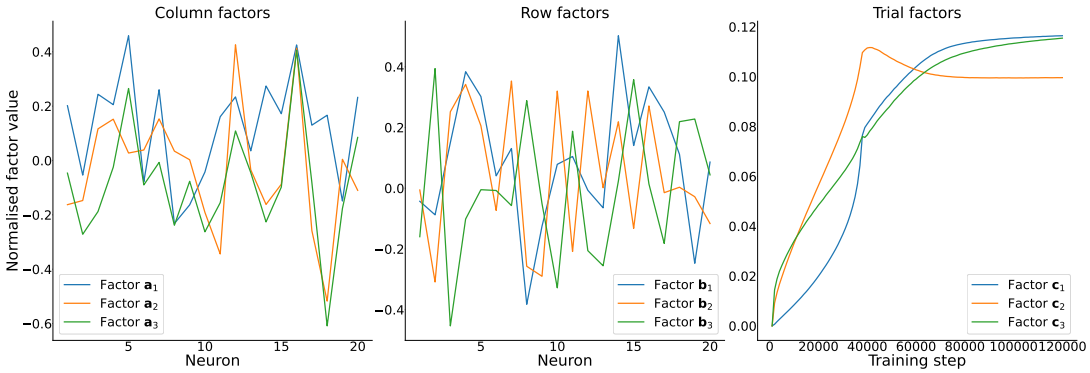


(c) Weight Tensor of Differences

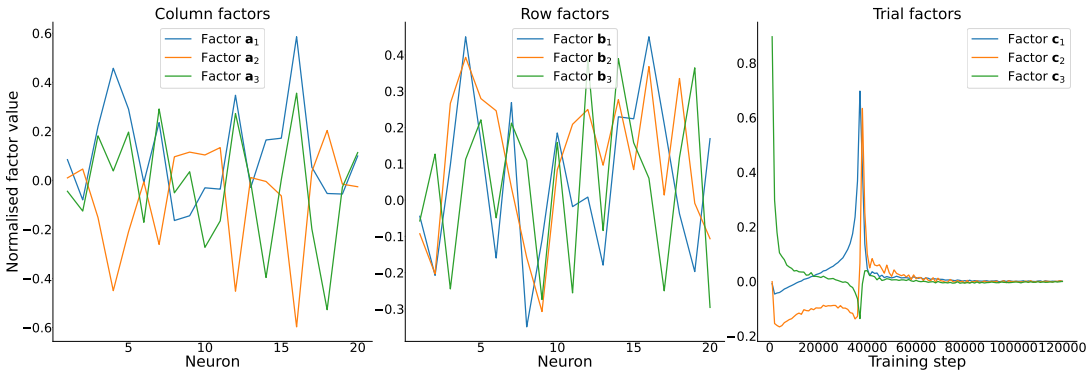
Figure B.3: Rank three decomposition results for RNN20-3. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor

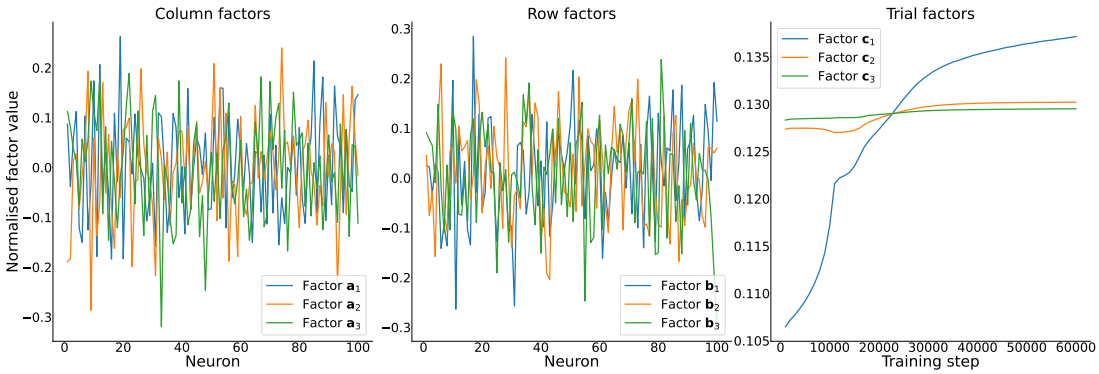


(b) Delta Weight Tensor

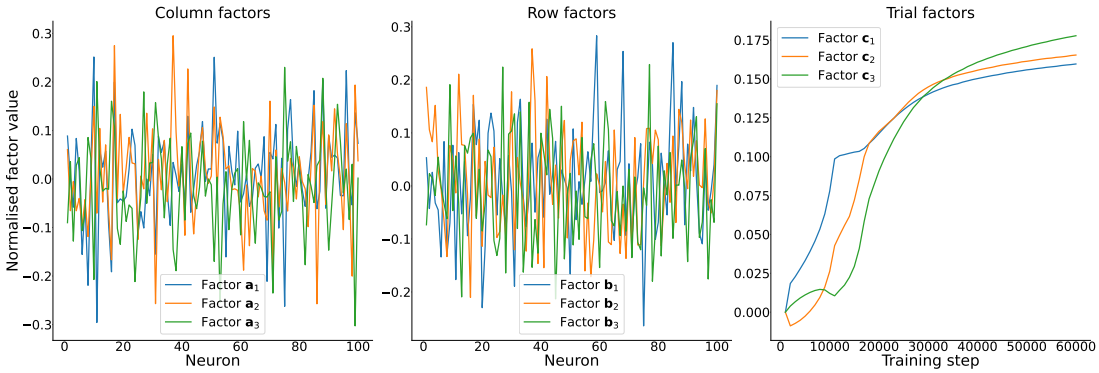


(c) Weight Tensor of Differences

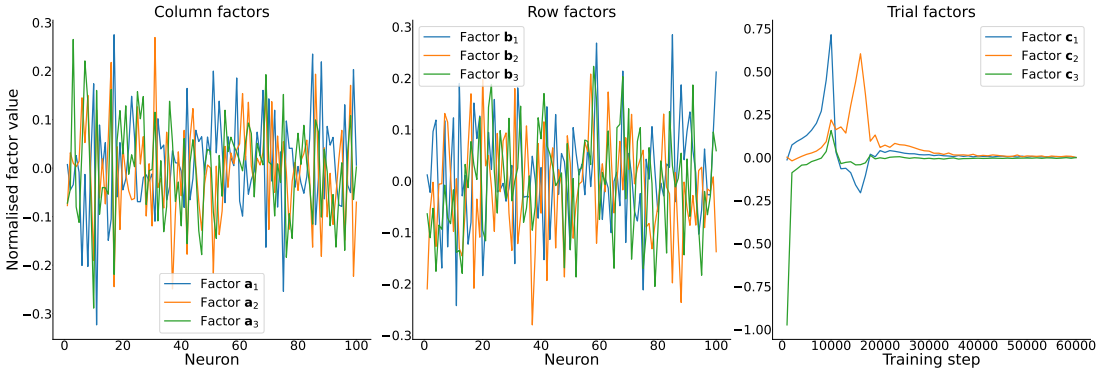
Figure B.4: Rank three decomposition results for RNN20-5. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor

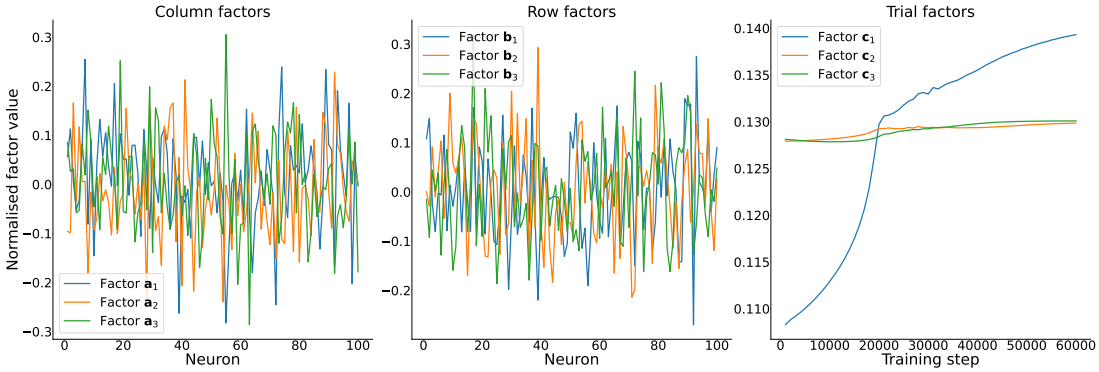


(b) Delta Weight Tensor

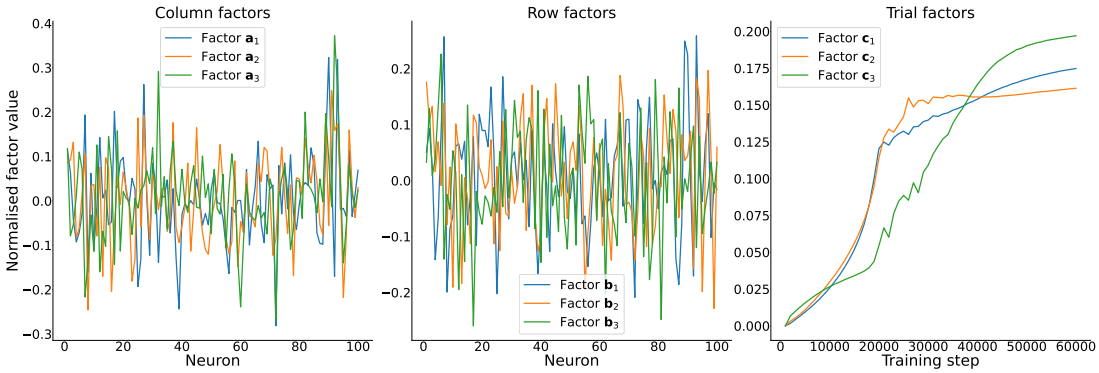


(c) Weight Tensor of Differences

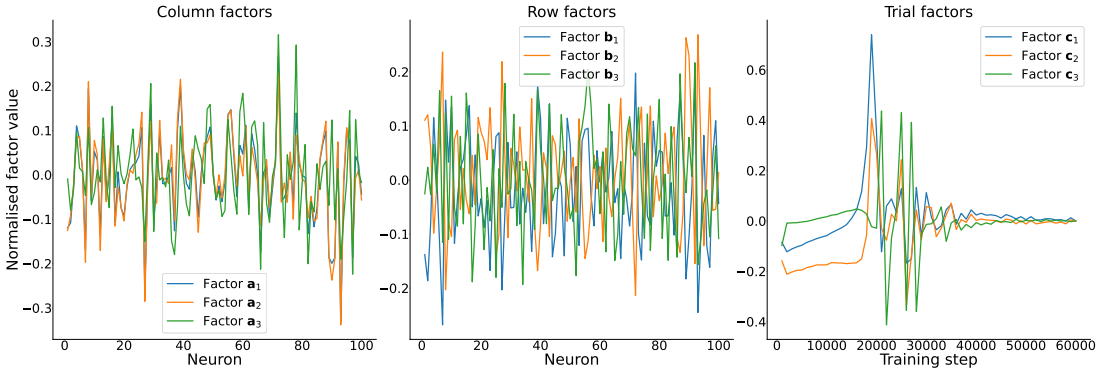
Figure B.5: Rank three decomposition results for RNN100-1. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor

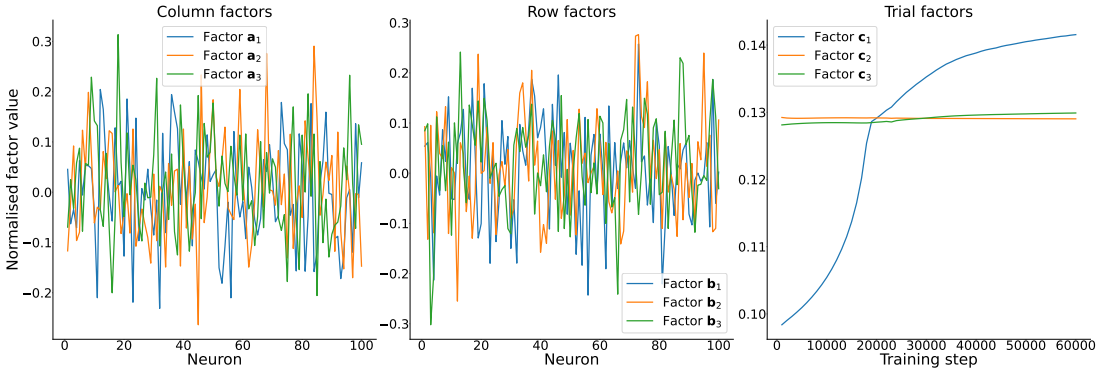


(b) Delta Weight Tensor

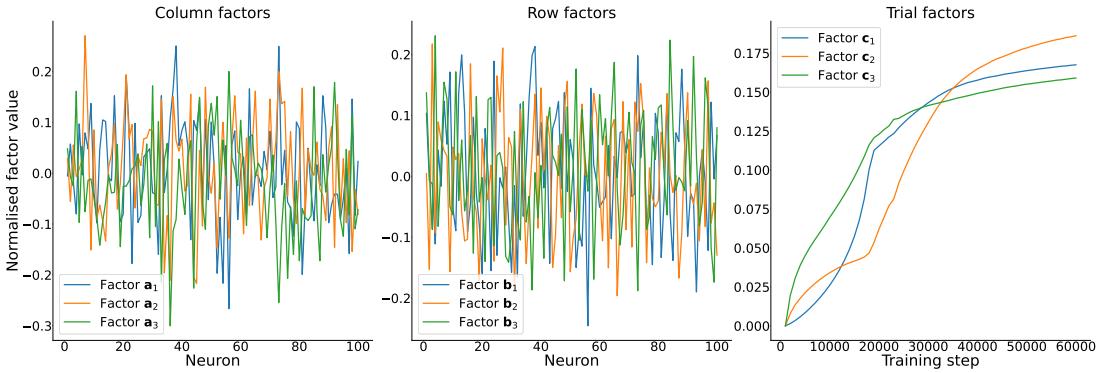


(c) Weight Tensor of Differences

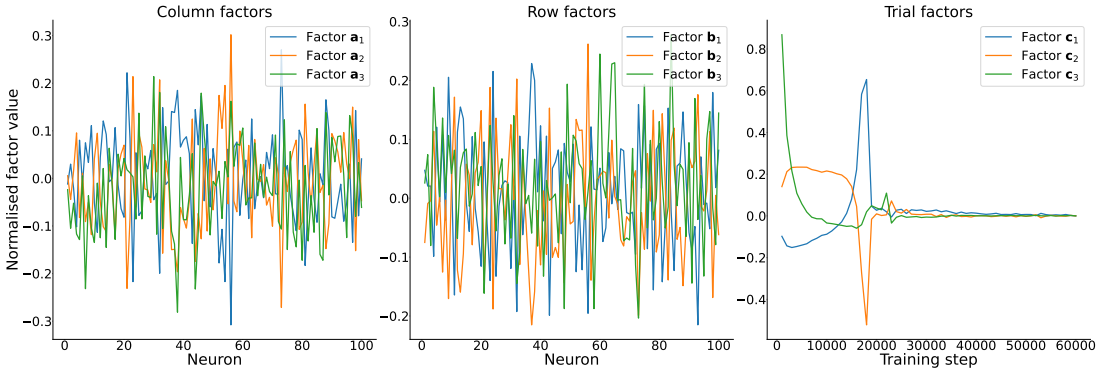
Figure B.6: Rank three decomposition results for RNN100-2. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor

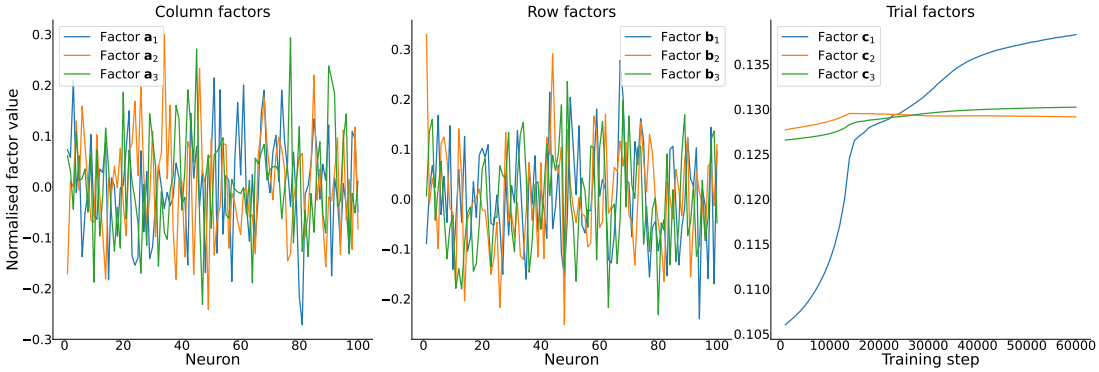


(b) Delta Weight Tensor

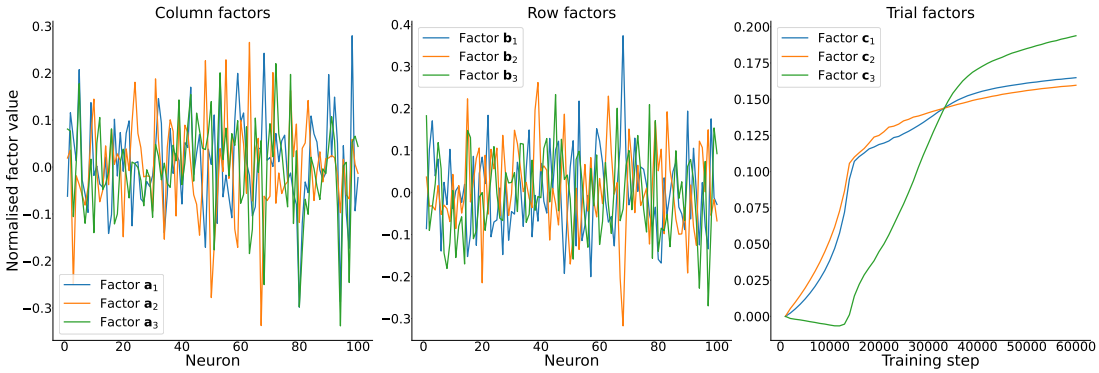


(c) Weight Tensor of Differences

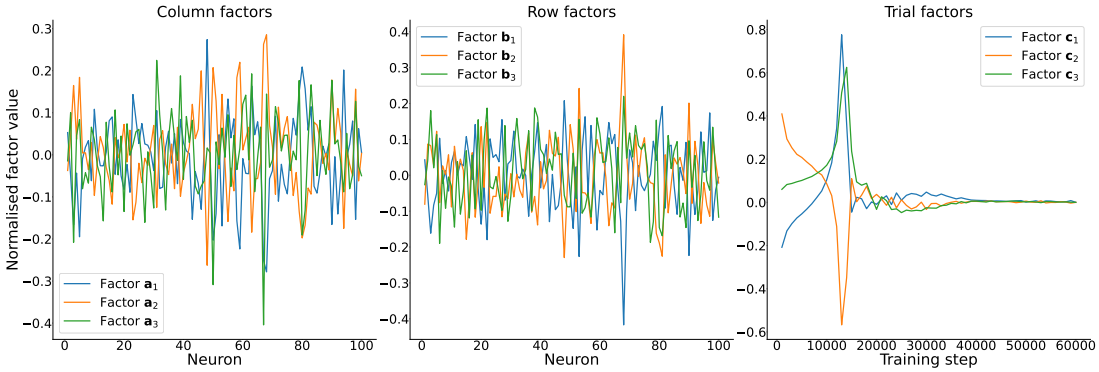
Figure B.7: Rank three decomposition results for RNN100-3. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor

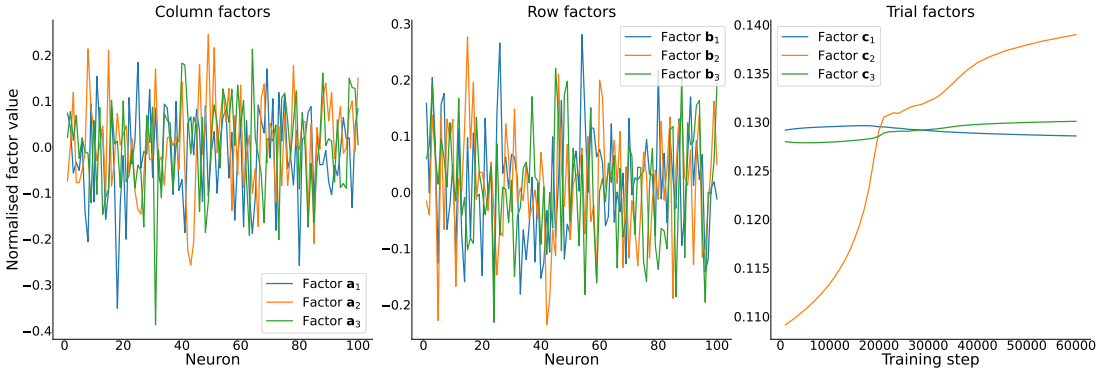


(b) Delta Weight Tensor

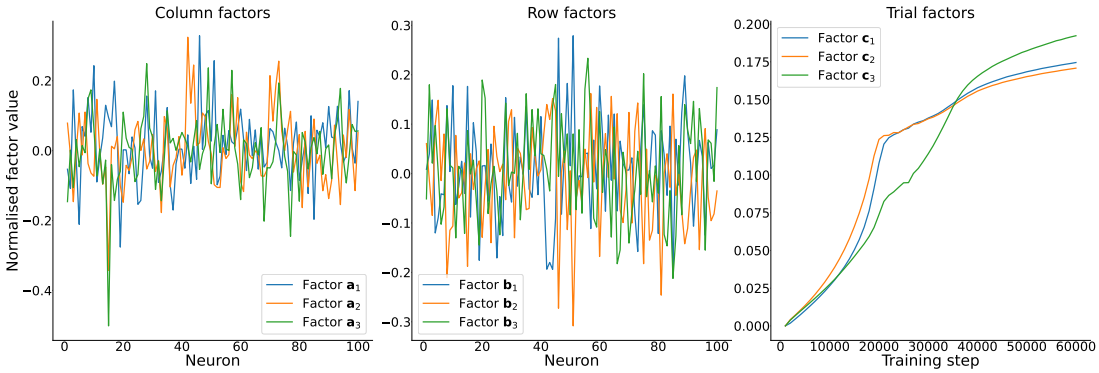


(c) Weight Tensor of Differences

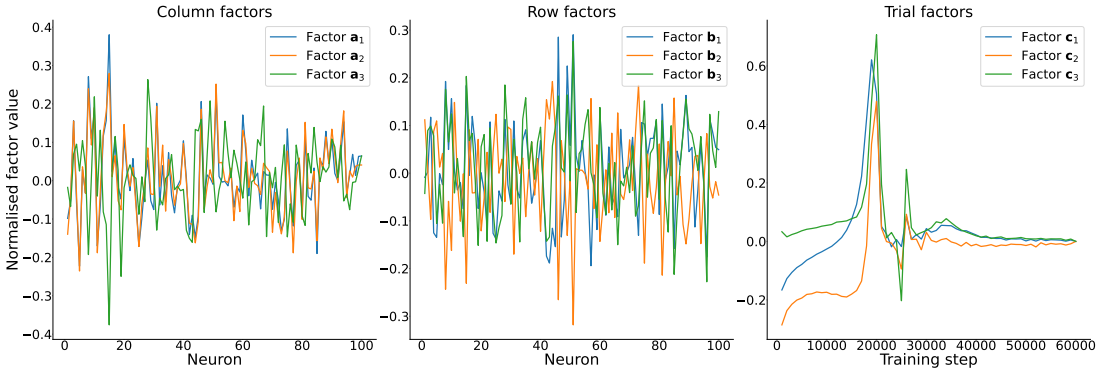
Figure B.8: Rank three decomposition results for RNN100-4. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.



(a) Full Weight Tensor



(b) Delta Weight Tensor



(c) Weight Tensor of Differences

Figure B.9: Rank three decomposition results for RNN100-5. Each subplot shows all of the resulting factors for all three modes (row,column,trial) for one of the tensor types of the recurrent weights of the RNN: (a) Full Tensor, (b) Delta Tensor, and (c) Tensor of Differences.

Appendix C

Trial factors for different ranks for each RNN

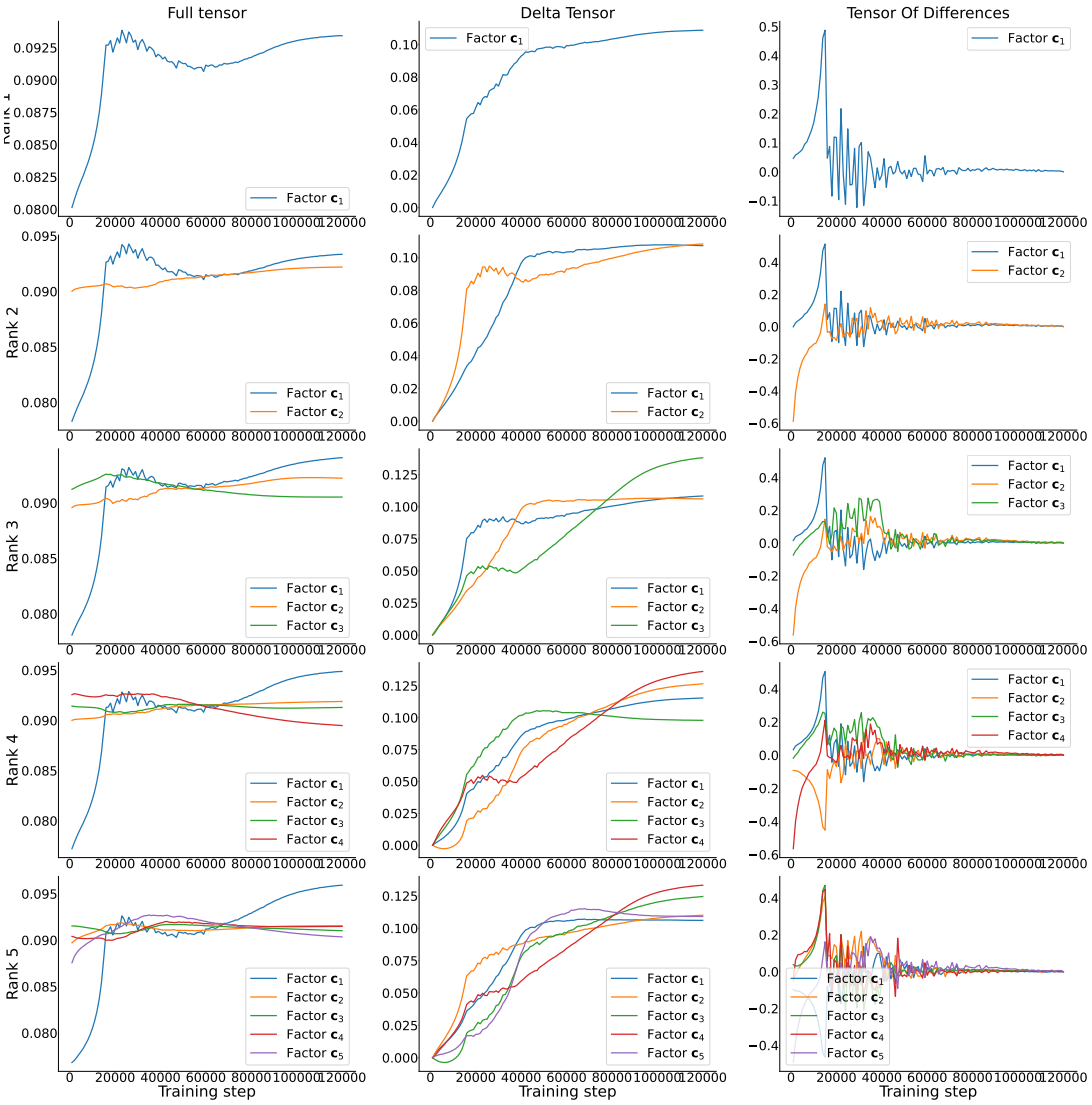


Figure C.1: Trial factors of RNN20-1 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

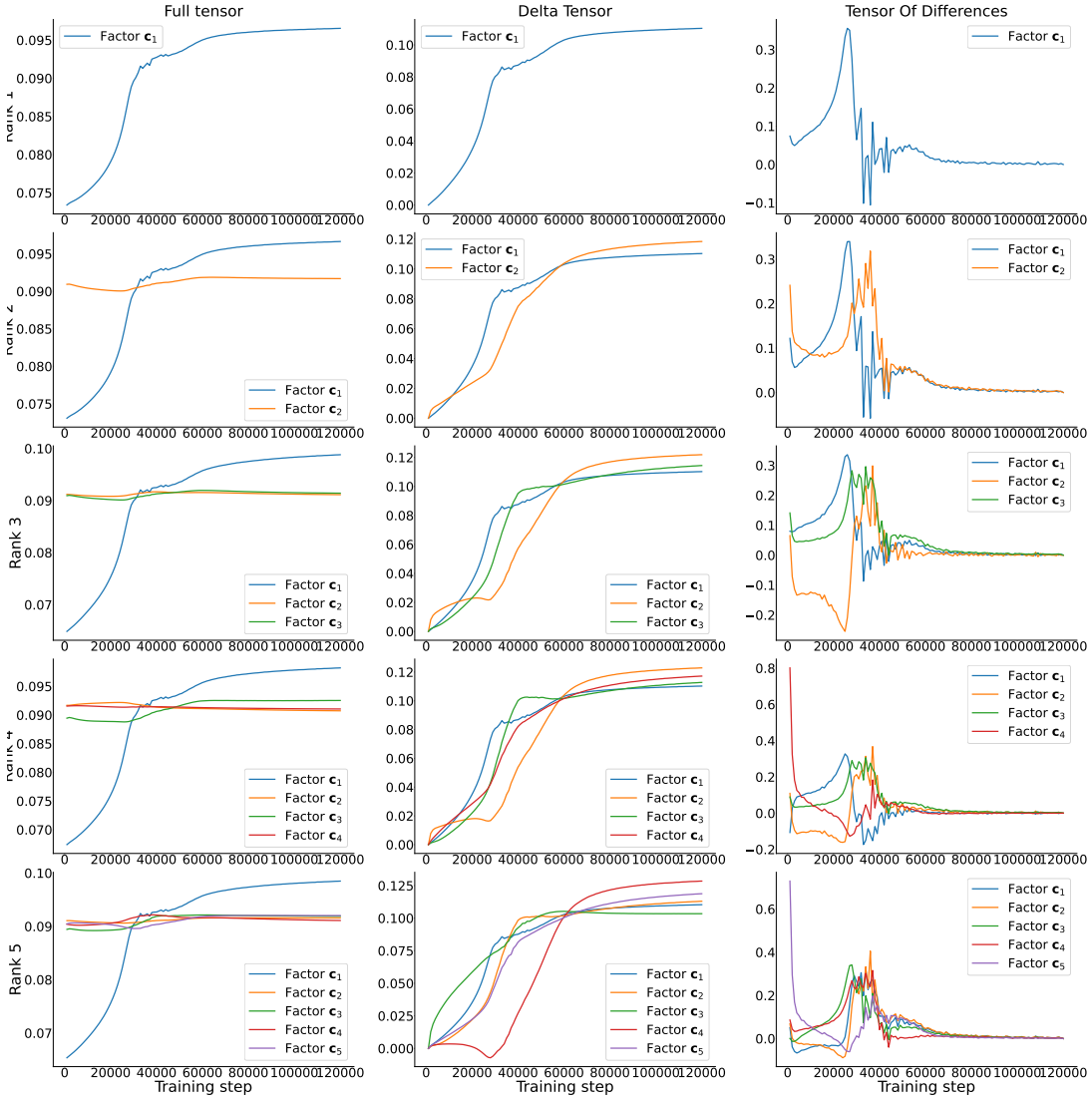


Figure C.1: Trial factors of RNN20-2 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

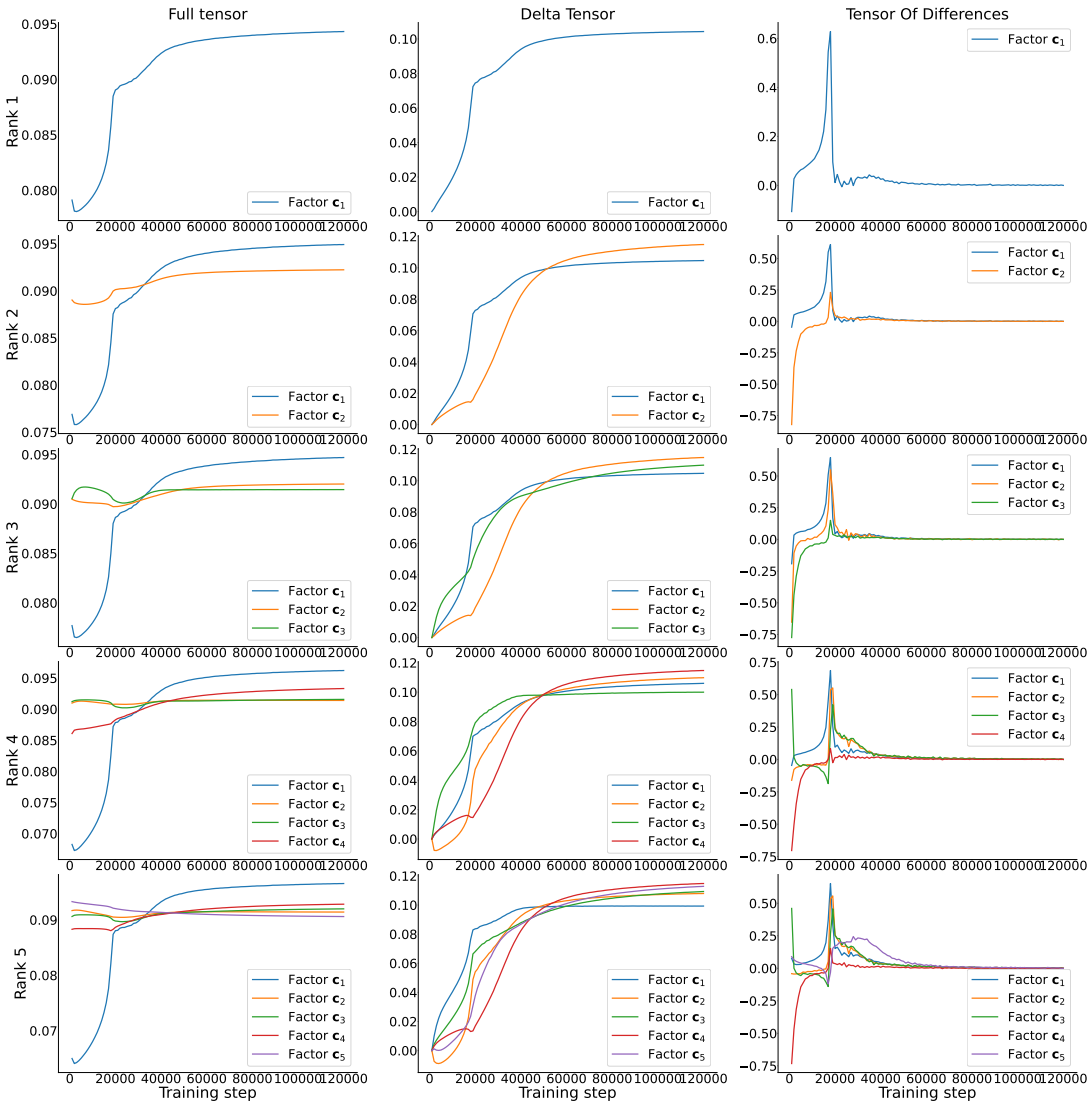


Figure C.1: Trial factors of RNN20-3 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

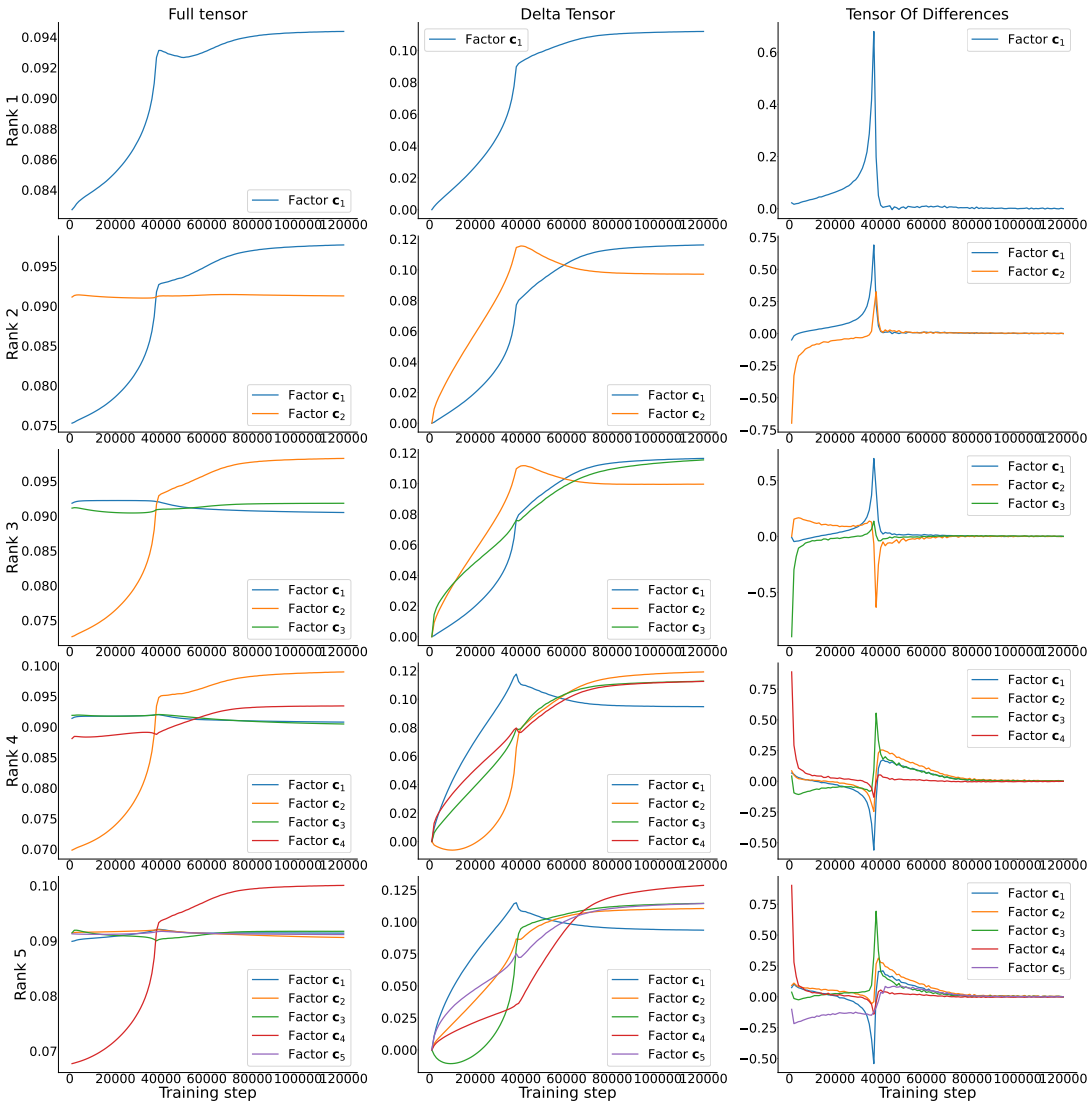


Figure C.1: Trial factors of RNN20-5 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

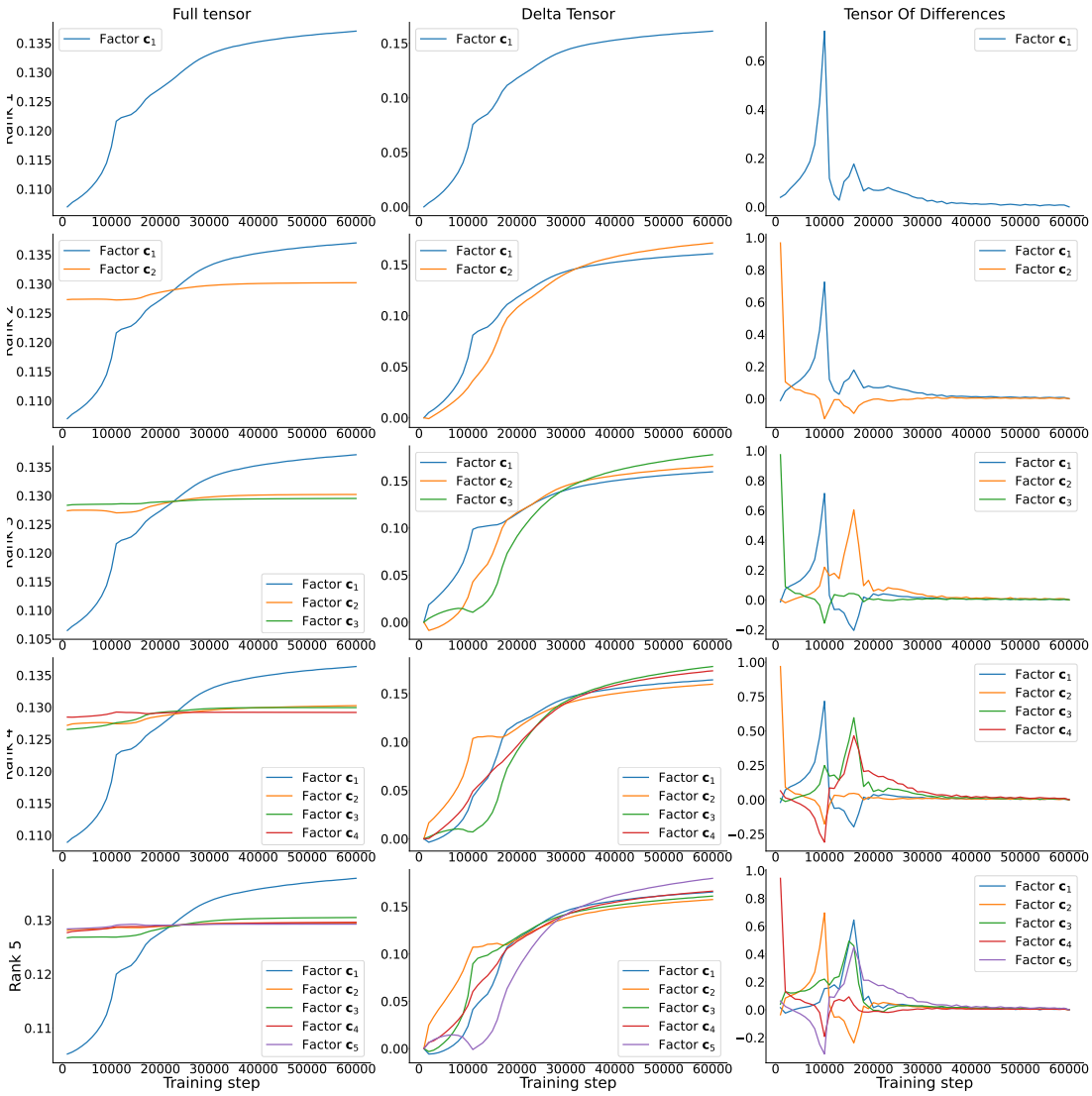


Figure C.2: Trial factors of RNN100-1 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

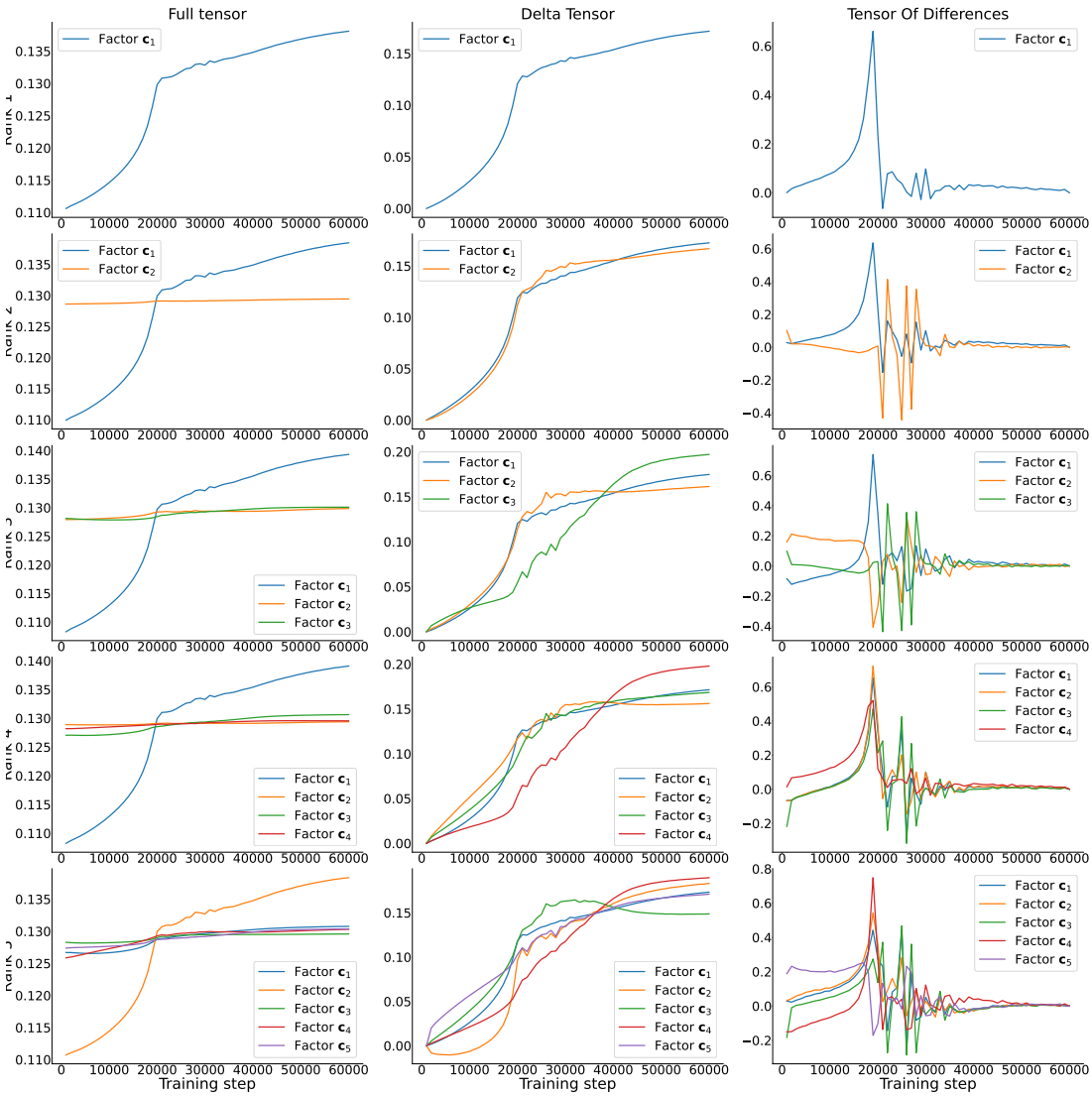


Figure C.2: Trial factors of RNN100-2 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

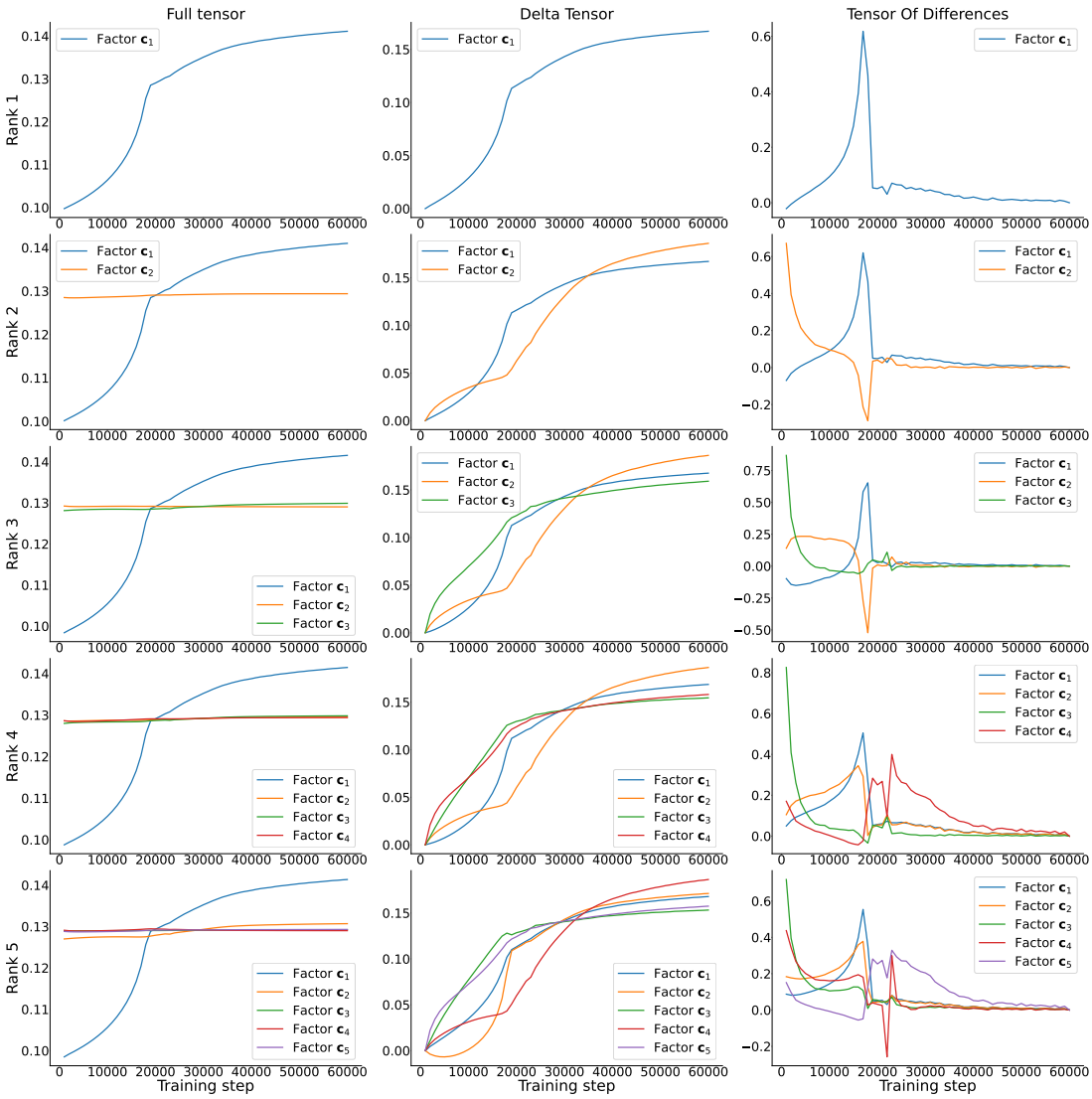


Figure C.2: Trial factors of RNN100-3 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

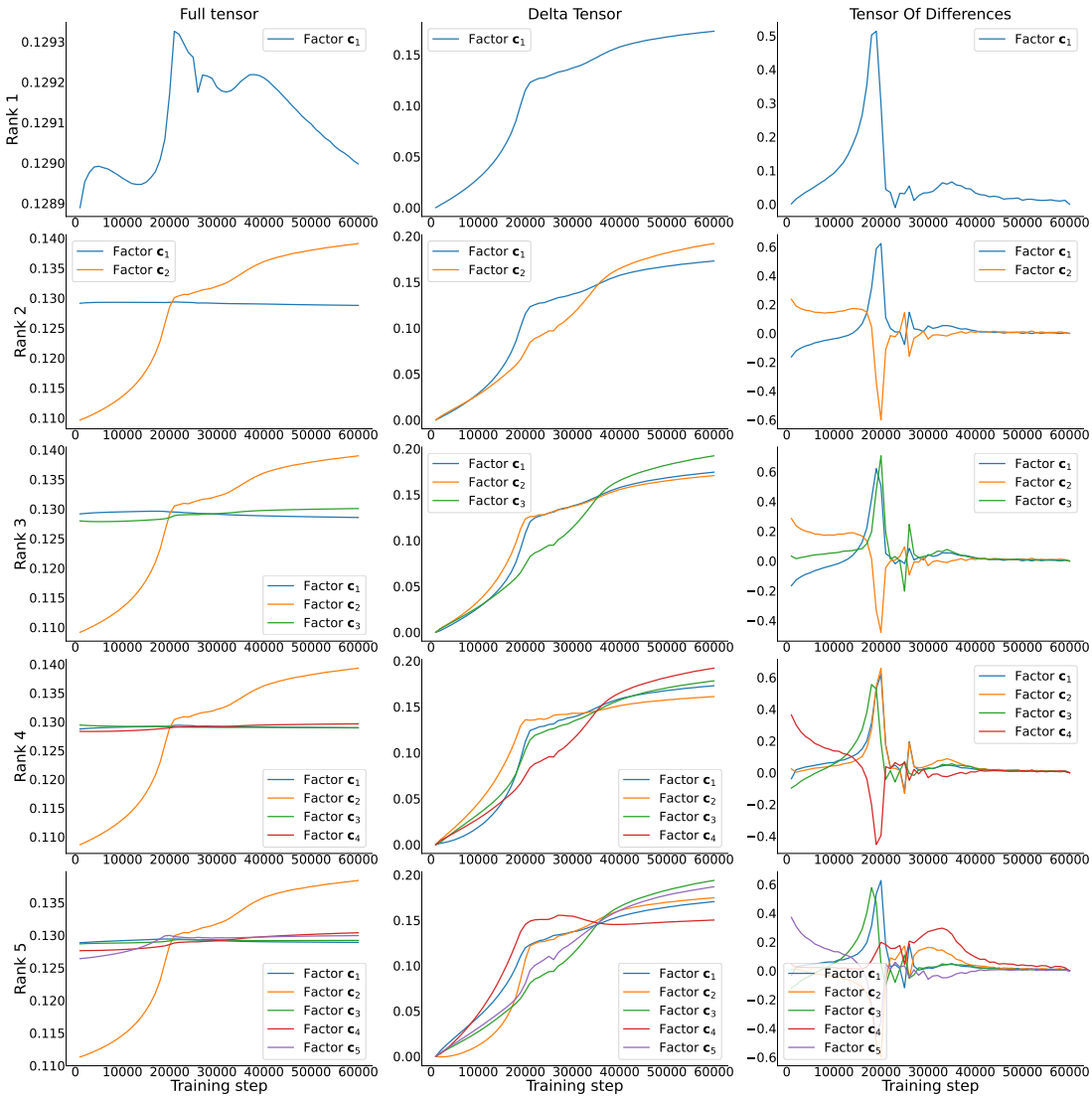


Figure C.2: Trial factors of RNN100-4 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

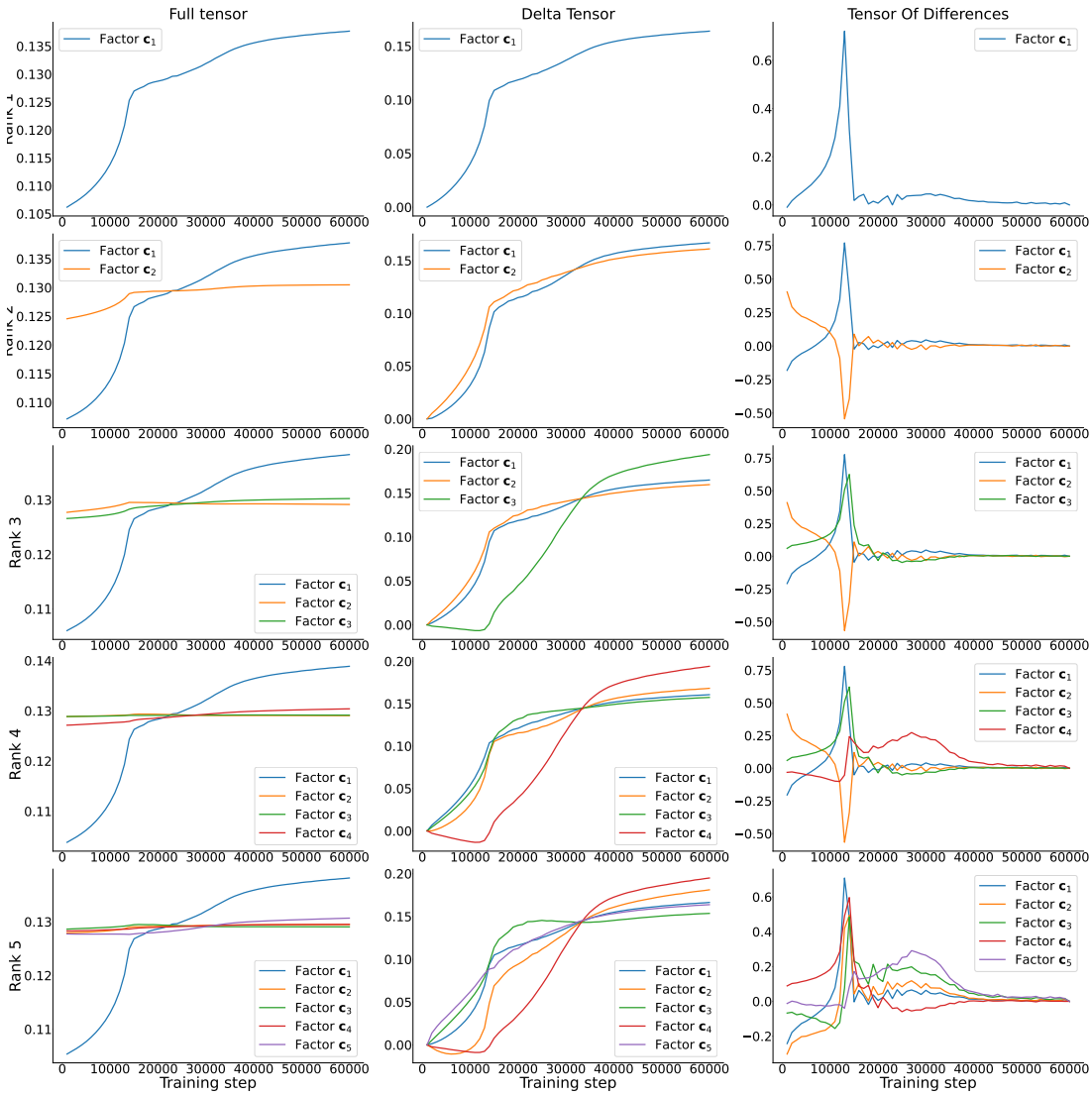


Figure C.2: Trial factors of RNN100-5 for different rank decompositions. Each row used a different rank for the decomposition of each of the tensor types (Full Tensor, Delta Tensor, Tensor of Differences) ranging from one to five.

Appendix D

Comparison of trial factors and performance matrices for each RNN

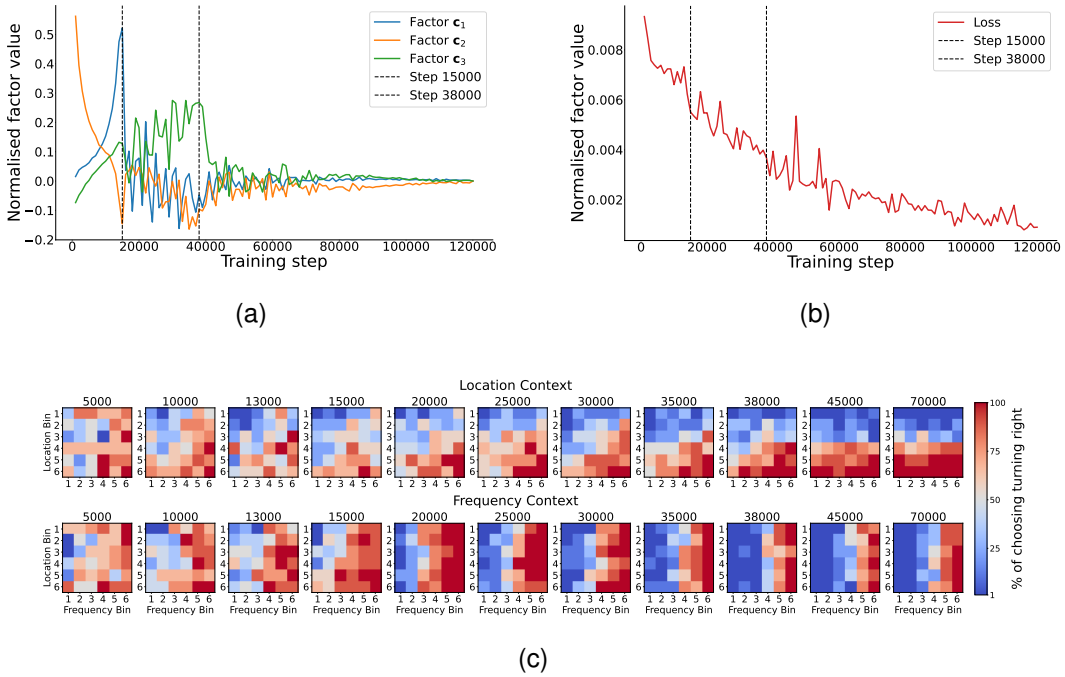


Figure D.1: Different performance measures of RNN20-1 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

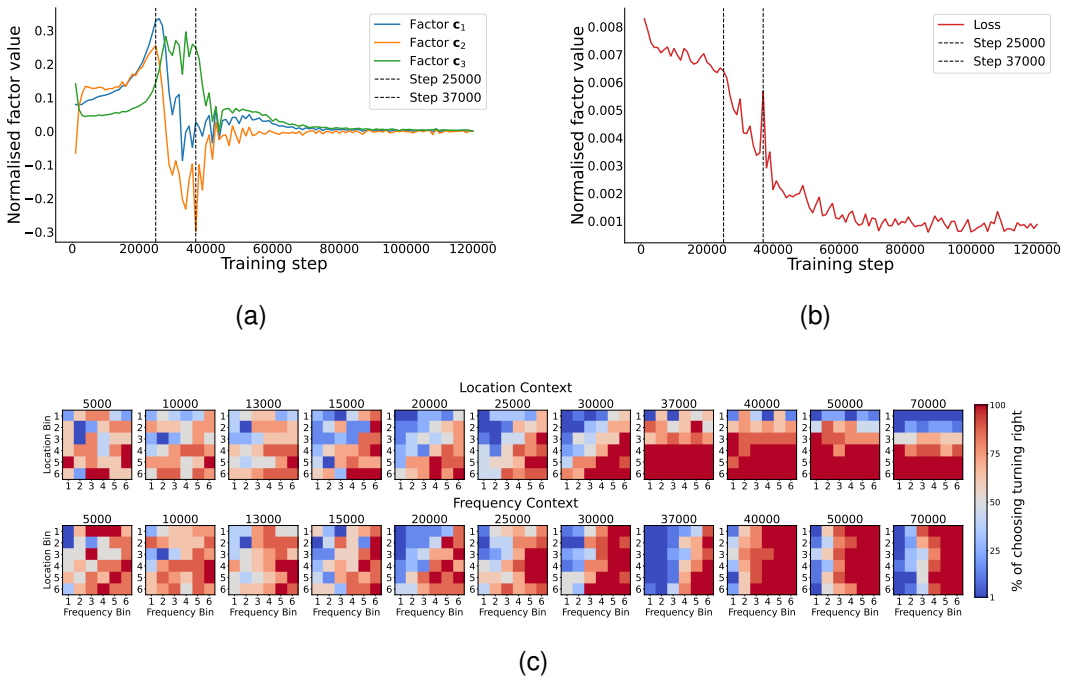


Figure D.2: Different performance measures of RNN20-2 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

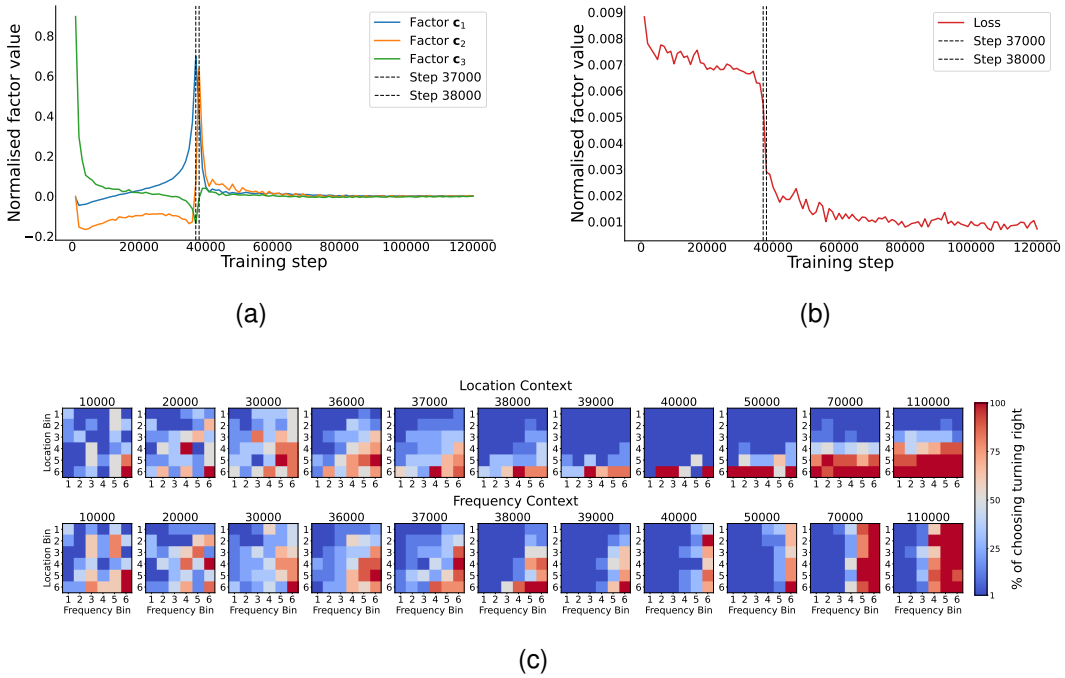


Figure D.3: Different performance measures of RNN20-5 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

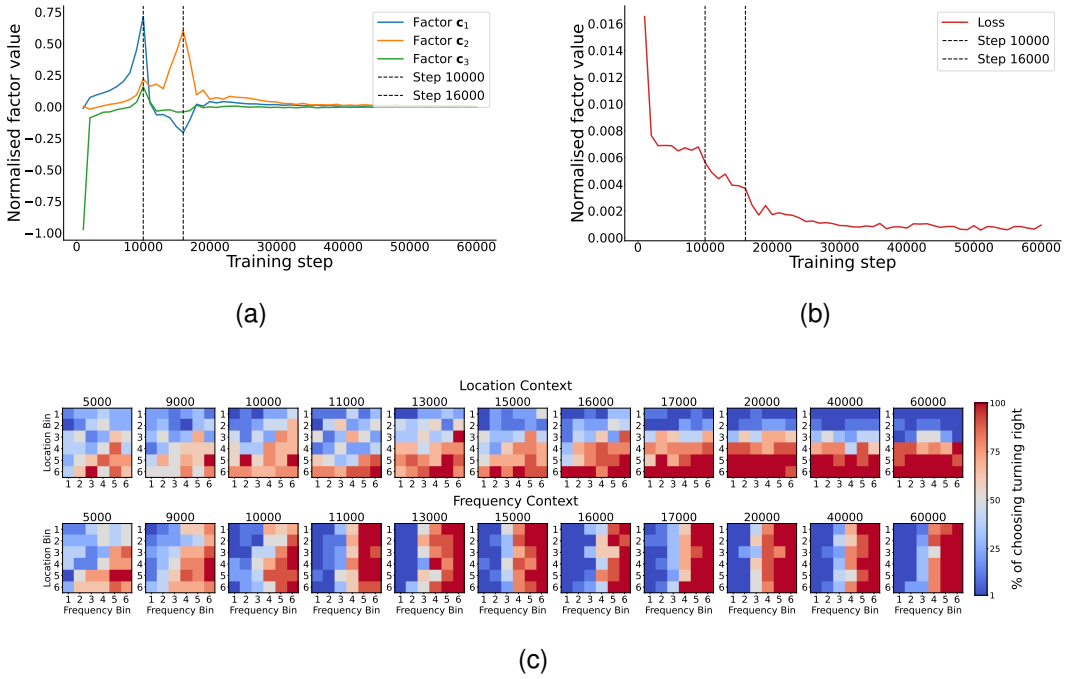


Figure D.4: Different performance measures of RNN100-1 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

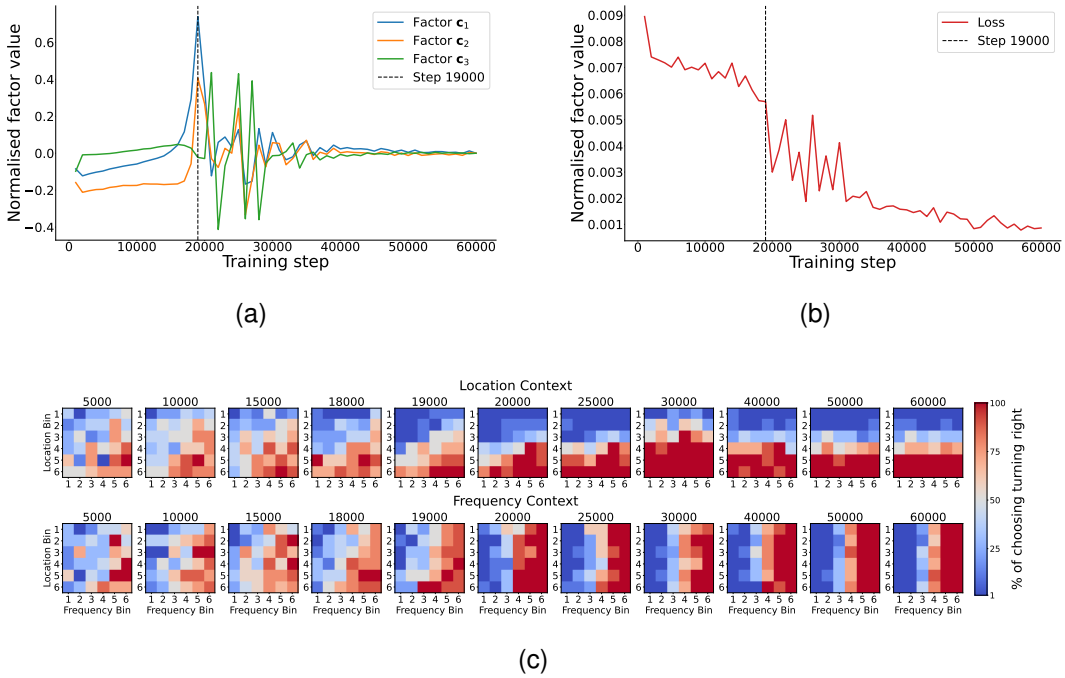


Figure D.5: Different performance measures of RNN100-2 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

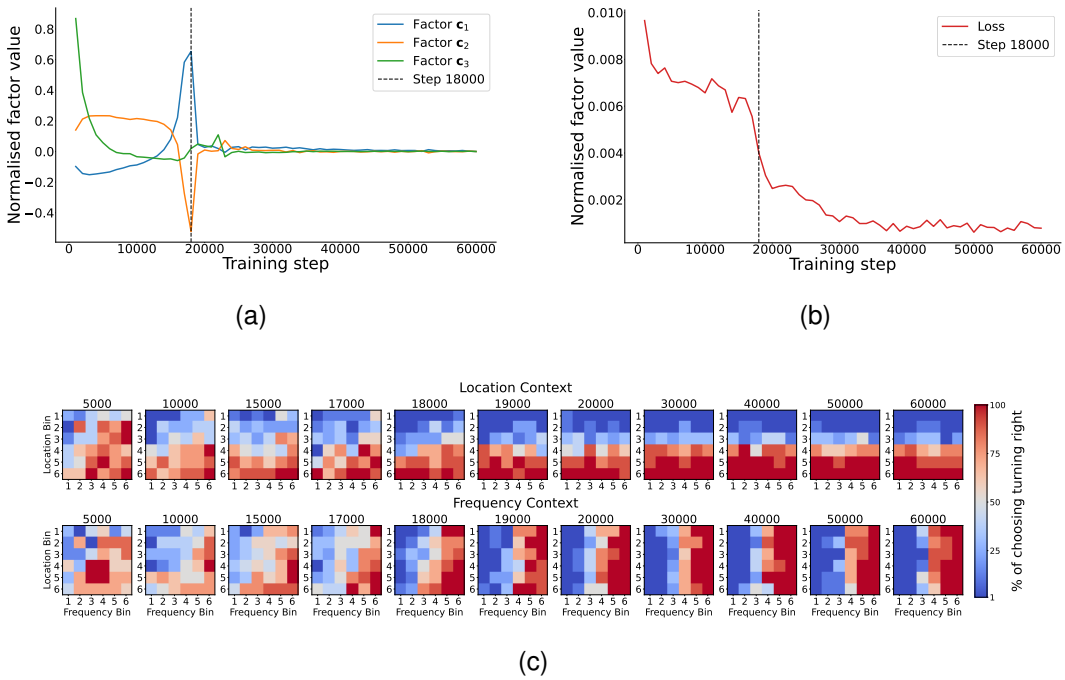


Figure D.6: Different performance measures of RNN100-3 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

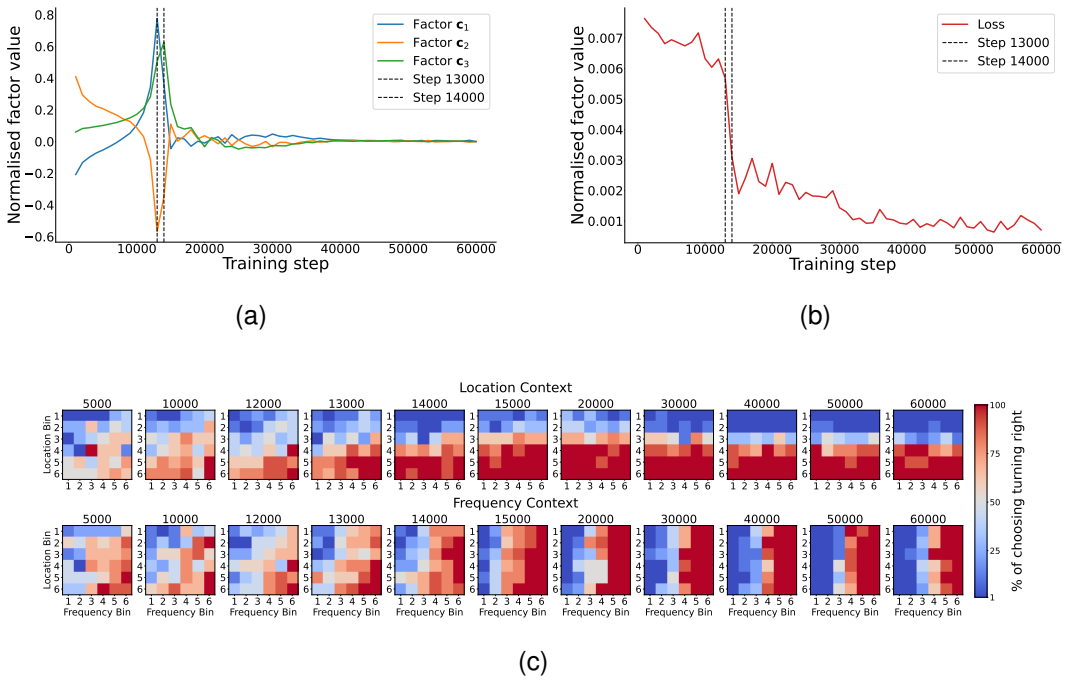


Figure D.7: Different performance measures of RNN100-4 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

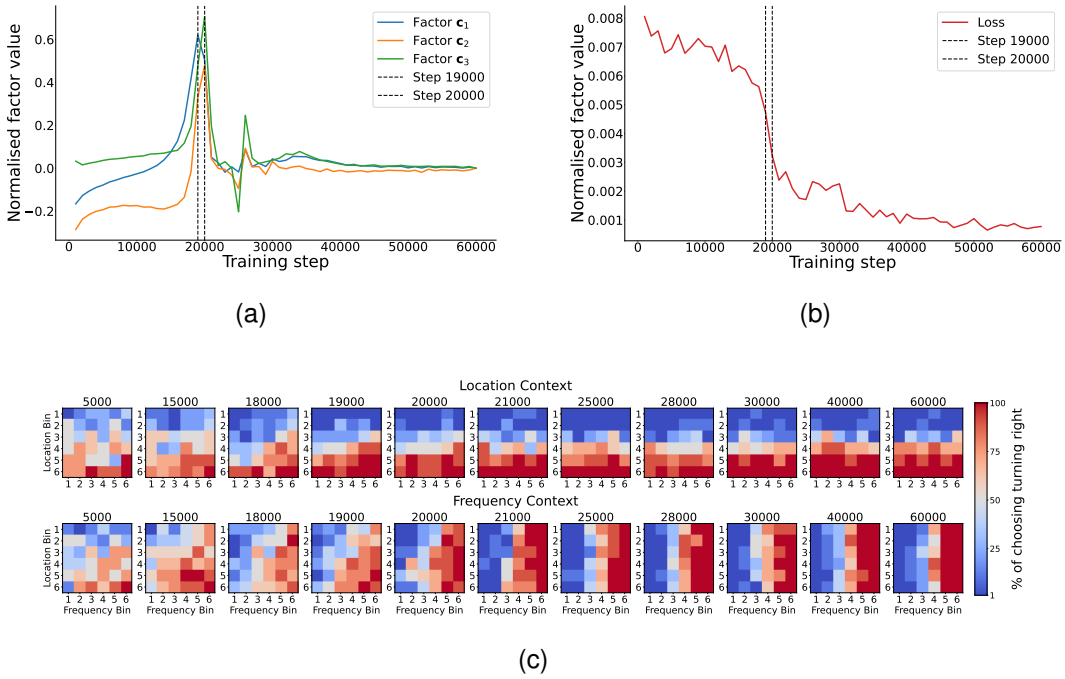
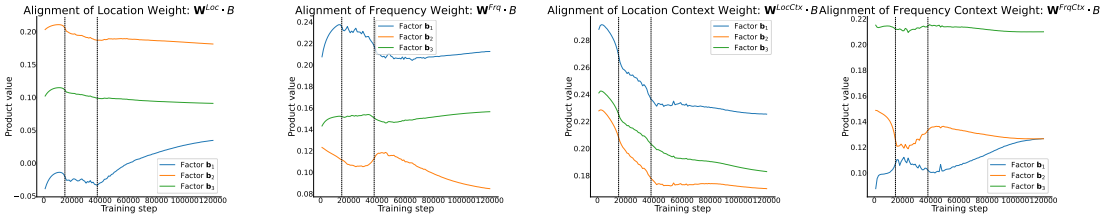


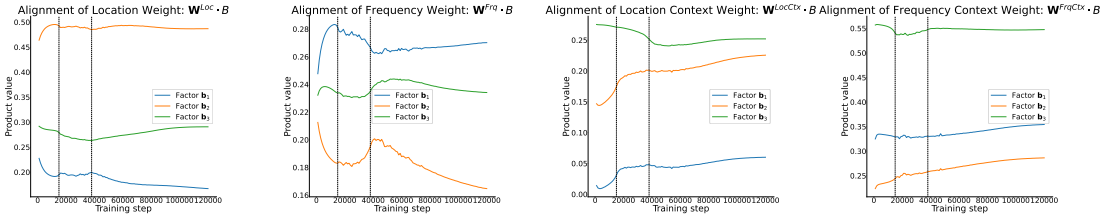
Figure D.8: Different performance measures of RNN100-5 in comparison to the trial factors of its weight tensor. (a) The trial factors for the Tensor of Differences. Vertical lines indicate the trial point at which the peaks occur. (b) Loss of the RNN during training. (c) Percentage of times the RNN chooses the direction right across trials for a specific location and frequency proportion (see Table 3.2) and context for different training steps.

Appendix E

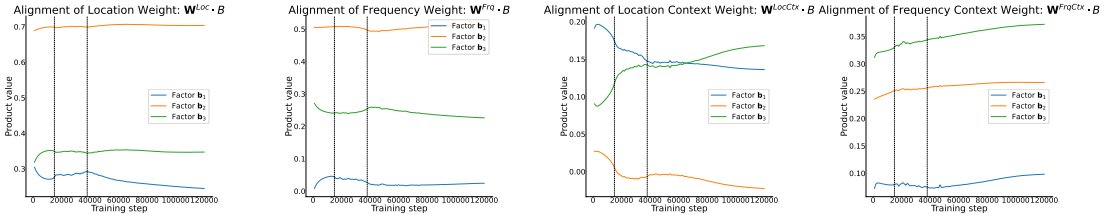
Alignment of input weights with row factors for each RNN



(a) Full Tensor

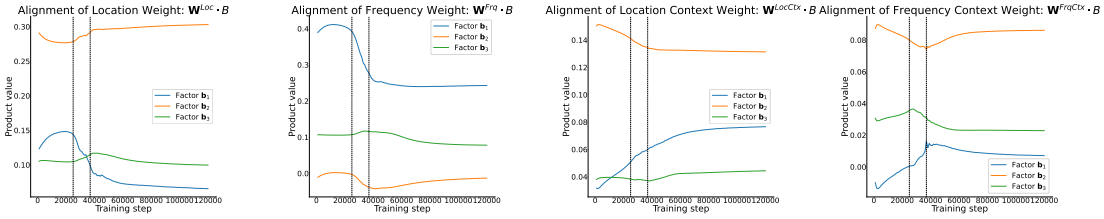


(b) Delta Tensor

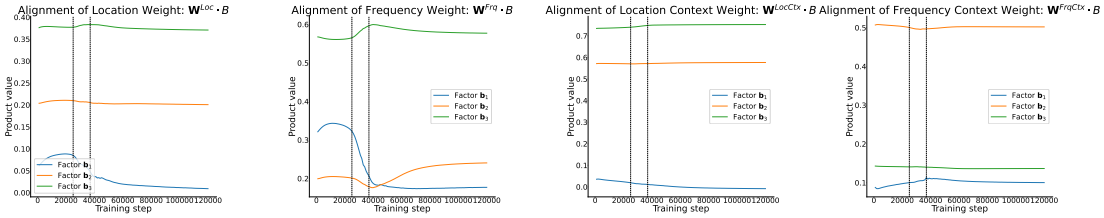


(c) Tensor of Differences

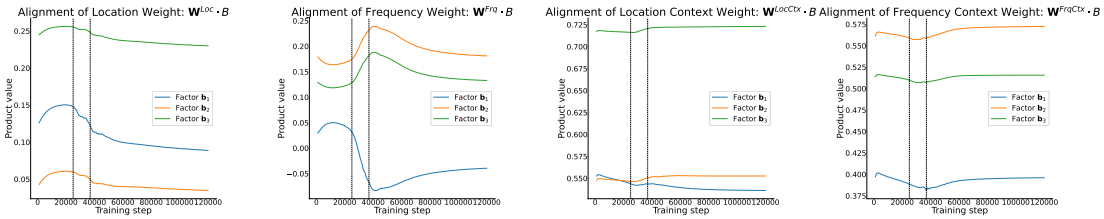
Figure E.1: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN20-1 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor

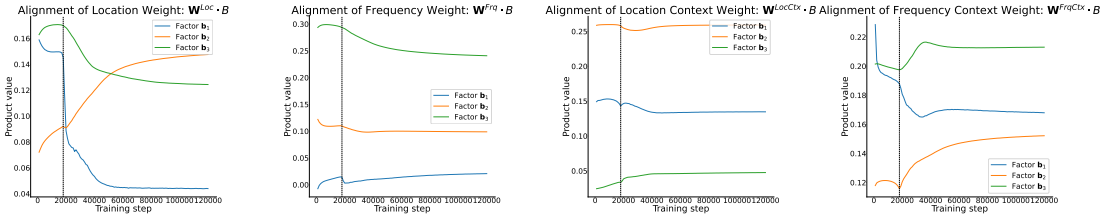


(b) Delta Tensor

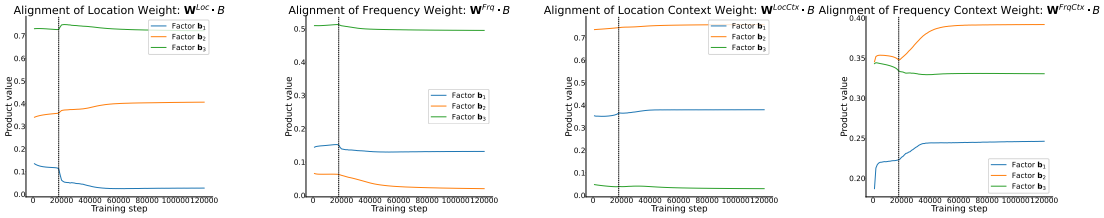


(c) Tensor of Differences

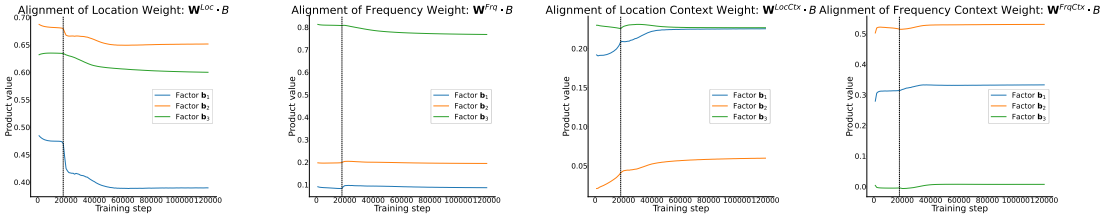
Figure E.2: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN20-2 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor

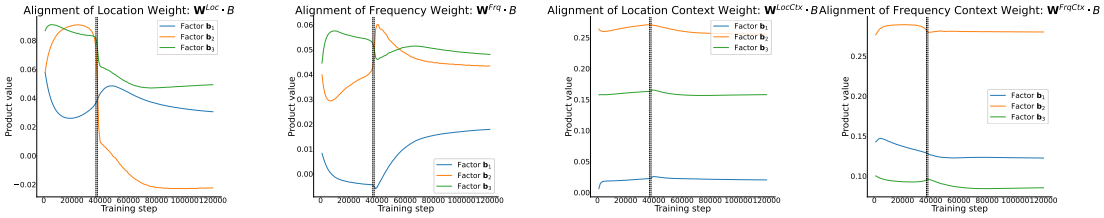


(b) Delta Tensor

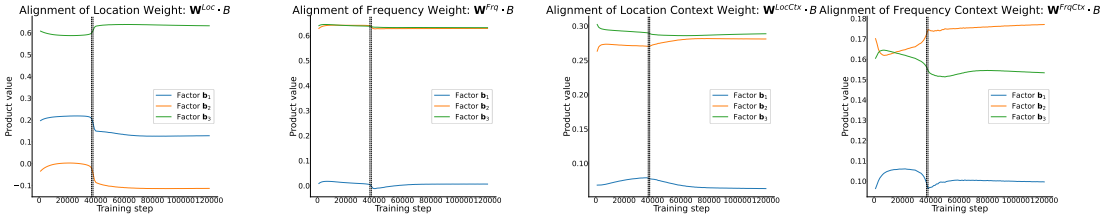


(c) Tensor of Differences

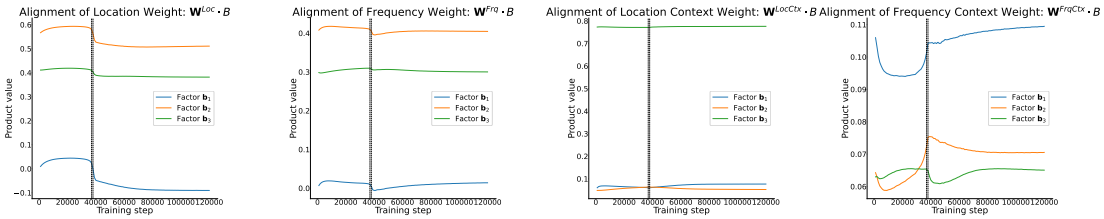
Figure E.3: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN20-3 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor

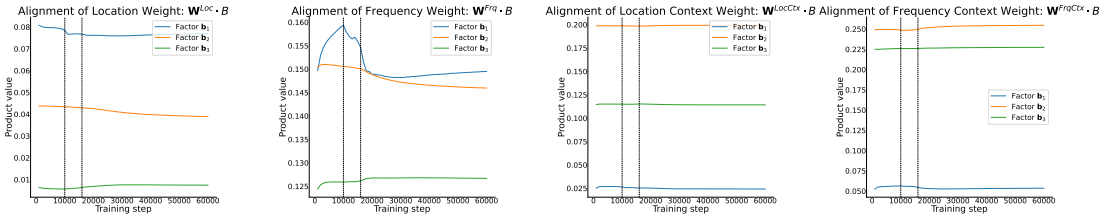


(b) Delta Tensor

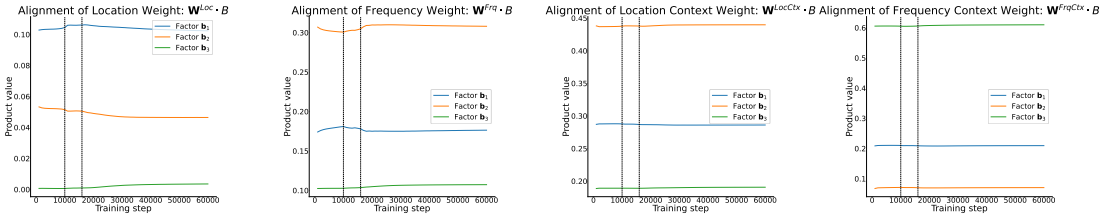


(c) Tensor of Differences

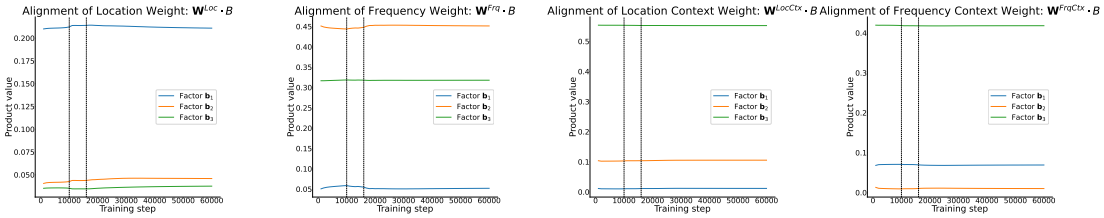
Figure E.4: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN20-5 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor

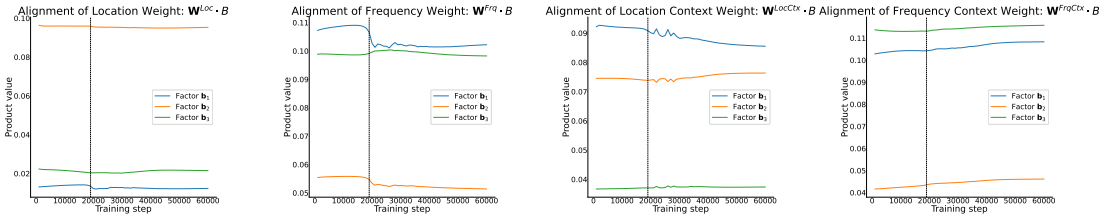


(b) Delta Tensor

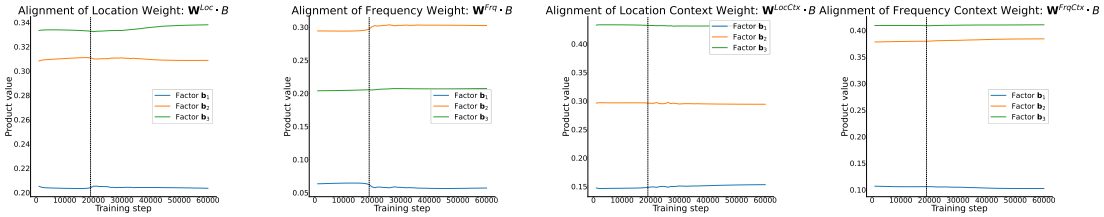


(c) Tensor of Differences

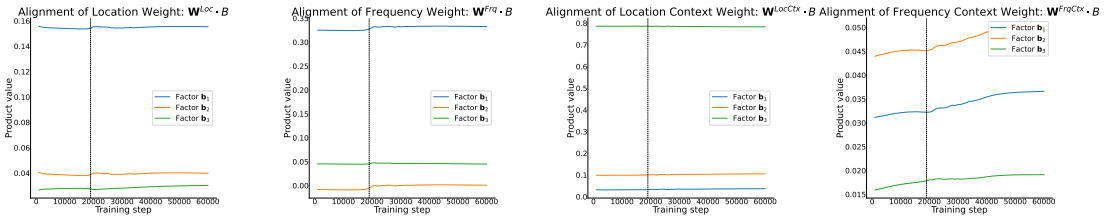
Figure E.5: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN100-1 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor

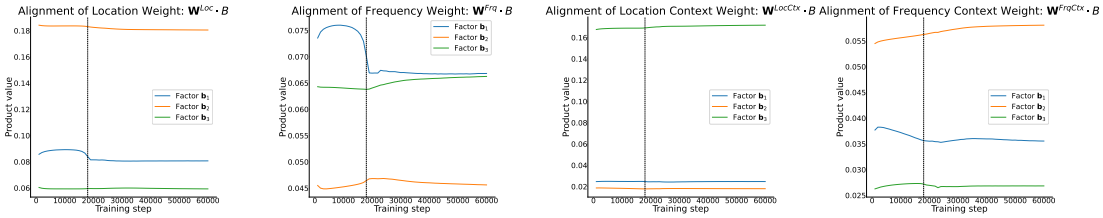


(b) Delta Tensor

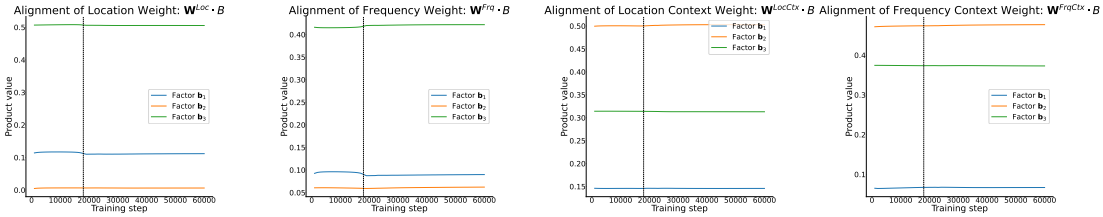


(c) Tensor of Differences

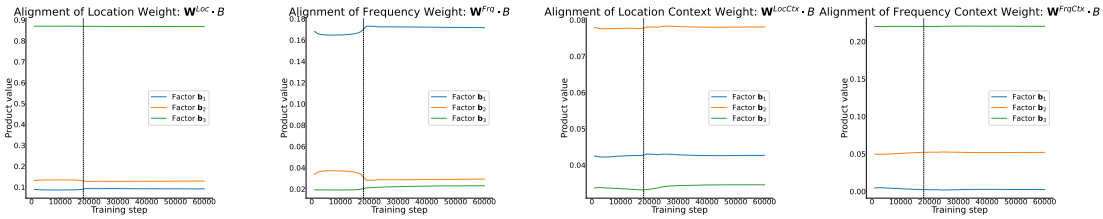
Figure E.6: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN100-2 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor

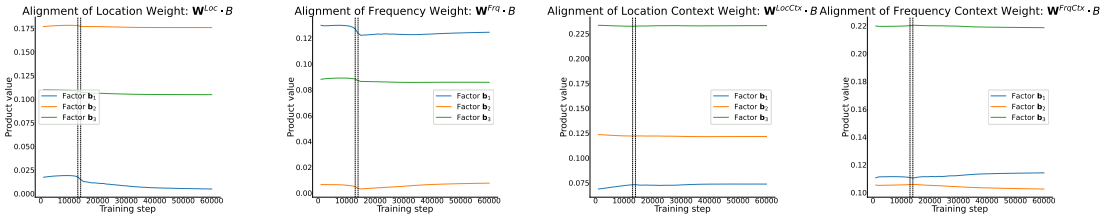


(b) Delta Tensor

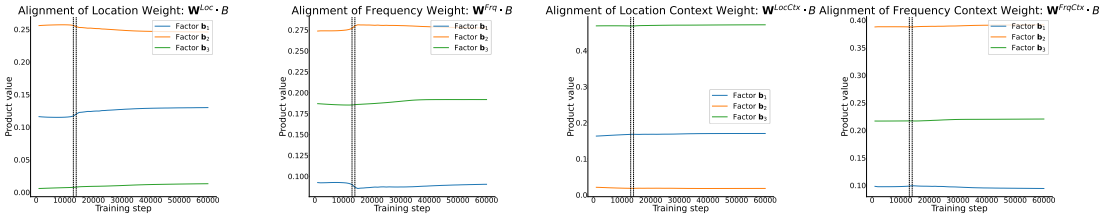


(c) Tensor of Differences

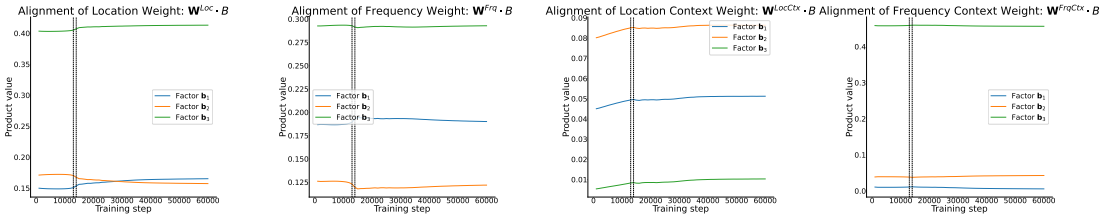
Figure E.7: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN100-3 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor

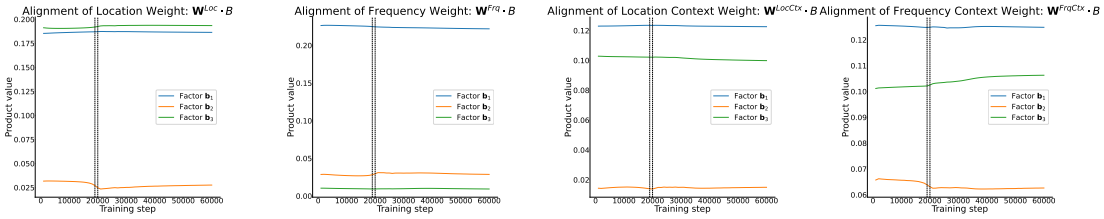


(b) Delta Tensor

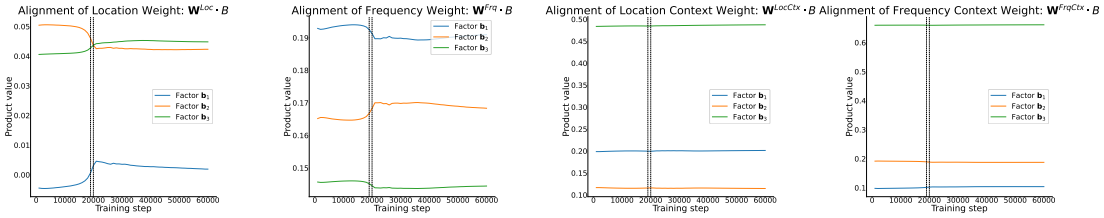


(c) Tensor of Differences

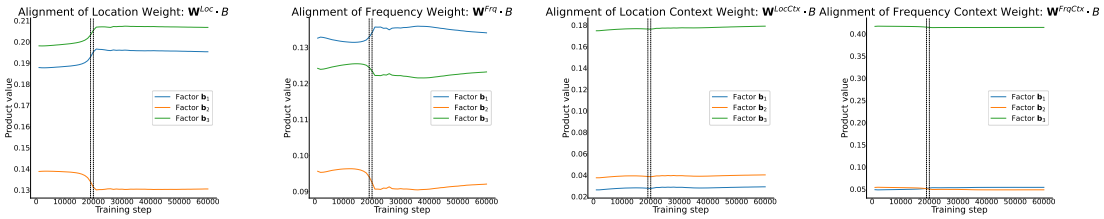
Figure E.8: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN100-4 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.



(a) Full Tensor



(b) Delta Tensor



(c) Tensor of Differences

Figure E.9: Alignment of each input weight $\boldsymbol{W}^{\mathrm{Loc}}, \boldsymbol{W}^{\mathrm{Frq}}, \boldsymbol{W}^{\mathrm{LocCtx}}, \boldsymbol{W}^{\mathrm{FrqCtx}}$ with the row factors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ of the rank three decomposition of RNN100-5 over time. Each subfigures is one of the tensor types: (a) Full tensor, (b) Delta tensor, (c) Tensor of Differences.

Appendix F

LtrRNN results

F.1 Hyperparameters for LtrRNN

The hyperparameters for which the LtrRNN was tuned alongside their considered and chosen values can be seen in Table F.1. Any other hyperparameters were left as recommended by Pellegrino et al. [32] in their example code. The code was adapted to keep `control_dim` as is, even when `in_space_control=True`.

Hyperparameter	Considered Values	Chosen value
`rank`	1-10	3
`l`	1000,5000,10000,20000,50000	20000
`rnn_dim`	20	20
`in_space_control`	True, False	True
`control_execution`	True, False	True
`control_preparatory`	True, False	False
`sigma_observation`	0.1,0.01,0.001,0.0001	0.01
`regularization`	0.01,0.001	0.001
`control_dnn_dim`	[20,20]	[20,20]
`control_hidden_dim`	[20]	[20]

Table F.1: Hyperparameters for LtrRNN that were changed from the default. All considered values during hyperparameter tuning as well as the final chosen value are shown.

F.2 Variable Input Results for all RNNs with 20 units

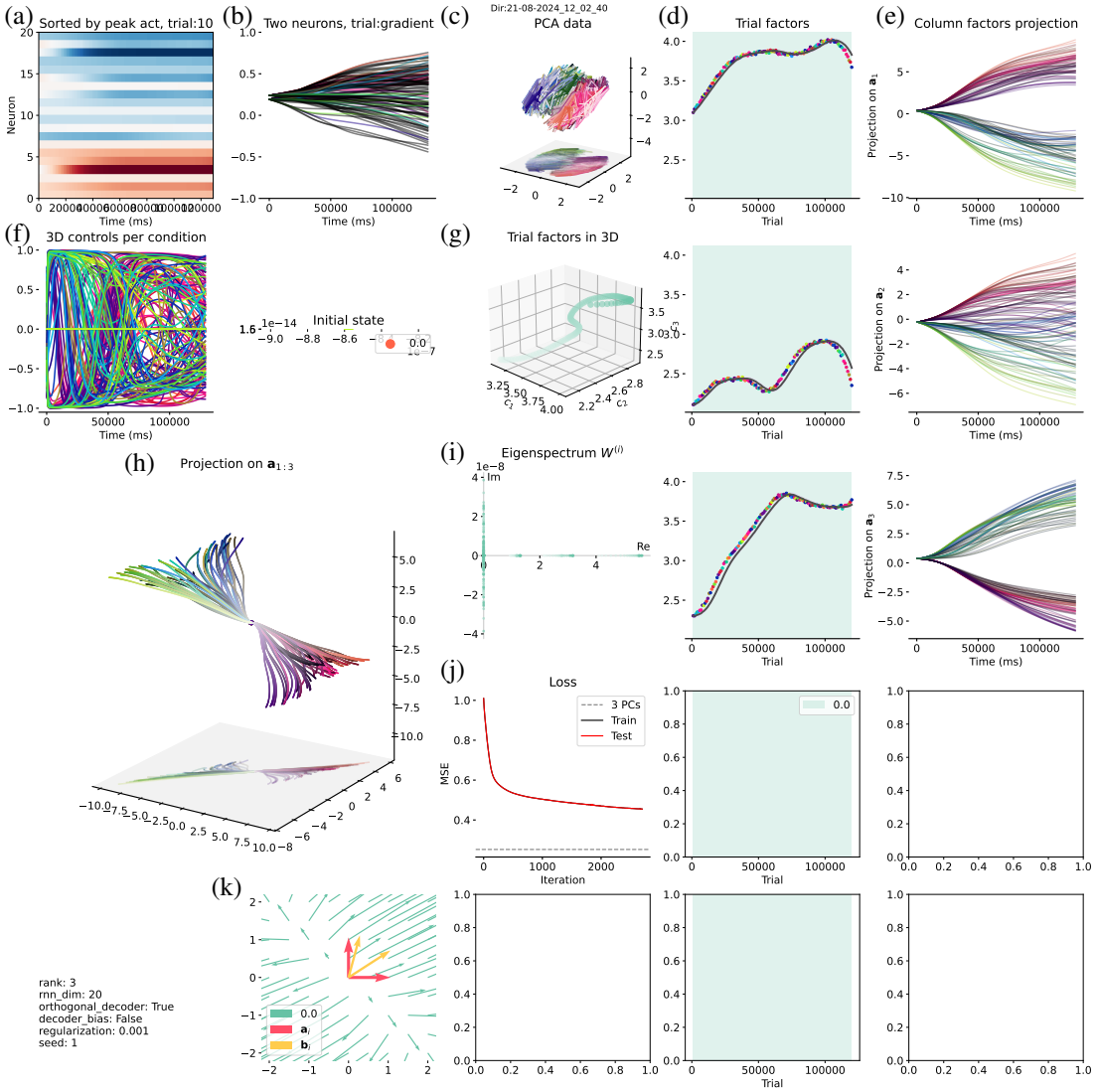


Figure F.1: Results of using LtrRNN on the fixed input activity of RNN20-1 with rank three. The subfigures are the same as in Figure 4.2.

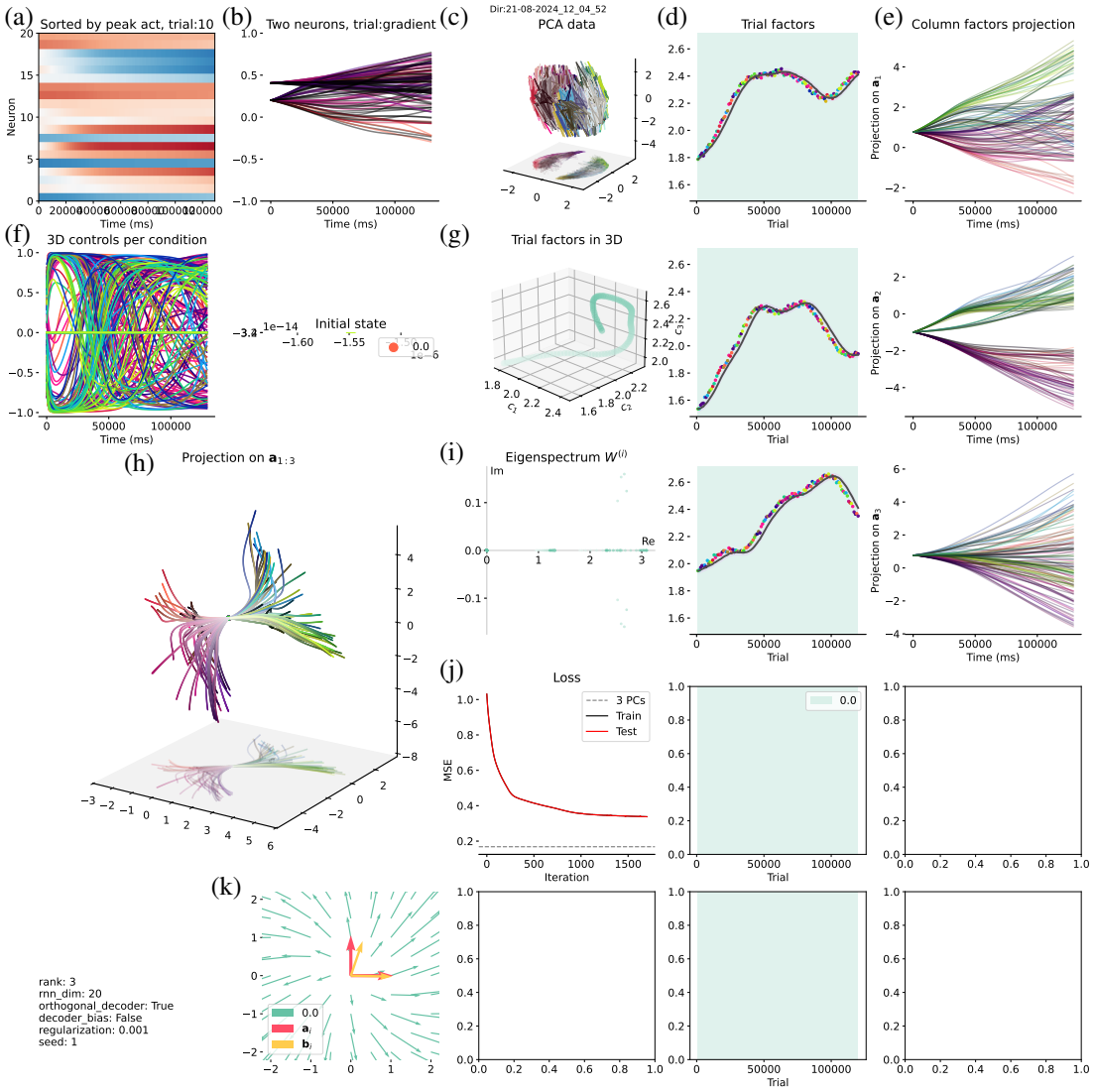


Figure F.2: Results of using LtrRNN on the fixed input activity of RNN20-2 with rank three. The subfigures are the same as in Figure 4.2.

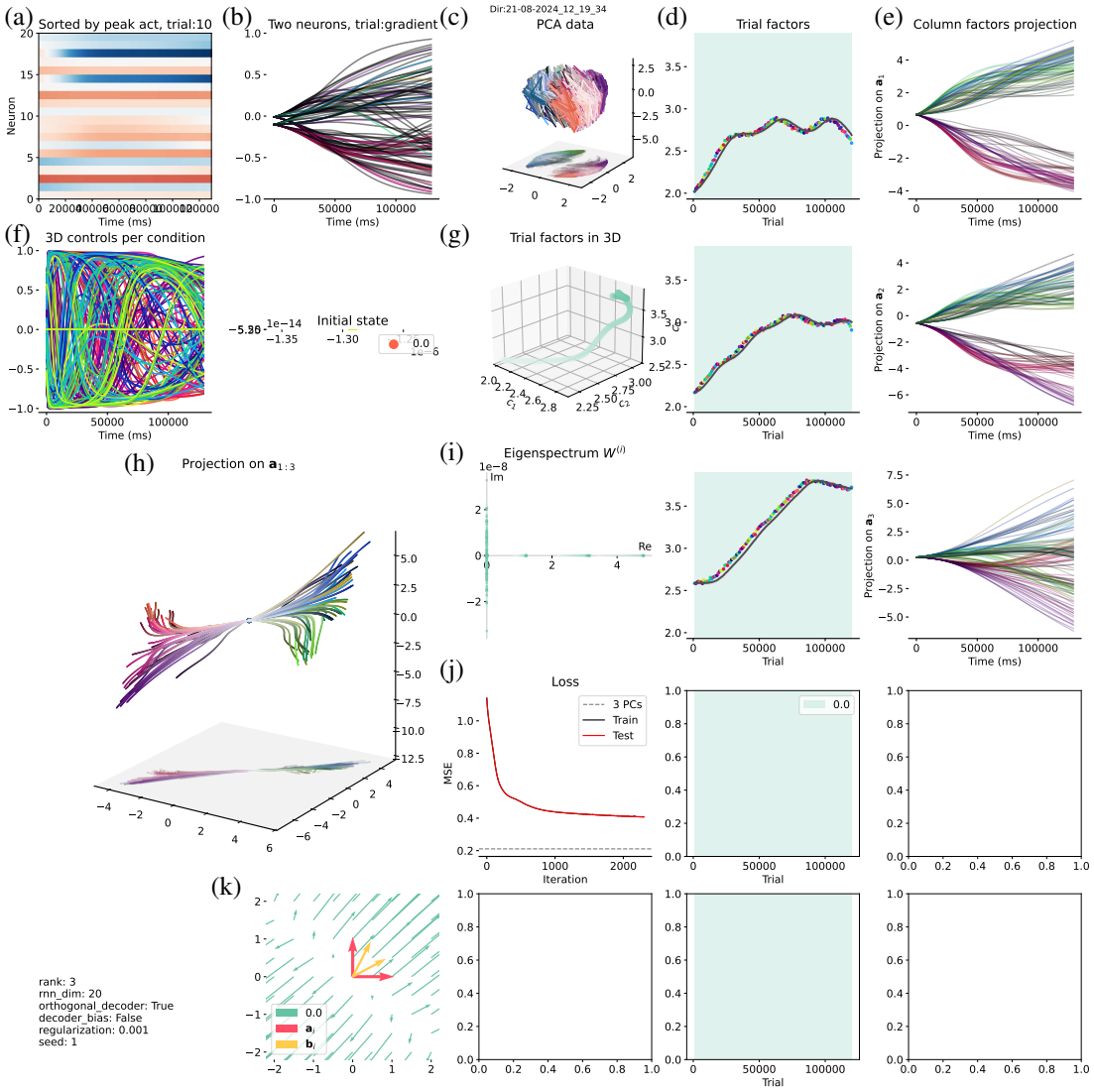


Figure F.3: Results of using LtrRNN on the fixed input activity of RNN20-3 with rank three. The subfigures are the same as in Figure 4.2.

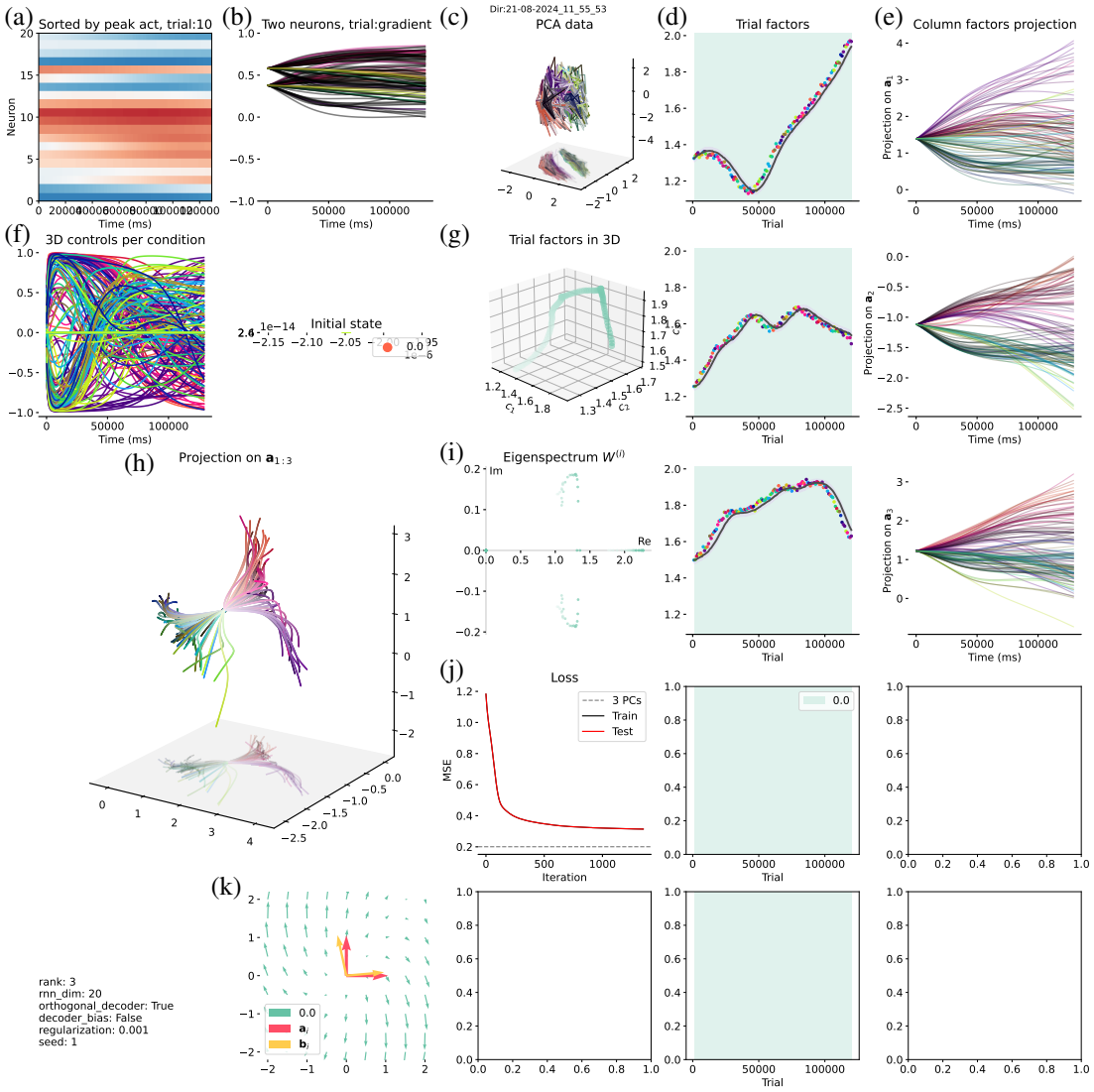


Figure F.4: Results of using LtrRNN on the fixed input activity of RNN20-5 with rank three. The subfigures are the same as in Figure 4.2.