Decision making in low-rank recurrent neural networks

Erfan Baradarantohidi and Ryan Gelston

July 31, 2023

1 Introduction

In this paper we explore the use of low-rank recurrent neural network on two tasks, a decision making task and an integration task. This work is largely a reproduction of portions of Debreuil et al. [1]. As a part of understanding the behavior and connectivity of each network we looked to the statistics of each networks' parameters and created a new network by resampling the parameters from an identical distribution. Likewise, we attempted to reproduce the behavior of our trained network using simple one and two dimensional dynamic systems.

This report is written as the final project of the Programming Tutorial of the course Models of Higher Brain Function by Prof. Dr. Henning Sprekeler.

2 Low-rank Networks

Low-rank networks simplify their connectivity matrix $\boldsymbol{J}$ to the sum of R outer products between unique vectors $\boldsymbol{m}^{(i)}$ and $\boldsymbol{n}^{(i)}$. With this construction we are guaranteed that the rank of $\boldsymbol{J}$ is no larger than R, and equal to R assuming for $i \neq j$ that no $\boldsymbol{m}^{(i)}$ is parallel to $\boldsymbol{m}^{(j)}$. By using a low-rank connectivity matrix we are able to visualize collective network activity in terms of the R orthogonal basis vectors that span $\boldsymbol{J}$.

In our experiments we used a rank 1 ($R = 1$) and rank 2 ($R = 2$) connectivity matrix to model an interconnected population of $N = 128$ neurons, and normalized the matrix by the number of neurons. $\boldsymbol{J}$ was calculated as follows:

$$\boldsymbol{J} = \frac{1}{N}(\boldsymbol{m}^{(1)}\boldsymbol{n}^{(1)T} + ... + \boldsymbol{m}^{(R)}\boldsymbol{n}^{(R)T}) \tag{1}$$

Our network received a one-dimensional time-series $u(t)$ as input. At each time-step the input value was multiplied by the N-dimensional vector $\boldsymbol{I}$, which is the input pattern of the network, or rather scales the input value $u(t)$ for each neuron in the network. The hidden state of the network was represented by $\boldsymbol{x}$ and the neuron's transfer function ϕ is the hyperbolic tangent function.

The dynamics of the ith hidden unit of the network is described below:

$$\tau\frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij}\phi(x_j) + I_i u(t) \tag{2}$$

or, in vector representation:

$$\tau\frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x} + \boldsymbol{J}\phi(\boldsymbol{x}) + \boldsymbol{I}u(t) \tag{3}$$

$$= -\boldsymbol{x} + \boldsymbol{m}(\frac{1}{N}\boldsymbol{n}^T\phi(\boldsymbol{x})) + \boldsymbol{I}u(t) \tag{4}$$

From the above equation we see that $\boldsymbol{n}$ integrates the network's output from the previous time step and $\boldsymbol{m}$ defines the pattern to be added back to the network.

The output of the network at a given time-step $z(t)$ is a weighted average of the values of the hidden states at said time-step after being passed through the transfer function. This is expressed as:

$$z(t) = \frac{1}{N}\sum_{i=1}^{N} w_i\phi(x_i) = \frac{1}{N}\boldsymbol{w}^T\phi(\boldsymbol{x}) \tag{5}$$

The trainable parameters in our model were $\boldsymbol{m}$ and $\boldsymbol{n}$ and were instantiated with samples from a normal distribution with $\mu = 0$ and $\sigma = 1$. Variables $\boldsymbol{I}$ and $\boldsymbol{w}$ were fixed and instantiated to values pulled from a normal distribution with $\mu = 0$ and $\sigma = 1$ and 4, respectfully. In addition, we set $\tau = 100$ms and had our timestep $dt = 20$ms. All networks were trained using the Adam [2] optimizer with batch sizes of 32 and a learning rate of 5e-3.

The RNN class we used was validated by assuring we were able to overfit a rank 1 network to a single sample from the single decision data, that we could overfit a rank 2 network to a single sample from the parametric decision data, and that we were able to train a full rank network to converge on the single decision data.

3 Perceptual Decision Making

We trained a rank 1 network to perform a decision making task. For each trial the stimulus $u(t)$ was randomly drawn from $\pm\frac{3.2}{100}\{1,2,4,8,16\}$. The stimulus was present between time-steps 5 and 45. The input signal also included background noise $\xi(t)$ for the entire duration of the trail, which was drawn from a normal distribution of $\mu = 0$ and $\sigma = 0.03$.

$$u(t) = \begin{cases} \bar{u} + \xi(t) & \text{if } 5 < t \leq 45 \\ \xi(t) & \text{otherwise} \end{cases} \tag{6}$$

The expected output of the network $y(t)$ for $\bar{u} > 0$ was 1, and for $\bar{u} < 0$ was -1.

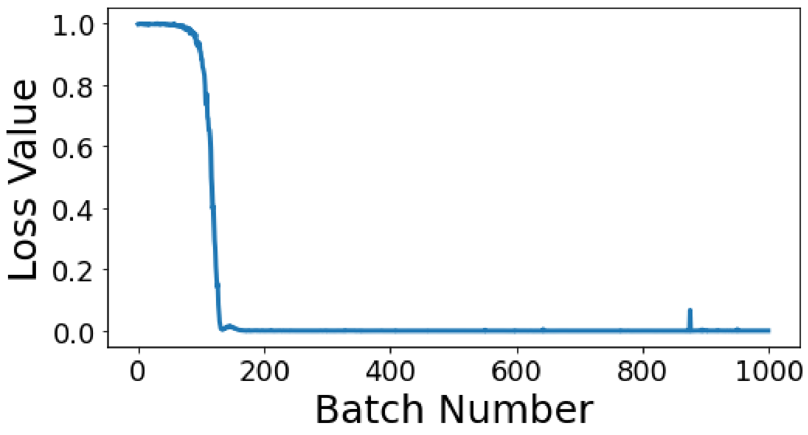

Figure 1: Loss curve when training the rank 1 network on the perceptual decision making task.

We defined the training error as the mean squared error of the last 15 time steps. The network was trained on a data set of size 32,000 over one epoch. During training the network regularly converged to a value under 5e-3. The loss curve for training can be seen in Figure 1

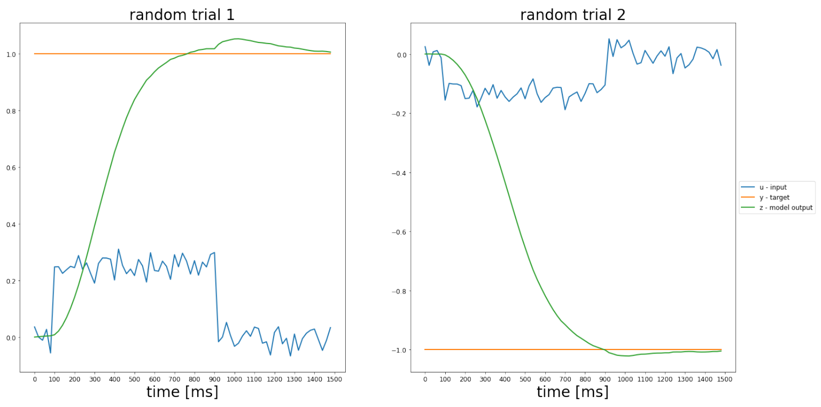

Figure 2: Dynamics of the trained model for two randomly generated data samples from the perceptual decision making task.

3.1 Connectivity Patterns

Debreuil et al. [1] found $\sigma_{\boldsymbol{nI}}$, $\sigma_{\boldsymbol{mn}}$, and $\sigma_{\boldsymbol{wm}}$ to be key parameters in the computation of the decision task. Figure 2b in Debreuil et al. shows these covariances to have a value around 1.5, with $\sigma_{\boldsymbol{mI}}$ and $\sigma_{\boldsymbol{wn}}$ having a value around 0.5, and $\sigma_{\boldsymbol{wI}}$ having a value around zero.

The covariance between two vectors can act as a measurement of their alignment. In the case where the mean value for each vector is zero or close to zero compared to most points in the dataset, the covariance is similar to the dot product normalized by the vector length. Furthermore the dot product suggests something about the relationship between the signs of the entries of the vector at shared indices, as weighted by the product of the entries. When entries are of the same sign, they will positively contribute to the sum of products, and when of different signs, negatively contribute. It follows that if a dot product is positive there is a higher weight of entries with matched signs, and if negative a higher weight of entries with opposing signs. Given that the mean values for all parameters in our model are seemingly far closer to zero than the majority of their entries we will think of the covariance as a scaled dot product.

The network is trained to output 1 when $\bar{u} > 0$ and -1 when $\bar{u} < 0$. It follows that for $\bar{u} > 0$ the ideal $\boldsymbol{x}$ value for the last 15 outputs should reach a state such that $\boldsymbol{w}^T\phi(\boldsymbol{x}) = 1$. From this we would expect $\boldsymbol{w}$ to have a non-negative dot

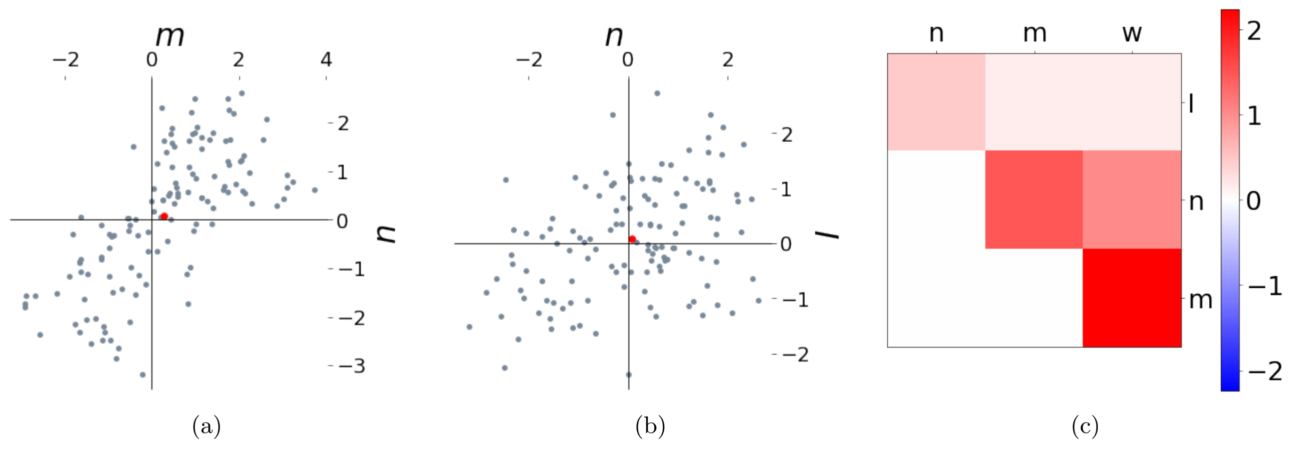

Figure 3: Above is a recreation of Figure 2b from Debreuil et al. [1]. The red point in the center of figures (a) and (b) is the mean of $\boldsymbol{m}$, $\boldsymbol{n}$, $\boldsymbol{I}$. (c) A selected portion of the covariance matrix of the rank 1 network.

product with $\boldsymbol{x}$. A symmetric argument can be made in the case when $\bar{u} < 0$ and $y = -1$. Given that the components of $\boldsymbol{x}$ are $\boldsymbol{m}$ and $\boldsymbol{I}$, we would expect $\boldsymbol{m}$ to move towards alignment with $\boldsymbol{w}$ in order to minimize the error function. With this we would expect $\boldsymbol{m}^T\boldsymbol{w}$ to have a relatively high magnitude, and thus the same for their covariance value. For similar reasons, we can expect $\sigma_{\boldsymbol{mn}}$ and $\sigma_{\boldsymbol{nI}}$ to have a relatively high magnitude so that $\boldsymbol{n}$ can properly integrate $\phi(\boldsymbol{x})$.

We would expect $\boldsymbol{w}$ and $\boldsymbol{I}$ to have a relatively low covariance, given that they are both randomly sampled static variables. Likewise, we would expect $|\sigma_{\boldsymbol{mI}}|$ to be low due to lack of interaction within the system and the tendency for $\boldsymbol{m}$ to move towards $\boldsymbol{w}$, considering $\sigma_{\boldsymbol{wI}}$ often has a very low magnitude.

One anomaly between our figure 3c and figure 2b in Debreuil et al. are the relative magnitudes between $\sigma_{\boldsymbol{nI}}$ and $\sigma_{\boldsymbol{nw}}$. While looking at the covariance matrices for different instances of our model it was often the case that $|\sigma_{\boldsymbol{nI}}| \geq |\sigma_{\boldsymbol{wn}}|$. This might be due to the high values of $|\sigma_{\boldsymbol{mn}}|$ and $|\sigma_{\boldsymbol{wm}}|$.

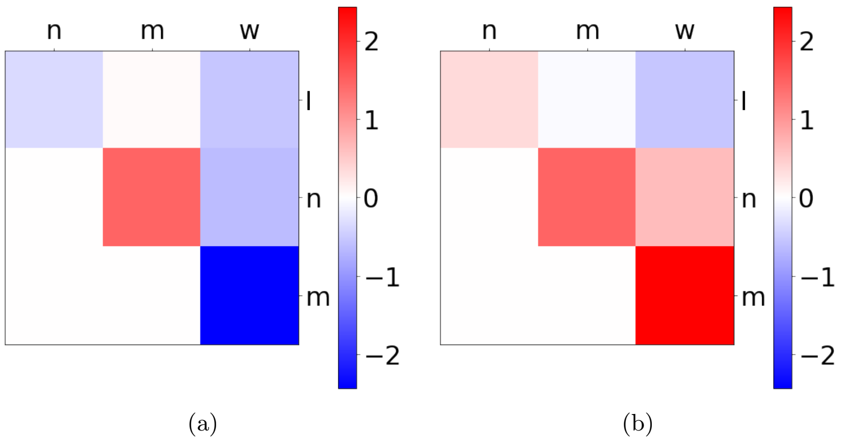

Figure 4: (a) A covariance matrix from a rank 1 network trained on the decision making task with negative covariance values. (b) The same covariant matrix but with $\boldsymbol{m}$ and $\boldsymbol{n}$ multiplied by -1.

In our modeling we found that the $\sigma_{\boldsymbol{nI}}$, $\sigma_{\boldsymbol{wn}}$, and $\sigma_{\boldsymbol{wn}}$ regularly become negative, however maintaining the same relative magnitudes. If $\boldsymbol{m}$ and $\boldsymbol{n}$ are multiplied by -1, all covariance values become positive. This is seen in Figure 4. This flip in covariance values does not change the computational dynamics in the system and likely is a result of the random initialization of parameter.

3.2 Network Dynamics

The dynamics in the network occur within the span of $\boldsymbol{m}$ and $\boldsymbol{I}$. This follows from the update function and the initial state of the network. The network adds a combination of vectors $\boldsymbol{m}$ and $\boldsymbol{I}$ to $\boldsymbol{x}$, as modulated by $\frac{1}{N}\boldsymbol{n}^T\phi(x)$ and $u(t)$ respectively. Given that the network starts with an initial value of zero, there is no opportunity for component vectors aside from $\boldsymbol{m}$ and $\boldsymbol{I}$ to span $\boldsymbol{x}$. It should be noted that $\boldsymbol{m}$ and $\boldsymbol{I}$ are often not orthogonal.

Without explicitly knowing the components that span the internal network states we could use a form of dimensionality reduction, such as PCA, to find a set of vectors that span the internal network states. Analysis of this sort, however, is outside the scope of this paper.

3.2.1 Trained Recurrent Network

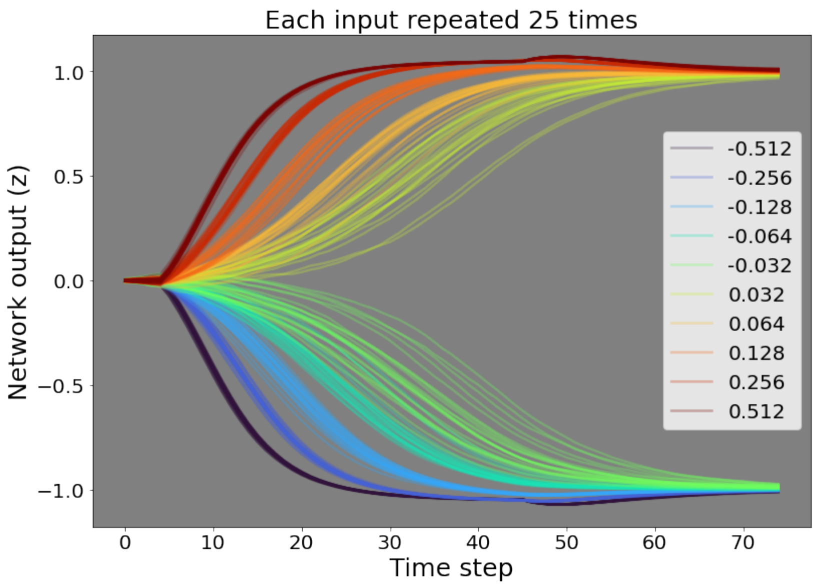

Figure 5: Network output for a trained RNN across 25 trials of each input. Each color represents a different $\bar{u}$ value, as stated in the legend.

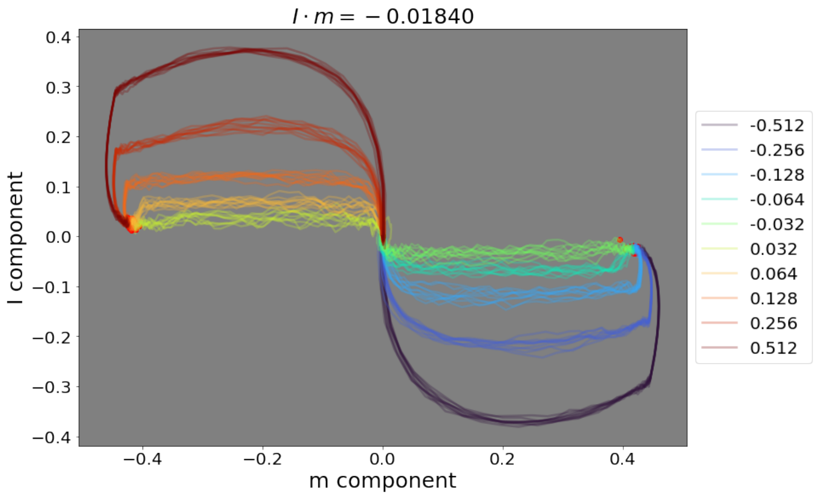

Figure 6: The dynamics of RNN hidden unit activity as projected onto $\boldsymbol{m}$ and $\boldsymbol{I}$ across 25 trials of each input. Each color represents a different $\bar{u}$ value, as stated in the legend.

Our model was trained over one epoch using a generated data set of 32,000 samples.

Across all trials the trained RNN made the correct decision. Trials with a $\bar{u}$ value of higher magnitude take less time to reach their decision value and have less variability between trials, as seen in Figure 5.

As can be seen in figure 6, $\boldsymbol{x}$ starts from the origin, and steadily gains value from both the $\boldsymbol{m}$ and $\boldsymbol{I}$ components while the stimulus $\bar{u}$ is present. Once $\bar{u}$ is removed from the stimulus, the $\boldsymbol{I}$ component of $\boldsymbol{x}$ reduces significantly towards zero. For most networks the $\boldsymbol{I}$ component will never reach zero, or even approach zero given a sufficient number of time steps, due to $\boldsymbol{m}$ and $\boldsymbol{I}$ not being completely orthogonal. In other words, the $\boldsymbol{I}$ component can never get to zero because there's a bit of $\boldsymbol{I}$ in $\boldsymbol{m}$. As seen in the figure, the magnitude of the $\boldsymbol{I}$ component is proportional to both the magnitude and sign of $\bar{u}$. In cases where the covariance matrix is predominantly negative, as explained in the previous subsection, this plot is flipped along the vertical axis.

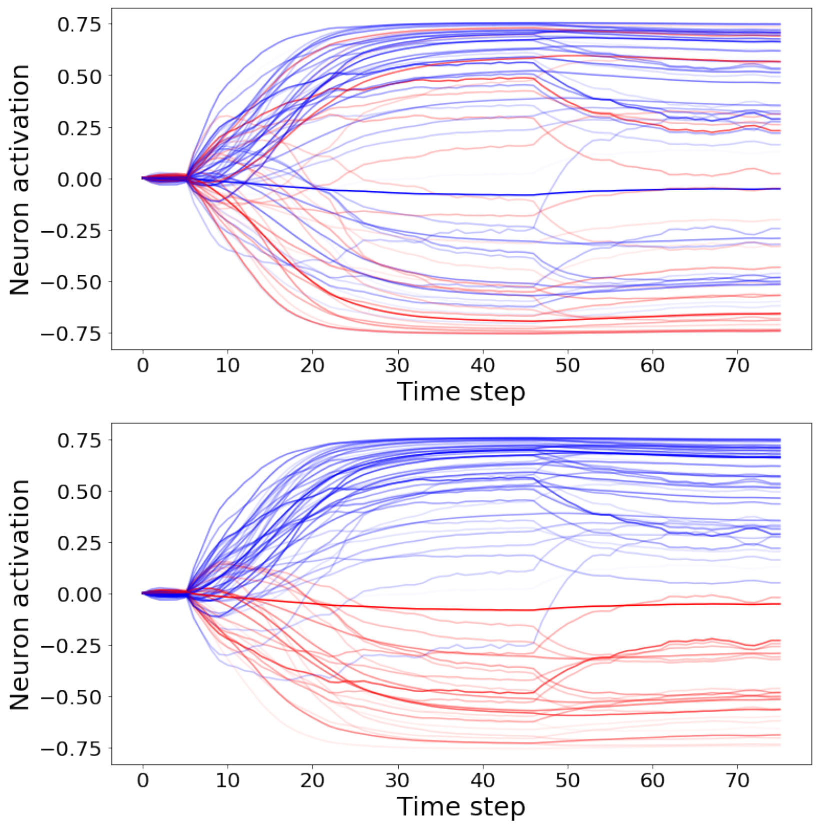

Figure 7: Neural activity $\phi(\boldsymbol{x})$ over time of one trial where $\bar{u} = 0.032 * 8$. The alpha value for each line is the corresponding value in $\boldsymbol{w}$ normalized by the maximum absolute value in $\boldsymbol{w}$. (a) Blue lines are neurons with a corresponding positive weight in $\boldsymbol{w}$ and red lines neurons corresponding with a negative weight. (b) Each neuron is multiplied by the sign of its corresponding weight. Blue lines are neurons that positively contribute to the network output, red lines are neurons that negatively contribute.

3.2.2 Resampled Recurrent Network

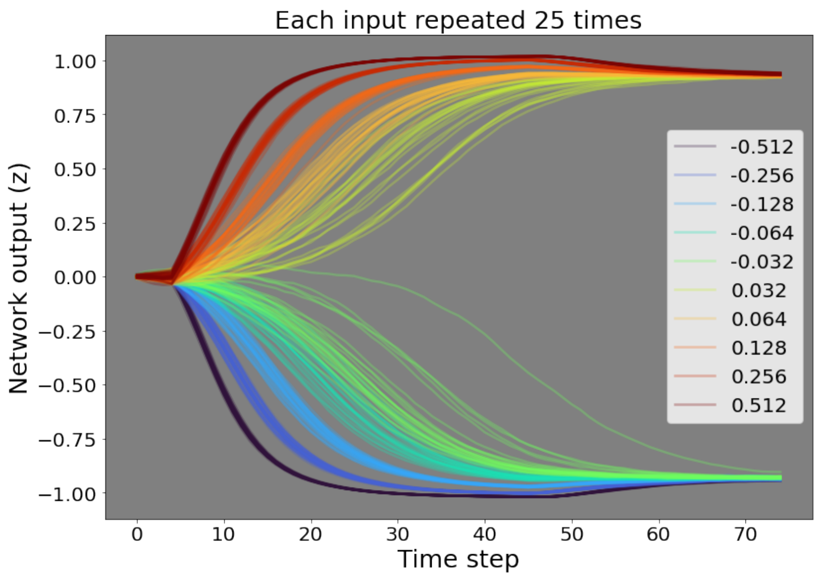

Figure 8: Network output for a resampled RNN across 25 trials of each input. Each color represents a different $\bar{u}$ value, as stated in the legend.

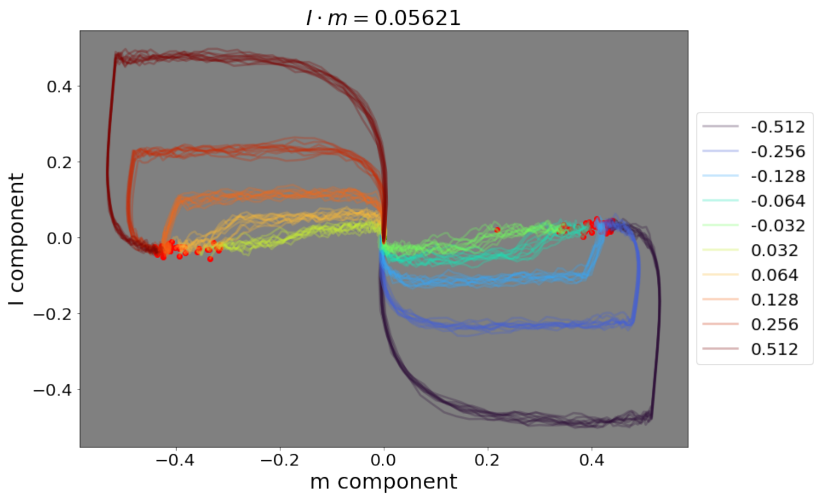

Figure 9: The dynamics of the resamples RNN hidden unit activity as projected onto $\boldsymbol{m}$ and $\boldsymbol{I}$ across 25 trials of each input. Each color represents a different $\bar{u}$ value, as stated in the legend.

All parameters in this network are sampled from a multi-dimensional Gaussian distribution with the same means and covariances as found in the parameter vectors of an equivalent RNN model. By doing so we are able to asses whether the correlation of the parameters is significant to performing the decision task.

During our testing resampled RNNs were able to successfully complete the decision task by outputting a value with the same sign as $\bar{u}$ by the end of the trial. Unlike what is seen in figure 8, the output of resampled RNNs did not always converge to 1 or -1. Occasionally they converged around other values, though these values were always symmetric. From these results we are able to see that indeed the structure of the covariance matrix is a significant aspect of the computation for the decision task.

3.2.3 Reduction to Dynamical System

The low rank RNN dynamics can also be reduced to a one dimensional dynamic system that is driven from statistics of the connectivity vector, mainly the parameter σ_{mn} and σ_{nI}, σ_{mw}

$x(t)$ can be composed to two components, the activity on the recurrent space, which in the rank one case is simply the sub space spanned by the vector m, and the component of the activity on the input space, which is the sub space spanned by the vector I (orthogonal input vector w.r.t m).

$$x(t) = \kappa(t)m + v(t)I \tag{7}$$

By projecting activation vector $x(t)$ on m, we will get the internal collective variable $\kappa(t)$, and $v(t)$, the external collective variable, is the projection on I.

The resulting dynamics of the collective variable is defined by projecting the dynamic of activation and by assumption of multivariate Gaussian distribution with mean zero for our connectivity pattern which can be expressed by following compact description:

$$\tau \frac{d\kappa}{dt} = -\kappa(t) + \tilde{\sigma}_{mn}\kappa(t) + \tilde{\sigma}_{nI}v(t) \tag{8}$$

$$\tau \frac{dv}{dt}(t) = -v(t) + u(t) \tag{9}$$

$$\langle \Phi' \rangle(\Delta(t)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi'(\Delta z) e^{-z^2/2} dz \tag{10}$$

$$\Delta(t) = \sqrt{\sigma_m^2 \kappa(t)^2 + \sigma_I^2 u(t)^2} \tag{11}$$

$$z(t) = \sigma_{mw}\kappa(t) \tag{12}$$

$\tilde{\sigma}_{mn}$ acts as the self feedback of $\kappa(t)$. $\tilde{\sigma}_{nI}$, is the effective coupling of between $\kappa(t)$ and the input $u(t)$. $\langle \Phi' \rangle(\Delta(t))$ is the average gain. Any of these effective couplings are computed by multiplying the covariances by the the average gain:

$$\tilde{\sigma}_{ab} = \sigma_{ab}\langle \Phi' \rangle(\Delta(t)) \tag{13}$$

External inputs $v(t)$ is filtered by the neuronal time constant $\tau = 100ms$.

Table 1: Rank 1 Covariances in the Paper vs. Trained Model

Covariance	Paper Value	Trained Model Value
σ_{mn}	1.4	1.4
σ_{nI}	2.6	-0.48
σ_{mw}	2.1	-1.9
σ_{m}	1	1.3
σ_{I}	1	0.97

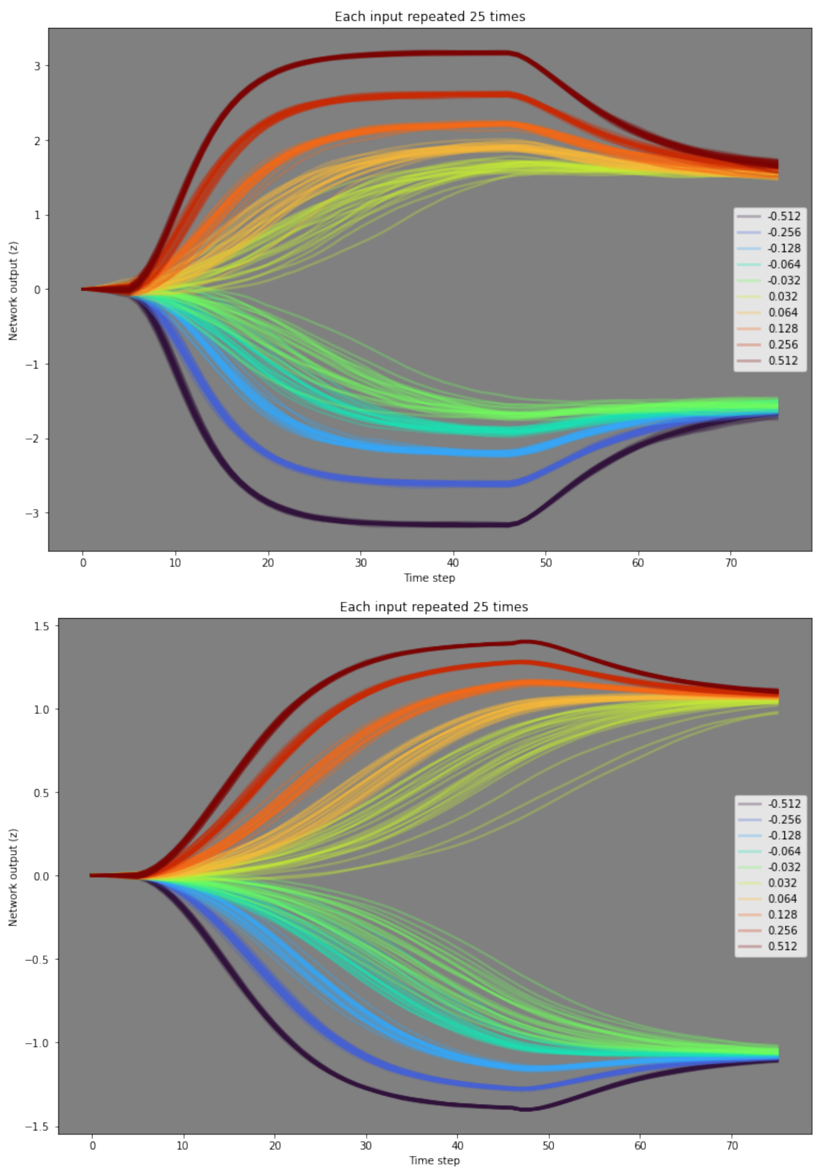

Figure 10: Reduced one dimensional dynamical system output for different input strength, (top) using given parameters, (bottom) trained model parameters. By deciding on the sign of the output both models perform well, although the given parameters of the paper results in converging to values higher than one.

The implementation was done both with our trained network statistic and given covariances from the paper. The performance of the reduced models are shown in Figure 10. Both reduction from trained model parameters and given parameters of the paper performed the task qualitatively well but we observed that the using the paper's parameters the dynamics converges to output values of greater magnitude than the target output, specially when the strength of the input is high. The reduced model from trained parameters quantitatively converges to the target values.

We are able to further analyze the reduced dynamic system by locating the fixed points of equation 8, which can be found by minimizing $\frac{d\kappa}{dt}$ given the dependant variable κ. By plotting $\frac{d\kappa}{dt}$ against κ we observe three fixed points, one at zero, which is unstable, and two symmetrically positioned around zero with similar absolute values. The value of κ, when initialized at zero, converges to one of the symmetric fixed points as determined by the sign of $\bar{u}$, regardless of the magnitude of $\bar{u}$. We notice that the choice of fixed point in the dynamic system using parameters from our trained model is opposite of using models given Debreuil et. al. [1] Results are illustrated in Figure 11.

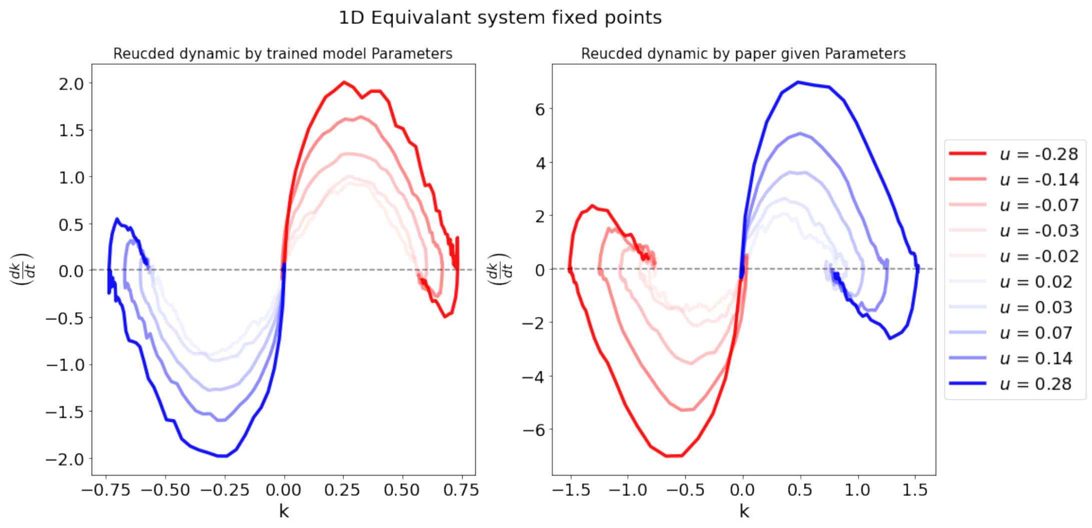

Figure 11: $\frac{d\kappa}{dt}$ for all possible values of the input strength, (left) trained model dynamics, positive strength input settle in the left fixed point, negative strength input settle in the right fixed point (right) given parameters dynamics showed the opposite behaviors, in addition the Scale of the changes match the dynamics of the output that has been shown in 10, the values of σ_{mw} has opposite sign for our model and Paper model. This justify the opposite dynamic $\frac{d\kappa}{dt}$ while similar output $z(t)$.

4 Parametric Working Memory

We trained a rank 2 network for this task. Each input, $u(t)$, consists of two periods of non-zero stimulus, u_1 and u_2 with a pause of p time steps in between. u_1 and u_2 are the mean adjusted values of f_1 and f_2 where drawn from a finite set and the maximum and minimum values of that set are f_{max} and f_{min} respectively. The expected output is the difference between u_1 and u_2. $u(t)$ is specifically defined as:

$$u(t) = \begin{cases} u_1 & \text{if } 5 \leq t \leq 10 \\ u_2 & \text{if } p+10 \leq t \leq p+20 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Where u_1 and u_2 are:

$$u_i = \frac{1}{f_{max} - f_{min}} \left(f_i - \frac{f_{max} + f_{min}}{2} \right), \ i = 1, 2 \tag{15}$$

And the expected output is:

$$y = \frac{f_1 - f_2}{f_{max} - f_{min}} \tag{16}$$

For each trial f_1 and f_2 were independently and uniformly sampled from $\{10, 14, 18, 22, 26, 30, 34\}$. It follows $f_{min} = 10$ and $f_{max} = 34$.

4.1 Standard Trained Recurrent Network

We attempted to train the network with a data set of 32,000 generated input-output pairs with $p = 50$, minimizing the mean squared error of the last 5 time steps. We were not able to get the network to converge to a sufficiently low training error, with the training error usually settling somewhere between 0.08 to 0.15. Due to this, we did not use or further explore this model.

4.2 Curriculum Learning Recurrent Network

In another attempt, we trained the network multiple data sets with successively increasing p values. The specific p values used were 25, 30, 35, 40, 45, and 50. Each data set contained 32,000 input-output pairs. We kept the mean squared error of the last 5 time steps as the error function. This network converged to a suitable training error of around 0.0002.

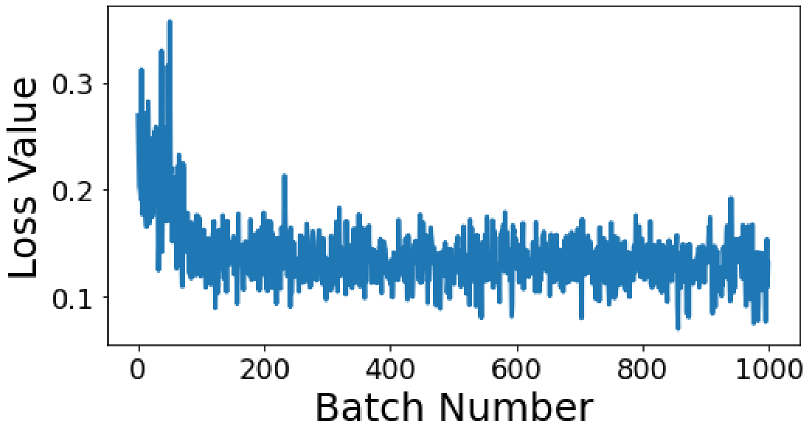

Figure 12: Loss curve for standard training, batch size 32, data set of 3200 trials, Loss defined as the mean squared error of the last 5 time steps.

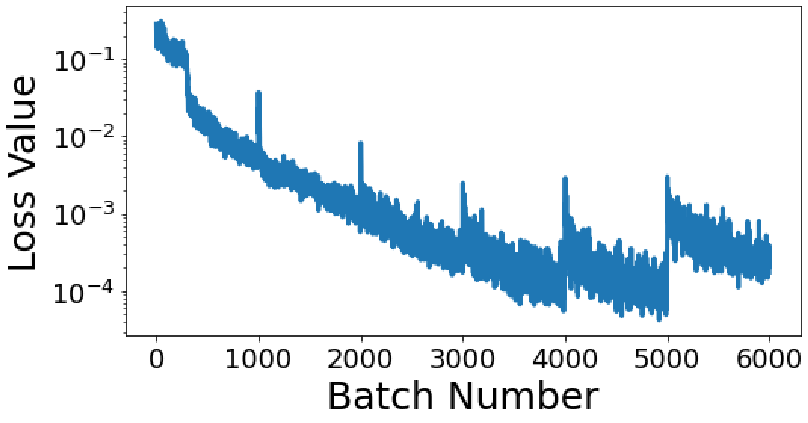

Figure 13: Training loss using curriculum learning. Note that the y-axis of this plot is logarithmic. We see spikes in the loss value every time we increment the value of p.

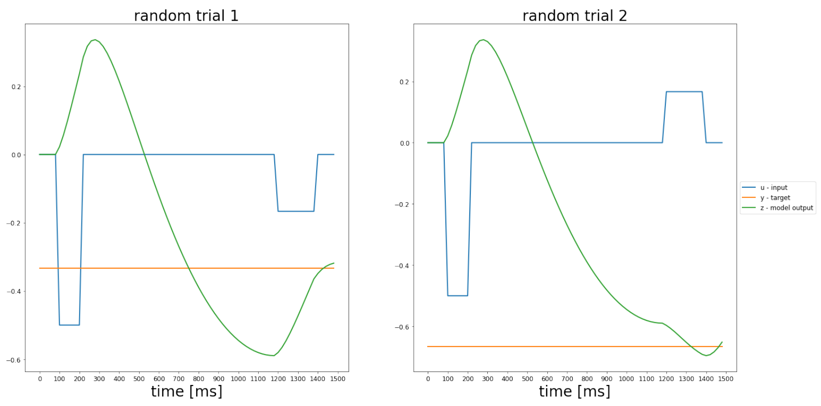

Figure 14: dynamic of the trained model for two randomly generated data sample of the parametric working memory task

4.3 Connectivity Patterns

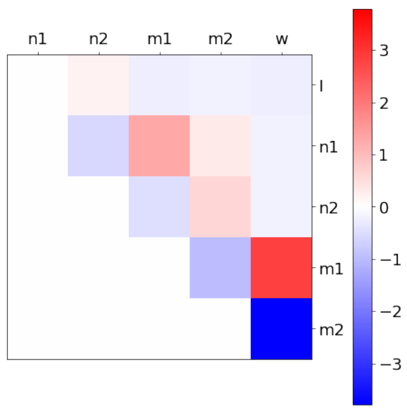

Figure 15: The covariance matrix for the rank 2 network trained with curriculum learning.

Our covariance matrix does not match the patterns found in the covariance matrix in Figure 3b of Dubreuil et. al. They found $\sigma_{\boldsymbol{n}^{(2)}\boldsymbol{I}}$ and $\sigma_{\boldsymbol{m}^{(1)}\boldsymbol{n}^{(1)}}$ to have the highest magnitudes. The highest covariance values in our matrix come from $\sigma_{\boldsymbol{w}\boldsymbol{m}^{(1)}}$ and $\sigma_{\boldsymbol{w}\boldsymbol{m}^{(2)}}$. This might be due to the longer training time for our rank 2 network, allowing these two parameters, to better align with $\boldsymbol{w}$.

Like the covariance matrix for our rank 1 network and like in Debreuil et al.[1] we find positive correlations for complimentary $\boldsymbol{m}^{(i)}$ $\boldsymbol{n}^{(i)}$ pairs.

4.4 Network Dynamics

4.4.1 Trained Network

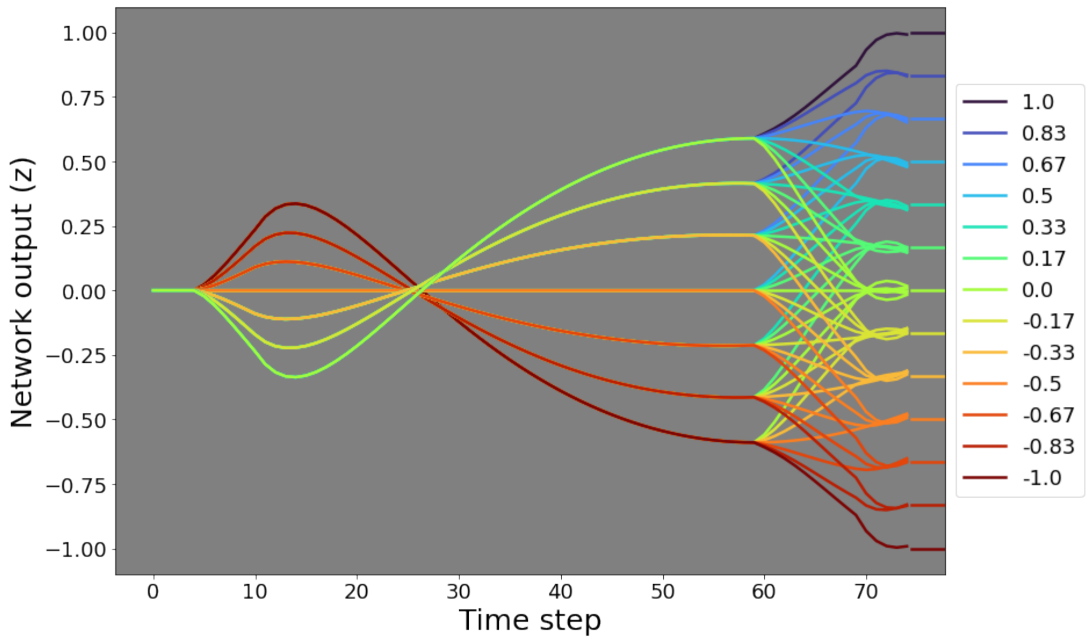

Figure 16: The trajectories of our rank 2 network trained with curriculum learning for every input in the parametric working memory task with $p = 50$. Each color represents the expected output of the trajectory and the colored lines on the far right of the plot are the values of the expected output.

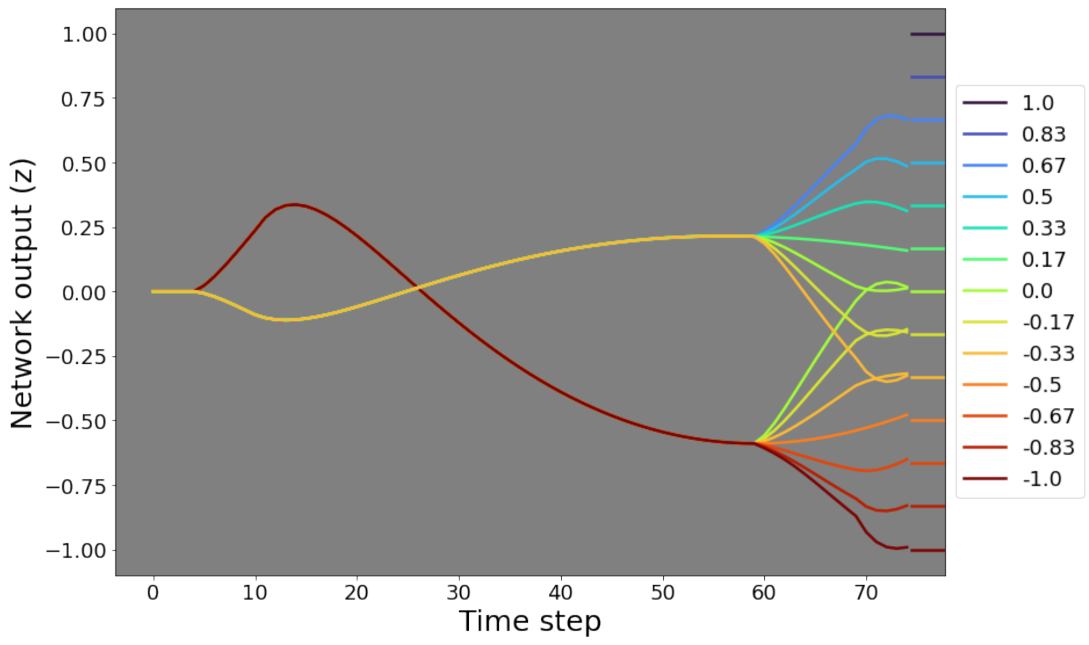

Figure 17: Trajectories of our rank 2 network trained with curriculum learning for $f_1 = 10, 26$ and all f_2 values with $p = 50$.

As can be seen in Figures 16 and 17 the rank 2 network trained with curriculum learning was able to successfully and accurately complete the working memory task.

4.4.2 Resampled Recurrent Network

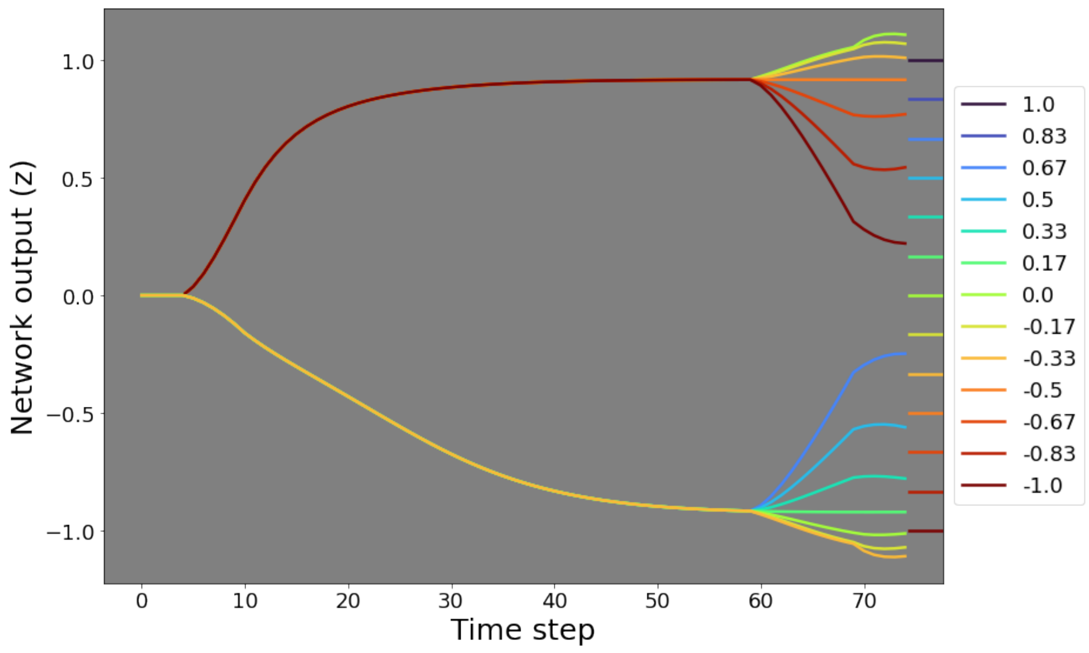

Figure 18: Trajectories of rank 2 network with resampled parameters for $f_1 = 10, 26$ and all f_2 values with $p = 50$.

Unlike the decision task, our resampled network performed very poorly on the working memory task, as seen in Figure 18. The network output in our resampled network failed to find the same or similar states as our trained network after it was presented with u_1. The resampled network continued to go in its initial direction rather than turning towards the opposite direction as is the case in the trained network. When presented with u_2 our resampled network moved towards its expected output, but did not make it close in any case during the remainder of the trial. This suggests that the distribution of the parameters in our trained network is not sufficient alone to describe the behavior of the network.

4.4.3 Reduction to Dynamical System

Similar to the 3.2.3, in the rank 2 network, dynamics can be reduced to two collective variable $\kappa_1(t)$ and $\kappa_2(t)$. $x(t)$ is composed of components in the first and second recurrent mode, a sub space spanned by m_1 and m_2, and component in the input space, I. Here $\sigma_{n_1 m_1}$ and the $\sigma_{n_2 m_2}$ are the self feedback weights for the two collective variables $\kappa_1(t)$ and $\kappa_2(t)$. The gain and average coupling are computed as before. External input $v(t)$ is again filtered by the neuronal time constant $\tau = 100ms$.

$$\tau \frac{d\kappa_1(t)}{dt} = -\kappa_1(t) + \tilde{\sigma}_{m_1 n_1}\kappa_1(t) + \tilde{\sigma}_{n_1 I}v(t) \tag{17}$$

$$\tau \frac{d\kappa_2(t)}{dt} = -\kappa_2(t) + \tilde{\sigma}_{m_2 n_2}\kappa_2(t) + \tilde{\sigma}_{n_2 I}v(t) \tag{18}$$

$$\tau \frac{dv}{dt} = -v(t) + u(t) \tag{19}$$

$$\Delta(t) = \sqrt{\sigma_{m_1}^2 \kappa_1^2(t) + \sigma_{m_2}^2 \kappa_2^2(t) + \sigma_I^2 v^2(t)} \tag{20}$$

$$x(t) = \kappa_1 m_1 + \kappa_2 m_2 + v(t) I \tag{21}$$

$$z(t) = \sigma_{m_1 w}\kappa_1(t) + \sigma_{m_2 w}\kappa_2(t) \tag{22}$$

Table 2: Rank 2 Covariances in the Paper vs. Trained Model

Covariance	Paper Value	Trained Model Value
σ_{m_1}	1	1.7
σ_{m_2}	1	1.4
σ_I	1	1.6
$\sigma_{n_1 m_1}$	1	1.2
$\sigma_{n_2 m_2}$	1	0.6
$\sigma_{n_1 I}$	0.5	0.02
$\sigma_{n_2 I}$	1.9	-0.2
$\sigma_{m_1 w}$	2.8	-2.8
$\sigma_{m_2 w}$	-2.2	3.8

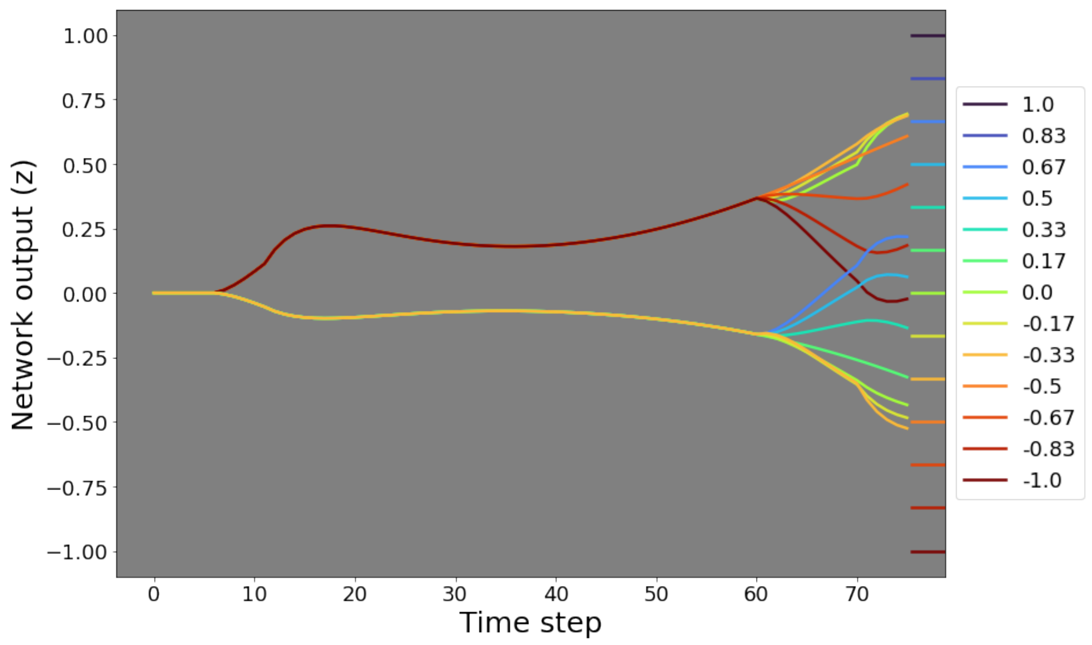

Figure 19: Output of two dimensional dynamical system using parameters from our rank 2 trained model. The above trials had $f_1 = 10, 26$ and all f_2 values with $p = 50$. The legend states the target value each trial was expected to reach. This model was unable to perform the task.

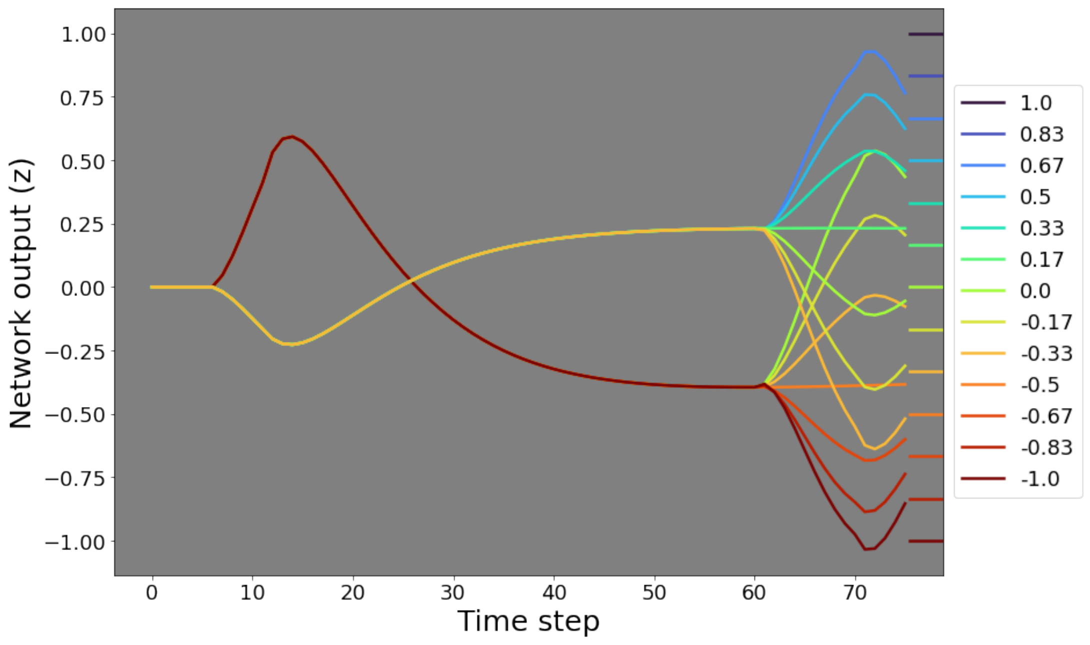

Figure 20: Output of two dimensional dynamical system using the parameters found in Dubreuil et. al [1]. The legend states the target value each trial was expected to reach. The above trials had $f_1 = 10, 26$ and all f_2 values with $p = 50$. This model was able to perform the task better than if it used our parameters, though still over or undershot the target value in many trials.

5 Discussion

During our experimentation all models that were applied to the perceptual decision making task were able to successfully generate network output that matched the sign of the stimulus. On parametric working memory task, however, most of our models were unable to get close to the target value. Our model trained using curriculum learning was able successfully complete the task, but only this model. The models tested by resampling network parameters, or reduction to a dynamic system failed to accurately or reliably hit target values. Assuming a correct implementation, we question whether using parameter resampling or reduction to a dynamical system is an useful way of describing the inner workings of low-rank recurrent networks.

References

[1] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. July 2020.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.