Decision making in low-rank recurrent neural networks

Myriam Hamon, Taisiia Tikhomirova

Introduction

This project approaches the question of how neural computations work. Based on the work of Dubreuil et al. we try to recreate their study that suggests that dimensionality of the dynamics and cell-class structure play central roles in neural computations modeling.

As stated in the paper and the project sheet, the low-rank framework offers an approach to understanding the brain's computational abilities. The challenge lies in determining the specific connectivity patterns necessary for specific computations. To address this issue, an alternative approach was found, involving training artificial neural networks to perform the desired tasks without explicitly specifying the connectivity patterns, leaving that as the remaining challenge. Even though the network in this case can perform the needed task - we cannot investigate the inner work of these networks due to their complexity and inability to be interpreted.

It was proposed to investigate recurrent neural network (RNN) models with low-rank connectivity, so it can be interpreted and comprehended. In this project, we focus on training and reverse-engineering such low-rank RNNs with the goal of distilling the high-dimensional RNN into an equivalent low-dimensional circuit.

The main roles in this project are played by the network connectivities, represented in terms of patterns over neurons. I begin a feed-forward input pattern, w - linear read out of the output, m and n specifying the recurrent structure of the model - m determining a principal direction of activity in state space and n determining which input patterns activate the corresponding mode. We also focus on kappa values, as we use them to form dynamical systems, representative to the dynamics of the networks. This way, as stated above, approaching the connectivity patterns helps us to look inside the decision-making process of the recurrent network and model it, using just the connectivity patterns.

Results

In our networks, the dynamics of the 128 units x_i were defined by the differential equation (1) and the output of the networks was defined by equation (3), with φ = tanh the activation function.

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij}\phi(x_j) + I_i u(t), \qquad i = 1, \dots, N \quad (1)$$

$$J = \frac{1}{N} m n^T \ (2)$$

$$z(t) = \frac{1}{N} \sum_{i=1}^{N} w_i \phi(x_i) \ (3)$$

Simulation of the network was done using a forward Euler and the training of the network was done by minimizing the mean-squared error using stochastic gradient.

Rank 1

For the rank 1, the decision-making task consisted of noisy input with different stimulus strength and the target was the sign of that stimulus strength (Figure 1a). The network was trained by minimizing the mean-squared error and it learned to do the task with a final loss of 0 and an accuracy of 100% (Figure 1b). The output over time of the trained network shows a clear differentiation of the 2 different labels over time (Figure 1c).

Once the network was trained, it was possible to look at the patterns of correlations between the connectivity vectors (Figure 1d-f). As expected, the correlation coefficient between w-I is very low, as they are both randomly generated and not trained, and the coefficient between n-m is rather high, which is similar to the paper. The other correlation coefficient seems to point toward a different pattern from the paper, but this could be due to the different architecture of the network (number of neurons), and might not matter to much in the task, as m and n are the most important vectors for the linear discrimination. This relationship between I and m is also confirmed by the scatter plots which show the low correlation between I and n, and the rather high correlation between n and m. Overall, this pattern of correlation works for the task as this task only requires a linear discrimination.

Once the network was functioning, it was possible to try and approximate it with a equivalent dynamical system, which is built on gaussian approximations and dimensionality reduction.

In order to test the Gaussian approximations, we fitted a 4-dimensional Gaussian distribution to the connectivity vectors, then sampled from that distribution and tested the resampled network's performance. The sampled distribution had similar histograms to the original distribution but showed additional noise and an overall less smooth histogram (Figure 2a). This resulted in a resampled network that had an accuracy of ~ 75% (Figure 2d). This drop in accuracy didn't affect the pattern of correlations, which can be seen with scatter plots of the original network and the resampled network that were compared using Kernel Density Estimation and showed very similar density distribution (Figure 2b-c).

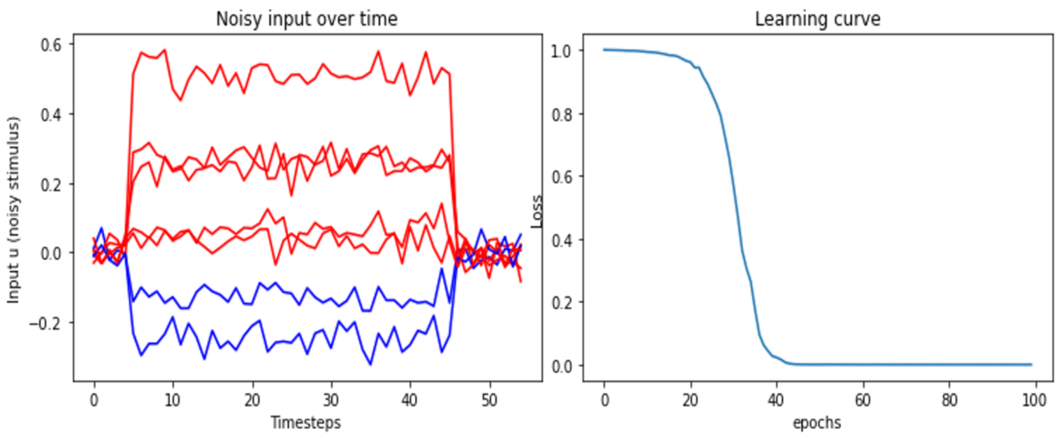
To verify the coherence of using dimensionality reduction, we verified that the dynamics of this model lived in the 2-dimensional space spanned by the vector I and m. These vectors are the only ones that are trained and they therefore perform the decision-making task. This is confirmed by projecting the dynamics x(t) onto the m – I plane (Figure 2d). In the case that m and I would not have been known, another way to do dimensionality reduction would have been Principal Component Analysis.

Once that both the gaussian approximation and the dimensionality reductions assumptions are verified, it was possible to build the equivalent one-dimensional dynamical system. The system was able to perform the task with an accuracy of 100% (Figure 2e).

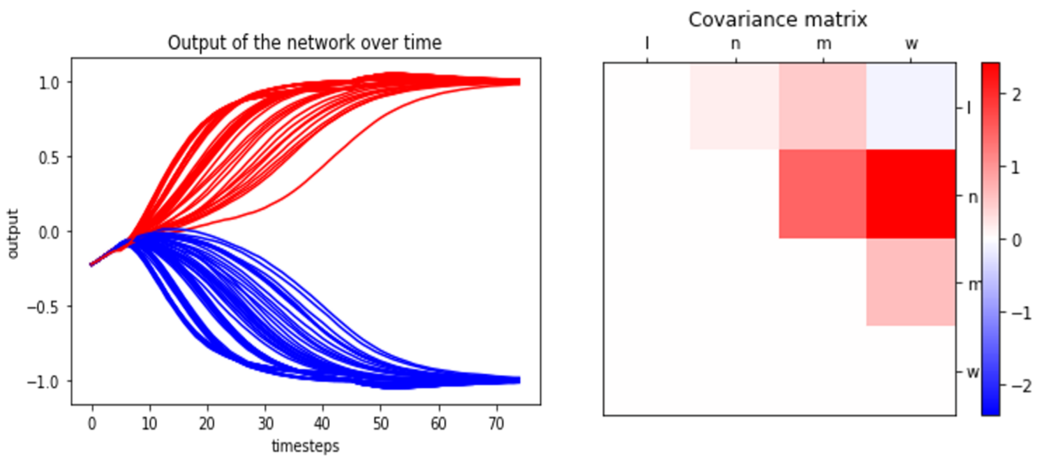
$$\frac{dk}{dt} = -k + \tilde{\sigma}_{mn} k(t) + \tilde{\sigma}_{nI} v(t)$$

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

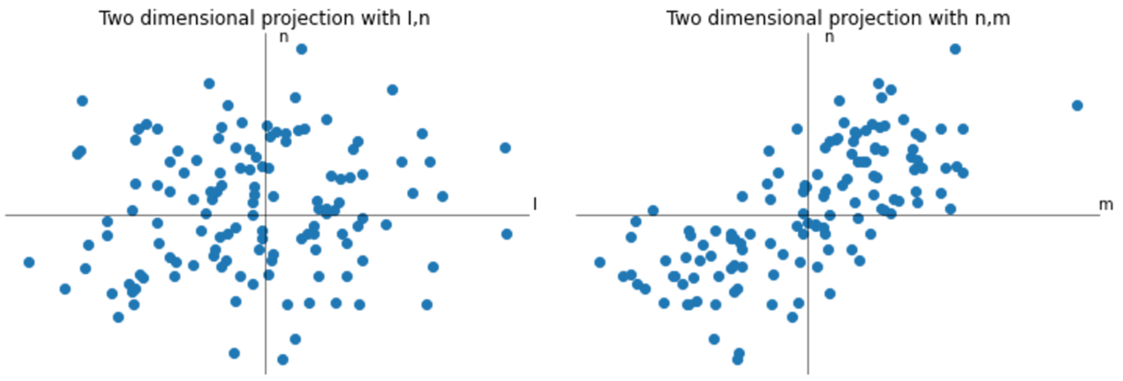
To better understand this system, we evaluated the fixed points. Using the function minimize of scipy, the fixed point that was found was 0, and in a one-dimensional system with the sign of the input being the threshold, a fixed point of 0 makes sense as it is the threshold.



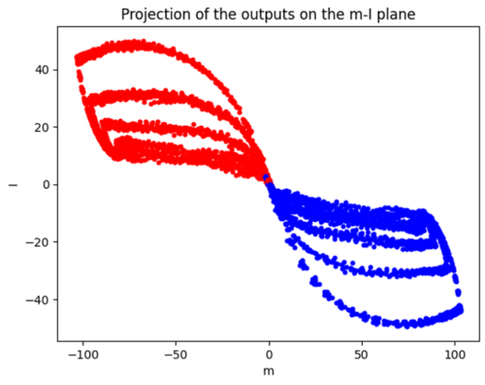
(a) (b)



(c) (d)



(e) (f)



(g)

Figure 1: Construction of network rank 1 and statistics. (a) is the input given to the network, with red lines describing inputs having reached the upper threshold and blue lines describing inputs having reached the lower threshold. (b) is the learning curve of the network over the epochs. (c) is the output of the trained network over time (accuracy 100%). (d) Covariances between input, connectivity and readout pattern. (e) and (f) are the 2 dimensional projections of the loading space. (g) projection of the output over time onto the m-I plane.

(a)

(b)

(c)

(d)

(e)

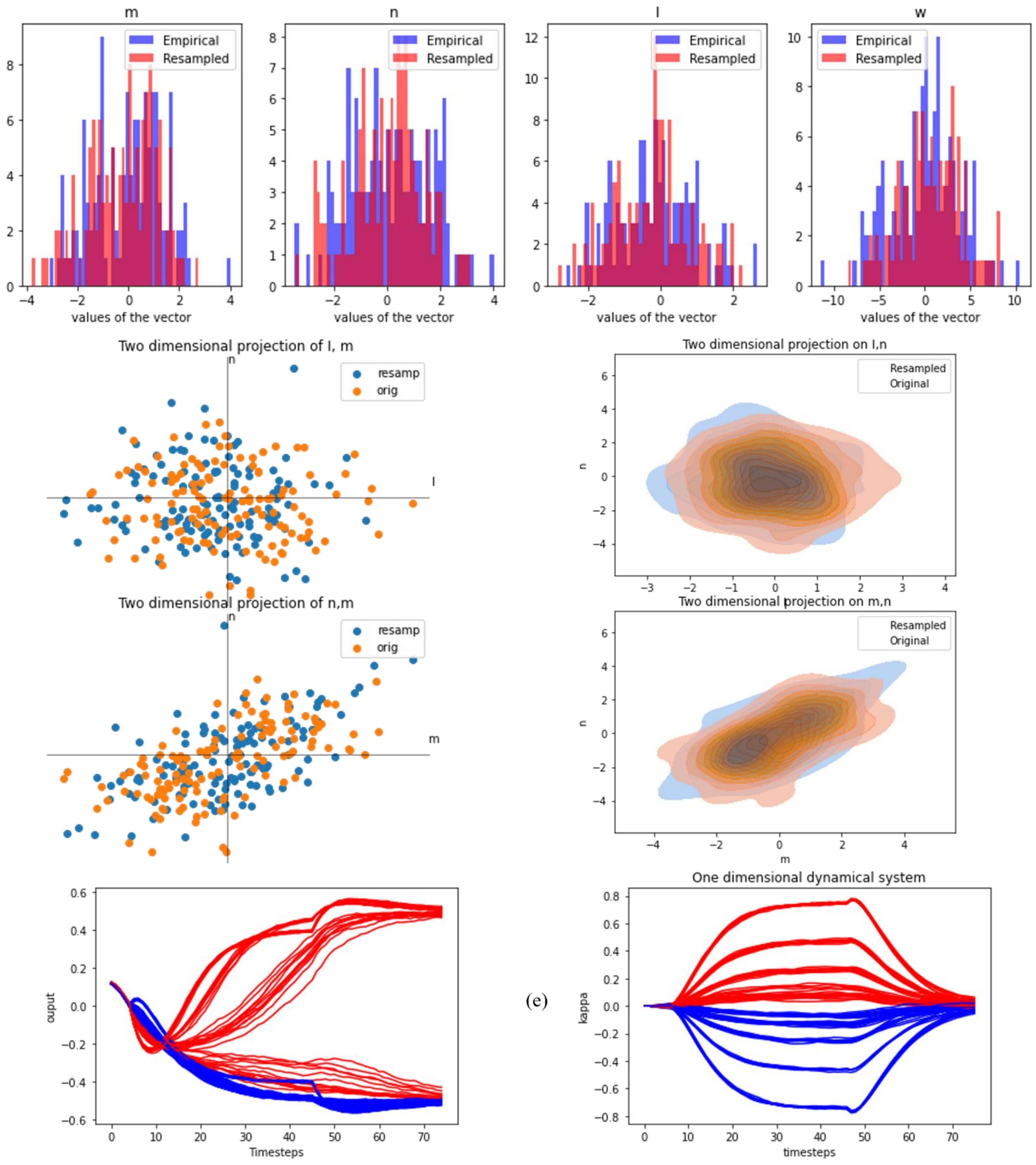


Figure 2: Verification of the Gaussian assumption for the network rank 1. (a) Distributions of the connectivity vectors (m,n,I and w) for the original and resampled networks. (b) and (c) are the 2 dimensional projections of the loading space. Right : scatter plot of the projection. Left : Kernel Density approximation of the projection. (d) output over time of the trained resampled network (final accuracy 75%). (e) output over time of the equivalent one dimensional system.

Rank 2

A rank-1 network means that the rank of the connectivity matrix (m) and the right-connectivity matrix (n) is 1, which implies that they can only capture linear relationships between the neurons. Linear relationships are limited in their ability to capture complex patterns and non-linear transformations of data.

As stated in the paper, the task of parametric working memory needs R = 2 connectivity with two internal collective variables, such that the first variable accumulates the memory of the first stimuli during the delay period.

Knowing that, after working on the low-rank RNN we wanted to check how this framework applies to a higher-rank network and whether we can create a higher-dimensional equivalent circuit of 2D RNN. To do this all the steps described before are recreated, but the shapes of the left and right connectivity vectors m and n are changed to *[hidden size, new rank]* - hidden size being 128 neurons of the hidden layer, and new rank being 2.

The data generation function was changed to create input *u* consisting of two stimuli with a 50-timesteps delay between them and target *y* being the normalized difference between the stimuli (Figure 3a) . For this part, we generated 100 input-target pairs.

The architecture of the network remained the same. After training we reached the loss value of 0.005 (Figure 3b). In the Figure 3c, we can see that the network was able to grasp two “pulses” of our stimuli that result in “decisions” being made. For this task the decision, as a target value, was presented as the normalized difference between the stimuli. That is why we can see two distinct destination points at the end of the timescale.

The obtained connectivity patterns differed from those reported in the paper (Figure 3d) - this can be explained by the different structure of the networks (precisely - different number of hidden neurons).

As reported in the paper, m2 connectivity vector shows positive correlation with I input connectivity vector and n1 connectivity vector, but in our case it has a small negative connectivity with n2. Also n2 shows negative connectivity with the I vector, which is not reported by the paper. Pairs m1 - n1 and m2 - n2 are negatively correlated.

This raises the question of why those values work. Our theory is that the covariation signs in this case are not as important as the comparability of the projections of the six-dimensional loading space into the connectivity vectors space (Figure 3e-f) and they, as those reported in the paper, are normally distributed).

To obtain latent kappa values k1 and k2 the definition of kappa values from the paper was used - projection on activation vector on m space. We see Figure 3g that the mean of kappa sequences across all represent the output of our RNN network quite well, as their behavior is comparable with the dynamics of the outputs - one kappa value for negative outputs, and one for positive.

$$K_r(t) = \frac{1}{\|m^{(r)}\|^2} \sum_{j=1}^{N} m_j^{(r)} x_j(t)$$

After training the network we tested if the covariance patterns characterize the network connectivity by fitting and resampling a 6-dimensional Gaussian to the connectivity patterns and after resampling connectivity vectors m1,m2,n1, and n2, as well as input and output connectivity vectors I and w the loss of the network with resampled connectivity was 0.136. In the Figure 4a-e we can see that the original and resampled connectivity distributions are closely aligned.

After that, a two-dimensional circuit with given dynamics and coupling terms was simulated and compared to the 2-rank RNN model.

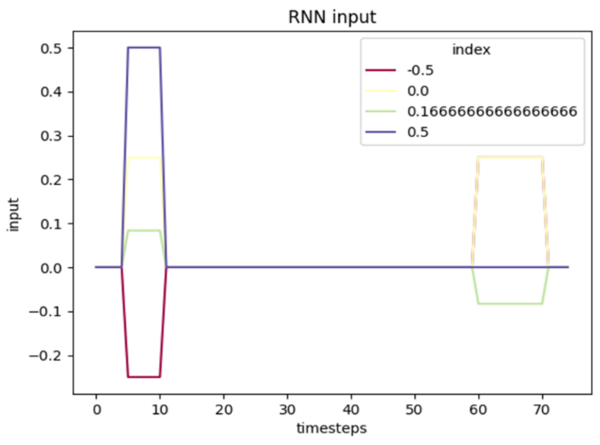
$$\frac{dk_1}{dt} = -k_1 + \tilde{\sigma}_{m^1 n^1} k_1 + \tilde{\sigma}_{n^1 I} v(t)$$
$$\frac{dk_2}{dt} = -k_2 + \tilde{\sigma}_{m^2 n^2} k_2 + \tilde{\sigma}_{n^2 \mathrm{I}} v(t)$$
$$\Delta = \sqrt{\sigma_{m_1}^2 \mathrm{k}_1^2 + \sigma_{m2}^2 k_2^2 + \sigma_{\mathrm{I}}^2 v^2}$$

The simulated circuit behaved almost the same as the trained network, but all the signs were reversed. As stated in the task - it behaved qualitatively but not qualitatively the same (Figure 4e)

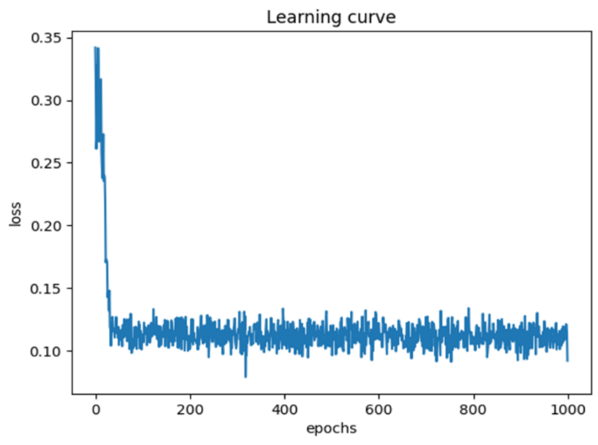
Discussion

It seems like the harder is the computation we want to model , the more complex should be the connectivity structure, leading to the higher rank models. And if we are able to interpret 1- and 2- rank connectivity structures, it yet seems unclear how to work with more complex dynamics. For example, will we be able to model motor control circuits, combining afferent and re-afferent inputs, state-predictions, sensory feedback and motor commands? Obviously, we can reduce and simplify this procedure, but it would not be representative to the neural computations anymore. So this is the main question and limitation - how much we learn from this model about how neural computations work?

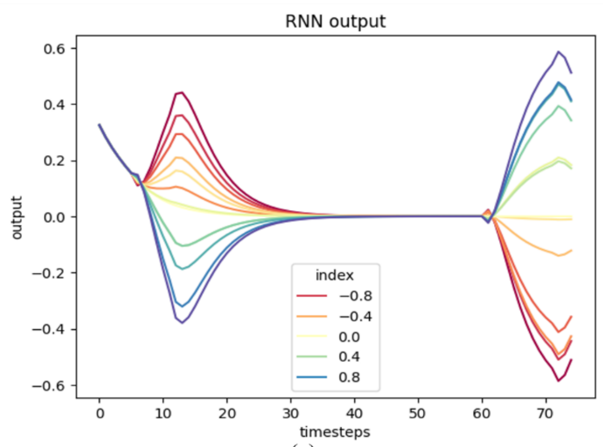
In the meantime, from what we studied during MHBF lectures, it seems like this approach can be used to investigate different variations of decision-making tasks (perceptual, multi-sensory, context-dependent) as well as parametric working memory, perceptual bistability & multisensory integration.



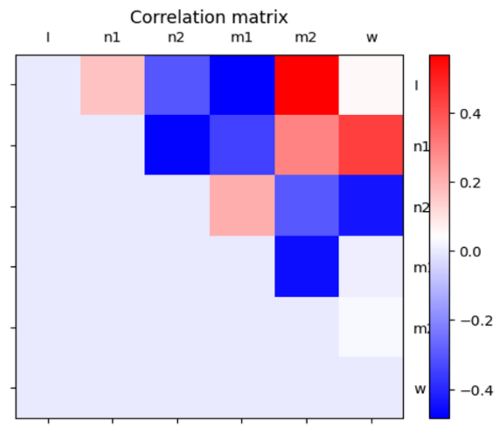
(a)



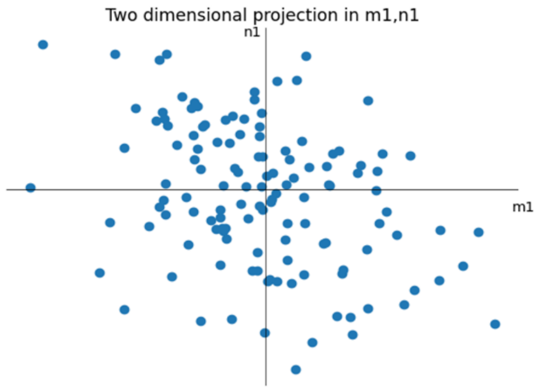
(b)



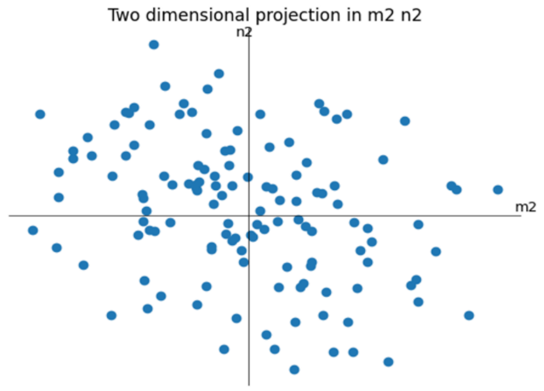
(c)



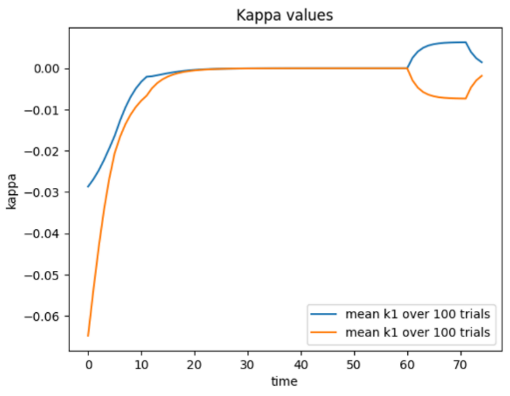
(d)



(e)



(f)



(g)

Figure 3: Construction of network rank 2 and statistics. (a) is the input given to the network. (b) is the learning curve of the network over the epochs. (c) is the output of the trained network over time (accuracy 100%). (d) Covariances between input, connectivity and readout pattern. (e) and (f) are the 2 dimensional projections of the loading space. (g) Latent variables K1 and k2 over time

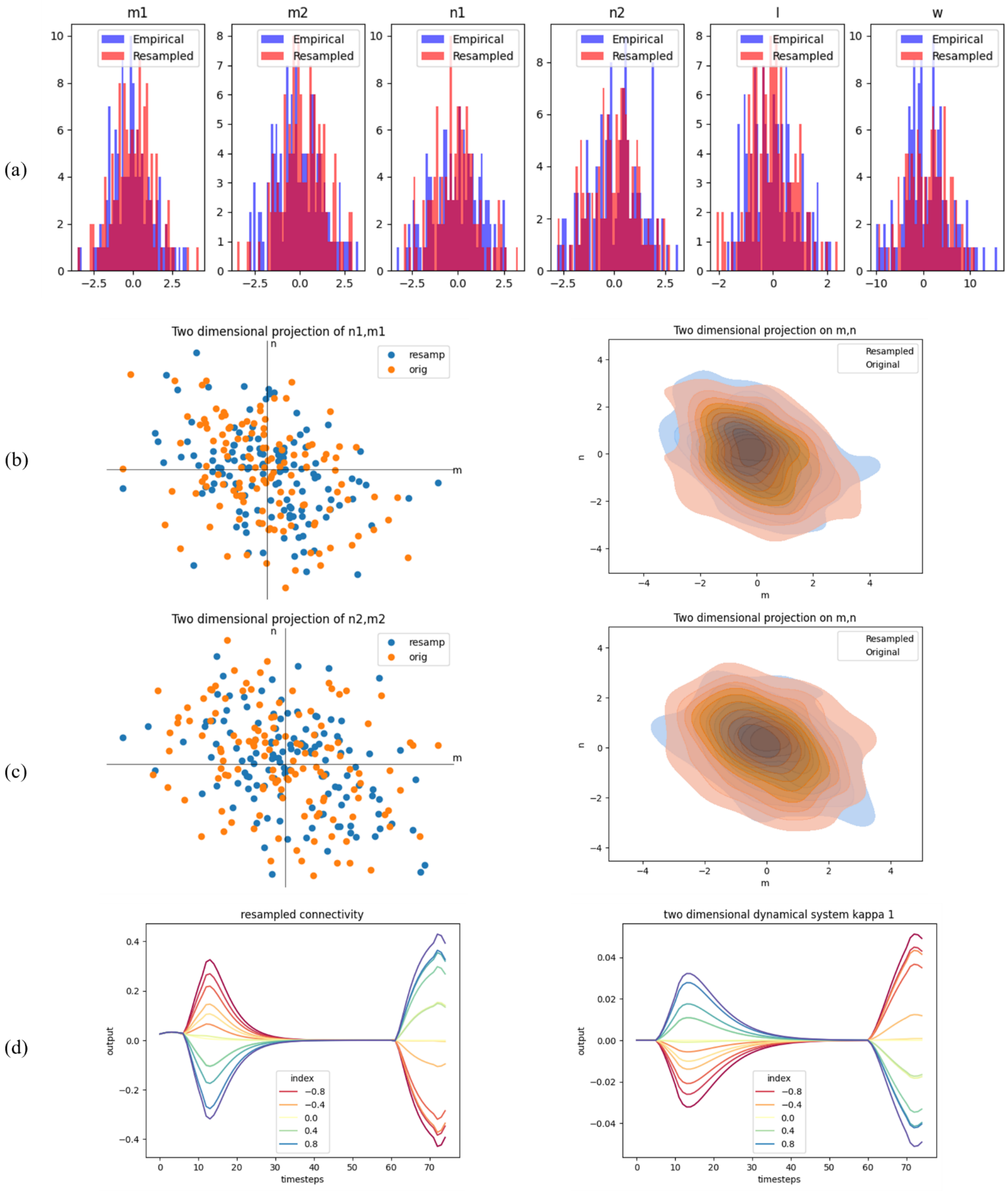


Figure 4: Verification of the Gaussian assumption and equivalent circuit for network rank 2. (a) Distributions of the connectivity vectors (m,n,I and w) for the original and resampled networks. (b) and (c) are the 2 dimensional projections of the loading space. Right : scatter plot of the projection. Left : Kernel Density approximation of the projection. (d) output over time of the trained resampled network (final accuracy 75%). (e) output over time of the equivalent two dimensional system.