# An Investigation of Memory in Recurrent Neural Networks

Anonymous Authors[1]

## Abstract

We investigate the learned dynamical landscape of a recurrent neural network solving a simple task requiring the interaction of two memory mechanisms: long- and short-term. Our results show that while long-term memory is implemented by asymptotic attractors, sequential recall is now additionally implemented by oscillatory dynamics in a transverse subspace to the basins of attraction of these stable steady states. Based on our observations, we propose how different types of memory mechanisms can coexist and work together in a single neural network, and discuss possible applications to the fields of artificial intelligence and neuroscience.

## 1. Introduction

Recurrent neural networks (RNN) are widely used to carry out tasks that require learning temporal dependencies across several scales. Training RNN's to perform such tasks offers its share of challenges, from well-known exploding and vanishing gradients, to the difficulties of storing, accessing, and forgetting memories (Pascanu et al., 2013; Bengio et al., 1994). Viewed as dynamical system, the activity structure of recurrent network state spaces can reveal how networks learn tasks, and can help guide training and architecture design. In this study, we perform a dynamical system analysis of a trained RNN on a simple tasks that requires two types of memory paradigms interact: short-term memory of past inputs and a delayed output during classification.

While gating units found in LSTM (Schmidhuber & Hochreiter, 1997) and in a variety of other architectures (e.g., Cho et al., 2014; van der Westhuizen & Lasenby, 2018) directly aim at addressing these long-scale temporal learning issues, they are always used in conjunction with so-called "vanilla" recurrent units that shoulder the majority of computation. It is not yet well understood how internal network dynamics supported by such circuits combine information from external inputs to solve complex tasks that require remembering information from the past and delaying output changes. On one hand, attractor networks are a known solution to keep finite memories indefinitely (Hopfield, 1982). On the other, orthogonal transformations (e.g., identity and rotations) are used to build explicit RNN solutions to recall tasks (Jing et al., 2016; Vorontsov et al., 2017; Arjovsky et al., 2015). Indeed, for the well-studied copy task, where a sequence of symbols needs to be outputted after a long delay, it is known that the best solution is to use rotations to store the sequences, much like clocks that align at the time of recall (Henaff et al., 2016). However, it is unclear how attractor dynamics and orthogonal (rotational) transformations interact when a task requires both long term memory and sequential recall. We explore this situation here.

Leveraging slow-point analysis techniques (Sussillo & Barak, 2013), we uncover how short-term memory tasks with delayed outputs give rise to attractor dynamics with oscillatory transients in low-dimensional activity subspaces. Our result uncovers how the boundaries of basins of attractions that are linked to memory attractors interact with transverse oscillatory dynamics to support timed, sequential computations of integrated inputs. This provides novel insights into dynamical strategies to solve complex temporal tasks with randomly connected recurrent units. Moreover, such transient oscillatory dynamics are consistent with periodic activity found throughout the brain (Llinás, 2014), and we discuss the impact of our findings on computations in biological circuits.

## 2. Preliminaries

In this paper we replicate and advance further the study of Sussillo & Barak (2013) on the implementation of memory tasks in artificial RNNs. This simple analysis allows for careful examination of the inner workings of RNNs, which we leverage here to study the interaction of two distinct memory mechanisms in it with the introduction of delayed recall in a memory task.

While the original study (i.e., Sussillo & Barak, 2013) used continuous time, we used a discrete-time network and trained it with standard optimization algorithms. This is in

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena. Do not distribute.

order to verify if the original findings translate well between the different types of network, and to explore network architectures that are more widely used in the machine learning community. Detailed description of our setup is provided in Section 3.

We base our analysis on a densely connected single-layer RNN with a hyperbolic tangent activation function. Specifically, let $X$ be the input signal, $S$ the hidden state of the network and $Y$ the output of the network. We use $N = 100$ neurons with input matrix $W_{in}$, output matrix $W_{out}$, and recurrent connection (i.e., used to update the hidden layer over time) matrix $W$ . Then, the update rule for the hidden states and outputs of the network are defined as:

$$S(t) = \tanh(WS(t-1) + W_{in}X(t))$$
$$Y(t) = \tanh(W_{out}S(t))$$

For training, we unfold the network in time, up to ten timesteps, and use the standard Adam (Kingma & Ba, 2014) optimization algorithm. Only the $W$ and $W_{out}$ matrices were trained. See Appendix A for more details.

We characterized the behavior of the neural circuit using tools from dynamical system theory, which enable us to identify internal states and transitions over time in the neural circuit. In particular, we use slow point analysis, developed in Sussillo & Barak (2013), to find approximate fixed points. Formally, let $S(t)$ be the state of the network at time $t$. We define a Lyapunov function $q$ inspired by the physical formula for potential energy. We adapt this original approach to a discrete-time system as

$$q = \frac{1}{2}\left|S(t+1) - S(t)\right|^2 \tag{1}$$

The minima of this function indicate the fixed points of the system, and their types (e.g., sink or saddle point) can be determined by the number of positive eigenvalues of the Jacobian matrix at that point.

## 3. Problem setup

We use a delayed *3-bit flip-flop* task to illustrate our findings, as it requires the interaction of two types of memory mechanisms and lends itself to easily interpretable network dynamics. The network has three independent channels with an associated input ($W_{in}$) and output ($W_{out}$) neuron each. Each input neuron sends short spikes of magnitude 1 or -1 at random time intervals (Poisson distribution with homogeneous rate $\lambda = 50$) and is set at zero otherwise. The output channel must maintain the value of the last input spike sent by its associated input neuron.

To study the interaction between long term memory and sequential recall, we add a time delay to this task. This in turn requires the network to delay its change of internal state upon the reception of a novel input by a period $\Delta t$. Representative examples of both tasks are demonstrated in Figure 1.

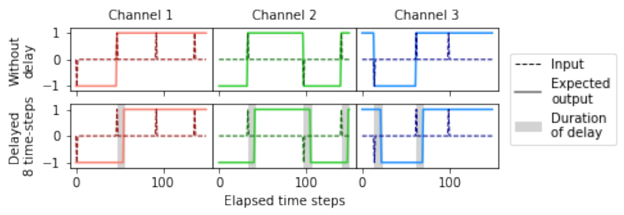


*Figure 1.* Task definition. The neural network is expected to match and maintain the last input for each channel while preventing cross-talk. For the delayed task definition, the neural network must, in addition, delay its response (and therefore remember the input) during a certain number of time steps by $\Delta t$, illustrated in grey.

## 4. Time-delay impact on internal RNN state space

We begin by replicating the observations from Sussillo & Barak (2013) in our discrete-time setting. We tested the trained network during roughly a hundred thousand steps, during which the network receives approximately 2000 inputs, saving the neurons states and the network output at every step. We then analyzed the resulting dynamics, identifying slow and fixed points with slow-point analysis, and projecting dynamics in the first three principal components (PC), i.e., using PCA.

As we can see in figure 2a, we do observe the characteristic “cube” discovered in the original research. Here, each vertex represents a certain output state (since there are three binary channels, this means there are eight possible states), with adjacent vertices varying by a single output. The slow point analysis also revealed attractors at the corners of the cube described by the path of the states, saddle points with one unstable dimension on the edges, saddle points with two unstable dimensions on the faces and a saddle point with three unstable dimensions at the center. The attractors implement long-term memory by “trapping” trajectories in the absence of inputs. Each of the eight attractors encodes a specific output state. Saddle-points channel the dynamics during a switch of output state. Their associated stable subspaces form the separatrices between the basins of attraction of the stable fixed points on each side. The configuration of fixed points observed here is also consistent with the findings of Sussillo & Barak (2013).

We next turn to the trajectories produced by the network trained to perform the delayed task (see Figure 2b). The position of the “corner” attractors is fairly similar to that of the zero-delay case. However, the steady-state dynamics

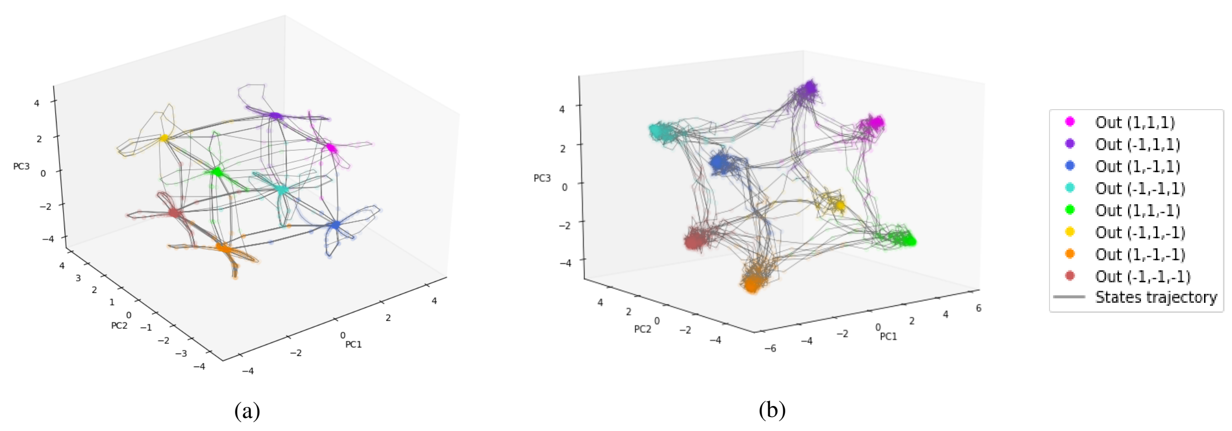


*Figure 2.* **(a)** Dynamics during 3000 time-steps without delay, projected in the first three principal components. Each point represents the network's state at a different time-step and is color-coded according to the effective output value at that time. In the absence of input, the network will maintain its position in phase-space at one of 8 stable fixed points, encoding the current binary combination of the 3 input channels. Injecting an input in the network induces a switch of the network towards different attracting states, and thus, produces distinct outputs. **(b)** Same as in (a) for dynamics with delayed output. The network must generate the same output as before but delay its output change for $\Delta t = 8$ time-steps.

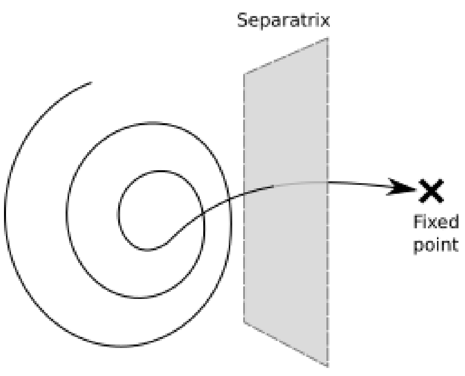
surrounding these corners have changed considerably. One hint about the hidden dynamics implementing sequential recall is the "loops" protruding from the corners of the initial cube and disappearing with the addition of delay. Since we use a tanh activation function that limits outputs between 1 and -1, overshooting the attractor assures that the correct output is expressed immediately after receiving the input, while undershooting it would give a visible transient period. However, in the delayed task, inputs have to be treated in a subspace orthogonal to the output space since no output change must be expressed before the delay has elapsed. This means that the input pulse is now invisible in the output space. With these observations in mind, we would like to further investigate the different hidden dynamics brought by the added delay. Since standard PCA and slow-point analysis failed to sufficiently inform us about this aspect, we now turn to triggered activation averages described in Section 5.

## 5. Characterization of memory mechanisms with spectral analysis

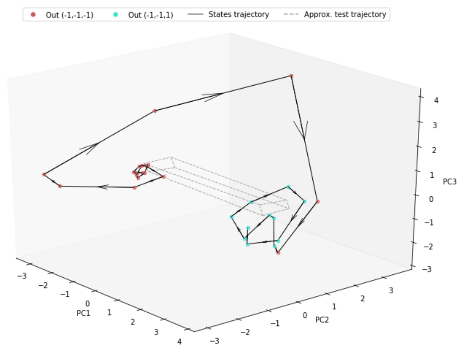
For simplicity, we focus at this point on characterizing a single type of transition in the network. Without loss of generality, we chose the transition from state $(-1, -1, -1)$ to state $(-1, -1, 1)$, and searched every instance during testing when the network had to perform this switch. We note that this choice of input switch is arbitrary but fixed, although other switches can also be analyzed in a similar manner. For each instance, we kept a trajectory of RNN states (i.e., hidden-layer activations) starting a few steps before the switch and ending a few steps after it. We then averaged these trajectories over instances, aligned at the switch time, in order to obtain a single short average trajectory (in the hidden-layer activation space) representing the mean activity surrounding the examined switch. PCA projection of this new signal is shown in Figure 3.

In figure 3c, we see a decomposition akin to Fourier modes of a step function, with increasingly rapid oscillations occurring in one forth of a period phase shifted pairs. Here, individual neurons appear to lock into dynamical regimes associated to precise frequencies. From the way the cube is flattened in figure 3b, we also conclude that the subspace where the rotational transformations are implemented is mostly orthogonal to the separatrices of the attractors corresponding to the different memorized outputs. A schematic of the transition is presented at figure 3a. We note that such rotating dynamics have indeed been observed previously when implementing sequential recall in artificial neural networks, and it is known that initializing RNNs with orthogonal rotation matrices helps solve sequential recall tasks (Henaff et al., 2016).

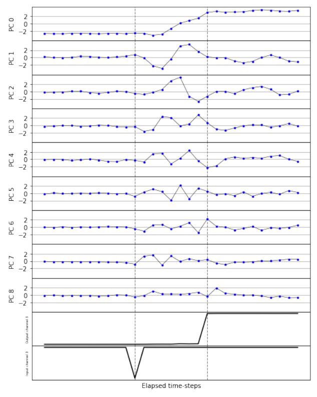
The periodic nature of the observed signals motivated us to explore the eigenspectrum of the matrix $W$ (see Figure 4). Indeed, when considering the complex eigenvalues of this matrix, their complex parts correspond to rotating speeds in different subspaces. For comparison, the network without delay has only three eigenvalues that distinguish themselves from the random cloud around zero, which is expected from random Gaussian matrices. These are on or close to the real axis, and slightly higher than one. Considering the hyperbolic tangent activation function applied at every time-step, this indicates three dimensions in which the state compo-



(a)



(b)



(c)

*Figure 3.* **(a)** Schematic of the transition for the delayed task. Oscillatory dynamics perpendicular to the separatrix delays the passage to the new fixed point used for the long term memory classification task. The rotational dynamics are a result of the implementation of sequential recall, they do not appear in the network without delay. **(b)** The switch from output (-1,-1,-1) to output (-1,-1,1) for the delayed network was studied using a triggered average. The resulting signal was then plotted according to the first principal components. For clarity, a cube is used to mark the approximate trajectory of neurons states during the test. Trajectory in the space spanned by the first three principal components. **(c)** Same trajectory but with the first 9 principal components represented.

nents are maintained at about 1.

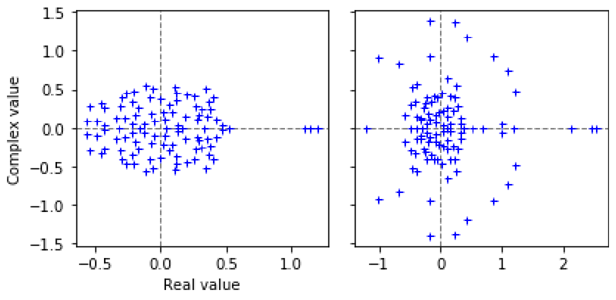


*Figure 4.* Analysis of the eigenvalues of the recurrent connection matrix $W$ for the studied RNNs. The eigenvalues of the network without delay is displayed to the left and the ones for the 8 time-steps delayed network to the right.

In the delayed network, the rotating dynamics are clearly indicated by eigenvalues with large imaginary parts, in addition to the three real eigenvalues corresponding to the long term memory subspace. This additional information is consistent with our interpretation that oscillatory dynamics implement sequential recall, much like orthogonal RNNs do during simple recall tasks (Jing et al., 2016; Vorontsov et al., 2017; Arjovsky et al., 2015; Henaff et al., 2016). However, in this case they do so in a localized fashion to facilitate interaction with other computation mechanisms implemented by the network.

## 6. Conclusion & discussion

We have seen in this study that long-term memory and sequential recall can be implemented by a simple RNN fairly easily, and in parallel, by acting on different subspaces of the RNN phase space. Specifically, sequential recall is achieved by rotational dynamics localized around the origin, which occur in a subspace orthogonal to the separatrices of the basins of attraction that solve the classification task. Our findings suggest that this population-level periodic activity may serve as a general "precision timing" mechanism that can be combined with distinct, learned computations. Indeed, oscillations enable the introduction of small delays, transverse to low dimensional activity of recurrent neural circuits. An interesting line of future work would be to investigate more thoroughly this mechanism in the presence of distinct computational tasks, such as character-level prediction, or arithmetic operations. We believe that learning a delayed recall in conjunction with any task will lead to generic, emergent oscillations that enable transient dynamics transverse to the subspaces used to perform other computations. It may be possible to leverage this geometric understanding for faster training by initializing networks in a way that promotes transverse rotations.

Furthermore, this oscillatory mechanism is consistent with observations of oscillatory dynamics in the brain (Llinás, 2014). Together with the known phenomena whereby neurons in the brain perform tasks with low-dimensional activity patterns, and that the same neurons engage in oscillatory activity when viewed at the population-level, our findings are consistent with a general principle of delayed recall in neural networks, either biological or artificial.

## References

Arjovsky, M., Shah, A., and Bengio, Y. Unitary Evolution Recurrent Neural Networks. *arXiv.org*, November 2015.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv.org*, June 2014.

Henaff, M., Szlam, A., and LeCun, Y. Recurrent Orthogonal Networks and Long-Memory Tasks. *arXiv.org*, February 2016.

Hoerzer, G. M., Legenstein, R., and Maass, W. Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning. *Cerebral Cortex*, 24(3):677–690, March 2014. ISSN 1047-3211. doi: {10.1093/cercor/bhs348}.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, April 1982.

Jing, L., Shen, Y., Peurifoy, J., Skirlo, S., LeCun, Y., and Tegmark, M. Tunable Efficient Unitary Neural Networks (EUNN) and their application to RNNs. *arXiv.org*, December 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 0028-0836. doi: 10.1038/nature14539.

Llinás, R. R. Intrinsic electrical properties of mammalian neurons and CNS function: a historical perspective. *Frontiers in cellular neuroscience*, 8:320, 2014.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. *ICML (3)*, 2013.

Schmidhuber, J. and Hochreiter, S. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Sussillo, D. and Barak, O. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013.

van der Westhuizen, J. and Lasenby, J. The unreasonable effectiveness of the forget gate. *arXiv.org*, April 2018.

Vorontsov, E., Trabelsi, C., Kadoury, S., and Pal, C. On orthogonality and learning recurrent networks with long term dependencies. *arXiv.org*, January 2017.

## A. Implementation details

The network was a one-layered discrete and densely connected recurrent neural network with a $N = 100$ neurons. The network is composed of three real-valued matrix, the square matrix $W$ representing the recurrent connection between the neurons, the $3 \times 100$ $W_{in}$ matrix representing the strength of connections between the input channels and the neurons of the network and the $100 \times 3$ $W_{out}$ matrix representing the strength of the connection between the neurons of the network and the output neurons. At each neuron, the signal goes through an "activation function", in this case the hyperbolic tangent. A schematic of the neural network is shown on figure 5a.

The hyperbolic tangent is often used as the activation function in studies about neural networks as it is a good representation of biological spiking neurons. It models the way the number and strength of excitatory and inhibitory signals entering the neuron affects its spiking frequency (Hoerzer et al., 2014).

A discrete, as opposed to a continuous-time, network was chosen for a few reasons. First, we wanted to see if the findings of Sussillo & Barak (2013) could be transposed to a discrete-time network, as the researchers themselves where wondering in their paper. This was further motivated by the fact that in the field of artificial intelligence, discrete-time networks are widely used (LeCun et al., 2015) as they are better adapted to the computer architecture. Any link between the study of neural networks for neuroscience and for artificial intelligence would be beneficial to both fields. Last but not least is the fact that discrete-time networks are much easier and faster to implement than their continuous counterparts, and a wide array of resources to train them are freely available.

The $W_{in}$, $W$ and $W_{out}$ weights matrices were originally set randomly (using a linear probability function) between $-\frac{1}{\sqrt{3}}$ and $\frac{1}{\sqrt{3}}$ for $W_{in}$ and between $-\frac{1}{\sqrt{N}}$ and $\frac{1}{\sqrt{N}}$ for $W$ and $W_{out}$ where $N$ is the number of neurons. This strategy is often used in the literature to get an initial output with a probability distribution approaching a normal law with average 0 and variance 1. Only the $W$ and $W_{out}$ matrices were trained.

Since the network is recurrent, the output is potentially the result of an infinite number of steps, which complicates training. The solution is to "unfold" the network, i.e., to consider only a certain number of steps back in time during training. In figure 5b, we see the unfolded version of the network. By trial and error we chose a 10 time-steps deep unfolded network for training.

## B. Additional figures

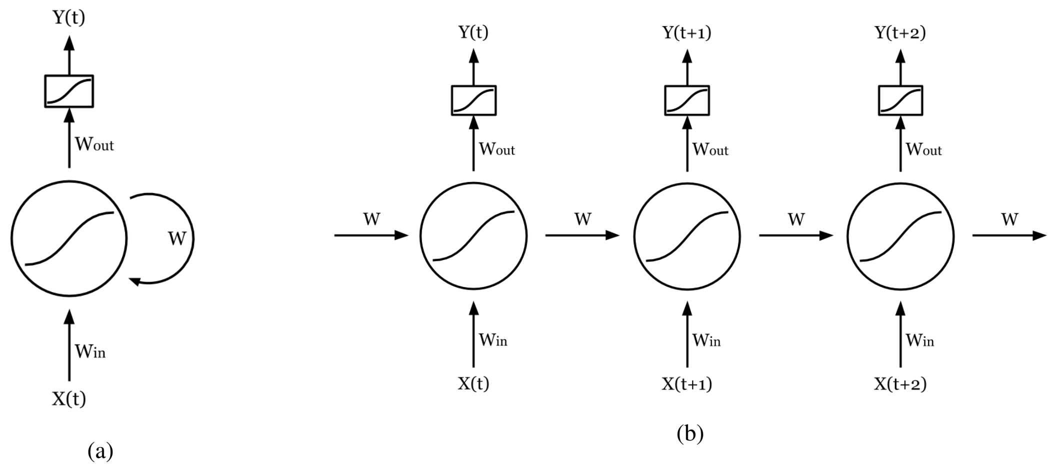


*Figure 5.* **(A)** Diagram of the recurrent neural network. Each time step, the neurons receive the previous states of all neurons through a weight matrix representing the strength of the connecting axons. Every neuron in the network uses the hyperbolic tangent as their internal function. **(B)** The unfolded diagram. A truncated unfolded version of the recurrent neural network is used during training.

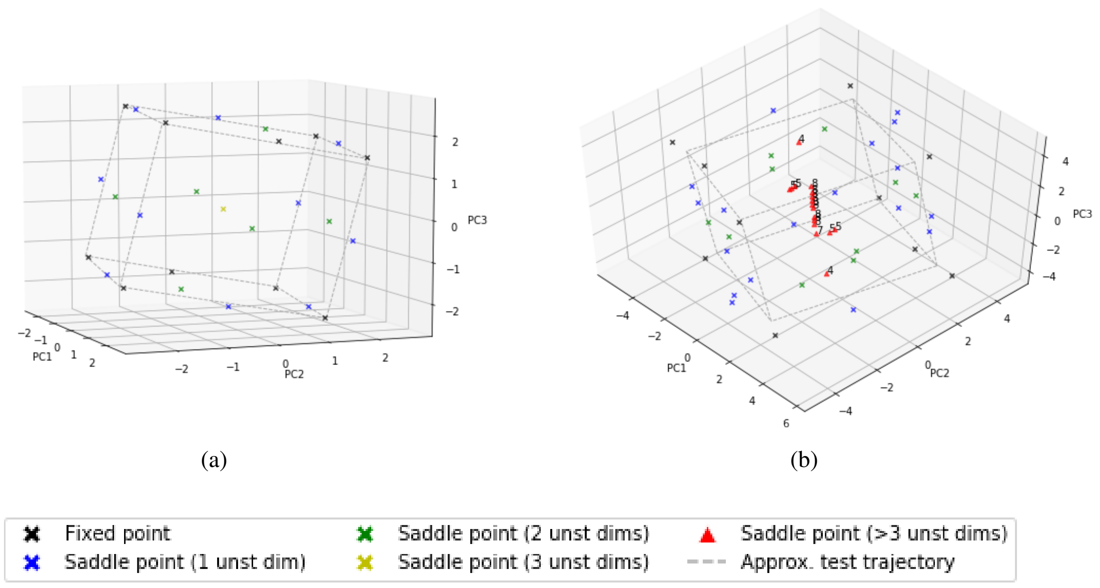


*Figure 6.* **(a)** Slow point analysis of RNN without delay. For clarity, a cube is used to mark the approximate trajectory of neurons states during the test. The slow points are color coded with respect to their number of unstable dimensions. This is consistent with the findings of Sussillo and Barak. **(b)** Slow point analysis of RNN delayed by 8 time-steps. There are now slow points with more than three unstable dimensions. Those are marked by red triangles and the number of unstable dimensions is written above. We posit that this is a result of the networks' capacity being stretched to the limit (it could not perform the task for longer delays) and not an actual change in strategy for the long-term memory task.