A RESERVOIR COMPUTING MODEL OF EPISODIC MEMORY

David Bhowmik, Kyriacos Nikiforou, Murray Shanahan
Department of Computing
Imperial College London
London, UK

Michail Maniadakis, Panos Trahanias
Institute of Computer Science
Foundation for Research and Technology Hellas
Heraklion, Greece

***Abstract* — We present a novel neural episodic memory architecture that utilizes reservoir computing to extract and recall information gleaned over time from a multilayer perceptron that receives sensory input. Reservoir computing models project input data into a high-dimensional dynamical space and also serve as a fading memory that holds on to past inputs thereby enabling the direct association of the current input with the past. The architecture presented utilizes these capabilities via an abstract feedback mechanism and in doing so creates attractor-like states within the reservoir that are associated with each discrete memory and associates these states and therefore memories over time into episodes. In addition, the feedback mechanism provides stabilization to an otherwise chaotic complex dynamical system.**

Keywords: reservoir computing, episodic memory, attractors, stabilization.

I. INTRODUCTION

In recent years, the neural networks community has shown increasing interest in reservoir computing (RC), which provides a new and powerful biologically inspired computation paradigm. The main advantage of RC is that it projects the input data into a high-dimensional space and therefore simple learning methods, e.g. linear regression, can be used to train the readout. Moreover, the reservoir, a pool of neurons with random recurrent connections, serves as a fading memory that holds past inputs therefore enabling the direct association of the current input with the past. In that way, the reservoir acts as a spatiotemporal kernel, projecting the input signal onto a high-dimensional feature space [1]. The transient dynamics developed in the reservoir provide substantial processing power to the network, enabling properly trained readout neurons to easily extract problem specific information.

Reservoir computing is typically distinguished from traditional recurrent neural network approaches in that, learning does not adapt the whole network but is focused only on readout neurons [2]. Previous studies showed that optimal network performance is accomplished when perturbations to the system's trajectory in its phase space neither spread nor die out [3; 4]. To improve performance further, new RC schemes have been proposed as summarized in [2].

This work has been partially supported by the EU FET grant (GA: 641100) TIMESTORM - Mind and Time: Investigation of the Temporal Traits of Human-Machine Convergence and EPSRC Doctoral training fund (Award reference 1506349).

Interestingly, the inherent capacity of RCs to integrate and process temporally neighboring information seems particularly appropriate for modeling episodic memory which assumes incremental and directional information recall from the storage medium [5; 6]. However, when working on strongly time dependent problems, the use of a single layer of neurons all of them working on the same time scale does not help making complex associations that involve different time steps. In such cases, the use of feedback may provide enhanced computing power to the network [7], by refreshing and providing second level processing of past experiencing. This is particularly important for episode recall because bringing in a memory and processing a bunch of events facilitates recalling some new memories which may be further combined with the former ones to bring some even newer memories and so on.

Intuitively, the use of feedback as an auxiliary input may enhance the overall performance because it helps shifting the internal dynamics in the directions that make them better combinable into the desired output values. Even if complex transitions over time constitute an important mechanism for episodic recall, the use of feedback often gives rise to instability issues [8]. The reason for these instabilities is that even if the model can predict the signal almost accurately, going through the feedback loop, small errors get amplified, making the output gradually diverge from the target.

The present work puts forward a novel RC architecture capable of memorizing and reconstructing a sequence of images, representing the recall of an episode from memory. At each processing step we use a feature-based compressed and therefore abstract form of the input images to be learned as part of the output and feedback. The use of image features in the processing loop facilitates the abstraction and compression of the scene in memory, without however reducing the amount of information that may be recalled back.

The architecture can be seen in figure 1. In brief, it works as follows: On recall a pulse identifying an episode that was learnt is sent to the reservoir causing it to place itself in an appropriate unique position in state space. This state space position is the same as where it was positioned at the onset of

978-1-5090-0620-5/16/$31.00 ©2016 IEEE

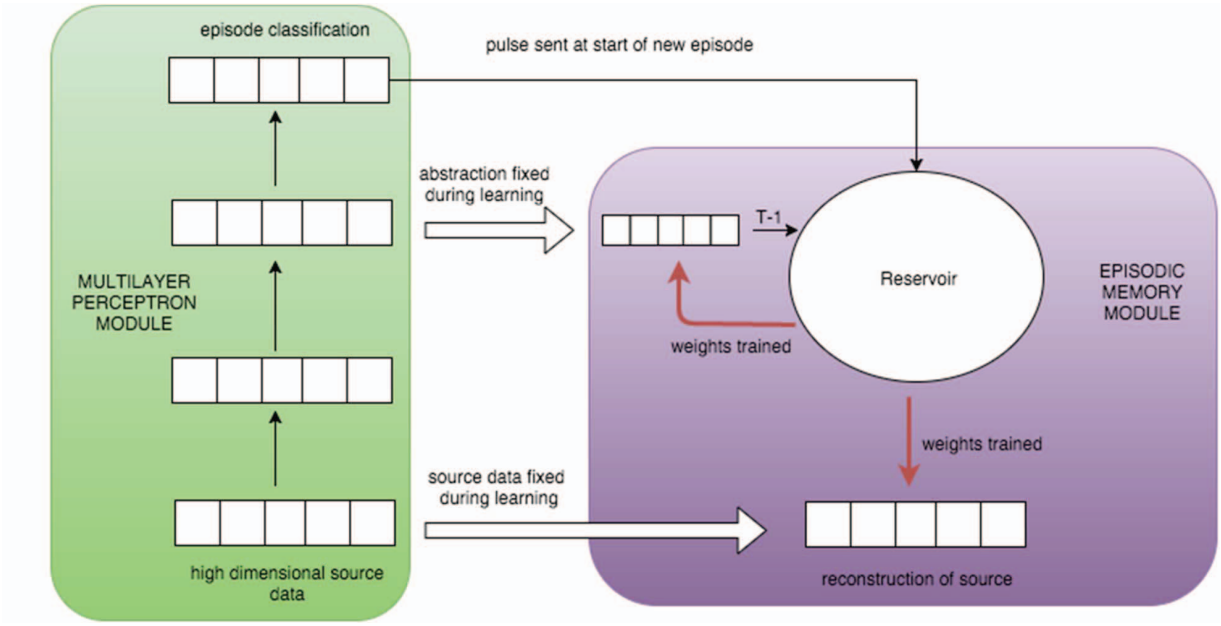

Figure 1. Episodic Memory Architecture. The architecture consists of an MLP that provides input over time into a reservoir.

learning the same episode. The state space positions are associated during learning to an image and its feature-based abstraction. During recall the reservoir output weights are used to reconstruct both a feature-based abstraction of the input image at time point T as well as a 16384 pixel high dimensional image at time point T. The feature-based abstraction is then used as feedback into the reservoir in order to drive the state space trajectory forward in a stable, guided fashion until it reaches the position at time point T+1 that was arrived at the same time during learning. This state space position is next used by the reservoir output weights to construct the feature-based abstraction at time point T+1, as well as the 16384 pixel high dimensional image at time point T+1. The latter abstraction is then used as feedback to drive time point T+2, and so on.

The obtained results show that the proposed architecture can successfully store and recall many whole episodes preserving temporal association between neighboring moments. Moreover, the use of the compressed input as part of the feedback nearly eliminates stability issues even when the system works under noisy conditions.

II. Methods

The neuron model chosen for the reservoir implementation is the continuous time rate based neuron model developed by Sompolinsky [9]. Randomly connected networks of these neurons have been shown to exhibit a transition from a stationary phase to a chaotic phase depending on the value of the gain parameter g_G of the model. In this work, a suitable learning algorithm, namely the First-Order Reduced and Controlled Error (FORCE) learning approach developed by Sussillo and Abbott [10], has been used to stabilize the reservoir and reproduce the high dimensional data by modifying the feedback and readout connectivity matrices respectively. The methodology presented here has been inspired from to the work of Laje and Buonomano [11] for generating motion patterns from recurrent neural networks and adapted for the architecture developed. The details of the neuron model and the learning approach are described in the following sections.

A. Neuron Model

The general equation defining the rate based model to be used is the following [9]:

$$\tau \dot{x} = -x + g_G W^{Res} r + W^{In} S \quad (1)$$

$$z = W^{Out} r \quad (2)$$

$$r = tanh(x) \quad (3)$$

In the above equations τ= 10ms is a time constant, scaling how fast the changes in activity are realized in the network, x is a vector with the current activity of the network, $\dot{x}$ contains the time derivatives of the network, r is the transformed activation vector of the network and z is the activation of the readout neurons. The scaling factor g_G of the connections in the reservoir, defined as $g(P_cN)^{-0.5}$, determines the dynamical characteristics of the reservoir, where g = 1.5 is the gain, P_c= 0.1 is the probability of connection between any two neurons and N= 1600 is the number of neurons in the reservoir. The connectivity matrices W^{Res}, W^{In} and W^{Out} define the weights of the connections between neurons within the reservoir, from inputs to the reservoir neurons and from the reservoir neurons to the readouts respectively. For W^{Res} the weights are drawn

uniformly from the interval [-1,1] with probability P_c, scaled by g_G and left untrained. The inputs are fully connected to the reservoir with weights drawn uniformly from the interval [-1,1] and left untrained. The reservoir is also fully connected to the readouts with weights initially uniformly drawn from [-1,1] and subsequently trained in a supervised manner using FORCE learning. S contains the external input signals, which in the architecture presented here are the activations from the different layers of the multilayer perceptron.

From a biological perspective, the activity of this type of neurons can represent the ensemble average activity of different neuronal populations that are effectively coupled to each other in either an excitatory or inhibitory way. From a mathematical perspective, the activity can be thought of as the displacement from resting state of each of the oscillators in a network of coupled oscillators. As can be seen from (1), every neuron is using the sum of external input signals, a nonlinear transformation of the activities of the neurons that it is connected to, as well as its own state to determine the change in its activity. This mechanism can be utilized to perform information processing on input signals as has been illustrated by [12]. In this work a recurrent neural network was optimized to perform context dependent computation on an input signal where the output depended on the contextual cue provided to the network. It was additionally shown that the network dynamics employed to perform this computation closely resembled the dynamics of recorded prefrontal cortex (PFR) activity from macaque monkeys solving the same task, demonstrating how such networks of artificial neurons can be employed to model and study computations in the brains of mammals.

B. Training with FORCE Learning

FORCE learning is based on the Recursive Least Squares (RLS) update rule [12] and proposed by Abbott and Sussillo [10]. This algorithm appears to be well suited for yielding very stable and accurate pattern generators [14] by utilizing a 2nd-order learning rule that modifies the synaptic weights in a manner that the output error becomes small from the beginning of training. This feature is well suited to this study, since the network presented is stabilized by feeding a trained output back into the network.

When the FORCE learning approach was developed, it was aiming to solve the problem of noisy input or erroneous feedback from the output neuron to the reservoir, as implemented in the Echo State Network (ESN) architecture [15], which was causing instabilities during training from the delayed effects from synaptic modifications through the feedback signal. Previous training methods avoided this problem by training the network in the absence of any such feedback, but with FORCE learning such instabilities can 'be sampled and stabilized' [10]. A second problem solved by FORCE learning was that of credit assignment for the output error. This problem requires the identification of neurons from the reservoir that are most responsible for the error observed between the output and the target function and adjusting the readout weights accordingly. What follows is a brief description of how the FORCE learning algorithm works and how it solves these problems.

FORCE learning relies solely on error-based synaptic modifications of the readout, as well as the reservoir weights if required [11]. It allows the readout signal to be fed back into the network without catastrophic effects, since the main idea behind FORCE learning is that it quickly suppresses the output error by performing appropriate modifications from the beginning of training, which result in significant suppression of the output error early on. Additional modifications are applied to the readout weights following this initial period, in order to completely minimize the error between the target function and the network output. Weight updates for the readout matrix W^{Out} are performed through a variant of the delta rule:

$$W^{Out}(t) = W^{Out}(t - \Delta t) - e(t)\eta(t)r(t) \quad (4)$$

Where $\eta(t)$ is the learning rate and the error $e(t)$ is calculated as the difference of the readout and the target function $f(t)$ according to the equation:

$$e(t) = W^{Out}(t - \Delta t)\, r(t) - f(t) \quad (5)$$

What is particularly interesting about the FORCE algorithm is that instead of a scalar global learning rate $\eta(t)$, the product of the error and the activity of the reservoir $e(t)r(t)$ is multiplied by a neuron specific learning rate in matrix form $P(t)$, which can be understood as a running estimate of the inverse of the correlation matrix of the network rates with a regularization term [13] in the form:

$$P(t) = (r(t)r(t)^T + \alpha I)^{-1} \quad (6)$$

and is calculated in an iterative manner using (8) with an initial value as shown in (9), where $\alpha = 1$, I is the identity matrix and β is defined in (7).

$$\beta := 1/(1 + r(t)^T P(t - \Delta t) r(t)) \quad (7)$$

$$P(t) = P(t - \Delta t) - \beta P(t - \Delta t) r(t) r(t)^T P(t - \Delta t) \quad (8)$$

$$P(0) = \alpha^{-1} I \quad (9)$$

This estimate is used to allocate individual learning rates to neurons in order to ensure that appropriate weight modifications can be applied to effectively reduce the error. The RLS-based update rule in (8) is used to update this estimate iteratively, while avoiding a computationally expensive matrix inversion in the calculation, as described by equation (6). An intuitive explanation of how this algorithm works is that initially all weights have the same learning rate as shown by (9).

This ensures that large modifications of the weights are made early on to reduce the initial error. While learning proceeds, P approaches the inverse of the correlation matrix, which is proportional to the inverse of the eigenvalue matrix as can be extracted by Principal Component Analysis of the network activity. As a result, neurons that have a high contribution to the overall variance of the network activity (and are hence associated with higher eigenvalues) have their learning rate decrease earlier than neurons that contribute less to the variance of network activity. In this way, the neurons with the highest variation in their activities are fixed early on to a suitable value, which keeps the output error small. The rest of the neurons continue to be adjusted with a learning rate that decreases slower to further minimize this error. An additional property of P is that the magnitudes of its elements decrease with increasing number of iterations, resulting in convergence to a solution at sufficiently long training time. These two properties solve simultaneously the problems of credit assignment for the output error as well as the instabilities introduced through feedback of the output. For a more detailed explanation of the algorithm, the interested reader is recommended to read the original work [10].

FORCE learning has been shown to work particularly well in the chaotic regime, which can be achieved by setting the scaling gain g_G of the weight matrix W^{Res} to a value above 1 [9; 10]. It has also been observed that less training cycles are required for networks that exhibit chaotic spontaneous activity with $g_G>1$ compared to inactive networks with $g_G<1$, indicating the practical benefit of such property. In the same study, it was shown that the magnitude of the weights significantly decreased after training if a value of $g_G>1$ was used and this has also been observed in the work presented in this paper. Having smaller magnitudes for the readout weights is usually preferred for many tasks because larger magnitudes are associated with over-fitting to the training data and significant decrease in the generalization ability of the network, as well as its robustness to noise.

C. The Architecture

The episodic memory architecture shown in Figure 1 consists of two parts. The first is a multilayer perceptron (MLP) that has been pre-trained to classify episodes. The episode classification is used as input to the second part of the architecture which is a reservoir used for the storage and recall of temporal information. The reservoir consists of 1600 neurons. Connections between neurons within the reservoir are formed with a probability of 0.1.

The reservoir may be viewed as a dynamical state space with each neuron in the reservoir being one dimension in the state space and the activity of each neuron, which ranges from -1 to 1, being the range of each dimension. In this case, there are 1600 neurons and so 1600 dimensions. The state of the network at one particular moment in time can therefore be viewed as a point in the state space, defined by the activation value of each neuron on its corresponding dimension in the state space. The network performs numerical integration of the neuron update equations 50 times at each time step. Left to its own devices, and because of the recurrent connectivity, the network state will evolve and change over time leading to a succession of points in state space which if connected together in temporal order form a trajectory in state space. The choice of 50 numerical integrations is required so that enough distance exists between adjacent points in the trajectory at adjacent time steps (Euclidean distance between two consecutive points is about 4 for 1600 neurons in this example) so as to facilitate the separation that is required for the output matrices to learn to differentiate and reproduce two distinct output vectors from two distinct network states.

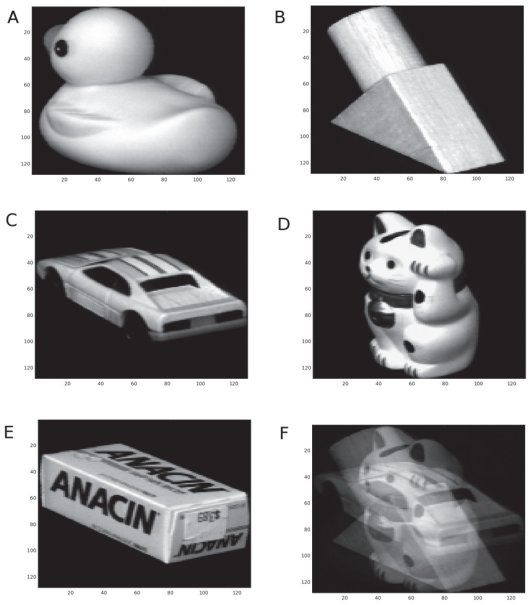

Figure 2. Recall of objects from the COIL Dataset. (A-E) The first five objects in the dataset. (F) Merged images in the reservoir output due to close proximity of the positions of attractors at branch point.

The classification neurons in the MLP are connected to all neurons in the reservoir. When a new episode classification occurs an input pulse representing the classification is sent to the reservoir, such that the signal is input for 20 times steps and therefore a total of 1000 numerical integration steps of the reservoirs neuron update equations. This process causes the reservoir to move to a bounded region in state space that is unique to the particular classification input. Every time an identical classification is sent to the reservoir in this manner, the position in state space is moved to the same region, the Euclidean distance between points being up to 10^{-12} apart. Left to its own devices, and due to the recurrent connectivity, the reservoir will traverse a trajectory from this starting point. As mentioned in the introduction, the reservoir is a complex dynamical system and small differences can escalate in a chaotic manner. In this case, the small difference in the seemingly same starting point on two separate occasions, but with an error of 10^{-12}, grows exponentially until after some

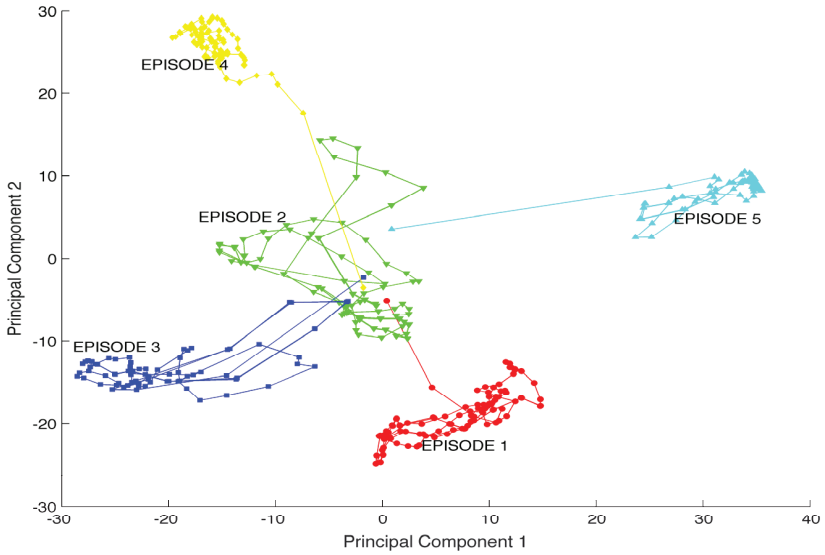

Figure 3. State space of five unique episodes. Each episode is colored differently and contains 65 attractor points.

time the states traversed by the two trajectories are totally different. This is undesirable as we wish the same starting points to produce exactly the same trajectories in order for recall to be the same each time.

In order to stabilize the trajectories a feedback mechanism is used. All reservoir neurons are connected to 40 additional output neurons. The weights connecting to these neurons are trained to reproduce the activations of a mid-level layer in the MLP that also has 40 neurons. By definition this mid-level layer provides an abstraction of the high dimensional data input into the MLP, but not as abstract as the output layer which classifies the entire episode. In short, this mid-level abstraction contains features of the input data. As the input data change over time, the episode classification output may remain the same however the mid-level features change.

During training the mid-level feature abstraction from the MLP, as well as being used to train the output weights mentioned, is also fixed as input into the reservoir. After training and during testing the trained reservoir output to the 40 feedback neurons is used instead of the MLP as input to the reservoir therefore providing feedback. For training, the current mid-level MLP abstraction is used, whereas what is input into the reservoir is the MLP input from the time step before, similar to Echo State Network training [16]. During testing the reservoir output from the time step before is used as input. The system permits the reservoir to learn an abstraction of the current MLP state and then use that abstraction on the next update as input to drive the reservoir to a new position in its state space. On testing, the feedback facilitates a self-perpetuating system that produces output that reconstructs features at time point T that are then used as input to drive the reservoir and feature reconstruction at time point T+1, and so on. The mechanism not only aides stabilization but also gives the reservoir low complexity input that describe the current state in time of the MLP, and therefore historical data regarding the features being presented over time.

The input from the mid-level of the MLP during training and the feedback during testing differ with an error of 10^{-17}. This small error can also cause chaotic effects. Therefore, prior to both learning and feedback, each neuron input is rounded to 3 decimal places, thereby removing all discrepancy between the MLP mid-level abstraction and the reservoir feedback reconstruction signals. In doing so, total stabilization is achieved.

Given a stable system that contains abstract historical information regarding an experience, that is further used to guide a trajectory though state space, full episodic recall can be implemented. In addition to the 40-neuron reconstruction output, the reservoir also has output weights connected to a large number of neurons, equal to the size of input data to the MLP. In this case the input data used is the Columbia University Image Library of 20 objects (COIL-20 from http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php). The images are 128 x 128 grey scale images, giving a total of 16384 inputs. During training the inputs to the MLP at the current time step are used to train the weights of the output of the reservoir to the 16384 neurons. Therefore there are two reservoir outputs. One reconstructs the 40-neuron mid-level abstraction MLP data at time point T, and the other reconstructs the high dimensional input of 16384 neurons at time point T. The 40-neuron feedback mechanism provides a manageable stabilizing input that would not be achievable if the high dimensional 16384 reconstruction was used for feedback. In addition, it provides contextual temporal input regarding features of the episodic experience that can be used for computational purposes.

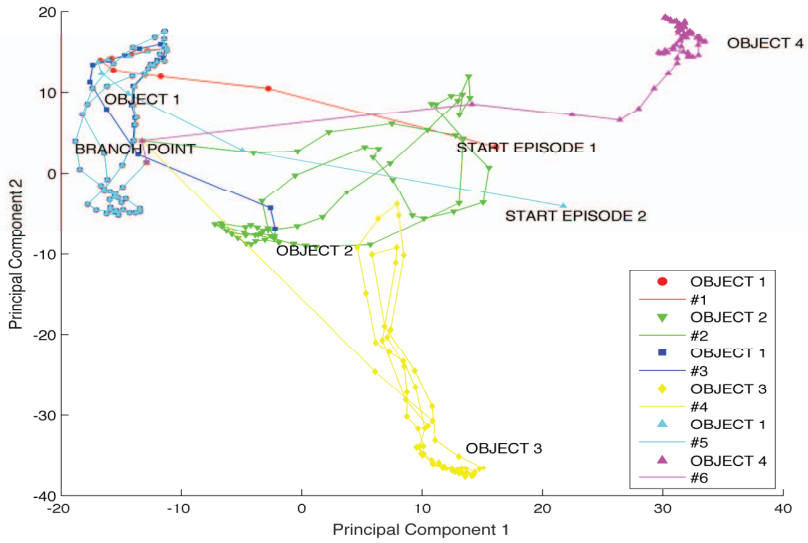

Figure 4. State space of two complex Episodes. Note how the same area of state space is returned to when recalling object 1, and how at the end of showing object 1 there is a point which branches in three directions.

III. RESULTS

The input data to the MLP is the Columbia University Image Library of 20 objects. Each of the 20 objects in the data set is an animation consisting of 64 unique frames with each frame being a 128 x 128 grey scale image. The episodes to recall are sequences of these images. Examples of image reconstruction of these objects can be seen in Figure 2.

In order to illustrate the state space trajectory we performed Principal Component Analysis on the state space of the reservoir and used the two components that explain the highest amount of variance to project the state space trajectories into two dimensions. Figure 3 shows five trajectories of five unique episodes. Each episode was initiated with a unique pulse that caused the reservoir to be positioned in a unique place in the state space. As each trajectory evolved from its start point the input image to the MLP at each time point was associated with the position in the state space at that time by training the reservoir output weights. Each episode contained one object with 64 images of that object being presented twice. Because the data set contains animations of rotating objects, the 64th image precedes the 1st image in animation succession. Therefore presenting 64 images twice produces smooth animations that are twice the length (128 frames).

It is interesting to note that for each trajectory in the dimensionality reduction of Figure 3 there are only 65 state space points. 1 point is the position of the initialisation pulse and first image. The next 63 points are the state space positions of next 63 images belonging to the same object. These locations have been guided to from the previous state space position by the abstract feedback. The next point is the first image again; however the state space position is different to the starting point of the episode as it has also been guided in state space from the previous state space position by the abstract feedback rather than a pulse. The state space positions for the following 63 images appear overlaid on the previous 63 images. This indicates that the feedback mechanism is creating a temporary attractor-like system in which the same feedback signal drives the state space trajectory to almost the same position as that same signal did the previous time it was presented.

To explore this phenomenon further we created more complex episodes. Figure 4 shows the trajectories of 2 such episodes. Each episode contains multiple objects each of which is differently coloured to identify them. Episode 1 starts with a unique pulse and consists of 64 frames of object 1 (red) followed by 64 frames of object 2 (green), followed by 64 frames of object 1 again (blue), and finally followed by 64 frames of object 3 (yellow). Episode 2 starts with a unique pulse and consists of 64 frames of object 1 (cyan), followed by 64 frames of object 4 (magenta). It can be seen from the trajectory of episode 1 that when object 1 is presented on two different occasions during the episode, after a few frames the red and blue trajectories overlay each other. This indicates attractor-like behaviour. However the first presentation of object 1 is followed by object 2, and the second presentation of object 1 is followed by object 3. This behaviour is preserved on recall. On each recall of object 1 the trajectories reach similar attractor-like state space positions for frame 64. However, they branch to different places, one going off to object 2 the other going off to object 3. There is enough difference in the state space caused by differing histories in order to distinguish where it should branch to and when. However, it is interesting to note that at the branch point indicated on the diagram, there is not enough difference

between the positions in state space of each of the branches in order for the output weight matrix to distinguish between them and reconstruct different high dimensional images. As a result of this, the learning process blurs them into one image at that point, until the trajectories separate enough to identify the objects individually again. This blurring can be seen in panel F of Figure 2 and is further indication of the attractor-like mechanism at play. It can be seen from the trajectory of episode 2 that when object 1 (cyan) is presented, after a few frames it overlays the red and blue trajectories which are also attractor-like stable trajectories for object 1. In addition, the same branching mechanism can be seen at play as previously discussed.

The mean absolute error between each pixel of the high dimensional image reconstructions and the 64 training images for each of the 20 objects (1280 unique images in total) is 0.11%. There are 1600 neurons in the reservoir and 16384 pixels in each image, which entails a 1600 x 16364 high dimensional output weight matrix. The minimum capacity for this matrix would be 1600 images each of 16384 pixels. Reducing the size of the reservoir and consequently the size of the output matrix reduces the quality of image reconstruction. However, we have increased the number of unique images well beyond the minimum capacity without reducing image quality. Furthermore, a very high level of image compression is possible due to the fact that natural video follows frame by frame with very similar images, and the ability of the reservoir to keep the state space positions of similar images close together, means that the output weight matrix is not put under any strain. Currently, the limits of this compression are unknown and this leaves an important area for future research.

IV. Discussion

The first point of interest of the mechanism for memory storage and recall in the proposed architecture is that the memory storage and retrieval of an episode depend on three sets of connections; this is of course without taking into account the trained connections in the MLP, where the target function for the feedback is extracted from. The first set includes the untrained connections inside the reservoir, compatible with RC practice [1], which are specified completely by the W^{Res} matrix and determine the magnitude of interaction between connected neurons and hence the dynamic behavior of the network in the absence of any input or feedback. The second set of connections includes the subset of the W^{Out} matrix that is trained to perform a mapping from the trajectory of network activity to the visual image and contributes only in reconstructing the high-dimensional visual input. Using the activity trajectory as the medium of memory storage, as opposed to utilizing attractors of the system, is in agreement with recent theories regarding neural computation [17]. The last set of connections includes all the connections that are part of the feedback loop, these being the subset of connections in W^{Out} that are not used to reconstruct the high-dimensional input and all the connections in W^{In}. The former are trained to reconstruct the abstracted signal from the MLP, while the latter regulate the effect that the external input or feedback signal has on the network activity. Training a subset of the readouts from the network to reconstruct an external signal from the MLP, is equivalent to training the network to drive its own dynamic activity in an identical way to the way that the signal from the MLP would have driven it. In this way the network is able to autonomously reproduce a complete memory-related response by only using a pulse as an initial cue. Since the reproduced trajectory of the network can be used by the readout neurons to reconstruct the image or perform any other relevant mapping, and training the feedback loop is sufficient to autonomously reproduce this trajectory, then we can conclude that training this feedback loop is sufficient and equivalent to storing the memory of a whole coherent episode in the reservoir.

The implemented model works on the basis of serial recall which has been shown to enable the recall of memories at a reasonable level through the storage and reconstruction of order, rather than detailed temporal, information [18; 19]. Even if we do not use explicit timing information (e.g. time labels) our model builds on the ordering of events. Partial memories are assigned a place in time with respect to other memories and the wider context, therefore separating episode storage and recall. The mechanism of associative chaining appears in various theories of memory (for a review, see [20]). The most successful model of serial memory encoding and recall is TODAM [21; 22]. The main constraint of the serial recall approach is the assumed accuracy in the order of recalled items in the episode. If the recall fails mid-sequence, then the chain is broken and recall must necessarily cease. To address chaining failure, the output of TODAM in response to a recall cue is not an exact copy of an item but rather a blurry approximation. To recover the item representation, the noisy output vector must first be de-blurred by determining which of a pool of items it approximates best. If this process is successful, then the de-blurred item is retrieved and used to cue the next response. However, if this process fails, then the associative chain is not necessarily broken, because the blurry output vector can still be used as a retrieval cue, often successfully retrieving the correct next item.

Hippocampal models working in a serial recall manner face similar problems of noise accumulation during mental travelling over time [23]. To overcome this issue, in a more recent version of this model, the authors use a less accurate, hierarchical representation of the environment to stabilize functionality [24]. Both approaches share the same idea with the one used here. In our work, the use of feedback keeping an abstracted representation of the input has been shown to eliminate instabilities, supporting at the same time the emergence of the well-known recency effects in serial memory recall. It is also noted that previous work [23; 24] explored the encoding and recall of rather simple memory items (usually a low dimensional vector), our model deals with sequences of 128 x 128 grey scale images, which is a harder version of the problem, still successfully accomplished.

A variety of models for episodic memory have been proposed [25; 26; 27], ranging from abstract models to models considering neural pathways and structures found in the brain of primates. Norman and O'Reilly [28] developed the Complementary Learning Systems (CLS) framework based on anatomical and psychological studies to shed light on the contributions of neocortex and hippocampus to recognition memory in humans. In our model the MLP and the RC are considered cortical and hippocampal respectively. The CLS hippocampal network architecture of [29] even though not identical to, resembles the proposed architecture presented in Figure 1, mainly in the use of a layered architecture, recurrent connections and a feedback loop from higher levels into lower ones. Moreover, the authors report that after appropriate modifications to the recurrent connections and the feedback loop, the network is able to recall stored memories from an input cue, which is in close agreement with the results of this study where a short pulse into the network is able to initiate episodic memory recall through the feedback loop. This illustrates that even though the actual implementation of the architecture presented in this paper utilizes models and techniques from the machine learning literature, while other episodic memory models come from the field of computational psychology and neuroscience, the functionality of the proposed connectivity results in a biologically plausible mechanism for memory recall. The presented architecture could further be evolved towards more biological plausibility by either adopting an appropriate learning algorithm [30] or by using spiking neurons as in the paradigm of Liquid State Machines [17]. Even though there have been recent advancements in using spiking neurons for spatiotemporal data [31], these biologically more realistic approaches cannot yet offer the versatility of rate based neurons.

AKNOWLEDGMENTS

We would like to thank Radu Cristian Cioata for his assistance in training the MLPs used in this work.

REFERENCES

[1] Hermans, M., Schrauwen, B. (2012). Recurrent Kernel Machines: Computing with Infinite Echo State Networks. *Neural Computation*, 24(1), 104–133.

[2] Lukosevicius, M., Jaeger, H. (2009). Reservoir Computing Approaches to Recurrent Neural Network Training. *Computer Science Review*, 3 (3), 127–149.

[3] Natschlager, T., Maass, W. (2004). Information dynamics and emergent computation in recurrent circuits of spiking neurons. NIPS 2003 proceedings, 1255–1262.

[4] Büsing, L., Schrauwen, B., Legenstein R. (2010). Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural Computation*, 22 (5), 1272-311.

[5] Sederberg, P., Howard, M., Kahana, M. (2008). A context-based theory of recency and contiguity in free recall. *Psychol. Rev*. 115 (4), 893-912.

[6] Jakimovski, P., Schmidtke, H. R. (2011). Delayed Synapses: An LSM Model for Studying Aspects of Temporal Context in Memory. *Modeling and Using Context*, Springer, Berlin, pp. 138–144.

[7] Lukosevicius, M. (2012). A Practical Guide to Applying Echo State Networks. *Lecture Notes in Computer Science*, Vol 7700, 659-686.

[8] Jarvis, S., Rotter S., Egert U. (2011). Increased robustness and intermittent dynamicsin structured Reservoir Networks with feedback. *ESANN 2011* proceedings, 111-116.

[9] Sompolinsky, H., Crisanti, A., Sommers, H. J. (1988). Chaos in random neural networks. *Physical Review Letters*, *61*(3), 259–262.

[10] Sussillo, D., Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, *63*(4), 544–57.

[11] Laje, R., Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature Neuroscience*, *16*(7), 925–33.

[12] Mante, V., Sussillo, D., Shenoy, K. V., Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*(7474), 78–84.

[13] Haykin, S. (2014). The Recursive Least Squares Algorithm. *Adaptive Filter Theory,* PEARSON (Fifth Ed, pp. 449–472).

[14] Lukoševičius, M., Jaeger, H., Schrauwen, B. (2012). Reservoir Computing Trends. *Künstliche Intelligenz*, *26*(4), 365–371.

[15] Jaeger, H., Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science (New York, N.Y.)*, *304*(5667), 78–80.

[16] Jaeger, H. (2010). The " echo state " approach to analysing and training recurrent neural networks – with an Erratum note 1. *GMD Report*, (148).

[17] Maass, W., Natschläger, T., Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560.

[18] Neath, I., Suprenant, A. M. (2003). Human memory: An introduction to research, data and theory, (2nd ed.), Belmont, CA: Wadsworth.

[19] Fradera, A. J. (2006). Memory for time: The use of temporal codes versus contextual information. Doctoral thesis, University of London.

[20] Hurlstone, M., Hitch, G., Baddeley, A. (2013). Memory for Serial Order Across Domains: An Overview ofthe Literature and Directions for Future Research. Psychological Bulletin, Vol 140(2), 339-373.

[21] Lewandowsky, S., Murdock, B. B. (1989). Memory for serial order. *Psychological Review*, 96, 25–57.

[22] Murdock, B. B. (1995). Developing TODAM: Three models for serial order information. *Memory & Cognition*, 23, 631–645.

[23] Erdem, U. M., Hasselmo, M. E. (2012). A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *The European Journal of Neuroscience* 35 (6), 916–931.

[24] Erdem, U. M., Hasselmo, M. E. (2014). A biologically inspired hierarchical goal directed navigation model, *Journal of Physiology – Paris,* 108, 28–37.

[25] Busemeyer, J. R., Wang Z., Eidels, A. J. T. T. (2015). *The Oxford Handbook of Computational and Mathematical Psychology,* Oxford University Press.

[26] Norman, K., Detre, G., Polyn, S. (2008). Computational models of episodic memory. *The Cambridge Handbook of Computational Psychology*, Cambridge University Press, 189–224.

[27] Sato, N., Yamaguchi, Y. (2010). Simulation of Human Episodic Memory by Using a Computational Model of the Hippocampus. *Advances in Artificial Intelligence*, 1–10.

[28] Norman, K. A., O'Reilly, R. C. (2003). Modeling hippocampal and neocortical contributions to recognition memory: a complementary-learning-systems approach. Psychological Review, 110(4), 611–646.

[29] Norman, K. A. (2010). How hippocampus and cortex contribute to recognition memory: revisiting the complementary learning systems model. *Hippocampus*, *20*(11), 1217–27.

[30] Yusoff, M. H., Jin, Y. (2014). Modeling neural plasticity in echo state networks for time series prediction. 14th UK Workshop on Computational Intelligence (UKCI), 1–7.

[31] Kasabov, N., Scott, N. M., Tu, E., Marks, S., Sengupta, N., Capecci, E., Yang, J. (2015). Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications. *Neural Networks*: Special Issue on Learning in Big Data.