IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 015053 https://doi.org/10.1088/2632-2153/ad2feb

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
26 May 2023

REVISED
9 October 2023

ACCEPTED FOR PUBLICATION
4 March 2024

PUBLISHED
21 March 2024

PAPER

The impact of memory on learning sequence-to-sequence tasks

Alireza Seif[1], **Sarah A M Loos**[2,5], **Gennaro Tucci**[3], **Édgar Roldán**[2] and **Sebastian Goldt**[4,*]

[1] Pritzker School of Molecular Engineering, University of Chicago, Chicago, IL 60637, United States of America
[2] ICTP—The Abdus Salam International Centre for Theoretical Physics, Trieste, Italy
[3] Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany
[4] International School of Advanced Studies (SISSA), Trieste, Italy
[5] DAMTP, Centre for Mathematical Sciences, University of Cambridge, Cambridge CB3 0WA, United Kingdom
[*] Author to whom any correspondence should be addressed.

E-mail: sgoldt@sissa.it, seif@uchicago.edu and sl2127@cam.ac.uk

Keywords: sequence-to-sequence task, auto-regressive model, recurrent neural network, memory, non-Markovianity, statistical physics

Abstract

The recent success of neural networks in natural language processing has drawn renewed attention to learning sequence-to-sequence (seq2seq) tasks. While there exists a rich literature that studies classification and regression tasks using solvable models of neural networks, seq2seq tasks have not yet been studied from this perspective. Here, we propose a simple model for a seq2seq task that has the advantage of providing explicit control over the degree of memory, or non-Markovianity, in the sequences—the stochastic switching-Ornstein–Uhlenbeck (SSOU) model. We introduce a measure of non-Markovianity to quantify the amount of memory in the sequences. For a minimal auto-regressive (AR) learning model trained on this task, we identify two learning regimes corresponding to distinct phases in the stationary state of the SSOU process. These phases emerge from the interplay between two different time scales that govern the sequence statistics. Moreover, we observe that while increasing the integration window of the AR model always improves performance, albeit with diminishing returns, increasing the non-Markovianity of the input sequences can improve or degrade its performance. Finally, we perform experiments with recurrent and convolutional neural networks that show that our observations carry over to more complicated neural network architectures.

1. Introduction

The recent success of neural networks on problems in natural language processing [1–5] has rekindled interest in tasks that require transforming a sequence of inputs into another sequence (seq2seq). These problems also appear in many branches of science, often in the form of time series analysis [6, 7]. Despite their ubiquity in science, there has been little work analysing learning of seq2seq tasks from a theoretical point of view in simple toy models. Meanwhile, a large body of work emanating from the statistical physics community [8–12] has studied the performance of simple neural networks on toy problems to develop a theory of supervised learning, where a high-dimensional input such as an image is mapped into a low-dimensional label, like its class.

An important recent insight from the study of supervised learning was the importance of data structure for the success of learning. The challenge of explaining the success of neural networks on computer vision tasks [13–16] inspired a new generation of data models that take the effective low-dimensionality of images [17] into account, such as object manifolds [18], the hidden manifold [19, 20], spiked covariates [21, 22], or low-dimensional mixture models embedded in high dimensions [23–25]. By deriving learning curves for neural networks on these and other data models [26–29], the importance of data structure for the success of neural networks compared to other machine learning methods was clarified [21, 23, 24, 30].

The aforementioned breakthroughs on seq2seq tasks make it an important challenge to extend the approach of studying learning in toy models to cover the realm seq2seq tasks. What are key properties of sequences, akin to the low intrinsic dimension of images, that need to be modelled? How do these properties interact with different machine learning methods like auto-regressive (AR) models and neural networks?

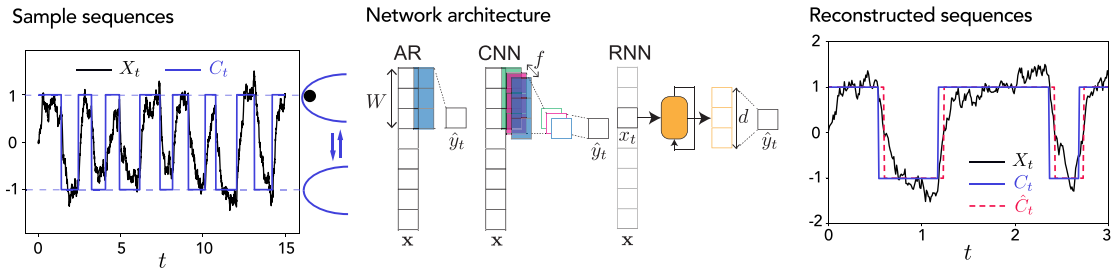

Figure 1. A flexible, minimal model for sequence-to-sequence learning tasks with varying degrees of memory along the sequence. Left: the motion of a Brownian particle (black filled circle) in a switching parabolic potential 'trap' yields dynamics as given by the SSOU (stochastic switching Ornstein Uhlenbeck) model given by equation (1). Example trajectories for sequences of the particle's position X_t (black line) and the trap centre C_t (blue line) as shown as a function of time t. The blue dashed line is set at the trap centres $C_0 = \pm 1$. Middle: we train three types of models to reconstruct the trap positions C_t from the particle trajectories X_t: auto-regressive models (AR) that makes predictions of the trap position based on the past W observations of the particles position, as well as 1D convolutional neural networks (CNN) that acts as a set of f parallel AR models with added nonlinearity in the output, and recurrent neural networks (RNN) that use feedback loops with a d dimensional internal state to capture dependencies across time steps. Right: sample reconstruction of the particle trap positions $\hat{y}_t$ (red dashed line) for one example input sequence (X_t, black solid line) compared with the actual hidden trap position (C_t, blue solid line). The sequence is a zoomed-in view of the sample sequence in the left panel. *Parameters*: $\kappa = 10$, $k = 1$, $D = 0.5$, simulation time step $\Delta t = 0.02$, AR model window size $W = 2$. For this example, we used 5000 training samples, evaluated them over 5000 test samples, a mini-batch size 32 and 5 epochs.

Here, we make a first step in this direction by proposing to use a minimal, solvable latent variable model for time-series data that we call the stochastic switching Ornstein–Uhlenbeck process (SSOU). The input data consists of a sequence $\mathbf{x} = (x_t)_{t=0}^{T}$, which depends on a latent, unobserved stochastic process $\mathbf{c} = (c_t)_{t=0}^{T}$. The learning task is to reconstruct the unobserved sequence $\mathbf{c}$ from the input sequence $\mathbf{x}$, cf figure 1. The key data property we aim to describe and control is the *memory* within the input sequence $\mathbf{x}$. In the simplest scenario, the sequence x_t is memoryless: given the value of the present token x_t, the value of the next token x_{t+1} is statistically dependent solely on x_t but not on previous tokens $x_{t'<t}$. Such a sequence can be described by a Markov process. By tuning the dynamics of the latent process c_t, we can control the memory of the sequence, i.e. we can increase the statistical dependence of future tokens on their full history $x_{t'<t}$ (we introduce a precise, quantitative measure for the memory below). Adding memory thus makes the process non-Markovian and allows us to model richer statistical dependencies between tokens. At the same time, the presence of memory in a process makes the mathematical analysis generally harder. Our goal is to analyse how different neural network architectures – AR, convolutional (CNN), and recurrent (RNN) – handle the memory when solving the seq2seq task $\mathbf{x} \to \mathbf{c}$.
Our **main contributions** can be summarised as follows:

1. We introduce the SSOU process as a latent variable model for seq2seq tasks (section 2)
2. We introduce a measure of non-Markovianity that quantifies the memory of the sequences, and we describe how to tune the memory of the input and target sequences (section 4.1)
3. We use analytical [31] and numerical evidence to identify two regimes in the performance of the AR model in the Markovian and non-Markovian case, respectively. These regimes emerge from the interplay of two different time scales that govern the sequence statistics (section 4.2)
4. We show that the task difficulty is non-monotonic in the sequence memory for all three architectures: the task is easy with strong memory or no memory, and hardest when there is a weak amount of memory. We explain this effect using the sequence statistics (section 4.3)
5. We finally find that increasing the integration window of AR models and the kernel size of convolutional neural network (CNN) improves their performance, while increasing the dimension of the hidden state of gated RNN achieves only minimal improvements (section 4.3)

Reproducibility. We provide code to sample from the SSOU model and reproduce our experiments at https://github.com/anon/nonmarkovian_learning.

2. A model for sequence-to-sequence tasks

We first describe a latent variable model for seq2seq tasks, which we call the SSOU model. This model was introduced recently in biophysics to describe experimental recordings of the spontaneous oscillations of the tip of hair-cell bundles in the ear of the bullfrog [31]. Furthermore, a simplified variant of the model with

exponential waiting times has also been fruitfully explored in the experimental context to describe the relaxation oscillations of colloidal particles in switching optical traps [32, 33].

We consider observable sequences $\mathbf{X} = (X_t)$ which are described by a one-dimensional stochastic process whose dynamics is driven by an autonomous latent stochastic process $\mathbf{C} = (C_t)$ and a Gaussian white noise that is independent to C_t. Here and in the following we index sequences by a time variable $t \geqslant 0$, and use bold letters such as $\mathbf{X}$ to denote sequences, or trajectories. A sequence of length T can be sampled from the stochastic differential equation

$$X_{t+\mathrm{d}t} - X_t = -\kappa\,(X_t - C_t)\,\mathrm{d}t + \sqrt{2D}\mathrm{d}B_t, \tag{1}$$

with $t \in [0, T-1]$. Here, $\mathrm{d}B_t$ is the increment of the Wiener process in $[t, t+\mathrm{d}t]$, with $\langle \mathrm{d}B_t \rangle = 0$ and $\langle (\mathrm{d}B_t)(\mathrm{d}B_{t'}) \rangle = \delta(t-t')\mathrm{d}t$. The angled brackets $\langle\cdot\rangle$ indicate an average over the noise process. We denote the parameters $D > 0$ and $\kappa > 0$ as diffusion coefficient and trap stiffness for reasons described below.

We employ this setup as a seq2seq learning task where we aim to reconstruct the hidden sequence of trap positions $\mathbf{C}$ given a sequence of particle positions $\mathbf{X}$, see figure 1 for an illustration. The key idea in our model is to let the location of the potential C_t alternate in a stochastic manner between the two positions $C_0 = \{-1, 1\}$. The waiting time τ spent in each of these two positions is drawn from the waiting-time distribution $\psi_k(\tau)$. For a generic choice of $\psi_k(\tau)$, the process C_t—and hence X_t—is non-Markovian and has a memory. Here, we use a one-parameter family of gamma distributions (defined in equation (6) below), which allows us to quantitatively control the degree of memory in the sequence of tokens $\mathbf{C}$ in a simple manner; we discuss this in detail in section 4.1.

2.1. Physical interpretation of the SSOU model

A well-known application of equation (1) in statistical physics is to describe the trajectories of a small particle (e.g. a colloid) in an aqueous solution undergoing Brownian motion [34] in a parabolic potential with time-dependent centre C_t. Such physical model is often realised with microscopic particles immersed in water and trapped with optical tweezers [32, 35, 36]. When $\kappa = 0$, the particle is driven only by the noise term $\mathrm{d}W_t$ and performs a one-dimensional free diffusion along the real line with diffusion coefficient D. By choosing κ positive, the particle experiences a restoring force $-\kappa(X_t - C_t)$ towards the instantaneous centre of the potential. Such force $F(X_t, C_t) = -\partial_X V(X, C)|_{X=X_t,\, C=C_t}$ tends to confine the particle to the vicinity of the point C_t, and is therefore often called a particle trap, which is modelled by a harmonic potential centred on C_t

$$V(X_t, C_t) = \frac{\kappa}{2}(X_t - C_t)^2. \tag{2}$$

This motivates the name 'stiffness' for κ; the higher κ, the stronger the restoring force that confines the particle to the origin of the trap C_t. From a physical point of view, the dynamics of X_t consists of the alternate relaxation towards the two minima of the potential between consecutive switches, cf figure 1.

3. Architectures and training

Auto-regressive model. We first consider the arguably simplest machine-learning model that can be trained on sequential data, an AR model of order W, AR(W), see figure 1. The output $\hat{\mathbf{y}} = (\hat{y}_t)$ of the model for the trap position given the sequence $\mathbf{x}$ is given by

$$\hat{y}_t = \sigma\left(\sum_{\tau=1}^{W} w_\tau x_{t-\tau+1} + b\right) \tag{3}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoidal activation function, and the weights w_τ and the bias b are trained. Its basic structure—one layer of weights followed by a non-linear activation function—makes the AR model the seq2seq analogue of the famous perceptron that has been the object of a large literature in the theory of neural networks focused on classification [8–10, 12]. Note that the window size $W \geqslant 1$ governs the number of tokens accessible to the model. We do not pad the input sequence, so the output sequence is shorter than the input sequence by $W - 1$ steps.

Convolutional two-layer network. The AR model can also be thought of as a single layer 1D CNN [37, 38] with a single filter and a kernel of size W with sigmoid activation. We also consider the natural extension of this model, CNN(W), a 1D CNN with two layers (see figure 1). The first layer has the same kernel size (W), but contains f filters with rectified linear activation [39]. In the second layer, we apply another convolution with a single filter and a kernel size of 1 with sigmoid activation. In this way, we can compare a CNN and an

AR model whose 'integration window' are of the same length W. This allows us to investigate the effect of additional nonlinearities of the second layer of the CNN on the prediction error.

Recurrent neural network. We also apply a recurrent network to this task that takes x_t as the input at step t, followed by a layer of gated recurrent units (GRUs) [40] with a d-dimensional hidden state, followed by a fully connected layer with a single output and the sigmoid activation function (see figure 1). We refer to this family of models by GRU(d).

Generating the data set. We train all the models on a training set with N sequences $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$, which can in principle be of different lengths. To obtain a sequence, we first generate a trajectory $\mathbf{C}$ for the latent variable by choosing an initial condition at random from $C_0 \in \{-1, 1\}$ and then drawing a sequence of waiting times from the distribution $\psi_k(\tau)$. Using the sampled trap positions, $\mathbf{C}$, we then sample the trajectory $\mathbf{X}$ from equation (1). Since in practice we do not have access to the full trajectory, as any experiment has a finite sampling rate, we subsample the process X_t by taking every sth element of the sequence. That is, for a given sequence $\{X'_t\}_{t'=0}^{T'}$ of length T', we construct a new sequence $\mathbf{x} = (x_t)_{t=1}^{T}$, where $x_t = X_{s\times t}$ and $T = \lfloor T'/s \rfloor$. The subsampled sequence $\mathbf{x}$ is then used as an input sequence. We verified that the finite time step for the integration of the SDE and the subsampling preserve the statistics of the continuum description that we use to derive our theoretical results, cf appendix B.2.2. For convenience, we also introduce $\mathbf{y} = (1 + \mathbf{c})/2$ to shift the trap position to 0 or 1. We then use this subsampled and shifted sequence of trap positions as the target.

Training and evaluation. We train the models by minimising the mean squared loss function using Adam [41] with standard hyperparameters. For each sample sequence, the loss is defined as

$$\ell(\mathbf{y}_t, \hat{\mathbf{y}}_t) = \frac{1}{T - \tau_0} \sum_{\tau=\tau_0}^{T} (y_\tau - \hat{y}_\tau)^2, \tag{4}$$

where the offset τ_0 in the lower limit of the sum is necessary for the AR and CNN models since their output has a different length than the input sequence. For those models we choose $\tau_0 = W - 1$. For RNN models, however, we use $\tau_0 = 1$ as the input and output sequence lengths are the same. We assess the performance of the trained models by first thresholding their outputs $\hat{y}_t$ at 0.5 to obtain the sequence $\hat{c}_t \in \{\pm 1\}$, since the location of the centre of the potential has two possible values ± 1. We then use the misclassification error ϵ defined as

$$\epsilon = \frac{1}{N_{\text{samples}} N_\tau} \sum_{\text{samples}, \tau > \tau_h} \delta(c_\tau \neq \hat{c}_\tau), \tag{5}$$

where N_τ is the length of the test sequence and $\delta(x) = 1$ if the condition x is true and it is 0 otherwise, as the figure of merit throughout this work. Unless otherwise specified, we used 50 000 training samples with mini-batch size 32 to train the models, and we evaluated them over 10 000 test samples. To consistently compare different models with different output lengths and to reduce the boundary effects we only consider the predictions after an initial time offset τ_h [42].

4. Results

4.1. The waiting time distribution of the trap controls the memory of the sequence

The key data property we would like to control is the memory of the sequence $\mathbf{X}$. In our SSOU model, the memory is controlled by tuning the memory in the latent sequence $\mathbf{C}$, which in turn depends on the distribution of the waiting time τ. To fix ideas, we choose gamma-distributed waiting times,

$$\psi_k(\tau) = \frac{1}{\tau} \frac{(\tau k)^k e^{-k\tau}}{\Gamma(k)}, \tag{6}$$

where we set $k \geqslant 1$, while $\Gamma(k) = \int_0^\infty \mathrm{d}x x^{k-1} e^{-x}$ denotes the Gamma function. Note that for any choice of k, the normalisation condition is $\int_0^\infty \psi_k(\tau)\mathrm{d}\tau = 1$ and the average waiting time between two consecutive switches is $\langle\tau\rangle_k = \int_0^\infty \tau\psi_k(\tau)\,\mathrm{d}\tau = 1$. We can control the shape of the distribution by changing the value of k: the variance of the waiting time for example is k-dependent, $\langle\tau^2\rangle_k - \langle\tau\rangle_k^2 = 1/k$ (cf figure 2). Furthermore, k also controls the 'degree of non-Markovianity' of the latent sequence $\mathbf{C}$ as follows.

For the choice $k = 1$, the waiting-time distribution is exponential, making the process C_t Markovian. This result, together with the fact that equation (1) is linear, implies that the observable process X_t is Markovian, too. Thus the probability distribution for the tokens x_t obeys the Markov identity [34]

$$p_{1|2}(x_{t_3}|x_{t_2}; x_{t_1}) = p_{1|1}(x_{t_3}|x_{t_2}), \tag{7}$$

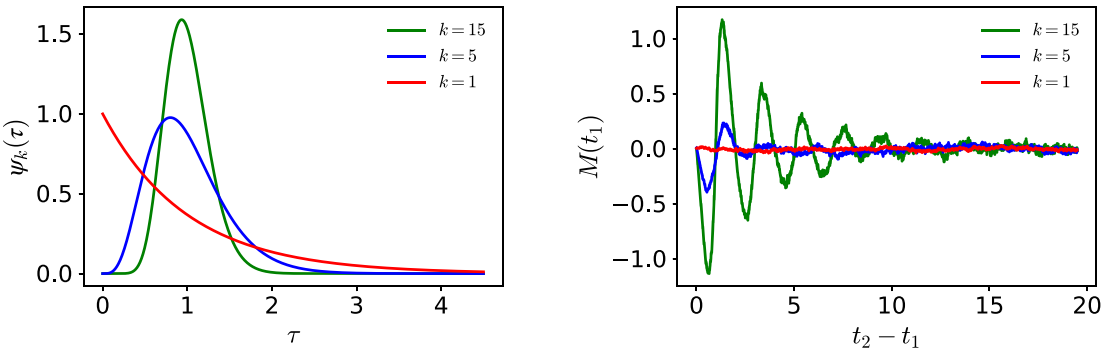

Figure 2. Quantifying the memory of non-Markovian input sequences. Left: waiting-time distribution $\psi_k(\tau)$ given by equation (6) for three choices of k. Right: measure $M(t)$ (see equation (8)) associated with the parameters in the left panel, which quantifies the memory of the past time t_1 in the sequence. We obtain a memoryless sequence X_t with $M(t)=0$ by choosing $k=1$ which creates an exponential waiting-time distribution, equation (6) (red curves). As the value of k increases, the memory becomes stronger (blue and green curves). *Parameters*: $D=0.5$, $\kappa=2$, $\Omega_1=\Omega_2=[0.5,1.5]$, $t_3-t_2=0.5$, by generating $N=10^5$ trajectories via the Euler–Maruyama method [43] with simulation time step $\Delta t=5\times10^{-3}$.

for any $t_1<t_2<t_3$, which represent different instants in time, and $p_{1|m}$ denotes the conditional probability density (at one time instant) with m conditions (at m previous time instants). Equation (7) is the mathematical expression of the intuitive argument that we gave earlier: in a Markovian process, the future state x_{t_3} at some time t_3 given the state x_2 at some earlier time $t_2<t_3$ is conditionally independent of the previous tokens x_1 for all times $t_1<t_2$.

However, most of the sequences analysed in machine learning are non-Markovian: in written language for example, the next word does not just depend on the previous word, but instead on a large number of preceding words. We can generate non-Markovian sequences by choosing $k>1$. To systematically investigate the impact of sequence memory on the learning task, it is crucial to quantify the *degree* of non-Markovianity of the sequence for a given value of k beyond the binary distinction between Markovian and non-Markovian. Yet defining a practical measure that quantifies conclusively the degree of non-Markovianity is a non-trivial task [44, 45] and subject of ongoing research mainly done in the field of open quantum systems [46–50].

Here, we introduce a simple quantitative measure of the degree of non-Markovianity of the input sequence motivated directly by the defining property of Markovianity given in equation (7), which reads

$$\begin{aligned} M(t_1,t_2,t_3) &\equiv \frac{\langle x_{t_3}|\, x_{t_2}\in\Omega_2, x_{t_1}\in\Omega_1\rangle}{\langle x_{t_3}|\, x_{t_2}\in\Omega_2\rangle}-1 \\ &= \frac{\int_{\mathcal{X}} x_{t_3}\, p_{1|2}\left(x_{t_3}|\, x_{t_2}\in\Omega_2, x_{t_1}\in\Omega_1\right)\mathrm{d}x_{t_3}}{\int_{\mathcal{X}} x_{t_3}\, p_{1|1}\left(x_{t_3}|\, x_{t_2}\in\Omega_2\right)\mathrm{d}x_{t_3}}-1. \end{aligned} \tag{8}$$

Equation (8) involves crucially conditional expectations[6], where the expectation of the future system state (at time t_3) is conditioned on the present state (at time t_2), or on the present and past state (at times t_2 and t_1). We have further introduced the notion of state space regions $\Omega_1\subseteq\mathcal{X}$ and $\Omega_2\subseteq\mathcal{X}$ which are subsets of the entire state space $\mathcal{X}=\mathbb{R}$ that is accessible by the process $\mathbf{X}$. The measure M defined in equation (8) quantifies the drop in uncertainty about the future mean values, when knowledge about the past (x_{t_1}) is given in addition to the knowledge of the present state. Because of noise, this generally depends on how far from the past the additional data point is, t_2-t_1, where the decay of M with t_2-t_1 quantifies the 'memory decay' with increasing elapsed time. From equation (7) it directly follows that for any Markovian process, the measure M defined in equation (8) vanishes at all times, whereas $M\neq0$ reveals the presence of non-Markovianity in the form of memory of the past time t_1. In a stationary process, M generally depends on the two time differences t_3-t_2 and t_2-t_1, and on the choices of Ω_1 and Ω_2. To spot the non-Markovianity, t_3-t_2 should be comparable to (or smaller than) the dynamical time scales of the process, which can be defined by the decay of correlation functions, which we show in figure 6 in the appendix (namely, if t_3-t_2 is so large that x_{t_3} and x_{t_2} are fully uncorrelated, M trivially vanishes even for non-Markovian processes). The choice of Ω_1 and Ω_2 is in principle arbitrary, but, for practical purposes, they should correspond to regions in the state space that are frequently visited.

[6] We denote $\langle X|Y\in\Omega\rangle$ the conditional expectation of the random variable X given that the random variable Y satisfies a certain criterion, symbolyzed here as belonging to the set Ω. Note that in general X and Y are statistically dependent, and that the conditioning may be done with respect to more than one random variables satisfying prescribed criteria.

We plot M in figure 2 for three different values of k obtained from numerical simulations of the SSOU. Here, we fix $t_{2,3}, \Omega_{1,2}$, and vary t_1. As expected, for a Markovian switching process with $k=1$, $M(t_1)$ vanishes at all times t_1, while for $k>1$, $M(t_1)$ displays non-zero values. The non-Markovianity measure $M(t_1)$ defined in equation (8) generally captures different facets of non-Markovianity. On the one hand, the decay with $t_2 - t_1$ measures how far the memory reaches into the past. On the other hand, the magnitude of $M(t_1)$ tells us how much the predictability of the future given the present state profits from additionally knowing the past state at time t_1. For the SSOU model, we observe in figure 2 that increasing k increases both the decay time and the amplitude of M, showing that that the parameter k controls conclusively the degree of non-Markovianity. We further note that $M(t_1)$ displays oscillations for $k>1$, reflecting the oscillatory behaviour of $\mathbf{C}$ and $\mathbf{X}$, and that $M(t_1)$ always decays to zero for sufficiently large values of $t_2 - t_1$, indicating the finite persistence time of the memory in the system. We note that these observations are robust with respect to the details of the non-Markovianity measure. For example, we numerically confirmed that when we change $t_3 - t_2$, or $\Omega_{1,2}$, the essential features of M and its dependence on k remain unchanged.

In conclusion of this analysis, we can in the following simply use k as control parameter of the degree of non-Markovianity of $\mathbf{X}$.

4.2. The performance of auto-regressive models

4.2.1. The interplay between two time scales determines prediction error

To gain some intuition, we first consider the simplest possible students, namely auto-regressive models AR(2) from equation (3) with window size $W=2$. The student predicts the value of c_t only using information about the present and the previous tokens x_t and x_{t-1}, giving it access to the current particle position and allowing it in principle also to estimate the velocity of the particle. We show the performance of this model obtained numerically in figure 3. The error of AR(2) varies significantly from less than 5% to essentially random guessing at 50% error as we vary the two time scales that influence the sequence statistics,

$$t_\kappa = 1/\kappa \qquad \text{and} \qquad t_{\text{diff}} = C_0^2/2D = 1/2D \tag{9}$$

the relaxation time and the diffusive time scale. The **relaxation time** t_κ determines how quickly the average position of the particle is restored to the centre of the trap when the the average waiting time $\langle\tau\rangle_k = 1$. A faster relaxation time means that the particle responds more quickly to a change in c_t, making reconstruction easier. The **diffusive time scale** t_{diff} sets the typical time that the system elapses to randomly diffuse a distance $\sim|C_0|$. The larger this time scale, the slower the particle moves randomly, as opposed to movement that follows the particle trap, hence improving the error.

4.2.2. A 'phase diagram' for learnability

For memoryless sequences ($k=1$), we can explain this performance more quantitatively by solving for the data distribution $p(x_t)$. Interestingly, we observe that along the 'phase boundary' between unimodal and bimodal particle distribution $p(x_t)$, the error is roughly homogeneous and approximately equal to 25%. The presence of two 'phases' in terms of the bimodality/unimodality of the distribution $p(x_t)$ is rationalised using recent analytical results for the loci of the phase boundary in the the case $k=1$ [31], which is given by

$$\frac{t_{\text{diff}}}{t_\kappa} = \frac{(t_\kappa + 1/2)\ {}_1F_1\left(1/2, t_\kappa + 1/2, -t_{\text{diff}}/t_\kappa\right)}{{}_1F_1\left(3/2, t_\kappa + 3/2, -t_{\text{diff}}/t_\kappa\right)}. \tag{10}$$

This analytical result is obtained by finding the parameter values at which the derivative of $p(x_t)$ at $x_t = 0$ changes sign, which corresponds to a transition between unimodal and bimodal (see appendix A.1 for a detailed derivation). The line separating the two phases is drawn in black in the density plot of figure 3. In particular, we observe that reconstructing the trap position is simplified when the marginal distribution of the particle distribution $p(x_t)$ is bimodal, i.e. when it presents two well defined peaks around the trap centres $\pm C_0$ (see figure 3 top left). On the contrary, when the distribution $p(x_t)$ is unimodal (corresponding to the case of fast relaxation times t_κ) the learning performance worsens as the data is not sufficiently informative about the latent state $\mathbf{C}$.

If we add memory to the sequence by choosing $k>1$, an analytical solution for the phase boundary is challenging, as it requires knowledge of the steady-state distribution of the joint system. For $k=1$, the system is Markovian and we can solve the Fokker–Planck equation for the steady-state distribution. For $k\neq 1$, the steady-state distribution is the solution of an integro-differential equation which is not solvable analytically. However, we can verify whether the same mechanism—a transition from a unimodal to a bimodal distribution for the particle distribution $p(x_t)$—drives the error also in the non-Markovian case by means of numerical simulations. Interestingly, when increasing the degree of memory to $k=5$ (top right of figure 3), we find that the AR(2) model yields a larger error than for $k=1$ Markovian sequences for all the parameter

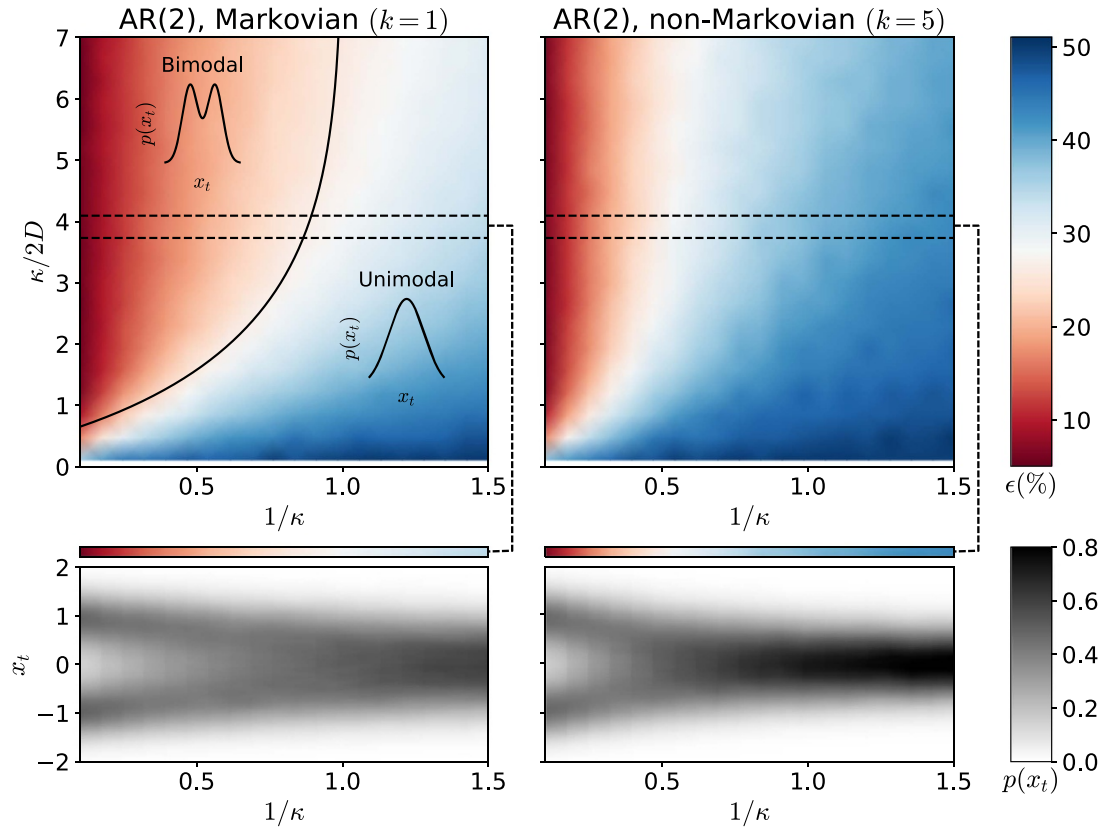

Figure 3. Performance of auto-regressive AR(2) models for Markovian and non-Markovian sequences of the SSOU model across its 'learnability phase diagram'. Reconstruction error of the AR(2) model (in % of correctly predicted trap positions) for different values of the diffusive time scale $t_{\text{diff}} = 1/2D$ and the relaxation time $t_\kappa = 1/\kappa$, see equation (9). We show these 'phase diagrams' for (Left) Markovian sequences ($k = 1$) and for (Right) non-Markovian sequences ($k = 5$). In these diagrams, the two axes represent ratios of time scales, t_{diff}/t_κ (vertical axis) vs $t_\kappa/\langle\tau\rangle_k = 1/\kappa$ (horizontal axis), since $\langle\tau\rangle_k = 1$ for all k. The bottom row depicts the distribution of particle's position $p(x_t)$ across the dashed line superimposed with the error heat map. The density shows a clear transition from a bimodal distribution to a unimodal one. *Parameters:* total simulation time for each parameter value $\tau = 30$, simulation time step $\Delta t = 0.01$, AR(W) window size $W = 2$, κ varying from 0.1 to 0.6, $\tau_h = 2$, remaining training parameters as in figure 1.

values explored in the learnability phase diagram. This means that not only that AR(2) is not able to extract all the available information from the sequence, but that the task has become harder by increasing k. A look at the density plots for the particle distribution $p(x_t)$ explains this finding: the transition to unimodality happens for a smaller value of the relaxation time t_κ in the non-Markovian case (bottom right of figure 3). We go into detail on how the degree of non-Markovianity makes the task harder by quantifying this transition in the probability distribution in the next section.

4.2.2.1. Further statistical characterisation of the error

We can quantify the degree of bimodality of the particle distribution $p(x_t)$, which drives the error of the AR model, by using the Sarle coefficient [51],

$$\text{Sa} = \left(\sigma^2 + 1\right)/\mathcal{K} \tag{11}$$

with σ and $\mathcal{K}$ the skewness and kurtosis of x_t, respectively. The Sarle coefficient takes values from 0 to 1 and is equal to 5/9 for the uniform distribution. Higher values may indicate bimodality. We can see a clear correlation between the Sarle coefficient and the error (ϵ) of the AR(2) model from the scatter plot on the right of figure 3. Indeed, there appears to exist an upper bound on the error in terms for the Sarle coefficient. Another measure we can compute is the excess kurtosis, which for a zero-mean random variable reads

$$\mathcal{K}_{\text{ex}} = \frac{\langle x^4\rangle}{\left(\langle x^2\rangle\right)^2} - 3, \tag{12}$$

where the average is over $p(x_t)$ and we have used the fact that $\langle x_t\rangle = 0$ by symmetry. The excess kurtosis vanishes for a Gaussian distribution and is positive or negative for non-Gaussian distributions, depending on

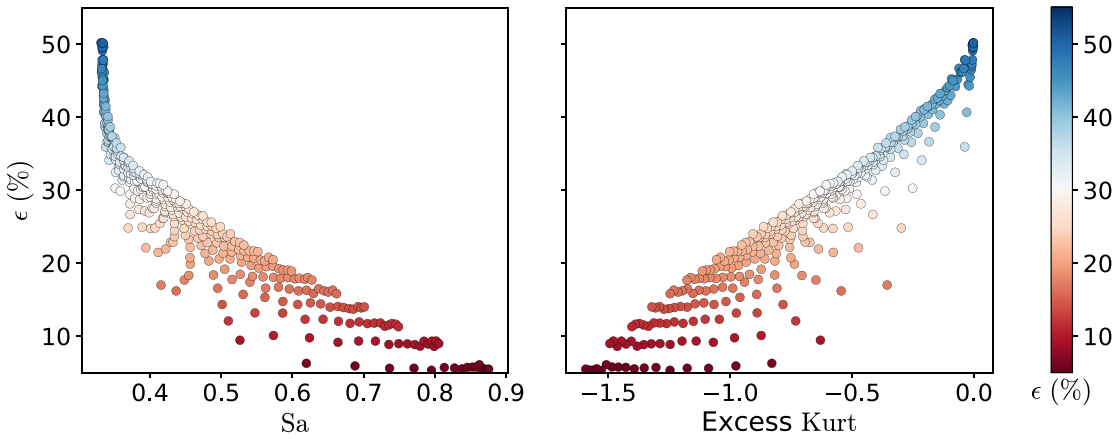

Figure 4. Statistical characterisation of the error. (Left) Scatter plot of the error of several AR(2) models trained on Markovian sequences versus the Sarle coefficient, a measure of the bimodality of the distribution $p(x_t)$, equation (11). (Right) Scatter plot of the error of several AR(2) models trained on Markovian sequences versus the excess kurtosis, equation (12), another measure of the bimodality of the distribution. *Parameters*: similar to those used in the right panel of figure 3.

the behaviour of the tails of the distribution (hence the name *excess* kurtosis). A negative value of the excess kurtosis instead indicates that the distribution is sub-Gaussian or platykurtic, i.e. having narrower tails than the Gaussian distribution. In our case, we obtain negative values of excess kurtosis whose magnitude gets larger when the distribution looks more bimodal. Furthermore, we find that the larger the more negative is the excess kurtosis, the lower is the prediction error of the AR(2) model (right of figure 4). Taken together, this analysis shows that the prediction error becomes large when the peaks of the distribution become less distinguishable and/or in the presence of fat tails.

4.3. The interplay between sequence memory and model architecture

4.3.1. The statistical properties of the input sequence $\boldsymbol{x}$ *with memory*

We saw in figure 3 that increasing the sequence memory by increasing the control parameter k of the waiting time distribution, equation (6), makes the problem *harder*: the error of the same student increased. We can understand the root of this difficulty by studying the variance of the distribution $p(x_t, c_t)$. Using the tools introduced by Tucci *et al* [31], we can calculate these correlations analytically for any integer k. As we describe in more detail in appendix A.2, we find that

$$\text{Var}\ (x_t) = \langle x_t^2 \rangle = \frac{D}{\kappa} + C_0^2 \kappa \sum_{n=0}^{k-1} \frac{Q_n}{k q_n + \kappa}, \tag{13a}$$

$$\text{Cov}\ (x_t, c_t) = \langle x_t c_t \rangle = 1 - \frac{2}{\kappa} \frac{(1+\kappa/k)^k - 1}{(1+\kappa/k)^k + 1}, \tag{13b}$$

where $Q_n \equiv -4(1-q_n)/(k q_n)^2$, and $q_n \equiv 1 - e^{i\pi(1+2n)/k}$ with $n = 0, \ldots, k-1$; we recall that $\langle x_t \rangle = \langle c_t \rangle = 0$. We plot both correlation functions as we vary k on the right of figure 5. First, we note that the variance of the particle distribution decreases with k (red line). In other words, as we increase the memory of the sequence, the particle spends on average more time around the origin, making reconstruction harder. This is also born out by a decrease in the correlation between x_t and c_t (blue line). This loss in correlations is closely related to the error of the simplest reconstruction algorithm, where we estimate the particle positions $\mathbf{c}$ by thresholding the particle position, $c_t = \text{sign}(x_t)$, and $|C_0|$ identifies the amplitude of the C_t process. As shown by a light blue line in the right panel of figure 5, the error of this parameter-free, memoryless algorithm increases monotonically with k.

4.3.2. Increasing the integration window of autoregressive and convolutional neural networks

The existence of memory in the sequences suggests choosing student architectures that can exploit these additional statistical dependencies. We first studied the performance of AR(W) models with varying W. As shown in figure 5, we observed that for a fixed k increasing W generally helps with reducing the error ϵ (solid lines), so models with larger integration window can take advantage of the additional temporal correlations in the sequence and make better predictions in the large k limit. For the convolutional networks described in section 3 with $f = 10$ filters (dashed lines), we observe that the additional filters and non-linearities in the CNN do not help with the predictions compared to the AR(W) models. We conclude that it is truly the size

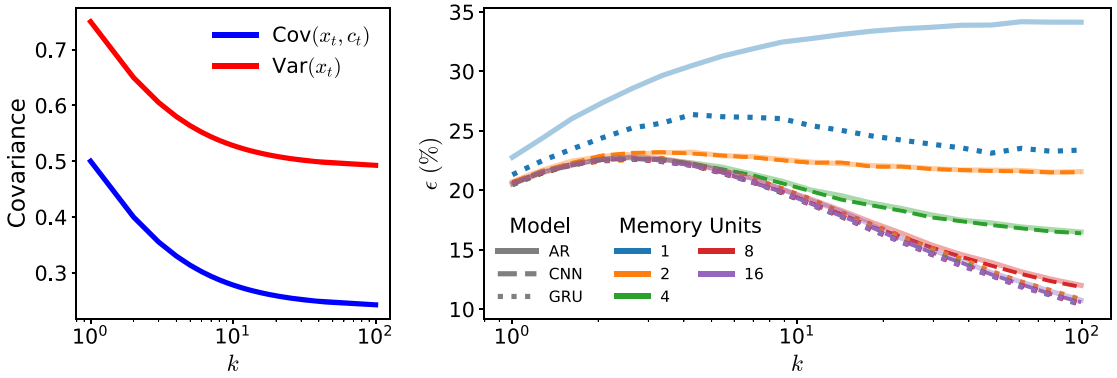

Figure 5. The impact of sequence memory on recurrent and convolutional neural networks. (Left) Analytical predictions for the covariance of the particle and trap position, equation (13a), in the non-Markovian case as a function of k. As we increase the non-Markovianity, correlations between x_t and c_t decay, complicating the reconstruction task. (Right) Prediction error ϵ for students with different architectures: auto-regressive (AR), convolutional (CNN) and gated recurrent neural network (GRU). The 'memory units' of the architectures correspond to the size of the kernel (W) in the AR and the first layer of CNN models, and the number of units (d) in the hidden state of GRU models. The CNN models all have $f = 10$ filters in their first layer. *Parameters*: similar to figure 3 with $\kappa = 2$.

of the integration window that makes a difference on this data set and that even a simple AR model can completely take advantage of the information in a given time window.

4.3.3. Recurrent neural networks

We numerically studied the performance of a gated recurrent unit GRU(d) [40] with hidden state size d, which can be thought of as a slightly more streamlined variant of the classic long short-term memory [52]. Although such an RNN reads in the input sequence one token at a time, it can exploit long-range correlations by continuously updating its internal state $\mathbf{h} \in \mathbb{R}^d$, cf figure 1. We show the test performance of GRU models with various internal state sizes d in figure 5. The GRUs display a good performance: as soon as the GRU(d) has a hidden state with dimension $d = 2$, it captures all the memory of the sequence at all k, achieving the limiting performance of AR(W) models with the largest window size. Indeed, the error curves for GRUs with $d \geqslant 2$ all coincide. This observation, together with the limited performance gain from increasing W in the AR(W) and CNN(W) models, suggest that the error is approaching the Bayes limit. Note that the rate of this convergence depends on k, since the size of the integration window required to achieve the optimal performance depends on the degree of non-Markovianity of the sequence (see figure 5). While we focused on comparing the performance of the models here, it is important to accurately compare the computational cost of these models as well. For a detailed comparison of the complexity we refer the reader to appendix A.3.

4.4. The interplay between sequence memory and model architecture

Finally, we found that for all three architectures—AR, CNN and GRU—there is a peak in the reconstruction error, usually around $k \approx 5$. This peak can be understood by noting that the tokens x_t get more concentrated around the origin as k is increased, as we have shown in equation (13a). Therefore, x_t is less correlated with c_t in the highly non-Markovian limit. However, as the degree of non-Markovianity is increased through increasing k, the history of the tokens position x_t can help with predicting the trap's position c_t more accurately. The trade-off between these two phenomena is evident in the non-monotonic dependency of the error as a function of k for models with larger integration window in figure 5. For small k, the correlation between different times in the sequence is not strong enough to compensate the loss of information about the trap's position in the signal, but for larger k the trap's position is more regular and the history of x_t can be used to infer the c_t more efficiently.

5. Concluding perspectives

We have applied the SSOU process to model seq2seq machine learning tasks. This model gives us precise control over the memory of the generated sequences, which we used to study the interaction between this memory and various neural network architectures trained on this data, ranging from simple AR models to convolutional and recurrent neural networks. We found that the accuracy of these models is governed by the interaction of the different time scales of the data, and we discovered an intriguing, non-trivial interplay between the architecture of the students and the sequence which leads to non-monotonic error curves.

An important limitation of the SSOU model considered here is that it only allows one to study correlations that decay exponentially fast. As a next step it would be interesting to explicitly consider the impact of long-range memory in the data sequence. This could be implemented in the SSOU model e.g. by choosing a long-ranged waiting time distribution, such as a power law. Another limitation concerns the linearity of the model analysed here. We speculate that a nonlinear model could provide insights into the different ways AR and CNN utilise the data. In our model, one could implement a tunable nonlinearity by considering e.g. a double-well potential with increasing amplitude of the potential barrier. Similarly, to enrich the statistical dependencies within the sequence, one could generate non-symmetric sequences with respect to a $x \to -x$ sign inversion, by implementing two different waiting-time distributions. It would also be interesting to analyse the impact of sequence memory on the dynamics of simple RNN models [53, 54], such as those with low-rank connectivity [55, 56].

Another exciting avenue would be to apply neural networks to noisy non-Markovian signals extracted from experiments in physical or biological systems [57–63]. Examples include the recent application of the SSOU to infer the mitigation of the effects of non-Markovian noise [64, 65] and the underlying heat dissipation of spontaneous oscillations of the hair-cell bundles in the ear of the bullfrog [31]. Applying our techniques to such a biological system could be fruitful to decipher the hidden mechanisms and statistics of switching in hearing and infer thermodynamic quantities beyond energy dissipation.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/alirezaseif/nonmarkovian_learning.

Acknowledgments

We thank Roman Belousov, Alessandro Ingrosso, Stéphane d'Ascoli, Andrea Gambassi, Florian Berger, Gogui Alonso, A J Hudspeth, Aljaz Godec, Marco Baiesi, and Jyrki Piilo for stimulating discussions. A S is supported by a Chicago Prize Postdoctoral Fellowship in Theoretical Quantum Science. S G acknowledges funding from Next Generation EU, in the context of the National Recovery and Resilience Plan, Investment PE1—Project FAIR 'Future Artificial Intelligence Research'. This resource was co-financed by the Next Generation EU [DM 1555 del 11.10.22]. E R acknowledges financial support from PNRR MUR Project PE0000023-NQSTI. S A M L acknowledges funding through the Marie Skłodowska-Curie Fellowship (Grant Ref. EP/X031926/1) undertaken by the UKRI, and through the Walter Benjamin Stipendium (Project No. 498288081) from the Deutsche Forschungsgemeinschaft (DFG).

Appendix A. Analytical details

A.1. Details on the calculation of the phase diagram

As anticipated in section 4.1, the process C_t becomes Markovian in the case of exponentially distributed waiting time distribution; in our units this coincides with $\psi_{k=1}(\tau) = e^{-\tau}$. For this simple case, one can characterise analytically the mono–bistable transition of the stationary density $p(x_t)$. This can be done by looking at the behaviour of $p(x_t)$ at the origin: if $x_t = 0$ is a point of maximum, $p(x_t)$ is unimodal, bimodal otherwise. We report from [31] the explicit expression of $p(x_t)$, which reads

$$p(x_t) = \frac{1}{\sqrt{\pi}} \frac{\Gamma\left(\zeta + \frac{1}{2}\right)}{\Gamma(\zeta)} \int_{-1}^{+1} \mathrm{d}z \, \frac{e^{-\chi(x_t - z)^2}}{\sqrt{\pi/\chi}} \left(1 - z^2\right)^{\zeta - 1}, \tag{A.1}$$

where we have set $C_0 = 1$, and we have defined the dimensionless parameters $\zeta = t_\kappa / \langle \tau \rangle_k = 1/\kappa$ and $\chi = t_{\mathrm{diff}}/t_\kappa = \kappa/(2D)$. One finds that $x_t = 0$ is an extremum point for $p(x_t)$, coinciding with the condition $p'(0) = 0$. The nature of $x_t = 0$ is understood by looking at the second derivative of $p(x_t)$, that is

$$p''(0) = \frac{2}{\sqrt{\pi/\chi^3}} \left[\frac{\chi}{\zeta + 1/2} \, {}_1F_1\left(\frac{3}{2}, \zeta + \frac{3}{2}, -\chi\right) - {}_1F_1\left(\frac{1}{2}, \zeta + \frac{1}{2}, -\chi\right) \right], \tag{A.2}$$

where ${}_1F_1$ denotes the confluent hypergeometric function. The mono-bimodal transition occurs upon crossing the critical value χ^* satisfying the condition $p''(0) = 0$, or equivalently, equation (10). For $\chi < \chi^*$, the second derivative $p''(0)$ is negative and $p(x)$ is unimodal, while it is bimodal otherwise.

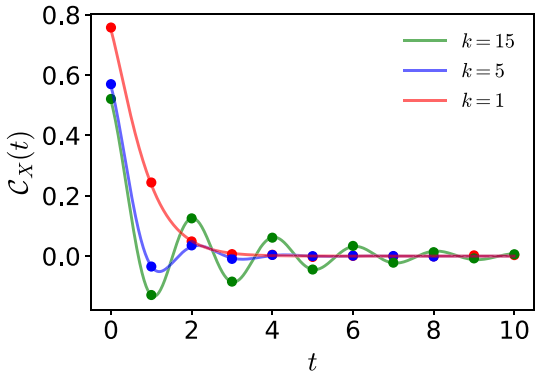

Figure 6. Autocorrelation function of the process C_t. We compare the analytical formula of $\mathcal{C}_X(t)$ in equation (A.5) (solid line) with its numerical prediction (circles), calculated by generating $N = 5 \times 10^4$ trajectories via Euler–Maruyama method with $\Delta t = 0.01$, $\kappa = 2$, $D = 0.5$ and various values of k.

When waiting times do not follow an exponential distribution, the system ceases to be Markovian. As far as our understanding goes, it becomes impossible to precisely compute the stationary probability density $p(x_t)$. Nevertheless, in the upcoming section, we will demonstrate that when $k > 1$, it remains feasible to derive a complete expression for the two-point time correlators.

A.2. Computation of the correlation functions in the non-Markovian case

In this section, we report the analytical expression for the (stationary) auto-correlation function $\mathcal{C}_X(t)$ of the process X_t in equation (1), and its Fourier transform, i.e. the power spectral density (PSD) $S_X(\omega)$. For Gamma-distributed waiting-time $\psi_k(\tau)$ as given in equation (6), the PSD $S_X(\omega)$ of the process X_t is given by [31]

$$S_X(\omega) = \frac{2D + \kappa^2 S_C(\omega)}{\kappa^2 + \omega^2}, \tag{A.3}$$

where $S_C(\omega)$ denotes the PSD of the process C_t, whose explicit expression reads

$$S_C(\omega) = \frac{4C_0^2}{\omega^2} \frac{R^2(\omega) - 1}{R^2(\omega) + 1 + 2R(\omega)\cos\phi(\omega)}, \tag{A.4}$$

with $R(\omega) = \left[1 + (\omega/k)^2\right]^{k/2}$, and $\phi(\omega) = k\arctan(\omega/k)$. According to the Wiener-Khinchin theorem [34], the inverse Fourier transform of $S_C(\omega)$ and $S_X(\omega)$ coincides with the auto-correlation functions $\mathcal{C}_C(t) \equiv \lim_{\tau\to\infty}\langle C_{t+\tau}\, C_\tau\rangle$ and $\mathcal{C}_X(t) \equiv \lim_{\tau\to\infty}\langle X_{t+\tau}\, X_\tau\rangle$. In the case of integer k, their expressions can be calculated explicitly according to

$$\mathcal{C}_C(t) = C_0^2 \sum_{n=0}^{k-1} Q_n\, e^{-tkq_n}, \tag{A.5}$$

$$\mathcal{C}_X(t) = \frac{D}{\kappa} e^{-\kappa t} + C_0^2 \kappa \sum_{n=0}^{k-1} Q_n \frac{kq_n e^{-\kappa t} - \kappa e^{-kq_n t}}{(kq_n)^2 - \kappa^2}, \tag{A.6}$$

where $Q_n \equiv -4(1-q_n)/(kq_n)^2$, and $q_n \equiv 1 - e^{i\pi(1+2n)/k}$ with $n = 0, \ldots, k-1$; in figure 6, we compare equation (A.6) with the numerical estimate of $\mathcal{C}_X$ for three different choices of k. Note that the value of the sum $\sum_{n=0}^{k-1} Q_n = 1$ implies the correct initial condition $C_c(0) = C_0^2$. Moreover, one can deduce the stationary expression of the stationary variance $\langle x_t^2\rangle$ of the process X_t from the initial value $\mathcal{C}_X(0)$. We conclude this section by reporting the value of the covariance $\langle x_t c_t\rangle$, which we derive here using similar methods as those in [31], and is given by

$$\langle x_t c_t\rangle = 1 - \frac{2}{\kappa} \frac{(1+\kappa/k)^k - 1}{(1+\kappa/k)^k + 1}. \tag{A.7}$$

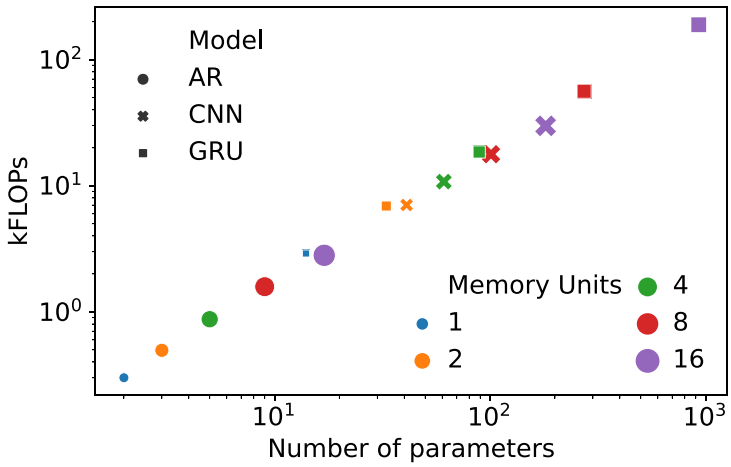

Figure 7. Complexity of the models. We analyse the size and computational complexity of the models used to generate figure 5, i.e. auto-regressive (AR), convolutional (CNN), and gated recurrent neural network (GRU) architectures. Computational complexity is measured in terms of floating-point operations (FLOPs). The 'memory units' in these architectures correspond to the kernel size (W) in AR and the GRU hidden state size (d). Notably, all CNN models employ 10 filters in their first layer.

A.3. Complexity of the models

In this section we analyse the size and computational complexity of the models used in this study. We note that the exact complexity depends on the specific implementation of the models. Therefore, we first analyse the scaling of the model complexity with respect to the number of model's memory units and then characterise the number of floating-point operations (FLOPs) in our particular implementation.

In this scaling analysis we do not consider the scaling with the sequence size it is the same for all of the models. The AR models we examine feature a 1D kernel of size W. Both the model's parameter count and computational complexity for its forward pass increase linearly with W. Similarly, 1D CNNs comprise f kernels of size W. For these models, the number of parameters and computational cost scale primarily as Wf. The GRUs have a hidden state of size d. The number of parameters and the computational cost of these model scale with d^2.

We employ software profiling to accurately compare the size and computational cost of the models used to produce figure 5. As demonstrated in figure 7, our analysis agrees with the numerical results. The AR models exhibit the smallest size and computational cost, followed by CNNs, and finally RNNs. Additionally, our simple analysis correctly predicts linear scaling of cost with the number of parameters.

Appendix B. Additional experimental results

We use a fixed number of 40 epochs and do not perform any hyperparameter tuning (with the exception of the GRU(1) model as detailed in appendix B) as the models are rather simple and varying these parameters have negligible effects on the results.

B.1. Additional details on creating the figures

In this section, we provide additional details on the generation of some of the figures of the main text.
Figure 3. The parameter sweep in the top left and top right panels was done by inspecting the intervals $\kappa \in [2/3, 10]$ and $D \in [1/21, 50]$ in a regular grid of size 20×20. We used a finite-sample estimate for Sarle's coefficient given by $(\sigma^2 + 1)/(3 + \omega\mathcal{K})$, with σ the skewness and $\mathcal{K}$ the kurtosis of the sequence $\mathbf{X}$, and $\omega = (T-1)^2/(T-2)(T-3)$ and $T = \tau/\Delta t = 3 \times 10^3$.

B.2. Ensuring the consistency between continuous-time theory and sampled sequences

Here we discuss how we ensured that our theory that is based on the continuous-time description of the stochastic processes X_t and C_t (cf section 2) and the sequences we find into the machine learning models are consistent. If the sequence is sampled with a very small time step, the student almost never sees a jump of the trap, and effectively samples from an (equilibrium) distribution in one trap. On the other hand, in the limit of very large time steps, the temporal correlations in the sequence is not visible, e.g. recall that the non-Markovianity parameter eventually decays to zero for infinitely large time differences. Thus, it is crucial

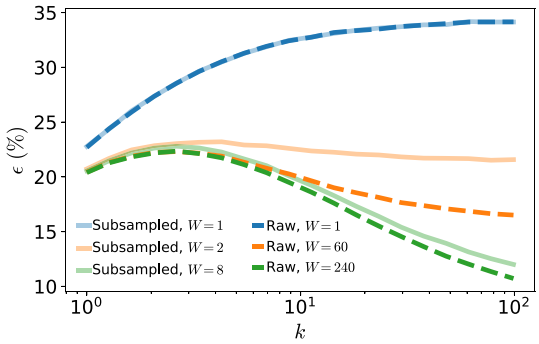

Figure 8. Subsampling effects. We compare the error of AR models trained on original raw dataset and the subsampled one with a subsampling factor of $s = 30$. The key qualitative features of the error curves as function of the non-Markovianity k are not affected by subsampling.

to sample with a time step that is smaller but comparable to the characteristic time scales of the data sequence. In practice, one would thus need to estimate those time scales in a preceding data analysis step, before feeding the data into the network. For this purpose, a suitable method is to compute the autocorrelation function of the input sequence data which reveals the time scales. For the SSOU model, we provide the analytical expression for the correlation function in equation (13a). For the SSOU, the characteristic time scales are given by the oscillation period and the decay time of the autocorrelation function, which are of order of magnitude 1 for our parameter choice (see figure 6).

B.2.1. Correlations function and integration time step

To generate trajectories with correct statistics we vary the time differences Δt, which corresponds to the discrete version of dt in numerically integrating equation (1). We find that $\Delta t = 0.005$ reproduces the correct statistics in figure 6.

B.2.2. The role of subsampling

As mentioned earlier the trajectories are first generated by numerically integrating equation (1) by choosing a sufficiently small dt such that the statistical properties of the trajectories match their theoretical values, which we verified by checking the correlation functions estimated analytically match our theoretical result (see figure 6).

As we discuss in the main text, we subsample the full trajectories before training the machine learning models. This subsampling also simplifies the optimisation and accelerates the training. However, in doing so, we need to make sure that the qualitative properties of the model remain unchanged. Therefore, we compare AR models trained on the original trajectories with those trained on subsampled trajectories. We observe that the key properties of the problem, namely the strictly increasing errors with k in the memoryless learning ($W = 1$), and non-monotonic behaviour of errors due to the trade-off between the memory (W) and the non-Markovianity (k) in more complicated models are preserved.

As mentioned in the main text we subsample the sequences by taking every sth element of the original sequence in the dataset when training the models in section 4.3. To investigate the effect of this subsampling we compare AR models trained on original dataset with those trained on subsampled sequences with $s = 30$, and show the results in figure 8. The qualitative features of the error curves as a function of non-Markovianity k are preserved. The monotonicity of errors for memoryless models and the trade-off between improvements by memory and the difficulty of correlating the particle's position x_t to the trap's position c_t, the two main features of figure 5, are present in both the original and the subsampled cases.

B.3. The thresholding algorithm

As a baseline benchmark, we consider a simple thresholding algorithm, where $\hat{c}_t = \mathrm{sgn}(x_t)$. Put simply, this algorithm infers the potential's location by only considering the position of the particle at a single point x_t and choose the closest configuration of potential c_t to that position. In figure 9 we compare the performance of the AR(1) model with the thresholding algorithm and observe that AR(1) matches the performance of the the thresholding algorithm.

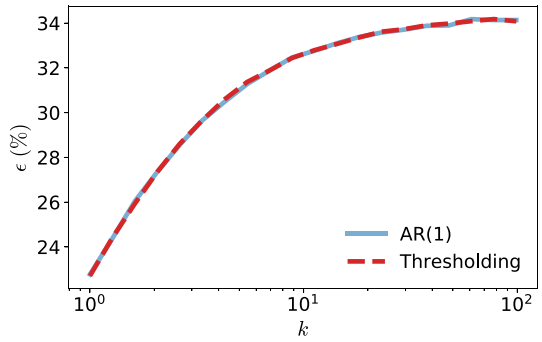

Figure 9. Comparing the AR(1) and the thresholding algorithm. Both the AR(1) model and the thresholding algorithm only have access to the position of the particle at a single point. As a consistency check we compare the performance of the two and observe that they match.

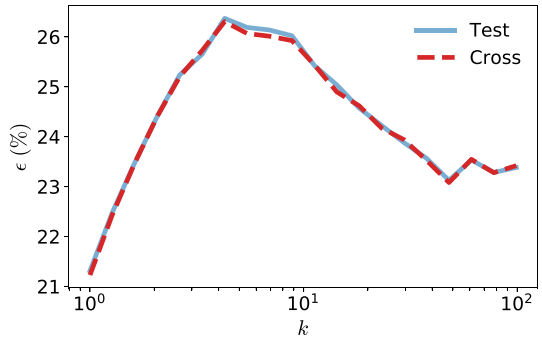

Figure 10. Validating the GRU(1) model. The accuracy of the best performing GRU(1) models over the cross validation and test sets match.

B.4. Selecting the best GRU(1) model

To find the best performing GRU(1) we train model for all k values shown in figure 5 we train five different instance of the network initialised randomly using as suggested in [66]. We choose the best performing model over 10 000 samples for each k, and report the accuracy on a new set of 10 000 samples. As shown in figure 10 the test and cross validation accuracy are in excellent agreement.

ORCID iDs

Alireza Seif https://orcid.org/0000-0001-5419-5999
Sarah A M Loos https://orcid.org/0000-0002-5946-5684
Édgar Roldán https://orcid.org/0000-0001-7196-8404
Sebastian Goldt https://orcid.org/0000-0002-5799-7644

References

[1] Devlin J, Chang M W, Lee K and Toutanova K 2019 BERT: pre-training of deep bidirectional transformers for language understanding (arXiv:1810.04805)
[2] Howard J and Ruder S 2018 arXiv:1801.06146
[3] Radford A *et al* 2018 Improving language understanding by generative pre-training
[4] Brown T *et al* 2020 Language models are few-shot learners *Advances in Neural Information Processing Systems* vol 33 (Curran Associates, Inc.) pp 1877–901
[5] OpenAI 2023 Gpt-4 technical report (arXiv:2303.08774v2)
[6] Kantz H and Schreiber T 2004 *Nonlinear Time Series Analysis* vol 7 (Cambridge University Press) (available at: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf)
[7] Box G E, Jenkins G M, Reinsel G C and Ljung G M 2015 *Time Series Analysis: Forecasting and Control* (Wiley)
[8] Gardner E and Derrida B 1989 Three unfinished works on the optimal storage capacity of networks *J. Phys. A: Math. Gen.* **22** 1983
[9] Seung H S, Sompolinsky H and Tishby N 1992 Statistical mechanics of learning from examples *Phys. Rev.* A **45** 6056
[10] Engel A and Van den Broeck C 2001 *Statistical Mechanics of Learning* (Cambridge University Press)
[11] Mezard M and Montanari A 2009 *Information, Physics and Computation* (Oxford University Press)
[12] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 Machine learning and the physical sciences *Rev. Mod. Phys.* **91** 045002

[13] Krizhevsky A, Sutskever I and Hinton G 2012 ImageNet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* pp 1097–105
[14] Simonyan K and Zisserman A 2015 Very deep convolutional networks for large-scale image recognition *Int. Conf. on Learning Representations*
[15] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 770–8
[16] Dosovitskiy A *et al* 2021 An image is worth 16x16 words: transformers for image recognition at scale *Int. Conf. on Learning Representations*
[17] Pope P, Zhu C, Abdelkader A, Goldblum M and Goldstein T 2021 The intrinsic dimension of images and its impact on learning *Int. Conf. on Learning Representations*
[18] Chung S, Lee D D and Sompolinsky H 2018 Classification and geometry of general perceptual manifolds *Phys. Rev.* X **8** 031003
[19] Goldt S, Mézard M, Krzakala F and Zdeborová L 2020 Modeling the influence of data structure on learning in neural networks: the hidden manifold model *Phys. Rev.* X **10** 041044
[20] Goldt S, Loureiro B, Reeves G, Krzakala F, Mezard M and Zdeborová L 2022 The Gaussian equivalence of generative models for learning with shallow neural networks *Proc. 2nd Mathematical and Scientific Machine Learning Conf.* (*Proc. Machine Learning Research* vol 145) ed J Bruna, J Hesthaven and L Zdeborová (PMLR) pp 426–71
[21] Ghorbani B, Mei S, Misiakiewicz T and Montanari A 2020 When do neural networks outperform kernel methods? *Advances in Neural Information Processing Systems* vol 33
[22] Richards D, Mourtada J and Rosasco L 2021 Asymptotics of ridge(less) regression under general source condition *Proc. 24th Int. Conf. on Artificial Intelligence and Statistics* (*Proc. Machine Learning Research* vol 130) ed A Banerjee and K Fukumizu (PMLR) pp 3889–97
[23] Chizat L and Bach F 2020 Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss *Conf. on Learning Theory* (PMLR) pp 1305–38
[24] Refinetti M, Goldt S, Krzakala F and Zdeborová L 2021 Classifying high-dimensional Gaussian mixtures: where kernel methods fail and neural networks succeed *Proc. 38th Int. Conf. on Machine Learning* (*Proc. Machine Learning Research* vol 139) ed M Meila and T Zhang (PMLR) pp 8936–47
[25] Loureiro B, Sicuro G, Gerbelot C, Pacco A, Krzakala F and Zdeborová L 2021 Learning Gaussian mixtures with generalized linear models: precise asymptotics in high-dimensions *Advances in Neural Information Processing Systems* vol 34
[26] Spigler S, Geiger M and Wyart M 2020 Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm *J. Stat. Mech.* 124001
[27] d'Ascoli S, Gabrié M, Sagun L and Biroli G 2021 On the interplay between data structure and loss function in classification problems *Advances in Neural Information Processing Systems* vol 34, ed M Ranzato, A Beygelzimer, Y Dauphin, P Liang and J W Vaughan (Curran Associates, Inc.) pp 8506–17
[28] Benna M K and Fusi S 2021 Place cells may simply be memory cells: memory compression leads to spatial tuning and history dependence *Proc. Natl Acad. Sci.* **118** e2018422118
[29] Gerace F, Saglietti L, Mannelli S S, Saxe A and Zdeborová L 2022 Probing transfer learning with a model of synthetic correlated datasets *Mach. Learn.: Sci. Technol.* **3** 015030
[30] Ghorbani B, Mei S, Misiakiewicz T and Montanari A 2019 Limitations of lazy training of two-layers neural network *Advances in Neural Information Processing Systems* vol 32 pp 9111–21
[31] Tucci G, Roldán E, Gambassi A, Belousov R, Berger F, Alonso R G and Hudspeth A J 2022 Modeling active non-Markovian oscillations *Phys. Rev. Lett.* **129** 030603
[32] Pietzonka P, Ritort F and Seifert U 2017 Finite-time generalization of the thermodynamic uncertainty relation *Phys. Rev.* E **96** 012101
[33] Di Terlizzi I, Gironella M, Herráez-Aguilar D, Betz T, Monroy F, Baiesi M and Ritort F 2023 Variance sum rule for entropy production (arXiv:2302.08565)
[34] Van Kampen N G 1992 *Stochastic Processes in Physics and Chemistry* (Elsevier)
[35] Martinez I A and Petrov D 2012 Force mapping of an optical trap using an acousto-optical deflector in a time-sharing regime *Appl. Opt.* **51** 5522–6
[36] Martínez I A, Roldán E, Parrondo J M R and Petrov D 2013 Effective heating to several thousand kelvins of an optically trapped sphere in a liquid *Phys. Rev.* E **87** 032159
[37] LeCun Y, Boser B, Denker J S, Henderson D, Howard R E, Hubbard W and Jackel L D 1989 Backpropagation applied to handwritten zip code recognition *Neural Comput.* **1** 541–51
[38] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (MIT Press)
[39] Fukushima K 1969 Visual feature extraction by a multilayered network of analog threshold elements *IEEE Trans. Syst. Sci. Cybern.* **5** 322–33
[40] Cho K, van Merriënboer B, Bahdanau D and Bengio Y 2014 On the properties of neural machine translation: encoder–decoder approaches *Proc. SSST-8, 8th Workshop on Syntax, Semantics and Structure in Statistical Translation* pp 103–11
[41] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[42] Seif A 2022 Code for data generation and training and testing machine learning models *The impact of memory on learning sequence-to-sequence tasks* (available at: https://github.com/alirezaseif/nonmarkovian_learning)
[43] Kloeden P E and Platen E 1992 *Numerical Solution of Stochastic Differential Equations* (Springer) pp 103–60
[44] Lapolla A and Godec A 2021 Toolbox for quantifying memory in dynamics along reaction coordinates *Phys. Rev. Res.* **3** L022018
[45] Lapolla A and Godec A 2019 Manifestations of projection-induced memory: general theory and the tilted single file *Front. Phys.* **7** 182
[46] Laine E-M, Piilo J and Breuer H-P 2010 Measure for the non-Markovianity of quantum processes *Phys. Rev.* A **81** 062115
[47] Hall M J W, Cresser J D, Li L and Andersson E 2014 Canonical form of master equations and characterization of non-Markovianity *Phys. Rev.* A **89** 042120
[48] Rivas A, Huelga S F and Plenio M B 2010 Entanglement and non-Markovianity of quantum evolutions *Phys. Rev. Lett.* **105** 050403
[49] Huang Z and Guo X-K *et al* 2021 Quantifying non-Markovianity via conditional mutual information *Phys. Rev.* A **104** 032212
[50] Strasberg P and Esposito M 2018 Response functions as quantifiers of non-Markovianity *Phys. Rev. Lett.* **121** 040601
[51] Ellison A M 1987 Effect of seed dimorphism on the density-dependent dynamics of experimental populations of atriplex triangularis (chenopodiaceae) *Am. J. Bot.* **74** 1280–8
[52] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80

[53] Sompolinsky H, Crisanti A and Sommers H J 1988 Chaos in random neural networks *Phys. Rev. Lett.* **61** 259
[54] Sussillo D and Abbott L F 2009 Generating coherent patterns of activity from chaotic neural networks *Neuron* **63** 544–57
[55] Mastrogiuseppe F and Ostojic S 2018 Linking connectivity, dynamics and computations in low-rank recurrent neural networks *Neuron* **99** 609–23
[56] Schuessler F, Mastrogiuseppe F, Dubreuil A, Ostojic S and Barak O 2020 The interplay between randomness and structure during learning in RNNs *Advances in Neural Information Processing Systems* vol **33** pp 13352–62
[57] Mindlin G B 2017 Nonlinear dynamics in the study of birdsong *Chaos* **27** 092101
[58] Vettoretti G and Peltier W R 2018 Fast physics and slow physics in the nonlinear Dansgaard–Oeschger relaxation oscillation *J. Clim.* **31** 3423
[59] Cavallaro M and Harris R J 2019 Effective bandwidth of non-Markovian packet traffic *J. Stat. Mech.* 083404
[60] Roldán E, Barral J, Martin P, Parrondo J M R and Jülicher F 2021 Quantifying entropy production in active fluctuations of the hair-cell bundle from time irreversibility and uncertainty relations *New J. Phys.* **23** 083013
[61] Belousov R, Berger F and Hudspeth A 2020 Volterra-series approach to stochastic nonlinear dynamics: linear response of the Van der Pol oscillator driven by white noise *Phys. Rev.* E **102** 032209
[62] Brückner D B, Fink A, Schreiber C, Röttgermann P J, Rädler J O and Broedersz C P 2019 Stochastic nonlinear dynamics of confined cell migration in two-state systems *Nat. Phys.* **15** 595
[63] Skinner D J and Dunkel J 2021 Estimating entropy production from waiting time distributions *Phys. Rev. Lett.* **127** 198101
[64] Mavadia S, Frey V, Sastrawan J, Dona S and Biercuk M J 2017 Prediction and real-time compensation of qubit decoherence via machine learning *Nat. Commun.* **8** 1–6
[65] Majumder S, Andreta de Castro L and Brown K R 2020 Real-time calibration with spectator qubits *npj Quantum Inf.* **6** 1–9
[66] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. 13th Int. Conf. on Artificial Intelligence and Statistics* pp 249–56