*Cambridge Large Two* (2025), 1–8

ARTICLE TYPE

# Recurrent Neural Networks as Nonlinear Dynamical Systems: A Study in ECG-Derived Time Series

Adam Nolan

The American University of Paris, Paris, France

**Abstract**

This report explores feedforward and recurrent neural networks from the perspective of discrete-time nonlinear dynamical systems. Neural architectures are interpreted as iterated maps on state space, where nonlinear activation functions and weight matrices govern state evolution over time. Within this framework, learning is viewed as a parameter-driven process that gradually reshapes the underlying dynamics of neural networks, making concepts such as fixed points, stability, and Jacobians natural tools for analysis. Both minimalist feedforward and recurrent models are first examined to clarify the underlying mechanisms of feed forward propagation and backpropagation through time (BPTT). These theoretical considerations are then illustrated through a case study involving heart rate variability (HRV) signals derived from electrocardiographic recordings in the MIT-BIH Arrhythmia Database. A simple, self-constructed recurrent neural network is used to model the resulting time series, not with the aim of clinical prediction, but to demonstrate how learned recurrent dynamics can reflect structure in feedback-driven biological signals. Throughout, the emphasis is placed on dynamical interpretation and conceptual understanding rather than predictive optimization.

**Keywords:** Recurrent Neural Networks, Nonlinear Dynamical Systems, Heart Rate Variability (HRV), Time Series Modeling, Electrocardiograph (ECG)

## 1. Introduction

Many systems encountered in mathematics, physics, biology, and computation evolve through time in ways that are nonlinear, memory-dependent, and difficult to capture using simple linear models (*e.g.*, straight-line regression, multiple linear regression, linear auto regression, *et al.*) In such systems, the dynamics themselves are crucial, where static input–output descriptions are simply inadequate to capture their full complexity. This leads us to models that explicitly represent internal state and temporal evolution.

Artificial neural networks are best understood when viewed through this lens. Rather than treating them solely as function approximators, it often proves useful to think of neural networks as *discrete-time dynamical systems* whose state evolves through the repeated application of nonlinear transformations (*i.e.,* sigmoid & tanh for our present considerations). Each layer update corresponds to a state transition shaped by learned parameters, and training alters the geometry of this state space over time.

Recurrent neural networks (RNNs) make this interpretation particularly transparent. By feeding the hidden state back into the network recursively, RNNs define nonlinear recurrence relations of the form

$$h_{t+1} = F(h_t, x_t; \theta),$$

where $h_t$ denotes the internal state, $x_t$ an external input, and $\theta$ the collection of learnable parameters, while F is a nonlinear function that maps the current state and input to the next state. Written in this way, recurrent networks sit comfortably within the confines of discrete-time nonlinear dynamical systems, which gives way to analysis using ideas familiar to dynamical systems such as fixed points, stability, and sensitivity to perturbations.

The aim of this project is to develop an intuitive and mathematical understanding of neural networks, in particular recurrent architectures, using the language and tools of dynamical systems. With this in mind, simplified feedforward and recurrent models are examined first, allowing the basic mechanisms of state evolution, nonlinearity, and learning to be studied without unnecessary architectural complexity. Only after this theoretical foundation is well established, will we then turn to a concrete application.

In the final portion of the report, recurrent neural networks are applied to heart rate variability (HRV) signals derived from electrocardiographic data in the MIT-BIH Arrhythmia Database. This data serves not as the central object of study, rather, it serves as a real-world example of how learned nonlinear dynamics can model irregular biological time series generated by feedback-driven processes as we see contained within the heart.

## 2. Mathematical Background

### *2.1 Discrete-Time Dynamical Systems*

A discrete-time dynamical system may be written as an iterated map, *i.e.*, a repeated functional composition of the form

$$x_{k+1} = F(x_k),$$

where $x_k \in R^n$ represents the state of the system at time step $k$, and $F : R^n \to R^n$ is a (typically nonlinear) update rule. Once an initial condition $x_0$ is properly specified, the system's evolution is then determined entirely by repeated application of our nonlinear function $F$.

Despite their simple form, discrete-time systems can exhibit a myriad of complex behaviors, including convergence to fixed points, periodic motion, and chaotic trajectories; quintessential concepts encountered in the study of dynamical systems. Qualitative analysis often focuses on stability, sensitivity to perturbations, and the geometry of trajectories in state space.

Additionally, the update rule depends on further inputs or parameters. A more general formulation is therefore

$$x_{k+1} = F(x_k, u_k; \theta),$$

where $u_k$ denotes an input signal and, as previously mentioned, $\theta$ represents the parameters governing the dynamics of the system. This formulation suits learning systems quite nicely, where parameters are adjusted in response to data.

### 2.2 Feedforward Networks and Continuous-Time Dynamics

Classical dynamical systems are often modeled in continuous time using ordinary differential equations of the form

$$\dot{x} = f(x(t)),$$

or equivalently, in component form,

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1, x_2, \ldots, x_n),\\
\dot{x}_2 &= f_2(x_1, x_2, \ldots, x_n),\\
&\vdots\\
\dot{x}_n &= f_n(x_1, x_2, \ldots, x_n),
\end{aligned}$$

where $x(t) \in R^n$ denotes the system state. Systems such as these generate smooth trajectories in state space, and their qualitative behavior is determined by the structure of the vector field $f$, including its equilibria and stability properties.

In contrast to continuous, neural networks operate in *discrete* time. Let us consider a feedforward network with a single hidden layer and nonlinear activation function $\sigma(\cdot)$ (often assumed to be sigmoidal, hence the Greek letter sigma),

$$h = \sigma(Wx + b).$$

Where $h$ denotes the hidden state vector, $W$ the weight matrix, $x$ the input vector, and finally $b$ the bias. Although this mapping is typically treated as static, it can be interpreted dynamically by introducing an iterative index,

$$h_{k+1} = \sigma(Wh_k + b).$$

Written in such a way, the network defines a discrete-time nonlinear evolution on $R^n$. Rather than specifying a vector field *explicitly*, as in the case of systems governed by known differential equations, the dynamics arise *implicitly* through repeated application of a nonlinear map. This perspective mirrors the relationship between continuous systems and their numerical discretizations, even though the neural update is not derived from an explicit underlying differential equation.

In continuous-time systems, local behavior near an equilibrium $x^*$ is analyzed by linearizing the vector field,

$$J_f(x^*) = \left[\frac{\partial f_i}{\partial x_j}\right]_{i,j=1}^{n},$$

with eigenvalues of the Jacobian determining its stability. In the discrete setting, fixed points satisfy $h^* = F(h^*)$, and local behavior is governed by the Jacobian of the update map,

$$J_F(h^*) = \left[\frac{\partial F_i}{\partial h_j}\right]_{i,j=1}^{n} = \left.\frac{\partial h_t}{\partial h_{t-1}}\right|_{h_t = h^*}.$$

While these Jacobians share the same mathematical form, their interpretation differs. In continuous systems, the Jacobian arises from *instantaneous rates of change*, whereas in neural networks it reflects how *small perturbations* propagate from one discrete update to the next. In both cases, however, the Jacobian provides a local description of sensitivity and stability.

### 2.3 Recurrent Neural Networks as Nonlinear Recurrences

Recurrent neural networks extend feedforward architectures by explicitly incorporating feedback. A vanilla RNN updates its hidden state according to

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h),$$

with output

$$y_t = W_{hy}h_t + b_y.$$

This structure defines a nonlinear recurrence relation driven by input, placing RNNs within the framework of discrete-time nonlinear dynamical systems. The hidden state serves as a memory variable, allowing information from previous inputs to influence future behavior.

A helpful illustration can be found in in sequence prediction tasks, such as language modeling, where networks are trained to predict the next element in a sequence. In this setting, the hidden state aggregates information from earlier inputs, enabling predictions to depend on a learned, variable-length history. Rather than relying on a fixed window, the network learns how far into the past relevant information must persist.

From a dynamical perspective, the recurrence

$$h_{t+1} = F(h_t, x_t)$$

defines a *trajectory* through state space driven by the input sequence. Stability, sensitivity, and memory properties depend on both the recurrent weight matrix and more importantly the choice of activation function.

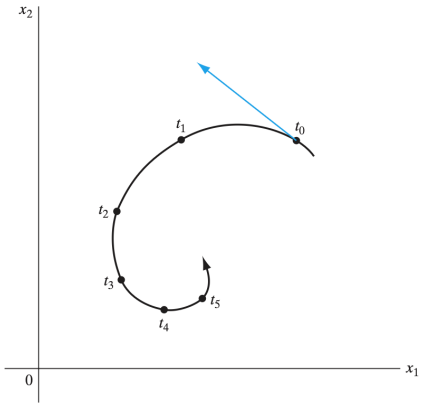


**Figure 1.** Two-dimensional trajectory of a dynamical system.

## 3. Learning Dynamics in Recurrent Neural Networks

### *3.1 Learning as a Dynamical Process*

Up to this point, neural networks have been interpreted as dynamical systems evolving in state space. It is natural, then, to ask how learning itself fits into this picture. Training introduces an additional layer of dynamics governing the evolution of the network parameters.

Let $\theta \in R^p$ denote the collection of all learnable parameters (in our case the weight matrices and biases). Gradient-based optimization updates these parameters according to

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$$

where $\mathcal{L}$ is a loss function and $\eta$ the learning rate. This update rule also defines a discrete-time dynamical system on parameter space.

The full learning process may therefore be viewed as *two coupled dynamical systems*: one governing the evolution of hidden states, and another governing the evolution of parameters. As parameters change, the state-space dynamics themselves are gradually reshaped. Hence, learning optimizes performance as well as deforms the geometry and stability of the neural dynamical system over time.

### *3.2 Backpropagation and Backpropagation Through Time*

In order to better understand learning dynamics in neural networks, it is important to clarify how derivatives are computed. Although gradients are often written using partial derivative notation, the derivatives arising in neural networks differ conceptually from classical Cartesian partial derivatives taken with respect to independent coordinates. Instead, they are computed along a computational graph defined by the composition of functions that make up the network.

In a feedforward neural network, each layer defines a mapping from its input to its output, and the full network represents a nested composition of functions. The loss function depends on the output of the final layer, which in turn depends on the parameters through this sequence of transformations. Differentiation therefore proceeds by repeated application of the chain rule along the directed edges of the computational graph.

Backpropagation is best understood as a systematic method for propagating sensitivities backward through the graph. Each intermediate variable contributes to the loss indirectly through its influence on downstream computations. Gradients are accumulated by composing local derivatives, rather than by varying one coordinate independently while holding others fixed, as in traditional multivariable calculus.

In recurrent neural networks, this structure is extended across time. The hidden state is reused at each time step, and the same parameters influence the system repeatedly through the recurrence relation

$$h_{t+1} = F(h_t, x_t; \theta).$$

Backpropagation through time (BPTT) *unfolds* this recurrence into a deep feedforward network, with shared parameters appearing at each time step. Gradients with respect to the parameters are then computed by applying the chain rule across both layers and time steps.

It is important to note, the resulting derivatives reflect how perturbations to parameters affect the entire state trajectory, not just a single update. Each term in the gradient corresponds to a path through the computational graph, linking parameter changes to the loss via their influence on successive hidden states. This interpretation emphasizes that the gradients computed during BPTT are sensitivities of a dynamical process, rather than ordinary partial derivatives taken in isolation.

We are then free to connect backpropagation to the Jacobian-based analysis of recurrent dynamics. Just as perturbations to the hidden state propagate forward through products of Jacobian matrices, gradients propagate backward through transposed Jacobians. What should be understood and integrated is that the behavior of learning algorithms is deeply tied to the same stability properties that govern the state evolution.

### *3.3 Stability, Memory, and Gradient Propagation*

The ability of an RNN to retain information over time is tied to the stability properties of its hidden-state dynamics. Perturbations to the hidden state propagate forward through repeated multiplication by Jacobian matrices, and their long-term behavior is governed by the associated spectral properties *i.e.*, whether the Jacobian induces contraction, neutrality, or expansion of perturbations through its eigenvalue spectrum.

To make this connection precise, consider an autonomous discrete-time dynamical system defined by the recurrence

$$h_{t+1} = F(h_t),$$

a small perturbation evolves approximately as

$$\delta h_{t+k} \approx J^k \delta h_t,$$

where $J$ is the local Jacobian. Here, $\delta h_t$ denotes a small perturbation of the hidden state at time $t$, representing an infinitesimally tiny deviation from a reference trajectory in our state space. The superscript $k$ on the Jacobian indicates repeated matrix multiplication, so that $J^k$ describes the cumulative linearized effect of the dynamics over $k$ successive time

steps. If all eigenvalues of the Jacobian satisfy $|\lambda| < 1$, perturbations decay and memory is short. If $|\lambda| > 1$, perturbations grow, leading to instability. When $|\lambda| \approx 1$, perturbations persist, allowing information to be retained over much longer horizons.

These same mechanisms govern gradient propagation during training. Contractive dynamics (*i.e.*, $|\lambda| < 1$) lead to vanishing gradients, while expansive dynamics (*i.e.*, $|\lambda| > 1$) give rise to exploding gradients. From a dynamical systems perspective, these effects are natural consequences of the recurrence structure as opposed to optimization failures.

### 3.4 Activation Functions, Stability, and Gating

Our anteceding discussion highlights the sensitivity of recurrent neural networks to stability and saturation effects. Given that state evolution and gradient propagation are governed by repeated application of nonlinear maps, the choice of activation function is integral to shaping the underlying dynamics. As we will see, recurrent architectures employ the hyperbolic tangent function rather than the logistic sigmoid and this proves to be the more suitable choice for maintaining stable and expressive state dynamics.

Recall that the logistic sigmoid function,

$$\sigma(x) = \frac{1}{1 + e^{-x}},$$

maps inputs to the interval $(0, 1)$ and has historically been used in feedforward models. While this is ideal for bounded output representations, its use within recurrent hidden-state dynamics exhibits several limitations. In particular, the sigmoid function is asymmetric about zero and exhibits strong saturation for large positive or negative inputs. In a recurrent setting, this saturation leads to small derivatives over much of the state space, causing perturbations and gradients to decay rapidly.

By contrast, the hyperbolic tangent activation,

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}},$$

maps inputs to $(-1, 1)$ and is symmetric about the origin. This symmetry allows the hidden state to represent both positive (excitatory) and negative (inhibitory) deviations from a baseline, for a more balanced state evolution. Furthermore, the derivative of $\tanh(x)$ remains relatively large near zero, which helps preserve sensitivity to perturbations when the system operates in such a way. As a result, tanh is better suited for maintaining stable yet expressive recurrent dynamics.

From a dynamical systems perspective, these differences can be understood in terms of how activation functions influence the Jacobian of the recurrence relation. The derivative of the activation function directly scales the Jacobian matrix governing state evolution. Activation functions that saturate too quickly induce strong contraction, leading to rapid loss of memory and vanishing gradients. Functions with broader linear regions allow perturbations to persist, supporting longer-term dependencies.

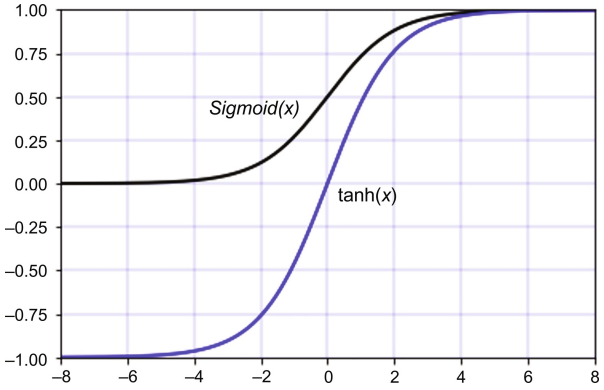


**Figure 2.** Sigmoid versus hyperbolic tangent. Note the asymmetry about the origin for $\sigma(x)$, and the larger first derivative of $\tanh(x)$.

Though not involved in the hidden state nonlinearity, the sigmoid function does indeed play a significant role in recurrent architectures when used for gating (about the output) rather than state evolution. In this way, sigmoid nonlinearities act as smooth, bounded control variables that regulate information flow by controlling how much of a signal is transmitted or suppressed. When combined with a state activation such as tanh, this separation of roles allows recurrent networks to balance stability and flexibility.

Although the present work focuses on a minimal recurrent architecture, these considerations illustrate how activation functions are not just simple implementation details, but genuinely integral components shaping the qualitative behavior of recurrent dynamical systems. Their selection reflects a trade-off between stability, sensitivity, and representational capacity, all of which are paramount in learning and memory in recurrent networks.

## 4. From Abstract Dynamics to Biological Time Series

### 4.1 Heart Rate Variability as a Nonlinear Biological Signal

Heart rate variability (HRV) refers to fluctuations in the time intervals between successive heartbeats. Rather than being random noise, these fluctuations arise from the interaction of multiple physiological control mechanisms operating across several different time scales. The resulting signal exhibits irregular yet structured behavior characteristic of nonlinear, feedback-driven systems.

From the perspective developed in earlier sections, HRV may be viewed quite comfortably as an observable generated by an underlying dynamical system with internal state and memory. The timing of a given heartbeat depends not only on immediate conditions, but also on the system's recent history, which leads to a temporal dependence. These properties make HRV poorly suited to simple linear or memoryless models, while naturally motivating the use of recurrent architectures capable of representing internal state.

### *4.2 From ECG Signals to an HRV Time Series*

Electrocardiographic (ECG) recordings provide measurements of the electrical activity of the heart over time. While the ECG waveform contains detailed morphological information, HRV analysis focuses instead on the timing between successive ventricular contractions. This is achieved by identifying R-peaks in the ECG signal and computing the intervals between consecutive peaks, which yields a sequence of RR intervals.

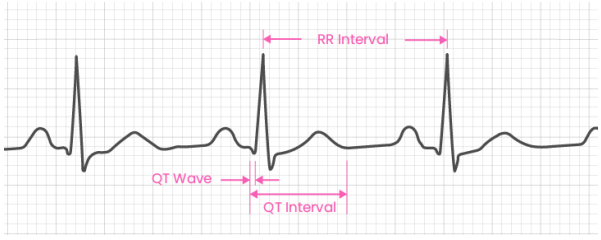


**Figure 3.** Canonical "P-Q-R-S-T" ECG morphology highlighting the distance from one "R" peak to another.

The resulting RR interval series forms a univariate time series indexed by beat number rather than continuous time. This representation emphasizes temporal structure while abstracting away high-frequency waveform details. From a modeling standpoint, this transformation isolates the dynamics of beat-to-beat timing, producing a signal whose evolution reflects the action of underlying regulatory processes.

In the present work, RR interval sequences derived from the MIT-BIH Arrhythmia Database are used as a concrete biological example of a nonlinear time series with memory and feedback.

## 5. Case Study: Modeling HRV with a Vanilla Recurrent Neural Network

### *5.1 Data Source and Preprocessing*

As previously mentioned, ECG recordings from the MIT-BIH Arrhythmia Database serve as the empirical basis for this study. It is important to note that the dataset is employed not for clinical diagnosis or classification, but as a source of real-world time series data set generated by a biological system exhibiting nonlinear temporal structure.

After extracting R-peak locations from the ECG annotations, successive RR intervals are computed and expressed in seconds. To ensure numerical stability during training, the RR sequence is standardized using a z-score normalization, a technique well known to the statistical and mathematical community at large. The normalized series is then segmented into overlapping, fixed-length windows, each serving as a short trajectory of the underlying system.

Each window of length $T$ is used to construct a one-step-ahead prediction task: the network is provided with RR values $\{x_1, \ldots, x_T\}$ and trained to predict the subsequent values $\{x_2, \ldots, x_{T+1}\}$. This framing allows the recurrent network to be interpreted as learning a local update rule governing the evolution of the signal.

### *5.2 Model Architecture*

The model employed is a minimal *vanilla* recurrent neural network constructed explicitly from first principles. The network consists of a single hidden layer containing $n = 16$ hidden units, chosen to balance simplicity with sufficient expressive capacity. The network is trained on fixed-length sequences of $T = 50$ time steps, so that the recurrence is applied repeatedly over a finite temporal window. At each time step, the hidden state evolves according to

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h),$$

where $h_t \in R^{16}$ denotes the hidden state, $x_t \in R$ is the scalar HRV input, and $W_{hh}$, $W_{xh}$, and $b_h$ are learned parameters.

The network output is given by a linear readout

$$y_t = W_{hy}h_t + b_y,$$

which produces a scalar prediction of the next RR value. The use of the hyperbolic tangent activation ensures bounded hidden states and introduces nonlinearity into the recurrence, while simultaneously preserving symmetry about the origin.

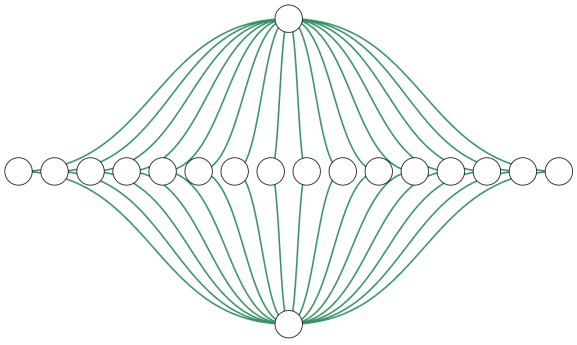

**Figure 4.** Dimensionality of the employed Neural net architecture encapsulating a single time step $h_t$

Importantly (to ensure the point has been sufficiently ingrained), this architecture defines a discrete-time nonlinear dynamical system in hidden-state space. The weight matrices determine how information is propagated and mixed across time, while the activation function shapes local stability and sensitivity. Over a sequence of length $T = 50$, perturbations to the hidden state, as well as gradients during learning, propagate through repeated application of the local Jacobian

$$J_t = \frac{\partial h_t}{\partial h_{t-1}},$$

so that sensitivities accumulated during backpropagation through time take the form

$$\frac{\partial \mathcal{L}}{\partial h_0} = J_{t+1}^{\top} J_{t+2}^{\top} \cdots J_{50}^{\top} \frac{\partial \mathcal{L}}{\partial h_{50}}.$$

### *5.3 Training Procedure and Loss Dynamics*

For simplicity purposes, training is performed using mean squared error loss, measuring the discrepancy between the

predicted RR values and the observed next-step values across each window. Parameter updates are computed via explicit backpropagation through time, allowing gradients to flow backward along the unfolded recurrence.

Although the optimization procedure minimizes prediction error, the primary objective is not predictive optimality. Rather, training is viewed as a process that gradually reshapes the parameters governing the recurrence relation, thereby modifying the underlying hidden-state dynamics.

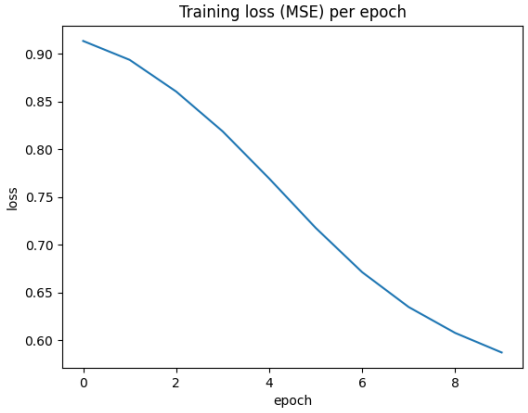


**Figure 5.** Training loss (mean squared error) as a function of epoch. The monotonic decrease indicates stable learning dynamics in the parameter space.

The observed monotonic decrease in training loss is indicative of stable gradient propagation and provides empirical evidence that the learned update rule is well behaved over the training window.

### 5.4 Qualitative Prediction Behavior

To assess the learned dynamics, the trained network is evaluated on a representative HRV window. The predicted RR sequence is compared to the observed sequence on a step-by-step basis. Because the model is trained under teacher forcing, this comparison probes whether the learned recurrence produces locally consistent updates when conditioned on true past values.

The predictions track the general rhythm and variability of the signal while exhibiting a degree of smoothing. This behavior is consistent with a recurrence operating in a stable, contractive regime, where perturbations neither grow uncontrollably nor vanish immediately.

### 5.5 State-Space Geometry and Poincaré Analysis

Additional insight is provided by examining the geometry of the HRV signal in a low-dimensional embedding. The RR–$RR_{n-1}$ (Poincaré) plot represents a two-dimensional delay embedding of the time series, in which each point corresponds to a pair of successive beat-to-beat intervals. This construction provides a geometric view of short-term variability and correlation between consecutive heartbeats, revealing structure that is not immediately apparent in the time-domain signal alone.

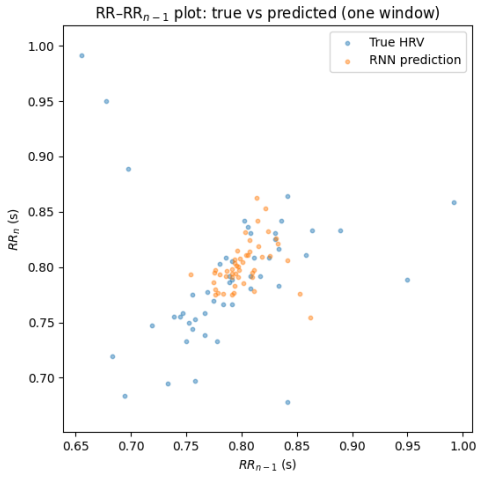


**Figure 6.** Observed and predicted RR intervals over a single window. The network captures local temporal structure while remaining stable over the prediction horizon.

From a dynamical systems perspective, the Poincaré plot may be interpreted as a projection of the underlying state-space trajectory onto a low-dimensional subspace. The shape, dispersion, and orientation of the resulting point cloud reflect properties of the system's dynamics, such as regularity, variability, and the presence of feedback-driven structure.

### 5.6 Local Stability and Spectral Radius Along Learned Trajectories

To connect the theoretical discussion of stability with the trained model, the *spectral radius* of the local Jacobian was evaluated along a representative hidden-state trajectory. At each time step $t$, the Jacobian was computed. Here, the spectral radius $\rho(J_t)$ is defined as $\rho(J_t) = \max_i |\lambda_i(J_t)|$.

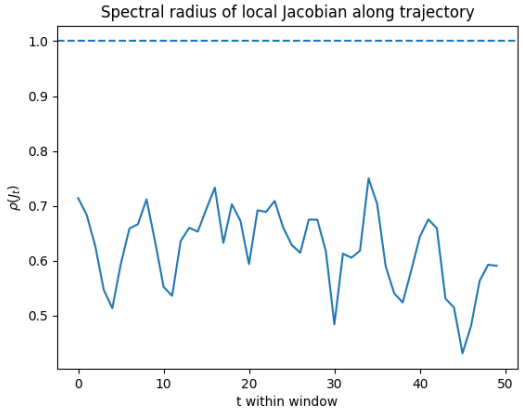


**Figure 7.** Spectral radius of the local Jacobian $J_t$ along a representative HRV window. The dashed line indicates the stability threshold $\rho = 1$.

As shown in Figure 7, the spectral radius remains below unity throughout the window, placing the learned dynamics in a desirable, *mildly contractive* regime. This implies that small

perturbations to the hidden state decay over time, ensuring stability of the recurrence. At the same time, spectral radii approaching unity indicate that information is retained over multiple time steps rather than being immediately forgotten.

This behavior provides a dynamical explanation for the qualitative prediction results observed earlier. The smoothing of fluctuations in the predicted HRV signal is consistent with contractive dynamics, while the absence of rapid collapse is indicative of operation near, but below, the stability boundary. In this way, the spectral radius plot offers a concrete link between eigenvalue analysis, observed prediction behavior, and the underlying learned recurrence.

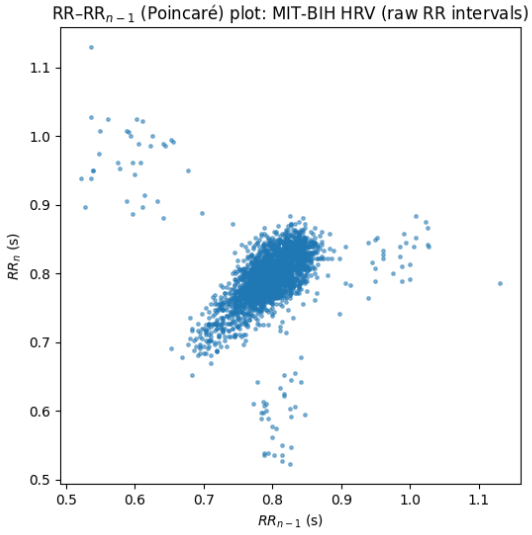


**Figure 8.** RR–RR$_{n-1}$ (Poincaré) plot constructed from the observed HRV signal.

A corresponding plot constructed from the network's predictions reveals qualitative similarity in geometric structure. This agreement suggests that the recurrent network has internalized aspects of the signal's dynamical structure, rather than simply reproducing values.

## 6. Conclusion

This work has examined recurrent neural networks through the lens of discrete-time nonlinear dynamical systems, with the aim of understanding their internal mechanisms rather than optimizing predictive performance. By interpreting neural networks as iterated state-space maps, concepts like stability, memory, and sensitivity acquire precise mathematical meaning, allowing tools from classical dynamical systems theory to be applied directly to learning-based models.

A minimal vanilla recurrent neural network was constructed explicitly from first principles and applied to heart rate variability data derived from electrocardiographic recordings. Despite its simplicity, the model exhibits several behaviors predicted by the theoretical framework developed earlier in the report. Training dynamics in parameter space are stable, as reflected by

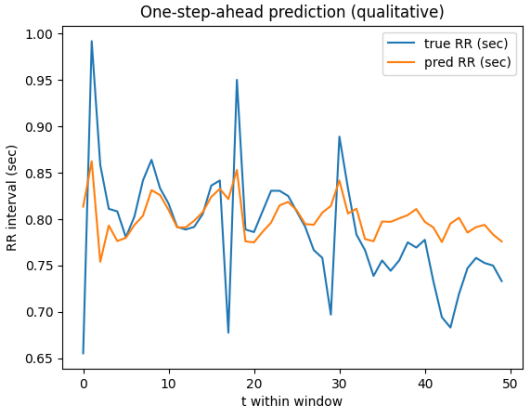


**Figure 9.** Comparison of RR–RR$_{n-1}$ plots for observed and predicted HRV, illustrating similarity in learned state-space geometry.

the monotonic decrease in loss, while qualitative time-domain predictions demonstrate that the learned recurrence captures local temporal structure without instability. State-space analysis via Poincaré plots further shows that the network leads us to a geometry sufficiently qualitatively similar to that of the underlying biological signal.

Crucially, Jacobian-based analysis of the hidden-state dynamics provides a concrete link between theory and implementation. The measured spectral radii of the local Jacobians remain below unity along representative trajectories, placing the learned system in a contractive regime. This observation explains both the stability of the hidden-state evolution and the smoothing behavior observed in the predictions, illustrating how eigenvalue analysis offers direct insight into memory and gradient propagation in recurrent networks.

Several limitations of the present study should be acknowledged. The recurrent architecture considered is intentionally minimal and does not incorporate gating mechanisms or architectural modifications designed to extend memory or stabilize training. Furthermore, the discrete-time formulation abstracts away continuous-time physiological dynamics underlying cardiac regulation. Nonetheless, these simplifications are deliberate: they allow the essential dynamical structure of recurrent networks to be exposed without obscuring it behind architectural complexity.

Taken together, this work demonstrates that recurrent neural networks need not be treated as opaque black-box predictors. Instead, they may be demystified and understood as structured dynamical systems whose behavior can be analyzed, interpreted, and reasoned about using familiar analytical tools. Viewing learning as a process that reshapes state-space dynamics provides a principled foundation for understanding existing recurrent architectures and suggests a pathway toward developing future models grounded in dynamical intuition rather than purely empirical design.

**Acknowledgments**

I would like to express my deepest and abiding gratitude and reverence for my advisor and professor, Ruth Corran, whose guidance and instruction have played a central role in my mathematical development. Her emphasis on clarity, rigor, and conceptual understanding has shaped the way I think about mathematics and its applications, and has provided the foundation necessary to approach a project of this scope.

Much of the mathematical literacy and confidence required to engage with nonlinear dynamical systems, learning theory, and their intersection was cultivated through her teaching and mentorship. This work is a direct reflection of that influence, and I am sincerely grateful for the intellectual discipline and encouragement she has provided throughout my studies.

**References**

[1] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, 2nd ed., Westview Press, Boulder, CO, 2015.

[2] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., Pearson Education, Upper Saddle River, NJ, 2009.

[3] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[4] GeeksforGeeks, "Introduction to Recurrent Neural Network," available at: https://www.geeksforgeeks.org/machine-learning/introduction-to-recurrent-neural-network/. Accessed 2025.

[5] Task Force of the European Society of Cardiology and the North American Society of Pacing and Electrophysiology, "Heart Rate Variability: Standards of Measurement, Physiological Interpretation and Clinical Use," *Circulation*, vol. 93, no. 5, pp. 1043–1065, 1996.

[6] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[7] PhysioNet, "MIT-BIH Arrhythmia Database (mitdb), version 1.0.0," available at: https://www.physionet.org/content/mitdb/1.0.0/. Accessed 2025.

[8] ProtoBioEngineering, *Working with MIT-BIH ECG Data in Python, Part 2: Visualizing the Heartbeats with Matplotlib*. Medium, 2018. Available at: https://medium.com/@protobioengineering/working-with-mit-bih-ecg-data-in-python-part-2. Accessed January 2026.