DEPARTMENT OF DIGITAL SYSTEMS

NCSR DEMOKRITOS
INSTITUTE OF INFORMATICS AND
TELECOMMUNICATIONS

Analyzing Long-Term Memory in Sequential Models Through Internal State Dynamics

by

Christoforos Romesis

Submitted

in partial fulfilment of the requirements for the degree of

Master of Artificial Intelligence

at the

UNIVERSITY OF PIRAEUS

February 2026

Author . . . . . .

Christoforos Romesis
II-MSc “Artificial Intelligence”
February 26, 2026

Certified by. . . . . .

Stasinos Konstantopoulos
Research Associate, NCSR “Demokritos”
Thesis Supervisor

Certified by. . . . . .

Theodoros Giannakopoulos
Grade A Researcher, NCSR “Demokritos”
Member of Examination Committee

Certified by. . . . . .

Theodore Dalamagas
Grade A Researcher, “Athena” Research Center
Member of Examination Committee

Analyzing Long-Term Memory in Sequential Models Through Internal State Dynamics

By

Christoforos Romesis

Submitted to the II-MSc "Artificial Intelligence" on
February 2026,
in partial fulfillment of the
requirements for the MSc degree

Abstract

This thesis investigates the mechanisms governing long-term memory retention in sequential neural models, adopting a dynamical systems perspective on recurrent architectures. The study is conducted on fault-driven time-series data generated from a high-fidelity mathematical model of an aircraft engine, designed to include subtle intermediate events that are essential for correct fault discrimination despite identical final observable behavior. Recurrent models are trained and systematically analyzed beyond output-level accuracy, with emphasis on their internal state dynamics. While classification performance remains consistently high, memory retention exhibits non-monotonic dependence on sequence length, revealing alternating regimes of successful and degraded historical encoding. Dimensionality reduction of internal state trajectories exposes structured geometric organization in the learned state space, including attractor basins and rotational dynamics. The results suggest that long-term memory retention is not adequately explained by classical vanishing or exploding gradient arguments alone. Instead, it emerges as a geometric property of the internal dynamics, shaped by phase-dependent recurrent interactions and the effective training horizon. The thesis introduces the concept of memory crackpoints, referring to critical dynamical transitions where history-dependent structure collapses, leading to qualitative memory loss without apparent degradation in short-term predictive performance. These findings highlight the importance of internal dynamical analysis for diagnosing and improving memory mechanisms in sequential neural models.

Thesis Supervisor: Stasinos Konstantopoulos
Title: Research Associate, NCSR "Demokritos"

Acknowledgments

I would like to express my deepest gratitude to my family, and especially to Miranda and Nikolas, for their unwavering support, patience, and encouragement throughout this journey. I am also sincerely grateful to my colleagues and to my supervisor at NCSR Demokritos for their guidance, insight, and continuous support.
Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of University of Piraeus and Inst. of Informatics and Telecom. of NCSR “Demokritos”.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Sequential data is widespread in many practical domains. Speech signals, text, sensor measurements, and system logs evolve over time and can rarely be understood in isolation from past observations. Machine learning models applied to such problems cannot assume that individual observations are conditionally independent. Rather, they need mechanisms to carry information across time. Sequential neural models were created for this reason. Recurrent Neural Networks (RNNs) and their gated variants, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), maintain an internal state that is updated at every timestep of the input sequence. This internal state can accumulate information from the past, and for this reason, its behavior is often referred to as memory.

However, memory in sequential models can be fragile and poorly understood. Experiments show that memory retention can vary significantly across training conditions, sequence length, and internal dynamics. This is observed even when keeping network architectures and hyperparameters constant. This is highly counter-intuitive in situations where a model is asked to remember an important piece of information that occurs briefly and needs to be stored for a long time, regardless of what comes afterwards.

This is often critical in safety-relevant applications. In industrial anomaly diagnosis, it is not uncommon for two different system faults to exhibit exactly the same behavior for most of the time window we are interested in. The only difference is a brief event occurring earlier in the time sequence. Failing to remember this event will lead to a highly confident wrong prediction. Large time-wise classification accuracies hide the fact that models do not steadily store information about the critical event that defines the true class.

Benchmarking sequential models is typically done at the output level. Accuracy, precision, F1-scores are mandatory to measure before deploying a model. But they do not allow us to understand the underlying reasons for failures in particular examples. In the aforementioned scenario, where two classes can only be distinguished by a transient intermediate event, forgetting this event leads to a targeted drop in performance. Models might still be able to correctly classify every individual time step, but fail at the sequence-level task. These are, therefore, not issues with classification in general, but of memory. Motivated by these deficiencies, we aim to better understand how memory emerges, is stored, and can disappear in neural sequence models. Instead of conceptualizing memory loss as a performance decay, we study if there are scenarios in which memory can abruptly fail depending on evolving internal patterns.

1.2 Research Problem and Objectives

The guiding question of this thesis is straightforward: *What does it mean for a sequential model to retain memory?* Memory failure is often translated into prediction failure. A model is expected to learn dependencies given enough data and capacity. Failure to learn these depen-

dencies is diagnosed through poor task accuracy. This intuition can be misleading, though. Consider sequence-learning tasks where local accuracy can be high but long-term dependencies are required for correct final outputs. Models may learn to react to local inputs accurately, while forgetting past inputs required for the final task decision. Memory failure does not necessarily correlate with sudden accuracy drops and may go undetected by traditional metrics.

This thesis investigates the conditions under which information about a transient but task-critical event is forgotten by sequential models, leading to selective failure. We focus on cases where models accurately classify every timestep but fail to differentiate between fault types that diverge only at earlier timesteps. We hypothesize that memory retention is neither gradual nor monotonic in model parameters. Rather, it depends on the match between the learned internal dynamics and other training parameters, such as sequence length and the training horizon. Memory retention can depend on both the total sequence length T, which dictates the forward timescale of the task, and the effective train horizon H, which dictates how far back gradients flow. Although these quantities coincide under full backpropagation through time, they represent conceptually distinct aspects of sequential learning. Memory can abruptly change due to any of these quantities. In this thesis, we introduce the term memory-crackpoints to refer to transitional regimes of successful and failed memory retention, for a fixed model architecture. In Chapter 3, we give a formal interpretation of this phenomenon.

Motivated by this definition, this thesis has three main goals. First, we construct a task where long-term memory failures can be isolated and consistently reproduced. Second, we analyze internal model dynamics, such as hidden states, cell states, and gate activations to visualize how stored representations change over time. Third, we use the framework of dynamical systems to interpret memory retention and failure.

1.3 Research Questions

The questions we address in this thesis are:

1. What information from the past is encoded in the internal states of sequential models? Here, we are primarily concerned with the hidden states and cell states as ways of remembering information that is not visible at the output.

2. How and when do sequential models lose memory? Specifically, we try to identify circumstances under which memory is lost without any obvious evidence of task failure.

3. How does the memory behavior of different recurrent architectures differ? Here we train RNNs, GRUs, and LSTMs in the same setting to see if the qualitative behavior of their memories differs.

4. Is loss of memory always accompanied by vanishing or exploding gradients? Motivated by classical explanations of failure to capture long-term dependencies, this question will lead us to new ways of thinking about memory based on internal dynamics.

5. Can we diagnose memory failures invisible to output metrics with tools from the analysis of internal state-spaces?

6. Can tools from geometry and dynamics help us understand when and how memory fails?

1.4 Contributions of This Thesis

The primary contributions of this work are listed below:

- A realistic synthetic fault time series dataset is generated from a high-fidelity mathematical gas-turbine engine model to expose problems related to long-term dependencies. The dataset is made publicly available to support further research.
- Custom implementations of popular recurrent architectures (RNN, GRU, LSTM) with full visibility into internal states are provided to allow for systematic logging of hidden states, cell states, and gating activations at each time step.
- It is demonstrated empirically that memory is non-monotonic. Classification performance with respect to fault discrimination degrades and improves as a function of sequence length, even with high time-wise classification accuracy.
- Memory crackpoints, that capture abrupt transitions in memory retention not reflected in standard performance metrics, are formally defined and experimentally shown.
- A dynamical explanation of memory retention and failure is offered in terms of the geometry and stability of the learned state space.

1.5 Structure of the Thesis

The rest of this thesis is structured as follows.

Chapter 2 covers background and related work. This chapter surveys sequential modeling and long-term dependencies, and points out limitations of prior work.

Chapter 3 presents the theoretical framework that views sequential models as dynamical systems and introduces the concept of memory-crackpoints.

Chapter 4 describes the experimental framework, including dataset construction, model implementations, and evaluation metrics.

Chapter 5 presents our experimental results. We analyze performance trends, gate behavior, and internal state-space geometry.

Chapter 6 synthesizes the theoretical and experimental findings, discusses implications and limitations, and outlines directions for future research.

2 Background and Related Work

2.1 Sequential Data and the Nature of Temporal Dependencies

Unlike static data, sequential data are temporally dependent: observations come in a sequence ordered in time, and often the meaning of a particular observation changes depending on previous observations. Speech signals, text, financial time series, physiology, and sensor readings from complex engineering systems are just a few examples of sequential data.

Temporal dependencies can vary significantly in structure and difficulty. For some tasks, all the information required to make a decision may remain available at all times in the input stream, leaving the model to rely mostly on short-term correlations between past and future observations. For other tasks, however, information may arrive infrequently, temporarily, or with a delay relative to when a decision needs to be made, forcing the model to memorize events that could have occurred arbitrarily long ago [1, 2]. Crucially, however, arbitrarily long sequences are not what make these tasks hard per se; rather, it is how and where information is presented over time that makes them challenging.

It is this difference that has led to the decoupling between local temporal processing and long-term dependency retention. A model can be very good at responding to short-term patterns while still showing sensitivity to how it encodes and reinforces information over longer time scales. Especially when critical information is no longer present in the input stream, successfully discriminating between two or more classes depends on whether the model's internal dynamics have captured the effect of past observations. As such, long-term dependencies should not simply be thought of with respect to how far away they are in time, but rather as a property of the internal state of the relevant models.

2.2 Classical Approaches to Temporal Modeling

Prior to neural networks, many sequential models were explicitly statistical: autoregressive (AR), moving average (MA), state-space models, etc. [3, 2]. They model temporal structure through explicitly encoded transition dynamics and noise models.

For example, state-space models make an explicit distinction between the evolution of a latent state and the observation process that maps this latent state to measurable outputs. State-space models tend to be very interpretable, but typically assume linear dynamics or require strong prior assumptions to remain tractable. As a result, they lack the expressiveness of complex nonlinear systems.

Neural sequential models learn transition dynamics from data. This allows neural models to learn arbitrary temporal structure, but hides the inner workings of how memory is encoded.

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are one of the first sequence models explicitly designed for sequence modeling [4]. At every timestep, the hidden state of an RNN is updated using some function of the previous hidden state and the current input. Because of these recurrent connections, information propagates forward through time.

Vanilla RNNs, however, suffer from the inability to learn long-term dependencies [1]. Due to vanishing/exploding gradients issues, during Back Propagation Through Time, RNNs are not efficient at learning relationships between distant events. Empirically, RNNs tend to model short-term dependencies. We can think of the hidden state as a summary of all previous inputs into the network. Since we've provided no explicit way to control what information is stored in the hidden state and what is forgotten, the compression is not selective. This problem can be somewhat alleviated by careful initialization [5], gradient clipping [6], and architectural constraints [7]. Although these methods improve training stability, they do not eliminate the fundamental difficulty of vanilla RNNs in preserving long-term information through repeated state updates [6].

2.4 Long Short-Term Memory Networks

Long short-term memory networks (LSTM) were developed as a solution to vanilla RNNs inability to learn long-term dependencies [2]. The primary characteristic that distinguishes LSTM's is a separation of the hidden state and an independent cell state to "learn" to store long-term information. Information is written to, erased from, and read from the cell state through gating mechanisms (input gate, forget gate, and output gate, respectively) [2].

LSTMs have been applied to many tasks that require learning long-term dependencies and performed well empirically on a variety of problems [8]. From an optimization standpoint, the cell state acts as a highway along which gradients can flow with less decay, alleviating the vanishing gradient problem [2, 6].

However, gating mechanisms do not guarantee stable memory retention across task regimes. How well an LSTM remembers depends on how the gates are trained. Empirical evidence has shown that LSTM's act like a short to medium range context, unless task structure explicitly reinforces longer term dependencies [9, 8].

2.5 Gated Recurrent Units

Gated Recurrent Units (GRU) provide an alternative architecture that simplifies the LSTM [10]. Instead of an input gate and a forget gate, GRUs have an update gate. There is no separate cell state; instead, memory content is stored in the hidden state without an associated cell-state track [10, 11].

The reduced number of gates typically means there are fewer parameters, and often training can be faster. Empirical studies have shown that GRUs achieve performance comparable to

LSTMs on a variety of sequence learning tasks [11].

The downside to this merging of gate functionality is that the evolution of the state and memory storage is more tightly coupled. There is less explicit separation of structure. Similar to other recurrent architectures, long-term memory in GRUs also depends on learned recurrent dynamics and time scales, beyond just the gating mechanism [9].

2.6 The Long-Term Dependency Problem

The difficulty of learning long-term dependencies has historically been investigated in terms of the vanishing and exploding gradient problem [1, 6]. During Backpropagation Through Time (BPTT), gradients involve repeated multiplication by Jacobian matrices of the recurrent transition. Depending on the spectral properties of these Jacobians, gradients either vanish or explode exponentially with the temporal separation.

While insightful for understanding the optimization process, this perspective has inspired workarounds such as gating mechanisms, but tells little about how representations themselves are encoded in state space.

In addition, gradient-based understandings generally describe memory difficulty as an exponentially decaying/enhancing phenomenon, implicitly assuming monotonic memory decay as a function of distance. Recent work has shown that memory horizons depend on learned timescales and spectral structure and can change non-monotonically with network and training details [7, 9].

2.7 Effective Memory Horizons and Capacity Measures

Several studies attempt to quantify memory through the concept of memory capacity or effective memory horizon [12, 13]. Memory capacity attempts to measure how much the past can affect the present output/state.

Metrics like memory capacity are helpful for benchmarking, but they shrink memory to a scalar or low-dimensional measurement. Networks with similar memory capacities may behave qualitatively differently in the presence of structured or transient temporal dependencies.

Capacity-based metrics don't explicitly describe how memory traces are stored in the geometry of the hidden space, nor how learned dynamics determine the decay of specific perturbations [14, 9].

2.8 Methods for Analyzing Memory in Sequential Models

Numerous techniques for understanding memory in recurrent networks have been proposed. Analyses of gate activation behavior examine the behavior of the gates in architectures such as LSTMs, aiming to determine whether information is actively preserved or discarded [15, 8]. These analyses, although informative, require aggregation of model behavior over time, where short-lived or trajectory-dependent effects may be missed.

Probing classifiers attempt to predict previous inputs based on hidden states [16, 17]. These methods quantify whether information can be retrieved from the representation, but they do not formally characterize the underlying dynamics that maintain information or cause it to decay.

State-space visualization techniques attempt to understand the geometry of hidden state space by projecting high-dimensional hidden states into lower-dimensional spaces [18]. These methods allow researchers to examine trajectories, clustering behavior, and transitions between internal regimes. These methods tend not to interface with a corresponding analysis of dynamical stability and attractor structures.

Influence-based methods, on the other hand, quantify the contribution of specific inputs to model outputs [19]. Although valuable for interpretability, such approaches focus on input–output relationships rather than on the internal organization of the recurrent state space.

2.9 Dynamical and Spectral Perspectives on Recurrent Models

Recent work views recurrent neural networks as dynamical systems, with an emphasis on the geometry and stability of the internal state space evolution over time [18]. Memory retention is linked to stability characteristics of the recurrent dynamics as well as spectral properties of the associated Jacobian matrices. Spectral analyses emphasize the role of eigenvalue magnitudes in determining stability and gradient propagation [6, 20]. Principles like dynamical isometry attempt to normalize Jacobian singular values to preserve stability during training [21].

While instructive about stability, these methods largely focus on norms/singular values. Structure in phase/mode coupling and its relationship to horizon are less explicitly treated.

2.10 Limitations of Existing Work

Despite extensive research, current methods for memory analysis of sequential models remain incomplete. While many existing works focus on optimization stability or on output-level performance, most do not explicitly consider the underlying dynamical processes of memory retention.

Oscillatory and phase-dependent dynamics can be investigated within constrained models such as unitary/orthogonal recurrent networks, but are not typically considered in general memory analysis frameworks [7, 22].

Therefore, existing viewpoints leave gaps in understanding non-monotonic memory retention, regime-dependent memories or sudden qualitative changes in internal dynamics.

2.11 Summary and Research Gap

This chapter provided an overview of the main architectural, optimization-based, and analytical techniques used to study memory in sequential neural models. Although great progress has been made towards stabilizing training and measuring memory capacity, current theories do not account for why memory retention can differ across regimes, nor for why qualitative changes in internal dynamics emerge as a function of training conditions.

To address these issues we instead view recurrent networks as learned dynamical systems, framing memory capacity in terms of state-space geometry, stability structure, and phase-dependent dynamics. We develop this theory in the following chapter together with the mathematical groundwork for the experimental investigations that follow.

3 Theoretical Framework: Memory Dynamics in Sequential Models

3.1 Sequential Neural Models as Discrete-Time Dynamical Systems

Models like RNNs, GRUs, and LSTMs take a sequence of data as input and process it by iteratively updating an internal state. At each time step, the model maintains a hidden state $h_t \in \mathbb{R}^d$, which summarizes past information and is updated based on the current input $x_t \in \mathbb{R}^m$.

Formally, the state update can be written as

$$h_{t+1} = F(h_t, x_t; \theta) \tag{3.1}$$

where θ denotes the learned parameters and F is a nonlinear transition function determined by the architecture.

This recursive formulation resemples in structure to a discrete-time dynamical system [18, 23], where a state evolves step-by-step according to a deterministic transition rule. In general, a discrete-time dynamical system takes the form

$$z_{t+1} = G(z_t) \tag{3.2}$$

or, in the presence of external input,

$$z_{t+1} = G(z_t, u_t) \tag{3.3}$$

The recurrent update equation of sequential neural models matches this structure exactly, where h_t acts as the system state and x_t as the external input. The evolution of the hidden state, therefore, defines a trajectory in the d-dimensional state space $\mathbb{R}^d$.

This perspective of viewing recurrent models as dynamical systems allows us to rephrase questions about these models not in terms of their output predictions, but rather in terms of the geometry of their internal-state evolution. Questions about memory can be reframed as how perturbations to the system's state (caused by transient inputs) propagate forward in time. If two trajectories that diverge due to an event in the past continue to access distinct regions of state space, then we say the system remembers that event. On the other hand, if two trajectories converge to the same region of state space, then we say the information about that event has been lost.

Crucially, this perspective does not depend on the presence of an output layer. The output $y_t = G(h_t)$ is a function of the state, but the memory properties of the system are determined by the internal transition dynamics F, not by the readout mapping.

With this dynamical systems perspective in mind, we can now proceed into our theoretical analysis.

3.2 Autonomous Dynamics Under Stationary Inputs

In practice, sequential tasks often involve input sequences that display long periods of quasi-stationarity. For instance, consider fault-diagnosis tasks in which the system could operate in a "healthy" regime for extended periods. In such a case, the observation at each time step would be nearly identical.

When the input becomes approximately constant over an interval,

$$x_t \approx x, \quad \forall t \in [t_0, t_1] \tag{3.4}$$

the recurrent update equation in Section 3.1

$$h_{t+1} = F(h_t, x_t; \theta) \tag{3.5}$$

reduces to the autonomous map

$$h_{t+1} = F_x(h_t) \tag{3.6}$$

where

$$F_x(h) := F(h, x; \theta) \tag{3.7}$$

This equation is known as an autonomous discrete-time dynamical system. Explicit dependence on time-varying external input does not appear in this form. Therefore, asymptotic behavior of this system is completely characterized by the transition function F_x [24].

The sequence of states

$$\{h_{t_0}, h_{t_0+1}, \ldots, h_{t_1}\} \tag{3.8}$$

generated by repeated application of F_x is referred to as a trajectory in the state space. Each initial condition h_{t_0} defines a trajectory, and different initial conditions may lead to different long-term behaviors.

The regime is the key to a dynamic perspective on memory. If a brief perturbation occurred prior to t_0, representing some past event, it takes the internal state away from its original position to some new initial condition. Will the system still feel the effect of that perturbation some time later? This depends on how the trajectory corresponding to that initial condition evolves under the autonomous dynamics. If two trajectories that correspond to two different past events do not converge in state space, memory has been preserved. If they do converge to the same area, then the information that differentiated them is lost.

Memory, therefore, can be analyzed in recurrent models based on the properties of the autonomous dynamics that occur after some input perturbs the system, independent of performance or output.

3.3 Fixed Points, Linearization, and the Jacobian

A central concept in the analysis of dynamical systems is the fixed point. In the autonomous regime introduced in Section 3.2, the state evolves according to

$$h_{t+1} = F_x(h_t) \tag{3.9}$$

A point $h^* \in \mathbb{R}^d$ is called a *fixed point* if it satisfies

$$h^* = F_x(h^*) \tag{3.10}$$

Conceptually, a fixed point is an unchanged state under the dynamics of the system. Once the system reaches h^* it will stay there indefinitely, unless perturbed. In our recurrent network interpretation, a fixed point may represent a stationary internal representation corresponding to a persistent input regime, such as a prolonged "healthy" operating condition. But having fixed points isn't enough for memory storage — it is the dynamics around fixed points that matter. Consider a small perturbation around the state:

$$\delta h_t = h_t - h^* \tag{3.11}$$

This perturbation may arise, for example, from a transient intermediate event that shifts the internal state away from equilibrium. The key question is whether this deviation grows, decays, or persists over time.

To analyze this behavior, the dynamics are linearized around h^*. A first-order Taylor expansion yields

$$\delta h_{t+1} \approx J_x(h^*)\, \delta h_t \tag{3.12}$$

where

$$J_x(h^*) = \left.\frac{\partial F_x}{\partial h}\right|_{h=h^*} \tag{3.13}$$

is the Jacobian matrix of the autonomous recurrent dynamics evaluated at the fixed point.

The Jacobian captures how small perturbations are transformed by one step of the dynamics. The evolution of the perturbation is therefore approximately governed by repeated multiplication by $J_x(h^*)$.

The eigenvalues λ_i of the Jacobian determine the local behavior of perturbations, meaning the behavior of sufficiently small deviations from the fixed point. Specifically:

- If all eigenvalues satisfy

$$|\lambda_i| < 1 \tag{3.14}$$

then perturbations decay exponentially with time. The fixed point is said to be locally stable (or asymptotically stable). Small deviations shrink, and trajectories return to h^*.

- If at least one eigenvalue satisfies

$$|\lambda_i| > 1 \tag{3.15}$$

perturbations grow exponentially along the corresponding direction. The fixed point is unstable, as nearby trajectories diverge away from it.

- If one or more eigenvalues satisfy

$$|\lambda_i| = 1 \tag{3.16}$$

the system has marginal stability along those dimensions. Perturbations along those directions neither decay nor grow exponentially. Whether they fade or not depends on higher-order non-linear effects.

We say a dynamical system is locally stable at a fixed point if arbitrarily small perturbations from that fixed point remain bounded and asymptotically return to the fixed point. If arbitrarily small perturbations can force the state to run away from the fixed point then it is locally unstable.

Local stability is directly related to memory. If all directions around the fixed point are strongly contracting ($|\lambda_i| \ll 1$), then the system quickly forgets small perturbations due to transient events. If some directions are close to marginal stability ($|\lambda_i| \approx 1$), then perturbations can survive for long timescales, allowing the system to remember.

This linearized analysis has been used to understand when and why recurrent networks can preserve or erase the effects of transient events [24, 25], depending on the spectral structure of their learned dynamics.

3.4 Mixed Stability and High-Dimensional Recurrent Dynamics

From the linearized analysis in Section 3.3, the local behavior of perturbations around a fixed point is controlled by the eigenvalues of the Jacobian matrix $J_x(h^*)$. In discrete-time dynamical systems, stability is defined with respect to the unit circle in the complex plane: eigenvalues inside the unit circle imply decay, while eigenvalues outside the unit circle imply growth.

For high-dimensional recurrent networks, the Jacobian almost always has eigenvalues with a range of magnitudes. It is rare for all eigenvalues to fall strictly inside or outside the unit circle [18, 20]. Instead, learned recurrent dynamics typically demonstrate mixed stability: different components of a perturbation decay/grow at different rates.

Consider concretely decomposing a small perturbation δh_t into eigenvector components of the Jacobian. Each component will then evolve approximately according to

$$\delta h_{t+1}^{(i)} \approx \lambda_i \, \delta h_t^{(i)} \tag{3.17}$$

where λ_i is the corresponding eigenvalue.

This implies:

- If $|\lambda_i| < 1$, the perturbation component shrinks over time (contracting mode).
- If $|\lambda_i| > 1$, the component grows (expanding mode).
- If $|\lambda_i| \approx 1$, the component persists over extended time scales (marginal mode).

A dynamical mode corresponds to an eigenvalue–eigenvector pair of the Jacobian, and indicates an independent direction along which the local state evolves.

If some eigenvalues are inside the unit circle while others are outside, the fixed point is a saddle: trajectories will converge to it along some directions and diverge along others.

Mixed spectra like these are very common in recurrent neural networks. Neural networks have learned weight matrices that are typically non-symmetric and high-dimensional, and their overall transition functions are nonlinear. Non-symmetric (nonnormal) matrices tend to have eigenvalues scattered throughout the complex plane, often with non-orthogonal eigenvectors [26]. Nonlinearities in the activation functions or gating mechanisms can further weight the local Jacobian depending on the operating point. For these reasons, neural network dynamics have state-dependent stability properties.

The result is that the state space has a heterogeneous stability structure: some directions in state space have qualitatively different dynamics than others. If the network experiences some transient perturbation, some components of the state vector will cause the response to quickly decay back to the fixed point, while others will cause it to persist.

This heterogeneity in stability is central to how memory actually works. Memory isn't stored uniformly across all state dimensions. Instead, memory is stored along those directions / subspaces which correspond to eigenvalues near the unit circle, so that perturbations neither decay nor grow too quickly. Directions that strongly contract determine what the network forgets, while marginally stable directions provide a basis for learned internal representations.

This relationship between heterogeneous stability structure and memory capacity is key to understanding the mechanism by which recurrent models can selectively remember task-relevant signals while forgetting irrelevant input perturbations.

3.5 Attractors, Manifolds, and Basins of Attraction

The local linear analysis of Section 3.3 describes the behavior of trajectories in a small neighborhood of a fixed point. However, the overall behavior of nonlinear dynamical systems is often governed by more general invariant structures.

In the autonomous regime introduced in Section 3.2, where the input is approximately stationary, and the system is described by

$$h_{t+1} = F_x(h_t) \tag{3.18}$$

the long-term behavior of trajectories is determined entirely by the internal dynamics. This setting allows the application of classical concepts from nonlinear dynamical systems theory [24].

An attractor is a set $A \subset \mathbb{R}^d$ where trajectories initialized sufficiently close to A converge toward it under repeated application of the dynamics. Unlike an unstable fixed point, which repels nearby trajectories, an attractor draws them in over time.

Attractors are not necessarily isolated fixed points. In general, attractors of nonlinear systems can be:

- stable fixed points,
- low-dimensional invariant manifolds,
- limit cycles (closed orbits),
- or more complicated invariant sets.

A basin of attraction is the set of asymptotic trajectories that converge to an attractor. Basin structure plays an important role when multiple attractors are present. Networks with sufficiently high dimensionality can support multiple attractors, in which case the state space gets partitioned into basins of attraction. Different basins correspond to alternative long-term dynamical behaviors of the system.

The partitioning of state space into basins provides a natural definition of memory. A transient external event takes the system's internal state to some initial condition. This disturbance shifts the trajectory to another basin of attraction. Even if the external input eventually returns to the autonomous regime, the dynamics will drive the internal state to a different asymptotic value.

On the other hand, if nearby initial conditions share the same basin of attraction, then the effects of the perturbation will eventually be erased. Eventually, the system will return to its attractor, and the memory of the perturbation will be lost.

Retention of information in the autonomous regime is equivalent to stability of basin separation when the input is stationary. The basin geometry determines whether past events lead to persistent internal differences or are forgotten.

3.6 Memory as Geometry in State Space

The previous sections demonstrated that, when inputs are constant, recurrent neural networks are autonomous discrete-time dynamical systems whose asymptotic dynamics are characterized by fixed points, invariant sets, and basins of attraction. Given this perspective, we may interpret memory geometrically.

Memory is not stored in any particular neuron, gate, or scalar activation value. Memory is the separation between trajectories in state space. A transient event slightly perturbs the network's internal state, leading to a new initial condition for the autonomous system. Remembering that event depends critically on the geometry of trajectories relative to basins of attraction.

If that initial condition lies in the basin of attraction corresponding to a different asymptotic state, then the eventual configuration will encode that event. Memory is not stored as symbolic tokens, but as separations between trajectories that are robust to dynamics.

If, however, that initial condition lies in the same basin of attraction, then both trajectories will eventually lie on top of each other, on some invariant set. The separation decays, and the internal state forgets the transient event.

3.7 Vanishing and Exploding Gradients as Dynamical Phenomena

The spectral properties we discussed in the previous sections control not only the forward-time evolution of state perturbations but also backward-time propagation of gradients during training.

Remember that linearized forward dynamics around a fixed point drives small perturbations according to repeated multiplication by the Jacobian matrix:

$$\delta h_{t+k} \approx J^k \delta h_t \tag{3.19}$$

Something very similar shows up in Backpropagation Through Time (BPTT). The gradient of the loss $\mathcal{L}$, with respect to some past hidden state h_t, is computed by the chain rule as

$$\frac{\partial \mathcal{L}}{\partial h_t} = \left(\prod_{k=t}^{T-1} J_k^\top \right) \frac{\partial \mathcal{L}}{\partial h_T} \tag{3.20}$$

where

$$J_k = \frac{\partial F}{\partial h}(h_k, x_k) \tag{3.21}$$

is the Jacobian of the recurrent transition at time k.

Therefore, back-propagating gradients are repeatedly multiplied by the same Jacobians that determine how forward-propagating perturbations evolve.

The consequence is that if these Jacobians have dominant eigenvalues with norm smaller than one, then repeated multiplication will exponentially contract gradients (vanishing gradients). Conversely, if their norm is larger than one, then gradients will exponentially explode (exploding gradients) [1, 6].

Importantly, this should come as no surprise: exploding/vanishing gradients is a direct symptom of the same local stability structure that controls how long-forward information can persist. If there are contracting directions, then both forward propagating perturbations and back-propagating gradients will be suppressed along those directions. If there are expanding directions, they will be amplified.

Notice that we did not say that knowing the Jacobian's eigenvalues is sufficient to understand the qualitative dynamics of how perturbations evolve. Eigenvalue magnitude still does not give the full picture. For example, as we will see in the next section, complex eigenvalues can

cause perturbations to "rotate" as they evolve. Similarly, interactions can occur between multiple expanding/contracting directions.

3.8 Oscillatory Modes and Complex Eigenvalues

In Section 3.4, each eigenvalue–eigenvector pair of the Jacobian was interpreted as a dynamical mode that represents an independent direction in the state space. Until now, the analysis focused on eigenvalue magnitude. However, in high-dimensional recurrent networks, the Jacobian matrix generally does not have purely real eigenvalues. Even for real-valued state vectors, complex eigenvalues may arise, appearing in conjugate pairs.

Let a complex eigenvalue be written in polar form:

$$\lambda = re^{i\phi} \tag{3.22}$$

where:

- $r = |\lambda|$ determines the exponential scaling rate of perturbations,
- ϕ determines the rotation angle per time step in the associated two-dimensional invariant subspace.

If a perturbation component evolves along such a mode, repeated application of the linearized dynamics yields

$$\delta h_{t+k} = \lambda^k \delta h_t = r^k e^{ik\phi} \delta h_t \tag{3.23}$$

Two effects are therefore superimposed:

- The factor r^k governs exponential scaling.
- The factor $e^{ik\phi}$ produces rotation.

Although the state vector is real-valued, a conjugate pair of complex eigenvalues corresponds to a real two-dimensional invariant subspace of the state space. Within that subspace, the dynamics take the form

$$\begin{pmatrix} x_{t+k} \\ y_{t+k} \end{pmatrix} = r^k \begin{pmatrix} \cos(k\phi) & -\sin(k\phi) \\ \sin(k\phi) & \cos(k\phi) \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix} \tag{3.24}$$

Thus, over k time steps, the perturbation undergoes a rotation by angle $k\phi$ combined with scaling by factor r^k.

- If $r < 1$, trajectories spiral inward toward the fixed point.
- If $r > 1$, they spiral outward.

- If $r \approx 1$, perturbations neither grow nor decay significantly but rotate persistently within this two-dimensional invariant subspace.

Oscillatory modes are thus inherently connected to the spectrum of non-symmetric recurrent dynamics [18, 26]. Here, trajectories of perturbations do not follow lines along contracting/expanding directions, but the projection onto the corresponding invariant subspace follows a spiral course.

We can update our stability argument from above: fixed points are stable not just by eigenvalue magnitude, but also by how the system evolves along imaginary directions.

3.9 Phase-Dependent Gradient Accumulation

The oscillatory structure described above has direct implications for gradient propagation during training.

As shown in Section 3.7, backward gradients sum up through repeated multiplication by Jacobian matrices. For a mode associated with a complex eigenvalue $\lambda = re^{i\phi}$, the backward contribution over N time steps is proportional to

$$\sum_{k=0}^{N-1} (re^{i\phi})^k \tag{3.25}$$

When $r < 1$, the series decays geometrically, corresponding to vanishing gradients. When $r > 1$, it grows rapidly, corresponding to exploding gradients.

The most subtle regime occurs when $r \approx 1$. In this marginally stable case, each term has approximately constant magnitude but also a phase factor $e^{ik\phi}$. The overall contribution reduces to the finite geometric series

$$S_N = \sum_{k=0}^{N-1} e^{ik\phi} = \frac{1 - e^{iN\phi}}{1 - e^{i\phi}} \tag{3.26}$$

Its magnitude is

$$|S_N| = \frac{|\sin(N\phi/2)|}{|\sin(\phi/2)|} \tag{3.27}$$

This expression makes explicit that reinforcement depends on phase alignment:

- If $\phi \approx 0$, successive contributions align and

$$|S_N| \approx N \tag{3.28}$$

producing near-linear accumulation.

- If $\phi = \pi$, successive terms alternate in sign, leading to near-cancellation.
- For intermediate values of ϕ, partial constructive or destructive interference occurs.

In recurrent networks trained with Backpropagation Through Time (BPTT), the parameter set θ is common for all time steps. The total weight update is therefore proportional to the sum of gradient contributions over the entire sequence:

$$\Delta\theta \propto \sum_{t=0}^{T} g_t \tag{3.29}$$

where each g_t denotes the gradient contribution from time step t.

This result is significant, as the updates are aggregated prior to implementation. The phase relationship between updates in state space is therefore significant: the gradient contributions corresponding to one particular mode will add constructively if they are synchronized in phase across time steps. The result is a large update vector pointing in that direction. If not, the updates will combine, potentially diminishing to nothing.

This means that a mode can have r close to 1, but if there is cancellation of its gradient vector when updating, it will have little reinforcement in training even though it is dynamically persistent in the forward evolution of states.

This shows that the spectral radius r alone does not determine memory characteristics. Since the phases of complex eigenvalues control the cancellation of gradients when they are summed over time, it determines which dynamical components of the internal state are learned.

3.10 Dependence on the Training Horizon

The training horizon refers to the number of time steps over which gradients are back-propagated during Backpropagation Through Time. In the phase-dependent framework developed above, this horizon effectively sets the number of terms N in the series

$$S_N = \sum_{k=0}^{N-1} e^{ik\phi} \tag{3.30}$$

Since the magnitude of S_N depends on both ϕ and N, changing the training horizon also changes the effective amplification of oscillatory modes.

In particular, if $r \approx 1$, oscillatory modes will have a gradient magnitude

$$|S_N| = \frac{|\sin(N\phi/2)|}{|\sin(\phi/2)|} \tag{3.31}$$

which oscillates as a function of N. Therefore, extending the training horizon does not guarantee monotonic improvement in memory reinforcement. Instead, by changing the sequence

length, certain dynamical modes may be alternately amplified or suppressed, based on their phase structure.

Thus, the interaction between phase ϕ and training horizon N provides a principled explanation for non-monotonic memory behavior as a function of sequence length.

3.11 Memory Crackpoints

The above perspective suggests that recurrent network memories depend on qualitative features of the learned dynamical system: spectral structure, oscillatory modes, basin geometry, and phase-dependent gradient reinforcement.

Since gradient reinforcement of marginally stable modes relies on both eigenvalue size and phase matching throughout the training sequence, small changes in sequence length can change which modes receive effective reinforcement. If such a variation causes a reinforced mode to become ineffective — or permits a previously suppressed mode to become reinforced — then the landscape of internal states can suddenly change organization.

This dramatic shift is realized through a change in the geometry of attractors and their basins. Trajectories may flow into separate basins following a transient perturbation in one regime, maintaining state separation history-dependently. In another regime, the same perturbation may end up at the same attractor, resolving any internal distinction.

We refer to the boundary between such regimes as a *memory-crackpoint*.

Thus, a memory-crackpoint does not refer merely to a decline in performance — it is a qualitative shift in the internal dynamical topology of the model. A memory-crackpoint is a phase transition between:

- regimes where history-dependent trajectory separation is maintained under autonomous dynamics, and
- regimes where internal trajectories collapse to a common attractor despite similar short-term predictive behavior.

Crucially, since these phenomena occur in the state space rather than at the output layer, memory-crackpoints can occur without evident decline when using traditional performance metrics like accuracy or F1. A model may still correctly predict individual time steps even as it loses the internal dynamical fidelity to separate long-term histories.

Memory crackpoints are therefore dynamical phase transitions caused by microscopic changes in spectral reinforcement. They theoretically explain sudden, non-monotonic changes in memorization that cannot be understood via vanishing gradients.

3.12 Summary

In this chapter we introduced a dynamical systems perspective on memory within sequential models. Viewing recurrent models as autonomous discrete-time systems in the presence of con-

stant inputs allowed us to interpret memory geometrically as a property of state space.

We explored this perspective starting from local stability around fixed points of the system and continuing towards understanding spectral mixture in large-dimensional spaces, which allowed us to consider oscillatory dynamics induced by complex eigenvalues.

This introduced spiral dynamics on invariant planes as well as insight that persistence is affected by eigenvalue magnitude, but also by phase structure. Therefore, memory for a fixed mode does not have to monotonically increase with sequence length; it can be reinforced or weakened as the horizon is extended.

Building upon these ideas, we introduced the concept of memory-crackpoints as divisions in the underlying dynamics where the internal geometry of state space (attractors/basins) can suddenly change. Memory crackpoints expose the phenomenon where learnt history dependence can disappear and reappear without traditional measures of performance decreasing.

The empirical investigations presented next study crackpoints within trained RNN, GRU, and LSTM models of various sequence lengths.

4 Experimental Framework and Instrumented Recurrent Modeling

4.1 Experimental Motivation and Research Focus

In Chapter 3, we introduced a dynamical systems perspective on memory in recurrent networks by interpreting memory capacity as a property of the recurrent state space geometry. In this chapter, we aim to empirically test this hypothesis. Instead of simply benchmarking predictions at long timescales, we investigate whether variations in sequence length cause qualitative reorganizations of learned recurrent dynamics.

We construct a task in which accurate discrimination at the terminal time-step requires memory of a transient intermediate cue. Because activity measurements return to steady state following the transient, successful task solution requires preservation of an internal memory trace.

As stated in Chapter 3, varying the temporal horizon can cause abrupt transitions between remembering and forgetting if attractor organization and basin geometry are qualitatively reorganized. We test this hypothesis below by correlating task performance with LSTM gate dynamics and hidden-state geometry.

4.2 Dataset Design

This section describes the physical system used for data generation and the methodology used to construct the time-series datasets for the experimental analysis. The objective is to combine physical realism through high-fidelity simulation with a controlled temporal structure that explicitly requires long-term memory retention.

4.2.1 Gas Turbine Simulation Framework

For the purposes of this study, a low bypass ratio turbofan engine typical of modern civil aviation applications is considered, as described in [27] and [28]. The engine configuration includes a fan, high-pressure compressor (HPC), high-pressure turbine (HPT), and low-pressure turbine (LPT), forming a representative mixed-flow turbofan architecture.

A schematic representation of the engine layout, including station numbering, monitored measurements, and component health parameters, is shown in Figure 4.1.

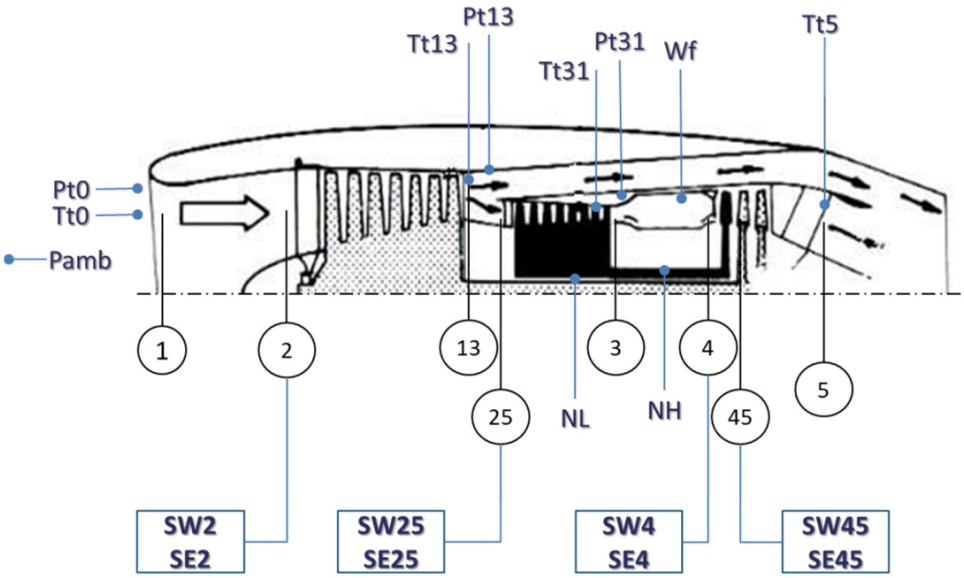

Figure 4.1: *Schematic layout of the considered turbofan engine, indicating measurement stations and associated health parameters (source: [28]).*

During engine operation, three categories of quantities are defined:

- Operating conditions: $\mathbf{u} \in \mathbb{R}^{n_u}$, which determine the thermodynamic operating point (e.g., altitude, flight speed).
- Measured quantities: $\mathbf{Y} \in \mathbb{R}^{n_Y}$, which are used for condition monitoring.
- Component health parameters: $\mathbf{f} \in \mathbb{R}^{n_f}$, which characterize the internal condition of engine components.

Table 4.1 summarizes the measured quantities and component health parameters. Typically, two health parameters are used for each component: the flow factor (SW), indicating the component's swallowing capacity, and the efficiency factor (SE), representing thermal efficiency. The application of such parameters for assessing engine component health has been discussed in [29]. A deviation of a health parameter from its nominal value signals a fault in that component. As extensively discussed in [30], the ratio of deviations between two health parameters can serve as a characteristic metric for faults such as fouling, increased tip clearance, erosion, and foreign object damage. The severity of the fault correlates with the level of this deviation.

Table 4.1: *Measured quantities and component health parameters of the considered turbofan engine.*

Operating Conditions (u)	
Name	**Symbol**
Ambient pressure	Pamb
Total pressure at station '0'	Pt0
Total temperature at station '0'	Tt0
LP shaft speed	NL

Measured Quantities (Y)		**Health Parameters** (f)	
Name	**Symbol**	**Name**	**Symbol**
Fuel flow rate	Wf	FAN Flow factor	SW2
HP shaft speed	NH	FAN Efficiency factor	SE2
Total pressure at station '13'	Pt13	HPC Flow factor	SW25
Total temperature at station '13'	Tt13	HPC Efficiency factor	SE25
Total pressure at station '31'	Pt31	HPT Flow factor	SW4
Total temperature at station '31'	Tt31	HPT Efficiency factor	SE4
Total temperature at station '5'	Tt5	LPT Flow factor	SW45
Total pressure at station '45'	Pt45	LPT Efficiency factor	SE45

For a generic health parameter f_c, its relative deviation from nominal is defined as

$$\Delta f_c = \frac{f_c - f_{c,0}}{f_{c,0}} \tag{4.1}$$

where $f_{c,0}$ denotes the nominal value.
In vector form, the health deviation vector is

$$\Delta \mathbf{f} = \frac{\mathbf{f} - \mathbf{f}_0}{\mathbf{f}_0} \tag{4.2}$$

where division is element-wise.

The engine behavior is modeled by a nonlinear Engine Performance Model (EPM), defined as

$$\mathbf{Y} = \mathbf{g}(\mathbf{u}, \mathbf{f}) \tag{4.3}$$

as described in [27] and based on the generic simulation framework introduced in [31]. The EPM used in this work is implemented in MATLAB and is employed exclusively as a high-fidelity data-generation mechanism.

For nominal (healthy) operation, where the health parameter vector is denoted $\mathbf{f}_0$, and a given operating point $\mathbf{u}$, the corresponding nominal measurement vector is

$$\mathbf{Y}_0(\mathbf{u}) = \mathbf{g}(\mathbf{u}, \mathbf{f}_0) \tag{4.4}$$

This explicitly emphasizes that nominal measurements depend on the operating point.

For a given operating point $\mathbf{u}$ and health state $\mathbf{f}$, the percentage deviation of measured quantities is defined element-wise as

$$\Delta\mathbf{Y}(\mathbf{u}, \mathbf{f}) = \frac{\mathbf{g}(\mathbf{u}, \mathbf{f}) - \mathbf{g}(\mathbf{u}, \mathbf{f}_0)}{\mathbf{g}(\mathbf{u}, \mathbf{f}_0)} \tag{4.5}$$

This vector $\Delta\mathbf{Y} \in \mathbb{R}^{n_Y}$ constitutes the fault signature corresponding to health state $\mathbf{f}$ under operating point $\mathbf{u}$.

The EPM is not modified in this work; it is used solely as a deterministic mapping from operating and health states to measurement deviations.

4.2.2 Generation of Controlled Fault Signatures and Time-Series Datasets

Five distinct health conditions were considered, as summarized in Table 4.2. The first health condition represents the nominal, healthy operation of the engine, with no deviations in health parameters across any components.

The second health condition (coded as TIP) models an increased clearance of the compressor blades. As reported in [30], this fault results in a ratio of the related health parameter deviations, $\Delta SW/\Delta SE$, of 0.2; the actual deviations of ΔSW and ΔSE are around -1% and -5% (deviations range from 0.7 up to 1.3 of these values), respectively, thereby simulating tip clearance faults of varying severity.

The third health condition pertains to mistuning of the inlet guide vanes (VGV) of the compressor. In these instances, deviations in the compressor's health parameters ΔSW and ΔSE typically range from approximately $+3\%$ to -1%, respectively, resulting in a ratio of -3.

The fourth health condition involves damage caused by foreign objects (FOD) within the compressor. In such cases, deviations in both health parameters are generally around -4%, yielding a ratio of 1.

The fifth health condition represents the concurrent occurrence of VGV and FOD faults (denoted as VGV+FOD). The combined effect in this scenario is additive, with deviations approximately around $+1\%$ and -5%, respectively, leading to a ratio of 0.2.

Notably, under steady-state conditions, the measurement deviations associated with the TIP fault coincide with those of the combined VGV+FOD fault. Consequently, these two fault types are indistinguishable when considered as static signatures and differ only through their temporal structure.

Table 4.2: *Considered health conditions and characteristic deviation ratios.*

label	health condition	ΔSW	ΔSE	$\Delta SW/\Delta SE$
1	Healthy	0%	0%	N/A
2	TIP fault	-1%	-5%	0.2
3	VGV fault	+3%	-1%	-3
4	FOD fault	-4%	-4%	1
5	VGV+FOD faults	-1%	-5%	0.2

Data were generated for ten distinct operating points

$$\mathbf{u}_1, \ldots, \mathbf{u}_{10} \tag{4.6}$$

representing typical flight conditions.
For each operating point, the following steady-state health configurations were simulated:

- 1 healthy condition,
- 60 TIP variations,
- 60 VGV variations,
- 60 FOD variations.

Thus, for each operating point:

$$1 + 3 \times 60 = 181 \tag{4.7}$$

distinct steady-state health states were generated.
Across all operating points:

$$10 \times 181 = 1810 \tag{4.8}$$

static fault signatures were obtained.
Each signature corresponds to a measurement deviation vector

$$\Delta\mathbf{Y}_{ij} = \Delta\mathbf{Y}(\mathbf{u}_i, \mathbf{f}_j) \tag{4.9}$$

forming the static dataset

$$\mathcal{D}_{\text{static}} = \{\Delta\mathbf{Y}_{ij}\} \tag{4.10}$$

From the static signatures, time-series datasets were constructed for sequence lengths

$$T \in \{60, 100, 150, 200, 250, 300, 400, 500, 750, 1000, 1800, 3600\} \tag{4.11}$$

Each time series corresponds to a fixed operating point $\mathbf{u}$ and is generated by defining a time-dependent health state function

$$\mathbf{f}(t), \quad t = 1, \ldots, T \tag{4.12}$$

The clean (noise-free) measurement deviation at time t is given by

$$\Delta\mathbf{Y}(t) = \Delta\mathbf{Y}(\mathbf{u}, \mathbf{f}(t)) \tag{4.13}$$

Three categories of time series were constructed.

(a) Healthy Sequences:

For healthy sequences,

$$\mathbf{f}(t) = \mathbf{f}_0, \quad \forall t \tag{4.14}$$

All time steps correspond to healthy operation.

(b) Single-Fault Sequences (TIP, VGV, FOD):

For each fault type and each of the 60 magnitudes,

$$\mathbf{f}(t) = \begin{cases} \mathbf{f}_0, & t < t_f \\ \mathbf{f}_{\text{fault}}, & t \geq t_f \end{cases} \tag{4.15}$$

where the final fault onset time is defined as

$$t_f = \begin{cases} T - 40, & T \geq 150 \\ T - 20, & T = 60 \text{ or } 100 \end{cases} \tag{4.16}$$

Thus, the sequence consists of an initial healthy phase followed by a persistent fault in the final segment.

Each of the 60 sequences corresponds to a different fault magnitude drawn from $\mathcal{D}_{\text{static}}$.

(c) VGV+FOD Sequences with Transient Precursor:

For the combined VGV+FOD condition, the time-dependent health state is defined as

$$\mathbf{f}(t) = \begin{cases} \mathbf{f}_0, & t < t_p \\ \mathbf{f}_{\text{VGV}}, & t_p \leq t < t_p + 10 \\ \mathbf{f}_0, & t_p + 10 \leq t < t_f \\ \mathbf{f}_{\text{VGV+FOD}}, & t \geq t_f \end{cases} \tag{4.17}$$

where

- t_f denotes the onset of the final persistent fault,

- t_p denotes the onset of the transient VGV precursor event.

The precursor event lasts exactly 10 time steps and occurs at a predefined temporal distance

$$d = t_f - t_p \tag{4.18}$$

with

$$d \in \{3400, 1600, 900, 700, 450, 350, 250, 200, 150, 125, 100, 90, 80, 70, 50, 40, 30, 20, 10\} \tag{4.19}$$

For shorter sequence lengths, only admissible values of d satisfying $t_p > 0$ are used.

Crucially, the steady-state deviation vector associated with the final VGV+FOD fault is identical to that of the TIP fault. Therefore, discrimination between these two classes cannot rely on steady-state measurements but depends entirely on whether the transient precursor event is retained in the model's internal state.

To emulate realistic instrumentation conditions, additive Gaussian noise is superimposed on the clean measurement deviations.

For each time step,

$$\mathbf{x}_t = \Delta\mathbf{Y}(t) + \boldsymbol{\epsilon}_t \tag{4.20}$$

where

$$\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma) \tag{4.21}$$

and Σ is a diagonal covariance matrix whose standard deviations correspond to the measurement noise levels reported in Table 4.3, consistent with [28].

Table 4.3: *Considered measurement noise levels.*

Measurement	**Symbol**	**Noise (%)**
Ambient pressure	Pamb	0.14
Total pressure at station '0'	Pt0	0.10
Total temperature at station '0'	Tt0	0.23
LP shaft speed	NL	0.05
Fuel flow rate	Wf	0.15
HP shaft speed	NH	0.05
Total pressure at station '13'	Pt13	0.17
Total temperature at station '13'	Tt13	0.23
Total pressure at station '31'	Pt31	0.17
Total temperature at station '31'	Tt31	0.10
Total temperature at station '5'	Tt5	0.10
Total pressure at station '45'	Pt45	0.17

The final observed time series for each sequence is therefore

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\} \tag{4.22}$$

A key design principle of the dataset is that, for each sequence length T, the temporal distance d between the precursor event and the final fault onset remains fixed across experiments.

This ensures that:

- The memory-critical temporal separation is identical for all sequences of a given length.
- Variations in T modify only the overall temporal horizon.
- Comparisons of model performance across sequence lengths reflect differences in memory retention capability rather than differences in event positioning.

Longer sequences permit a larger set of admissible precursor distances; however, within each fixed T, all comparisons are made under controlled temporal separation.

Time series corresponding to seven operating points were used for training, with 20% of the training data reserved for validation during optimization. The remaining three operating points were reserved exclusively for evaluation.

This split ensures:

- No leakage across operating conditions,
- Evaluation under unseen thermodynamic regimes,
- Assessment of genuine generalization capability.

4.3 Model Families and Dynamical Formulation

This section defines the model architectures considered in the experimental analysis. The objective is not just to compare predictive performance across different neural network types, but also to examine how their different mechanisms affect the internal state-space dynamics introduced in Chapter 3.

Throughout this section, the observed time series is denoted

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}, \quad \mathbf{x}_t \in \mathbb{R}^{n_Y} \tag{4.23}$$

as defined in Section 4.2.

For recurrent models, the hidden state is denoted

$$\mathbf{h}_t \in \mathbb{R}^{n_h} \tag{4.24}$$

and, where applicable, the cell state

$$\mathbf{c}_t \in \mathbb{R}^{n_h} \tag{4.25}$$

where n_h denotes the dimensionality of the hidden state vector. For multi-layer configurations, all layers share the same hidden dimensionality n_h.

All models produce, at each time step, an output vector

$$\widehat{\mathbf{y}}_t \in [0,1]^K \tag{4.26}$$

corresponding to the predicted fault labels, where K denotes the number of monitored fault labels.

The learning task is formulated as multi-label classification at each time step. The corresponding ground-truth vector is denoted

$$\mathbf{y}_t \in \{0,1\}^K \tag{4.27}$$

4.3.1 Non-Sequential Baseline: Multilayer Perceptron

To isolate temporal dependency from classification complexity, a non-sequential baseline model is introduced.

The multilayer perceptron (MLP) operates independently at each time step. Its mapping is defined as

$$\widehat{\mathbf{y}}_t = \Phi(\mathbf{x}_t; \theta) \tag{4.28}$$

where Φ denotes a feedforward neural network with parameters θ.

Formally, the MLP does not maintain any internal state. Thus, the prediction at time t depends exclusively on the instantaneous measurement vector $\mathbf{x}_t$.

Including this model has a methodological motivation as well. If we are able to solve a fault discrimination task with the MLP, then the task does not depend on temporal memory. If the MLP fails selectively on certain fault classes, we know that instantaneous measurements do not suffice to solve the task, and we have isolated temporal dependency as the distinguishing factor.

Notice that the MLP represents a fixed decision boundary in measurement space. It creates no dynamics in state space.

4.3.2 Recurrent Architectures as Discrete-Time Dynamical Systems

In contrast to the MLP, recurrent models define a discrete-time nonlinear dynamical system

$$\mathbf{h}_t = \mathcal{F}(\mathbf{h}_{t-1}, \mathbf{x}_t; \theta) \tag{4.29}$$

where $\mathcal{F}$ denotes the architecture-specific state transition function.

The output mapping is given by

$$\widehat{\mathbf{y}}_t = \Psi(\mathbf{h}_t; \theta) \tag{4.30}$$

with Ψ typically implemented as a linear layer followed by sigmoid activation.

Thus, recurrent models evolve an internal state trajectory $\mathbf{h}_t$, whose geometry determines whether information about earlier events is retained or dissipated.

Three recurrent architectures are considered: Vanilla RNN, GRU, and LSTM.

4.3.3 Vanilla RNN

The vanilla recurrent neural network [4] updates its hidden state according to

$$\mathbf{h}_t = \tanh\left(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}\right) \tag{4.31}$$

where

- $\mathbf{W}_x \in \mathbb{R}^{n_h \times n_Y}$,
- $\mathbf{W}_h \in \mathbb{R}^{n_h \times n_h}$,
- $\mathbf{b} \in \mathbb{R}^{n_h}$.

Recall that this architecture describes a simple autonomous system driven by input $\mathbf{x}_t$. The recurrent weight matrix $\mathbf{W}_h$ controls the contractive/expansive behavior of the hidden-state dynamics and thus memory (see Chapter 2 and Chapter 3 for details).

The standard RNN lacks gating. Memory here relies solely on the spectral characteristics of $\mathbf{W}_h$ and the nonlinearity.

4.3.4 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) [10] introduces gating mechanisms that regulate state updates.

The reset and update gates are defined as

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{4.32}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{4.33}$$

The candidate hidden state is

$$\widetilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{4.34}$$

and the final update is

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \widetilde{\mathbf{h}}_t \tag{4.35}$$

The update gate $\mathbf{z}_t$ determines the degree to which the previous hidden state is retained, while the reset gate $\mathbf{r}_t$ modulates the influence of the previous state in the candidate computation.

4.3.5 Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) architecture [2] introduced an explicit cell state $\mathbf{c}_t$, designed to alleviate vanishing gradient effects and enable long-term information retention.

The gating equations are

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{4.36}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{4.37}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{4.38}$$

The candidate cell update is

$$\widetilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{4.39}$$

The cell and hidden updates are

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{c}}_t \tag{4.40}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{4.41}$$

The forget gate $\mathbf{f}_t$ directly controls memory decay, while the input gate $\mathbf{i}_t$ governs new information injection.

4.3.6 Multi-Layer and Parameter Configuration

All recurrent architectures were implemented in single- and two-layer configurations. For multi-layer models, the hidden state of layer ℓ is denoted

$$\mathbf{h}_t^{(\ell)} \tag{4.42}$$

The layer-wise updates are defined as

$$\mathbf{h}_t^{(1)} = \mathcal{F}^{(1)}(\mathbf{h}_{t-1}^{(1)}, \mathbf{x}_t) \tag{4.43}$$

$$\mathbf{h}_t^{(\ell)} = \mathcal{F}^{(\ell)}\big(\mathbf{h}_{t-1}^{(\ell)}, \mathbf{h}_t^{(\ell-1)}\big) \tag{4.44}$$

Hidden dimensionality n_h varies according to the parametric study described later in Section 4.5.

4.3.7 Instrumented State Logging

To enable dynamical analysis, internal variables were recorded at each time step:

- Hidden state $\mathbf{h}_t$ (all architectures),
- Cell state $\mathbf{c}_t$ (LSTM),
- Gate activations ($\mathbf{z}_t, \mathbf{r}_t, \mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$).

These logged trajectories form the basis for:

- Gate activation heatmaps,
- PCA and UMAP projections,
- Attractor Separation Index computation.

Thus, recurrent models are treated explicitly as dynamical systems whose internal trajectories constitute the primary object of analysis.

4.4 Training Protocol and Optimization Framework

This section describes the training methodology considered for all neural architectures in the present study. The objective is to ensure methodological consistency across models while preserving full reproducibility of the experimental results. The notation introduced in Sections 4.2 and 4.3 is retained throughout this section.

4.4.1 Loss Function and Optimization Procedure

Training is formulated as multi-label classification at each time step. Given the predicted output vector $\widehat{\mathbf{y}}_t \in [0,1]^K$ and the corresponding ground-truth label vector $\mathbf{y}_t \in \{0,1\}^K$, learning is performed using the Binary Cross-Entropy (BCE) loss applied element-wise to each label.

For a sequence of length T, the sequence-level loss is defined as the temporal average:

$$\mathcal{L}_{seq} = \frac{1}{T} \sum_{t=1}^{T} BCE(\mathbf{y}_t, \widehat{\mathbf{y}}_t) \tag{4.45}$$

where $BCE(\cdot, \cdot)$ denotes the standard binary cross-entropy applied independently to each output dimension.

This formulation guarantees that each timestep is weighted equally when minimizing the loss function. Diagnostic problem demands detection not only at the final timestep of fault onset but at all timesteps, so we weight all timesteps equally.

The Adam optimizer [32], an adaptive moment first-order stochastic gradient optimizer, was used to train each network. The Adam hyperparameters were kept consistent across architectures.

4.4.2 Validation and Early Stopping

Early stopping based on the validation loss was used to avoid overfitting and achieve stable convergence, with patience fixed. Optimization was performed with mini-batch training.

4.4.3 Initialization and State Reset

For every architecture, the hidden state $\mathbf{h}_0$ (and cell state $\mathbf{c}_0$ for LSTM) was initialized to zero at the start of each sequence. This allows memory to be built solely from the input sequence and ensures that no information leaks between sequences. Therefore, each time series can be treated as an independent realization of the dynamical system.

Random seeds were fixed across experiments to ensure reproducibility and to isolate the effect of architectural and horizon variations from stochastic initialization effects.

4.4.4 Sequence Handling and Horizon Dependence

Each model is trained using fixed-length sequences, corresponding to a given training horizon T. A separate instance of the model is trained for each T. This is deliberate. As seen in Chapter 3, changing the training horizon can induce different dynamical regimes in recurrent models. Training separate models for each value of T allows us to directly investigate the existence of such regime transitions/memory-crackpoints as a function of T.

4.4.5 Generalization Across Operating Conditions

As mentioned in Sections 4.2, we considered time series from seven operating points for training and time series from other three operating points only for evaluation.

The split ensures that models will be evaluated on unseen thermodynamic regimes. Thus, the scores measure the generalization over operating conditions rather than memorization of operating points.

4.5 Hyperparameter Strategy and Parametric Sensitivity Design

The goal of our hyperparameter study is not to achieve peak performance. Instead, we want to understand if the qualitatively different memory regimes and attractor reorganizations observed depend on architectural capacity or constitute intrinsic dynamical phenomena.

Two architectural parameters are systematically varied:

- Hidden state dimensionality n_h,
- Number of recurrent layers L.

In addition, multiple random seeds are used to evaluate sensitivity to initialization.

4.5.1 Hidden State Dimensionality

For recurrent architectures, the hidden state dimension

$$\mathbf{h}_t \in \mathbb{R}^{n_h} \tag{4.46}$$

determines the dimensionality of the internal state space. Increasing n_h expands the model's representational capacity and potentially increases the complexity of attractor geometries.

The following values were considered:

$$n_h \in \{16, 32, 64\} \tag{4.47}$$

This range spans low, medium, and moderately high-dimensional state spaces without introducing excessive parameter growth that would obscure interpretability.

4.5.2 Depth of Recurrent Architecture

Single-layer and two-layer recurrent configurations were evaluated.

For multi-layer models, the hidden state of layer ℓ evolves according to

$$\mathbf{h}_t^{(\ell)} = \mathcal{F}^{(\ell)}\left(\mathbf{h}_{t-1}^{(\ell)}, \mathbf{h}_t^{(\ell-1)}\right) \tag{4.48}$$

with identical hidden dimensionality n_h across layers.

The depth parameter explored is:

$$L \in \{1, 2\} \tag{4.49}$$

4.5.3 Sensitivity to Initialization

To evaluate robustness with respect to stochastic initialization, experiments were conducted using multiple random seeds.

Let $\theta^{(s)}$ denote the parameter realization corresponding to seed s. For each configuration (n_h, L), performance metrics are computed across a set of seeds:

$$s \in \{7, 21, 42, 84, 168\} \tag{4.50}$$

Mean and standard deviation of performance are reported:

$$\bar{\text{F1}} = \frac{1}{S}\sum_{s=1}^{S} \text{F1}^{(s)} \tag{4.51}$$

$$\sigma_{\text{F1}} = \sqrt{\frac{1}{S}\sum_{s=1}^{S}\left(\text{F1}^{(s)} - \bar{\text{F1}}\right)^2} \tag{4.52}$$

Low variance across seeds indicates structural behavior, while high variance would suggest sensitivity to initialization.

4.6 Metrics for Performance and Memory Dynamics

This section defines the quantitative and geometric metrics used to evaluate model performance.

The analysis proceeds at two complementary levels:

1. Output-level performance metrics,
2. Internal dynamical and geometric metrics.

4.6.1 Performance Metrics

Predictive performance is evaluated using the F1 score, computed independently for each fault class.

Let: TP, FP, FN denote true positives, false positives, and false negatives for a given class.

Precision and recall are defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.53}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.54}$$

The F1 score is given by:

$$\text{F1} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.55}$$

Macro-averaged F1 is computed across classes to provide an overall performance indicator.

The F1 score is preferred over accuracy because:

- The task is multi-label,
- Class imbalance may arise due to temporal structure,
- Both precision and recall are equally important in fault diagnosis.

Per-class F1 values are also reported to detect selective classification failures between specific fault pairs.

4.6.2 Gate Activation Statistics

To examine internal mechanisms of memory retention, gate activations are analyzed for GRU and LSTM architectures.

Let $\mathbf{g}_t \in [0,1]^{n_h}$ denote a generic gate vector (e.g., update, reset, forget, input, output gate).

For a given sequence, the mean gate activation at time t is:

$$\bar{g}(t) = \frac{1}{n_h} \sum_{i=1}^{n_h} g_{t,i} \tag{4.56}$$

Heatmaps of gate activations across time and hidden dimensions are also visualized.

These statistics allow detection of:

- Persistent retention (e.g., sustained forget gate suppression),
- Reversion to baseline values after transient events,
- Saturation behavior.

Such patterns provide qualitative insight into how internal state transitions relate to memory preservation or collapse.

4.6.3 Hidden-State Geometry via PCA

To examine the geometry of hidden-state trajectories, Principal Component Analysis (PCA) [33] is applied to the collection of hidden-state vectors.

Let:

$$\mathbf{h}_t^{(i)} \tag{4.57}$$

denote the hidden state at time t for sequence i.

The set of all hidden states from training sequences is used to fit a unified PCA transformation:

$$\mathbf{z}_t^{(i)} = \mathbf{P}^\top \mathbf{h}_t^{(i)} \tag{4.58}$$

where $\mathbf{P}$ contains the principal directions.

The same transformation is then applied to test sequences.

This unified projection ensures that differences across sequence lengths are interpreted within a common latent coordinate system.

PCA enables visualization of:

- Trajectory convergence,
- Attractor formation,
- Basin separation or merging.

4.6.4 Nonlinear Embedding via UMAP

In addition to PCA, we also use Uniform Manifold Approximation and Projection (UMAP) [34]. UMAP learns a nonlinear low-dimensional embedding while preserving local neighborhood information.

Similar to PCA, we train one UMAP model on the hidden states from selected training regimes and apply it to test trajectories. UMAP here is used only qualitatively, to check that separation or merging of attractors is not caused by projecting into linear space. If both PCA and UMAP embeddings agree, we are more confident in our geometric observations.

4.6.5 Attractor Separation Index (ASI)

Visual assessment of PCA and UMAP projections allows us to qualitatively reason about the formation and merging of attractors. To quantitatively compare geometric separation across sequence lengths/architectures, however, we need a defined metric.

We propose the Attractor Separation Index (ASI) for this purpose. ASI is based on a commonly utilized principle in discriminant analysis/cluster validity: measuring separability between two collections of points by comparing inter-class distance to intra-class scatter [35].

Let:

- $\mathcal{H}_A$ denote the set of latent points associated with class A,
- $\mathcal{H}_B$ denote the set associated with class B.

Define centroids:

$$\mathbf{c}_A = \frac{1}{|\mathcal{H}_A|} \sum_{\mathbf{z} \in \mathcal{H}_A} \mathbf{z} \tag{4.59}$$

$$\mathbf{c}_B = \frac{1}{|\mathcal{H}_B|} \sum_{\mathbf{z} \in \mathcal{H}_B} \mathbf{z} \tag{4.60}$$

The inter-class distance is:

$$D_{between} = \|\mathbf{c}_A - \mathbf{c}_B\| \tag{4.61}$$

The intra-class dispersion is defined as the average distance of points from their centroid:

$$D_{within} = \frac{1}{2} \left(\frac{1}{|\mathcal{H}_A|} \sum_{\mathbf{z} \in \mathcal{H}_A} \|\mathbf{z} - \mathbf{c}_A\| + \frac{1}{|\mathcal{H}_B|} \sum_{\mathbf{z} \in \mathcal{H}_B} \|\mathbf{z} - \mathbf{c}_B\| \right) \tag{4.62}$$

The Attractor Separation Index is defined as:

$$\text{ASI} = \frac{D_{between}}{D_{within}} \tag{4.63}$$

Two variants are evaluated:

- ASI_{fault}: separation between the two steady-state fault attractors,
- $\text{ASI}_{healthy}$: separation between healthy states before and after a transient precursor event.

High ASI values indicate strong basin separation, while values close to zero indicate attractor merging.

4.6.6 Integrated Interpretation

The metrics introduced above serve complementary purposes:

- F1 score captures observable classification behavior.
- Gate statistics reveal local dynamical mechanisms.
- PCA and UMAP expose global state-space geometry.
- ASI provides quantitative attractor separation measurement.

Together, these metrics enable a transition from performance-level observations to mechanistic dynamical interpretation, addressing the theoretical framework developed in Chapter 3.

5 Experiments and Dynamical Analysis of Long-Term Memory

In this chapter, experimental results are presented from the dynamical perspective introduced in Chapter 3. Connecting classifier performance, internal gate dynamics, hidden-state geometry (PCA, UMAP), and a quantitative attractor separation index (ASI), we aim to link observable behavior to organization in state-space.

Unless otherwise noted, all trained models described use one recurrent layer and hidden size 8; the effect of various architectures is explored separately in the parametric sensitivity study.

Our aim is not simply to benchmark predictive performance, but to uncover if horizon-dependent fluctuations in performance are due to task difficulty or a dynamical reorganization of internal memory.

5.1 MLP Baseline and the Role of Temporal Dependency

We begin by evaluating the non-sequential MLP baseline introduced in Section 4.3.1.

The MLP performs classification independently at each time step:

$$\widehat{\mathbf{y}}_t = \Phi(\mathbf{x}_t; \theta) \tag{5.1}$$

with no recurrence and no internal state evolution.

Figure 5.1 shows the macro-F1 scores for all sequence lengths and considered faults. The model achieves high performance across most fault categories, confirming that the measurement deviations are, in general, discriminative.

On the other hand, we observe a catastrophic failure to discriminate between the two fault classes which have the same steady-state signature but are preceded by different transient events (i.e., TIP fault and VGV+FOD fault). Since MLP cannot see beyond history, it arbitrarily maps these sequences to one of the two steady-state classes.

This lays down an important point: the problem per se is not difficult to classify. The difficulty lies only if one has to discriminate based on knowledge of previous events. Therefore, memory of past events is required.

5.2 Performance Across Recurrent Architectures

We now evaluate RNN, GRU, and LSTM architectures across sequence lengths. The following figures (Figure 5.2 to Figure 5.4) illustrate performance trends.

From these figures, the following key observations emerge:

- All models demonstrate high performance on detecting Healthy condition, FOD faults, and VGV faults, under all sequence lengths.

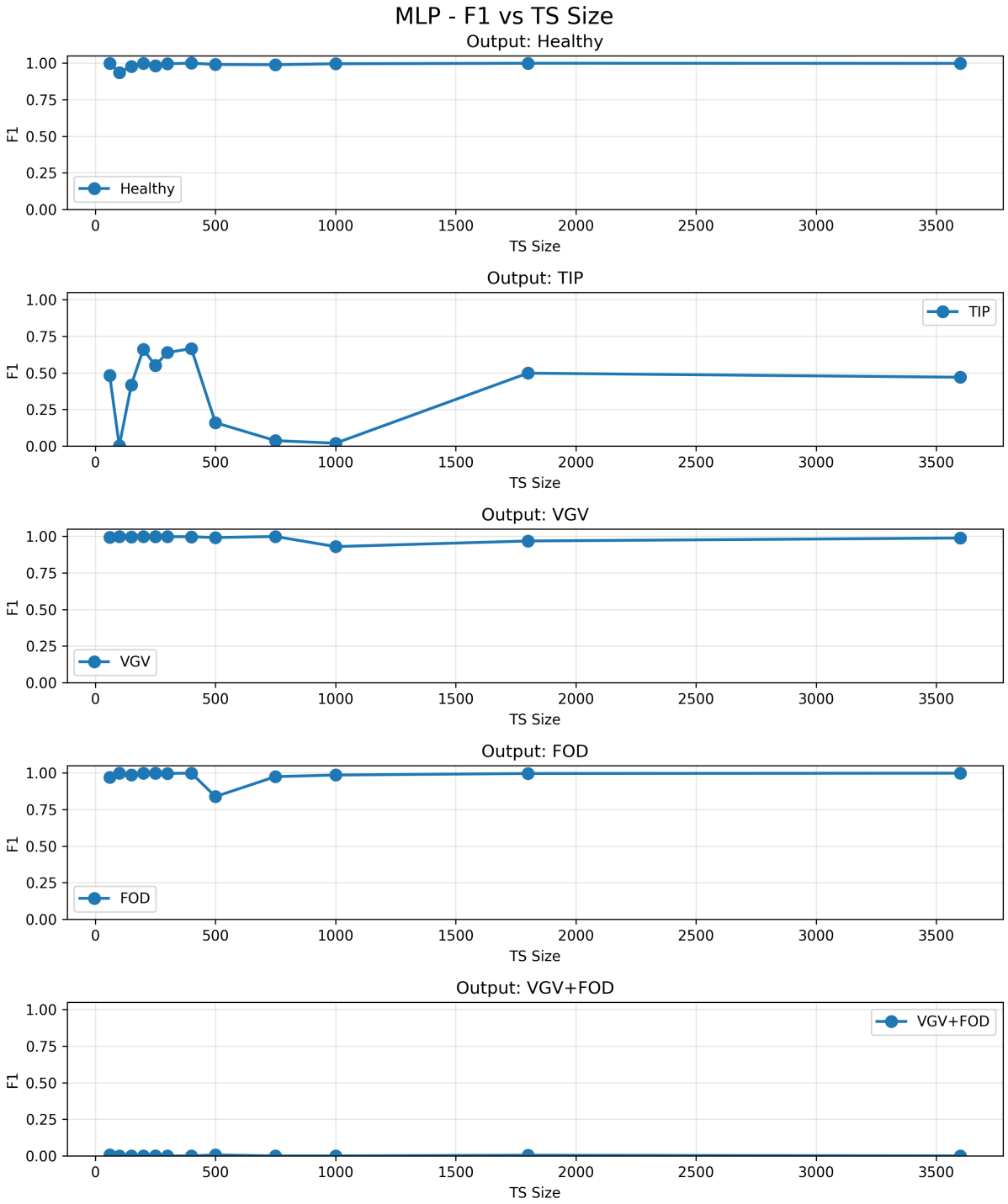

Figure 5.1: *MLP performance across sequence lengths.*

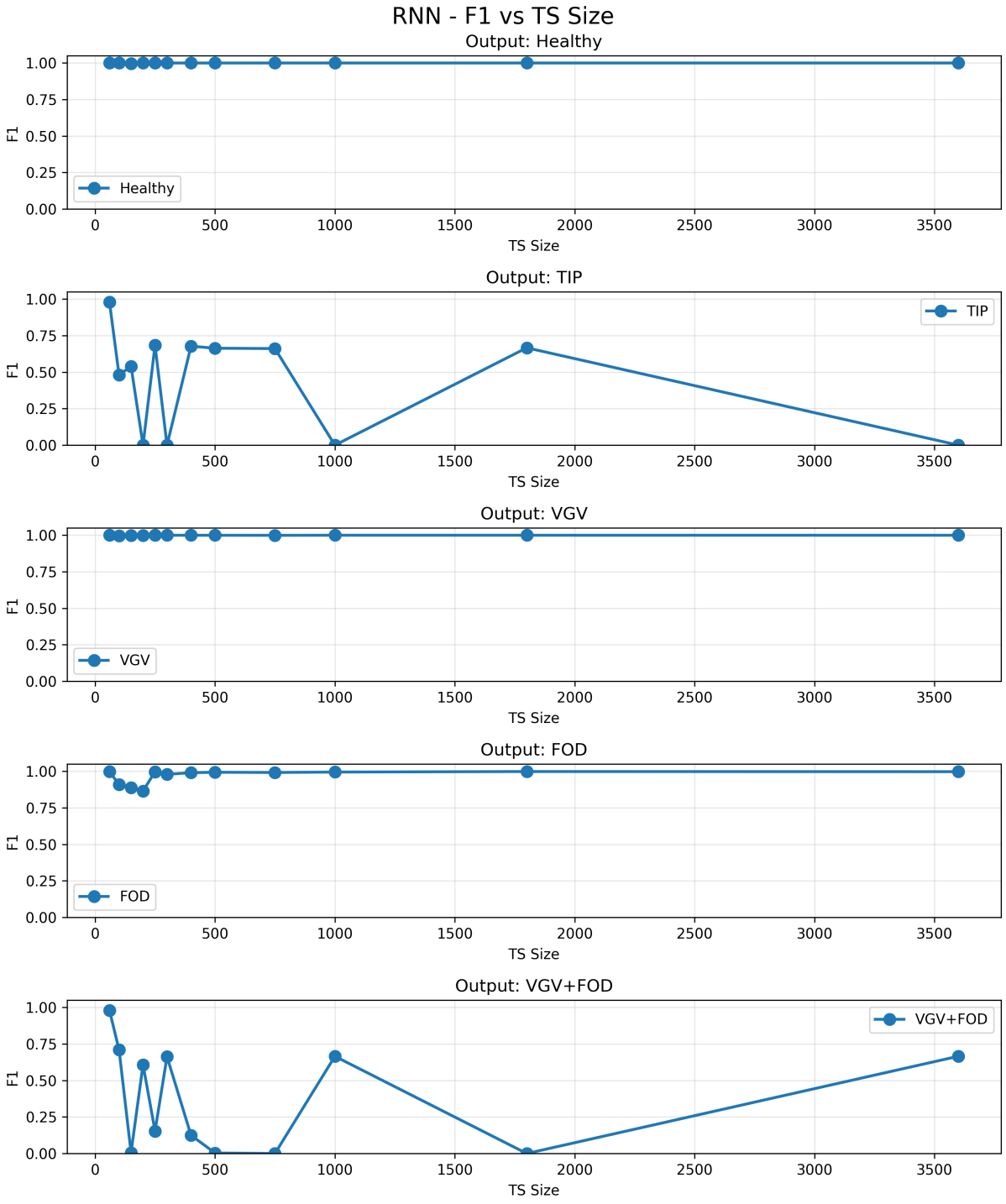

Figure 5.2: *RNN performance across sequence lengths.*

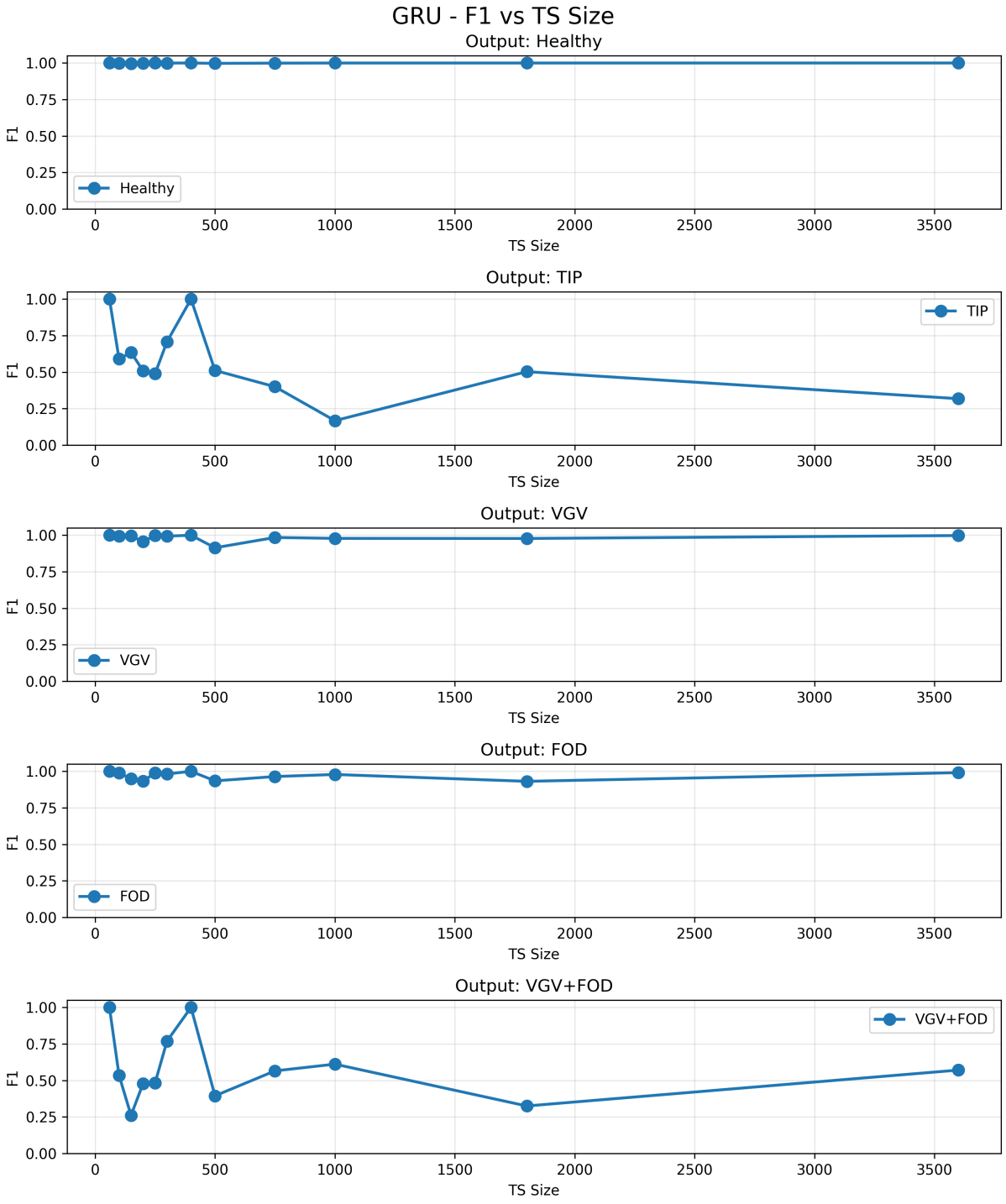

Figure 5.3: *GRU performance across sequence lengths.*

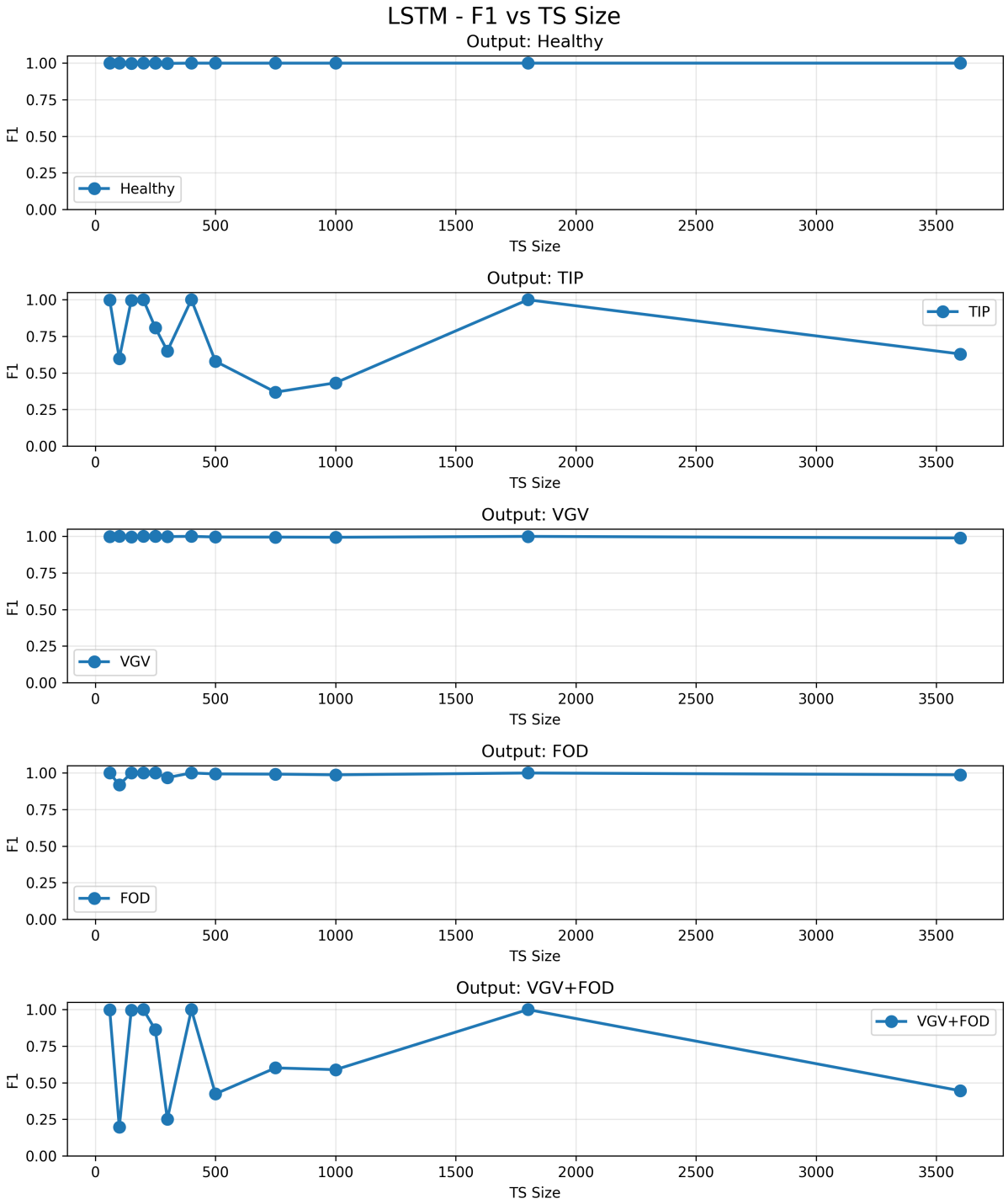

Figure 5.4: *LSTM performance across sequence lengths.*

- On TIP and VGV+FOD faults, performance does not degrade monotonically with increasing sequence length. Instead, oscillatory behavior appears across various sequence lengths.

As expected from the theory derived in Chapter 3, shifting the training horizon changes how gradients accumulate during training and can lead to restructuring of learned attractor dynamics.

Below are some illustrative examples of time-dependent classification to accompany the overall performance and geometry discussed above.

Figure 5.5 illustrates the per-time-step predictions produced by the RNN, GRU, LSTM, and MLP models trained on sequences of length $T = 400$. Two representative test sequences are shown: one ending in a TIP fault and one ending in a VGV+FOD fault. In the latter case, a transient VGV event occurs at timestep $t = 234$ and lasts for 10 time steps.

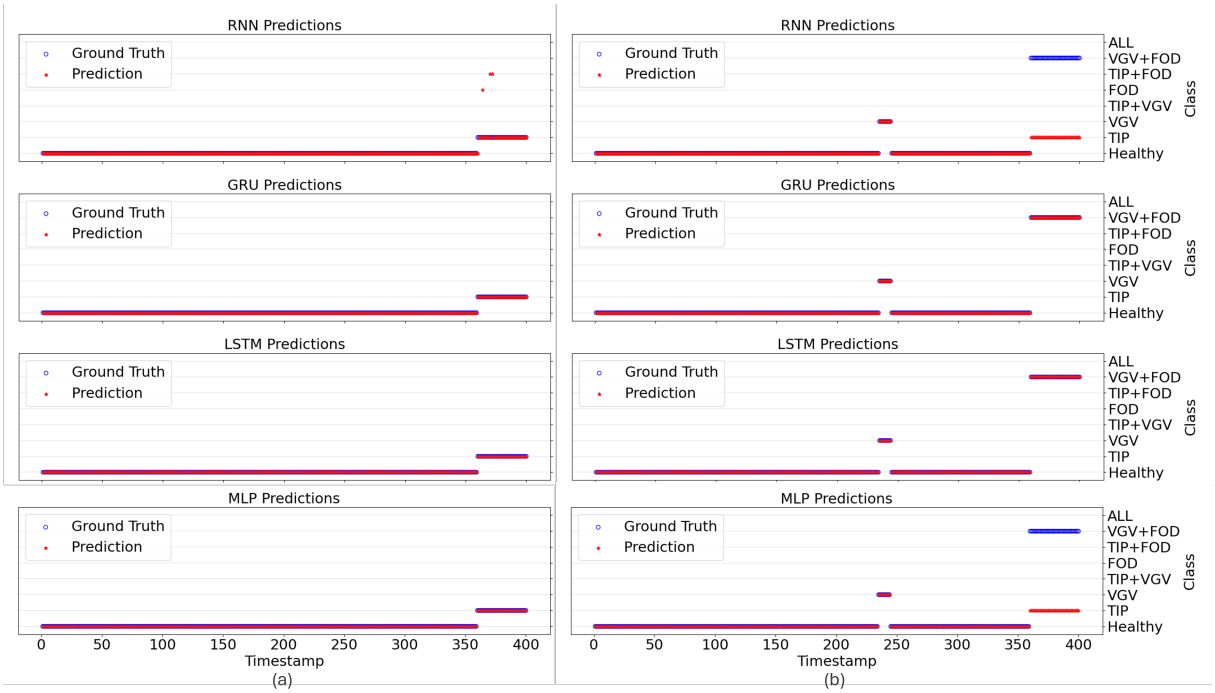

Figure 5.5: *Time-wise classification provided by all models at $T = 400$, for (a) a TIP fault and (b) a VGV+FOD fault timeseries.*

Ground-truth labels are indicated by blue markers, while model predictions are shown in red. For the TIP sequence, all models correctly identify the fault state at the final stage. For the VGV+FOD sequence, a clear architectural distinction emerges: the GRU and LSTM correctly classify the final fault, whereas both the RNN and the MLP misclassify VGV+FOD as TIP.

This figure shows the same trend as observed in the previously shown F1 scores. At $T = 400$, both GRU and LSTM models have achieved an F1 score of 1 for all faults, while the weaker

scores from the RNN for these two faults can be seen here as well. Since MLP lacks temporal memory, it cannot exploit transient events and defaults to steady-state similarity.

Therefore, we see, with time-resolved predictions, that successful discrimination is associated with a persistent internal encoding of the transient precursor.

Figure 5.6 presents the corresponding time-wise predictions for models trained on sequences of length $T = 500$, using the same types of test sequences.

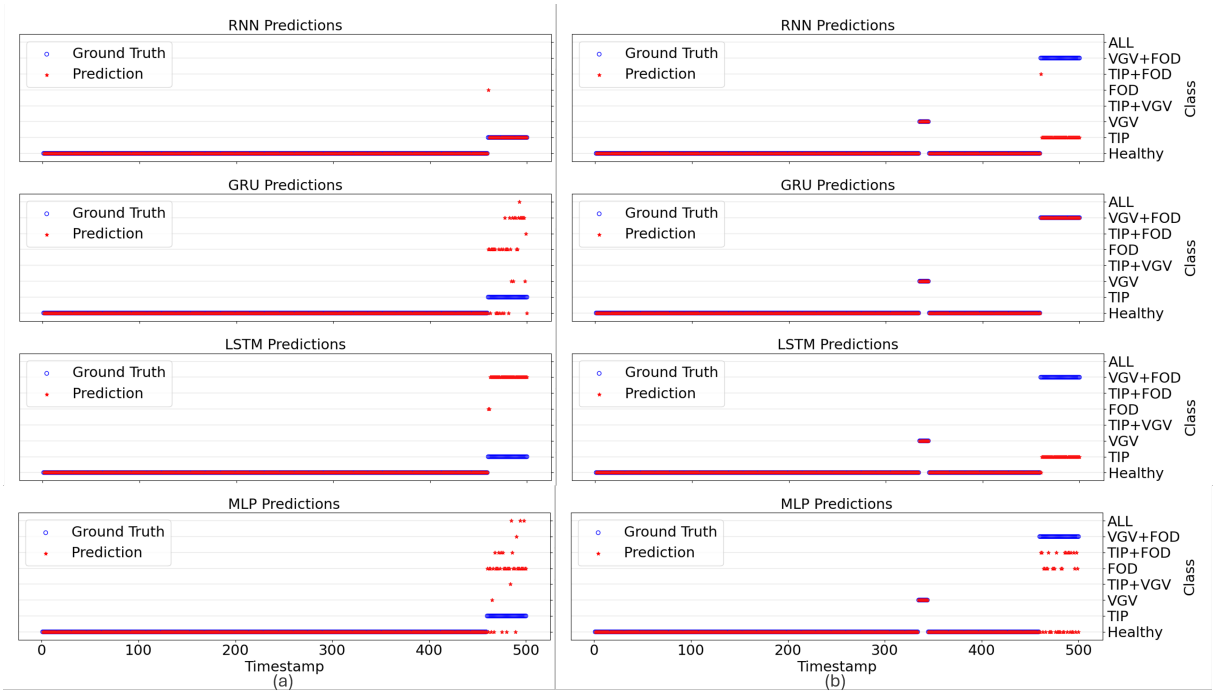

Figure 5.6: *Time-wise classification provided by all models at $T = 500$, for (a) a TIP fault and (b) a VGV+FOD fault timeseries.*

Here, all models generalize very poorly at discriminating VGV+FOD from TIP. While the transient VGV signal occurs earlier in the time series, most models completely lose track of its effect prior to ultimate fault classification. For these reasons, VGV+FOD is often mistaken for TIP.

Recall that the only difference between the models of Figures 5.5 and 5.6 was training horizon length. No modifications to the architecture or optimization were made between the two experiments. We therefore attribute the degradation in performance not to model form or hyperparameter choice, but rather to a reorganization of the learnt internal dynamics as a function of training horizon.

This result establishes an important point: changing sequence length can cause regime shifts in recurrent networks by changing the geometry of internal state trajectories, impacting their ability to retain memory.

5.3 Parametric Sensitivity Analysis

To determine whether observed oscillations arise from insufficient representational capacity, we conducted the parametric study described in Section 4.5.

Hidden dimensionality $n_h \in \{16, 32, 64\}$, depth $L \in \{1, 2\}$, and multiple random seeds $s \in \{7, 21, 42, 84, 168\}$ were evaluated. The following figures (Figure 5.7 to Figure 5.9) present the mean and standard deviation of macro-F1 across the above parameters for the RNN, GRU, and LSTM models, respectively, with sequence length $T = 300$. Similar results are obtained with models using $T = 400$.

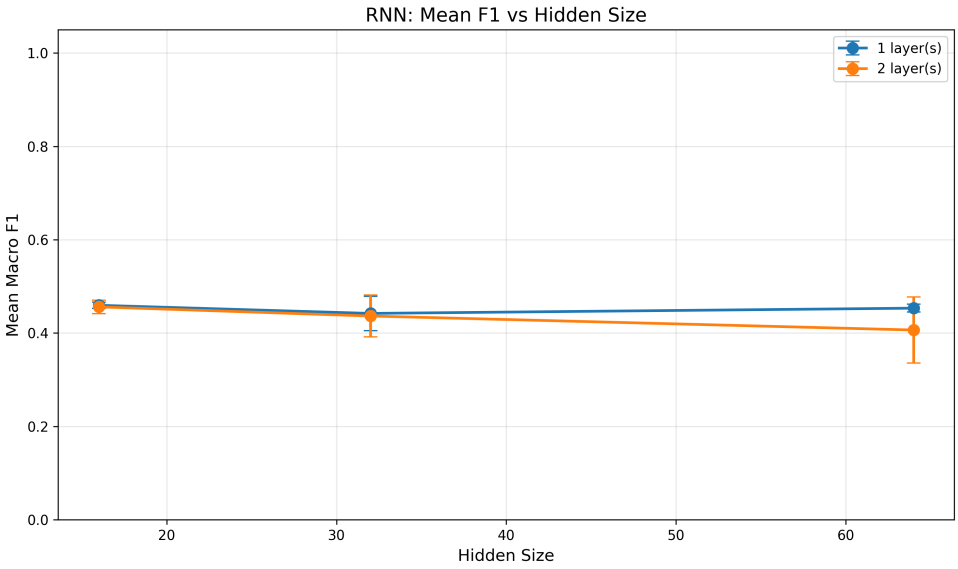

Figure 5.7: *Parametric sensitivity of macro-F1 across hidden sizes and layers, for the RNN model with* $T = 300$.

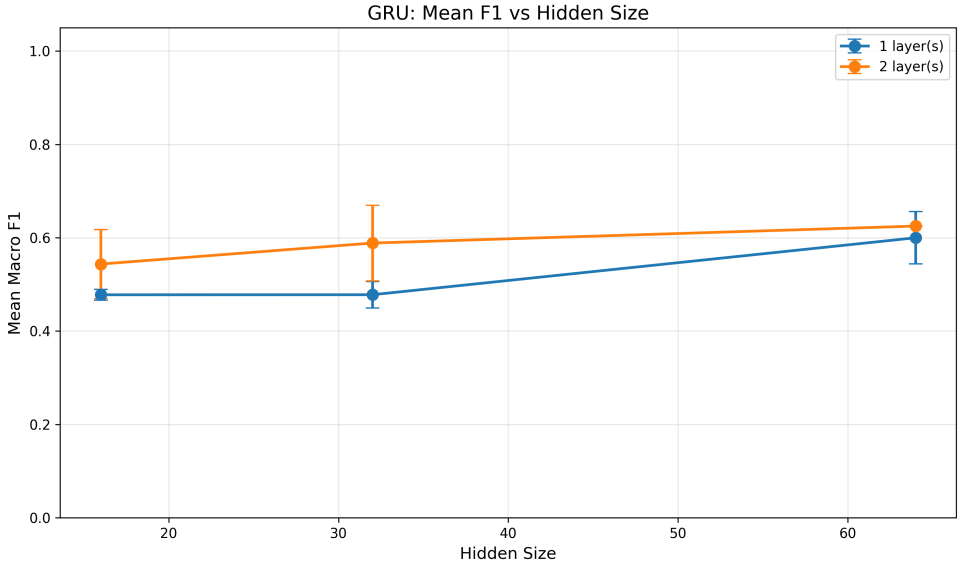

Figure 5.8: *Parametric sensitivity of macro-F1 across hidden sizes and layers, for the GRU model with* $T = 300$.

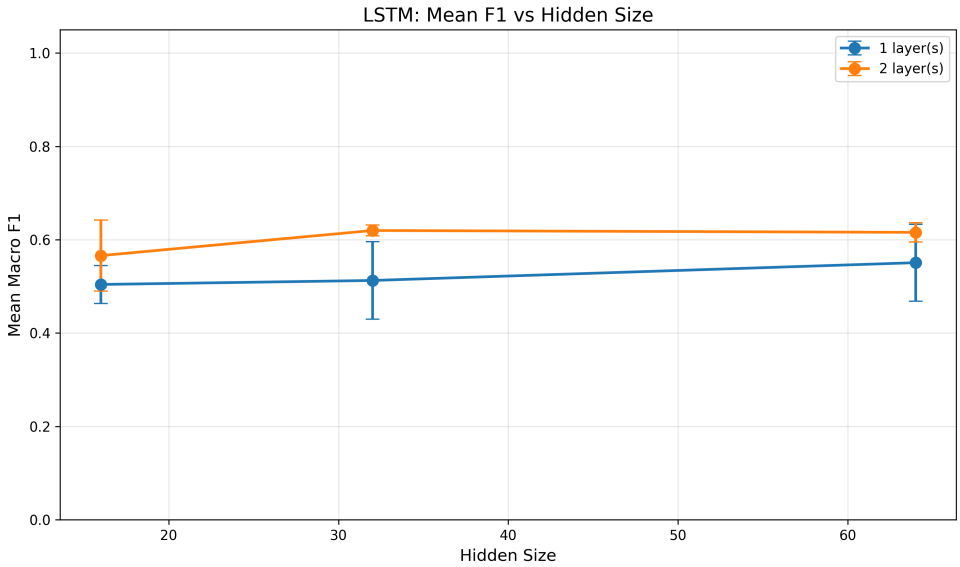

Figure 5.9: *Parametric sensitivity of macro-F1 across hidden sizes and layers, for the LSTM model with* $T = 300$.

These figures show that neither the size of the hidden layer nor the number of layers significantly impacts the overall performance of the models. Similarly, the variance observed across different random seeds remains small compared to the variation across different horizons.

5.4 Gate Activation Dynamics

To investigate the internal mechanism underlying successful and failed memory retention, we examine the temporal evolution of LSTM gate activations and internal states.

Figure 5.10 and Figure 5.11 present representative examples for the same VGV+FOD sequence under two different training horizons.

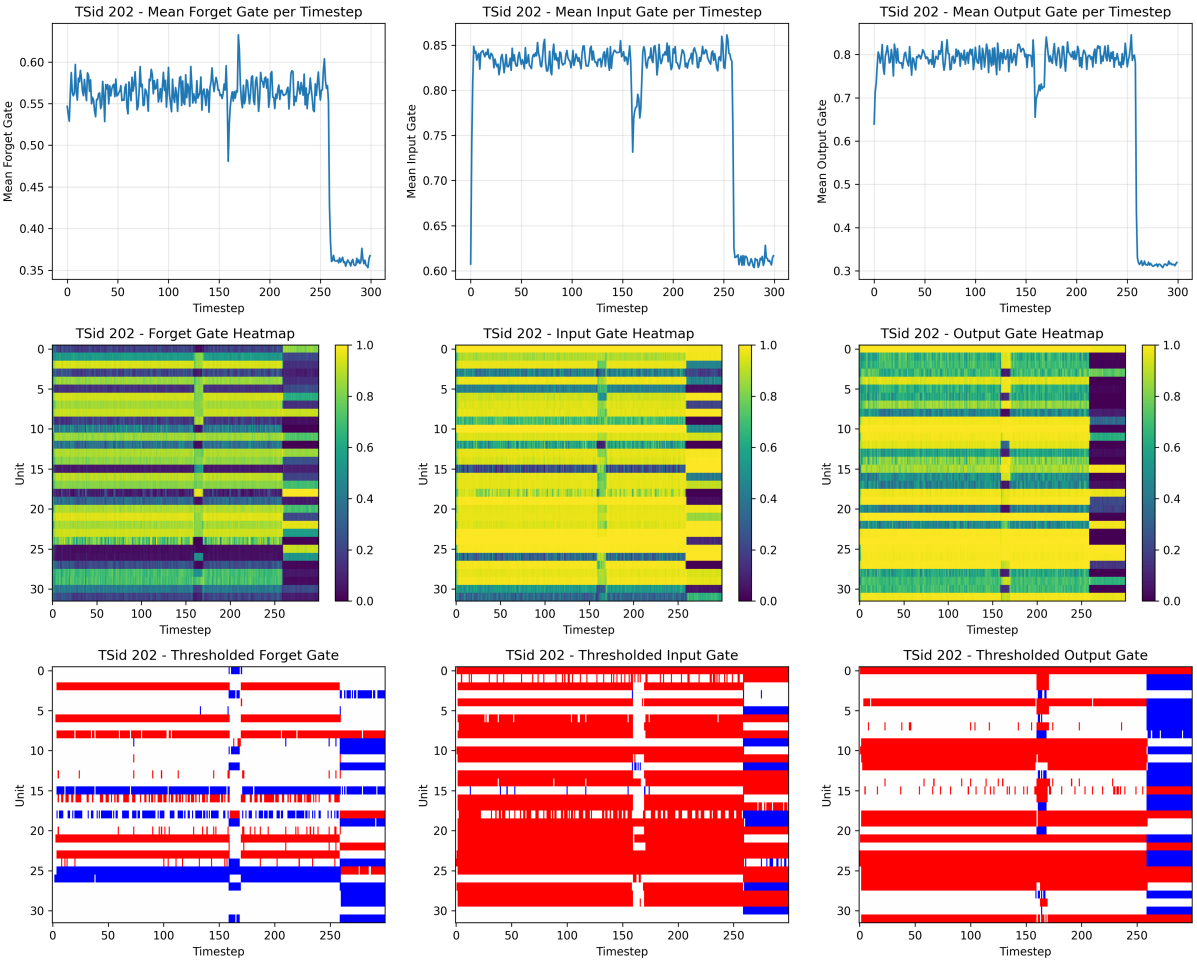

Figure 5.10: *Mean gate activations and heatmaps for LSTM for a VGV+FOD sequence with* $T = 300$.

The first row displays, from left to right: Mean forget gate activation, Mean input gate activation, Mean output gate activation, computed as the average across the 32 hidden units at each time step.

At time step 160, a transient VGV event occurs for 10 time steps, followed by a return to healthy operation, and finally the persistent VGV+FOD fault.

The second row shows full gate activations (32 units $\times$ time) as heatmaps.

The third row presents thresholded heatmaps, where values:

- 0–0.1 are shown in blue,
- 0.1–0.9 in white,
- 0.9–1 in red,

to reveal possible saturation effects.

From this figure, we see no systematic gate saturation. Furthermore, we notice how, after the transient VGV event, both the mean trajectories and the full heatmaps return to patterns indistinguishable from those in the healthy regime before the transient occurred.

This implies that the transient perturbation did not cause a lasting shift in the dynamics trajectory through the internal state-space. Therefore, when we experience the final VGV+FOD fault, the network no longer has the information to distinguish it from the TIP fault.

Figure 5.11 shows the gate activations for an LSTM model trained on sequences of length $T = 400$.

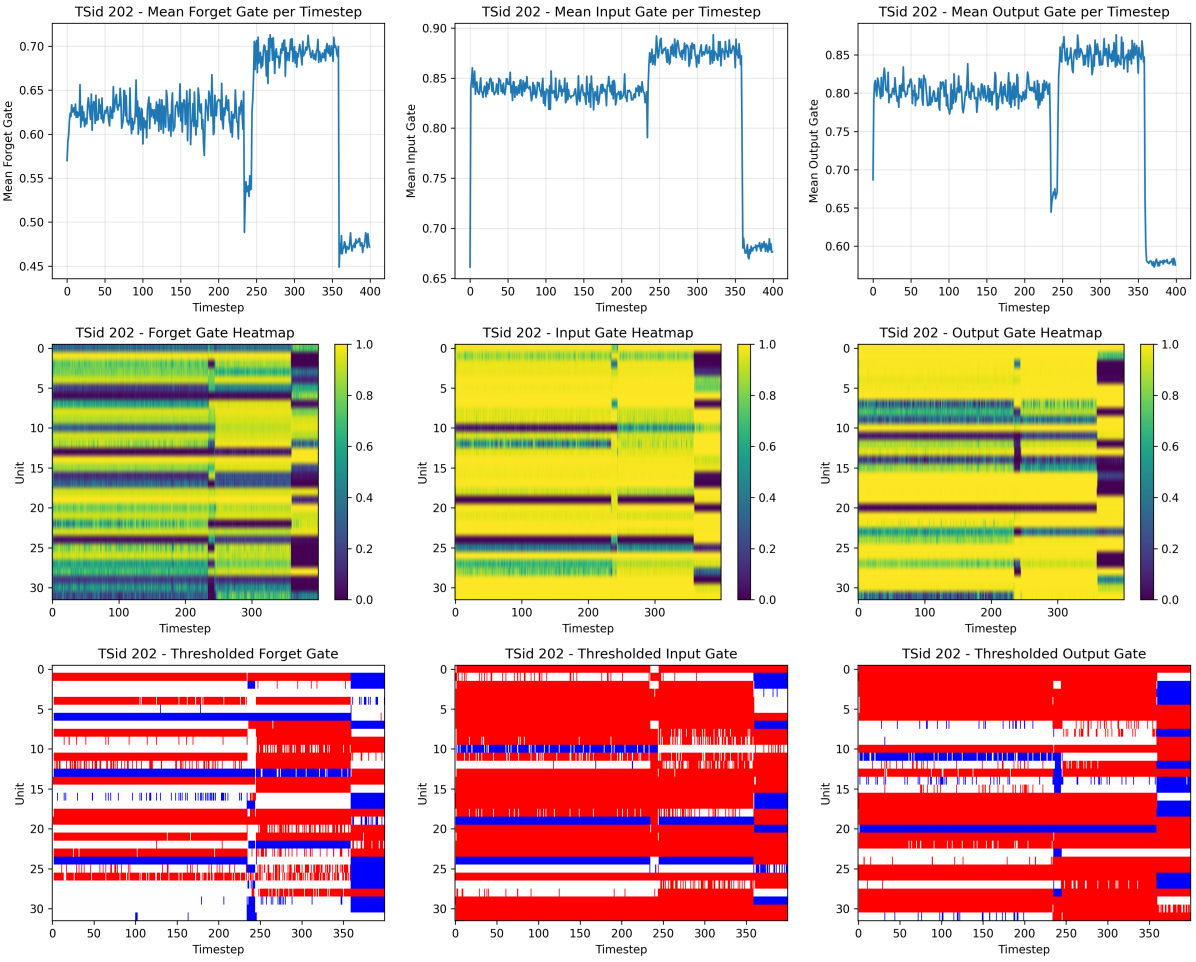

Figure 5.11: *Mean gate activations and heatmaps for LSTM for a VGV+FOD sequence with* $T = 400$.

In contrast to Figure 5.10, gate activations after the transient VGV do not return to their original values, and the heatmap shows persistent structural changes among hidden units.

This suggests that the transient event causes a lasting shift in the internal attractor state. Upon experiencing the final VGV+FOD fault, the network remembers the activity pattern of the precursor, allowing for proper discrimination.

Note that while Figures 5.10 and 5.11 are examples of LSTM networks, this phenomenon can be seen in GRUs and with other sequence lengths.

5.5 Hidden-State Geometry: PCA Analysis

To examine the global organization of recurrent state-space dynamics, Principal Component Analysis (PCA) was applied as described in Section 4.6.3. A unified PCA transformation was trained separately for each model (RNN, GRU, and LSTM) using hidden-state (or cell-state for LSTM) activations extracted from training sequences with $T = 200, 300, 400, 500$. This transformation was applied to test sequences.

The purpose of this unified approach is to ensure that differences across horizons are interpreted in the same latent coordinate system.

5.5.1 Representative Trajectories ($T = 300$ vs $T = 400$)

The following figures (Figure 5.12 to Figure 5.14) show the PCA projection of hidden-state (or cell-state for LSTM) trajectories for four representative test sequences, for RNN, GRU, and LSTM, respectively, and sequences of length $T = 300$.

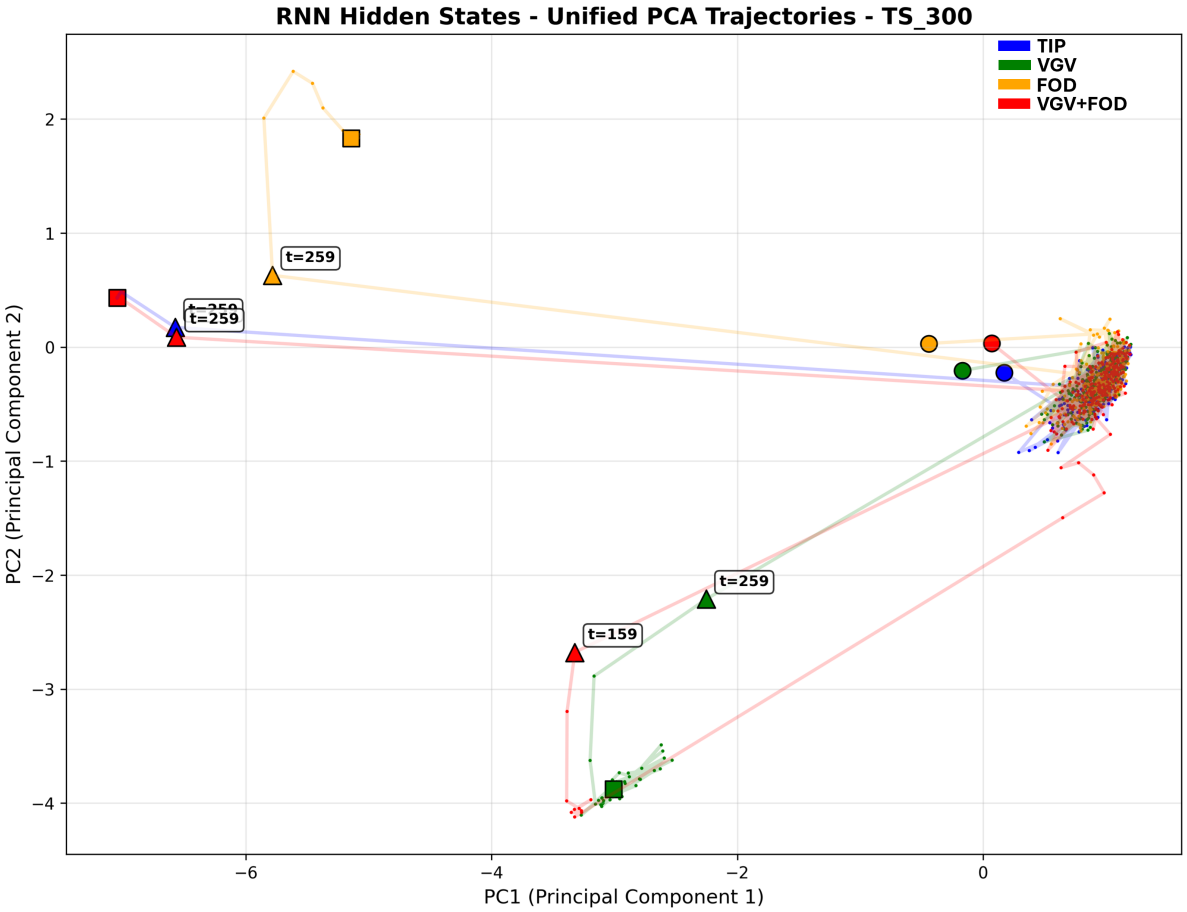

Figure 5.12: *PCA projection of hidden-state trajectories for four representative test sequences, for RNN ($T = 300$).*

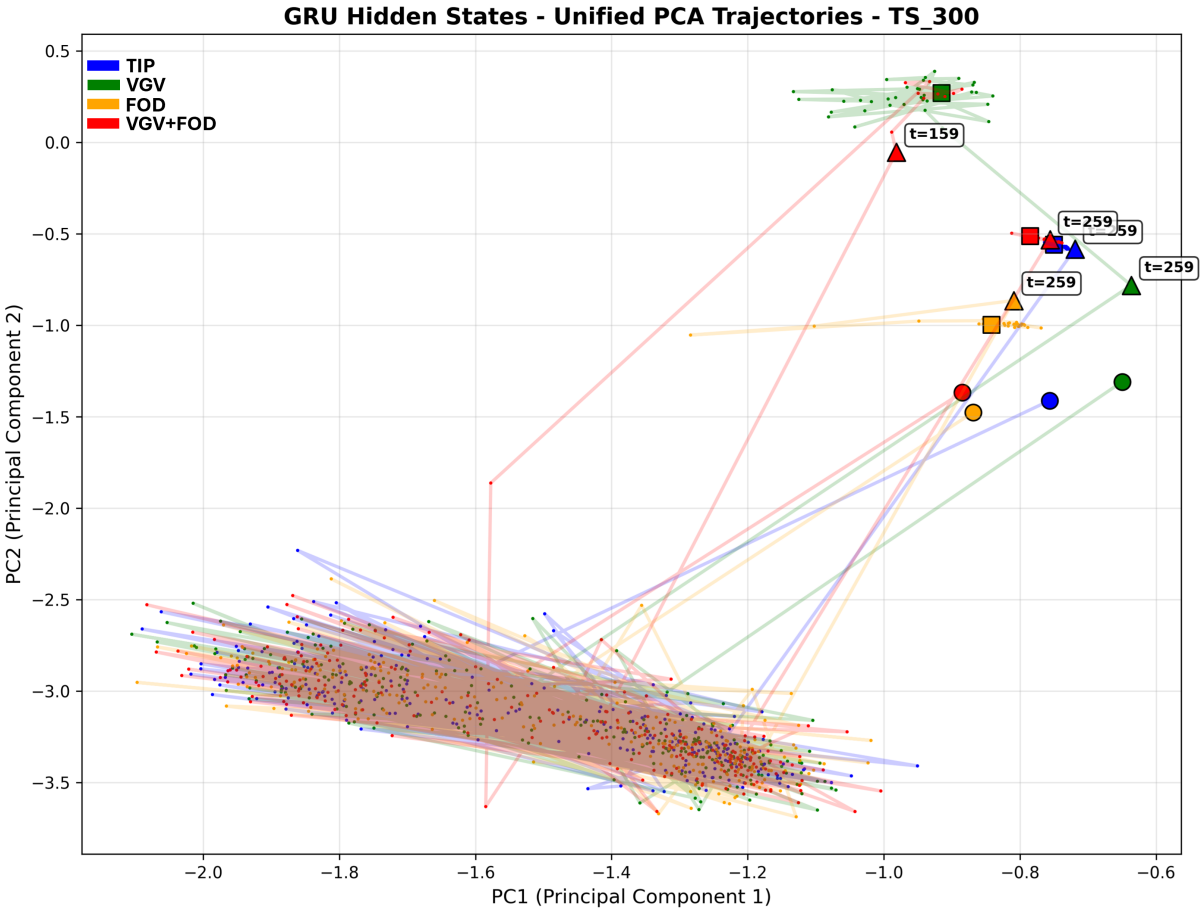

Figure 5.13: *PCA projection of hidden-state trajectories for four representative test sequences, for GRU ($T = 300$).*

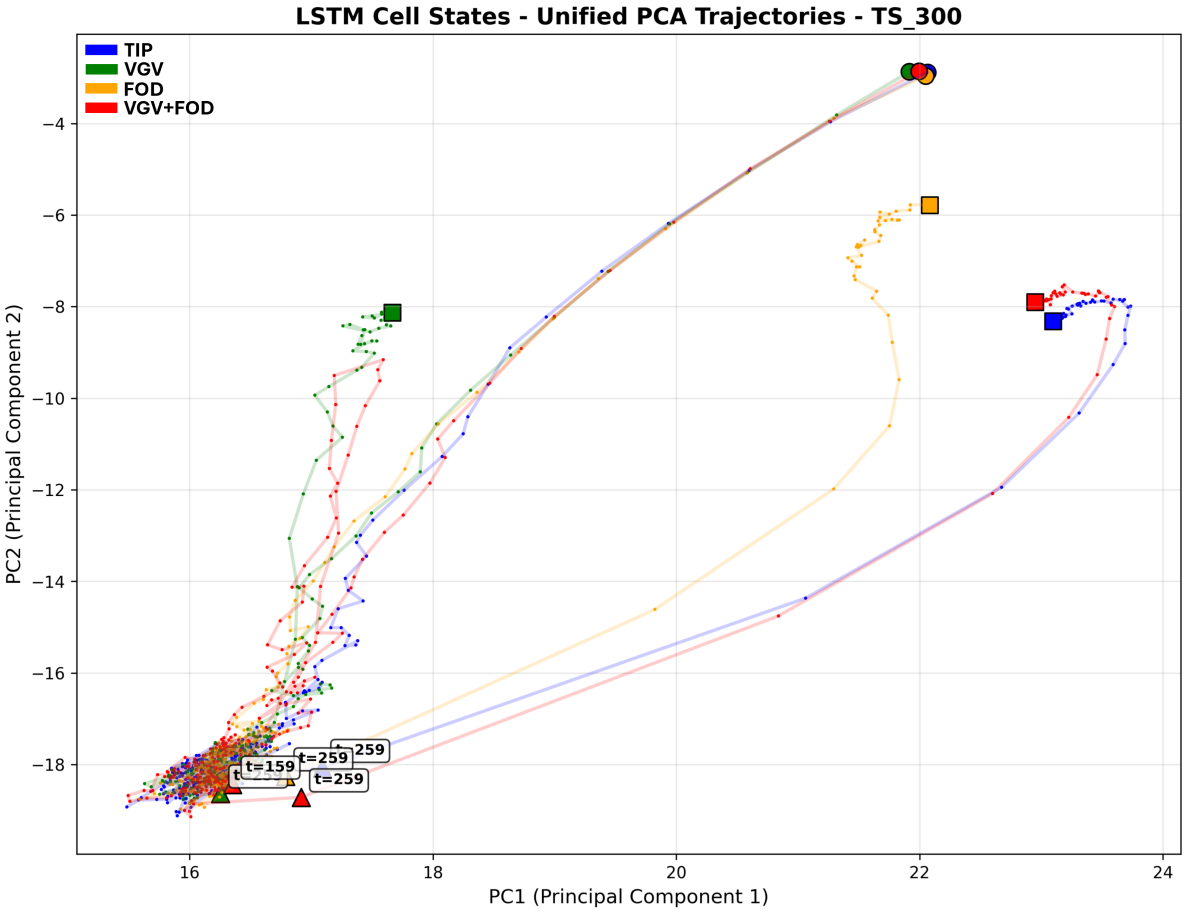

Figure 5.14: *PCA projection of cell-state trajectories for four representative test sequences, for LSTM ($T = 300$).*

In these figures the trajectories of four sequences are visualized, with different colors. The trajectory of a sequence that ends in a TIP fault (blue), a sequence that ends in a VGV fault (green), a sequence that ends in a FOD fault (yellow), and a sequence that ends in a VGV+FOD fault (red).

Each trajectory begins at a large circular marker, evolves during the healthy regime, encounters a fault at $t = 259$ (marked with triangle), and ends at the final square marker. In the case of VGV+FOD fault a transient VGV event occurs at $t = 159$ (marked with triangle), for 10 time-steps, before it returns to the healthy regime.

We can see from these figures that all trajectories start in a relatively compact initial region. During healthy operation, hidden states oscillate around a common basin (which is much more visually apparent in the GRU model plot in the lower-left). When a fault occurs, trajectories move toward class-specific regions.

In contrast to previous examples, we see that, for the VGV+FOD test sequence, during the

transient VGV event, the red trajectory temporarily moves toward the green (VGV) basin. When this transient event ends, the trajectory returns to the healthy basin. However, when the sequence finally experiences a VGV+FOD fault, it converges to the same terminal region as the TIP trajectory did.

The following figures (Figure 5.15 to Figure 5.17) show the PCA projection of hidden-state (or cell-state for LSTM) trajectories for four similar representative test sequences, but this time for models with sequences of length $T = 400$. In this case, each trajectory encounters a fault at $t = 359$, while in the case of VGV+FOD fault the transient VGV event occurs at $t = 234$.

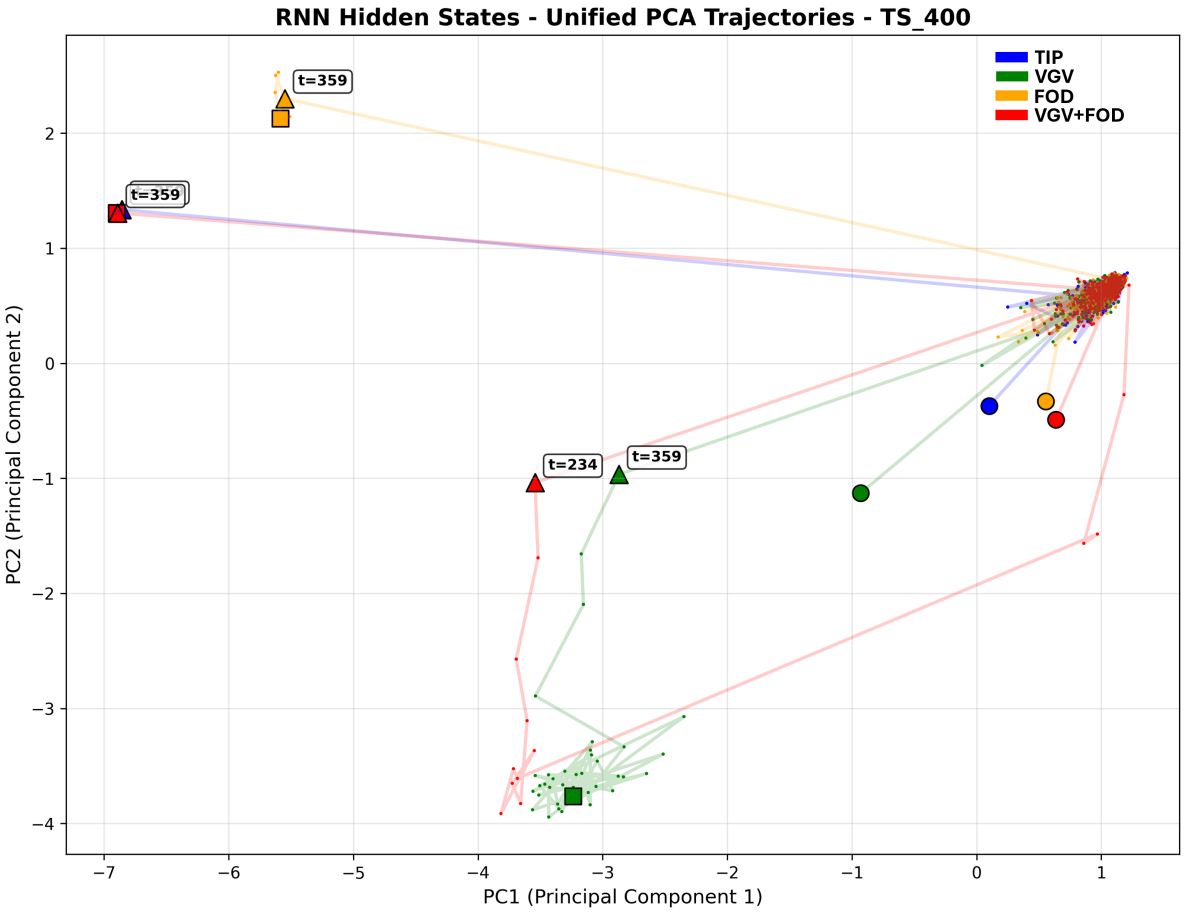

Figure 5.15: *PCA projection of hidden-state trajectories for four representative test sequences, for RNN ($T = 400$).*

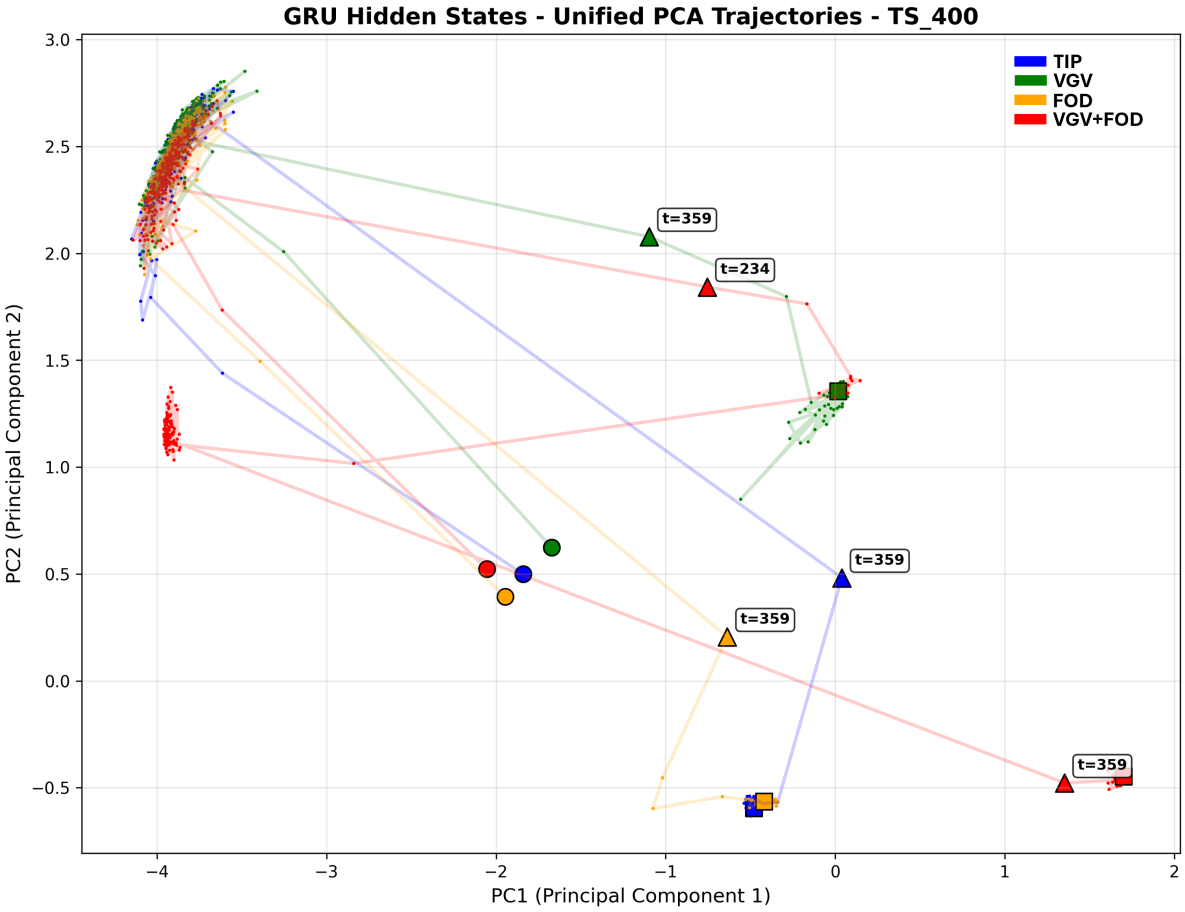

Figure 5.16: *PCA projection of hidden-state trajectories for four representative test sequences, for GRU ($T = 400$).*

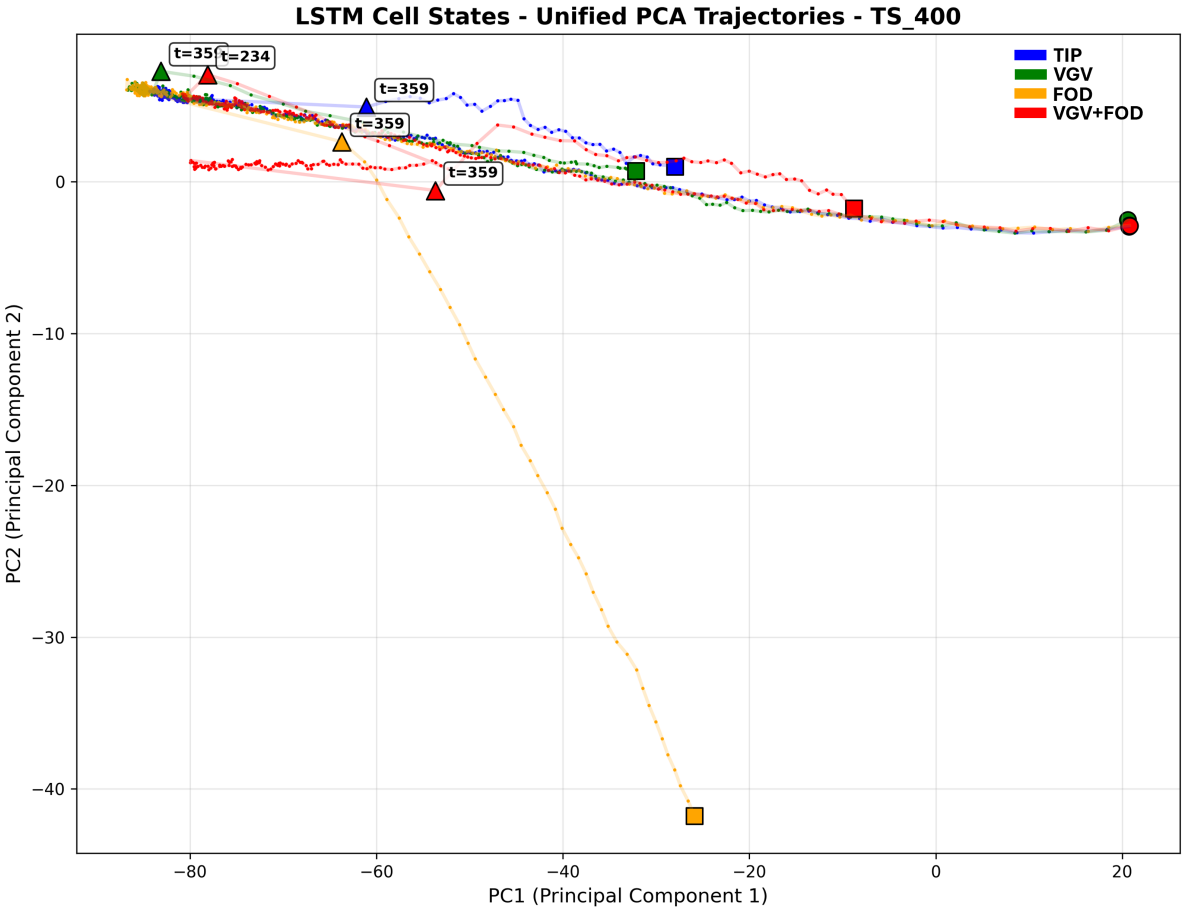

Figure 5.17: *PCA projection of cell-state trajectories for four representative test sequences, for LSTM ($T = 400$).*

From these figures, we note a qualitatively different geometry: The healthy regime is contained within a different basin, but once the transient VGV occurrence happens, the red trajectory does not return to the healthy basin of origin. Instead, it moves to a distinct post-transient healthy region.

This geometric separation is consistent with the improved F1 scores observed at $T = 400$, except for the vanilla RNN, which remains unable to maintain long-term separation — likely as a consequence of its small intrinsic memory.

5.5.2 Full Test-Set Geometry ($T = 300$ vs $T = 400$)

The following figures (Figure 5.18 to Figure 5.20) show the PCA projection of hidden-state (or cell-state for LSTM) of all test-set time steps, for RNN, GRU, and LSTM, respectively, and sequences of length $T = 300$.

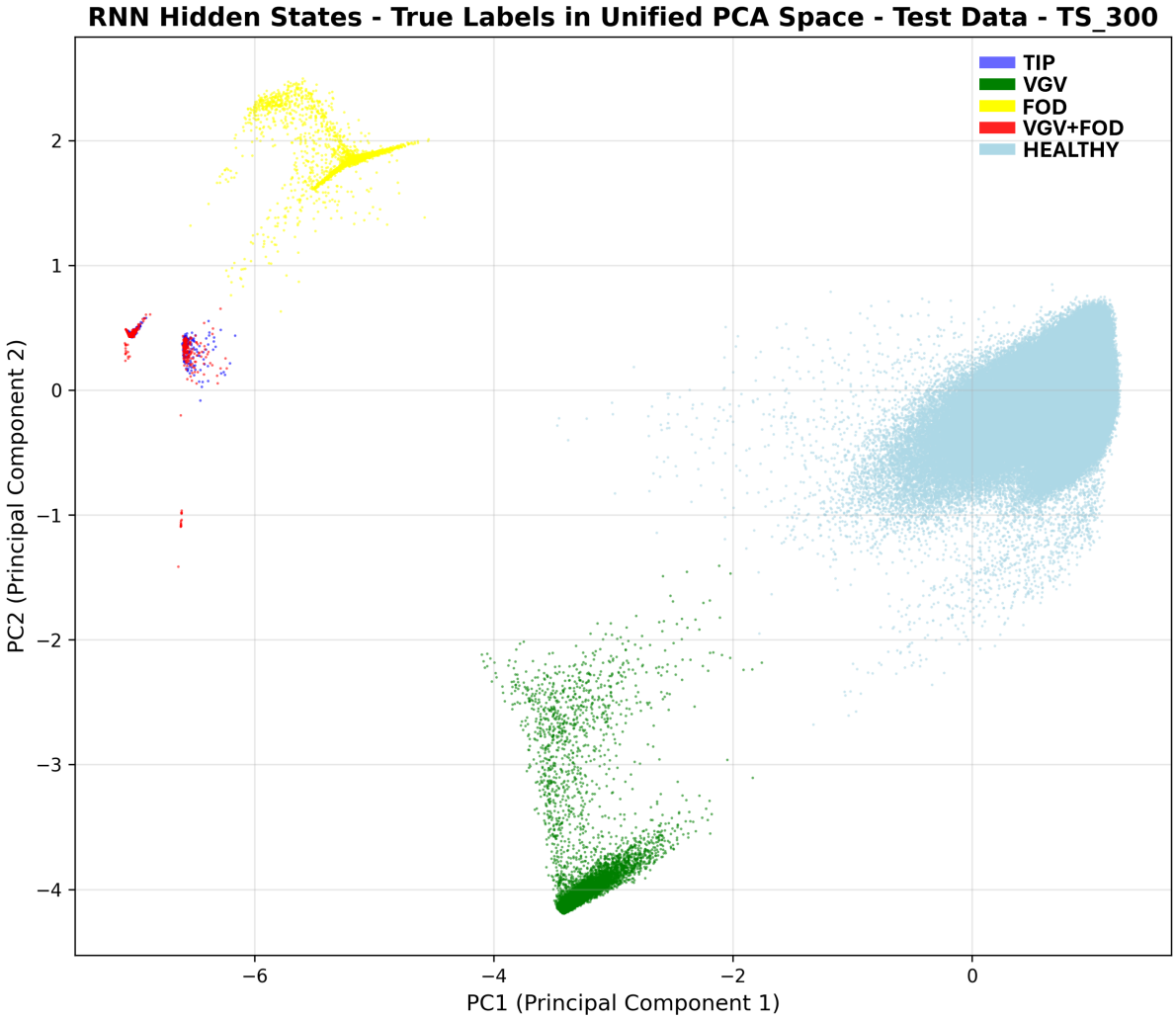

Figure 5.18: *Unified PCA projection of hidden-state of all test-set time steps, for RNN ($T = 300$).*

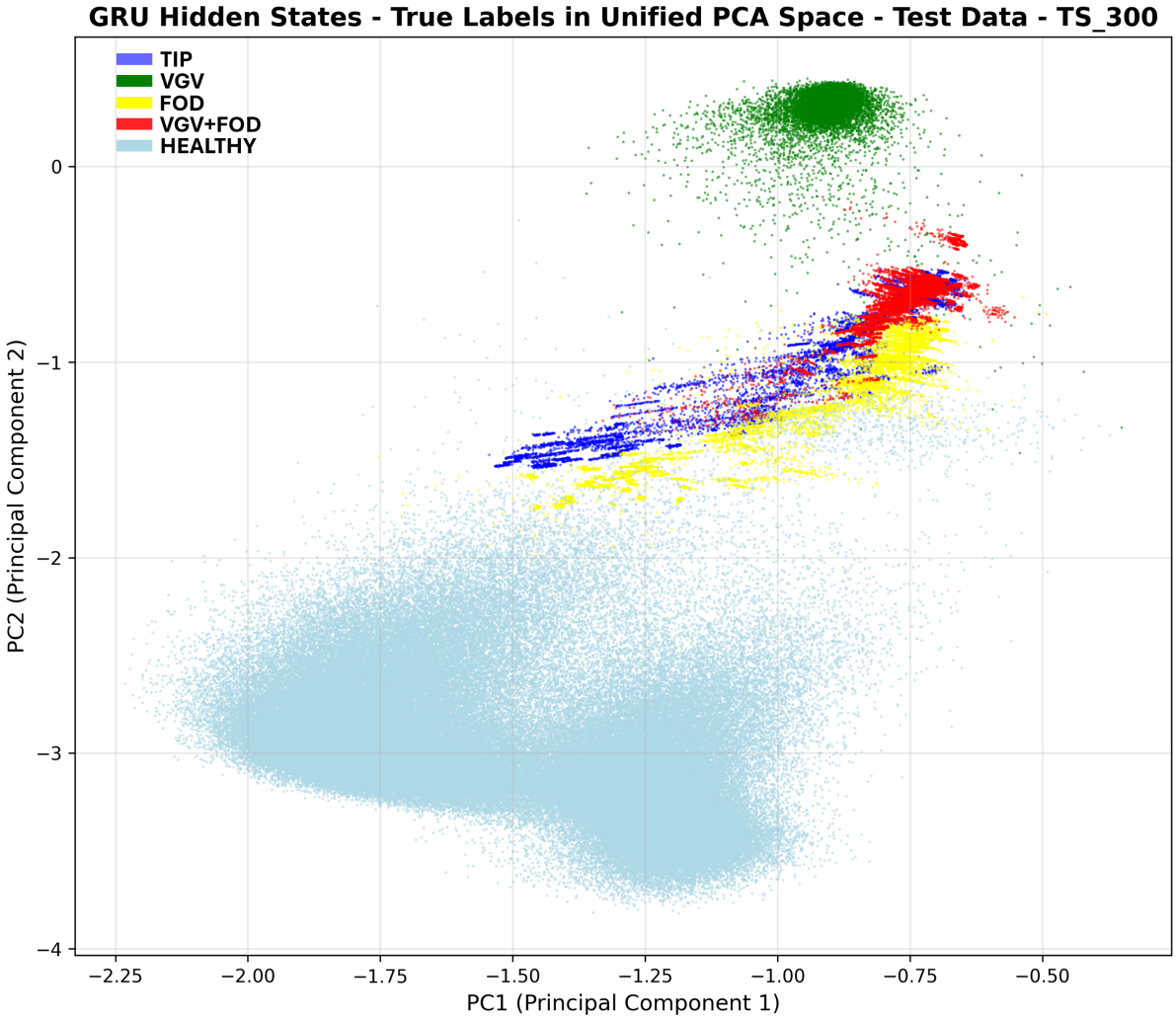

Figure 5.19: *Unified PCA projection of hidden-state of all test-set time steps, for GRU ($T = 300$).*

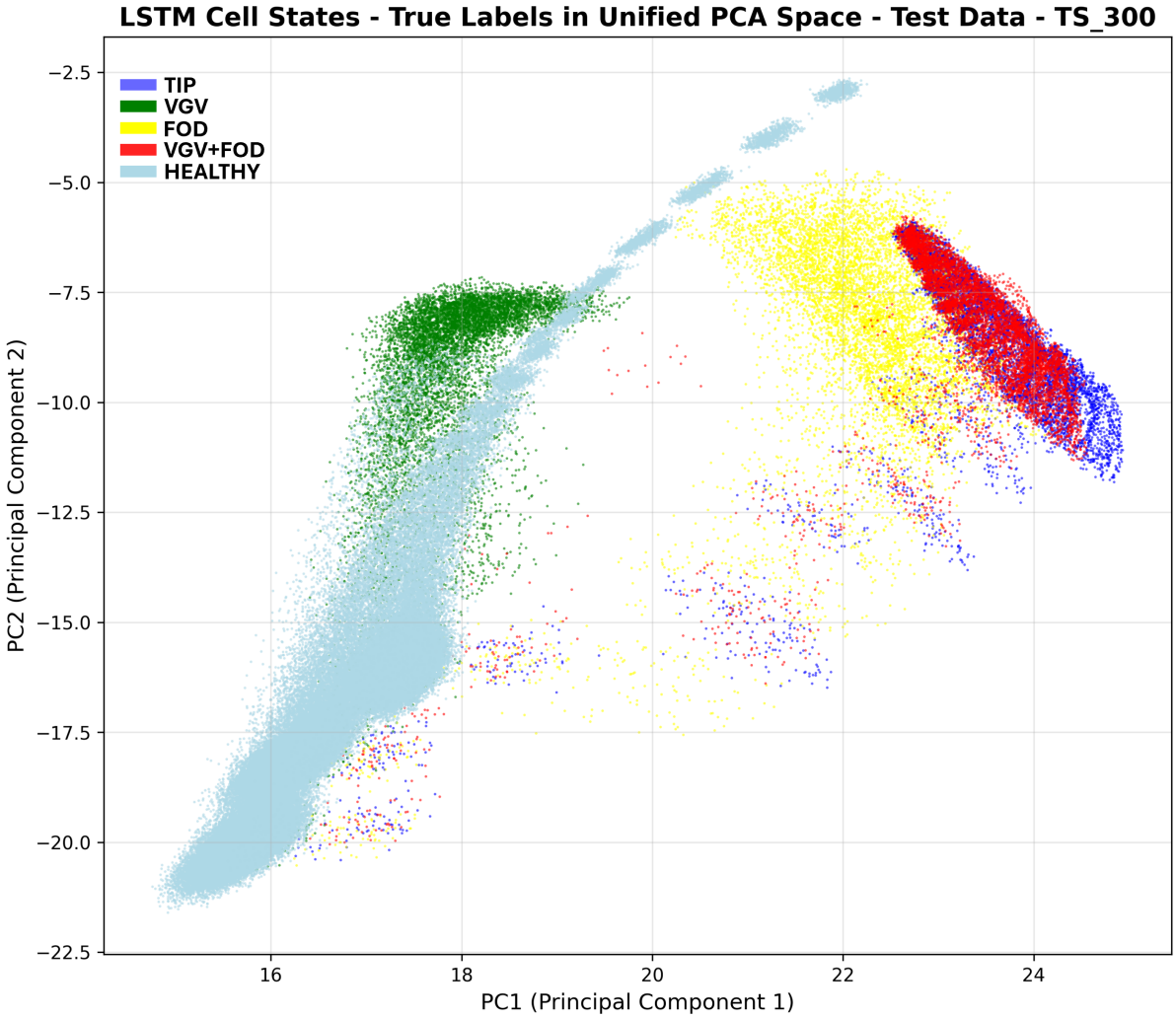

Figure 5.20: *Unified PCA projection of cell-state of all test-set time steps, for LSTM ($T = 300$).*

In these figures, the projection of the hidden-states (cell-states for LSTM) is visualized, with different colors. Each point is colored by label: Healthy (light blue), TIP (blue), VGV (green), FOD (yellow), VGV+FOD (red).

From these figures, we can see that TIP and VGV+FOD regions overlap significantly. Also, the healthy state forms a single unified basin, and no geometric separation exists between healthy states before and after the transient.

This confirms attractor merging at both the steady-state fault level and the healthy regime level.

Similarly, the following figures (Figure 5.21 to Figure 5.23) show the PCA projection of hidden-state (or cell-state for LSTM) of all test-set time steps, but for models with sequences of length $T = 400$.

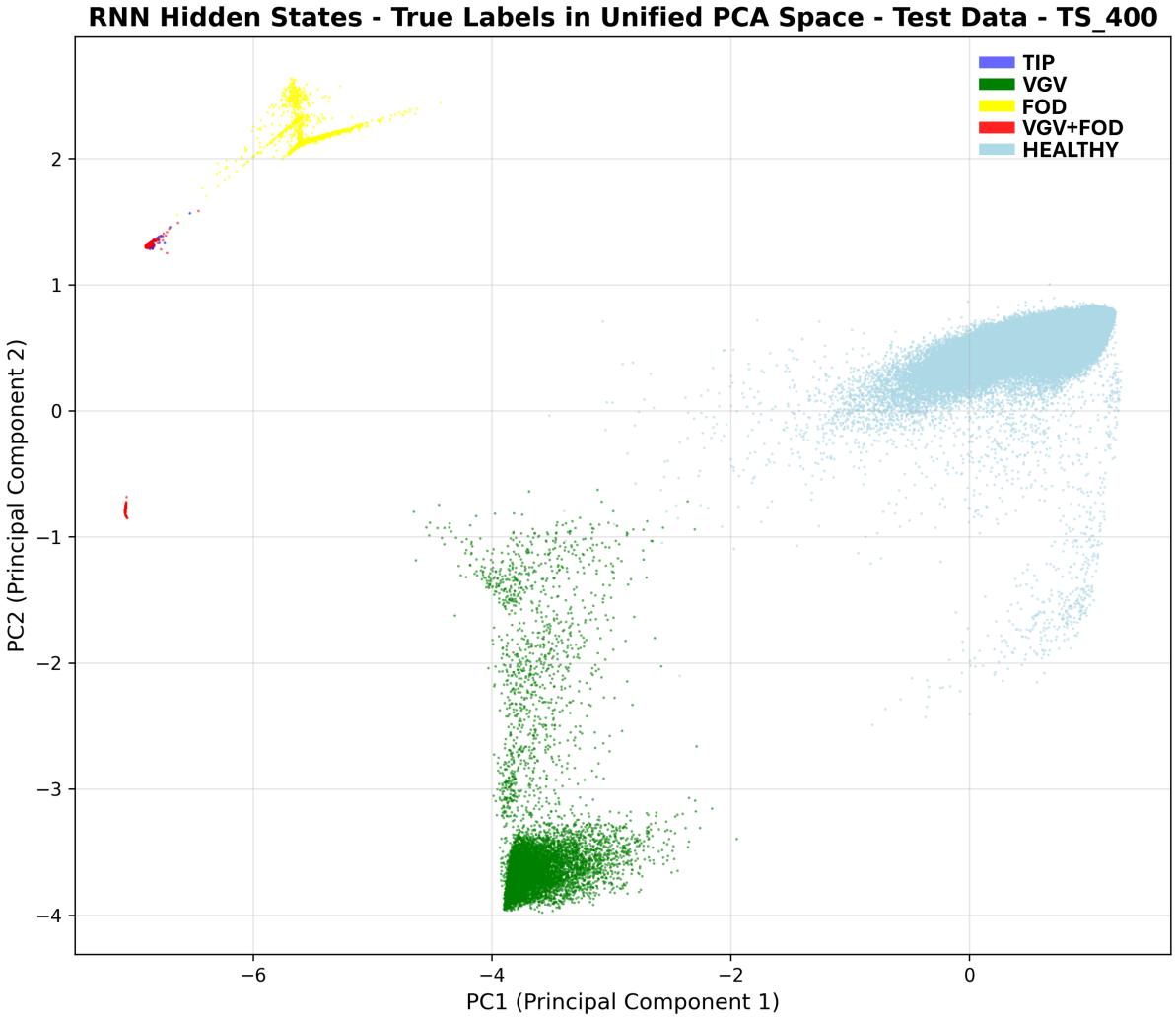

Figure 5.21: *Unified PCA projection of hidden-state of all test-set time steps, for RNN ($T = 400$).*

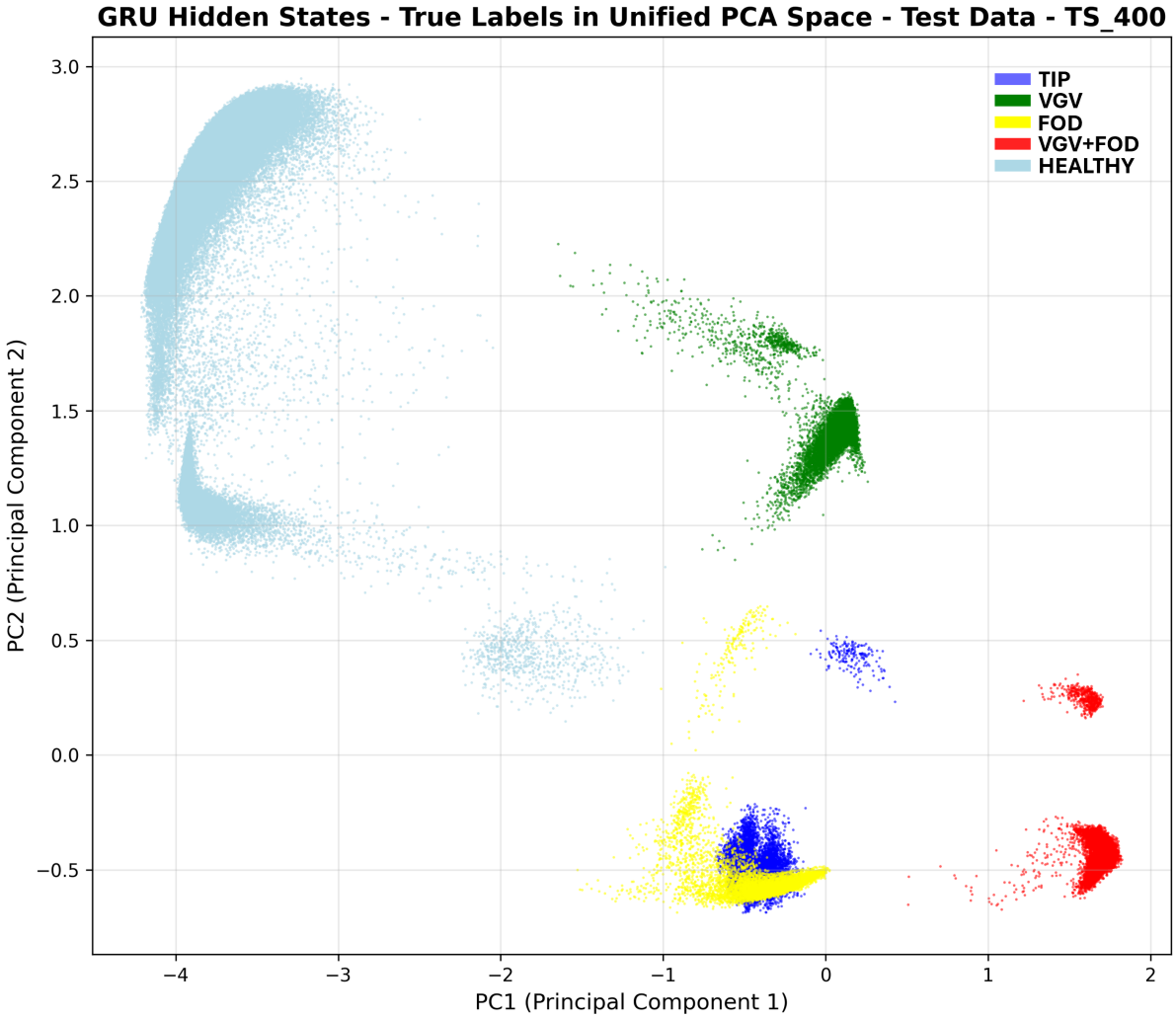

Figure 5.22: *Unified PCA projection of hidden-state of all test-set time steps, for GRU ($T = 400$).*

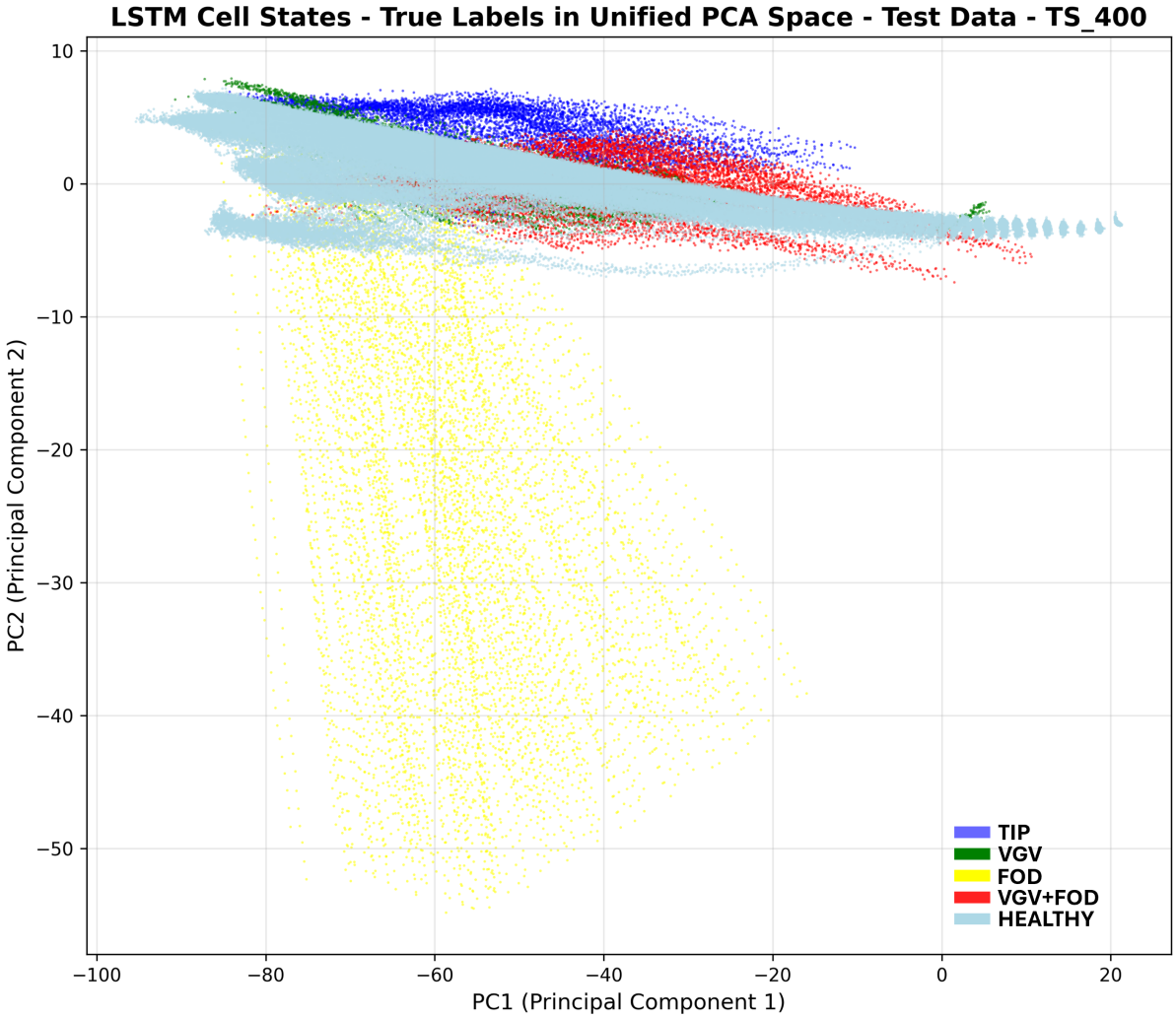

Figure 5.23: *Unified PCA projection of cell-state of all test-set time steps, for LSTM ($T = 400$).*

From these figures we note a qualitatively different geometry: TIP and VGV+FOD occupy clearly distinct regions. The healthy state splits into three subregions (Initial time-step region, pre-transient healthy basin, post-transient healthy basin). This geometric separation is consistent with the improved F1 scores observed at $T = 400$.

The above PCA analysis demonstrates that performance oscillations are associated with qualitative restructurings of the learned state-space topology. The fact that this happens across RNN, GRU, and LSTM suggests that this is a dynamical property independent of the architecture.

5.6 Nonlinear Manifold Structure: UMAP Analysis

To verify that the geometric phenomena observed in PCA are not due to linear projection, Uniform Manifold Approximation and Projection (UMAP) was applied using the unified procedure described in Section 4.6.4, similar to the procedure followed for the PCA analysis. Separate unified UMAP models were trained for RNN, GRU, and LSTM using hidden states (or cell-state for LSTM) from training sequences with $T = 200, 300, 400, 500$.

5.6.1 Representative Trajectories ($T = 400$ vs $T = 500$)

The following figures (Figure 5.24 to Figure 5.26) show the UMAP projection of hidden-state (or cell-state for LSTM) trajectories for four representative test sequences, for models with sequences of length $T = 400$, that correspond to the results of the PCA projection presented in Figures 5.15 to 5.17.

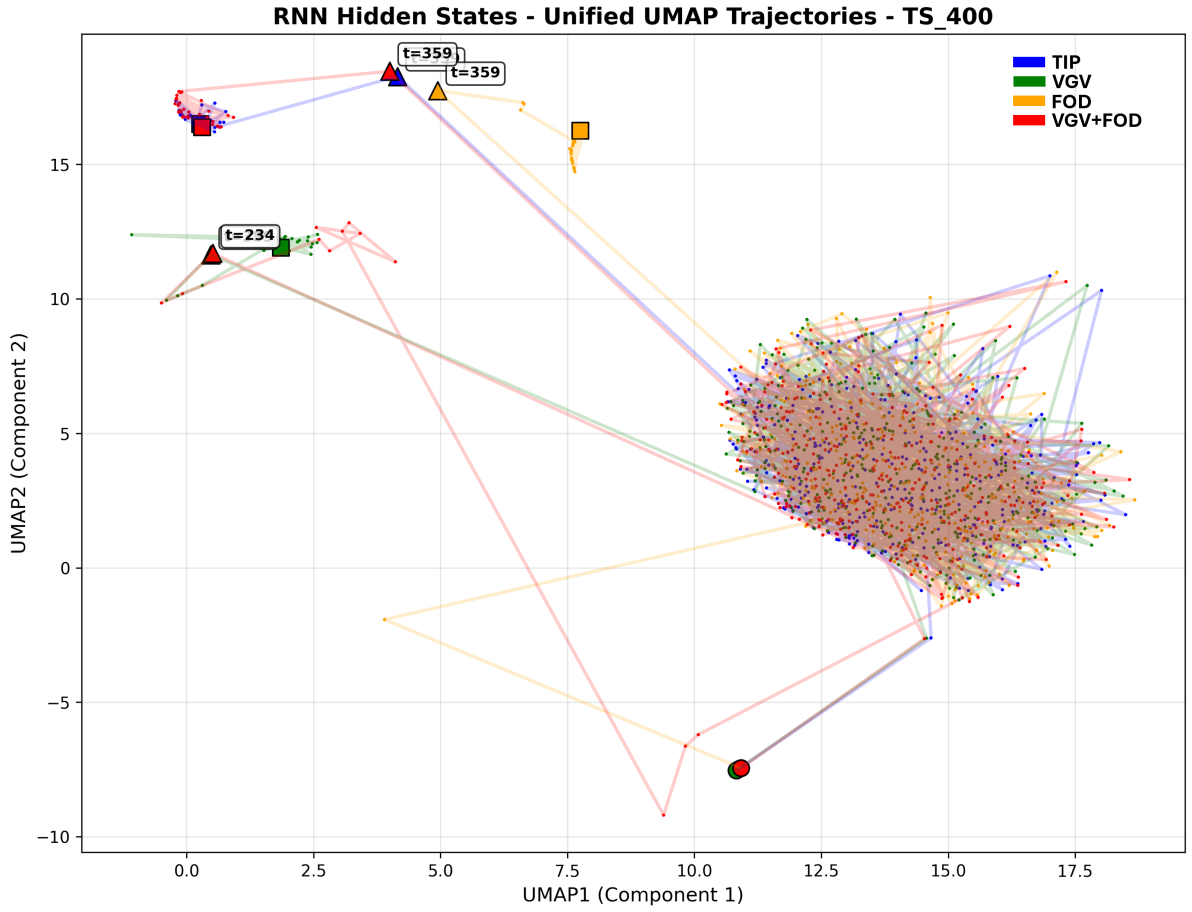

Figure 5.24: *UMAP projection of hidden-state trajectories for four representative test sequences, for RNN ($T = 400$).*

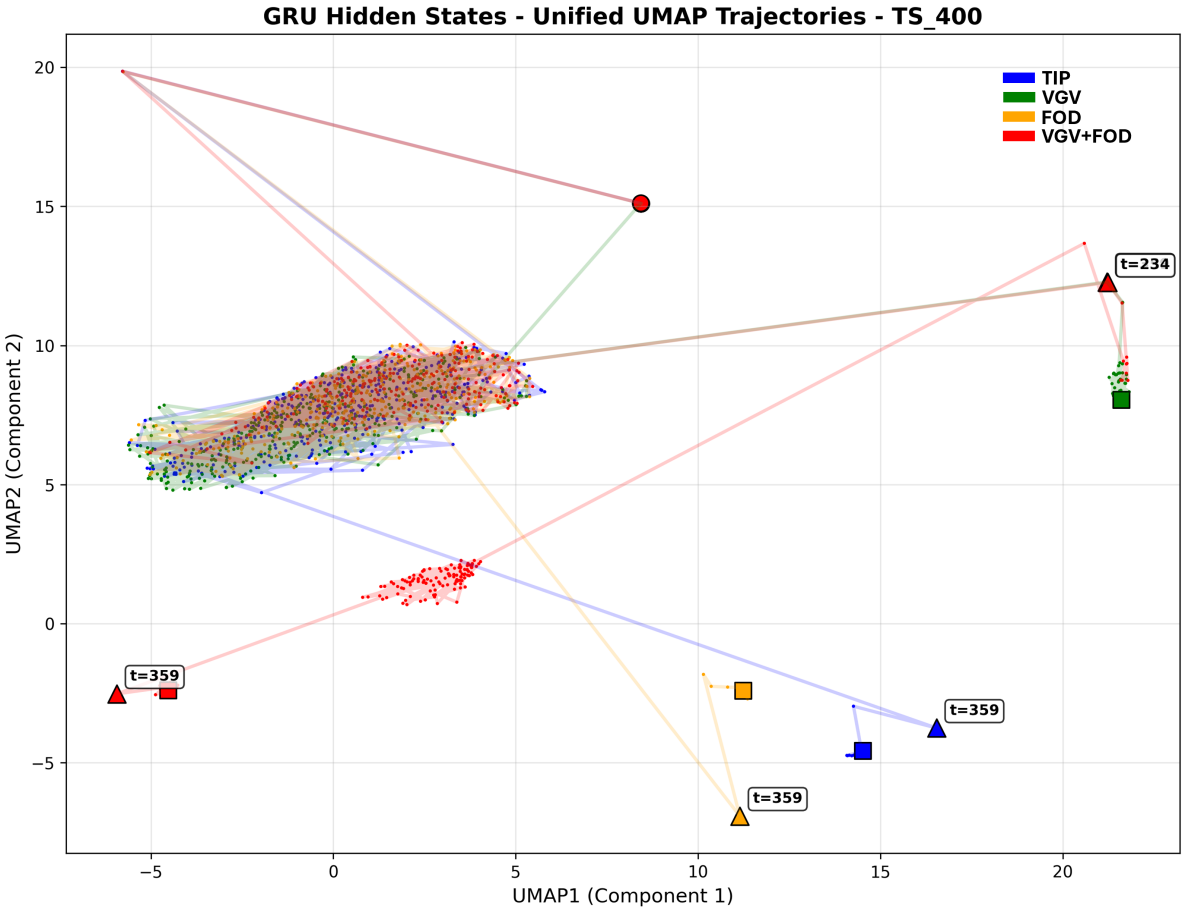

Figure 5.25: *UMAP projection of hidden-state trajectories for four representative test sequences, for GRU ($T = 400$).*

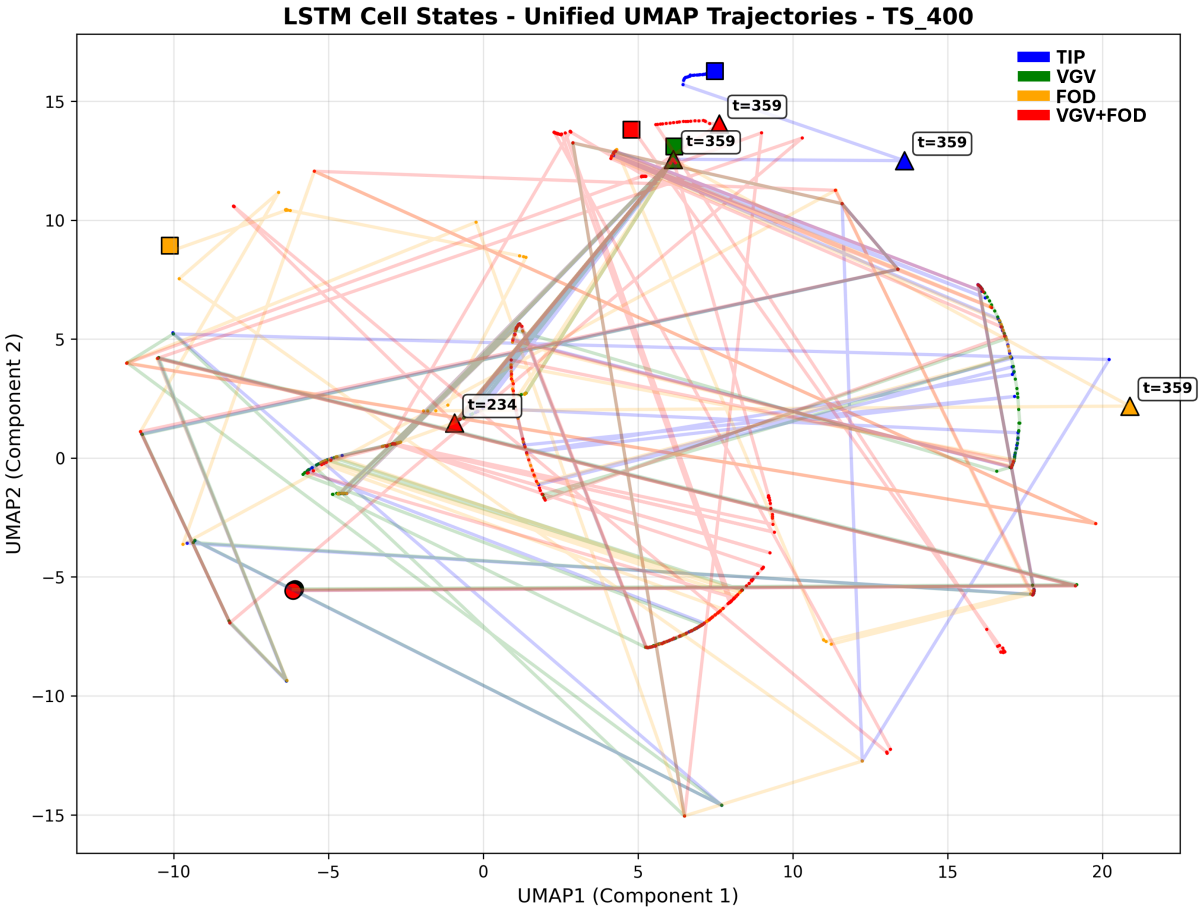

Figure 5.26: *UMAP projection of cell-state trajectories for four representative test sequences, for LSTM ($T = 400$).*

From these figures we note that GRU exhibits a trajectory separation analogous to PCA, while LSTM presents a not clear healthy basin, but distinct terminal regions are still observable.

The following figures (Figure 5.27 to Figure 5.29) show the UMAP projection of hidden-state (or cell-state for LSTM) trajectories for four representative test sequences, for models with sequences of length $T = 500$.

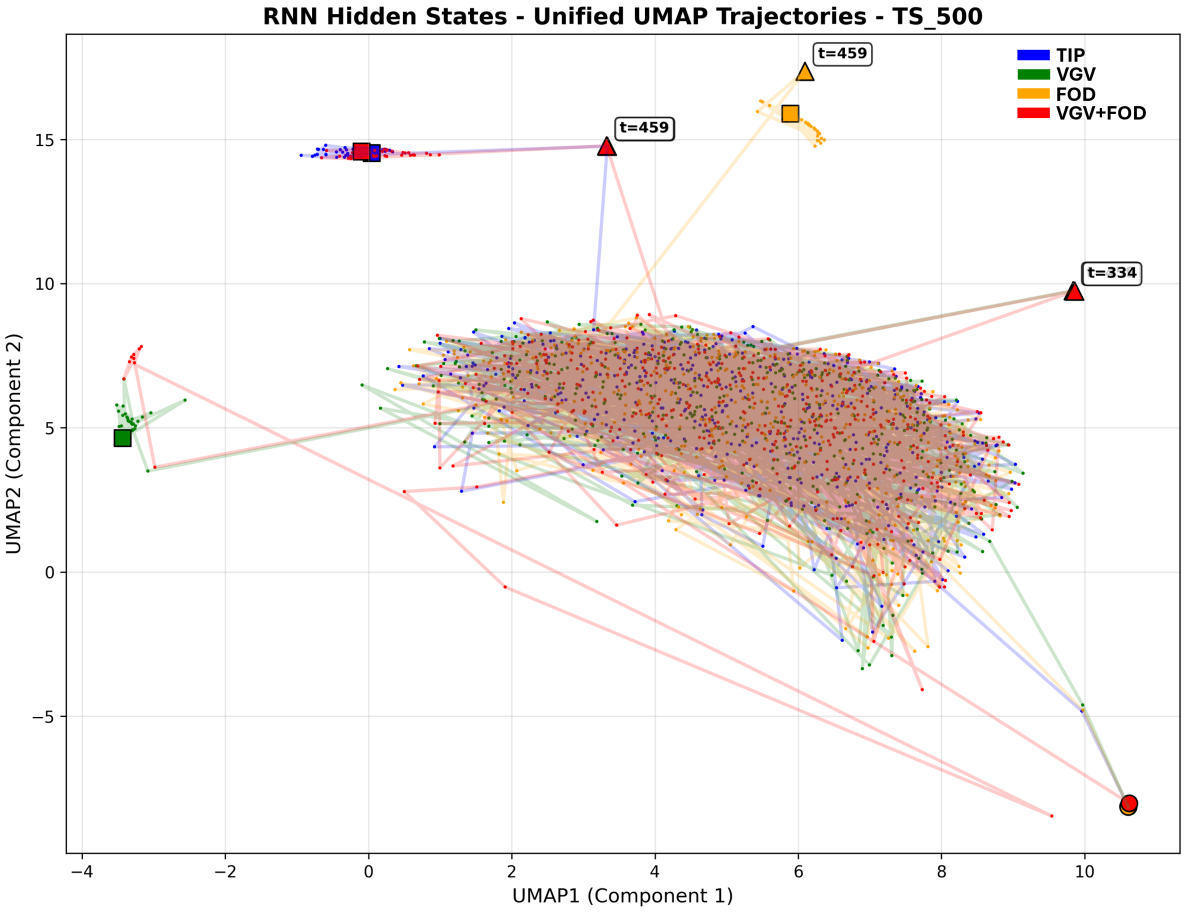

Figure 5.27: *UMAP projection of hidden-state trajectories for four representative test sequences, for RNN ($T = 500$).*

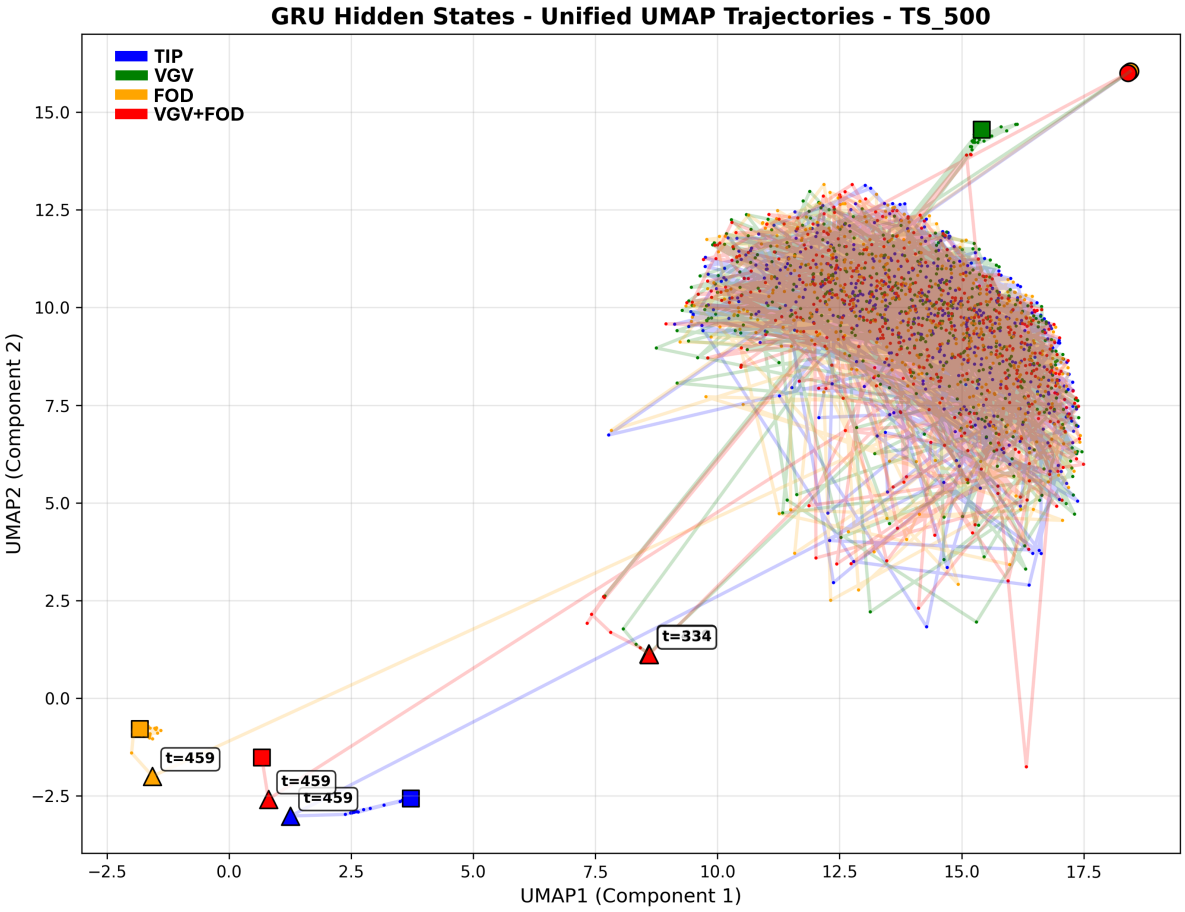

Figure 5.28: *UMAP projection of hidden-state trajectories for four representative test sequences, for GRU ($T = 500$).*

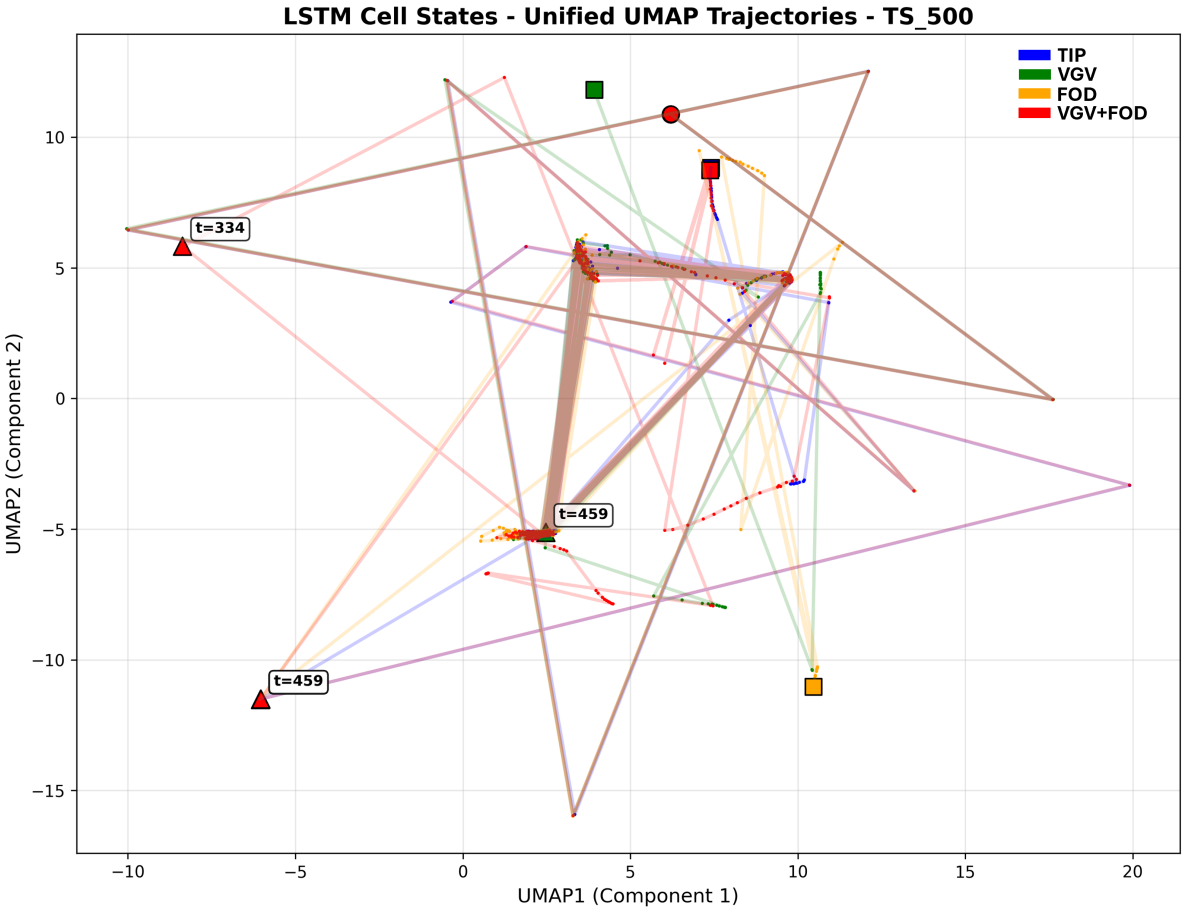

Figure 5.29: *UMAP projection of cell-state trajectories for four representative test sequences, for LSTM ($T = 500$).*

From these figures, it is observed that the healthy states collapse mostly onto the manifold from before the transient, meaning that they return to the same macroscopic basin of attraction in the projected space. Notice that for both the RNN and LSTM, the terminal clusters associated with TIP and VGV+FOD overlap heavily, consistent with the degradation observed in classification performance at this horizon.

However, for the GRU, we see slightly more complex behavior. While the behavior after the transient still appears to return to the same healthy basin in the low-dimensional projection, the final state ends up in partially distinct terminal regions. This means that the return to the healthy manifold, after the transient event, does not necessarily imply the complete erasure of this transient information. Some information —possibly encoded in higher-dimensional components not fully captured by the two-dimensional embedding— may still influence the final state transition.

Importantly, this geometric behavior aligns with the performance results. In the sequence lengths where recurrent models generally fail, they do not exhibit the complete structural failure

observed in the MLP baseline. The reason the MLP fails deterministically is because it doesn't have any memory at all. RNNs and LSTMs, on the other hand, have decreased F1 scores, not dissolutions of their F1 score. Geometrically this maps to our plots showing partial overlap of attractors, versus complete overlap. The models retain some probabilistic memory, reflecting incomplete but not catastrophic memory loss. Therefore, we should view horizon failures of recurrent models as a fading of attractor separability.

5.6.2 Full Test-Set UMAP Geometry

The following figures (Figure 5.30 to Figure 5.32) show the UMAP projection of hidden-state (or cell-state for LSTM) of all test-set time steps, for RNN, GRU, and LSTM, respectively, and sequences of length $T = 400$.

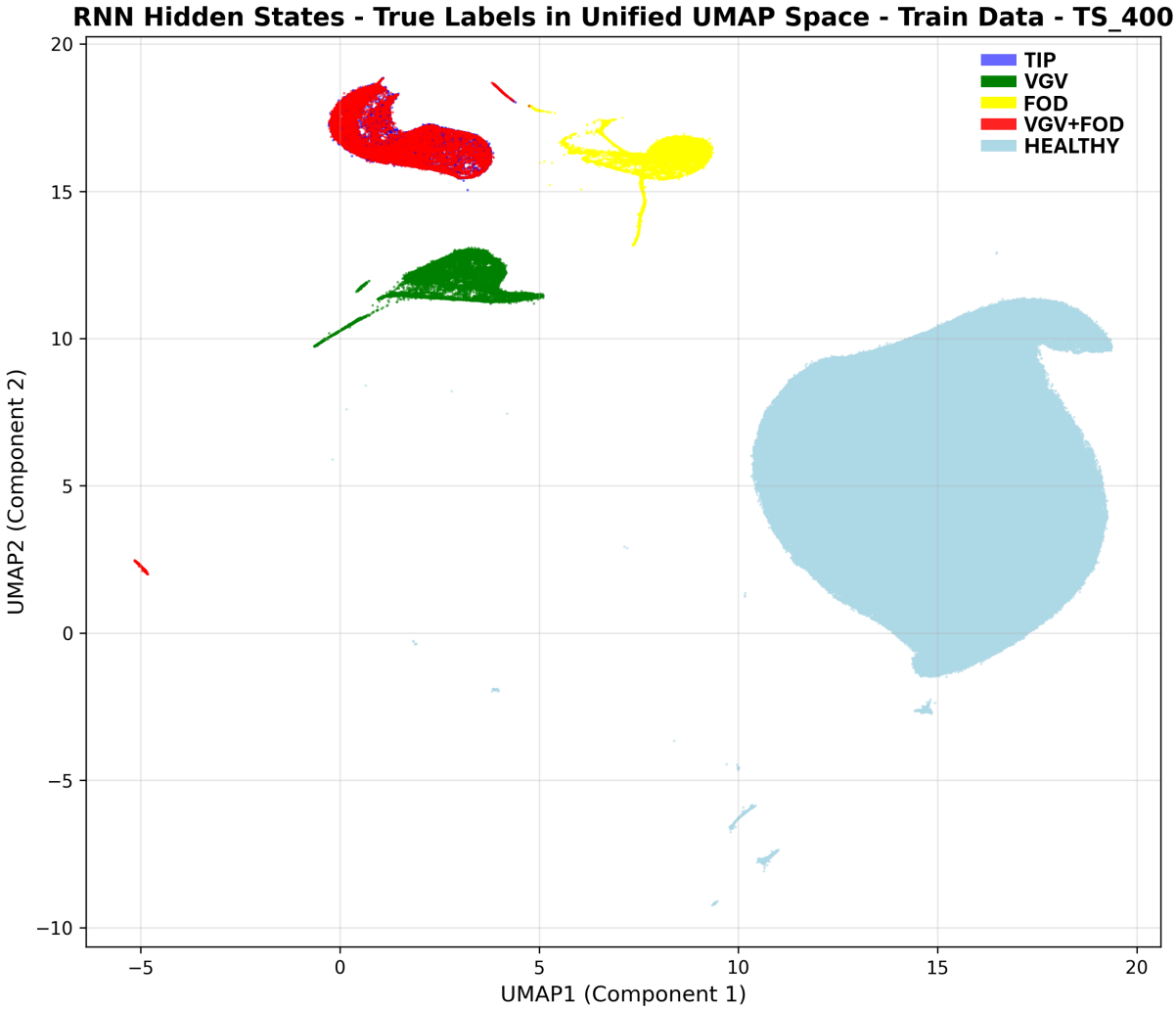

Figure 5.30: *Unified UMAP projection of hidden-state of all test-set time steps, for RNN ($T = 400$).*

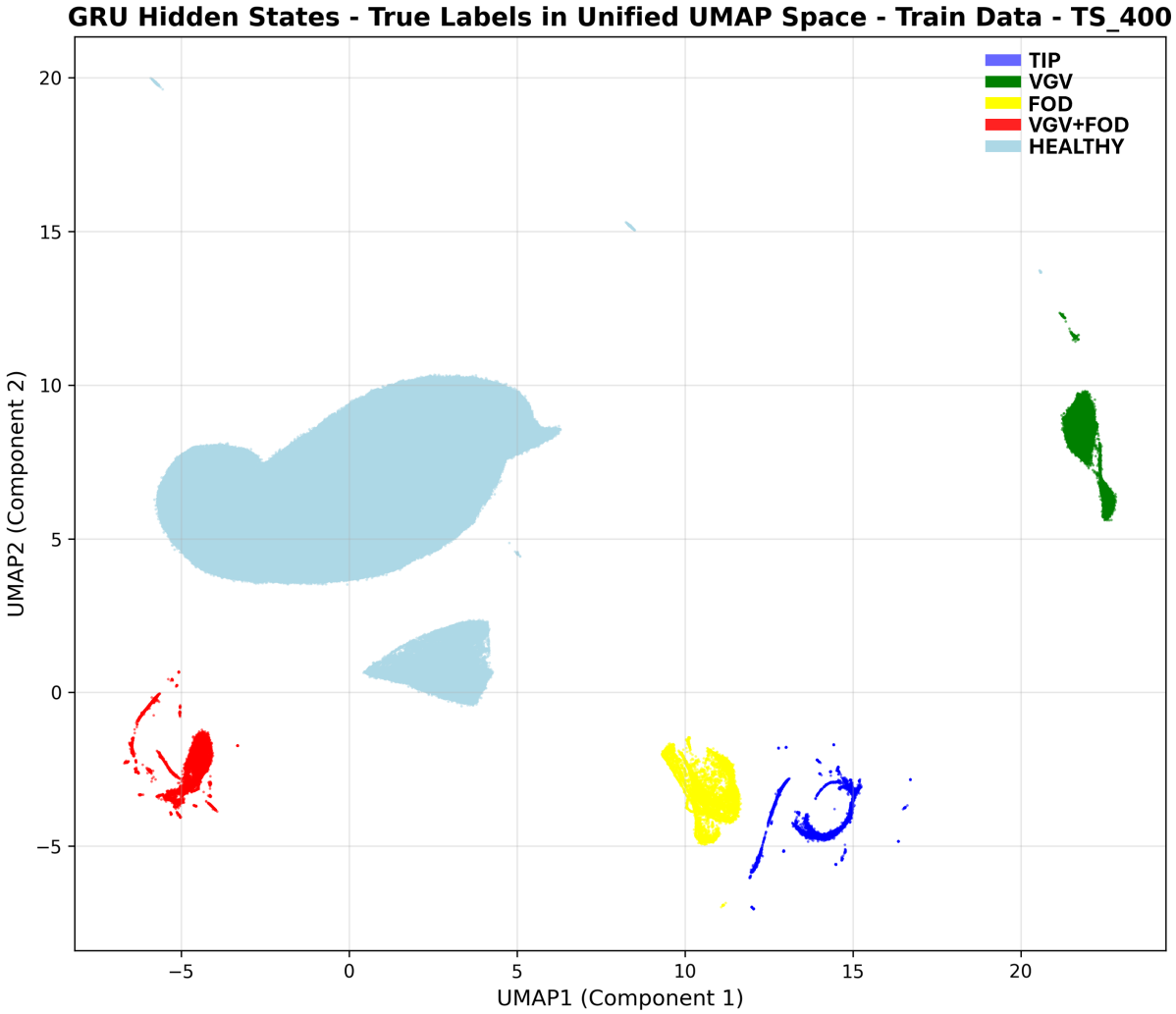

Figure 5.31: *Unified UMAP projection of hidden-state of all test-set time steps, for GRU ($T = 400$).*

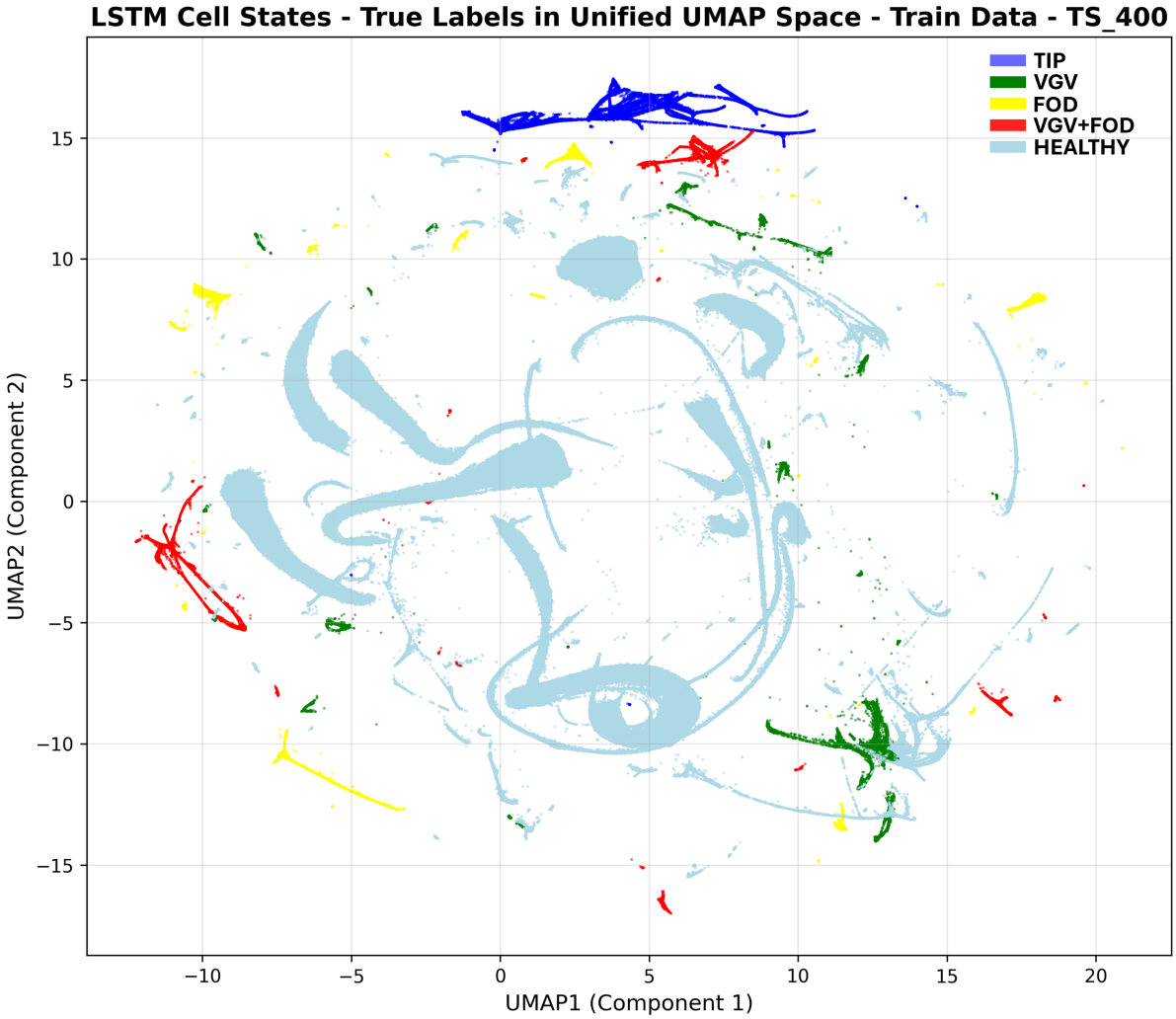

Figure 5.32: *Unified UMAP projection of cell-state of all test-set time steps, for LSTM ($T = 400$).*

In these figures the projection of the hidden-states (cell-states for LSTM) are visualized, with different colors. Each point is colored by label: Healthy (light blue), TIP (blue), VGV (green), FOD (yellow), VGV+FOD (red).

From these figures, we can see that, for GRU and LSTM, separate basins for each fault are visible, and healthy states form distinct clusters, although it is hard to distinguish pre- and post-transient healthy basins. The vanilla RNN, on the other hand, demonstrates a single healthy basin and overlapping regions of TIP and VGV+FOD faults. This behavior is consistent with the corresponding results from PCA analysis, on models with $T = 400$.

Similarly, the following figures (Figure 5.33 to Figure 5.35) show the UMAP projection of hidden-state (or cell-state for LSTM) of all test-set time steps, but for models with sequences of length $T = 500$.

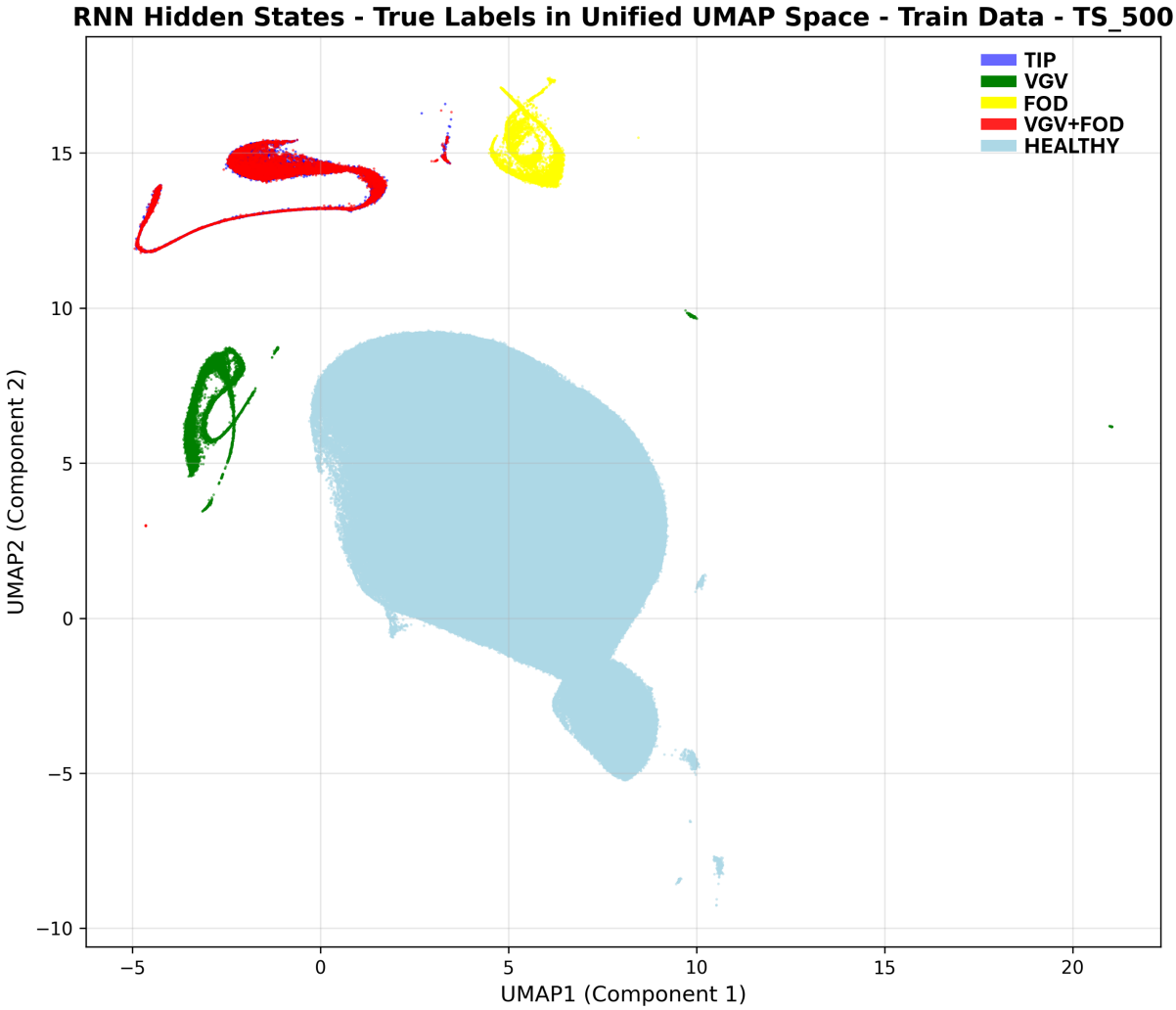

Figure 5.33: *Unified UMAP projection of hidden-state of all test-set time steps, for RNN ($T = 500$).*

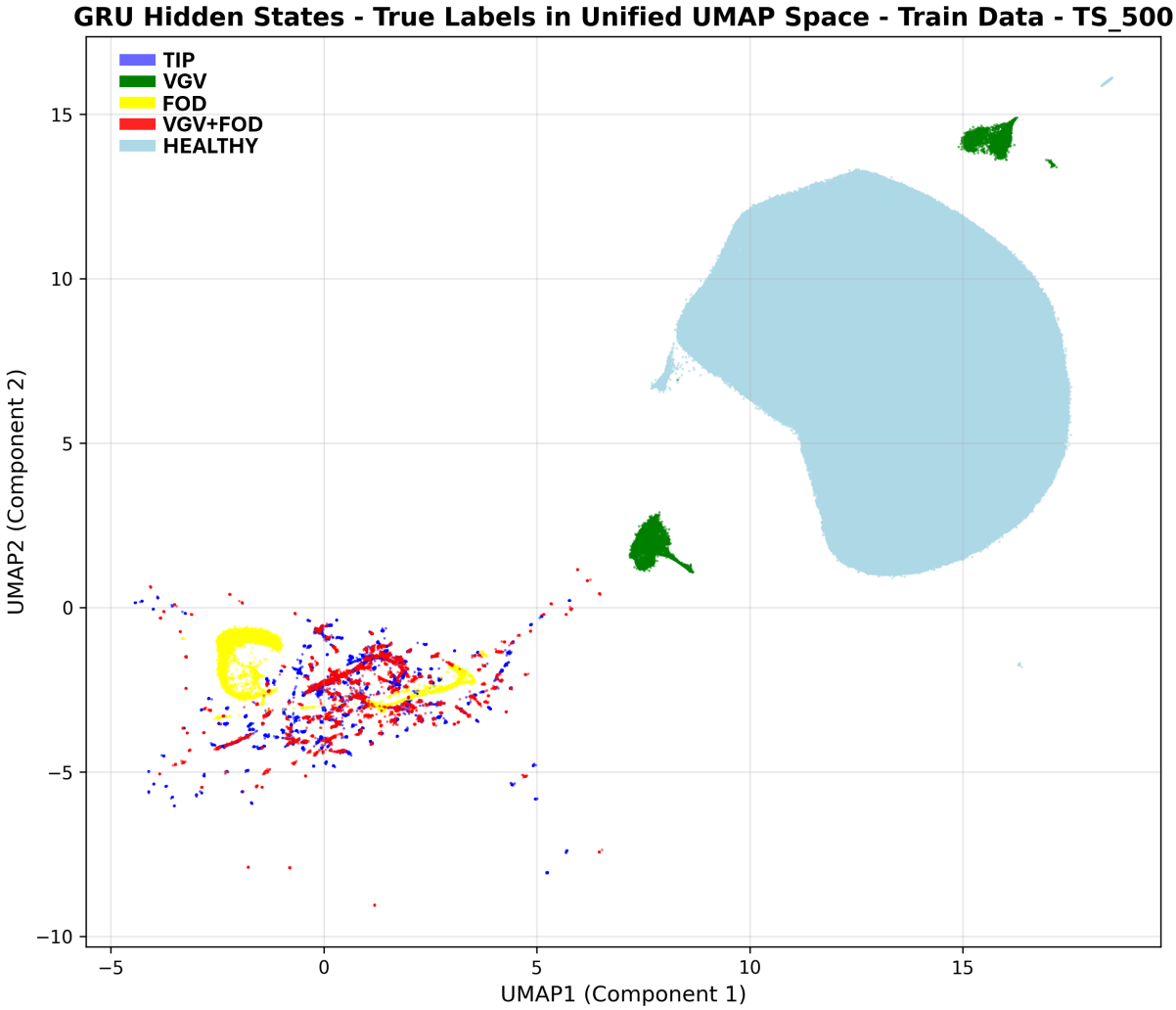

Figure 5.34: *Unified UMAP projection of hidden-state of all test-set time steps, for GRU ($T = 500$).*

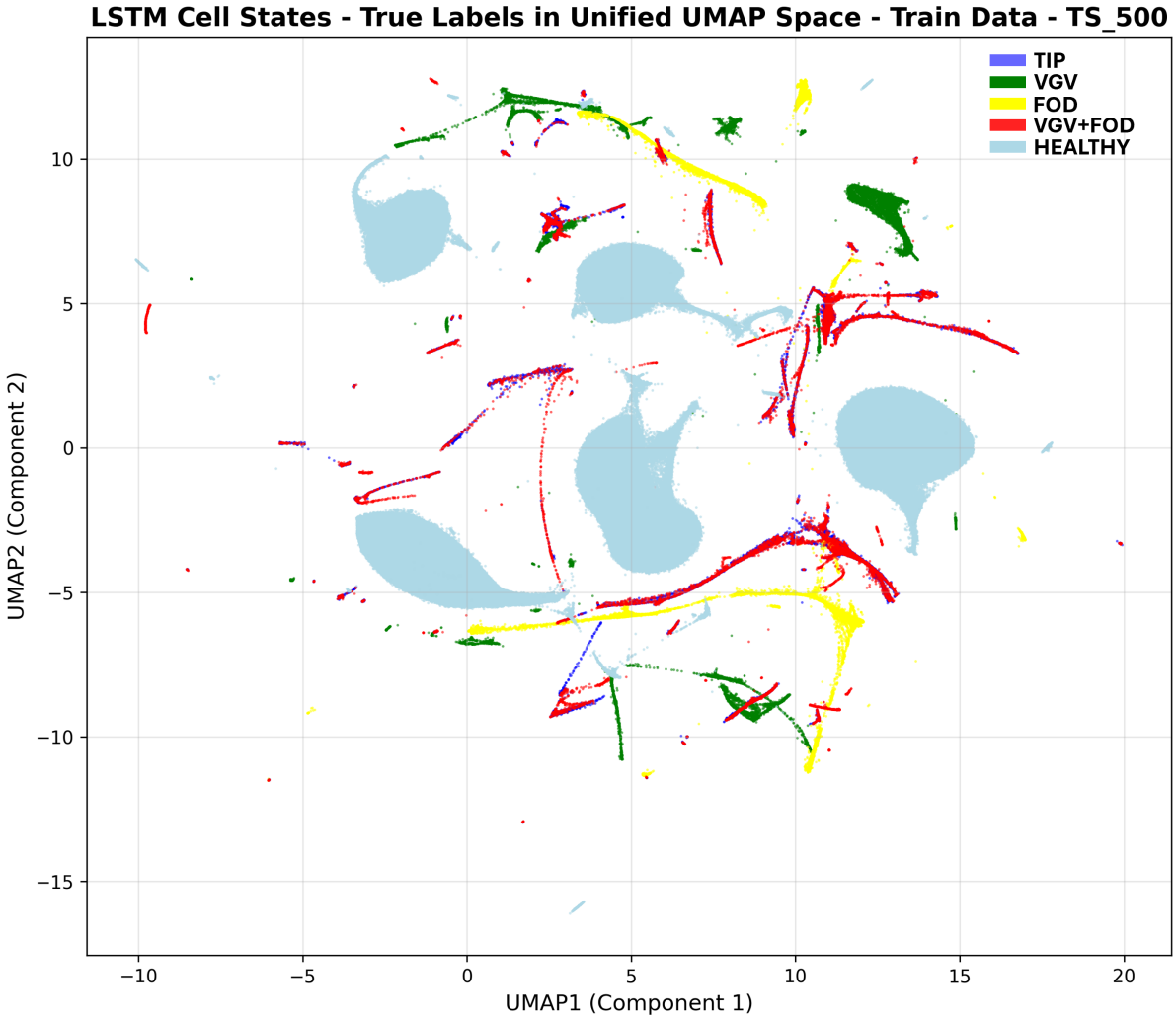

Figure 5.35: *Unified UMAP projection of cell-state of all test-set time steps, for LSTM ($T = 500$).*

From these figures we can see that TIP and VGV+FOD regions overlap significantly. Also, the healthy state forms a single unified basin, and no geometric separation exists between healthy states before and after the transient.

5.7 Quantitative Attractor Separation: ASI Analysis

PCA and UMAP reveal qualitative evidence of attractor merging and bifurcation; however, we need a quantitative measure to systematically evaluate geometric separation across sequence lengths and architectures. To this end, the Attractor Separation Index (ASI), introduced in Section 4.6.5, is computed for all models and horizons.

Two variants are evaluated:

- ASI_{fault}, measuring separation between TIP and VGV+FOD steady-state attractors,
- $\text{ASI}_{healthy}$, measuring separation between healthy states before and after the transient VGV event.

5.7.1 Fault-Level Separation

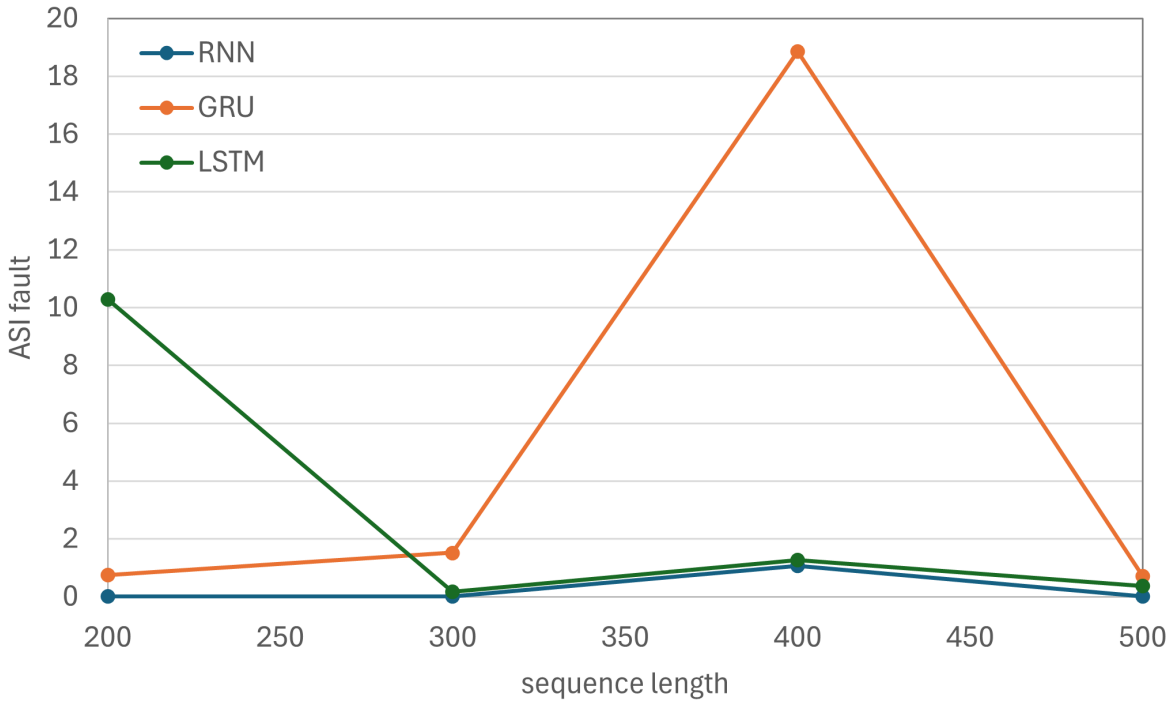

Figure 5.36: *Attractor Separation Index (fault-level) as a function of sequence length for RNN, GRU, and LSTM.*

Figure 5.36 presents the ASI_{fault} as a function of sequence length for RNN, GRU, and LSTM. Its values reveal strong horizon-dependent oscillatory behavior:

- For $T = 200$, LSTM exhibits extremely high separation (ASI ≈ 10.28), while RNN shows near-zero separation.
- At $T = 300$, GRU achieves moderate separation (ASI ≈ 1.52), whereas LSTM collapses (ASI ≈ 0.17).
- At $T = 400$, GRU exhibits a dramatic increase (ASI ≈ 18.85), with LSTM and RNN also showing positive separation.

- At $T = 500$, separation collapses again across architectures.

These oscillations mirror the macro-F1 trends observed previously. When ASI_{fault} approaches zero (e.g., RNN at $T = 200, 300, 500$; LSTM at $T = 300$), classification confusion between TIP and VGV+FOD increases. Conversely, high ASI_{fault} values correspond to strong discriminative performance.

This confirms that classification success is directly linked to geometric attractor separation.

5.7.2 Healthy-State Separation

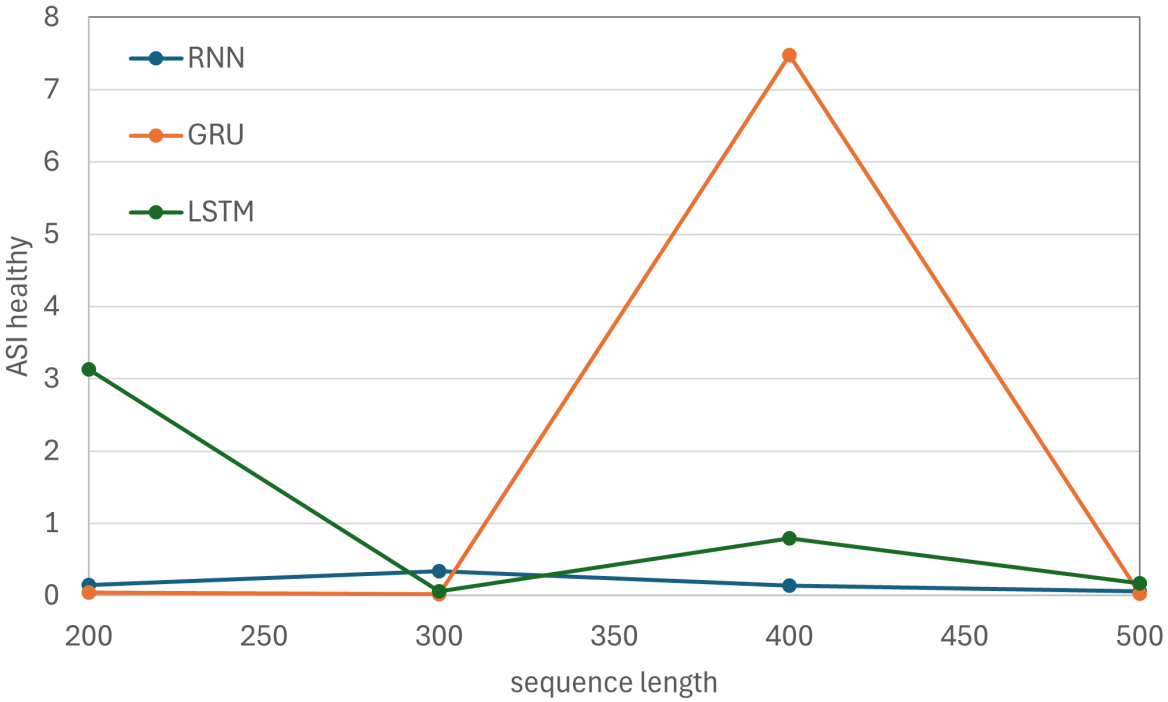

Figure 5.37: *Attractor Separation Index (healthy pre/post transient) versus sequence length.*

Figure 5.37 presents the $\text{ASI}_{healthy}$ as a function of sequence length for RNN, GRU, and LSTM. This metric quantifies whether the transient event induces persistent modification of the healthy manifold. From this figure, a consistent pattern emerges:

- At horizons where $\text{ASI}_{healthy} \approx 0$ (e.g., GRU at $T = 300$, RNN at $T = 500$), the healthy state before and after the transient collapse into a single basin.
- At $T = 400$, GRU exhibits strong healthy-state separation (ASI ≈ 7.48), consistent with the bifurcation observed in PCA.

- LSTM shows moderate healthy separation at $T = 400$ (ASI ≈ 0.80), aligning with partial geometric splitting.
- Extremely low $\text{ASI}_{healthy}$ corresponds to the reversion phenomena described in Section 5.4.

Thus, persistent memory encoding of the transient event manifests geometrically as healthy-state bifurcation.

5.7.3 Relationship Between Geometry and Performance

To map geometric separation directly to discriminative performance, Figure 5.38 correlates fault-level ASI (ASI_{fault}) with classification performance across architectures and sequence lengths. The ASI axis in this figure is bounded to the range 0–3 to maintain resolution among the other datapoints.

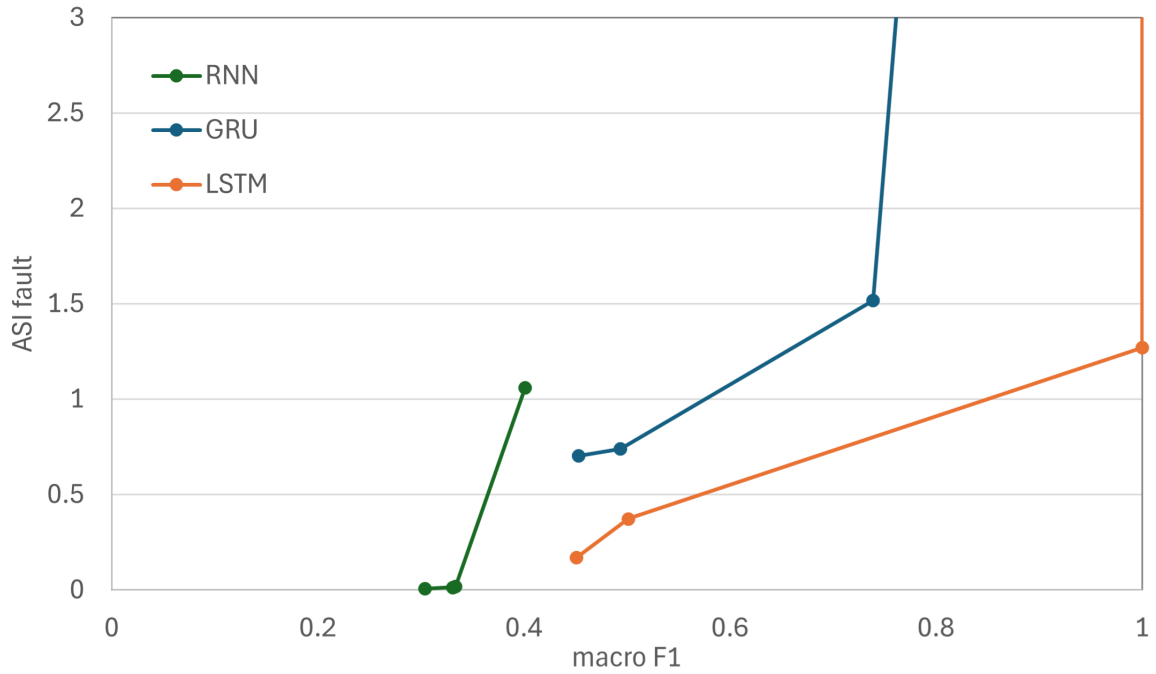

Figure 5.38: *Relationship between fault-level Attractor Separation Index (ASI_{fault}) and two-class macro-F1 (TIP vs VGV+FOD) across architectures and sequence lengths.*

Performance is measured using a two-class macro-F1 score computed exclusively over the TIP and VGV+FOD faults:

$$\text{Macro-F1}_2 = \frac{F1_{TIP} + F1_{VGV+FOD}}{2} \tag{5.2}$$

This choice is deliberate. ASI_{fault} measures the distance between the TIP and VGV+FOD attractors, so we want the performance metric to only measure separation between those two classes. Fault classes with F1 scores near 1 across all horizons would bias overall performance higher and make the geometry–performance connection more unclear.

The resulting scatter reveals a clear monotonic trend: higher ASI_{fault} values correspond to higher macro-F1 scores.

Although the relationship is not strictly linear, each architecture shows that as the attractors move farther apart, fault discrimination improves. Moreover, degradation does not occur suddenly but progressively. As ASI_{fault} decreases, macro-F1 declines gradually rather than collapsing to zero, reflecting partial attractor overlap rather than total geometric collapse. Quantitatively, Figure 5.38 demonstrates the geometric narrative established through PCA and UMAP: classification performance in long-sequence fault diagnosis is tightly coupled to horizon-dependent rearrangement of internal state-space geometry. Changes in training horizon modify attractor separation, and this geometric rearrangement directly affects the model's performance.

5.8 Final Synthesis of Geometric and Performance Findings

The experimental analysis presented in this chapter supports the existence of an organized relationship between classification accuracy and latent space topology in recurrent networks.

Both the PCA and UMAP embeddings above illustrate that correct discrimination maps onto separation of the final attractors for TIP and VGV+FOD. Where models obtain high two-class macro-F1 scores, hidden-state trajectories converge to well-separated regions of latent space. Low performance, on the other hand, corresponds to partial or total overlap of these regions. Since this trend holds for both linear PCA and nonlinear UMAP projections, it becomes evident that this phenomenon is not a projection artifact, but a structural property of the learned dynamics.

We quantify this relationship by measuring the Attractor Separation Index (ASI). ASI_fault increases monotonically with two-class macro-F1 across all architectures and sequence lengths. The failure is not sudden but progressive: As attractor separation decreases, performance degrades smoothly rather than catastrophically. We see something similar to partial geometric collapse, rather than complete structural breakdown.

Importantly, these performance oscillations as a function of sequence length are not due to classification difficulty or representational limitations. As shown by the MLP baseline in Section 5.1, failure to differentiate between TIP and VGV+FOD occurs exclusively when temporal memory is required. Furthermore, as shown by the parametric study in Section 4.5, increasing hidden units or layers does not remove horizon-dependent behavior. Therefore, the behavior is not driven by a lack of capacity but by dynamical reorganization.

Examining gate activation further illuminates this point: when the network fails, it consistently returns to its pre-transient basin of attraction after some time, "forgetting" the transient event. When the network succeeds, the transient pushes trajectory initialization far enough that

there are now two distinct terminal attractors. Results from $\mathrm{ASI}_{healthy}$ also point to this conclusion: how much the transient event permanently alters the size of the healthy basin.

Together, these results validate the main claim of this chapter: changing training horizon causes non-monotonic reconfiguration of the internal state-space of recurrent models. Performance on the classification task is therefore a consequence, not of lack of layer capacity, but of horizon-dependent changes to the internal geometry of these networks.

Memory limitations, as observed in longer time series, therefore arise from memory-crackpoints in this internally learned geometry. Recall that in Chapter 3 we developed a dynamical systems picture of how memory is encoded in RNNs. We showed that transition points should exist where attractor separation is lost. This behavior is observed here: the experiment clearly shows a relationship between the predictive performance of the network and the internal geometry it learns to represent.

5.9 Reproducibility and Code Availability

All source code implementing the experimental framework described in this chapter is publicly available at:

10.5281/zenodo.18839879

The repository includes:

- The full dataset generation pipeline based on the turbofan Engine Performance Model (EPM),
- Implementations of the RNN, GRU, and LSTM architectures with internal state and gate logging,
- Training and evaluation scripts for all sequence lengths and architectural configurations,
- The complete set of experimental results,
- All generated figures presented in this chapter,
- Additional results not included in the main text.

Throughout this chapter, representative and characteristic results were selected for clarity of presentation. The online repository contains the complete collection of experiments across all sequence lengths, seeds, and architectural variations discussed.

The provided code and results allow full reproducibility of the findings reported in this thesis and enable further exploration of horizon-dependent dynamical regimes and attractor reorganization phenomena.

6 Discussion and Conclusions

6.1 Overview and Synthesis of Findings

In this thesis we tried to investigate a fundamental question: What does it mean for a sequential model to have memory?

To answer this question, we went beyond equating memory with predictive performance, and we considered a dynamical-systems approach: memory becomes geometric. We conceptualize memory as trajectory separation in recurrent models' internal state spaces. Then, memory retention becomes equivalent to basin stability under autonomous dynamics; forgetting is attractor merging.

These notions are supported by the experiments of Chapter 5.

1. The MLP baseline showed us that sequence memory is difficult due to classification. Remembering vs. forgetting TIP vs. VGV+FOD faults is exclusively due to having to remember a transient precursor. Omission of temporal information leads to deterministic failure.

2. Next, we saw that recurrent architectures (RNN, GRU, LSTM) do not monotonically improve with larger training horizons. Memory does not uniformly improve when trained on longer sequences. Instead, there are regimes of T where perfect separation is possible and neighboring regimes where it is not. These are oscillatory.

3. Internal gate analysis showed that winning vs. losing the memory game corresponds to sustained displacement from healthy trajectory AFTER the transient has passed. When the models lose memory, the internal trajectories return back to the healthy manifold.

4. PCA/UMAP showed us that performance oscillations correspond to qualitative changes in state-space geometry. When the models remember, state-space is qualitatively bifurcated into basins of attraction for TIP and VGV+FOD. When they fail, these states begin to merge.

5. Lastly, we quantified, via Attractor Separation Index (ASI), this geometry dependence with memory retention. Monotonicity of ASI_{fault} with respect to macro-F1 indicates that degradation is not random noise. When models fail to remember, geometric contraction of basin separation occurs.

6.2 Memory as a Dynamical Phenomenon

A central conceptual contribution of this work is that memory is a property of internal dynamics, not capacity or optimization.

Traditional explanations of long-term dependency failure center on vanishing/exploding gradients. Although this phenomenon explains, at some point, optimization instability, it doesn't

directly account for the oscillatory, regime-dependent phenomena we've observed. All of our models train successfully and have stable gradients, yet experience intermittent memory loss across the horizon.

The theoretical framework of Chapter 3 fills this gap. When trained on quasi-stationary inputs, the dynamics of recurrent models recurse autonomously. In this regime:

- Finite events become perturbations in state space
- Memory retention depends on whether those perturbations push the system into separate basins of attraction
- The geometry of those basins is defined by spectral structure and phase dependent gradient reinforcement.

Phase dependent gradient reinforcement means that increasing the training horizon selectively reinforces different dynamical modes. This can, and does, reorganize attractor structure for small changes in sequence length.

This is a principled reason why we might observe non-monotonic memory behavior in the experiments.

Memory is thus not a function of how many timesteps your gradients can reach. Instead, it is a question of whether training reinforces modes that maintain separation in state space.

6.3 Memory-Crackpoints as Dynamical Phase Transitions

The concept of memory-crackpoints, introduced theoretically in Chapter 3 and demonstrated experimentally in Chapter 5, is another key contribution of this thesis.

Recall that a memory-crackpoint is qualitatively different from mere performance degradation. A memory-crackpoint is a transition in the topology of internal state-space:

- In one phase, trajectories arising from different trajectories settle onto different attractors.
- In the other phase, different trajectories settle onto the same attractor.

Crucially, this transition need not be accompanied by catastrophic loss of time-wise accuracy. A model can classify the majority of timesteps correctly, while internally "forgetting" the geometry necessary to discriminate correctly at final timestep.

Memory-crackpoints are therefore dynamical phase transitions of learned recurrent systems.

6.4 Architectural Implications

The comparative analysis across RNN, GRU, and LSTM reveals something else. The simplest RNN has a narrow basin that is inconsistent with its lack of gates and explicit memories.

GRU and LSTM have more flexible dynamical behavior. But gating isn't enough for consistent memory. Both have horizon dependent bifurcation and continuation regimes. We learn gating provides capacity for richer dynamics, but how the gating is trained determines if memories will persist. Architectural capacity alone does not prevent memory-crackpoints.

6.5 Methodological Contributions

In addition to providing theoretical understanding, this thesis provides experimental advancements:

1. A synthetic dataset derived from a high-fidelity turbofan engine model was created with only long-term memory requirements.
2. The recurrent networks were modified to log hidden states, cell states, and gate values at each timestep.
3. A combined framework of PCA and UMAP was implemented to correlate state-space geometry with performance.
4. A metric to quantify attractor separation (ASI) was introduced to quantify qualitative geometric findings.
5. This integrated methodology allowed for a systematic investigation of memory as a dynamical phenomenon.

6.6 Limitations

Several limitations should be stated.

First, we only experiment with single-layer recurrent nets. In deeper models, although briefly experimented with, the geometric analysis becomes more complex.

Second, we use a synthetic dataset, although realistic. Real-world industrial datasets may exhibit more randomness, non-stationarity, and distributional drift.

Third, we use dimensionality reduction (PCA/UMAP) for geometric analysis. While agreement between linear and nonlinear embedding increases our confidence, information is lost in the projection.

Fourth, our theoretical result requires local linearization and spectral reasoning. Global nonlinear dynamics may exhibit additional complexity not visible in the Jacobian alone.

Despite these limitations, the consistency between theory and experiment strongly supports the validity of the proposed framework.

6.7 Directions for Future Research

Several promising research directions emerge.

1. Monitoring Jacobian spectra during training. Keeping track of the spectra of the Jacobian matrix along horizons would allow us to better validate our phase dependent interpretation.

2. Setting the spectrum of recurrent matrix to initialize desired spectral regime. One could initialize the recurrent matrix with specific eigenvalue spectra in order to intentionally set certain memory regimes.

3. Application to Transformers. Do similar geometric phase transitions occur in attention based models? This could help in extending our dynamical view to transformers.

4. Deployment on industrial fault-diagnosis applications. Run attractor-separation metrics on real fault diagnosis tasks where failure of such metrics guarantees safety.

5. Mathematical characterization of attractor bifurcation. Provide mathematical definition for bifurcation of attractors for learned high dimensional dynamical systems.

6.8 Final Conclusions

This thesis demonstrates that long-term memory in recurrent neural networks is rather a dynamical and geometric phenomenon.

Memory retention depends not only on gradient flow or architectural capacity, but also how the learned internal state space is organized. Training horizon modifies spectral reinforcement, which reorganizes attractor geometry, leading to regime-dependent trajectory separation.

Performance oscillations at various sequence lengths are therefore not anomalies. They show the underlying dynamical phase transitions, memory-crackpoints, predicted by the theoretical framework developed in Chapter 3.

By combining spectral theory, nonlinear dynamics, internal-state geometry, and experimental demonstration, this work provides a mechanistic explanation of the limits of memory retention in recurrent neural networks.

Therefore, the analysis of sequential models is not tied solely to output metrics, but also to internal dynamical structure — and point towards geometry-conscious methods to design and diagnose memory.

References

[1] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1045-9227, 1941-0093. DOI: 10.1109/72.279181.

[2] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". en. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.

[3] George E. P. Box et al. *Time series analysis: forecasting and control*. eng. Fifth edition. Wiley series in probability and statistics. Hoboken, New Jersey: Wiley, 2016. ISBN: 978-1-118-67502-1 978-1-118-67492-5.

[4] Jeffrey L. Elman. "Finding Structure in Time". en. In: *Cognitive Science* 14.2 (Mar. 1990), pp. 179–211. ISSN: 0364-0213, 1551-6709. DOI: 10.1207/s15516709cog1402_1.

[5] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. *A Simple Way to Initialize Recurrent Networks of Rectified Linear Units*. arXiv:1504.00941 [cs]. Apr. 2015. DOI: 10.48550/arXiv.1504.00941.

[6] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. arXiv:1211.5063 [cs]. Feb. 2013. DOI: 10.48550/arXiv.1211.5063.

[7] Martin Arjovsky, Amar Shah, and Yoshua Bengio. *Unitary Evolution Recurrent Neural Networks*. Version Number: 4. 2015. DOI: 10.48550/ARXIV.1511.06464.

[8] Klaus Greff et al. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (Oct. 2017), pp. 2222–2232. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2016.2582924.

[9] Corentin Tallec and Yann Ollivier. *Can recurrent neural networks warp time?* Version Number: 1. 2018. DOI: 10.48550/ARXIV.1804.11188.

[10] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. arXiv:1406.1078 [cs]. Sept. 2014. DOI: 10.48550/arXiv.1406.1078.

[11] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv:1412.3555 [cs]. Dec. 2014. DOI: 10.48550/arXiv.1412.3555.

[12] Herbert Jaeger. "Long Short-Term Memory in Echo State Networks: Details of a Simulation Study". In: 2012.

[13] Olivia L. White, Daniel D. Lee, and Haim Sompolinsky. "Short-Term Memory in Orthogonal Neural Networks". en. In: *Physical Review Letters* 92.14 (Apr. 2004), p. 148102. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.92.148102.

[14] Surya Ganguli, Dongsung Huh, and Haim Sompolinsky. “Memory traces in dynamical systems”. en. In: *Proceedings of the National Academy of Sciences* 105.48 (Dec. 2008), pp. 18970–18975. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0804451105.

[15] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. *Visualizing and Understanding Recurrent Networks*. Version Number: 2. 2015. DOI: 10.48550/ARXIV.1506.02078.

[16] Yossi Adi et al. *Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks*. Version Number: 3. 2016. DOI: 10.48550/ARXIV.1608.04207.

[17] Yonatan Belinkov and James Glass. *Analysis Methods in Neural Language Processing: A Survey*. Version Number: 2. 2018. DOI: 10.48550/ARXIV.1812.08951.

[18] David Sussillo and Omri Barak. “Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks”. en. In: *Neural Computation* 25.3 (Mar. 2013), pp. 626–649. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/NECO_a_00409.

[19] Pang Wei Koh and Percy Liang. *Understanding Black-box Predictions via Influence Functions*. Version Number: 3. 2017. DOI: 10.48550/ARXIV.1703.04730.

[20] Eugene Vorontsov et al. *On orthogonality and learning recurrent networks with long term dependencies*. Version Number: 4. 2017. DOI: 10.48550/ARXIV.1702.00071.

[21] Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. *The Emergence of Spectral Universality in Deep Networks*. Version Number: 1. 2018. DOI: 10.48550/ARXIV.1802.09979.

[22] Kyle Helfrich, Devin Willmott, and Qiang Ye. “Orthogonal Recurrent Neural Networks with Scaled Cayley Transform”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 1969–1978.

[23] Randall D. Beer. “On the Dynamics of Small Continuous-Time Recurrent Neural Networks”. In: *Adaptive Behavior* 3 (1995), pp. 469–509.

[24] Steven Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. eng. Second edition, first issued in hardback. A Chapman & Hall book. Boca Raton London New York: CRC Press, 2019. ISBN: 978-0-8133-4910-7 978-0-367-09206-1.

[25] Hassan K. Khalil. “Nonlinear systems. Hauptbd.” eng. In: 3. ed. Num Pages: 750. Upper Saddle River, NJ: Prentice Hall, 2002. ISBN: 978-0-13-067389-3.

[26] Lloyd N. Trefethen. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. eng. Princeton: Princeton University Press, 2005. ISBN: 978-0-691-11946-5 978-0-691-21310-1.

[27] Christoforos Romesis and Stasinos Konstantopoulos. *A Challenging Dataset of Jet Engine Fault Scenarios*. en. July 2025. DOI: 10.5281/ZENODO.15856441.

[28] Christoforos Romesis, Nikolaos Aretakis, and Konstantinos Mathioudakis. “Model-Assisted Probabilistic Neural Networks for Effective Turbofan Fault Diagnosis”. en. In: *Aerospace* 11.11 (Nov. 2024), p. 913. ISSN: 2226-4310. DOI: 10.3390/aerospace11110913.

[29] K Mathioudakis et al. “Performance analysis of industrial gas turbines for engine condition monitoring”. en. In: *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* 215.2 (Jan. 2001), pp. 173–184. ISSN: 0957-6509, 2041-2967. DOI: 10.1243/0957650011538442.

[30] Konstantinos Mathioudakis et al. “Signatures of Compressor and Turbine Faults in Gas Turbine Performance Diagnostics: A Review”. en. In: *Energies* 17.14 (July 2024), p. 3409. ISSN: 1996-1073. DOI: 10.3390/en17143409.

[31] A. Alexiou and K. Mathioudakis. “Development of Gas Turbine Performance Models Using a Generic Simulation Tool”. In: *Volume 1: Turbo Expo 2005*. Reno, Nevada, USA: ASMEDC, Jan. 2005. DOI: 10.1115/gt2005-68678.

[32] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980 [cs]. Jan. 2017. DOI: 10.48550/arXiv.1412.6980.

[33] Ian T. Jolliffe. *Principal component analysis*. eng. 2. ed., [Nachdr.] Springer series in statistics. New York Berlin Heidelberg: Springer, 2004. ISBN: 978-0-387-95442-4.

[34] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv:1802.03426 [stat]. Sept. 2020. DOI: 10.48550/arXiv.1802.03426.

[35] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. eng. Second edition. Springer Series in Statistics. New York, NY: Springer, 2017. ISBN: 978-0-387-84857-0 978-0-387-84858-7.