TU WIEN Informatics

Modeling Dynamics on a Canonical Neural Manifold in C. elegans

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Data Science

eingereicht von

Liana Akobian, BSc
Matrikelnummer 01635540

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser
Zweitbetreuung: Univ.-Prof. Dr. Manuel Zimmer
Mitwirkung: Charles Fieseler, MSc. PhD

Wien, 2. Dezember 2024

Liana Akobian | Peter Filzmoser

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Modeling Dynamics on a Canonical Neural Manifold in C. elegans

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Data Science

by

Liana Akobian, BSc
Registration Number 01635540

to the Faculty of Informatics

at the TU Wien

Advisor:	Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser
Second advisor:	Univ.-Prof. Dr. Manuel Zimmer
Assistance:	Charles Fieseler, MSc. PhD

Vienna, December 2, 2024

Liana Akobian	Peter Filzmoser

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Liana Akobian, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 2. Dezember 2024

Liana Akobian

Acknowledgements

I am greatly indebted to my mentor, Charles Fieseler. The lessons I have learned through you on this journey are invaluable and have shaped my way of doing science. I hope I can pass on the knowledge you have instilled in me to other people. Thank you, Charlie.

I would like to thank my supervisor Professor Manuel Zimmer from the University of Vienna for this opportunity, the guiding discussions, and for building an environment that fosters curiosity, individuality, and scientific integrity. I am also grateful to my supervisor Professor Peter Filzmoser from the Technical University of Vienna for his patience and support during various projects in my Master's.

I am deeply grateful to past and current members of the Zimmer lab for making this year so bright, from weekly lab lunches to hourly computer room giggles. My heart is full when I look back at the moments I shared with all of you. A special thanks to Itamar for answering all my worm- and shakshuka-related questions and Eva, Chanuka, and Samuel for existing. I want to thank my friends from my time at KU Leuven who showed me that home is not bound to a single place. My fellow students from the Technical University and the University of Vienna for their support and guidance. And, really, everyone who contributed to my academic growth.

On a personal note, I want to thank my parents, Nona and Aschot, for their unwavering support and love: Շնորհակալություն. My close friends Michi, Flo and Kang Da and my sisters, Ani and Biajna, for being the best fixed point attractors one could ask for. My friends and boulder buddies, and all the people along this trajectory that have made me me.

The computational results of this work have been achieved using the Life Science Compute Cluster (LiSC) of the University of Vienna.
Whole-brain Ca^{2+}-imaging data have been provided by Rebecca Kresnik and Kerem Uzel, alumni of the Zimmer Lab.

Kurzfassung

Wie Neuronen miteinander interagieren um Verhalten zu erzeugen, stellt eine zentrale Frage in der Neurowissenschaft dar. Die Dynamiken dieser interagierenden Neuronen definieren die neuronalen Berechnungen, die der Verarbeitung sensorischer Informationen, der Entscheidungsfindung und der Erzeugung von Motorik zugrunde liegen. Jüngste Fortschritte in der Modellierung dynamischer Systeme haben beobachtete neuronale Aktivität als zeitliche Entwicklung von Zuständen innerhalb eines neuronalen Raumes, der durch dynamische Gesetze geregelt wird, formalisiert. Bedeutende Fortschritte wurden erzielt, indem angenommen wurde, dass diese Gesetze autonomer Natur sind, was bedeutet, dass sich neuronale Zustände deterministisch entwickeln. Solche Modelle bieten jedoch möglicherweise nicht genügend biologische Interpretierbarkeit, da sie unvorhersehbare externe Einflüsse nicht erfassen. Wir schlagen ein kontrolliertes, zerlegtes lineares dynamisches System (cdLDS) vor, eine Erweiterung des autonomen linearen dynamischen Systems *dLDS*, indem Eingaben integriert werden, die das System steuern. Wir wenden cdLDS auf eine neuronale Mannigfaltigkeit, eine niederdimensionale dynamische Struktur, von 23 *C. elegans*-Individuen an und zeigen, dass es erfolgreich intrinsische neuronale Dynamiken von Steuerungssignalen entflechtet und Einblicke in Störungen der neuronalen Dynamik bietet. Dieses Framework bildet eine Grundlage zur Identifizierung der neuronalen Korrelate von Steuerungssignalen und zum Verständnis der Auswirkungen von Steuerungsmechanismen auf die neuronale Dynamiken.

Abstract

Understanding how neurons interact with each other to produce behavior is a key challenge in neuroscience. The dynamics of these interacting neurons define the computations that underlie the processing of sensory information, decision making, and the generation of motor output. Recent advances in dynamical system modeling have formalized observed neural activity as the temporal evolution of states within a neural state space governed by dynamical laws. Significant progress has been achieved by assuming these laws to be of autonomous nature, meaning that neural states evolve deterministically. However, such models may not provide sufficient biological interpretability as they fail to capture unpredictable external forces. Here, we propose a controlled decomposed linear dynamical system (cdLDS), an extension of the autonomous dynamical system model *dLDS*, by incorporating inputs that control the system. We apply cdLDS to a neural manifold, a low-dimensional dynamical structure, from 23 *C. elegans* individuals and show that it successfully disentangles intrinsic neural dynamics from control signals, offering insight into perturbations of neural dynamics. This framework provides a foundation for identifying the neural correlates of control signals and for understanding the impact of control mechanisms on neural dynamics.

Contents

CHAPTER 1

Introduction

1.1 Inside the Brain: Understanding Neural Dynamics

Our brain comprises a hundred billion neurons that constantly communicate with one another, can modulate and connect, and can form complex circuits that essentially underlie every behavior. A primary objective of neuroscience is to understand the dynamics of the entire brain, specifically the way neurons in a brain process, receive, and pass signals over time. It is understood that neurons always behave collectively and that the actual unit of processing is not a single neuron, but rather a collection of neurons, known as a neural population [SC19][EH21]. A population perspective gives rise to the idea that any observed neural state, which can be seen as a snapshot of all neural activity, does not necessarily have to be described at the level of individual neurons. Rather, we can infer a neural state from the activity of a neural population.
This belief aligns with the promising hypothesis that neural activity lies on trajectories in a low-dimensional space. In neuroscience, such low-dimensional structures are loosely termed neural manifolds [Mit+23]. In other words, it is sufficient to describe any neural state with dimensions far fewer than the total number of neurons. In that subspace, known as the state space, each point is a state that represents the pattern of the population activity. However, not all potential states in that space are explored by the brain, and those that are form the neural manifold [EH21].
Similarly to how snapshots of neural activities are connected through time, states that lie on the manifold are also temporally connected. And the rules that move the brain from one neural state to another over time also apply to the neural manifold. Identifying these *dynamical* rules is ubiquitous for understanding any neural dynamics, as they are one and the same, and characterizing the neural manifold facilitates this discovery.

1.2 In search of a canonical neural manifold

The presence of a neural manifold offers many advantages, a major one being their capacity of simplifying complex patterns into a more interpretable representation. Furthermore, the shape of a manifold structure can inform us of what and how many high-level features are encoded in the representation. For example, in their 2019 study, Chaudhuri et al. found that neural measurements of the entire head direction circuit in mice lie on a ring manifold, reducing the dimensionality of the data from 8-30 neurons per dataset to a single dimension per dataset, which could be interpreted as the angle of the mouse head [Cha+19].
Previous work has discerned neural manifolds in various organisms, although primarily within single individuals [Cha+19][Gal+20]. To find a manifold in multiple individuals, one would need to align their neural activity recordings, which are likely to vary over time and space. However, a neural manifold that is truly consistent across multiple individuals in an organism could reveal features that are robust and universal. With such a *canonical neural manifold*, one could find all possible neural states that the brain of an organism explores.
Due to the variability between neural activity recordings, be it due to instabilities of the recording device, or different timing or different subsets of neurons, it is challenging to compare the neural activity patterns of different recordings directly. Many researchers circumvent this problem of variability across recordings by building invariance across recordings by computing, for example, correlations of neuron firing and then extracting certain metrics like connectivity or entropy. But most statistical approaches such as these are invariant to the specific subset of neurons, omitting valuable information about neurons [DKD23]. A practical solution is recording the same subset of neurons between individuals. However, capturing whole-brain neural activities is currently infeasible in complex organisms such as mammals, due to technological limitations in achieving sufficient spatial and temporal resolution. In contrast, whole-brain imaging has been successfully demonstrated in a model organism widely studied in neurobiology: *Caenorhabditis elegans* (*C. elegans*).
C. elegans is a nematode with a fixed number of 302 neurons that have been fully mapped, along with their connections, and recorded simultaneously in many studies [Kat+15][Ngu+16]. This makes *C. elegans* an ideal organism for us to study neural dynamics at a global level.
In particular, we want to discover a canonical neural manifold in *C. elegans*, by aligning whole-brain recordings from multiple different individuals while preserving most of the information. This canonical neural manifold then constrains the state space of all neural states.

1.3 Modeling the neural manifold as a proxy for the brain

To study the temporal evolution of states on the manifold, as a proxy for the high-dimensional neural states explored by the brain, we need to identify the governing

dynamical rules [DS21]. With these dynamical rules, any state on the manifold can be computed. In other words, for any population activity, there exists an algorithm or computation that dictates what the next population activity looks like. One might find that a system that is defined by dynamical rules and states is a dynamical system. In fact, significant advances were made in the field of dynamical systems modeling to study neural dynamics [DS21][Bru+16][FZK20][Lin+16].
The theory of dynamical systems gives access to a myriad of analytical tools; in particular, they facilitate the location of fixed point attractors. It is assumed that neural states towards which the brain converges are fixed point attractors, and uncovering those could further characterize the computation that drives a brain. In this thesis, we refer to these attractive neural states as latent states. A latent state can correspond to an observable behavior. For example, if a worm is feeding or foraging for food, its neural circuits will converge to and stay in a latent state associated with these behaviors. The neural dynamics of the worm is constituted by precisely these states and their transitions over time, defined by a dynamical system.
The study of attractive states and the transitions between states gives rise to two schools of thought, where the dynamics of any system is often modeled either as an *autonomous* or as a *non-autonomous* system. The autonomous case highlights the system's inherent ability to transition from one state to another, whereas the non-autonomous case relies on external forces that move the system from state to state.

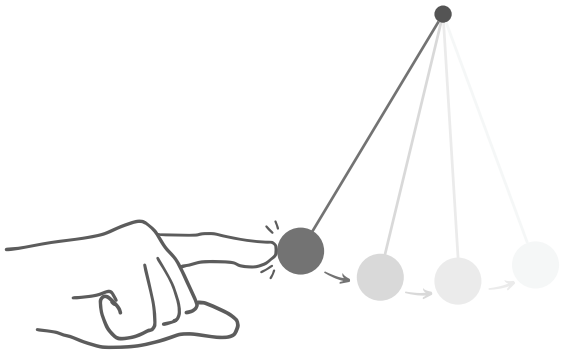


Figure 1.1: **The pendulum is one of the best-studied systems in dynamical systems.** The swing of the pendulum follows rules from an autonomous dynamical system. However, if the swing is perturbed by, say, a push, the swing dynamics become subject to a non-autonomous dynamical system.

A simple example that displays the difference between an autonomous dynamical system and a non-autonomous one is a pendulum that swings from left to right: the state of that pendulum at a given time point depends only on its current angle to the vertical and the angular velocity. These variables change according to the rules that govern the pendulum swing, making it an autonomous dynamical system. Now, if we gave the pendulum an occasional push, the state of the system would additionally depend on an external force, and the governing rules alone do not suffice to describe the current state of the pendulum

(see Figure 1.1).
In general, with enough complexity, both autonomous and non-autonomous dynamical systems models can approximate a wide range of neural states. However, we hypothesize that a system that has deterministic intrinsic dynamics that drive it from one state to another, and on top of that also receive stochastic inputs, may result in a biologically more realistic model.

Finally, we come to the aim of this thesis. We implement a non-autonomous dynamical system that describes dynamics on a canonical neural manifold, as a proof of principle, showing that the observed difference between intrinsic dynamics and actuated dynamics can be modeled in a systematic way. Furthermore, we will link observed and modeled latent states to observed behavior states in *C. elegans*.
Similarly to this introduction, the remainder of this thesis can be divided into two connected parts: the canonical neural manifold and the dynamical systems model. To construct a canonical neural manifold, we must overcome the caveats that an alignment of neural recordings entails. For this, we define a pipeline of preprocessing techniques that make high-dimensional recordings from 23 different *C. elegans* individuals directly comparable. We then validate the universality of the neural manifold by utilizing classic machine learning methods such as classification.
For the second part, we formalize an existing dynamical system model, *decomposed linear dynamical system (dLDS)*, as a neural network and extend it to include external forces, or as we refer to them in this thesis, control signals. For testing and validation, we generate synthetic data based on assumed dynamical principles before applying and interpreting the extended model on our canonical neural manifold.

CHAPTER 2

Preliminaries

2.1 *C. elegans*

C. elegans is a nematode that is found living in soil and has a transparent body of less than 1 mm. The roundworm serves as a model organism in many fields of biology as it can be easily cultivated in the laboratory and it passes through different stages of life rapidly. The nematode has 302 neurons that have been fully mapped, that is, they are identifiable and consistent across worms [Var+11]. This characteristic, along with the transparency and size of the worm, facilitates optical recording of all its cells simultaneously.

2.1.1 Locomotion

C. elegans exhibits various behaviors, such as roaming, dwelling, escaping, egg-laying, mating, mate finding, etc. Part of most of these behaviors is locomotion, i. e. the ability to move. The main motions are forward-directed crawling (forward), backward-directed crawling (reversal) and turns. All motions occur in an undulatory way: The worm lies either on its left or right side and performs front (ventral) and back (dorsal) bends in order to propel either forward or backward. A turn is a localized ventral or dorsal bend, which is often observed in roaming and reorientation behavior. In general, depending on various factors such as speed, strength, or extent of movement, these three categories of motion can be split into observed fine-grained behaviors such as shallow turns, sharp turns (omega), and slowing. More categories or behavior states have been derived by analyzing neural correlates of the main observed motions.
The driver neuron for initiating reversals is AVA.[1][Alt+]. Furthermore, consistent with the approach of Kato et al., we use the following motor command neurons as a readout for forward and turning motion: RIB for forward, SMDV for ventral turns, and SMDD

[1] It has been shown in ablation studies that *C. elegans* can perform reversals even without AVA but at a much lower frequency [Pig+11]

for dorsal turns [Kat+15].
AVA, RIB, SMDV and SMDD belong to a set of neuron classes that consist of a single bilaterally symmetric pair of neurons, i. e. the neuron class AVA has two symmetric neurons, AVAR (right lateral) and AVAL (left lateral), which share the same wiring and anatomical and molecular characteristics [HGW16]. Correlating with the activity of these motor-command neurons are large neuronal populations, which are shown to coordinate their activity to generate locomotion. In order to understand how behavior-related decisions are formed in the brain, a vast field is devoted to analyzing and modeling the dynamics of these populations.

2.1.2 Whole Brain Ca^{2+}-imaging Data

To record the neural activity of *C. elegans*, a calcium imaging technique is used. This is done by growing worms that pan-neuronally express a biosensor protein, called GCaMP, that fluoresces when bound to Calcium ions (Ca^{2+}). When a signal propagates to a neuron, Ca^{2+} and other ions enter the cell and excite the neuron. As GCaMP is expressed, the neuron fluoresces, which can be imaged as a proxy for neural activity. Most signals in *C. elegans* are propagated in the form of graded potentials, rather than the all-or-none action potentials that dominate the mammalian brain. In this thesis, graded potentials are denoted as F, where F is the fluorescence intensity and can reach a value of 70 000. The F value of each neuron is normalized by the average fluorescence intensity of a recording, F0, and corrected for bleaching, which is a decay of fluorescence over time due to light exposure.
With this setting, 23 unstimulated individual worms were imaged for 18 minutes, 17 individuals by Kresnik and 6 individuals by Uzel, which constitute the data that will be used in this thesis [Kre21][UKZ22]. More details about the imaging conditions and tracking can be found in their work. To identify cells more easily, the worms were immobilized and using a tracking tool developed by Kato et al., Kresnik and Uzel have tracked more than 200 neurons per recording and identified approximately 41 to 72 neurons [Kre21].

2.1.3 Behaviour State Annotations

When we discover latent states from both the canonical neural manifold structures and the model, we want to compare and link them to behaviors of *C. elegans*. We can determine the behavior state at each time point or for each observation by comparing signals from the readout neurons AVA, RIB, SMDD and SMDV (see Section 2.1.1). In each recording, Kresnik and Uzel annotated AVA activity at each time point as one of the following four categories: 0-rise, 1-high, 2-fall, and 3-low. 0 indicates a rise in AVA activity and the start of a reversal, thus we define it as *reversal* and 1 or high AVA activity as *sustained reversal*. 3 or low AVA activity gives us the forward state. We define the remaining label, 2 or the fall of AVA activity, as follows: Similarly to Kato et al., we define a fall as either a *ventral* or a *dorsal* turn [Kat+15]. This is done by comparing their readouts, SMDV for ventral turn and SMDD for dorsal turn, and the neuron that

has higher activity during a fall period determines the turn. Note that here we take the average neural activity of the entire pair, i. e. the average of both the left neuron (e.g. SMDVL) and the right neuron (e.g. SMDVR) of the same class.

2.1.4 Pirouettes

We define a pirouette as the following motion sequence: The nematode reverses and turns sharply either dorsally or ventrally to resume forward crawling. Pirouettes are typically studied in *C. elegans* chemotaxis, that is, when the worm moves in response to stimuli, as the motion sequence serves as a strategy to change the direction of travel [Soh+18]. However, in recent years, the activity pattern that underlies a pirouette has also been observed in immobilized and unstimulated wild-type worms [MFK21].

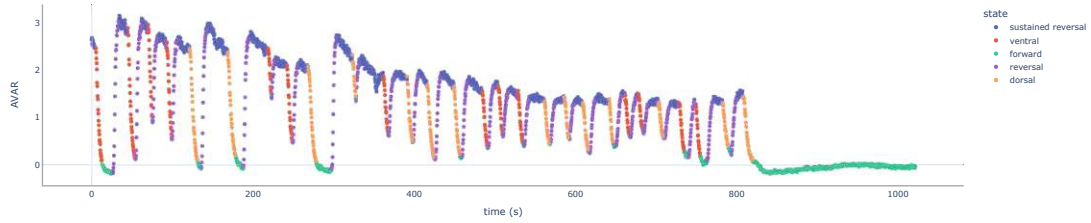


Figure 2.1: **Activity of reversal driver AVA and correlated neurons show 'pirouettes'.** What is from a behaviour perspective a period of frequent turns might be the system failing to stabilize.

From the perspective of dynamical systems, this behavior could be translated into the following: the system is in a reversal state and receives an external kick to transition to another state, but the reversal state is not destabilized, and hence it does not transition. This instance shows the presence of the two aforementioned models where a system is able to autonomously transition and receive external kicks.

2.2 Dynamical Systems

A dynamical system is any system that changes over time. Any state of a dynamical system depends on the previous state such that for the change in state we get

$$x_{t+1} = Ax_t \tag{2.1}$$

where A describes a discrete mapping that governs the evolution of state x_t to x_{t+1}. This is an autonomous dynamical system, since it assumes that the parameters or the environment of the system do not change. For the non-autonomous case, the system depends both on the previous state and on input, denoted as b:

$$x_{t+1} = Ax_t + b \tag{2.2}$$

The rate of change of x can inform us whether a system is stable or unstable. If the rate of change is zero, so $x_{t+1} = x_t$, the state at t is a fixed point or an equilibrium point. Small perturbations from a fixed point can generally lead to one of two behaviors: Either the system diverges away from that point in which case the fixed point is unstable, or it converges towards it, making it stable. An unstable fixed point is a repellor, whereas a stable fixed point is an attractor.
If a system has multiple fixed points, large perturbations could move the system from one fixed point to another. This transition can happen at various timescales; for example, if the system responds slowly to a particular disturbance, the transition is more gradual rather than abrupt [ZDS]. If the system is continuously perturbed, e.g. by external stimuli, it might stay continuously in a transitional phase.
An attractor or repellor of a dynamical system does not necessarily have to be a point but can be a set of points, a curve, a torus shape, etc. In this thesis, we consider the nervous system or brain of worms to be a nonlinear dynamical system in which attractors and repellors are the driving force. If, for example, the system is perturbed by external stimuli, it will eventually move back to some *resting* state (attractor) that correlates with typical behavior.

2.2.1 Autonomous and non-autonomous dynamical systems

An autonomous dynamical system that is often subject in studies for modeling neural dynamics is the *Switching Linear Dynamical System* (SLDS): a system is composed of discrete dynamical subsystems and a function that determines a switching between these subsystems. In the field of SLDS this function is commonly defined by a deterministic process, giving the system autonomy [Sun06]. Thus, in the context of neural dynamics, neural activity pattern at any time point is the result of the dynamics of a subsystem that is *active* at that time, according to the switching rule. Biologically speaking, a subsystem or latent state may correspond to a neural population that drives a specific behavior. A non-autonomous dynamical system is often referred to as a control system. Here, a continuous brain state, such as a neural activity pattern, is given by a global dynamical system and external inputs, or control signals. A control signal might be a neuromodulator, biological compounds that regulate the activity of neural populations. It could also be sensory feedback, e.g. a sensory neuron senses an elevated oxygen level and *pushes* the worm to exhibit avoidant behavior.

Switching Linear Dynamical Systems

In the field of dynamical systems modeling there is a widespread assumption that any nonlinear complex dynamics can be broken down into simpler linear dynamics. Switching Linear Dynamical Systems (SLDS) models presume that the system that is composed of these simple latent subsystems, autonomously switches from one subsystem to another.

A switching linear dynamical system can be formalized as:

$$x_{t+1} = A_{z_{t-1}} x_t, \tag{2.3}$$

where z denotes a discrete latent state that corresponds to a subsystem.

Control Systems

Control Theory is a field that studies how a system can be *controlled* to achieve a desired state. Originally from engineering sciences, control theory was linked to many other fields based on the notion that control mechanisms can be found in many natural systems [Son13]. There are two main purposes of control mechanisms:

- *correct* for errors in a model via feedback
- *optimize* a good model

An example of a closed-loop control system is active sensing, where an organism performs a motor action to acquire more sensory information and that information in turn influences the next motor action [MC20]. The sensory information here is the feedback, which we call the *control signal*. A control system can be linear or nonlinear.

$$x_{t+1} = Ax_t + Bu_t \tag{2.4}$$

In this thesis, we use the theory of controllability only at a conceptual level. In particular, capturing non-linear dynamics, or aspects thereof, has been previously addressed through the formulation of a control problem [FZK20].

2.3 Manifold Hypothesis

A manifold is a topological space that can be locally mapped to the euclidean space $\mathbb{R}^n$ without losing its topological properties. Thus, if a complex, curved space is a manifold, we can assume that the same fundamental notions and tools that apply to euclidean space hold here. A manifold can lie in a high-dimensional space $\mathbb{R}^D$ but be homeomorphic to a low-dimensional space $\mathbb{R}^d$ with $d < D$ [Cay08].
In other words, the intrinsic dimension of the manifold d is much lower than the space D in which the manifold is embedded. The manifold hypothesis proposes that real data lie on low-dimensional latent manifolds. If we consider, for example, a neural state space that encompasses all possible states of a brain and each state is described by N dimensions where N = number of neurons, we can assume that in that state space lies a manifold structure with fewer dimensions n < N.
The process of finding manifold structures is called Manifold Learning and comprises a set of non-linear techniques. In general, the task of discovering a low-dimensional latent

space or mapping from high-dimensional data is known as dimensionality reduction. Despite this subtle difference between Manifold Learning and Dimensionality Reduction, low-dimensional spaces discovered by linear methods are also often referred to manifolds or manifold-like, as we will see in Section 3.1. One of the well-researched and established dimensionality reduction techniques is principal component analysis (PCA), which is a linear method.

2.3.1 Principal Component Analysis

PCA performs a linear transformation of the data in a way that reduces its dimensionality while maximizing the explained variance of the data. This results in a few linear combinations of the original variables. These linear combinations, or principal components (PCs), are found on the basis of the following idea: There is a vector $\alpha_1' x$ of size p that maximizes the variance of the data points, where p is the number of variables and ' denotes a transpose. Then another vector $\alpha_2' x$, which is not correlated with the previous vector but maximizes the variance the most after the previous one, is discovered, etc. The vectors $\alpha_1' x, \alpha_2' x, ..\alpha_p' x$ are PCs, where each α_k, for k=1,..,p is the eigenvector of the covariance matrix $Cov(x) = \Sigma$ corresponding to the k-th largest eigenvalue [Jol02].
The first few PCs will explain most variance in the data if many of the original variables are heavily correlated. However, the data are explained in all its entirety by the sum of all PCs.

2.3.2 Intrinsic Dimensionality

Intrinsic Dimension (ID) is the minimum number of dimensions needed to sufficiently capture all information or variance in the data. Many manifold learning techniques that map high-dimensional data onto a lower-dimensional space require a priori knowledge of the ID.
To estimate the ID of data in advance, various algorithms have been proposed. A simple estimator, for example, is a projection method, like PCA, applied to many subspaces to find the best subspace based on some projection score, such as explained variance. Projection methods are typically global methods because they assume that all of the data is sampled from a single subspace. As data might also be sampled from multiple manifolds with different dimensions, local ID estimators were introduced [CS16]. Despite the growth of the field, the task of ID estimation becomes more complex with the variety of data and consensus on a single well-established ID estimator is still lacking. Thus, in this thesis, we choose the number of dimensions of manifolds based on the following criteria: Visualizability and Interpretability.
We will primarily visit manifolds or low-dimensional representations of 3 dimensions, as manifolds of more dimensions are difficult to visualize and link to behavioral correlates.

2.4 Artificial Neural Networks

An artificial neural network, or just neural network (NN), is a computational model that originated with the idea to resemble biological neural networks. It has nodes or neurons that are connected, and they receive input signals that are passed through computations and then to other nodes. These nodes are organized into layers; an input layer, one or many "hidden" layers, and an output layer.
A simple layer is the linear layer that applies a linear transformation to the input, formally as

$$y_i = \sum_{j=1}^{n} W_{ij} x_j + b_i, \tag{2.5}$$

where W is the weight matrix and b is the bias. Note that this equation is similar to Eq. 2.4. A linear layer is fully connected, that is, each input is connected to every node. Figure 2.2 shows a schematic of the structure of a linear layer.

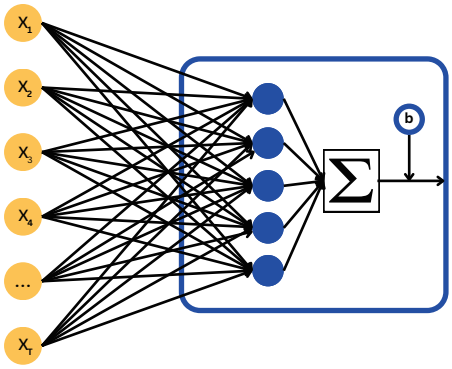


Figure 2.2: **A linear layer is a basic building block of a feed-forward neural network.** What is also known as a single-layer perceptron, is a linear combination of weights and an input vector.

Both W and b are model parameters that are learned during an optimization task. A defined loss function is minimized by propagating gradients that reflect error terms back to the model, allowing for the update of these parameters.

A neural network can be made up of many (linear) layers that are connected by an activation function, enabling the network to capture nonlinear relationships. If information passes from one layer to another unidirectionally, we refer to this network as a feed-forward network. In this thesis, we will implement a feed-forward neural network with fully connected linear layers.

2.5 Methods for Clustering and Classification

2.5.1 K-means

Here, we use the standard k-means algorithm, the Lloyd algorithm, which starts with k cluster centroids, where k is defined by the user. Each observation in the data is then assigned to the cluster centroids that it is closest to, given by the Euclidean distance d $= \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$. Then, the centroids are re-calculated as the mean of the assigned observations. This clustering method is implemented in and used from sklearn [Bui+13]. The default initialization of the centroids in this implementation is k-means++, which selects centroids from the observations given the empirical probability that they contribute to the overall variance.
After clustering the neural manifold data points with k-means, we compare the cluster membership with the dataset membership of the data points. To this end we use the AMI score, which is calculated as

$$AMI(U,V) = \frac{MI(U,V) - E(MI(U,V))}{\text{avg}(H(U),H(V)) - E(MI(U,V))} \tag{2.6}$$

where MI denotes mutual information, E of MI denotes Expected Mutual Information and H is the entropy. AMI computes the mutual information between two clusterings and adjusted this score based on mutual information due to chance [Bui+13].

2.5.2 Support Vector Machine

For classification, we utilize a Support Vector Machine Classifier (SVC). The idea behind Support Vector Machines is that any nonlinear data can be separated into two classes by a linear subspace. It achieves this by applying a kernel trick, projecting data onto a higher-dimensional space where the points can be linearly separated. Figure 2.3 gives an intuition of linear separability.

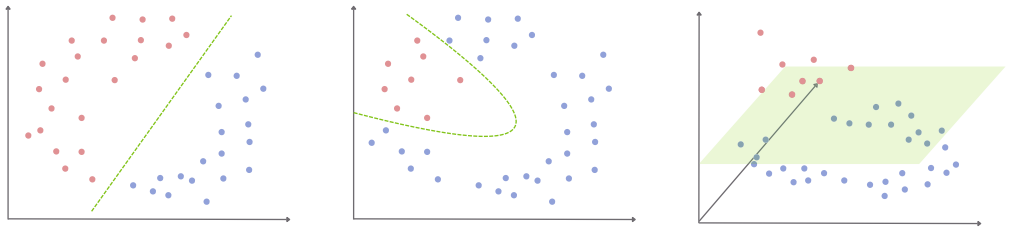


Figure 2.3: **Support Vector Machines can linearly separate data points by projecting them onto a higher-dimensional space.**

SVC can be extended further to the multiclass case with a One-vs-One strategy, where, instead of a single multiclass classifier, multiple binary classifiers are learned. When applied to predict the class label of an observation, the class that is selected by most classifiers is assigned [Bui+13]. Here, a class label is the recording or identity.
To evaluate the performance of a classification model we compute the following scores: accuracy, precision, recall, and f1. Accuracy is calculated as

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives} \tag{2.7}$$

and determines how close the class predictions are to the true class, thus identity.
Please refer to Appendix 8 for the remaining equations and note that precision, recall and the f1-score typically provide added value to the evaluation when there is class imbalance, which is not the case here.[2] Still, we added them for completion as these metrics are commonly considered together.

[2]The accuracy score could be high because the model performs well in the majority class. Precision, recall and f1 account for that by looking at relative proportions of correctly identified class labels, rather than absolute numbers.

CHAPTER 3

Related Work

In this chapter we will first dive into how different techniques for learning manifold structures have been applied to various organisms to gain insight on neural mechanisms. In the following section we contrast multiple prevalent dynamical systems models based on model complexity and interpretability, and go into more detail on the Decomposed Linear Dynamical Systems (dLDS) model as it will serve as a primary focus in this thesis. Finally, we will discuss some methods for modeling neural dynamics or neural manifolds proposed in recent times that are worth exploring in the future.

3.1 Neural Manifolds

The notion that a low-dimensional manifold confines coordinated neural activity is becoming increasingly prevalent in the field of computational neuroscience. As imaging techniques improve and the number of neurons that can be captured simultaneously increases, population studies overtake single-neuron studies, and the question arises of how to make sense of these large amounts of data.
A foundational review by Yu and Cunningham discusses the use of dimensionality reduction in the literature in the context of testing population structure hypotheses [CY14]. They highlight the importance of population analyses in settings where single-neuron responses cannot be mapped to observable sensory inputs or motor outputs, but rather appear to be part of a higher-level neural mechanism. In such settings, numerous studies have been conducted that use dimensionality reduction methods to abstract single neuron activities into shared latent variables that can be linked to observables. These shared latent variables are the dimensions or neural modes that span a manifold and can be found with common linear techniques such as PCA or factor analysis (FA), which differs from PCA in the sense that the neural modes are not necessarily composites of existing variables.

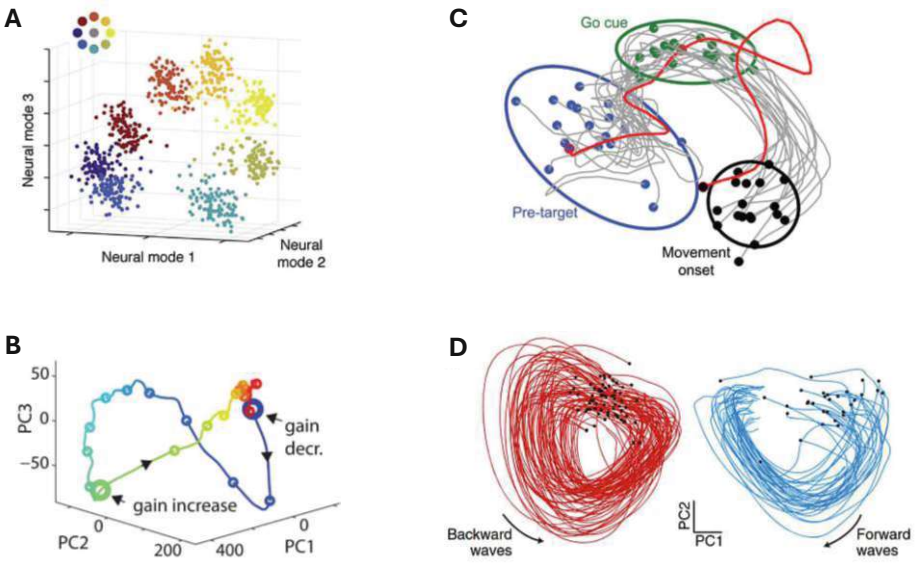


Figure 3.1: **Neural Manifolds for Motor Control. A.** Neural modes reveal a clustering of neural activities that are linked to specific movements [San+09][Gal+17]. **B.** A low-dimensional representation displays four types of neural dynamics during motor control in zebrafish [Ahr+12]. **C.** Neural trajectories of individual trials in a reaching-task collapse into a 2-dimensional manifold that shows distinct features for movement and for the preparation of movement [Chu+10]. **D.** Two-dimensional manifolds shows a difference in muscle activations timing for different locomotor waves in fruit flies [Lem+15].

FA methods were used by Santhanam et al. to correct for the variability of the neural response in the control of neural prosthetics used by patients with motor dysfunction. They identified neural modes that facilitate the discovery of target-specific latent activity, which can then be mapped to discrete neural responses that control neural prosthetics, improving the overall performance of such aids [San+09][Gal+17]. Another study for the control of movement obtained neural modes based on PCA and observed that there is a separation between preparatory muscle activity and generative muscle activity, with numerous other studies building on this work to formulate and test motor-cortex related hypotheses [Gal+17][Kau+14][Chu+10].
PCA has also been used in motor circuit studies in invertebrates such as zebrafish, fruit flies, and nematodes. Using PCA, a phase space representation of zebrafish neural recordings revealed four types of neural dynamics during motor adaptation that correlate with distinct brain areas [Ahr+12]. PCA was also used in drosophila to systematically discover muscle activation timings for backward and forward crawling [Lem+15][Zar+19]. In their 2015 work, Kato et al. found that the motor command sequence in *C. elegans* is embedded in a low-dimensional cyclical manifold spanned by three neural modes computed with PCA [Kat+15]. Quantifications of the neural manifold revealed previously

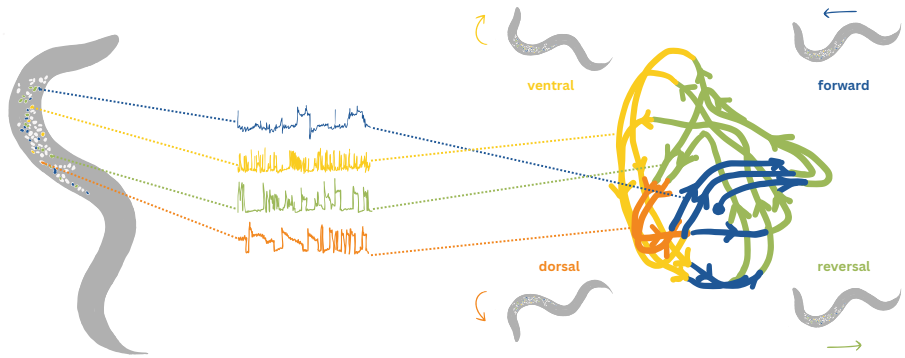


Figure 3.2: **The major motor command sequence is embedded in a low-dimensional cyclical neural state manifold.** Adapted from Kato et al. [Kat+15]

.

unidentified discrete fine-grained behavior states (see Figures 3.1 and 3.2).
The strength of linear methods lies especially in their mathematical tractability and interpretability, and in this thesis, we will visit PCA to construct neural manifolds on *C. elegans* neural activity data. In general, linear methods suffice if the neural dynamics explored account for a subregion of the manifold, where local linear approximations are possible [Gal+17]. However, if we assume that our data is nonlinear, then we can expect the underlying manifold to be nonlinear as well and non-linear dimensionality techniques, such as Locally Linear Embedding (LLE), t-distributed stochastic neighbor embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP) or Kernel-PCA (k-PCA), which is an extension of PCA that allows nonlinearity by mapping data onto a higher-dimensional space before PCA is performed, might be more suitable. However, the cost of these state-of-the-art nonlinear dimensionality reduction techniques is the lack of interpretability and identifiability, that is, the reproducibility of a representation that is learned, as discussed in a recent study by Schneider et al. [SLM23].
In this thesis, we want to show that PCA can recover an interpretable and identifiable low-dimensional representation of neural activity that is consistent across 23 whole-brain imaging recordings.

3.2 Modeling Whole Brain Dynamics as Dynamical Systems

Previous work has attempted to understand the evolution of neural states over time with computational models [DS21]. The aim of a computational model of the brain is often two-fold: accurately describe neural activity through model complexity and offer explanations to real neural mechanisms through model interpretability. However, as we will discuss in the next two sections, many state-of-the-art methods achieve either one of those goals, with a tendency towards model complexity, leaving a gap in the field

of interpretable models. The design of a computational model depends on the aspect of the brain that one wants to understand. In this thesis, we want to understand the latent brain states embedded in the neural manifold, specifically their stability, and the transitions between them.

3.2.1 Learning Latent States

A method from probability theory that serves both as a dimensionality reduction technique to discern states and as a model of state-transition dynamics is the Hidden Markov Model (HMM). HMM is an extension of Markov models, where observations are the outcome of subsequent events, also known as Markov process. In HMM, Markov processes are hidden, and they are learned and identified as discrete clusters or *modules*. These modules can then be interpreted as distinct discrete states. Another piece of an HMM that can be interpreted is the transition probability matrix: the probability of moving from one state to another. HMM has been used in fMRI neural activity data to discover distinct latent states that can be mapped to task-related motor cortex activities, and it has been used in behavior studies in nematodes to identify both different behaviors and state sequences [Vid+18][Gal+13]. A behavior study in fruit flies also discusses the use of HMMs to identify internal states underlying song patterning during courtship and moment-to-moment variation in decisions. However, they argue that standard HMMs produce transition probabilities that are constant, i. e., they do not change upon external inputs, and introduce a generalization of HMM, which allows for more characterization of the transition dynamics [CPM19]. Other variations of HMMs have been used with the aim of allowing for a better description of state dynamics over short timescales, like AR-HMM, which models for each distinct state how it changes over time [Wil+15][GF23].

3.2.2 Learning Transition Dynamics

A similar avenue to AR-HMM that promises even greater descriptive power is SLDS, which assumes underlying hidden dynamics within each state (see Section 2.2.1). Recurrent SLDS (rSLDS) proposed by Linderman et al. extends SLDS such that a transition probability between latent states depends not only on the current latent state, but also on the current observed location in phase space [Lin+16]. In a subsequent study, they applied rSLDS to multiple *C. elegans* neural activity recordings by incorporating the model into a hierarchical framework and obtained both model parameters that are consistent across individuals and model parameters that are unique to each worm [Lin+19].
SLDS and its variations belong to the field of dynamical systems, of which many methods have been explored in recent years. A caveat of SLDS is the assumption that the latent subsystems between which the high-level system switches never overlap in time. This is addressed in recent work by Mudrik et al., proposing Decomposed Linear Dynamical Systems (dLDS), a new dynamical systems model [Mud+23].

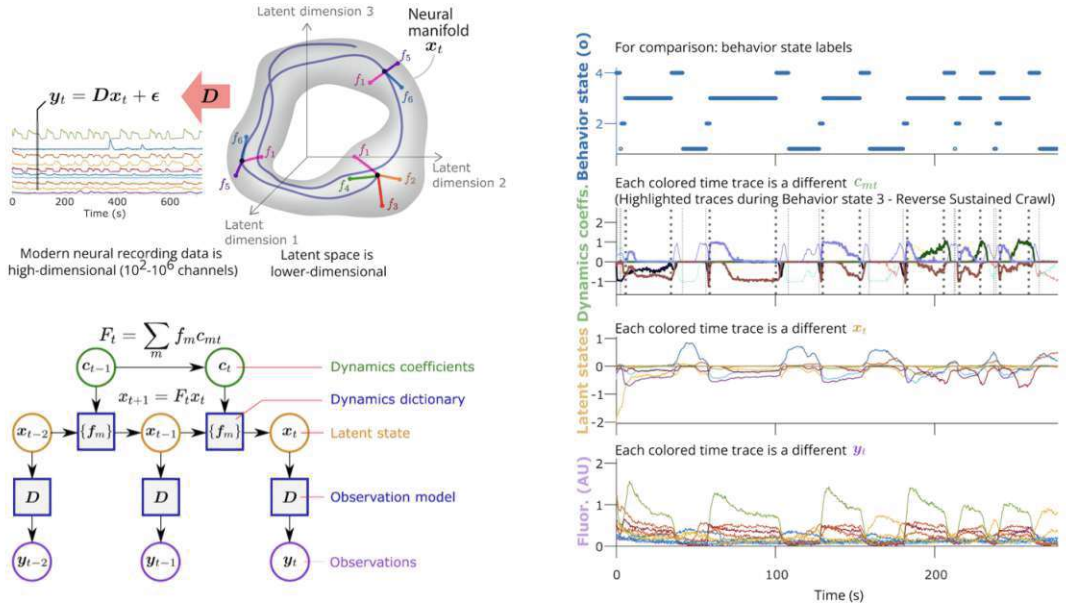


Figure 3.3: **dLDS infers behavior states from *C. elegans* whole-brain recordings.** dLDS models neural activity as dynamical flows on a low-dimensional manifold (Top Left). The model relates to a given data point at time t through the dynamics F_t and the dynamics coefficient c_t (Bottom Left). When applied to *C. elegans* data, the learned dynamics coefficients are correlated with true behavior states of individuals (Right) [YMC24].

Decomposed dynamical system model captures smooth transition dynamics

dLDS assumes that neural activity is produced by an autonomous dynamical system, where each state of the system is not described by a single subsystem, but by a linear combination of a few subsystems. The activity of the subsystems is controlled by a variable, the coefficients. The strength of dLDS lies in the interpretability and expressivity of these coefficients through their magnitude and continuity. In a subsequent study, Yezerets et al. applied dLDS on *C. elegans* and successfully decoded various discrete and continuous characteristics, such as behavior states, oxygen levels and speed from the coefficients alone (see Figure 3.3) [YMC24]. They were also able to correlate gradual changes in coefficient magnitude with the activity of certain neurons, providing further explanation for the participation of neurons in certain higher-order states. However, dLDS assumes that neural activity is predictable, not accounting for unknown forces or inputs that may perturb the system. In this thesis, we want to fill this gap by extending dLDS to the non-autonomous case using Control Theory.
Moreover, the EM procedure that underlies the dLDS algorithm poses a few drawbacks, as we will explore later: computational complexity due to its iterative nature and slow convergence, rigidity as its structure is fixed, and sensitivity to initialization, which is

an inherent difficulty in dynamical systems modeling. To account for these caveats, we will formalize dLDS as a neural network and apply the resulting model to the neural manifolds constructed with the aforementioned dimensionality reduction techniques.

3.2.3 Learning Control Signals

A framework that weaves control theory in traditional LDS was introduced by Fieseler et al., providing a different, yet biologically interpretable, explanation to how a system is actuated. They learn a global linear dynamical system with temporally sparse control signals, using Dynamic Mode Decomposition with Control (DMDc) [FZK20]. DMDc separates the underlying dynamics of a system from its control signals, which could be external inputs that perturb the system or internal inputs that aim to stabilize the system, by finding the best-fit linear system $X' = AX + BU$ (see Eq. 2.3), where A is the state dynamics and U are the controls [PBK16]. Since DMDc requires knowledge of U, Fieseler et al. extend the method to learn sparse control signals in an unsupervised fashion, utilizing sparse optimization. They applied the model on *C. elegans* imaging data and discovered previously unidentified neurons that contribute to learned control signals and therefore possibly to brain state transitions. Morrison et al. replaced the global linear dynamical system with a nonlinear dynamical system that has multiple fixed points [MFK21]. Furthermore, they show how model parameters can be linked to changes in stability within the system, potentially revealing the neural mechanism behind pirouettes.

3.3 Recent Advances in Neural Dynamics Modeling and Manifold Learning

3.3.1 Recurrent Neural Networks for Modeling Dynamics

A shortcoming of many computational models discussed here is their ability, or lack thereof, to capture long-term dependencies. A Recurrent Neural Network (RNN) is a Machine Learning model that is widely used for sequential data, e.g. continuous states that depend not on one previous state but on many previous states. It operates on a sequence of inputs by passing information about the current output to itself as a new input and repeating the process; therefore, the network being *recurrent*.
A promising approach to understand neural circuits is to model long-term neural dynamics with an RNN, which is essentially a nonlinear dynamical system, and disassemble the model into interpretable components. In other words, instead of modeling the parts that constitute whole-brain dynamics, we can model the dynamics and break the model into simple pieces. This framework is discussed in a key review by Vyas et al. They argue that to study the RNN, and hence the state space it constructs, one can use multiple LDS, each of which approximates a subregion of the entire state space. A nonlinear state space can be linearized around fixed points, and literature also shows how such linearization can help interpret local dynamics of RNNs [Vya+20].
Another autonomous dynamical systems model is LFADS (Latent factor analysis via dynamical systems), introduced in 2016 by Sussilo et al. LFADS learns a mapping between spiking data and a compressed representation with an encoder RNN and uses a generator RNN to approximate the dynamics. Furthermore, LFADS was extended to model non-autonomous dynamics, a system that depends not only on the current state but also on some eventual inputs, or as we refer to in this thesis, control signals, that perturb the system and that can be inferred [Sus+16][Pan+18].
Although LFADS comes close to what we want to achieve in this thesis regarding model complexity, it does not find latent neural states or state dynamics that are stable because it generates one global dynamical system. Later work by Sussilo et al. co-trains an SLDS variant to linearly approximate LFADS around fixed points [SLS21].

3.3.2 Dynamic Inverse Reinforcement Learning

Ashwood et al. proposed an entirely different approach to studying behavior and its neural underpinnings: they learn the dynamic intrinsic reward function of mice during navigation in a maze with a dynamic inverse reinforcement learning (DIRL) strategy [AJP22]. This reward function emits a linear combination of spatial maps at each time point, where a spatial map describes the amount of reward in each region of the environment. The reward function then defines the extent to which each spatial map is active at a given time. Using this model, they recovered some spatial maps between which the animals transitioned, characterizing explorative behavior in mice. Furthermore, they proposed that neural correlates of inferred reward variability could be used to study brain

dynamics during exploration. A possible future avenue for understanding whole-brain dynamics with DIRL is to recover spatial maps where each region or node of the map is a brain state, thus discovering brain states where the reward is highest, potentially indicating stability.

3.3.3 Neural Networks for Manifold Learning

Returning to neural manifolds, recent work by Kumar et al. involves a neural network that learns to preserve neural dynamics related to a given behavioral context in a low-dimensional Markovian embedding [Kum+23]. They trained the network on multiple *C. elegans* individuals and obtained a consistent geometry that revealed predominant behavior motifs that were more fine-grained than the information to which the network had access.
Schneider et al. proposed another nonlinear framework that promises interpretability and identifiability [SLM23]. CEBRA is a recent self-supervised encoding method that produces a latent space consistent over multiple data. It is a convolutional neural network that is trained on sample pairs in the latent space found with a contrastive learning algorithm: for each data point, a *positive* and a *negative* example based on some similarity measure is defined, and then the model is optimized to learn an embedding space such that the data points mapped to that space are closer to positive embeddings and farther away from negative embeddings.

3.3.4 A Framework for Manifold Universality

Combining probabilistic and deterministic approaches, Brennan C. and Proekt A. constructed a consistent manifold across *C. elegans* individuals by averaging neural trajectories with respect to the phase of the flux, which defines observed transition probabilities and was computed with asymmetric diffusion maps. They showed that the phase of the flux is an important macroscopic variable that constitutes a manifold that is conserved not only in a single individual but in many individuals [BP17].

CHAPTER 4

Manifold Alignment

In this chapter, we discuss methods for preprocessing the 23 whole brain Ca^{2+} imaging recordings, for dimensionality reduction to construct the canonical neural manifold, and methods for evaluating the universality of the manifold. As we explore the preprocessing methods, we continuously assess the visual structure of the resulting manifold produced by each preprocessing method. This prompts a crucial question: What do we expect a (canonical) neural manifold to look like?

In general, we want the manifold to exhibit a geometry that can be interpreted. Prior work has defined the interpretability of a manifold through a lack of trajectory tangling, that is, little crossing of trajectories while traveling to different directions [Per+23]. Perkins et al. emphasize how low-tangled trajectories are typically sparsely distributed such that certain regions of the state space remain unoccupied. Furthermore, low trajectory tangling indicates stereotyped sequences of neural states, where similar neural states consistently lead to similar future neural states.

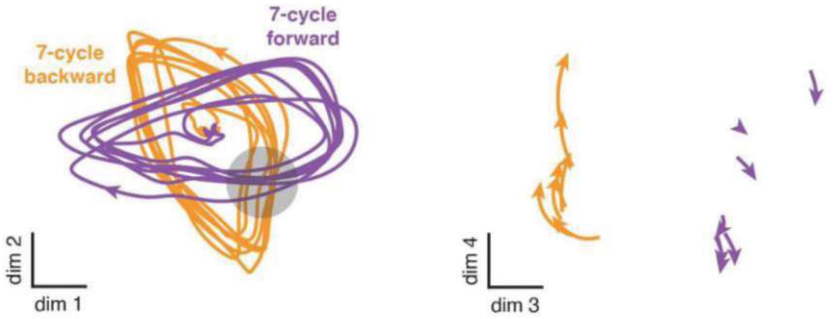


Figure 4.1: **Low tangling implies separation between trajectories that might otherwise be close** [Per+23].

Kumar et al. have referred to these stereotyped, low-tangled trajectories as *bundles* [Kum+23]. They propose that the interpretability of manifolds depends on how compact and well-separated these bundles are. In this thesis, our aim is to discover a manifold that is characterized by trajectories that have low variance within a behavior state and a structure that displays large voids in the state space.

4.1 Methods for Data Preprocessing

To directly compare recordings, we establish a pipeline of preprocessing techniques that are catered to the specific Ca^{2+}-Imaging data. Most of these methods are part of a standard data preprocessing pipeline [Li19]. We will go into detail about the methods that are required or used for the final version of the neural manifold. Note that we explored more methods that did not make it into this thesis. For example, we applied robust regression diagnostics, like Mahalanobis distances on data to spot outliers in observations. However, we found that the discovered outliers did not have a substantial impact on the resulting manifolds. We also computed first-order derivatives of the data after preprocessing them and looked at their projection onto a low-dimensional space, but the resulting manifold structures were visually difficult to interpret due to high tangling.

4.1.1 Data Cleaning

Across all recordings, 95 unique neurons were identified (IDed), of which only 10 neurons are IDed in each of the 23 recordings. Quantifications of neuron IDs are found in the Appendix A1, A2. Most remaining neurons occur in most of the datasets, but to ensure data integrity, we decided to omit all neurons that are IDed less than 10 times and we end up with 74 unique neurons.

4.1.2 Up- and Downsampling

As the acquisition rate of the 23 recordings varies from 2.9 to 5 fps as seen in A3, we resample all data so that all recordings have the same number of frames or time points. Nine datasets were recorded at a frame rate of 3.26 fps and ten datasets were upsampled to that rate, while four were downsampled. This was done with linear interpolation, which is implemented in the Python package numpy [Har+20]. With 3.26 fps, each recording now has 3529 time points. However, the first and last 100 time points were removed as we observed some edge effects, such as a substantial brightness at the beginning and dimness at the end.
To resample behavior state annotations accordingly, we added a NaN value for each interpolated data point and then replaced that behavior state value via backward filling, i. e. we took the previous state.

4.1.3 Quartile Normalization

Due to inconsistencies within experimental settings, e.g. different settings of laser power or varying expression levels of GCamP across worms, the same neuron might have different scales in two or more recordings, making them incomparable. Furthermore, the effects of bleaching, which is a decay of fluorescence over time, and bleach correction (2.1.2) persist in some recordings, leading to a downward or upward shift of the baseline. Hence, we normalize each neuron per recording by a quantile range. We found that a larger quantile range of the 1st and 99th percentiles is capable of capturing larger amplitudes while minimizing the influence of outliers. For normalization, we use the RobustScaler algorithm from sklearn [Bui+13]. We assume that a true baseline F value exists and that a lower percentile gives a robust estimate of this. To do this, we subtract the lower 20th percentile of all data points.

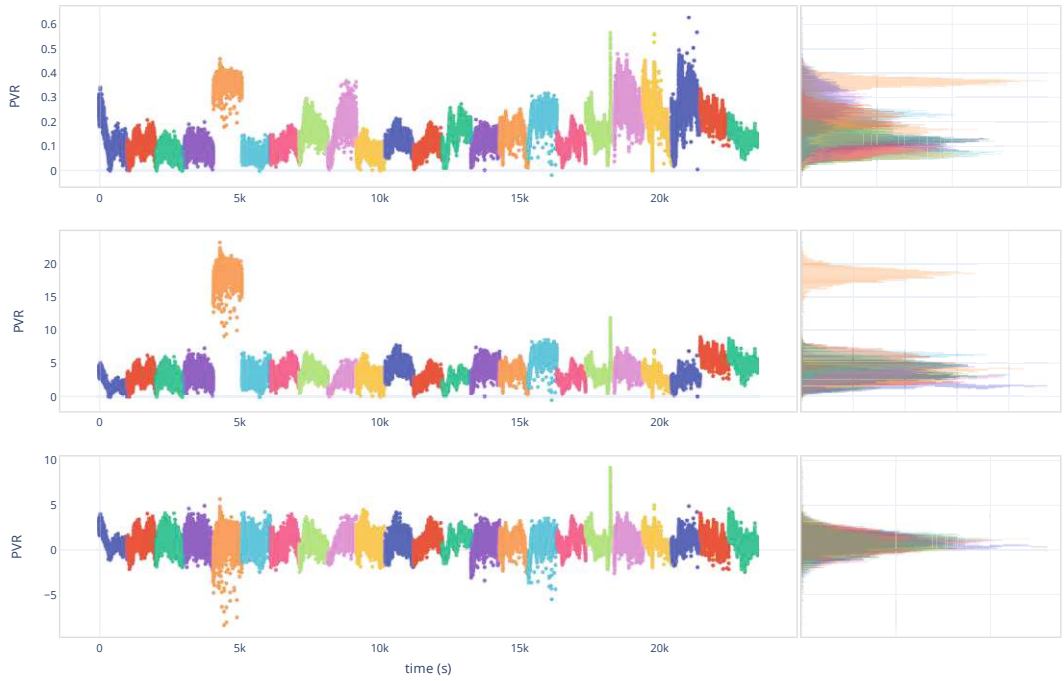


Figure 4.2: **Quartile Normalization minimizes variability in amplitude scale.** Data points from different recordings, indicated by coloring, show varying amplitude scale (Top). Applying a quartile normalization and a subtraction of a lower percentile reduces this variability (Middle and Bottom).

4.1.4 Time Delay Embedding

As we do not have access to the activity of all 302 neurons, one might wonder how representative our data is of the true state space. A theorem from dynamical systems theory, Taken's theorem, assumes that the true underlying dynamics of a system can

be reconstructed from a few observed variables. Taken's embedding gives then an augmentation of a sequence of data points (time series), by sampling multiple time intervals at different delays [Tak06]. Here, we choose a delay of 1 and an embedding dimension of 10, meaning we have 10 samples per variable, or neuron. To this end, we utilize the Takens Embedding implementation from the Giotto-tda Python package [Tau+21].
Time delay embedding is a common step in the literature before constructing a neural manifold. Brennan and Proekt found that without time delay embedding, their state transition probability matrix fails to produce any meaningful predictions [BP17]. It appears as if delay embedding restores the ability to predict across worms and that instantaneous neuronal activity is not sufficient to specify the state of the brain.
We can project our resampled and normalized recording data to a higher-dimensional space and then construct the lower-dimensional neural manifold from that. An idea that might seem counterintuitive, yet is the basis of some machine learning techniques, such as the kernel trick.

4.1.5 Data Imputation

To substitute missing neuronal traces in the recordings, a probabilistic principal component analysis (PPCA) method was used. PPCA is a prevalent data imputation technique that estimates missing values via Expectation-Maximization (EM). Like PCA, PPCA assumes a linear relationship between variables. A different way to put this is that if a neuron is missing in most recordings and is unique, in the sense that its traces do not correlate with traces of any other neuron, then it cannot be imputed.
To determine whether PPCA is a suitable technique for our data, the linearity between each neuron y and all other neurons X is measured by employing a Partial Least Squares Regression (PLSR) model, implemented in the Python package sklearn [Bui+13]. PLSR finds linear combinations of neurons that maximize the covariance between X and y. Then each linear model is evaluated by measuring the proportion of variance explained, which is known as the coefficient of determination, R^2. As more than 69% of all neuronal models have an average R^2 value of at least 0.7 with a standard deviation of 0.12, we assume that most neurons have a linear relationship and resume impute missing data with PPCA (see Figure A4). A5 shows an example of a dataset with imputed neuronal traces.

4.2 PCA for Dimensionality Reduction

To obtain a manifold-like structure, we fit a PCA model on all recordings, that is, we find a subspace spanned by three principal components that align with the directions of the greatest variance in the collection of the recordings.[1] We then project either single recordings or all recordings onto that space. Figure 4.3 shows the neural manifold we obtain after data imputation but before resampling, normalization, and time delay

[1] For the choice of the dimensionality please see Section 2.3.2.

embedding. This structure shows high tangling as trajectories belonging to the dorsal state are not visible, and trajectories belonging to the same behavior state are not start and end at similar states in the state space.

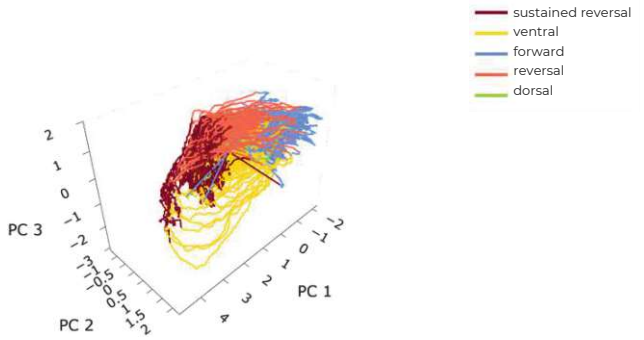


Figure 4.3: **PCA applied to unprocessed neural activity data.** Certain behavioral states remain submerged in the high-dimensional noise. This underscores the importance of effective preprocessing prior to PCA for revealing meaningful manifold structures.

After performing the preprocessing techniques, we will assess the neural manifold on these exact metrics. We will also look at characteristics typical for manifolds such as bifurcations, i. e. regions where trajectories branch into two different behavior states, indicating stochasticity within the system, or drifts, a reflection of gradual changes in neural activity over time, due to e. g. bleaching. Furthermore, we assess the consistency of the structure in individual datasets.

4.3 Evaluation of Canonical Manifold

To show that the neural manifold is canonical, i. e. representative of all individuals, we need to quantify the extent to which the variability in trajectories can be explained by the individuality of worms. In case the reduced space embeds local or individual features, we can assume that points in that space can be associated with the individuals to which they belong. Conversely, if points in the reduced space can be linked to the individuals to whom they belong, we can assume that the space embeds local features. This could indicate either that the recordings are not aligned or that the neural manifold inherently encodes local features. Here, we want to show that the neural manifold does not encode individuality. Furthermore, we want to know how well trajectories belonging to different behavior states are separated. To gain an understanding of the variability explained by both individuals and behavior states, we frame the question of canonicality as follows: does an observation in the low-dimensional state space have features that can be linked to a single individual? To answer this question, we define clustering and classification tasks.

4.3.1 Clustering

Clustering encompasses all computational and statistical methods for finding groupings of data points based on some characteristic. An unsupervised way of learning these groupings is provided by clustering methods. And as we obtain clusters of neural activity patterns that are similar to each other on some metric, we might find that these clusters correspond to the groupings determined from a behavioral feature.
Similarly, we want to determine whether an unsupervised clustering of data points lying on the canonical neural manifold correlates to a grouping of data points based on worm identity. If there is a high correspondence, we should believe that the variability within the canonical neural manifold is given by the individuality of *C. elegans* rather than by the behavior states. To test this, we will apply the clustering algorithm k-means to cluster data points from the neural manifold and compute the similarity between these predicted clusters and the true individual clusters via the adjusted mutual information (AMI) score, a common clustering evaluation metric.
To validate the AMI score, we apply Stratified K-Fold Cross-Validation (CV). CV tests a model on portions of the data, called folds, that are chosen with a given sampling strategy. The resulting metric score for all models is then averaged. Stratified K-Fold is a variation of CV in which samples are chosen such that the proportion of different labels, here recordings, is the same. In this way, we will not get a label imbalance in a fold, which in our case means that instead of training the clustering algorithm on observations that all belong to a single individual, we train it on observations from all individuals. The AMI score of the k-means clustering shows the dataset membership correspondence of both the low-dimensional data and the unpreprocessed high-dimensional data. In general, the AMI score practically ranges from 0 to 1, where 0 indicates random agreement between the predicted clustering and the membership of the dataset, and 1 indicates strong agreement.

4.3.2 Classification

A different way to test whether the neural manifold is invariant to identity is by measuring the performance of a recording classifier. In particular, we want to see, when we train a model that predicts the recording membership of an observation, if the model yields high accuracy after being trained on the high-dimensional data vs. the neural manifold. Again, we aim for low accuracy because we do not want observations from different recordings that share a similar pattern to be distinguishable.
We train an SVC with a nonlinear kernel and a One-vs-One strategy on both the unpreprocessed and preprocessed high-dimensional data and finally on the unpreprocessed and preprocessed low-dimensional data. We then evaluate the model with Stratified K-Fold CV based on our chosen metrics.

Behavior State Separation

As we define a new classification task based on the behavior state annotations, we can measure how well the neural manifold encodes behavior states.
The only other modification to the classification setup is that, instead of using the Stratified K-Fold CV, we evaluate the models using an expanding window CV. In general, we have time series data, it is important that we do not draw out samples at random time points but always a batch of samples that are connected through time. This would introduce a heavy class imbalance in a dataset classification task but not if we want to classify behavior states. We can leverage an expanding window split that splits the data first arbitrarily small and with every iteration the train and test set sizes get larger. In other words, this variation ensures that observations in a fold are subsequent in time on which the model is trained. To this end, we use the TimeSeriesSplit from sklearn [Bui+13].

CHAPTER 5

Controlled Decomposed Linear Dynamical Systems

The canonical neural manifold defines a low-dimensional state space that can be used as a proxy to model the brain of the worm. We observe distinguishable regions on the manifold by color-coding the data points by the annotated behavior states. These regions are believed to correspond to discrete latent states that together shape the dynamics that underlie locomotion.

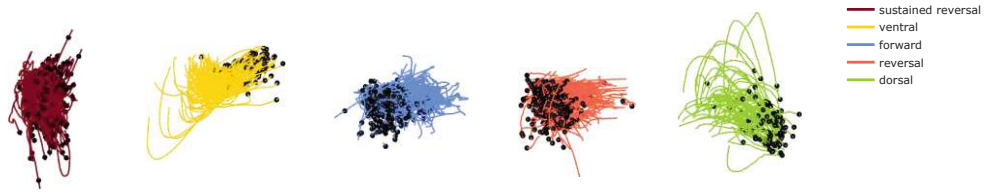


Figure 5.1: **Regions on the neural manifold that correspond to different behavior states can be modeled as subsystems of the brain.**

However, as Kato and others have observed a more fine-grained organization of the motor command sequence, the question remains how many latent states are actually present. To systematically discover latent states, a large effort has been made in the field to model latent states in an unsupervised way and correlate them to observable behavior.
The decomposed Linear Dynamical Systems (dLDS) model introduced by Mudrik et al. assumes that latent states can be modeled as linear dynamical systems (LDS) (see 3.2.2)

[Mud+23]. And each observation in the low-dimensional state space is described by a linear combination of one or more LDS. The transition between LDS is governed by a variable, the dynamics coefficients, that controls the activity of each LDS.
Neural activity then is seen as a dynamical flow on the neural manifold and the flow is locally described by a few dynamical systems. These dynamical systems do not change once they are found. Thus, dLDS assumes that the processes that produce neural activity are deterministic and are only of a slow scale. In other words, the brain is a stable system that moves from one brain state to another in a predictable manner. However, we observe fast-scale perturbations to the brain in the form of fast state switches (see Section 2.1.4. These perturbations are not predictable and they can not be expressed by fixed dynamical systems. Thus, we assume that the neural activity is produced by two processes: predictable slow-scale state dynamics and non-predictable fast-scale state dynamics. A model that can disentangle these processes is not only more realistic, but also provides a systematic way of discovering properties that are recurring versus ones that are sporadic.
In this chapter, we discuss dLDS and how we extend it to include control signals. We first go over the original iterative algorithm and the drawbacks that it poses. We will then pivot to a neural network architecture implemented after dLDS and in the final section we propose the same architecture extended by control signals.

5.1 dLDS

Each LDS, known as the subsystem f, has a corresponding coefficient c that describes to what extent a subsystem is active at a time, providing an additional interpretable component of the model. Furthermore, dLDS learns a mapping, here named the observation model and denoted as D, between the observed states y and the continuous latent states x, or manifold, which is then decomposed into subsystems. If the observed states y are observations of points in a low-dimensional manifold, D can also be set to be an identity matrix such that $y_t = x_t$. dLDS is regularized to produce sparse and smooth coefficients, i.e. only a few subsystems are active at a given time and their activity does not change rapidly, encouraging the system to exhibit smooth transitions between states.

5.1.1 Original Algorithm

dLDS uses an Expectation-Maximization (EM) procedure for finding model coefficients x, c and model parameters D, f. In the Expectation step, the latent states and the coefficients are inferred at each time step with the following LASSO problem:

$$\hat{x}_t, \hat{c}_t = \arg\min_{x_t, c_t} \left[\|y_t - Dx_t\|_2^2 + \lambda_0 \|x_t - \tilde{F}_t c_t\|_2^2 + \lambda_1 \|x_t\|_1 + \lambda_2 \|c_t\|_1 + \lambda_3 \|c_t - \hat{c}_t\|_2^2 \right] \tag{5.1}$$

where $\tilde{F}_t c_t$ is the dynamics prediction $\sum_{m=1}^{M} c_{mt} f_m \hat{x}_{t-1}$ and computes the current continuous latent state x_t given the f dynamics of all subsystems, the previous continuous

latent state x_{t-1} and the current coefficients c_t of all subsystems. $\lambda_2\|c_t\|_1$ sparsifies the coefficients so that the subsystems are not equally *active*. In this thesis, we will refer to this term as the sparsity term. $\lambda_3\|c_t - \hat{c}_t\|_2^2$ is the smoothness term as it encourages smoothness of the coefficients with the L2-norm of the differences between each pair of consecutive time points in the coefficients. Finally, in the Maximization step, the model parameters D and f are updated by computing a gradient over them.

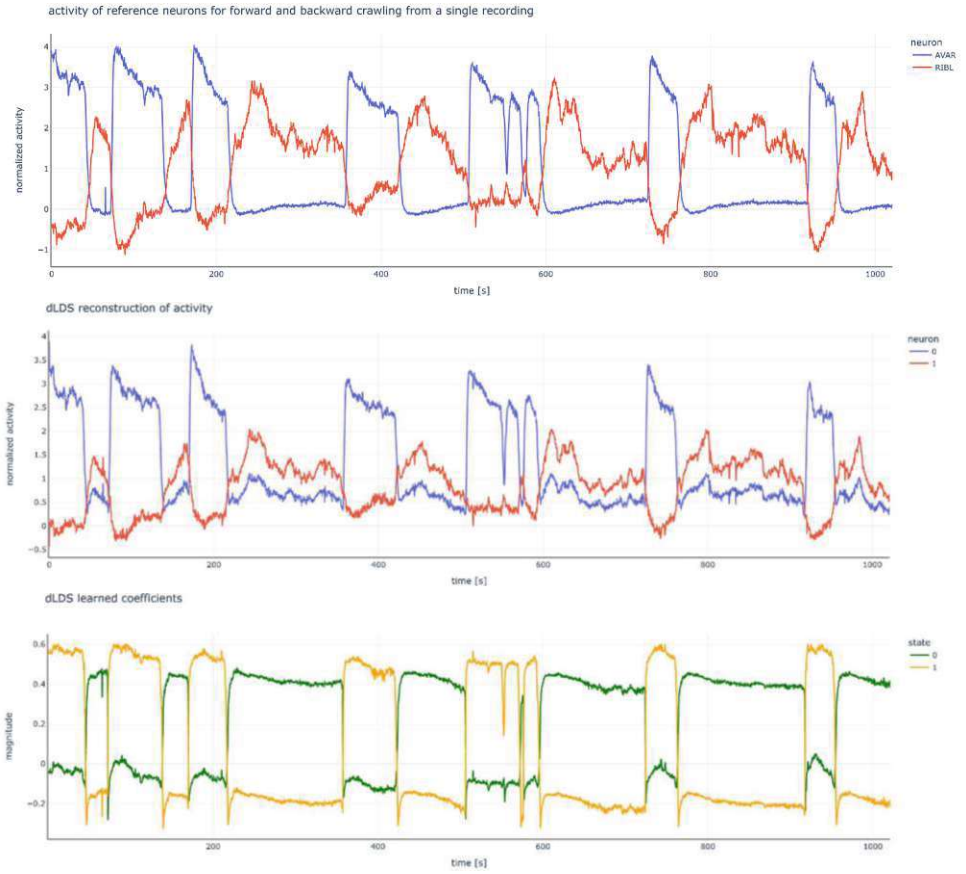


Figure 5.2: **dLDS infers dynamics coefficients of the two major behaviors in *C. elegans*.** When applied to two major motor command neurons, AVAR and RIBL, in a single recording, dLDS is able to reconstruct the traces and infer dynamics coefficients.

We applied dLDS on the recordings, the neural manifold, and synthetic data generated from an SLDS (see Section 6.1.1 for data generation). Figure 5.2 shows an example of a dLDS reconstruction of neural activity from two neurons: AVAR, a reversal driver neuron, and RIBL, a forward driver neuron. We can observe a noticeable deviation from the original activity traces, especially in the reconstruction of RIBL. Furthermore, the resulting learned coefficients illustrate the case for fast-scale states and the need to extract

them into a separate component. At time points 500 to 600 the learned coefficients seem to express fluctuations in magnitude at a similar frequency or regularity as the fluctuations in activity. In other words, at certain time periods, the coefficients seem to encourage state transitions at a faster rate than in other periods. A similar trend is observed in other experiments.
An extensive hyperparameter sweep to find optimal values for the smoothness and regularization terms could potentially increase the performance of dLDS. However, as dLDS has a long runtime and does not allow for parallelization due to its iterative nature, such an exploration seems not feasible.
Furthermore, the EM algorithm is rigid in terms of integrating additional components such as the control signals, as the update rules are fixed. These caveats incentivize a formalization of the decomposed Linear Dynamical System (dLDS) as a feed-forward neural network. A neural network essentially parameterizes steps in the iterative algorithm and learns the parameters from data, discarding the need for a manual definition of operations.

5.1.2 Neural Network Architecture

The proposed network consists of N linear layers, f_1, f_2, .., f_N, where N is a manually defined number of subsystems, a single learnable parameter, the coefficients c and another linear layer bias that learns a bias from the coefficients. We decoupled the bias term from the subsystem linear layers, as the bias is related to the coefficients. The input of the network is a $m \times d$ matrix x, where m is the number of variables, such as the latent variables spanning the neural manifold, and d is the number of time frames. The output x_{t+1} of the network is also a $m \times d$ matrix, computed as

$$x_{t+1} = \sum_{n=1}^{N} (c_{nt} f_n(x_t) + bias(c_t)), \tag{5.2}$$

where c is of dimensions $N \times d$ and $f_n(x)$ is the n-th linear layer. Figure 5.3 shows a schematic of the architecture. This layer transforms a given input by a weight matrix, which is similar to an LDS 2.3. The weight matrix thus defines a mapping between x and x_{t+1}. This mapping is learned in an optimization task that defines a certain loss function that needs to be minimized.
We implement the neural network by leveraging the Python package Pytorch [Pas+19].

Loss Functions

We define three loss functions: a reconstruction loss and a smoothness and regularization term as in the dLDS equation 5.1. The reconstruction loss is calculated as the Mean Squared Error,

$$MSE = \frac{1}{d}\sum_{i=1}^{d}(x_{i+1} - \hat{x}_{i+1})^2, \tag{5.3}$$

which is the squared difference between each element of the computed output $\hat{x}_{i+1}$ and the true output x_{i+1}.
The equations from dLDS for the sparsity loss and smoothness loss stay:

$$sparsity = \lambda_1 \|\mathbf{c}\|_1 = \lambda_1 \sum_{i=1}^{d} |c_i| \tag{5.4}$$

$$smoothness = \lambda_2 \|(\mathbf{c}_{t+1} - \mathbf{c}_t)\|_2 = \lambda_2 \sqrt{\sum_{i=1}^{d}(c_{i+1} - c_i)^2} \tag{5.5}$$

The ideal values for the parameters λ_1 and λ_2 are discovered via a hyperparameter sweep as we will discuss in Section 6.3.

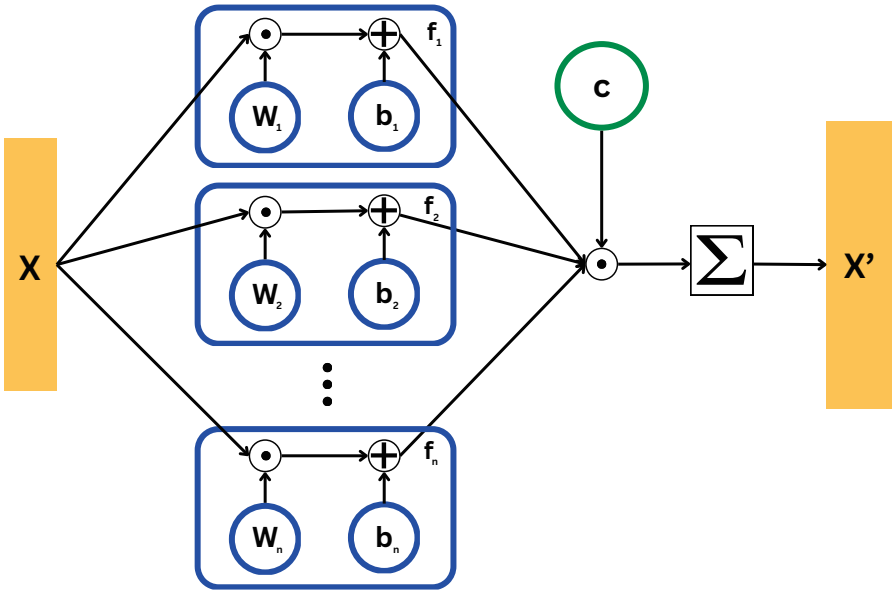


Figure 5.3: **dLDS formalised as a feed-forward neural network.**

Training

For the training of the network, all input data is loaded with PyTorch DataLoader, which incorporates a batching and shuffling process. During training, the model iterates over

each batch for multiple training cycles, also called epochs. Both the batch size and the number of epochs are manually defined variables that we will optimize in Section 6.3. In each training cycle, for a given batch, the output and the loss are calculated. Finally, to learn the weights of the parameters, the gradients of the losses are computed and backpropagated with Adam, an adaptive optimization algorithm based on stochastic gradient descent. The learning rate defines the magnitude of these gradients and is another tuning parameter (see 6.3).

5.2 cdLDS

We define the controlled decomposed Linear Dynamical Systems (cdLDS) model as

$$x_{t+1} = \sum_{n=1}^{N} (c_{nt} f_n(x_t) + B(u_t) + bias(c_t)), \tag{5.6}$$

where B is a linear layer without bias and u is a learnable parameter of dimensions $1 \times d$. u holds the control signals, i. e., a vector with sparse elements that influence the output.

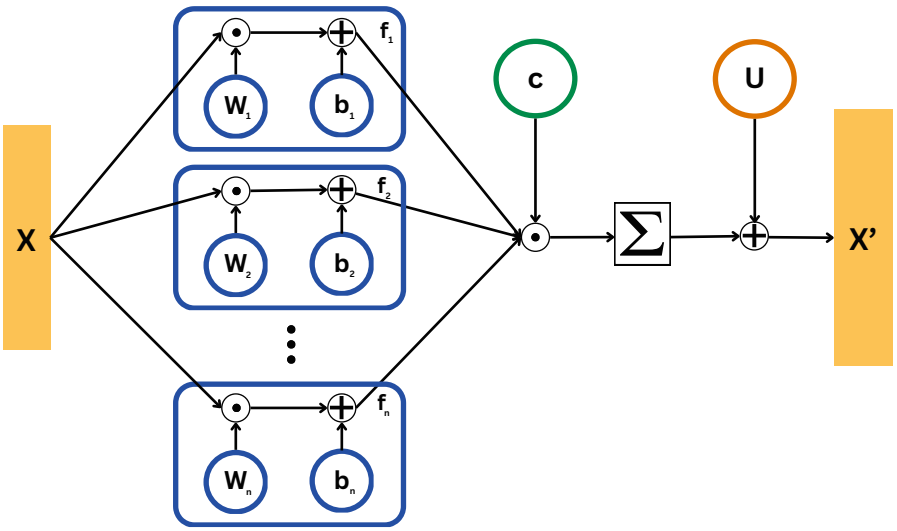


Figure 5.4: **The dLDS network extended by control signals.**

To ensure non-negative values for the control signals, u is passed through a RectifiedLinear (ReLu) layer

$$\mathrm{ReLU}(\tilde{u}) = \max(0, u) \tag{5.7}$$

before adding it to the output. Figure 5.4 shows a schematic of cdLDS.

Furthermore, we define another loss function, a control sparsity loss, as:

$$control\_sparsity = \lambda_3 \|\mathbf{u}\|_1 = \lambda_3 \sum_{i=1}^{d} |u_i| \tag{5.8}$$

This loss sparsifies the effective control signals.

CHAPTER 6

Experiments

To evaluate both the dLDS network and the cdLDS, we apply them to both toy data and the canonical neural manifold. For each dataset and each model, we define multiple experiments with different initializations. This chapter is outlined as follows: We first describe the advantages of toy data for testing models and how we generated them. We then go over the experimental conditions for the toy data. In the next section, we discuss the experimental conditions for the canonical neural manifold of *C. elegans* and finally, we briefly address hyperparameter tuning.

6.1 Toy Dynamics

Toy data are a simple small dataset, artificially generated to simulate a (realistic) system. They are designed for experimentation, to test the functionalities of an algorithm or model, or to illustrate their performance. Furthermore, their simplicity and size facilitate debugging and interpretation. A well-studied toy dataset for benchmarking dynamical system models is the Lorenz system, a chaotic system described by three simple differential equations. This system was also used by Mudrik et al. for demonstrating dLDS' performance [Mud+23]. However, as we want to model a more complex and biologically realistic system, we will construct the toy data manually.

6.1.1 Generation of Toy Data

We want our toy data to reflect the nature of the data that we are modeling. So in this case, we believe that the neural activity that we are observing is coming from a nonlinear system that can be split into simpler linear systems. A continuous state x at time t+1 is described as

$$x_{t+1} = \sum_{n=1}^{N} (c_{nt} A_n x_t + B_1 c_t), \tag{6.1}$$

where N is the total number of subsystems, A holds the dynamics of a subsystem, c denotes the dynamics coefficients, and t is a given time point.
The generation of the toy data can be separated into two tasks: Generating dynamics and generating data. We first construct the parameter that describes the dynamics of each subsystem, A. An arbitrary choice of the A matrices could result in unstable systems, where over time the continuous states of the system grow exponentially. This behavior is dictated by the eigenvalues of the corresponding A matrix.
If we come back to the first-order linear differential equation of a linear system as defined in Eq. 2.3, the solution of that system is defined as

$$x(t) = e^{At} x(0), \tag{6.2}$$

where x(0) is the initial state of the system and A is a $k \times k$ matrix. The eigendecomposition of A is given by

$$A = Q \lambda Q^{-1}, \tag{6.3}$$

where Q denotes the eigenvectors and λ is the eigenvalues of A. Then

$$e^{At} = Q e^{\lambda t} Q^{-1}. \tag{6.4}$$

$e^{\lambda t}$ gives us a diagonal matrix with entries $e^{\lambda^1 t}$,$e^{\lambda^2 t}$,..,$e^{\lambda^k t}$. Let λ^2 be a positive real value of 4, that means that the 2nd dimension of A is growing at a factor of 4 per time unit. As all components are linearly combined, the exponential growth of the 2nd dimension overpowers all other dimensions (even if other eigenvalues are negative).
Thus, the eigenvalues determine the stability of the entire system. Furthermore, if the eigenvalues are complex, the system will result in oscillatory behavior. Complex eigenvalues are of the form $a + bi$, where a denotes the real part and bi the imaginary part.
Therefore, we constrain random eigenvalues to be on the unit circle with a radius of 1. We sample complex values with an imaginary part equal to 0.06j which means that the system oscillates at a frequency of b=0.06 (see Figure 6.1). We then obtain a real matrix

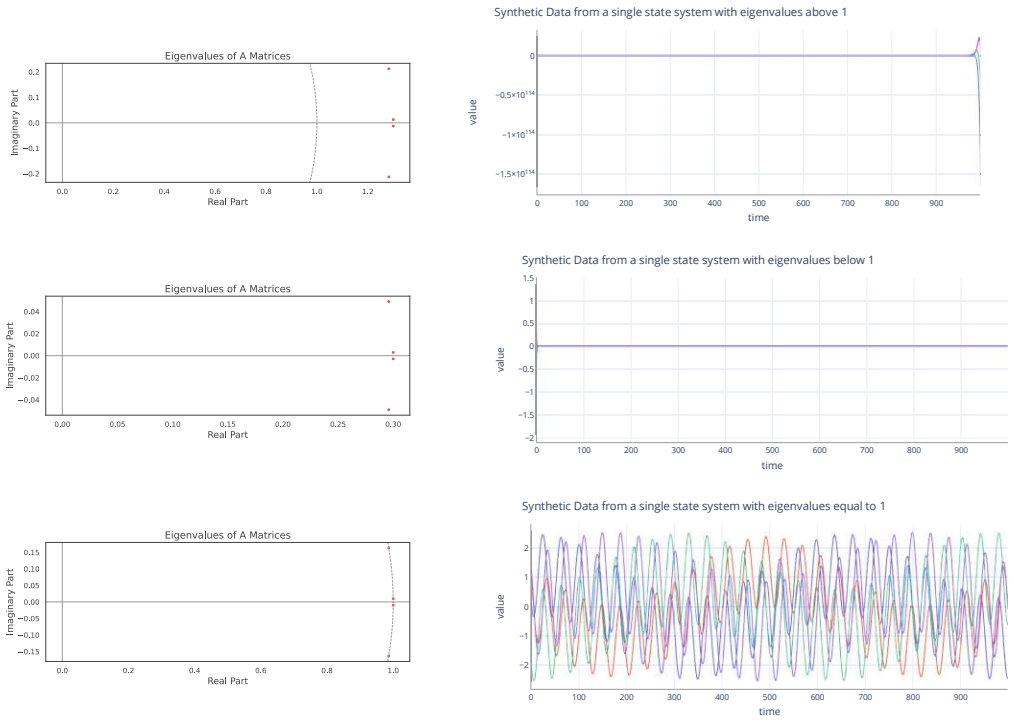


Figure 6.1: **The magnitude and complexity of the eigenvalues of a system determine its stability.** Any eigenvalues λ that are above 1 lead to an exponential growth of the state variables, marking an unstable system (Top). $\lambda < 1$ leads to a decay of the state variables and a stable system (Middle). $\lambda = 1$ makes a system that is neither exploding nor decaying (Bottom).

with complex eigenvalues by defining the Jordan normal form, which is a block diagonal matrix consisting of Jordan blocks. Here, a Jordan block C consists of k conjugate pairs of eigenvalues, i. e. $a + bi, a - bi$, such that

$$C = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}. \tag{6.5}$$

And the resulting Jordan normal form becomes

$$J = \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ 0 & 0 & C_3 \end{bmatrix}. \tag{6.6}$$

And J can be plugged into the eigendecomposition of A:

$$A = VJV^{-1}. \tag{6.7}$$

For V we construct a random matrix with the dimensions $2k \times 2k$ that is invertible. To ensure invertibility, we generate a random real $2k \times 2k$ matrix and perform a QR decomposition to obtain an orthogonal matrix, and orthogonal matrices are necessarily invertible.
Note that, due to the conjugate pairs, A becomes a squared matrix by default. Finally, A is multiplied by a parameter that defines the radius of the eigenvalues, to adjust the eigenvalue to our needs. Figure 6.1 illustrates the resulting dynamics with different eigenvalue radii. Ideally, we want eigenvalues that are not on the unit circle but very close to it and below. This results in a slow decay that resembles the activity of neurons that reach an equilibrium state. Figure 6.2 shows the dynamics of a single system with 4 dimensions.

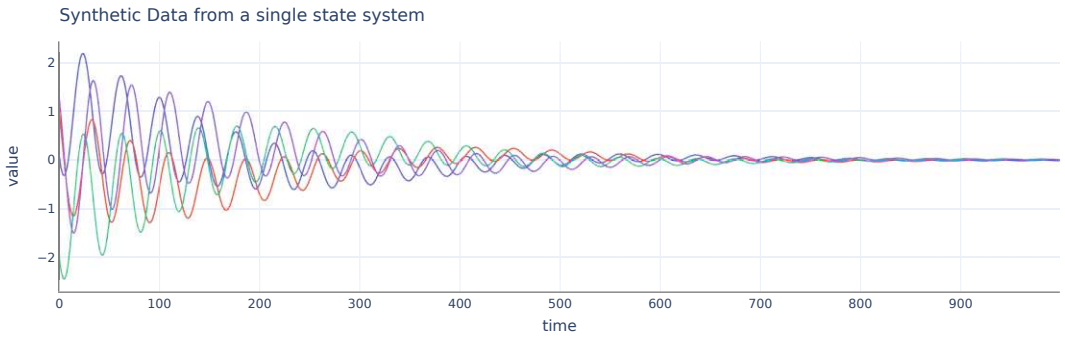


Figure 6.2: **A single system of four dimensions displays gradual decay with $\lambda = 0.94$.**

We use the same approach for constructing the matrix A for each subsystem, such that we have N number of As. Data points are then simulated from these dynamical systems with the Eq. 6.1. The coefficients c are generated as a one-hot encoded matrix of a random sequence of discrete states of size d. For 2 subsystems and 6 time points we may obtain a random c matrix such as:

$$c = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \tag{6.8}$$

which indicates that the first subsystem is *active* and the input has to pass through the first dynamical system for the first four time points and for the final two time points

through the second dynamical system.
Furthermore, c is added to each state as an offset to enforce a fixed point change. To this end we construct a random $k \times N$ matrix B_1 that is multiplied by the coefficients. Figure 7.9 illustrates the dynamics of two subsystems with 4 dimensions.

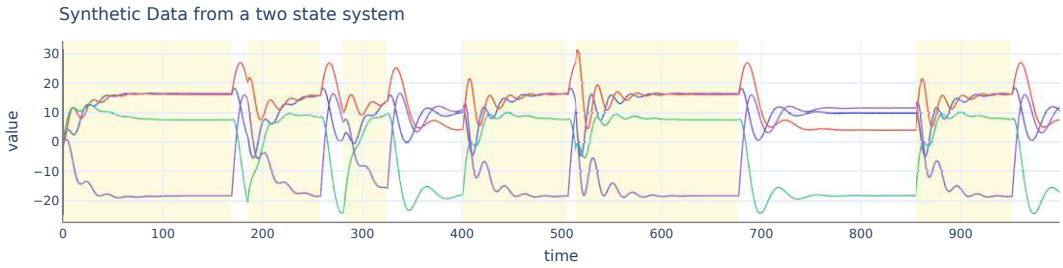


Figure 6.3: **A switching system composed of two subsystems.** The yellow highlights indicate the activity of the second subsystem.

Control Toy Data

To test the cdLDS model on the toy data we extend Eq. 6.1 to

$$x_{t+1} = \sum_{n=1}^{N} (c_{nt} A_n x_t + B_1 c_{nt} + B_2 u_t). \tag{6.9}$$

For the control signals u, we create a d-dimensional sparse vector with nonzero entries equal to 1, and we multiply it by a random $k \times 1$ matrix B_2. Figure 6.4 shows simulated data from a two state switching system and the generated control signals.

6.1.2 Sensitivity to Initial Conditions

For both dLDS and cdLDS, we define experimental conditions based on different initializations of the model parameters. In dynamical systems, the optimization of system parameters highly depends on the initialization of them, and different initializations could lead to dramatically different behaviors of the system. In neural networks, it has also been long known that initialization of the network weights affects the training process and the convergence to an ideal solution [LeC+02]. To test whether our neural network converges we formulate the following experimental conditions for dLDS:

1. **True** - Model parameters are initialized with the true components
2. **PartialRandom I** - The F dynamics are initialized randomly, while the coefficients remain true
3. **PartialRandom II** - The coefficients are initialized randomly, while the F dynamics remain true

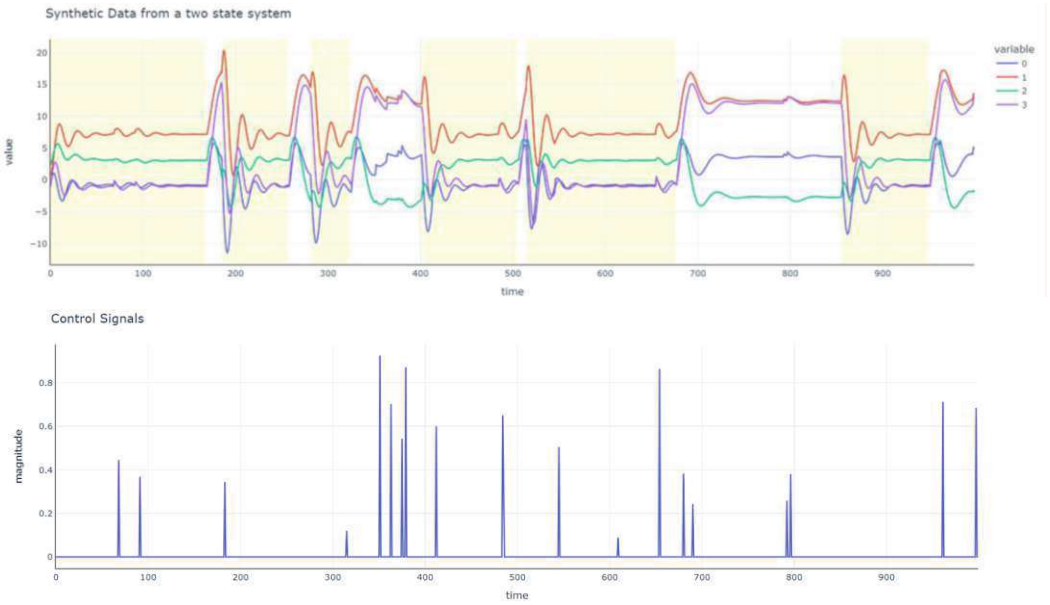


Figure 6.4: **A switching system composed of two subsystems and control signals.**

4. **Random** - Model parameters are initialized randomly

The first experimental condition is the simplest case, with all model parameters initialized with the true components generated in 6.1. Here, we expect the model parameters to not change during the training process. In contrast, the final experimental condition is the most challenging, but, ideally, the resulting model parameters should be close to the original ones.

For cdLDS we define the following experiments:

1. **TrueInit** - Model parameters are initialized with the true components
2. **PartialRandom** - F dynamics and the coefficients are initialized randomly, while the control signals remain true
3. **RandomInit** - Model parameters are initialized randomly

Note that for all experiments, we define the number of subsystems to be two, as this facilitates a more straightforward interpretation of the underlying dynamics. The primary objective of this analysis is to systematically compare all experimental settings and demonstrate the proof of principle of the model's capability to accurately produce coefficients, control signals, and data reconstructions.

6.2 *C. elegans* manifold

We apply dLDS and cdLDS on single recordings projected onto the canonical neural manifold with random initializations of the model parameters.
For cdLDS we will additionally explore an initialization for the control signals based on short state intervals (see Section 7.3).

6.3 Hyperparameter Sweep

To achieve optimal model performance, we have to find optimal values for the regularization terms as described in Section 5.1.2 and network parameters such as the number of epochs or the learning rate. These hyperparameters are separate from the learnable model parameters discussed previously. *Weights & Biases* (wandb), an AI developer platform supports Machine Learning Operations (MLOps) by providing tools for experiment tracking, visualization, and optimizing models [Bie20]. It provides a feature for finding optimal hyperparameters through "hyperparameter sweeps". Here, each sweep explores the defined value ranges for hyperparameters and optimizes them via Bayesian optimizations. For each experiment, we optimize for the following parameters:

- batch size
- epochs
- learning rate
- sparsity λ_1
- smoothness λ_2
- control sparsity λ_3 (for cdLDS)

Per experiment, a sweep contains up to a few hundred runs that are executed in parallel on multiple servers, which enables us to explore the hyperparameter space at a larger scale while reducing computational time. Examples of a hyperparameter sweep and the resulting losses are shown in A10.

6.4 Model Evaluation

We evaluate and compare different model configurations and experiments based on the reconstruction loss. To this end, we evaluate the MSE loss, described in 5.3, and visualize the data reconstruction on top of the ground truth for qualitative evaluation. Additionally, for the toy data, we compare experiments with different parameter initializations by calculating the coefficient correlation, as the Pearson correlation coefficient, to assess how close the learned coefficients are to the true ones. In the context of cdLDS, we also

compute a control correlation to evaluate the learned control signals. For the experiments on the neural manifold we plot the behavior states on top of the inferred coefficients to compare them.

CHAPTER 7

Results

7.1 Canonical Neural Manifold

A PCA-based low-dimensional representation of 23 whole-brain recordings shows a manifold-like structure

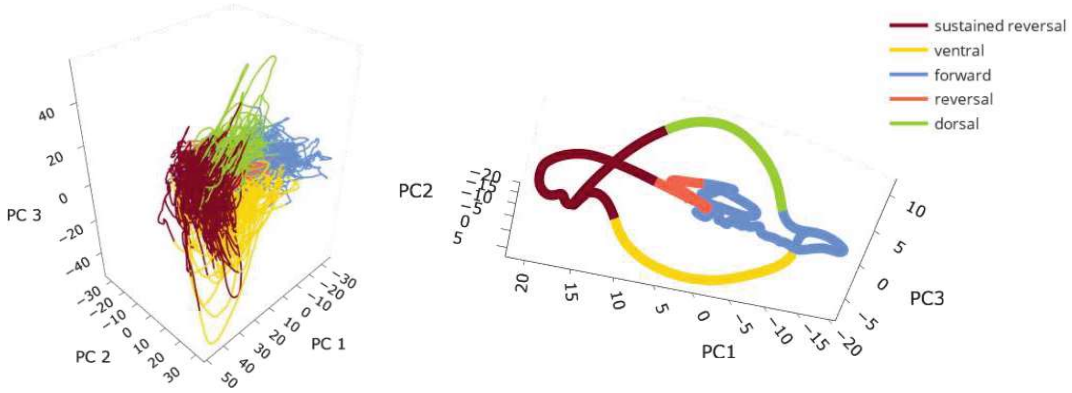


Figure 7.1: **PC Manifold of 23 whole-brain recordings.** The left figure shows the projection of the recordings onto a 3-dimensional state space. Observations in the state space are colored by the corresponding behavior state. On the right, we can see the phase plot, or the skeleton of the manifold, which is constructed by averaging trajectories within a behavior state.

The low-dimensional state space of the 23 recordings exhibits a distinct manifold-like structure, as shown in Figure 7.1. Examining individual recordings (as shown in A6) projected in that space highlights the consistency of this observed structure and offers insight into the specific contributions of each recording to the manifold. For example, the distinct shape observed in the dorsal and ventral turn of the neural manifold is present, to some extent, in all recordings, but especially in two recordings (8 and 13).
The neural manifold is characterized by a bifurcation in the sustained reversal state, which implies a decision point or an observed stochastic region of the manifold. Thus, it is not predictable whether a worm is going to enter a ventral turn state or a dorsal turn state. This has also been shown by Kato et al. [Kat+15]
In contrast, a deterministic transition is carved between a turn and the forward state as the ventral and dorsal trajectories collapse into a straightforward path merging into the forward trajectory. This reflects a destabilization of the turn states and a successful stabilization of the forward state. However, transitions from the forward or reversal state display stochasticity, which is characterized by substantial variability in the trajectories. Furthermore, we observe small-scale loops in the forward and reversal states that might indicate unstable or recurrent dynamics. This is reminiscent of pirouettes, events of failed state destabilization, as some trajectories in the state space seem to move from one behavior state region to another, only to subsequently loop back to their original region. We also observe a clear separation of the behavior states, contrasting the strong overlap seen in Figure 4.3. This could indicate scale differences between neurons corresponding to different behavior states, but we believe that this is not the case as the Appendix A7 shows a decrease in explained variance of the first principal component after applying preprocessing.

Clustering reveals differences due to individuality at the behavior state level

To show that the neural manifold is canonical, we quantify the explained variability due to individuality by formalizing a clustering task as described in Section 4.3. In particular, for data points that lie on the neural manifold, we expect two data points from the same cluster to not belong to the same individual. In contrast, for the high-dimensional observations, we expect a strong agreement between a clustering and the recordings, as a result of the record-to-record differences in scale.
However, the cluster analysis shows consistently low adjusted mutual information (AMI) scores in all three representations of the recordings: unpreprocessed high-dimensional (High-Dim (Raw)), unpreprocessed low-dimensional (Low-Dim (Raw)) and preprocessed low-dimensional (Neural Manifold) (see Table 7.1). The computed AMI scores, generally

AMI	High-Dim (Raw)	Low-Dim (Raw)	Neural Manifold
All states	0.332	0.277	0.219

Table 7.1: **Adjusted Mutual Information (AMI) scores for different representations of the recordings.**

ranging from 0 to 1, indicate little agreement between a clustering of 23 groups and 23 individual recordings across all variations of data. However, a progressive reduction in alignment is apparent after dimensionality reduction and preprocessing.
As we believe that a great amount of variability is due to the different behavior states that trajectories are linked to, we want to look at individuality at the state-level. Furthermore, we expect that specific states that represent the activity of influential neurons, such as the reversal driver-neuron AVA, characterize individual differences better.

AMI	High-Dim (Raw)	Neural Manifold
sustained reversal	0.423	0.26
reversal	0.482	0.306
ventral	0.513	0.306
dorsal	0.649	0.41
forward	0.424	0.385

Table 7.2: **Adjusted Mutual Information (AMI) scores for High-Dim (Raw) and Neural Manifold within each behavior state.**

In fact, Table 7.2 shows that within certain states, such as the turn states, the difference between the AMI scores of the low-dimensional and high-dimensional cases seems to be greater.
However, the agreement here seems to be higher overall, especially in the turn states.

This might be related to outlier trajectories that correspond to ventral and dorsal turns, as we observed in Section 4.2. These outliers also belong to only a few recordings, which would explain the increase in individuality in these states. That said, there is a clear imbalance in the number of data points between states, as seen in Figure 7.2.
Fewer observations are more susceptible to noise, outliers, and variability, in general. In the case of ventral and dorsal turns, the outliers might have a strong influence on the variability. Although the AMI scores indicate some agreement, when we match predicted and true clustering, we will find little to no correspondence, as we can see in Figure 7.2.

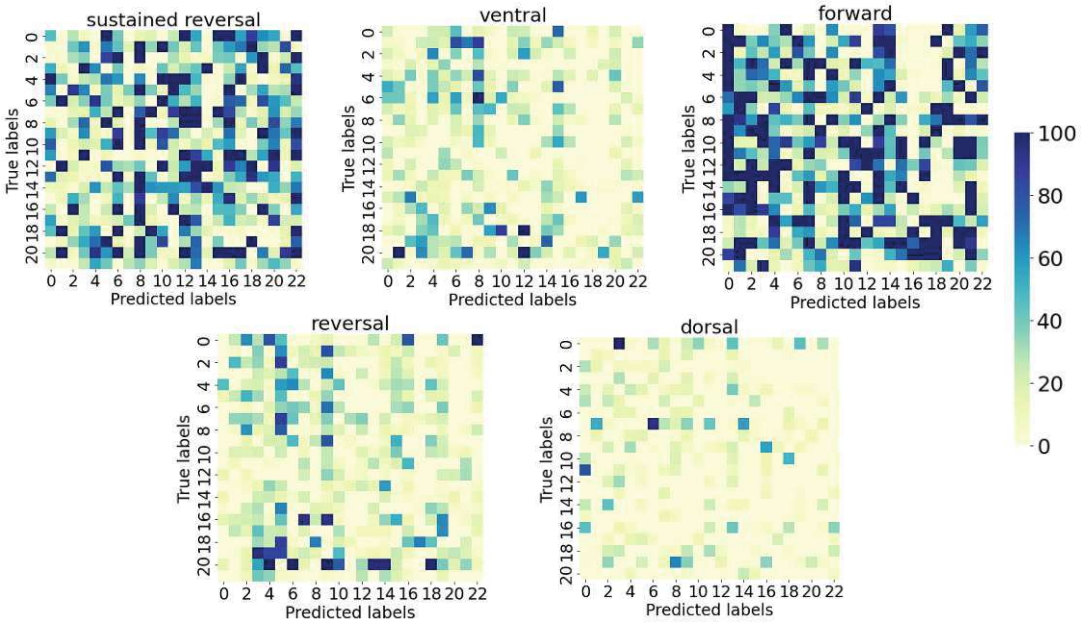


Figure 7.2: **Confusion matrices depicting the within-state performance of dataset membership clustering.** Confusion matrices of predicted cluster labels and true cluster labels within each state, where a label is the index of the recording or individual. Labels are matched using the Hungarian algorithm, a combinatorial optimization for assignment tasks. A strong agreement would be represented by a diagonal, which is not apparent in any of the behavior states.

Dimensionality reduction and preprocessing result in decreased variability caused by individuality.

To further validate the universality of our canonical manifold, we also define a classification task as outlined in Section 4.3. If trajectories on the canonical neural manifold cannot be uniquely assigned to an individual, we can infer that our neural manifold

encodes only global features.
After performing a classification task with four SVC models, each based on a different representation of the recordings, we find that the SVC trained and tested in the neural manifold produces by a large margin the lowest metric scores, as seen in Table 7.3. Even for high-dimensional data, preprocessing leads to a decrease in classification performance of more than 40%.

Metric	High-Dim (Raw)	High-Dim	Low-Dim (Raw)	Neural Manifold
accuracy	0.872	0.481	0.192	0.058
precision	0.905	0.522	0.181	0.061
recall	0.872	0.480	0.192	0.058
f1	0.869	0.475	0.166	0.058

Table 7.3: **Classification evaluation metrics reveal progressive reduction of performance for datasets after dimensionality reduction and preprocessing.**

The probability of guessing a random class is $\frac{1}{23} = 0.043$, as there is no class imbalance. The classification based on the neural manifold is close to chance, whereas the classification based on the raw low-dimensional representation is better than random guessing. Figure 7.3 shows the accuracy of the predictions as confusion matrices and, among them, a clear diagonal only in high-dimensional cases.

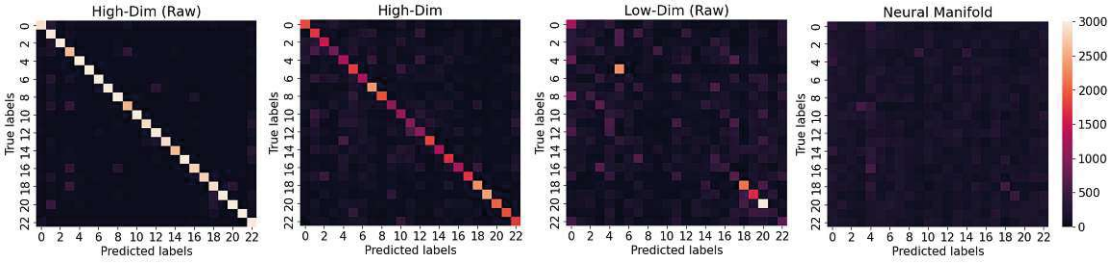


Figure 7.3: **Confusion matrices depicting the performance of dataset membership classification.** A cross-validated classification task performed on the (preprocessed) high-dimensional data yields the highest accuracy for classifying observations into their respective recordings. In contrast, the success of classifying observation on the neural manifold is close to chance.

Furthermore, the classification performance of the neural manifold-based model is equally low for each state but particularly for sustained reversal (see Table 7.4). As with clustering, for dorsal and ventral trajectories, we see increased precision in distinguishing recordings.

Metric	Forward	Reversal	Sustained Reversal	Dorsal	Ventral
accuracy	0.098	0.096	0.074	0.142	0.117
precision	0.102	0.095	0.086	0.182	0.105
recall	0.098	0.096	0.074	0.142	0.117
f1	0.096	0.089	0.075	0.125	0.102

Table 7.4: **Dataset classification within states reveals low identifiability in sustained reversal trajectories.** Confusion matrices showing the within-state classification performance are found in A8.

Canonical Neural Manifold retains a separation of the phase space into behavior states

We find that, after preprocessing, the behavior state classification performs better on all metrics for the high-dimensional data but only on one out of four metrics for the low-dimensional state space.

Metric	High-Dim (Raw)	High-Dim	Low-Dim (Raw)	Neural Manifold
Accuracy	0.732	0.901	0.809	0.760
Precision	0.777	0.901	0.78	0.799
Recall	0.732	0.901	0.809	0.760
F1	0.723	0.899	0.780	0.759

Table 7.5: **Comparison of Metrics Across Different Dimensions.** Confusion matrices of the state classification performance in each data are found in A9.

As seen in Table 7.5, preprocessing techniques facilitate great separability of behavior states in the high-dimensional state space and that the recording-to-recording variability lies primarily in behavioral differences rather than in individual differences. As behavior state separability is much lower for the preprocessed neural manifold compared to preprocessed high-dimensional data, it may be that the manifold is spanned by latent variables that encode other information in addition to that linked to the known behavior states. However, the separability is comparable to the raw high-dimensional state space and the raw low-dimensional state space, hence a behavior state separation is retained in the canonical neural manifold.

7.2 Decomposed Linear Dynamical Systems

dLDS performance is sensitive to the selection of hyperparameters

As described in Section 6.1.2, we first validate dLDS on the synthetic data under four experimental conditions to evaluate the robustness of the network. In the first condition, all model parameters are initialized with their true values. For the second and third conditions, we randomize either the dynamics F or the dynamics coefficients c. Finally, we initialize all model parameters randomly.
Upon applying the dLDS network to the toy data, we find that, within the four experimental conditions, the performance varies greatly with the choice of hyperparameters. This variability is reflected in the distribution of model losses seen in Figure 7.4. The distribution of each loss tends to increase with each experimental condition in both the overall loss and the reconstruction loss. However, for the coefficient correlation loss the model with the true components exhibits the largest distribution, which is an artifact of the log transformation, as the loss values for the coefficient correlation are very small in absolute numbers.

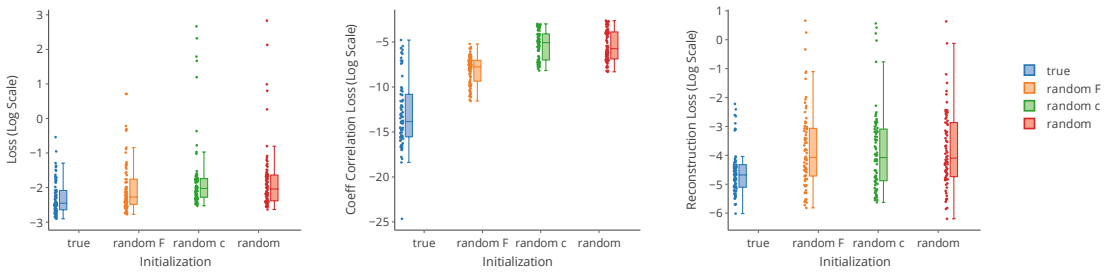


Figure 7.4: **dLDS loss distribution for each experimental condition.** The log-transformed loss distribution of 100 experiments illustrates a difference in variability across conditions, indicating different levels of sensitivity to hyperparameters.

The sensitivity to initial conditions is more prominent when the model parameters are initialized randomly, but it is already apparent in the case where the model parameters are initialized with the true components. For instance, Figure 7.5 shows an experiment run with arbitrarily chosen hyperparameters. Here we observe a poor reconstruction quality and noisy coefficients.
Figure 7.6 shows the importance of each hyperparameter for the different loss metrics and their correlation with the decrease or increase in losses. The sparsity regularization term has a large impact on the overall loss, and higher values for the parameter greatly impedes the performance of the model. Furthermore, while the smoothing term strongly affects the coefficient correlation loss, it appears to have a minimal impact on the overall loss. These results highlight the need for a rigorous search and optimization of hyperparameters as outlined in Section 6.3.

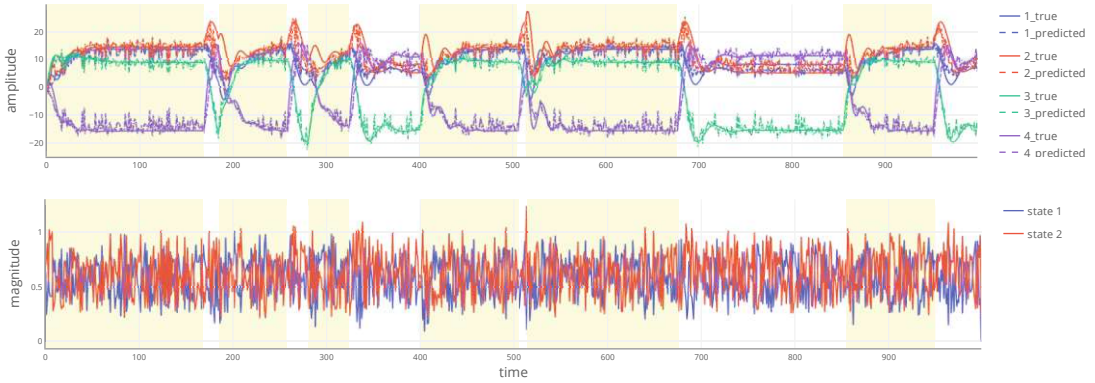


Figure 7.5: **dLDS applied to toy data with arbitrary hyperparameters.** The true active subsystem is highlighted in yellow.

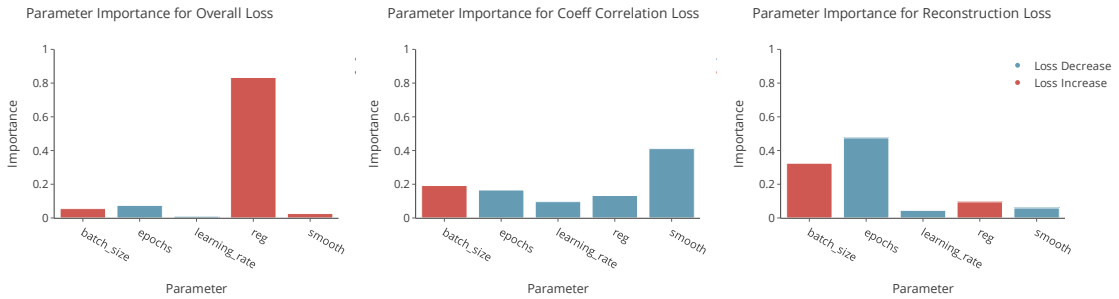


Figure 7.6: **Hyperparameter Importance for Random Initialization.** For each of the three losses, the reconstruction loss, the coefficient correlation loss and the overall loss, we computed the importance of each hyperparameter by training a random forest with the hyperparameters as the predictors and the loss as target. Then, the magnitude of a parameter indicates the extent of its importance. Furthermore, the color encodes the correlation of each parameter with the respective loss. Therefore, a negative correlation is interchangeable with a loss decrease and is colored as blue, while positive correlation is encoded as red.

A low dLDS loss does not imply interpretable coefficients

When implementing a model, a key question is what evaluation metrics or losses are required to obtain interpretable model parameters. To address this, we must examine what individual losses defined for dLDS mean in terms of model interpretability. In particular, when the true parameters of real data are unknown, can we expect the overall loss and the reconstruction loss of the model to be indicative of *accurate* output? Here we show that even when applied to synthetic data, dLDS can yield a low overall loss and reconstruct the data accurately while producing coefficients that are not

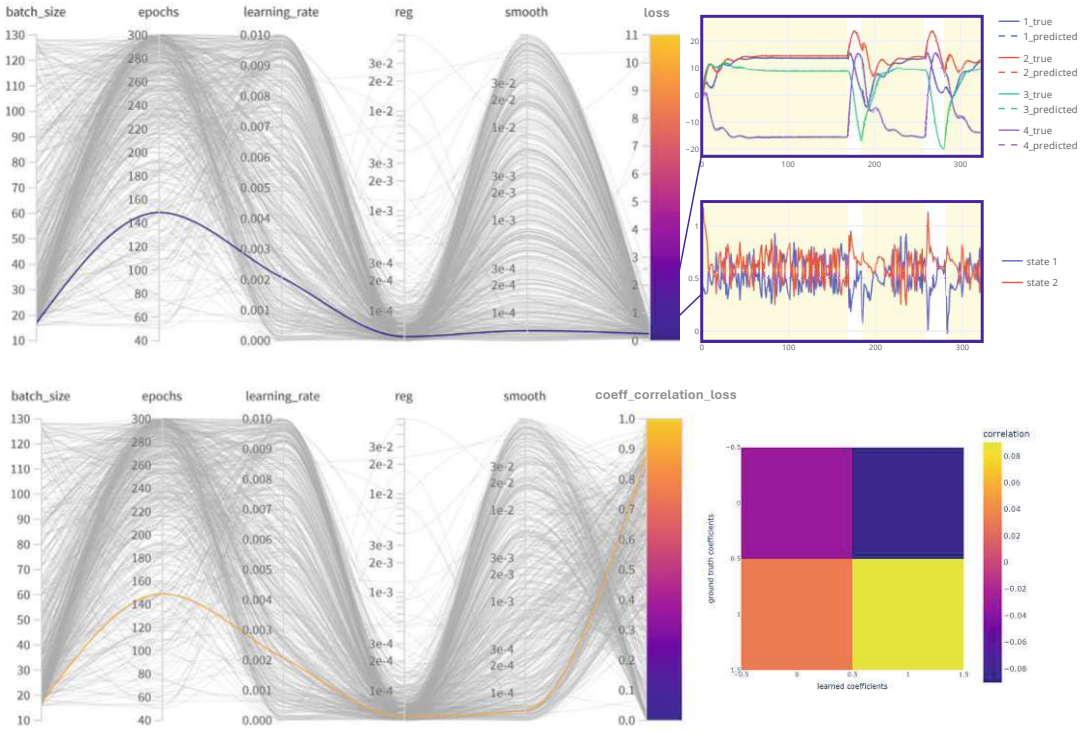


Figure 7.7: **dLDS model with a low loss yields uninterpretable coefficients.** An experiment that is in the bottom percentile of all experiments in terms of overall loss displays an accurate reconstruction but uninterpretable coefficients (Top). The coefficient correlation for the same experiment is in the upper percentile and shows weak agreement between the learned and the true coefficients (Bottom).

necessarily interpretable. Figure 7.7 illustrates an experiment with low loss and high reconstruction accuracy, yet the resulting coefficients are noisy and oscillatory, in contrast to the smooth coefficients we expected. This finding shows the necessity of evaluating dLDS based not only on reconstruction accuracy but also other metrics, such as coefficient correlation, to ensure a balance between model accuracy and model interpretability.

dLDS accurately infers model coefficients

Figure 7.8 shows the reconstruction and inferred coefficients of dLDS initialized with the true model components. We expected a marginal divergence from the true coefficients, which is apparent here at time points proximal to the state transition points. Furthermore, a random initialization of the dynamics, the coefficients, or the dynamics *and* the coefficients also leads to inferred coefficients comparable to the true ones, as seen in

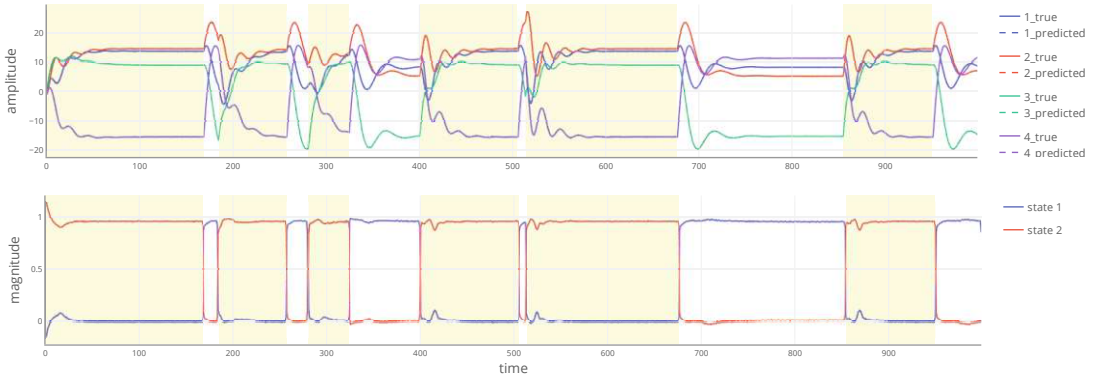


Figure 7.8: **dLDS with true components applied to toy data retains the true coefficients.** The reconstruction of the synthetic data (dashed) aligns with the true synthetic data (solid) (Top). dLDS retrieves the coefficients of two subsystems that align with the true coefficients (highlighted in yellow) (Bottom).

Figure 7.9. This result confirms that dLDS is capable of learning dynamics and the underlying latent states.

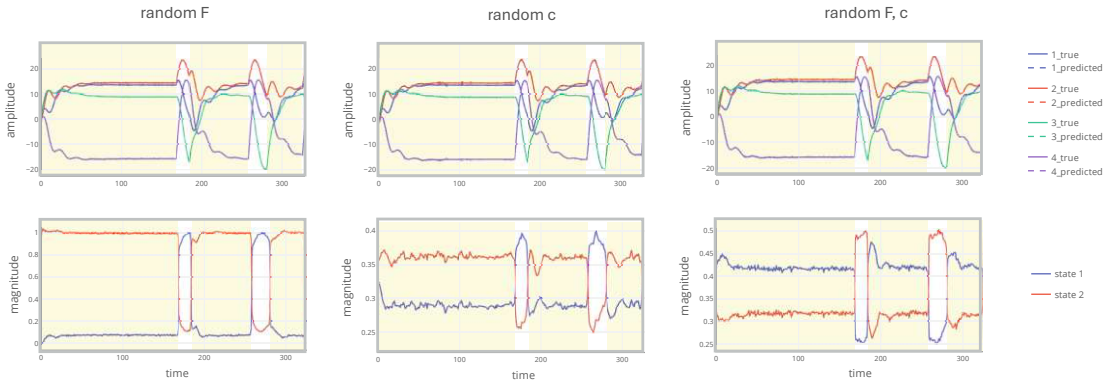


Figure 7.9: **dLDS applied to toy data yields coefficients close to the true coefficients.** Under the remaining three experimental conditions, dLDS successfully reconstructs the toy data (Top). With each condition, the coefficients of the subsystems become progressively more noisy (Bottom).

Neural Manifold latent states correspond to behavior states

When dLDS is applied to the canonical neural manifold, we cannot examine the inferred coefficients quantitatively as the true coefficients are unknown. However, we can qualitatively assess the inferred coefficients via characteristics that facilitate interpretability, for example those that are enforced by the smoothness and sparsity term. Furthermore, we investigate inferred coefficients in light of behavior state annotations.
Applying the dLDS model to a single recording projected on the canonical neural manifold reveals that the inferred coefficients of the two subsystems correspond to the two major behavior states: forward and reversal locomotion, as seen in Figure 7.10. This is

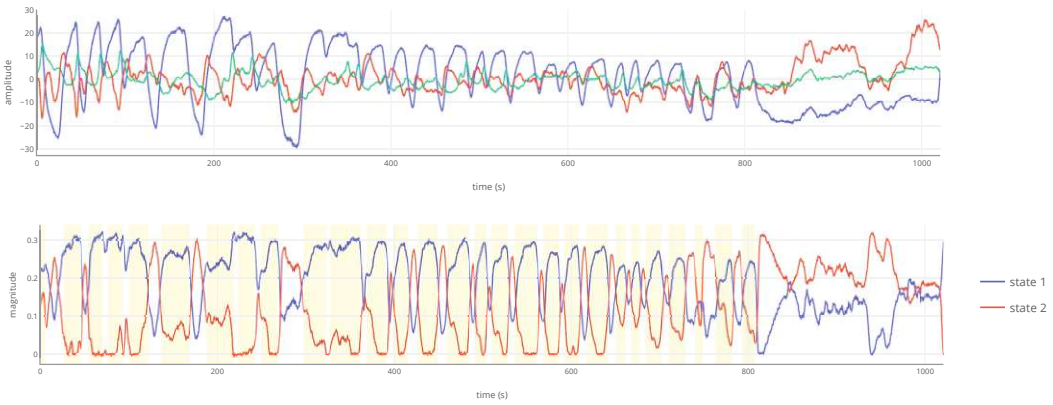


Figure 7.10: **dLDS applied to a single recording projected to the Canonical Neural Manifold.** The yellow highlights indicate where the worm is in a reversal state, as opposed to the forward state, which remains unhighlighted.

also observed in Figure 7.11, which shows the reconstruction and coefficients of all 23 recordings projected to the canonical neural manifold.
However, over time, the inferred coefficients do not consistently align with the two behavior states, which might indicate that either dLDS' performance is not robust for larger datasets or that the two subsystems do not in fact correspond to reversal and forward locomotion but to two unknown latent states. Further exploration could reveal neural correlates of the two subsystems and if they relate to known behaviors.

dLDS fails to consistently capture fast-scale transitions

dLDS applied to other recordings projected to the canonical neural manifold learns coefficients that do not capture all state transitions, as seen in Figure 7.12. However, most of these transitions appear to be of fast-scale, that is, the worm is briefly in a reversal state and then switches back to forward, or vice versa. This is characteristic of

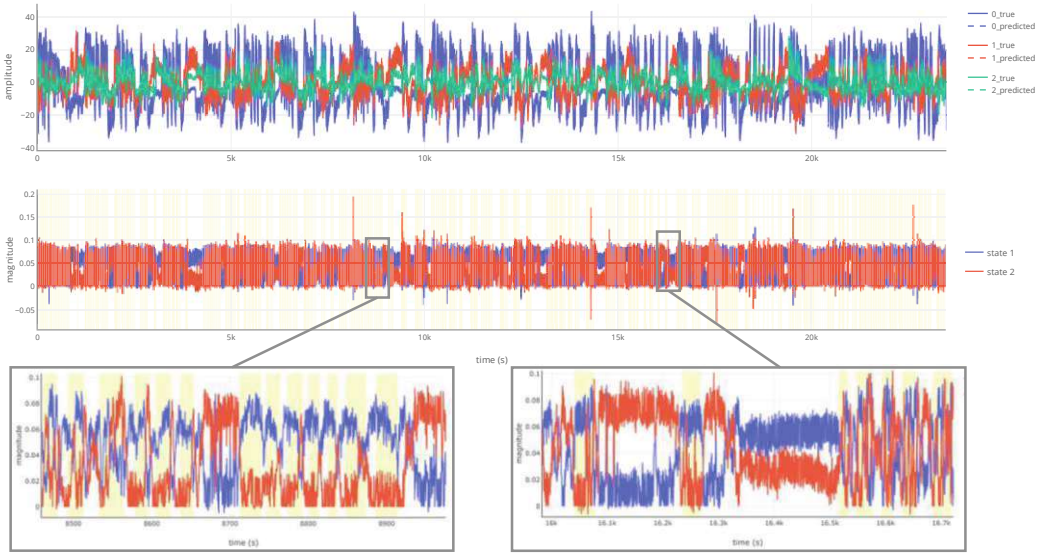


Figure 7.11: **dLDS applied to 23 whole-brain recordings projected to the Canonical Neural Manifold.** dLDS reconstructs data points on the manifold accurately (Top). The inferred coefficients of the two subsystems align with the two major behavior states (yellow highlights) for most but not all state intervals (Bottom).

pirouettes and may suggest that these transient states are actuated by control signals aimed at regulating the entire network.

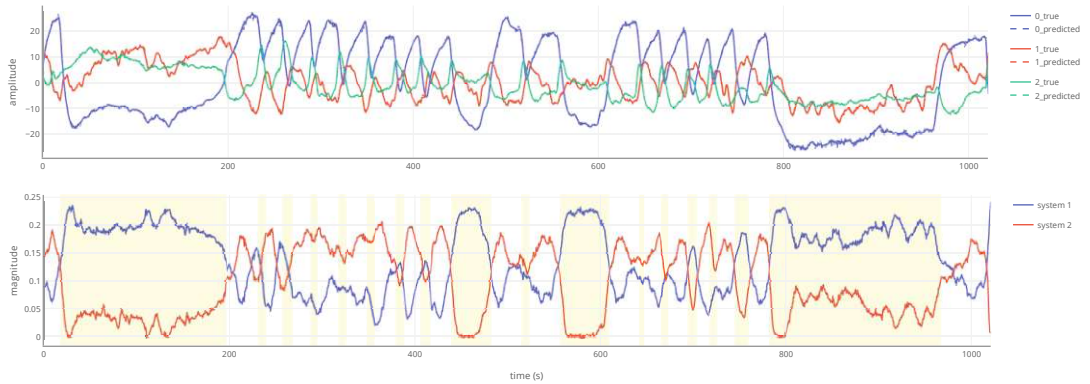


Figure 7.12: **dLDS applied to single recording projected to neural manifold misses fast-scale transitions.**

7.3 Controlled Decomposed Linear Dynamical Systems

cdLDS displays higher sensitivity to hyperparameter selection than dLDS

Applying the cdLDS network to the same toy data as before, we discover that the cdLDS appears to be more sensitive to hyperparameters than the dLDS, especially in terms of coefficient correlation loss. Furthermore, the loss of control correlation shows that the accuracy of the learned control signals varies considerably under conditions where the model is initialized with the true components and where the dynamics F and the dynamics coefficients c are randomly initialized (see Figure 7.13).

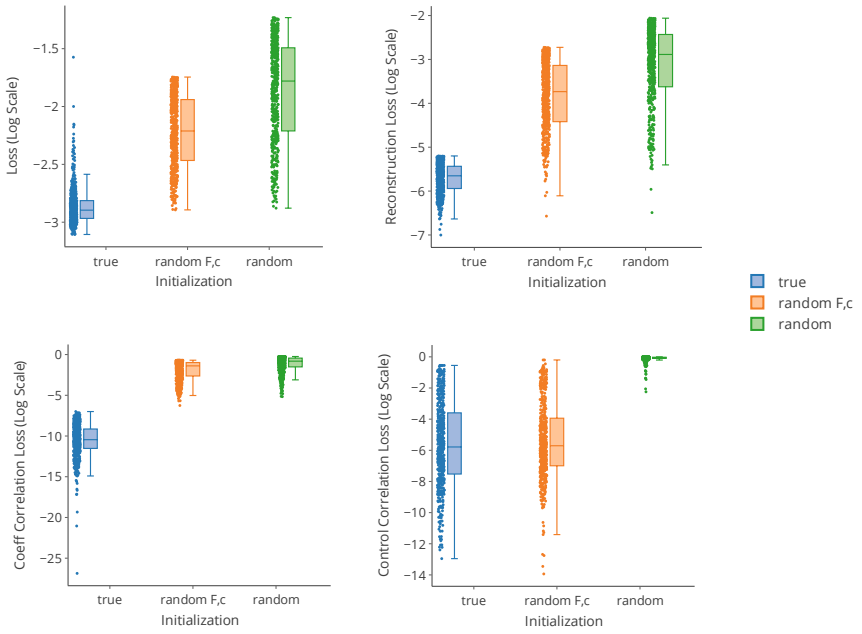


Figure 7.13: **cdLDS loss distribution for each experimental condition.** The log-transformed loss distribution of over 400 experiments illustrate a difference in variability across conditions, indicating different levels of sensitivity to hyperparameters.

However, in the case where the control signals are also arbitrarily initialized, the loss is almost consistently high, with a few exceptions. Figure 7.14 illustrates the importance of each hyperparameter. In contrast to dLDS, the coefficient sparsity term exhibits minimal impact on any of the losses. Similarly, the control sparsity term does not seem to have a large influence. However, these results show the importance of hyperparameter selection and, consequently, that the robustness of cdLDS may require further refinement.

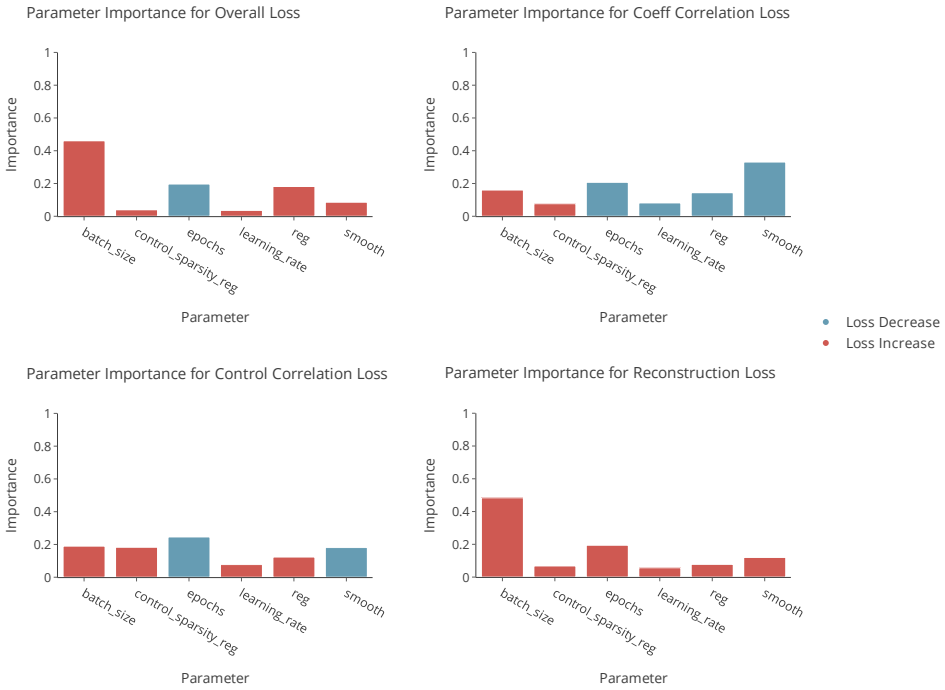


Figure 7.14: **Hyperparameter Importance for Random Initialization computed with Random Forest.** Parameter importance was calculated for the remaining experimental conditions in Appendix A11 and A12.

cdLDS disambiguates control signals from the coefficients

Under an optimized set of hyperparameters, cdLDS reveals that control signals can be disambiguated from dynamics and that they are separate from the coefficients. Specifically, Figures 7.15 and 7.16 show that perturbations in dynamics, such as those seen at time points around 80, caused by control signals, are not captured by the coefficients. However, they are retained in the learned control signals. In particular, in the experimental condition for initializing the dynamics and the dynamics coefficients and in the condition in which the control signals are also initialized randomly, most control signals are learned correctly. This result shows that cdLDS can successfully separate dynamics into slow-scale and fast-scale state transitions.

As observed in Figure 7.16, compared to true control signals, most learned control signals shrink or expand. Specifically, true control signals of smaller magnitude are represented by even smaller learned control signals, with few exceptions, whereas true control signals of larger magnitude lead to the opposite effect. This trend suggests that the network learns to amplify the impact of control signals in both directions. In the condition where all model components are initialized randomly, we observe that control signals are also learned where the system autonomously switches state, which might suggest the need for a non-random initialization strategy for the control signals when applied to real data.

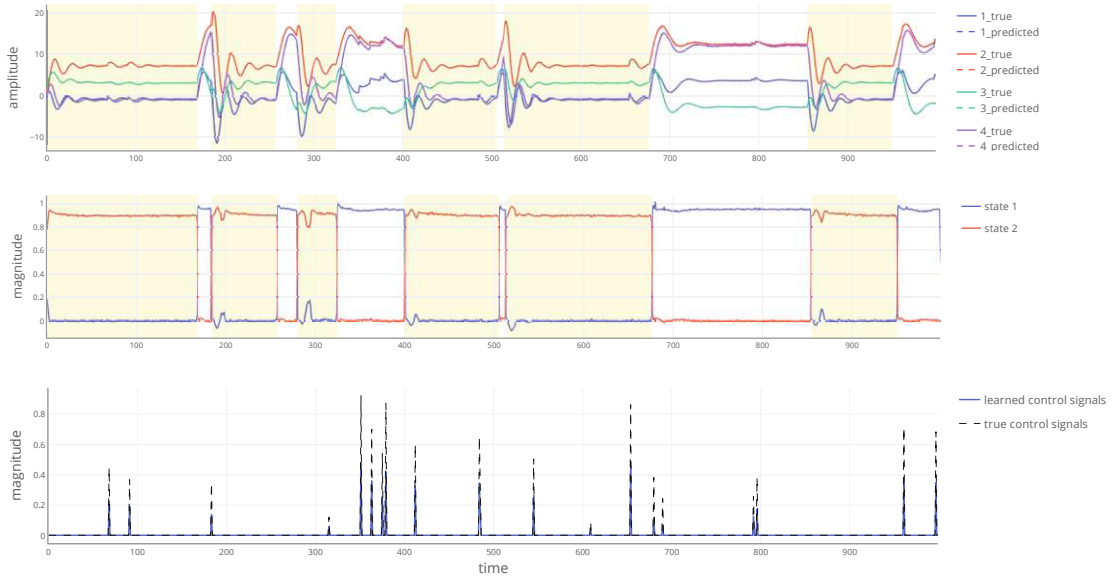


Figure 7.15: **cdLDS with true components applied to toy data retains true coefficients and control signals.**

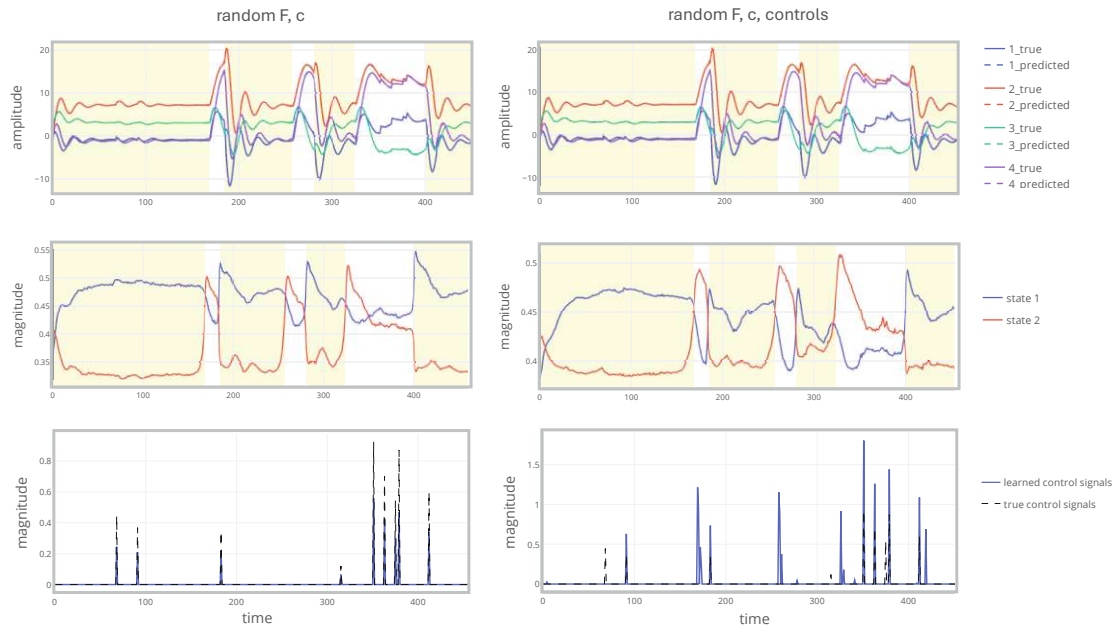


Figure 7.16: **cdLDS applied to toy data yields coefficients and control signals close to their true counterparts.** Random initializations of the dynamics and the dynamics coefficients lead to high reconstruction accuracy of cdLDS, high sparsity of the coefficients and the true control signals are retained (Left). In the case of randomly initializing all model components, the reconstruction is again accurate and the coefficients exhibit a similar pattern to the previous result, yet in addition to the true control signals, new ones emerge (Right).

cdLDS applied to the canonical neural manifold consistently separates coefficients and control signals

When applied to a single recording projected to the canonical neural manifold, cdLDS is able to learn sparse coefficients and control signals. To retrieve this result, the optimal set of hyperparameters is found in the course of more than 300 experiments, evaluated primarily based on the reconstruction loss and the sparsity loss of the coefficients and the control signals. Figure A13 shows the effect of hyperparameters on the reconstruction loss. Furthermore, as with dLDS, we evaluate the performance of cdLDS by assessing the interpretability of the coefficients and the control signals. In general, we observe a trade-off between interpretable coefficients and interpretable control signals. Figure 7.17 shows an experiment that scores well on reconstruction loss, sparsity of coefficients and controls, and the inferred control signals appear to be interpretable. Nevertheless, the coefficients are not consistently interpretable, as we see from time points 600 to 800; the coefficients are neither smooth nor do they exhibit clear state switches. In most experiments, the coefficients appear to be oscillatory or display transient states, where we expected cdLDS to generate control signals. Figure 7.17 shows this around the time points 400 to 600 where we observe multiple short state intervals. The same experiment infers coefficients that exhibit transient states within the first 100 time points, where control signals are learned and expected, considering the pirouette-like pattern observed in the neural manifold traces.

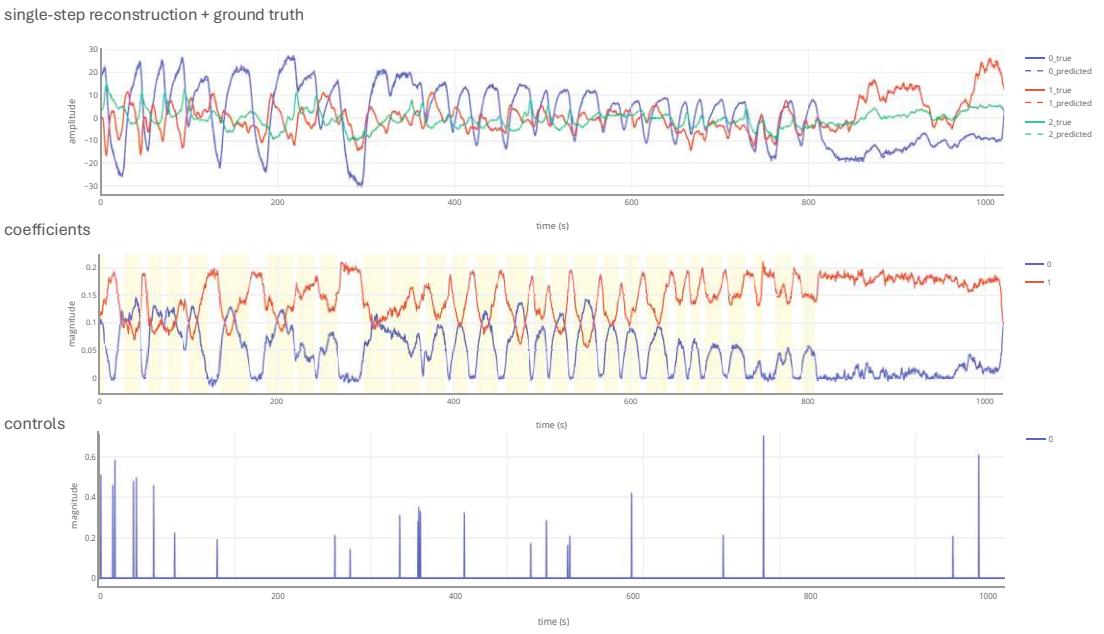


Figure 7.17: **cdLDS applied to a single recording projected to the canonical neural manifold disambiguates control signals and coefficients**

To ensure robustness of the learned control signals, we adopt an initialization strategy

based on the assumption that any fast-scale state switches are actuated by control signals. To this end, we quantify the duration of each forward or reversal state in a recording and define a control signal as any state interval that has a duration shorter than 20% of the overall distribution of state durations. Figure A15 shows an example of an initialization of the control signals.
Under this initialization strategy, cdLDS produces coefficients and control signals closer to the expected patterns. Figure 7.18 shows that the short-scale dynamics we attribute to control signals are displayed in the learned control signals themselves, not in the coefficients. We can also observe that some of the control signals die out, indicating that not all transient states are found to be due to control signals.
These results show that cdLDS effectively separates intrinsic dynamics, reflected by the dynamics coefficients, and actuated dynamics, represented by the control signals. Furthermore, as some initialized control signals diminish over time, it becomes clear that not all fast-scale transitions are, in fact, actuated.

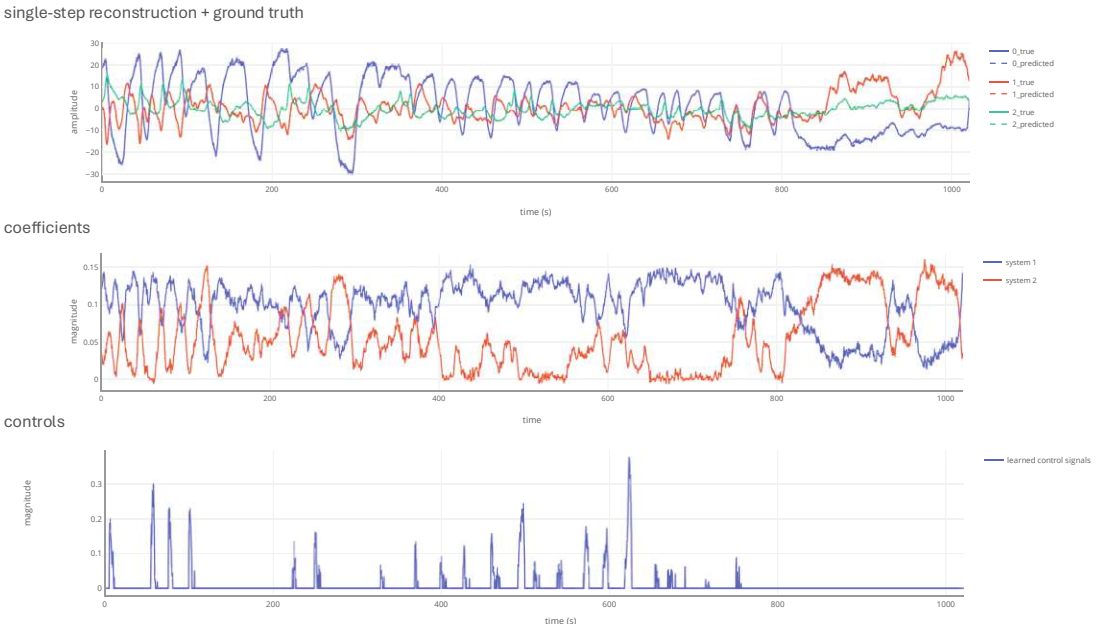


Figure 7.18: **cdLDS with a control initialization strategy applied to a single recording projected to the canonical neural manifold.**

CHAPTER 8

Conclusion

In this thesis, we have addressed a gap in understanding the neural dynamics that govern decision-making processes during locomotion in *C. elegans*. We have presented a canonical neural manifold that, for the first time, represents the neural activity patterns of 23 individuals during locomotion as a low-dimensional state-space. We then studied the underlying neural dynamics as a temporal evolution of states on the manifold, where the evolution is governed by dynamical rules. Prior work has described this temporal evolution as an autonomous process driven by a dynamical system [DS21]. However, prior work has also observed the presence of non-autonomous events within neural activity patterns during *C. elegans* locomotion [MFK21]. To account for these unpredictable events, we proposed an extension to an autonomous dynamical systems model.

To construct a canonical neural manifold, we aligned 23 whole-brain Ca^{2+}-imaging recordings by applying a myriad of preprocessing techniques, ranging from data imputation to quartile normalization. A PCA on the preprocessed space yielded a manifold that is spanned by three latent variables, each reflecting population activity. We show that this manifold exhibits high separability of the behavior states compared to the manifold of the raw recordings. To establish the global universality of the manifold, we examined to what extent observations on the manifold could be linked to their respective recordings. To this end, we formalized a clustering task to evaluate the agreement between clusters, found within the manifold in an unsupervised way, and the original recordings, as well as a classification task to assign manifold points to their corresponding recordings. We show that a dimensionality reduction of the original data decreases the identifiability of observations on the manifold. However, our results also demonstrate that the preprocessing pipeline we implemented further improves the universality of the manifold by minimizing variability due to individuality.

We discovered that the canonical neural manifold reveals a distinct structure comparable to that of the single *C. elegans* neural manifolds found in previous work by Kato et al. [Kat+15] It is characterized by a bifurcation during sustained reversal, indicating a

decision point or an unpredictable transition to the ventral or dorsal turn states. The manifold also displays fast-scale loops in the sustained reversal and the forward state, hinting towards recurring or unstable dynamics, reminiscent of pirouettes.

To interpret the structure from the perspective of dynamical systems, we extended the decomposed Linear Dynamical System (dLDS) introduced by Mudrik et al. in 2023 [Mud+23]. dLDS assumes that the dynamical system that describes the temporal evolution of brain states can be decomposed into several simpler dynamical systems that contribute to neural dynamics at a given time point to different degrees. These degrees, defined as the dynamics coefficients, are learned in an Expectation-Maximization algorithm, offering an interpretable component of a model that can be linked to observable features. To reduce computational cost and increase the modularity of the algorithm, we formalized dLDS as a feed-forward neural network. We demonstrated dLDS' sensitivity to hyperparameters and initialization of model parameters by comparing loss metrics across different experimental conditions. Furthermore, we show the necessity of various metrics, such as coefficient correlation loss, to evaluate the inference of dynamics coefficients. dLDS applied to the canonical neural manifold reveals two latent states that correspond to the two major behaviors: forward and reversal. This result aligns with insights from literature: neural dynamics vary across behavior states. In other words, the way neurons connect and activate changes over time, and this change can be discretized into separate dynamical systems.

We also observed instances where dLDS fails to capture fast-scale transitions between latent states, suggesting the presence of unpredictable perturbations to the system. We presented an extension of dLDS, controlled dLDS (cdLDS), as a proof of concept that control signals can be disambiguated from the intrinsic latent states. Given a control initialization based on short state intervals, cdLDS learns a separation of slow- and fast-scale state transitions.

The learned control signals can be further analyzed to discover neural correlates that modulate autonomous dynamical processes. Specifically, cdLDS could be applied to neural activity data from experimental settings, where the worm is exposed to a stimulus. If we believe the stimulus to affect intrinsic neural dynamics, we could discover neurons that modulate the entire neural network in response to the stimulus by analyzing correlations between neural activity and control signals. A future avenue is also the application of cdLDS with multiple latent states to investigate whether the latent states discovered in prior work are learned as intrinsic latent states or as control signals instead.

Overview of Generative AI Tools Used

In the preparation of the thesis, I utilized ChatGPT and Writefull to translate the abstract, and also to improve the grammar and clarity of the text.

Übersicht verwendeter Hilfsmittel

Bei der Erstellung der Masterarbeit habe ich ChatGPT und Writefull genutzt, um die Grammatik und Klarheit des Textes zu verbessern und für die Übersetzung der Kurzfassung.

List of Figures

List of Tables

Bibliography

[Ahr+12] Misha B. Ahrens et al. "Brain-wide neuronal dynamics during motor adaptation in zebrafish". In: *Nature* 485.7399 (May 9, 2012), p. 471. DOI: 10.1038/nature11057. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC3618960/ (visited on 10/25/2024).

[AJP22] Zoe Ashwood, Aditi Jha, and Jonathan W. Pillow. "Dynamic Inverse Reinforcement Learning for Characterizing Animal Behavior". In: *Advances in Neural Information Processing Systems* 35 (Dec. 6, 2022), pp. 29663–29676. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/bf215fa7fe70a38c5e967e59c44a99d0-Abstract-Conference.html (visited on 03/04/2024).

[Alt+] Z F Altun et al. *WormAtlas*. http://www.wormatlas.org. Accessed: 2024-10-08.

[Bie20] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[BP17] Connor Brennan and Alex Proekt. *Universality of macroscopic neuronal dynamics in Caenorhabditis elegans*. Nov. 22, 2017. arXiv: 1711.08533[q-bio]. URL: http://arxiv.org/abs/1711.08533 (visited on 11/07/2023).

[Bru+16] Bingni W. Brunton et al. "Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition". In: *Journal of Neuroscience Methods* 258 (Jan. 30, 2016), pp. 1–15. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2015.10.010. URL: https://www.sciencedirect.com/science/article/pii/S0165027015003829 (visited on 11/28/2024).

[Bui+13] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[Cay08] Lawrence Cayton. "Algorithms for manifold learning". In: (June 3, 2008). URL: https://escholarship.org/uc/item/8969r8tc (visited on 02/21/2024).

[Cha+19] Rishidev Chaudhuri et al. “The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep”. In: *Nature Neuroscience* 22.9 (Sept. 2019). Number: 9 Publisher: Nature Publishing Group, pp. 1512–1520. ISSN: 1546-1726. DOI: 10.1038/s41593-019-0460-x. URL: https://www.nature.com/articles/s41593-019-0460-x (visited on 02/22/2024).

[Chu+10] Mark M Churchland et al. “Stimulus onset quenches neural variability: a widespread cortical phenomenon”. In: *Nature neuroscience* 13.3 (Mar. 2010), pp. 369–378. ISSN: 1097-6256. DOI: 10.1038/nn.2501. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2828350/ (visited on 04/08/2024).

[CPM19] Adam J Calhoun, Jonathan W Pillow, and Mala Murthy. “Unsupervised identification of the internal states that shape natural behavior”. In: *Nature neuroscience* 22.12 (2019), pp. 2040–2049.

[CS16] Francesco Camastra and Antonino Staiano. “Intrinsic dimension estimation: Advances and open problems”. In: *Information Sciences* 328 (2016), pp. 26–41.

[CY14] John P. Cunningham and Byron M. Yu. “Dimensionality reduction for large-scale neural recordings”. In: *Nature Neuroscience* 17.11 (Nov. 2014), pp. 1500–1509. ISSN: 1546-1726. DOI: 10.1038/nn.3776.

[DKD23] Max Dabagia, Konrad P. Kording, and Eva L. Dyer. “Aligning latent representations of neural activity”. In: *Nature Biomedical Engineering* 7.4 (Apr. 2023). Publisher: Nature Publishing Group, pp. 337–343. ISSN: 2157-846X. DOI: 10.1038/s41551-022-00962-7. URL: https://www.nature.com/articles/s41551-022-00962-7 (visited on 03/04/2024).

[DS21] Lea Duncker and Maneesh Sahani. “Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings”. In: *Current Opinion in Neurobiology*. Computational Neuroscience 70 (Oct. 1, 2021), pp. 163–170. ISSN: 0959-4388. DOI: 10.1016/j.conb.2021.10.014. URL: https://www.sciencedirect.com/science/article/pii/S0959438821001264 (visited on 11/23/2024).

[EH21] R. Becket Ebitz and Benjamin Y. Hayden. “The population doctrine in cognitive neuroscience”. In: *Neuron* 109.19 (Oct. 6, 2021). Publisher: Elsevier, pp. 3055–3068. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2021.07.011. URL: https://www.cell.com/neuron/abstract/S0896-6273(21)00521-3 (visited on 11/27/2024).

[FZK20] Charles Fieseler, Manuel Zimmer, and J. Nathan Kutz. “Unsupervised learning of control signals and their encodings in Caenorhabditis elegans whole-brain recordings”. In: *Journal of The Royal Society Interface* 17.173 (Dec. 9, 2020). Publisher: Royal Society, p. 20200459. DOI: 10.1098/rsif.

2020.0459. URL: https://royalsocietypublishing.org/doi/full/10.1098/rsif.2020.0459 (visited on 11/07/2023).

[Gal+13] Thomas Gallagher et al. "The geometry of locomotive behavioral states in C. elegans". In: *PloS one* 8.3 (2013), e59865.

[Gal+17] Juan A. Gallego et al. "Neural Manifolds for the Control of Movement". In: *Neuron* 94.5 (June 7, 2017), pp. 978–984. ISSN: 1097-4199. DOI: 10.1016/j.neuron.2017.05.025.

[Gal+20] Juan A. Gallego et al. "Long-term stability of cortical population dynamics underlying consistent behavior". In: *Nature Neuroscience* 23.2 (Feb. 2020), pp. 260–270. ISSN: 1097-6256, 1546-1726. DOI: 10.1038/s41593-019-0555-4. URL: https://www.nature.com/articles/s41593-019-0555-4 (visited on 01/15/2024).

[GF23] Michele Ginesi and Paolo Fiorini. "Generalization of Auto-Regressive Hidden Markov Models to Non-Linear Dynamics and Unit Quaternion Observation Space". In: *IEEE Robotics and Automation Letters* (2023).

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[HGW16] Oliver Hobert, Lori Glenwinkel, and John White. "Revisiting neuronal cell type classification in Caenorhabditis elegans". In: *Current Biology* 26.22 (2016), R1197–R1203.

[Jol02] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

[Kat+15] Saul Kato et al. "Global Brain Dynamics Embed the Motor Command Sequence of Caenorhabditis elegans". In: *Cell* 163.3 (Oct. 22, 2015). Publisher: Elsevier, pp. 656–669. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2015.09.034. URL: https://www.cell.com/cell/abstract/S0092-8674(15)01196-4 (visited on 11/07/2023).

[Kau+14] Matthew T Kaufman et al. "Cortical activity in the null space: permitting preparation without movement". In: *Nature neuroscience* 17.3 (2014), pp. 440–448.

[Kre21] Rebecca A Kresnik. "„The Impact of Perturbations in Neurotransmitter Pathways on Global Brain Dynamics in Caenorhabditis elegans". MA thesis. University of Vienna, 2021.

[Kum+23] Akshey Kumar et al. *BunDLe-Net: Neuronal Manifold Learning Meets Behaviour*. preprint. Neuroscience, Aug. 12, 2023. DOI: 10.1101/2023.08.08.551978. URL: http://biorxiv.org/lookup/doi/10.1101/2023.08.08.551978 (visited on 11/07/2023).

[LeC+02] Yann LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[Lem+15] William C. Lemon et al. “Whole-central nervous system functional imaging in larval Drosophila”. In: *Nature Communications* 6.1 (Aug. 11, 2015). Publisher: Nature Publishing Group, p. 7924. ISSN: 2041-1723. DOI: 10.1038/ncomms8924. URL: https://www.nature.com/articles/ncomms8924 (visited on 10/25/2024).

[Li19] Canchen Li. *Preprocessing Methods and Pipelines of Data Mining: An Overview*. June 20, 2019. DOI: 10.48550/arXiv.1906.08510. arXiv: 1906.08510. URL: http://arxiv.org/abs/1906.08510 (visited on 11/24/2024).

[Lin+16] Scott W. Linderman et al. *Recurrent switching linear dynamical systems*. Oct. 26, 2016. DOI: 10.48550/arXiv.1610.08466. arXiv: 1610.08466[stat]. URL: http://arxiv.org/abs/1610.08466 (visited on 06/04/2024).

[Lin+19] Scott Linderman et al. *Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in C. elegans*. preprint. Neuroscience, Apr. 29, 2019. DOI: 10.1101/621540. URL: http://biorxiv.org/lookup/doi/10.1101/621540 (visited on 11/07/2023).

[MC20] Manu S Madhav and Noah J Cowan. “The synergy between neuroscience and control theory: the nervous system as inspiration for hard control challenges”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1 (2020), pp. 243–267.

[MFK21] Megan Morrison, Charles Fieseler, and J. Nathan Kutz. “Nonlinear Control in the Nematode C. elegans”. In: *Frontiers in Computational Neuroscience* 14 (2021). ISSN: 1662-5188. URL: https://www.frontiersin.org/articles/10.3389/fncom.2020.616639 (visited on 11/07/2023).

[Mit+23] Rufus Mitchell-Heggs et al. “Neural manifold analysis of brain circuit dynamics in health and disease”. In: *Journal of Computational Neuroscience* 51.1 (Feb. 1, 2023), pp. 1–21. ISSN: 1573-6873. DOI: 10.1007/s10827-022-00839-3. URL: https://doi.org/10.1007/s10827-022-00839-3 (visited on 02/29/2024).

[Mud+23] Noga Mudrik et al. *Decomposed Linear Dynamical Systems (dLDS) for learning the latent components of neural dynamics*. June 16, 2023. DOI: 10.48550/arXiv.2206.02972. arXiv: 2206.02972[cs,eess,q-bio,stat]. URL: http://arxiv.org/abs/2206.02972 (visited on 06/12/2024).

[Ngu+16] Jeffrey P Nguyen et al. “Whole-brain calcium imaging with cellular resolution in freely behaving Caenorhabditis elegans”. In: *Proceedings of the National Academy of Sciences* 113.8 (2016), E1074–E1081.

[Pan+18] Chethan Pandarinath et al. “Inferring single-trial neural population dynamics using sequential auto-encoders”. In: *Nature Methods* 15.10 (Oct. 2018). Publisher: Nature Publishing Group, pp. 805–815. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0109-9. URL: https://www.nature.com/articles/s41592-018-0109-9 (visited on 03/08/2024).

[Pas+19] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[PBK16] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. “Dynamic mode decomposition with control”. In: *SIAM Journal on Applied Dynamical Systems* 15.1 (2016), pp. 142–161.

[Per+23] Sean M. Perkins et al. “Simple decoding of behavior from a complicated neural manifold”. In: *eLife* 12 (Oct. 9, 2023). Publisher: eLife Sciences Publications Limited. DOI: 10.7554/eLife.89421.1. URL: https://elifesciences.org/reviewed-preprints/89421 (visited on 11/24/2024).

[Pig+11] Beverly J Piggott et al. “The neural circuits and synaptic mechanisms underlying motor initiation in C. elegans”. In: *Cell* 147.4 (2011), pp. 922–933.

[San+09] Gopal Santhanam et al. “Factor-analysis methods for higher-performance neural prostheses”. In: *Journal of neurophysiology* 102.2 (2009), pp. 1315–1330.

[SC19] Shreya Saxena and John P Cunningham. “Towards the neural population doctrine”. In: *Current Opinion in Neurobiology*. Machine Learning, Big Data, and Neuroscience 55 (Apr. 1, 2019), pp. 103–111. ISSN: 0959-4388. DOI: 10.1016/j.conb.2019.02.002. URL: https://www.sciencedirect.com/science/article/pii/S0959438818300990 (visited on 10/27/2024).

[SLM23] Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. “Learnable latent embeddings for joint behavioural and neural analysis”. In: *Nature* 617.7960 (May 2023). Number: 7960 Publisher: Nature Publishing Group, pp. 360–368. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06031-6. URL: https://www.nature.com/articles/s41586-023-06031-6 (visited on 10/11/2023).

[SLS21] Jimmy Smith, Scott W. Linderman, and David Sussillo. “Reverse engineering recurrent neural networks with Jacobian switching linear dynamical systems”. In: Neural Information Processing Systems. Nov. 1, 2021. URL: https://www.semanticscholar.org/paper/Reverse-engineering-recurrent-neural-networks-with-Smith-Linderman/33297fa35339b45221dc40bb (visited on 06/05/2024).

[Soh+18] Zu Soh et al. “A computational model of internal representations of chemical gradients in environments for chemotaxis of Caenorhabditis elegans”. In: *Scientific Reports* 8.1 (2018), p. 17190.

[Son13] Eduardo D Sontag. *Mathematical control theory: deterministic finite dimensional systems*. Vol. 6. Springer Science & Business Media, 2013.

[Sun06] Zhendong Sun. *Switched linear systems: control and design*. Springer Science & Business Media, 2006.

[Sus+16] David Sussillo et al. *LFADS - Latent Factor Analysis via Dynamical Systems*. Aug. 22, 2016. arXiv: 1608.06315[cs,q-bio,stat]. URL: http://arxiv.org/abs/1608.06315 (visited on 12/28/2023).

[Tak06] Floris Takens. “Detecting strange attractors in turbulence”. In: *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*. Springer. 2006, pp. 366–381.

[Tau+21] Guillaume Tauzin et al. “giotto-tda:: A topological data analysis toolkit for machine learning and data exploration”. In: *Journal of Machine Learning Research* 22.39 (2021), pp. 1–6.

[UKZ22] Kerem Uzel, Saul Kato, and Manuel Zimmer. “A set of hub neurons and non-local connectivity features support global brain dynamics in C. elegans”. In: *Current Biology* 32.16 (2022), pp. 3443–3459.

[Var+11] Lav R Varshney et al. “Structural properties of the Caenorhabditis elegans neuronal network”. In: *PLoS computational biology* 7.2 (2011), e1001066.

[Vid+18] Diego Vidaurre et al. “Discovering dynamic brain networks from big data in rest and task”. In: *NeuroImage* 180 (2018), pp. 646–656.

[Vya+20] Saurabh Vyas et al. “Computation Through Neural Population Dynamics”. In: *Annual review of neuroscience* 43 (July 8, 2020), pp. 249–275. ISSN: 0147-006X. DOI: 10.1146/annurev-neuro-092619-094115. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7402639/ (visited on 02/08/2024).

[Wil+15] Alexander B. Wiltschko et al. “Mapping Sub-Second Structure in Mouse Behavior”. In: *Neuron* 88.6 (Dec. 16, 2015), pp. 1121–1135. ISSN: 1097-4199. DOI: 10.1016/j.neuron.2015.11.031.

[YMC24] Eva Yezerets, Noga Mudrik, and Adam S. Charles. *Decomposed Linear Dynamical Systems (dLDS) models reveal context-dependent dynamic connectivity in C. elegans*. Pages: 2024.05.31.596903 Section: New Results. June 1, 2024. DOI: 10.1101/2024.05.31.596903. URL: https://www.biorxiv.org/content/10.1101/2024.05.31.596903v1 (visited on 07/12/2024).

[Zar+19] Aref Arzan Zarin et al. *A Drosophila larval premotor/motor neuron connectome generating two behaviors via distinct spatio-temporal muscle activity.* Pages: 617977 Section: New Results. Apr. 24, 2019. DOI: 10.1101/617977. URL: https://www.biorxiv.org/content/10.1101/617977v1 (visited on 10/25/2024).

[ZDS] K Zhang, J Dearing, and J Sadler. *Transient dynamics.* http://www.complexity.soton.ac.uk/theory/_Transient_Dynamics.php. Accessed: 2024-10-21.

Appendix

Preliminaries

Whole Brain Ca2+-imaging Data

To gain an understanding of the nature of the individual datasets, what problems we might face, and what decisions have to be made during preprocessing, we pose the following questions.

- Are all neurons in each dataset identified (IDed)?
- How many neurons are IDed in each dataset?
- How many times is each neuron IDed in total? (If too few, remove or impute)

To answer these questions, we quantify the identifications of recorded neurons in Figures A1 and A2.

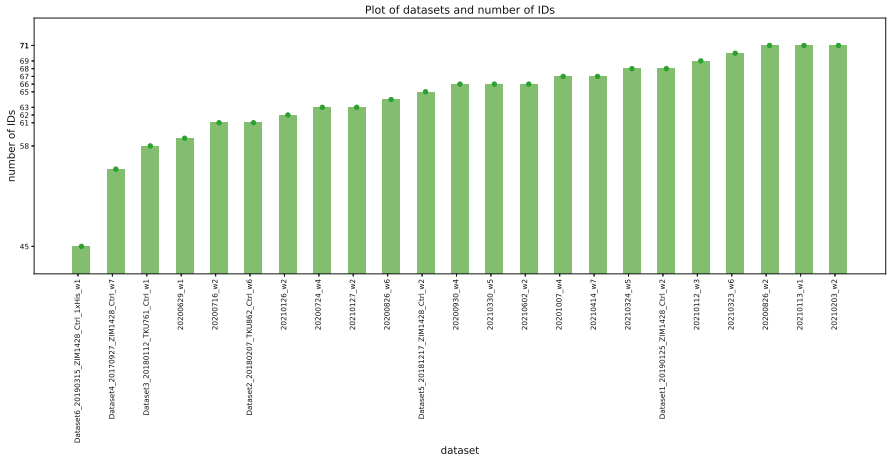


Figure A1: **Number of neurons that have been identified in each dataset.**

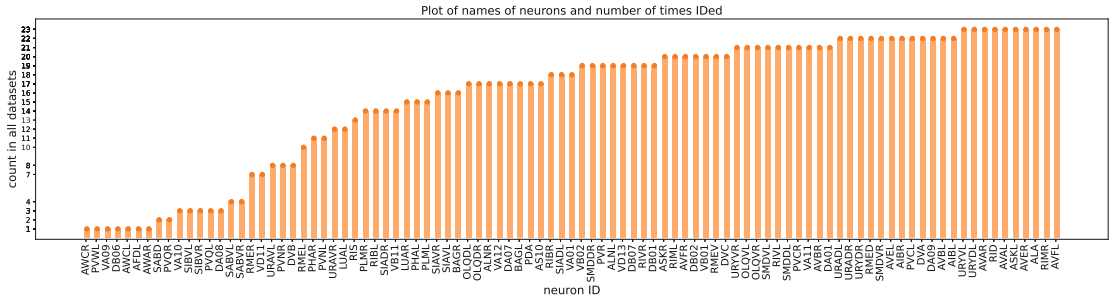


Figure A2: **Number of datasets in which each neuron is identified.** Given that quite a few neurons are IDed less than 10 times, it is necessary to decide whether these neurons should be kept and their missing activities imputed.

Manifold Alignment

Methods for Data Preprocessing

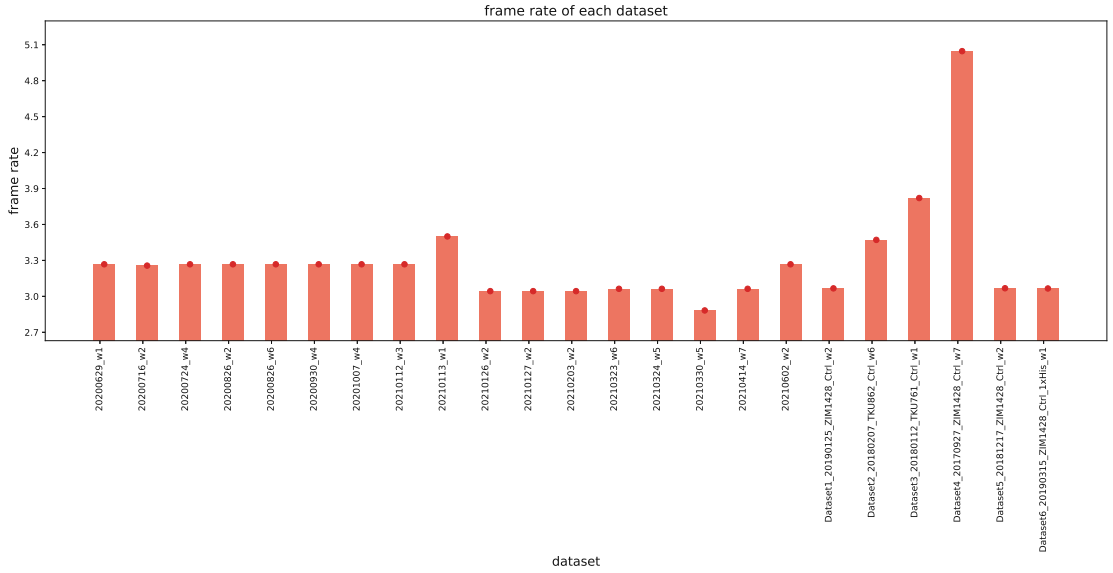


Figure A3: **The frame rate varies across recordings.** As the recording duration is identical in all datasets, yet the frame rate differs, we need to resample all datasets to obtain a consistent number of frames.

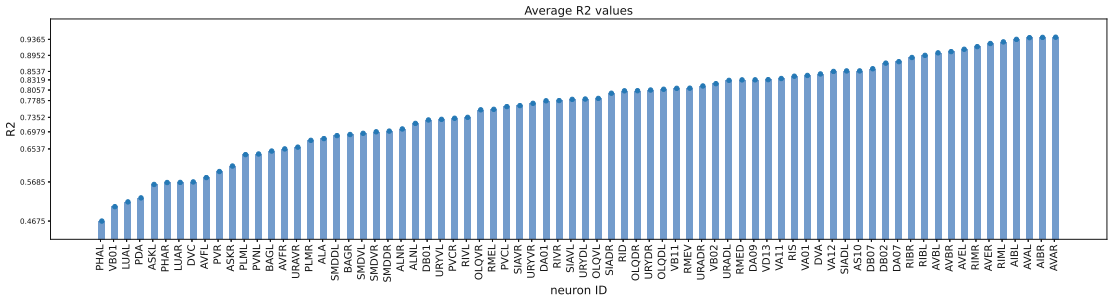


Figure A4: **PLSR determines a lack of *uniqueness* in most neurons.** PLSR models fitted to model individual neurons yield high R^2 scores, showing that, in most cases, the neural activities are linearly predictable.

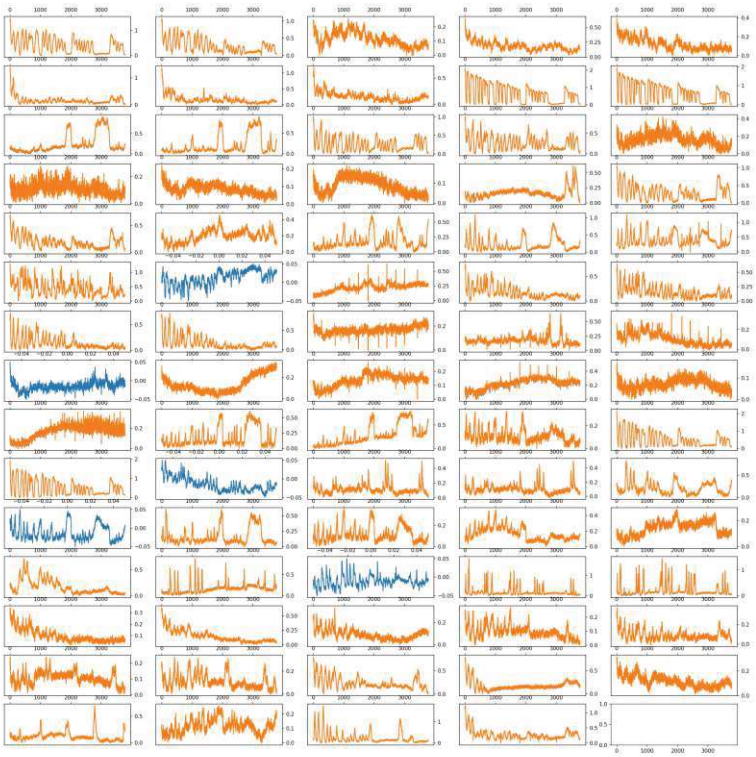


Figure A5: **A sample recording showing traces of 69 identified neurons and 5 imputed neurons.**

Evaluation of Canonical Neural Manifold

Precision, Recall and F1

For evaluating the performance of a classification model, we calculate precision, recall, and the f1-score as:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{8.1}$$

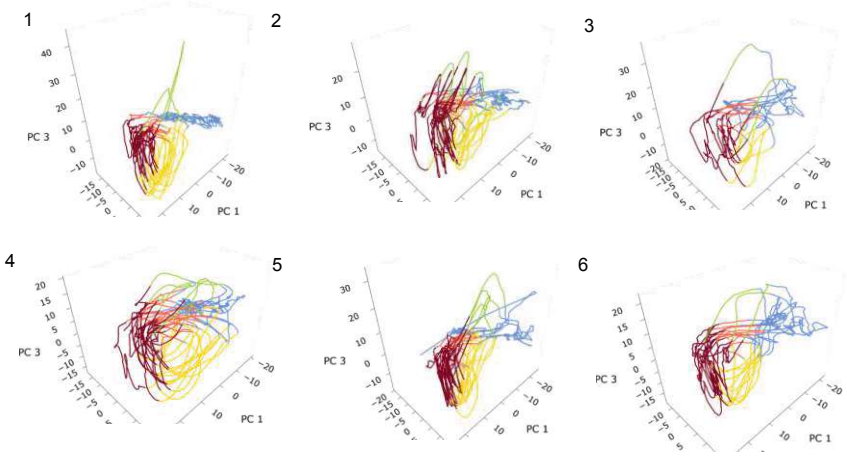
$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{8.2}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8.3}$$

Results

Canonical Neural Manifold

A PCA-based low-dimensional representation of 23 whole-brain recordings shows a manifold-like structure



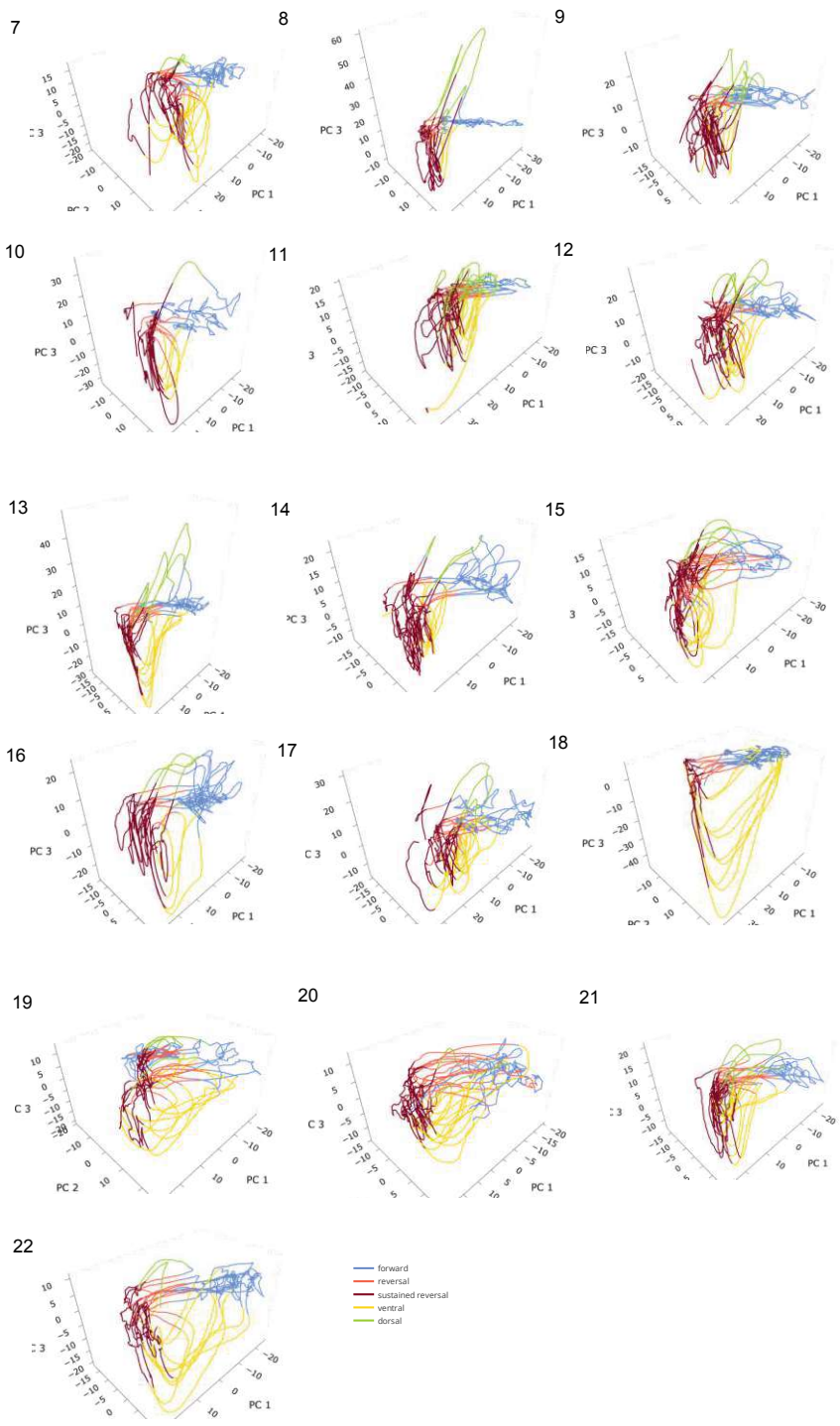


Figure A6: **22 Individual datasets projected onto the shared space exhibit a similar structure.**

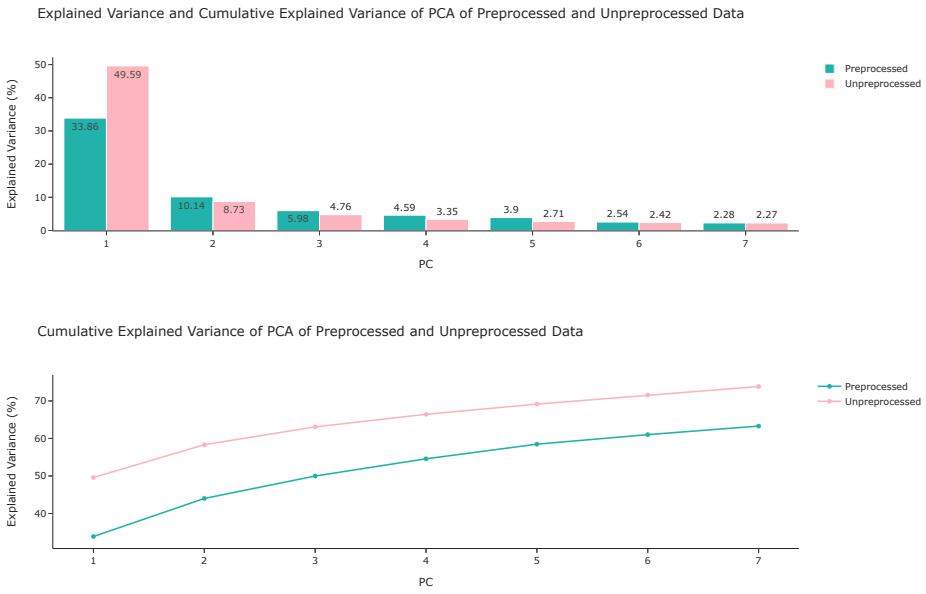


Figure A7: **After preprocessing, the explained variance of the first three PCs goes down.**

Dimensionality Reduction results in decreased variability due to individuality, and preprocessing amplifies this effect

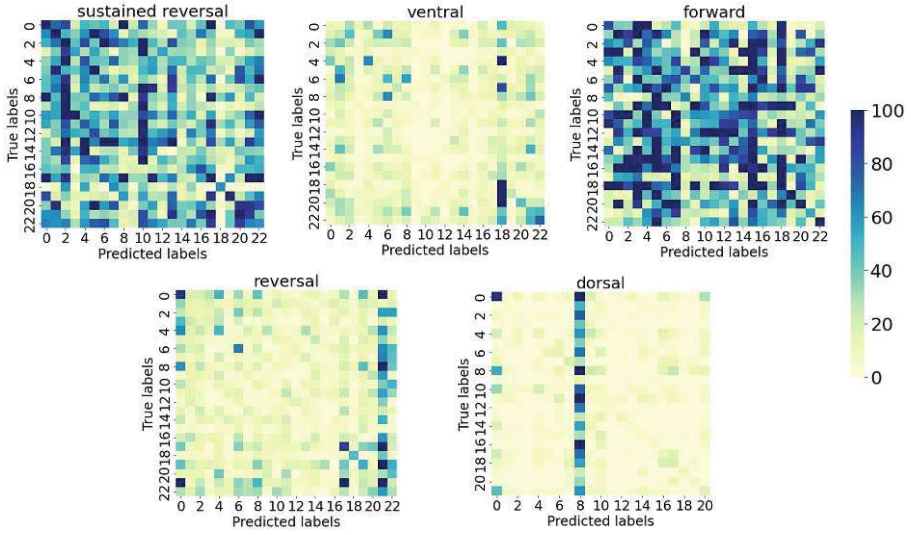


Figure A8: **Confusion matrices depicting the within-state performance of dataset membership classification.**

Canonical Neural Manifold retains a separation of the phase space into behavior states

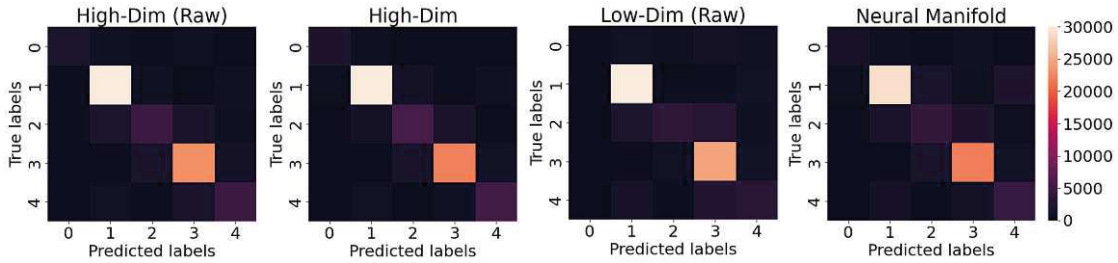


Figure A9: **Confusion matrices depicting the performance of behavior state classification within each data.**

Decomposed Linear Dynamical Systems

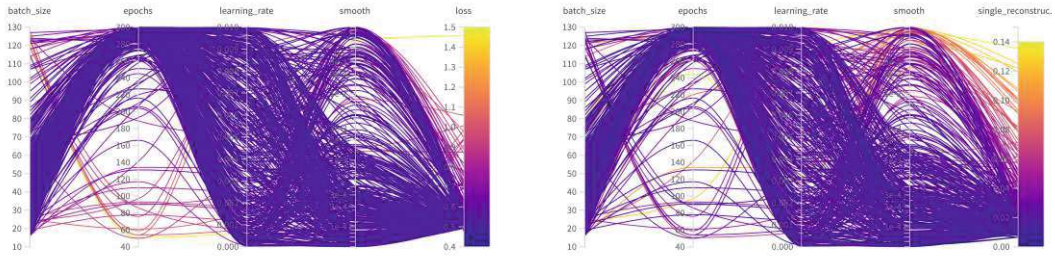


Figure A10: **Impact of hyperparameters values on loss and reconstruction loss.**

Controlled Decomposed Linear Dynamical Systems

cdLDS displays higher sensitivity to hyperparameter selection than dLDS

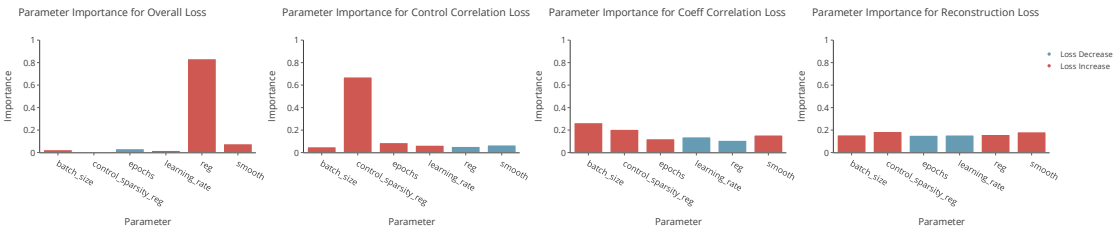


Figure A11: **Parameter importance of models initialized with true components.**

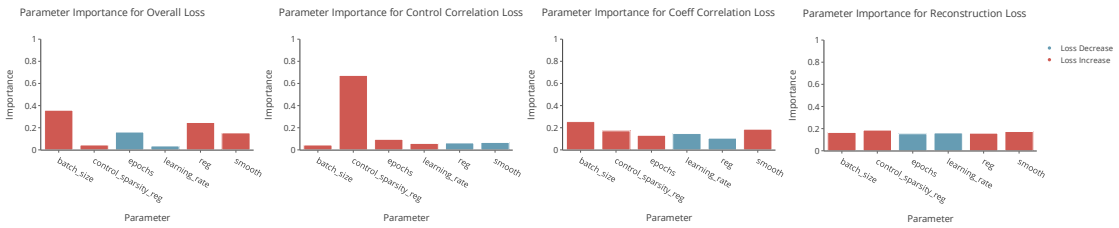


Figure A12: **Parameter importance of models initialized with random dynamics and random dynamics coefficients.**

cdLDS applied to the canonical neural manifold consistently separates coefficients and control signals

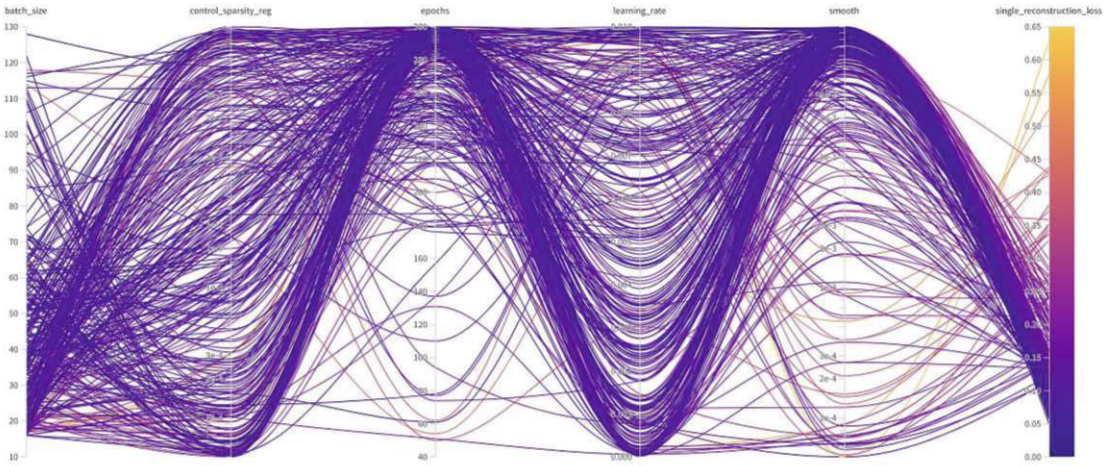


Figure A13: **A hyperparameter sweep of cdLDS applied to the canonical neural manifold shows optimal hyperparameter values for a low reconstruction loss.**

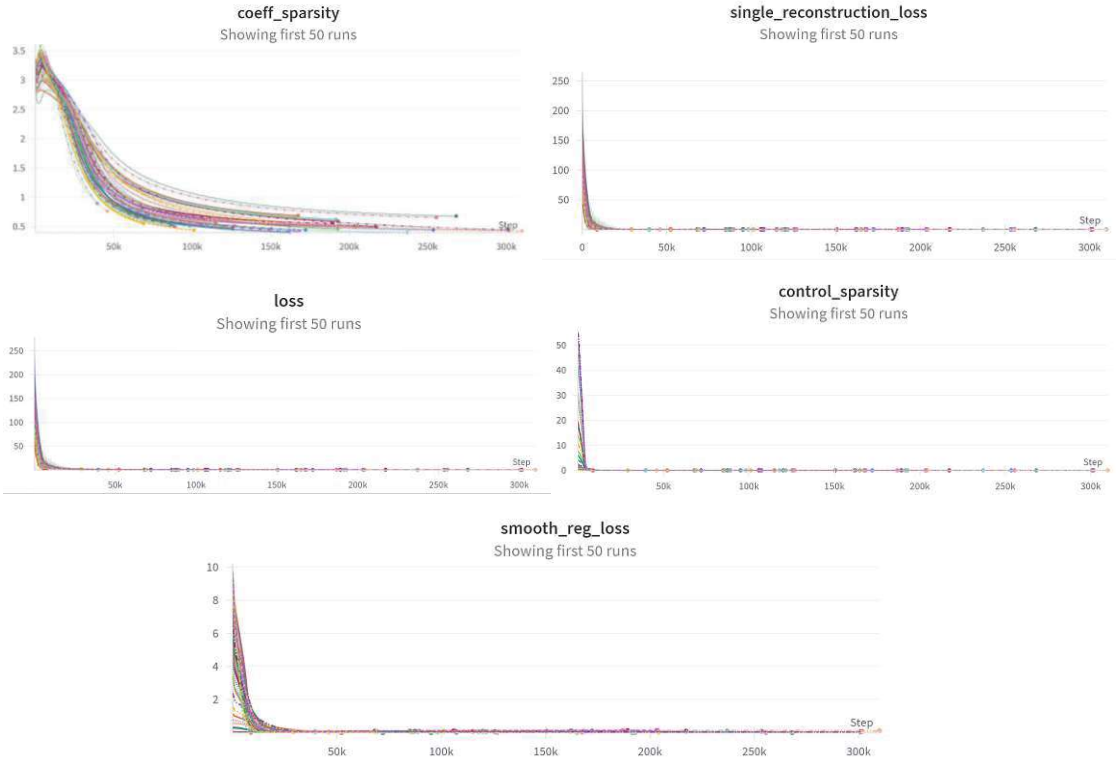


Figure A14: **Evolution of each loss over epochs.** Most loss functions of cdLDS applied to the canonical neural manifold converge fast, with the exception of the coefficients sparsity.

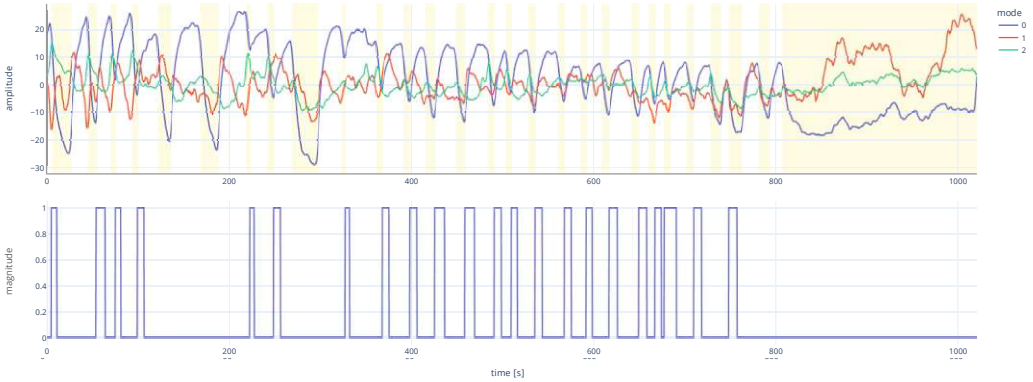


Figure A15: **Control signals initialized with short state intervals.** State intervals, either reversal or forward state intervals that belong to the 20-th bottom percentile in terms of duration are defined as control signals.