Deep Learning Approaches for Modeling and Inferring Neuronal Dynamics

Presented in Partial Fulfillment of
the Requirements for the Degree of

Master of Science

with a Major in

Chemical Engineering

in the

College of Graduate Studies

University of Idaho

by

Benjamin B. Plaster

Major Professor
Gautam Kumar, Ph.D.

Committee
D. Eric Aston, Ph.D.
James Moberly, Ph.D.
Fuchang Gao, Ph.D.

Department Administrator
Ching-An Peng, Ph.D.

December 2020

Authorization to Submit Dissertation

This thesis of Benjamin B. Plaster, submitted for the degree of Master of Science with a Major in Chemical Engineering and titled "Deep Learning Approaches for Modeling and Inferring Neuronal Dynamics," has been reviewed in final form. Permission, as indicated by the signatures and dates below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor:	Gautam Kumar, Ph.D.	Date
Committee Members:	D. Eric Aston, Ph.D.	Date
	James Moberly, Ph.D.	Date
	Fuchang Gao, Ph.D.	Date
Department Administrator:	Ching-An Peng, Ph.D.	Date

Abstract

Recent developments in GPU-accelerated computing, as well as the advent of artificial intelligence and the rise of deep artificial neural networks (ANNs), have created a wealth of opportunity to explore purely data-driven computational modeling techniques on datasets and dynamical systems previously deemed overly complex or outright infeasible to model. One such example of a domain in which these questions are now being brought to light is in the field of computational neuroscience, where both single neuron and neuronal network dynamical responses prove difficult and arduous to model in many cases.

This thesis work is dedicated to the development of computationally feasible, purely data-driven machine learning methods for inferring, learning, and modeling both single-neuron and many-neuron dynamics at multiple time-scales through novel employment of ANNs.

The first portion of this dissertation focuses on the development of purely data-driven recurrent neural network (RNN) models of hippocampal CA1 pyramidal neuron dynamics in response to constant-amplitude applied external input. These CA1 pyramidal neurons exhibit highly nonlinear dynamics, with multiple bifurcations in behavioral response dependent on the magnitude of the externally applied input to the system. This approach involves the use of deep LSTM networks in conjunction with a novel translation trick borrowed from natural language processing (NLP) applications in language translation problems. We demonstrate that the network is capable of learning a complete representation of the dynamics, including multiple bifurcations in behavior. Additionally, we demonstrate that predictive accuracy of the devised LSTM network increases as the length of timeseries on which it is trained does as well.

The second portion of this work focuses on the development of a generative machine learning method to infer firing rate dynamics of a neuronal population in an unsupervised autoencoding framework. This is centered around the idea that stochastic neuronal populations are better described using powerful rate-based dynamical models under the assumption that their firing activity can be described as a many-body nonhomogeneous Poisson point process. We use a sequential adaptation of a popular generative machine learning algorithm, the Variational Autoencoder (VAE) to infer firing rates that maximize the likelihood of the original data in an unsupervised manner, and demonstrate that this architecture is capable of discovering coherent dynamical representations of smoothed firing rates directly from binned spiking data.

Acknowledgements

I would like to express my sincerest thanks to my advisor, Dr. Gautam Kumar, for his constant support, guidance, patience, and help throughout the course of my graduate studies. As an instructor, he introduced me to my favorite academic subject and was a great mentor. As an advisor, he has supported me through some of the lowest and most trying periods of my life. I greatly appreciate the independence with which he allowed me to work on our research problems in our time together - this has undoubtedly left me a far better researcher and engineer. For that, I will be forever grateful, it has been an honor to work with him.

I must also thank my committee members, Dr. Eric Aston, Dr. Frank Gao, and Dr. James Moberly for their support in my various endeavors over my time at the university, as well as for their help in feedback and furthered development of this thesis work. They have all been instrumental in my development.

I would also like to gratefully acknowledge the University of Idaho College of Engineering and the Department of Chemical and Materials Engineering for their support over these last two and a half years.

Dedication

This thesis is dedicated to my parents, John and Leslie, who have been my main line of support and encouragement throughout my entire life. I would not be half the person I am today if not for them. I also would like to thank my sister, Mallory, and my brother, Nicholas, for their love and friendship. They are my best friends.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Advents in parallel processing due to recent advances in graphics processing unit (GPU) acceleration of large linear algebraic systems has led to renewed interest in the field of artificial neural networks (ANNs) and machine learning over the course of the last decade (2). ANNs have been employed in various novel manners in a variety of fields, producing incredible advances in the fields of machine vision (3), speech recognition (4) natural language processing (5; 6) and many others. Their purely data-driven nature combined with their ability to approximate nearly any function (7) have made them especially useful in modeling applications on datasets previously thought too complicated to handle.

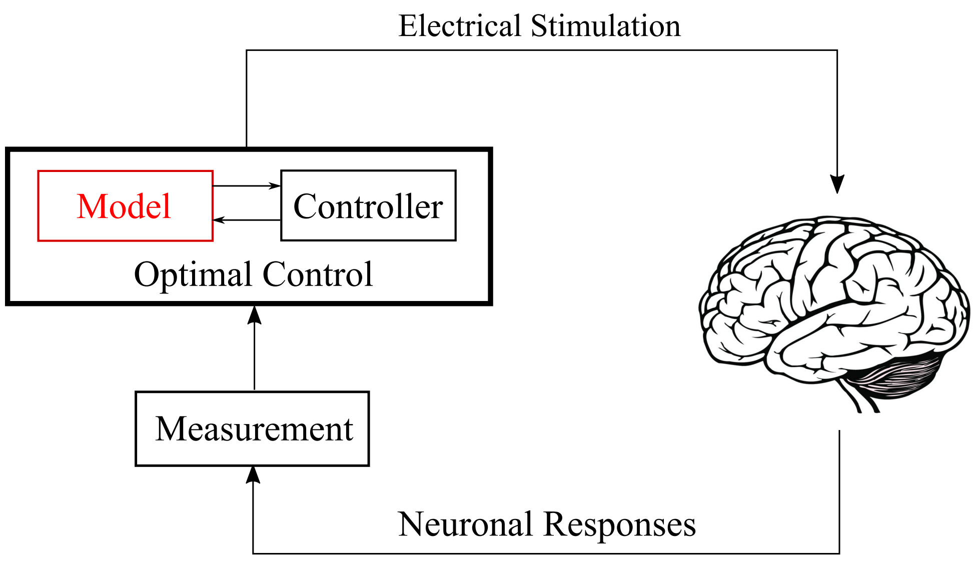


Figure 1.1: Illustration of a closed-loop feedback optimal control neurostimulation strategy.

This ability to handle and learn otherwise intractably complex relationships contained within high-dimensional datasets has also opened the door to a data revolution within the practice of computational neuroscience. Data sets and relationships previously thought infeasible to model and uncover have become available for investigation. Ultimately, this may allow for the development of data-driven, computationally feasible models of neuronal behaviors in response to externally applied inputs - this would be of tremendous use to modelers, biologists, mathematicians, and engineers alike as control-theoretic neurostimulation strategies could be systematically investigated and explored.

The focus of this thesis work is to develop purely data-drive, advanced, novel ANN-centered approaches to modeling neuronal behaviors at both the single-neuron level, as well as within the context of high-dimensional neuronal circuits.

1.1 ARTIFICIAL NEURAL NETWORKS

Broadly speaking, artificial neural networks (ANNs) are a class of parametric computing systems. At the most primitive level, they are comprised of artificial "neurons" arranged in a layer-wise orientation with weighted connections between neurons in neighboring layers. They accept input from neurons from the layer previous, perform a nonlinear transformation the weighted sum, and output a single value, which is passed on to other neurons in the subsequent layer. The inputs to the network are numerical values representing samples from the data. The network's output is the product of the nonlinear transformation of the final layer. The nature of the output from the network depends on the objective function used to train the network, this is known as the loss function. These details will be further laid out in sections 1.1.2, 1.1.3, and 1.1.4.

1.1.1 *Historical Background*

Unsurprisingly, artificial neural networks derive their original inspirations from the structure of biological neural networks. Donald Hebb first proposed that biological neural pathways strengthen when they are used repetitively and successively, especially when firing together. This led to the idea of 'Hebbian Learning', a model of neural plasticity and learning. He famously said in his book, *The Organization of Behavior*, that "Cells that fire together, wire together" (8). This observation led to interest in computing systems that sought to emulate this behavior.

The first artificial neural network was developed by Frank Rosenblatt in 1960, and is known as the Mark I Perceptron (9), as an attempt to understand and model the

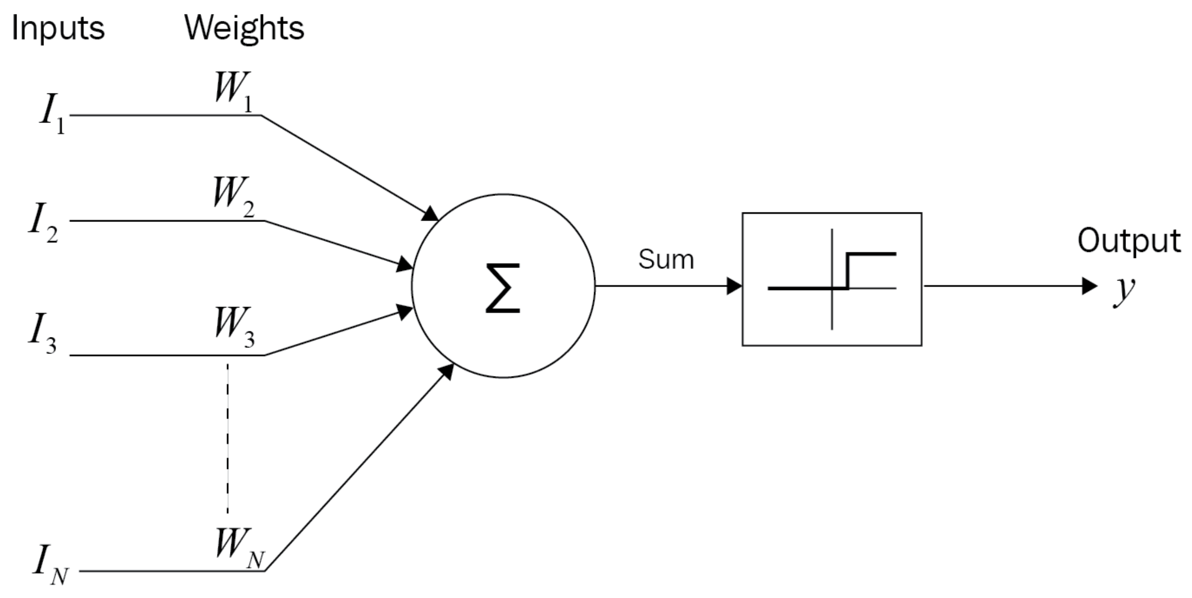


FIGURE 1.2: A McCulloch-Pitts neuron, on which the Perceptron was modeled. Figure sourced from O'Reilly - 'The McCulloch-Pitts Neuron'.

decision systems of the eye of a simple fly. This was modeled as a McCulloh-Pitts neuron, which can be visualized in Figure 1.2. It is comprised of a set of m inputs, **I**, with m weighted connections, **W**, and a simple summation with an linear output threshold, and some output quantity, y. Thus, the output from the neuron can be fully described as

$$y = \phi\left(\sum_{j=0}^{m} W_j I_j\right) \tag{1.1}$$

where ϕ represents a linear gating threshold, returning 0 if the weighted sum is greater than or equal to zero, and 1 otherwise. The key innovation was their iterative learning procedure, in which the weights were updated with each pass of new input information into the neuron.

The perceptron models of fly decision systems gave rise to a new interest in artificial neural networks, and their potential for applications in data and signal processing fields. The first application of these neural network models to real-world engineering problems came from Stanford researcher Brian Widrow and his graduate student, Marcian Hoff, in the chemical memistor solutions known as ADALINE (ADAptive

LINear Elements) and MADALINE (Multiple ADAptive LINear Elements), which were used to filter noisy signals along phone lines (10; 11).

However, from these primitive single-layered unsupervised neural networks came a long period of nearly zero activity on the front of research into the furthered development of these neural networks. This arose for several reasons; firstly, computational resources in the 1960's pale in comparison to modern-era digital computers. Secondly, the combination of backpropagation with stochastic gradient descent had not yet been pioneered for training multi-layer networks, limiting both the efficacy and feasibility of training the networks to learn anything meaningful.

1985 became the year that artificial neural networks came back to the forefront of parametric computing, with the introduction of iterative backpropagation through layers introduced by Rumelhart, Hinton, and Williams (12). Their innovation of teaching networks through iterative backpropagation based learning opened the door for neural networks to learn highly complex internal representations, and has resulted in the deep, complicated, and hugely parametrized ANN models of the modern era.

1.1.2 *Feedforward Networks*

Feedforward networks, often referred to as densely connected networks, vanilla ANNs, or multi-layer perceptrons (MLPs) form the most basic of the archetypal modern ANNs. In these networks, information flows unilaterally in one direction, from input neurons to output neurons, traversing all hidden (intermediate) layers in between.

Similarly to the earliest iterations of the perceptron, all feedforward networks can be viewed as multi-layered constructions of various types of McCulloch-Pitts neurons. However, there are some key differences between modern use cases and the original Mark I Perceptron. The perceptron made use of a discontinuous threshold activation, and thus outputs binary signals. Modern ANN neurons make use of a continuous transformation function, also known as an activation function, which affords them compatibility with backpropagation-based training methods. Common activation functions in modern ANNs include the sigmoid activation, $\sigma(x) = 1/(1+e^{-x})$, the hyperbolic tangent activation, $f(x) = tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$, the rectified linear unit (ReLU) function, $f(x) = max\{0, x\}$, and their many variants. Additionally,

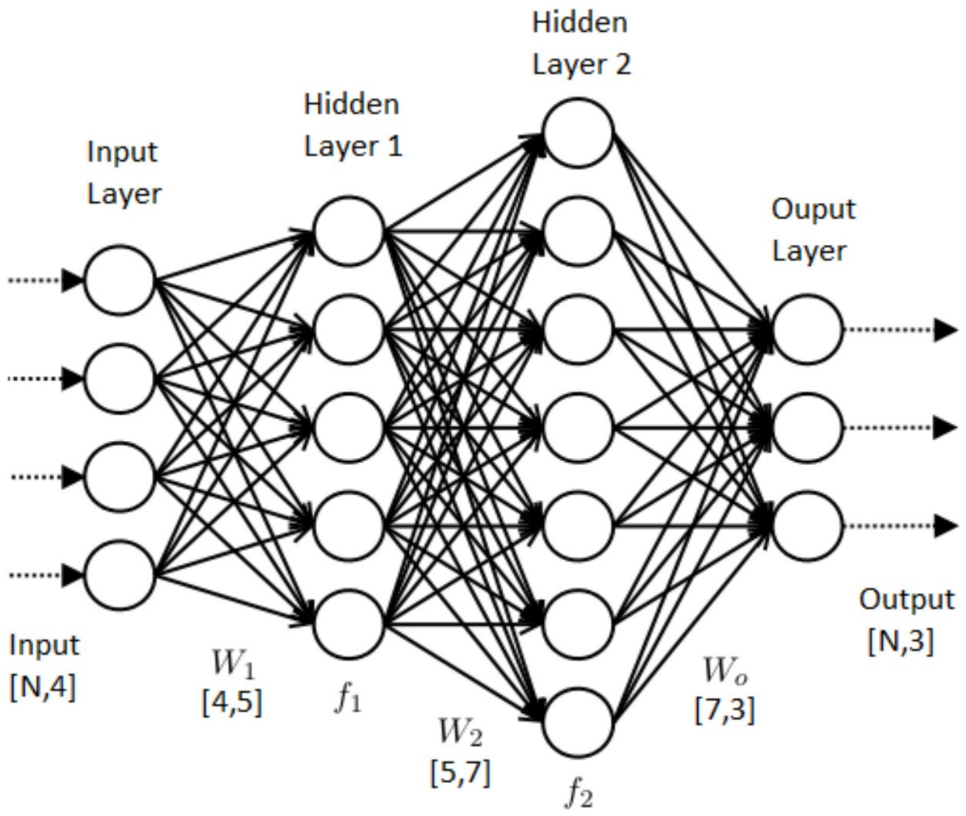


FIGURE 1.3: Illustration of typical ANN structure with an input layer, two hidden layers, and an output layer. Figure sourced from Medium - 'The Artificial Neural Networks Handbook'.

in modern ANNs, a learnable bias term is included in the affine transformation before the non-linear activation.

Thus, the output from the k^{th} neuron in the l^{th} layer of the feedforward network can be expressed as

$$y_k^{[l]} = \phi\left(\sum_j w_{kj}^{[l]} y_j^{[l-1]} + b_k^{[l]}\right) = \phi\left(\mathbf{W}_k^{[l]T}\mathbf{y}^{[l-1]} + \mathbf{b}_k^{[l]}\right) \tag{1.2}$$

where $\mathbf{W}_k^{[l]}$ and $\mathbf{b}_k^{[l]}$ represent trainable weight and bias matrices and vectors, respectively, corresponding to the k^{th} output from the network. The ability to stack weighted nonlinear transformations of the outputs from the previous layers allow these feedforward networks to approximate nearly any continuously valued function (7). Combined with their inherent dimensional flexibility, this makes them extremely useful for regression, estimation, and categorization problems on complicated and high dimensional datasets.

1.1.3 *Recurrent Neural Networks*

Recurrent neural networks (RNNs) are a class of ANNs suited for processing sequential or temporal data. Unlike traditional feedforward networks, where information simply flows unidirectionally from input to output, RNNs allow information to flow both forward to the subsequent layer, as well as in a recurrent loop back into the cell at the next time step. This allows them to inherit information from previous timesteps, thus giving the RNN the ability to learn and model data with temporal structure. This recurrent loop of information back into the cell at a subsequent timestep is a defining characteristic of the RNN, and can be visualized in Figure 1.4.

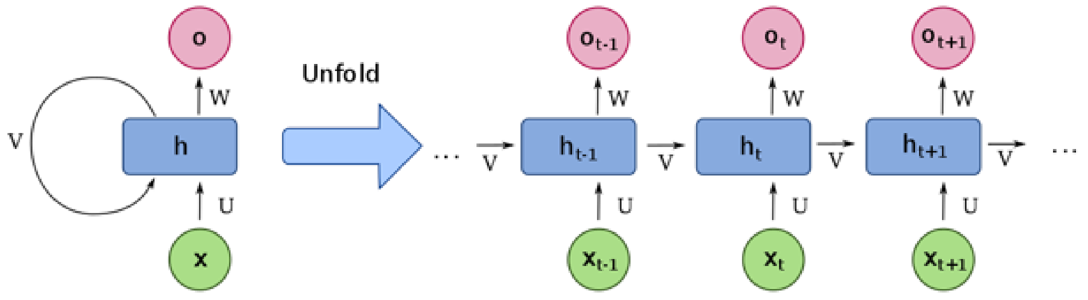


FIGURE 1.4: **Left:** RNN cell, complete with recurrent connections. **Right:** RNN cell, unfolded in time to reveal temporal connections between timesteps. Figure sourced from Wikipedia.

This recurrent connection that allows informational flow of previous outputs into the network is made possible through the creation of a second state within the recurrent neural network - this is known as the hidden state. The hidden state constitutes the RNN's second defining feature. The differences in output from the RNN arise from differences in the hidden state, **h**, which is updated at each time step. The following equations define an RNN's evolution over time

$$\mathbf{h}_t = g_\psi(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{1.3a}$$

$$\mathbf{y}_t = f_\theta(\mathbf{h}_t) \tag{1.3b}$$

Here, f_θ represents the nonlinear transformation of the hidden state vector, $\mathbf{h}$ into the network output at any given time, t, which is characterized by the set of weight and bias parameters, θ. Similarly, g_ψ represents the nonlinear transformation of both the previous timestep's hidden state, $\mathbf{h}_{t-1}$ and the input at the current timestep, $\mathbf{x}_t$, which is charracterized by the set of weight and bias parameters, ψ.

The nature of the general parametrized nonlinear functions, f_θ and g_ψ, as well as the inclusion of multiple variants and iterations on the concept and method of the cell having additional internal states constitutes the majority of difference amongst the various types of RNNs. In future sections, details of more complicated RNN structures will be covered in more detail. Here, we will cover the basic gating structure of what is known as a simple, or 'vanilla', RNN.

$$\mathbf{h}_t = tanh\big(\mathbf{W}_{xh}\mathbf{x_t} + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h\big) \tag{1.4a}$$

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y \tag{1.4b}$$

Here, $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{hy}$ are trainable weight matrices, and $\mathbf{b}_h$ and $\mathbf{b}_y$ are trainable bias vectors. The hidden state is a nonlinear transformation of the previous hidden state and the input to the network in the current time step, and the output is a linear transformation of the hidden state. Thus, all differences in output sequence are dependent upon differences in computation of the hidden state within each timestep.

1.1.4 *Generative Neural Networks*

Generative models constitute a class of unsupervised learning techniques in ANN-based modeling approaches to high-dimensional datasets. Broadly speaking, their goal is to learn the inherent relationships governing high-dimensional, stochastic datasets.

To begin, consider a set of n data points, $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ as samples from some distribution, $P(x)$. It is possible that these data arise from a high-dimensional, stochastic process involving the interactions of an incredible number of variables. In this case, it becomes arduous or outright infeasible to develop closed-form equations describing the interactions of these stochastic variables, casting doubt on the possibility to develop equations capable of re-generating the original input dataset, or even generating entirely new samples.

However, it is very well known how to generate simple random distributions, such as a Gaussian distribution. Additionally, it is well known that ANNs may learn to approximate any given function given appropriate parameters, θ. Thus, generative models aim to learn transformations that generate highly complex, deterministic transformations of simple stochastic distributions into highly complex distributions, such that the network's recreation of the data, $\hat{P}_\theta(x)$, closely resembles the original distribution of data, $P(x)$. This deterministic ANN that learns to transform samples from a simple distribution to approximate samples from a highly complex one is known as a *generative neural network*.

There are two major archetypes of ANN-based generative modeling approaches: the variational autoencoder (VAE) (13) and the generative adversarial network (GAN) (14). The VAE and GAN seek to solve the same types of problems in machine learning, however they take very different approaches to doing so.

The VAE poses this problem in the framework of a probabilistic graphical model, where we seek to maximize the evidence lower bound (ELBO) on the log-likelihood of the data. GANs pose training as a min-max game, in which two networks are employed, a generator network and a classifier network, also known as the discriminator. The generator seeks to turn samples of gaussian noise into complicated data distributions, while the discriminator tries to discern whether or not the generated samples are from the original data or not. Given proper training, the discriminator will grow better at discerning real samples from fake samples over time, while the generator improves at fooling the discriminator. The game has finished when the discriminator can no longer tell the difference between samples generated from the generator from those from the original dataset.

1.1.5 *Training Neural Networks*

Training neural networks is framed as a constrained optimization problem, in which the goal is to find an optimal set of weights in a given architecture that minimizes some objective function, $\mathcal{L}$, commonly referred to as a loss function. This can be formally expressed as

$$\theta^* = \arg\min_{\theta} \mathcal{L} \tag{1.5}$$

The exact form of the loss function varies wildly from problem to problem, and constitutes much of the novelty of modern machine learning. For regression problems, common loss functions include the mean squared error, the absolute mean error, the smooth mean absolute error, the quantile loss, and the log-cosh loss. For problems of classification, common loss functions include the cross-entropy loss, the hinge loss, and the Kullback-Leibler Divergence loss.

Training is carried out in an iterative learning procedure known as backpropagation (15; 16; 17), in which parameter updates are calculated as the solution to a stochastic gradient descent problem.

1.2 SPARSE IDENTIFICATION OF NONLINEAR DYNAMICS

The ability to distill closed-form dynamical equations governing a system's behavior has been of interest to scientists, engineers, and mathematicians from nearly every field for decades. Recent advances in neural networks have allowed for the modeling of hugely-complex time series - however, they still lack interpretability, leading to questions of the robustness of the internal representation that they possess when working to extrapolate outside of the dataset they were trained on.

Sparse identification of nonlinear dynamics (SINDy) (18) is a spare regression technique developed to find a closed-form representation of a non-linear dynamical system by performing sparse regression on a large set of basis candidate functions within an optimization framework. Similarly to many problems in modeling, it is desired in the SINDy framework to recover dynamical equations of the form

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}(\mathbf{x}(t)) \tag{1.6}$$

where d/dt represents the time derviative, $\mathbf{x}(t)$ is the state of the dynamical system, and $\mathbf{F}$ represents the nonlinear equations governing the time evolution of the system.

In this framework, model identification is formulated as a sparse regression optimization problem. SINDy assumes that time series data is observed at n discrete time intervals, such that m individually observed time-series from the system can be stacked into large matrix form, as

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m]^T \tag{1.7}$$

with numerically computed time-derivatives as

$$\dot{\mathbf{X}} = [\dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2, ..., \dot{\mathbf{x}}_m]^T \tag{1.8}$$

where $\mathbf{X}$ represents the m x n matrix of observed state values, $\dot{\mathbf{X}}$ represents the m x n matrix of numerically computed state derivatives, and $\mathbf{x}_i$ represent individually measured vectors of system states.

In this framework, the governing equations underlying X are unknown. Thus, a set of p candidate basis functions, $\mathbf{\Theta}$ must be constructed, such that each element of the candidate basis functions describing the right hand side of Equation 1.6, θ_j, is a candidate model term. This candidate library can be expressed as

$$\mathbf{\Theta}(\mathbf{X}) = [\boldsymbol{\theta}_1(\mathbf{X}), \boldsymbol{\theta}_2(\mathbf{X}), ..., \boldsymbol{\theta}_p(\mathbf{X})] \tag{1.9}$$

Now, the set of potential mathematical descriptors of the system's dynamical behaviors can be expressed as

$$\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X}) \tag{1.10}$$

However, this raises the issue of non-feasibility to compute in numerical integration, and is unlikely to model anything meaningful, as all candidate terms are contributing equally to the estimated dynamics. The solution to this issue is to introduce a p x n matrix of coefficients, Ξ, that can be optimized to provide insight into active dynamical terms. Formally, this matrix is defined as

$$\Xi = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, ..., \boldsymbol{\xi}_n] \tag{1.11}$$

where each individual term, ξ_j represents a vector determining which coefficients are active in the overall expression for the k^{th} term in $\mathbf{x}$, $\mathbf{x}_k$. Now, the estimated dynamics can be expressed as

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\Xi \tag{1.12}$$

with

$$\dot{\mathbf{x}}_k = \boldsymbol{\Theta}(\mathbf{x}^T)\boldsymbol{\xi}_k \tag{1.13}$$

This entire problem is then cast as a numerical optimization problem, in which the objective is to minimize the reconstruction cost of the observed timeseries data by optimizing the matrix of coefficients determining which terms in $\boldsymbol{\Theta}$ are active, that is

$$\Xi^* = \underset{\Xi}{\arg\min} \, \|\boldsymbol{\Theta}(\mathbf{X})\Xi - \dot{\mathbf{X}}\|_2 + \lambda \| \|\Xi\|_1 \tag{1.14}$$

where $\| \cdot \|_2$ represents the L2 norm, and $\| \cdot \|_1$ represents the L1 norm, and λ is a tunable scalar hyperparameter used to tune the sparsity of the regressed solution The inclusion of L2 error between the reconstructed dynamics and the data observed promotes accuracy of solution, while the L1 norm encourages sparsity and interpretability of the regressed solution.

1.3 NEURONAL MODELING

Neurons compromise the basic cellular building blocks of our nervous system. A typical biological neuron is comprised of a cell body, dendrites extending from the cell body, and a long axon. The dendrites extending away from the cell body establish synapses with the axons of other neurons, forming a large cellular network through which the information flows. It is well understood that neurons largely communicate through electric signals known as action potentials.

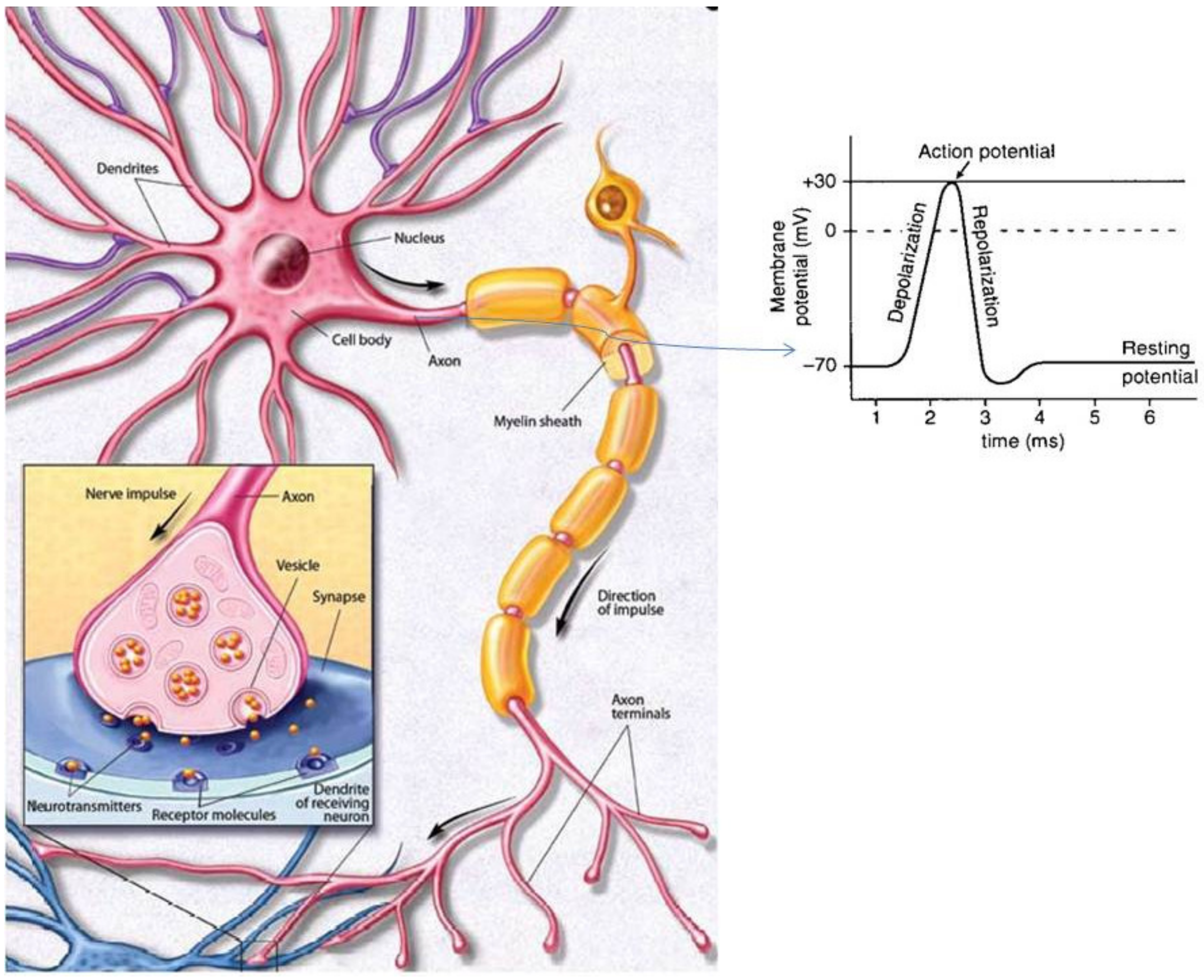


Figure 1.5: **Left:** A single neuron. Figure sourced from 'Brain Facts - Society for Neuroscience'. **Right:** Schematic of an action potential.

Broadly speaking, neurons receive electric inputs from other neurons at their dendrite terminals. This input is processed within the cell body, causing a rise in the voltage on the cell's membrane surface, a phenomenon known as depolarization. Once the input to the cell exceeds a certain threshold, there is a large spike of voltage, known as an action potential, followed by a strongly negative drop in potential, known as repolarization/hyperpolarization. After the repolarization, the neuron goes into a refractory period before another action potential can be generated based on the incoming inputs. Figure 1.5 shows a typical morphological structure of a neuron along with an action potential.

In computational neuroscience, a variety of modeling approaches and models exist to describe how a neuron process the information and generate action potentials. At a broader level, these models can be categorized into five classes based on the level of details in incorporating the underlying biological mechanisms: detailed compartmental models, reduced compartmental models, single-compartment models, cascade models, and black-box models (1). Figure 1.6 illustrates this hierarchy of model types.

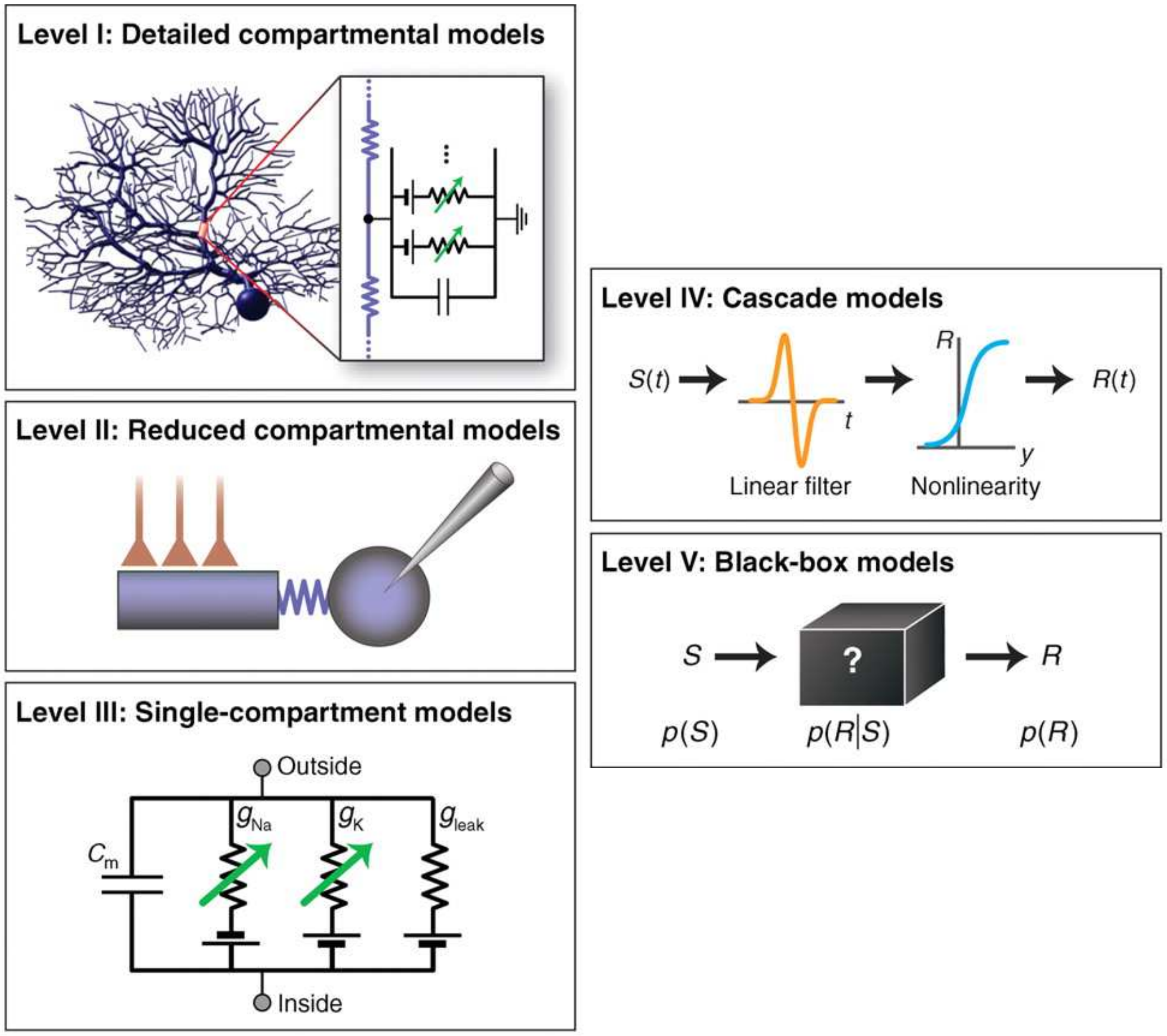


FIGURE 1.6: Illustration of various levels of neuronal modeling. Figured sourced from 'Modeling the Mind - Science (1)'.

Detailed compartmental models focus on the fine details of spatial structure of a neuron and their contributions to dynamical behaviors. This is also true, albeit to a lesser degree, of reduced compartmental models. However, both of these model types are extremely computationally expensive due to the incredible detail they possess - this renders them unsuitable for use in modeling the behaviors of neuronal networks.

Reduced compartmental models, such as the famous Hodgkin-Huxley model (19; 20; 21; 22), ignore the spatial structure of the cell, treating the cell as a walled objects, with ionic channels modeled as circuits governing the dynamical relationships between the cell's membrane potential and the ionic currents that flow across it. This model, as well as its expanded and simplified forms, have found great success in describing a variety of neurological behavioral responses to externally applied input currents, including bursting spiking, regular spiking, irregular bursting spiking, as well as bifurcations in dynamics of spiking responses (23; 24; 25; 26).

Cascade models and black box models opt for non-biophysical models of spiking activity, and instead focus on probablistic frameworks of collective neuronal behaviors. For this reason, they are useful when studying the behavior of large populations of neurons, when numerical simulation of individual neurons within the network becomes computationally infeasible when considering biophysical models.

1.3.1 *Neuronal Network Models*

It is apparent that neurons influence each other's behavior, as they pass electric signals along synaptic connections, which are then transformed and passed on once again, forming a biological network of informational flow. Mathematical models of how these synapses integrate and process these synaptic currents have been developed. Typically, this is modeled as

$$I^s(t) = -g_e(t)\big(v(t) - E_e\big) - g_i(t)\big(v(t) - E_i\big) \tag{1.15}$$

where $g_e(t)$ and $g_i(t)$ represent excitatory and inhibitory synapse conductances, and E_e and E_i are excitatory and inhibitory membrane reversal potentials. The synaptic conductances, $g_e(t)$ and $g_i(t)$, are modeled by as the weighted sum of all relevant connected neuron's presynaptic activity as given in (24)

$$g_e(t) = \sum_{j=1}^{N_e} \sum_f w_j K(t - t_j^f) \tag{1.16}$$

$$g_i(t) = \sum_{j=1}^{N_i} \sum_f w_j K(t - t_j^f) \tag{1.17}$$

where w_j is the weight of the j^{th} synapse connected to the post-synaptic neuron, and t_j^f is the event time of the f^{th} action potential incoming to the post synaptic neuron from the j^{th} neuron. Additionally, here the term $K(t - t_j^f)$ represents the time course of postsynaptic conductances following a pre-synaptic spike. Two typical expressions for are

$$K(t - t_j^f) = \frac{q_j}{\tau_s}(t - t_j^f) e^{-\left(\frac{t - t_j^f}{\tau_s}\right)} \Theta(t - t_j^f) \tag{1.18}$$

$$K(t - t_j^f) = \frac{q_j}{\tau_s} e^{-\left(\frac{t - t_j^f}{\tau_s}\right)} \Theta(t - t_j^f) \tag{1.19}$$

where q_j represents the maximum conductance transmittable through an action potential to the j^{th} synapse. Additionally, τ_s represents a time constant, and $\Theta(\cdot)$ is the Heavyside function, returning 1 for argument $t - t_j^f > 0$ and 0 otherwise.

When parameters are properly fit, these equations combined with a model of single-neuron dynamics, such as the Hodgkin-Huxley model, are capable of predicting qualitative behavior of large-scale neuronal networks in various dynamical regimes.

However, as neuronal network scale increases, so does computational complexity involved in numerical simulation. Additionally, fitting the large number of parameters involved poses a non-trivial multivariate numerical optimization problem of its own. This renders these types of models problematic for the development of high-speed dynamical inference algorithms.

To overcome this potential challenge, an alternate framework that has been employed is to treat neurons as a point process with a continuously-valued instantaneous firing rate (27; 28).

1.4 THESIS OVERVIEW

This thesis is organized in the following manner: In **Chapter 2**, we develop a deep long-short term memory (LSTM) approach to predict long-horizon dynamical responses of a experimentally validated, 9-dimensional Hodgkin-Huxley neuron. We demonstrate that as the length of the predictive horizon increases, so does the accuracy of the predictions from the network. In **Chapter 3**, we develop a generative machine learning method, specifically a sequential adaptation of a VAE framework, to infer poisson firing rates for circuit models of neuronal behaviors. We validate the performance of this approach on both chaotic and non-chaotic synthetic datasets, and demonstrate that the framework is capable of discovering cohesive dynamical representations of the underlying firing rates. Finally, we discuss limitations and future works in **Chapter 4**.

CHAPTER 2

Data-Driven Predictive Modeling of Neuronal Dynamics Using Long Short-Term Memory[1]

2.1 INTRODUCTION

Our brain generates highly complex nonlinear responses at multiple temporal scales, ranging from few milliseconds to several days, in response to an external stimulus (29; 30; 31). One of the long-time interests in computational neuroscience is to understand the dynamics underlying various cognitive and non-cognitive brain functions by developing computationally efficient modeling and analysis approaches. In the last four decades or so, several advancements have been made in the direction of dynamical modeling and analysis of brain dynamics (25; 32; 33). In the context of modeling the dynamics of single neurons, several modeling approaches, ranging from detailed mechanism-based biophysiological modeling to simplified phenomenological/probabilistic modeling, have been developed to understand the diverse firing patterns (e.g., simple spiking to bursting) observed in electrophysiological experiments (1; 34). These models provide a detailed understanding of various ionic mechanisms that contribute to generating specific spiking patterns as well as allowing the performance of large-scale simulations to understand the dynamics underlying cognitive behaviors. However, most of these models are computationally expensive from the perspective of developing novel real-time neurostimulation strategies for controlling neuronal dynamics at single neurons and network levels. In this chapter, we investigate purely data-driven long short-term memory (LSTM) based recurrent neural network (RNN) architectures in multi-timestep predictions of a single neuron's dynamics for the use in developing novel neurostimulation strategies in an optimal control framework.

The availability of an abundant amount of data and advances in machine learning has recently revolutionized the field of predictive data-driven dynamical modeling

[1]Plaster, B., Kumar, G. (2019). Data-Driven Predictive Modeling of Neuronal Dynamics using Long Short-Term Memory. Algorithms, 12(10), 203.

of complex systems using neural networks (NNs) and deep learning approaches. Various nonlinear system identification approaches have been developed to map static input-output relations using multi-layer perceptrons (MLPs) (35; 36; 37; 38) and their variations (39; 40). Reinforcement learning has recently been explored in robotics dynamical modeling in Reference (41). NN architectures that make use of vanilla recurrent neural network (RNNs) elements have also been explored for nonlinear system identification and modeling in References (42; 43; 44). However, network architectures that make use of vanilla recurrent layers often suffer from the exploding or vanishing gradient problem when used to model dynamics over long time series horizons (45). In Reference (44), a highly specialized multi-phase training algorithm was used to ensure that the network did not suffer from this problem. LSTM based approaches to modeling dynamical systems (46; 47; 48) mitigate the vanishing gradient problem but suffer from poor early trajectory predictive performance when using long predictive horizons (49; 46). LSTMs have been used to model high-dimensional chaotic systems (50) but these studies have been limited to single step prediction applications. Additionally, machine learning techniques have begun to be explored in neuroscientific modeling applications. Multi-layer Spiking Artificial Neural Networks (SANNs) for use in spatiotemporal spike pattern transformations have been developed in References (51; 52). This is achieved by using novel approximations and surrogates of the partial derivatives of the spike train functions with respect to the weights. This partial derivative is typically problematic when used in backpropagation, as it is undefined at spike times for many neuronal models, rendering it incompatible with traditional backpropagation-based approaches. In Reference (53), a novel gated recurrent unit (GRU) based encoder/decoder approach is used to learn and predict neuronal population dynamics and kinematic trajectories from single-trial spike train data.

In this chapter, we have developed a novel deep LSTM neural network architecture, which can make multi-timestep predictions in large-scale dynamical systems. In particular, we use a reversed sequence-to-sequence mapping technique, developed for language translation applications of multi-layer LSTM networks in Reference (6), and generalize the application of this technique to dynamical systems time-series forecasting. Figure 2.1 illustrates our overall approach.

In contrast to existing approaches in modeling dynamical systems using neural networks, our architecture uses (1) stacked LSTM layers in conjunction with a single densely connected layer to capture temporal dynamic features as well as input/output features; (2) sequence-to-sequence mapping, which enables multi-timestep predictions; and (3) reverse ordered input and measured state trajectories to the network, resulting in highly accurate early predictions and improved performance over long horizons. We show the efficacy of our developed approach in making stable multi-timestep predictions of various firing patterns exhibited by hippocampal CA1 pyramidal neurons, obtained from simulating an experimentally validated highly nonlinear 9-dimensional Hodgkin-Huxley model of CA1 pyramidal cell dynamics, over long time-horizons. Our approach is contingent on the network being trained on the entire state vector of the neuronal model.

The remaining chapter is organized as follows. In Section 2.2, we describe our developed deep LSTM neural network architecture and methodological approach to data-driven multi-timestep predictions of dynamical systems. We show the efficacy of our approach in making stable multi-timestep predictions over long time-horizons of neuronal dynamics in Section 2.3, which is followed by a thorough discussion on the limitations of our approach in Section 2.5.

In Section 2.2.1, we describe our developed deep LSTM neural network architecture which combines stacked LSTMs with a fully-connected dense output layer. We describe the sequence-to-sequence mapping with reversed order input sequences used throughout this chapter in Section 2.2.2. In Section 2.2.3, we provide the details on the synthetic data used to train our networks. Finally, in Section 2.2.5, we provide the details on the approach used to train the developed neural network architecture.

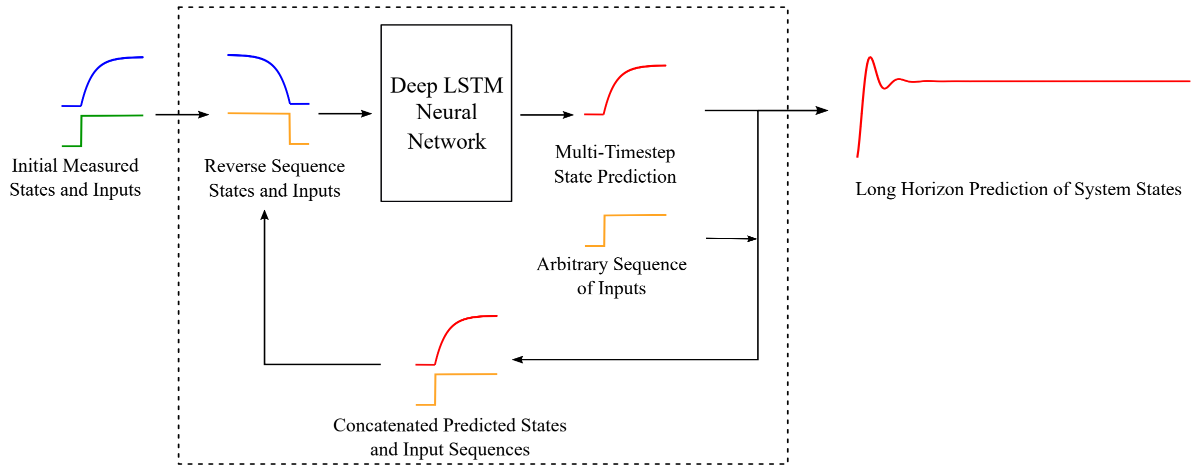

Initial Measured States and Inputs
Reverse Sequence States and Inputs
Deep LSTM Neural Network
Multi-Timestep State Prediction
Long Horizon Prediction of System States
Arbitrary Sequence of Inputs
Concatenated Predicted States and Input Sequences

Figure 2.1: A schematic illustrating the overall data-driven approach developed in this chapter for multi-timestep predictions of high-dimensional dynamical systems' behavior over a long time-horizon. An initial sequence of states and inputs are fed to the "Stacked LSTM Network" in a reverse-order for multi-timestep prediction of the system's states ("Reverse-order sequence-to-sequence mapping"). The predicted output from each stacked LSTM network is concatenated with the next sequence of inputs and fed into the next stacked LSTM network in a reverse-order to increase the predictive horizon. This process is iterated an arbitrary number of times, creating long dynamical predictions.

2.2 NEURAL NETWORK ARCHITECTURE, ALGORITHM AND APPROACH

In Section 2.2.1, we describe our developed deep LSTM neural network architecture which combines stacked LSTMs with a fully-connected dense output layer. We describe the sequence-to-sequence mapping with reversed order input sequences used in this work in Section 2.2.2. In Section 2.2.3, we provide the details on the synthetic data used to train our networks. Finally, in Section 2.2.5, we provide the details on the approach used to train the developed neural network architecture.

2.2.1 *Deep LSTM Neural Network Architecture*

Long short-term memory (LSTM) neural networks (54) are a particular type of recurrent neural networks (RNNs) which mitigate the vanishing or exploding gradient problem during the network training while capturing both the long-term and the short-term temporal features in sequential time-series data processing (45). Specifically, LSTM uses multiple gating variables that control the flow of information of a hidden cell state and assign temporal importance to the dynamical features that are present in the time series data flowing through the cell state. Figure 2.2 shows a schematic illustrating the internal gating operation in a single LSTM cell.

A forward pass of information through a single LSTM cell is described by the following cell and gating state equations (reference):

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c), \tag{2.1a}$$

$$h_t = o_t \circ \sigma_c(c_t). \tag{2.1b}$$

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f), \tag{2.1c}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i), \tag{2.1d}$$

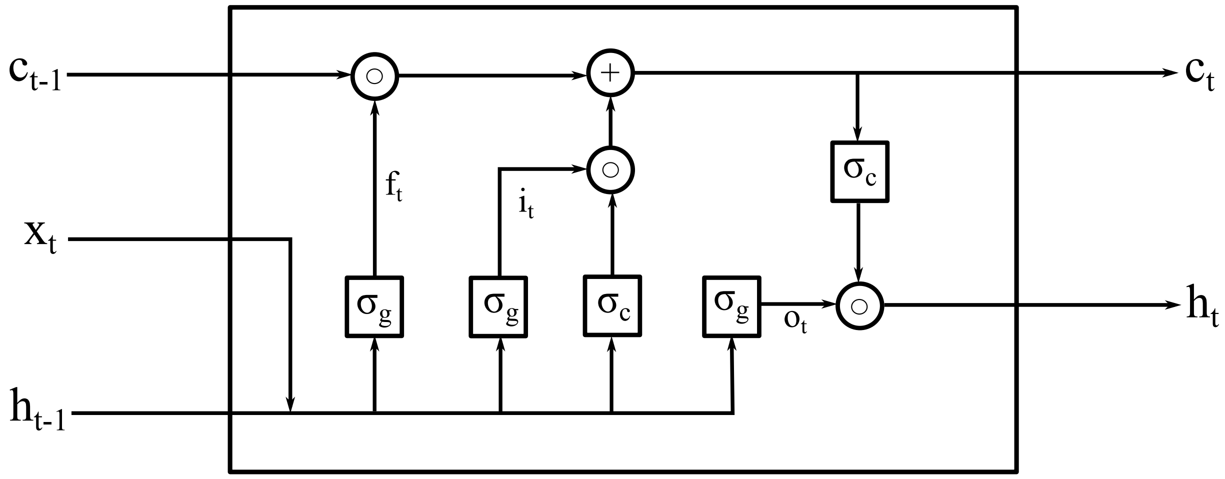


FIGURE 2.2: A schematic illustrating the internal gating operation in a single LSTM cell. The "+" represents an additive operation and the "∘" represents a multiplicative operation. σ_g is the sigmoidal activation function and σ_c is the hyperbolic tangent activation function.

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o), \tag{2.1e}$$

in Equations 2.1a–2.1e, $c_t \in \mathbb{R}^h$ and $h_t \in \mathbb{R}^h$ represent the cell state vector and the hidden state vector, respectively, at time t. $f_t \in \mathbb{R}^h$, $i_t \in \mathbb{R}^h$, and $o_t \in \mathbb{R}^h$ are the "forget gate", "input gate", and "output gate" activation vector, respectively, at time t. $x_t \in \mathbb{R}^d$ is the input vector to the LSTM unit at time t, and $h_{t-1} \in \mathbb{R}^h$ is the previous time step hidden state vector passed back into the LSTM unit at time t. The matrices W_f, W_i, and W_o represent the input weights for the "forget gate", "input gate", and "output gate", respectively. The matrices U_f, U_i and U_o represent the weights of the recurrent connections for the "forget gate", "input gate", and "output gate", respectively. The vectors b_f b_i, and b_o represent the "forget gate", "input gate", and "output gate" biases, respectively. $\circ$ represents the element-wise multiplication. The function σ_g represents the sigmoidal activation function, and σ_c is the hyperbolic tangent activation functions.

Here, we use stacked LSTM network integrated with a fully connected feedforward output layer to make multi-timestep state predictions. The use of a single feedforward dense output layer allows the network to effectively learn the static input-output features, while the stacked LSTM network captures the temporal dynamical features. To appropriately select the optimum dimensionality of the hidden states in a single hidden layer, we systematically varied the number of hidden states in a sequence of $\{n, n^2, 2n^2, 4n^2, \cdots\}$, where n is the dimension of the system's state and evaluated the training performance for each case. We found that for our application ($n = 9$), a hidden state dimensionality of $4n^2 = 324$ was optimal in learning dynamical behaviors while avoiding overfitting. To select the number of hidden layers, we systematically increased the number of hidden layers of identical hidden state dimensionality (i.e., 324 states) and compared the network performance during the training. We found that increasing the number of hidden layers beyond 3 layers did not improve the network performance on the training and validation dataset. Thus, we fixed the number of hidden layers to 3 in our study. Throughout this chapter, we utilized stateless LSTMs which reset the internal cell and hidden states to zero after processing and performing gradient descent for a given minibatch. We initialized the network weights using the Xavier method (55). Specifically, the initial weights were drawn from a uniform distribution using

$$W_{ij} \sim \mathcal{U}\Big(-\frac{6}{\sqrt{n_j + n_{j+1}}}, \frac{6}{\sqrt{n_j + n_{j+1}}}\Big), \tag{2.2}$$

where n_j is the dimensionality of the input units in the weight tensor, and n_{j+1} is the dimensionality of the output units in the weight tensor.

To generate a long time-horizon dynamical prediction beyond the multi-timestep prediction by a single stacked deep LSTM neural network (shown as "Deep LSTM" in Figure 2.3), we used an iterative approach as described here. We made copies of the trained single stacked LSTM network and connected them in the feedforward manner in a sequence. We concatenated the sequence of predicted output from the previous stacked LSTM network with an equivalent length sequence of new inputs to the system and fed them in the reverse sequence order to the next stacked LSTM network. Figure 2.3 illustrates this iterative approach.

2.2.2 *Sequence to Sequence Mapping with Neural Networks*

To make multi-timestep predictions of dynamical systems' outputs using the deep LSTM neural network architecture described in the previous section (Section 2.2.1), we formulate the problem of mapping trajectories of the network inputs to the trajectories of the predicted outputs as a reverse order sequence-to-sequence mapping problem. The central idea of the reverse order sequence-to-sequence mapping approach is to feed the inputs to the network in reverse order such that the network perceives the first input as the last and the last input as the first. Although this approach has been developed and applied in language translation applications (6), it has never been considered in the context of predicting dynamical systems behaviors from time-series data. Figure 2.4 illustrates the basic idea of the reverse order sequence-to-sequence mapping approach for translating letters (inputs) to their numerical indices (outputs).

As shown in Figure 2.4, in the forward sequence-to-sequence mapping approach (Figure 2.4a), that is, $A, B, C \rightarrow 1, 2, 3$, the distance between all mappings is same

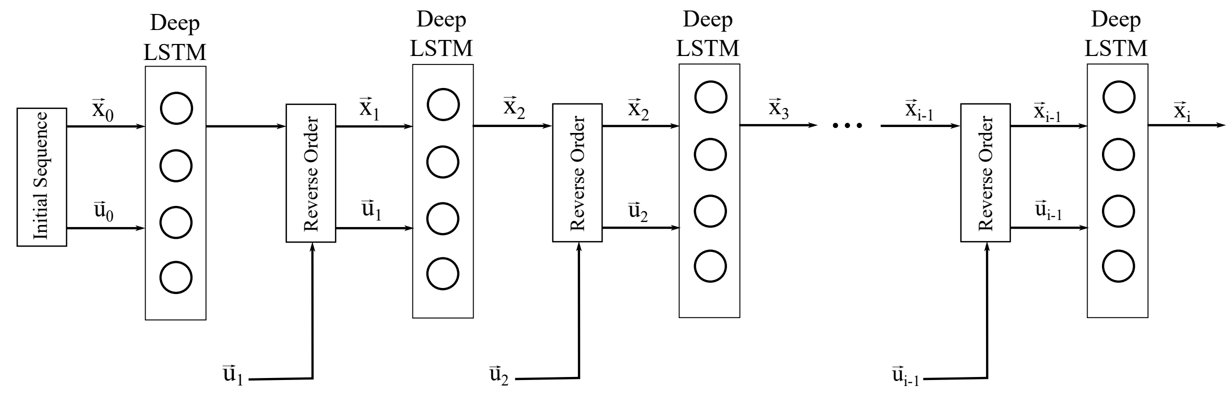


Figure 2.3: Iterative prediction of the system's outputs over a long time-horizon. Each "Deep LSTM" receives the predicted sequence of outputs from the previous "Deep LSTM" and an equivalent length of new system's inputs in reverse order and predict the next sequence of outputs of same time duration in future.

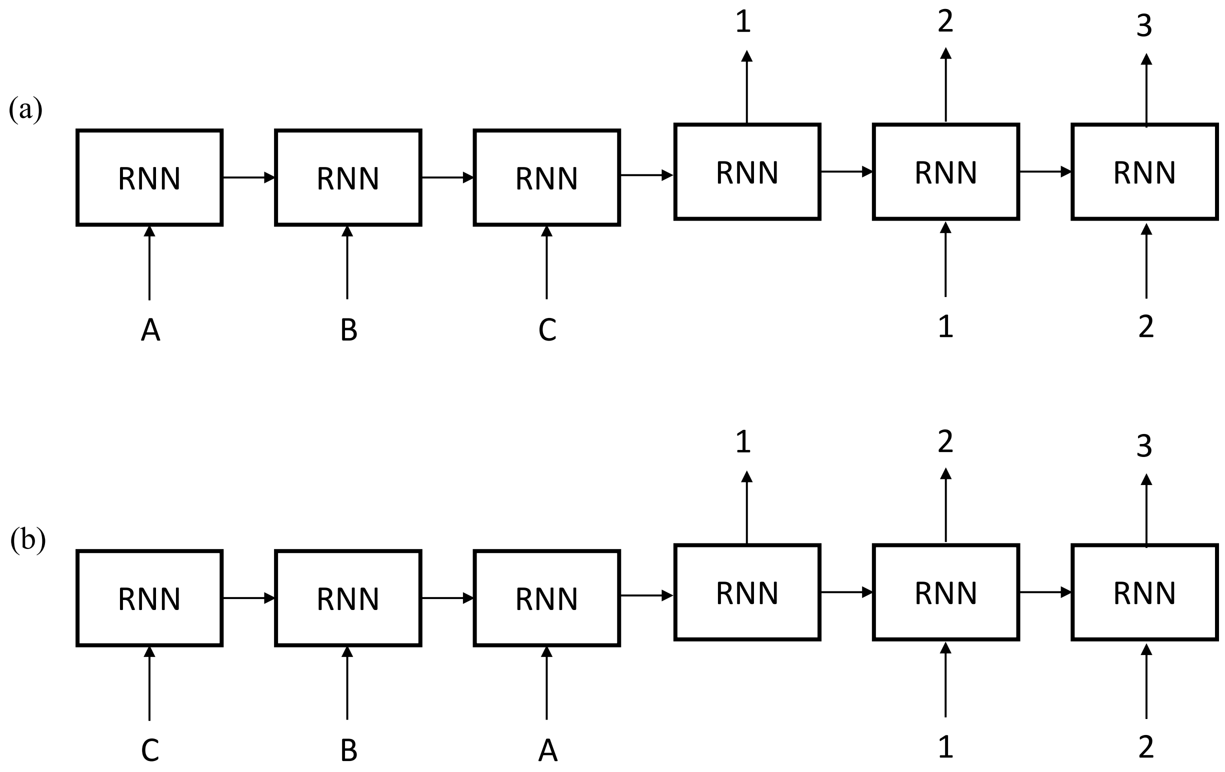


Figure 2.4: Forward and reversed sequence-to-sequence mapping approach for translating letters (inputs) to their numerical indices (outputs) in recurrent neural network (RNN). (**a**) shows the forward sequence-to-sequence mapping approach. The input is fed into the network in the same sequence as the desired output. The "distance" between all corresponding inputs and outputs is uniform. (**b**) shows the reversed sequence-to-sequence mapping approach. This approach introduces a temporal symmetry between input and output sequences while keeping the average "distance" between the corresponding inputs and outputs same as the forward approach. As shown in (**b**), A$\rightarrow$ 1 is the shortest "distance" to map, $B \rightarrow 2$ the second, and $C \rightarrow 3$ the furthest.

(i.e., 3 "units"). In the reverse sequence-to-sequence mapping approach (Figure 2.4b), the network receives the input in a reverse order to map to the target output sequence, i.e., $C, B, A \rightarrow 1, 2, 3$. As noted here, the average distance between the mappings remains the same for both approaches (i.e., 3 "units") but the reverse order approach introduces short and long-term symmetric temporal dependencies between inputs and outputs. These short and long-term symmetric temporal dependencies provide improved predictive performance over long temporal horizons (6).

2.2.3 *Synthetic Data*

Hippocampal CA1 pyramidal neurons exhibit various multi-timescale firing patterns (from simple spiking to bursting) and play an essential role in shaping spatial and episodic memory (56). In the last two decades, several biophysiological models of the CA1 pyramidal (CA1Py) neurons ranging from single compartmental biophysiological and phenomenological models (26; 57; 58) to detailed morphology-based multi-compartmental models (59; 60; 61; 62; 63; 64; 65) have been developed to understand the contributions of various ion-channels in diverse firing patterns (e.g., simple spiking to bursting) exhibited by the CA1Py neurons.

In this work, we use an experimentally validated 9-dimensional nonlinear model of CA1 pyramidal neuron in the Hodgkin-Huxley formalism given in Reference (26) to generate the synthetic data for the network training and validation. The model exhibits several different bifurcations to the external stimulating current and has shown its capability in generating diverse firing patterns observed in electrophysiological recordings from CA1 pyramidal cells under various stimulating currents. Figure 2.5 shows three different firing patterns generated from this model based on the three different regimes of the applied input currents.

To construct the synthetic training and validation dataset for the deep LSTM neural networks we designed in this chapter with different predictive horizons, we simulated the Hodgkin-Huxley model of CA1 pyramidal neuron given in (26) (see Section 2.2.4 for the details of the model) for 1000 ms duration for 2000 constant stimulating currents, sampled uniformly between $I = 0.0$ nA and $I = 3.0$ nA. From these 2000 examples, we randomly and uniformly drew 50 samples (i.e., 10^4 data points)

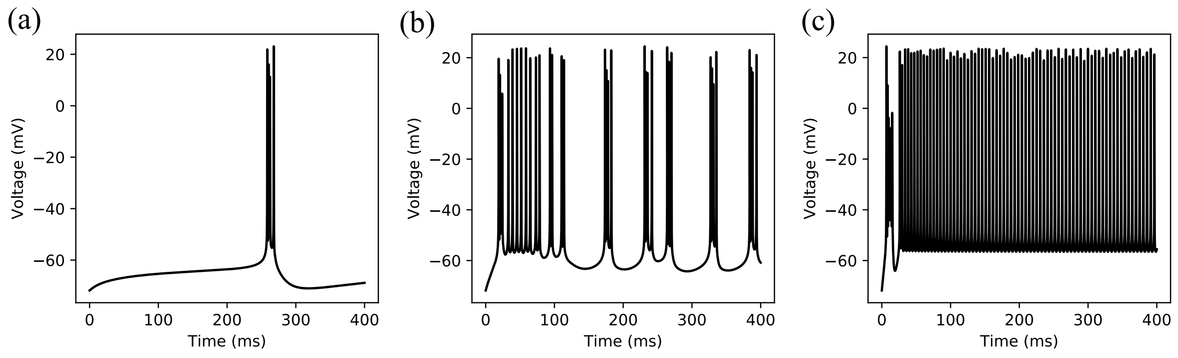


Figure 2.5: Diversity in the spiking patterns of hippocampal CA1 pyramidal neurons to applied currents. (**a**) Regular bursting in response to the external current of 0.23 nA. (**b**) Irregular bursting in response to the external current of 1.0 nA. (**c**) Plateau potentials followed by regular spiking in response to the external current of 3.0 nA.

of the desired predictive horizon as the input/output sequence data for training and validation. As described in Section 2.2.5, we used 1/32 of these data points for validation, that is, 96,875 data points for the training and 3125 data points for the validation. Since our deep LSTM neural network takes an initial sequence of outputs of appropriate predictive horizon length (i.e., $N_p = 1, 50, 100, 200$) as an input sequence to make the next time-horizon prediction of equivalent length of sequence, we assume that this initial output sequence data is available to the deep LSTM neural network throughout our simulations.

2.2.4 *Hodgkin-Huxley Model of CA1 Pyramidal Neuron Dynamics*

We used the following Hodgkin-Huxley model of CA1 pyramidal neuron from (26) to demonstrate the efficacy of our data-driven modeling approach presented in this chapter:

$$C\frac{dV}{dt} = -g_L(V - V_L) - I_{Na} - I_{NaP} - I_{Kdr} - I_A - I_M - I_{Ca} - I_C - I_{sAHP} + I_{app}, \quad (2.3)$$

where the ionic currents I_{Na}, I_{NaP}, I_{Kdr}, I_A, I_M, I_{sAHP}, I_C, and I_{Ca} are given by

$$I_{Na} = g_{Na} m_\infty^3(V) h_{Na}(V - V_{Na}), \quad (2.4a)$$

$$I_{NaP} = g_{NaP} p_\infty(V)(V - V_{Na}), \quad (2.4b)$$

$$I_{Kdr} = g_{Kdr} n_{Kdr}^4(V - V_K), \quad (2.4c)$$

$$I_A = g_A a_\infty^3(V) b_{Kdr}(V - V_K), \quad (2.4d)$$

$$I_M = g_M z_M(V - V_K), \quad (2.4e)$$

$$I_{Ca} = g_{Ca} r_{Ca}^2(V - V_{Ca}), \quad (2.4f)$$

$$I_C = g_C d_\infty([Ca^{2+}]_i) c_C (V - V_K), \tag{2.4g}$$

$$I_{sAHP} = g_{sAHP} q_{sAHP} (V - V_K), \tag{2.4h}$$

here, V is the membrane potential in mV, C is the membrane capacitance, V_L is the reversal potential of the leak current, g_L is the conductance of the leak current, and I_{app} is the externally applied stimulating current. The ionic currents I_{Na}, I_{NaP}, I_{Kdr}, I_A, I_M, I_{sAHP}, I_C, and I_{Ca} represent the transient sodium current, persistent sodium current, delayed rectifier potassium current, A-type potassium current, muscarinic-sensitive potassium current, slow calcium-activated potassium current, fast calcium-activated potassium current, and high threshold calcium current respectively. g_i, $i \in \{Na, NaP, Kdr, A, M, Ca, C, sAHP\}$ represents the conductance of the ion channel i. V_i, $i \in \{Na, K, Ca\}$ is the reversal potential of the ion channel i.

The dynamics of the transient activation/deactivation variables of the ionic and calcium currents, i.e., h_{Na}, n_{Kdr}, b_{Kdr}, z_M, r_{Ca}, c_C, q_{sAHP}, and $[Ca^{2+}]_i$, are given by:

$$\frac{dh_{Na}}{dt} = \phi \frac{h_\infty(V) - h_{Na}}{\tau_{h_{Na}}(V)}, \tag{2.5a}$$

$$\frac{dn_{Kdr}}{dt} = \phi \frac{n_\infty(V) - n_{Kdr}}{\tau_{n_{Kdr}}(V)}, \tag{2.5b}$$

$$\frac{db_{Kdr}}{dt} = \frac{b_\infty(V) - b_{Kdr}}{\tau_{b_{Kdr}}}, \tag{2.5c}$$

$$\frac{dz_M}{dt} = \frac{z_\infty(V) - z_M}{\tau_z}, \tag{2.5d}$$

$$\frac{dr_{Ca}}{dt} = \frac{r_\infty(V) - r_{Ca}}{\tau_{r_{Ca}}}, \tag{2.5e}$$

$$\frac{dc_C}{dt} = \frac{c_\infty(V) - c_C}{\tau_{c_C}}, \tag{2.5f}$$

$$\frac{dq_{sAHP}}{dt} = \frac{q_\infty(V) - q_{sAHP}}{\tau_{q_{sAHP}}}, \tag{2.5g}$$

$$\frac{d[Ca^{2+}]_i}{dt} = -\nu I_{Ca} - \frac{[Ca^{2+}]_i}{\tau_{Ca}}, \tag{2.5h}$$

here, $m_\infty(V)$, $h_\infty(V)$, $n_\infty(V)$, $p_\infty(V)$, $a_\infty(V)$, $b_\infty(V)$, $z_\infty(V)$, $r_\infty(V)$, $c_\infty(V)$, $q_\infty([Ca^{2+}]_i)$, and $d_\infty([Ca^{2+}]_i)$ are the steady-state activation/deactivation functions. ϕ is a scaling parameter. $\tau_{h_{Na}}(V)$, $\tau_{n_{Kdr}}(V)$, $\tau_{b_{Kdr}}$, $\tau_{r_{Ca}}$, τ_{c_C}, and $\tau_{q_{sAHP}}$ are the time constants. The steady-state activation/deactivation functions are given by:

$$m_\infty(V) = \frac{1}{1 + e^{-(V-\theta_m)/\sigma_m}}, \tag{2.6a}$$

$$n_\infty(V) = \frac{1}{1 + e^{-(V-\theta_n)/\sigma_n}}, \tag{2.6b}$$

$$h_\infty(V) = \frac{1}{1 + e^{-(V-\theta_h)/\sigma_h}}, \tag{2.6c}$$

$$p_\infty(V) = \frac{1}{1 + e^{-(V-\theta_p)/\sigma_p}}, \tag{2.6d}$$

$$b_\infty(V) = \frac{1}{1 + e^{-(V-\theta_b)/\sigma_b}}, \tag{2.6e}$$

$$z_\infty(V) = \frac{1}{1 + e^{-(V-\theta_z)/\sigma_z}}, \tag{2.6f}$$

$$a_\infty(V) = \frac{1}{1 + e^{-(V-\theta_a)/\sigma_a}}, \tag{2.6g}$$

$$r_\infty(V) = \frac{1}{1 + e^{-(V-\theta_r)/\sigma_r}}, \tag{2.6h}$$

$$c_\infty(V) = \frac{1}{1 + e^{-(V-\theta_c)/\sigma_c}}, \tag{2.6i}$$

$$d_\infty([Ca^{2+}]_i) = \frac{1}{(1 + a_c/[Ca^{2+}]_i)}, \tag{2.6j}$$

$$q_\infty([Ca^{2+}]_i) = \frac{1}{1 + (a_q^4/[Ca^{2+}]_i^4)}, \tag{2.6k}$$

here, a_c, a_q, θ_i, σ_i for $i \in \{m, n, h, p, b, z, a, r, c\}$ are the model parameters. The voltage dependent time constants $\tau_{h_{Na}}(V)$ and $\tau_{n_{Kdr}}(V)$ are given by

$$\tau_{h_{Na}}(V) = 1 + \frac{7.5}{1 + e^{-(V-\theta_{ht})/\sigma_{ht}}}, \tag{2.7a}$$

$$\tau_{n_{Kdr}}(V) = 1 + \frac{5}{1 + e^{-(V-\theta_{nt})/\sigma_{nt}}}, \tag{2.7b}$$

where θ_{ht}, θ_{nt}, σ_{ht}, and σ_{nt} are model parameters.

Throughout this chapter, we used the following numerical values for the unknown model parameters (26): $C = 1$ F/cm^2, $g_L = 0.05$ mS/cm^2, $V_L = -70$ mV, $\nu = 0.13$ cm^2/(ms $\times$ A), g_{Na} = 35 mS/cm^2, V_{Na} = 55 mV, g_{NaP} = 0.4 mS/cm^2, g_{Kdr} = 6.0 mS/cm^2, $V_K = -90$ mV, g_A = 1.4 mS/cm^2, g_M = 0.5 mS/cm^2, $g_{Ca} = 0.08$ mS/cm^2, g_C = 10 mS/cm^2, V_{Ca} = 120 mV, and g_{sAHP} = 5 mS/cm^2, θ_m = -30 mV, σ_m = 9.5 mV, θ_h = -45 mV, σ_h = -7 mV, θ_{ht} = -40.5 mV, σ_{ht} = -6 mV, ϕ = 10, θ_P = -47 mV, σ_P = 3 mV, θ_n = -35 mV, σ_n = 10 mV, θ_{nt} = -27 mV, σ_{nt} = -15 mV, θ_a = -50 mV, σ_a = 20 mV, θ_b = -80 mV, σ_b = -6 mV, θ_z = -39 mV, σ_z = 5 mV, θ_r = -20 mV, σ_r = 10 mV, τ_r = 1 ms, θ_c = -30 mV, σ_c = 7 mV, θ_c = 2 ms, a_c = 6, τ_q = 450 ms, and a_q = 2.

Unless otherwise stated, we used the following initial conditions to simulate the Hodgkin-Huxley model for generating the synthetic data: $V_0 = -71.81327$ mV, $h_{Na0} = 0.98786$, $n_{Kdr0} = 0.02457$, $b_{KA0} = 0.203517$, $u_{KM0} = 0.00141$, $r_{Ca0} = 0.005507$, $[Ca]_{i0} = 0.000787$, $c_{C0} = 0.002486$, $q_{Ca0} = 0.0$.

2.2.5 *Network Training*

We formulated the following optimization problem to train a set of network weights θ:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta), \tag{2.8}$$

where the loss function $\mathcal{L}(\theta)$ is given by

$$\mathcal{L}(\theta) = \frac{1}{N_P} \sum_{k=0}^{N_P} (\vec{x}(k) - \hat{x}(k|\theta))^T (\vec{x}(k) - \hat{x}(k|\theta)). \tag{2.9}$$

Here N_P represents the length of horizon over which the predictions are made, $\vec{x}(k)$ is the known state vector at time step k, and $\hat{x}(k|\theta)$ is the neural network's prediction of the state vector at time k, given θ.

To solve the optimization problem in Equations 2.8 and 2.9, we used the standard supervised backpropagation learning algorithm (15; 16; 17) along with the Adaptive Moment Estimation (Adam) method (66). The Adam method is a first-order gradient-based optimization algorithm and uses lower-order moments of the gradients between layers to optimize a stochastic objective function.

Given the network parameter $\theta^{(i)}$ and the loss function $\mathcal{L}(\theta)$, where i represents the algorithm's training iteration, the parameter update is given by Reference (66)

$$m_\theta^{(i+1)} \leftarrow \beta_1 m_\theta^{(i)} + (1-\beta_1)\nabla_\theta \mathcal{L}^{(i)}, \tag{2.10}$$

$$v_\theta^{(i+1)} \leftarrow \beta_2 m_\theta^{(i)} + (1-\beta_2)(\nabla_\theta \mathcal{L}^{(i)})^2, \tag{2.11}$$

$$\hat{m}_\theta = \frac{m_\theta^{(i+1)}}{1-(\beta_1)^{i+1}}, \tag{2.12}$$

$$\hat{v}_\theta = \frac{v_\theta^{(i+1)}}{1-(\beta_2)^{(i+1)}}, \tag{2.13}$$

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \frac{\hat{m}_\theta}{\sqrt{\hat{v}_\theta} + \epsilon}, \tag{2.14}$$

where m_θ is the first moment of the weights in a layer, v_θ is the second moment of the weights in a layer, η is the learning rate, β_1 and β_2 are the exponential decay rates for the moment estimates, ∇ is the differential gradient operator, and ϵ is a small scalar term to help numerical stability. Throughout this work, we used $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\eta = 0.001$ (66).

It should be noted that there is a trade-off between the predictive time-horizon of deep LSTM neural network and the computational cost involved in training the network over the predictive horizon. As the predictive horizon increases, the computational cost of training the network over that horizon increases significantly for an equivalent number of examples. To keep the computational tractability in our simulations, all networks with long predictive horizons (i.e., $N_P = 50, 100, 200$) were trained for 200 epochs except the one-step predictive network, which was trained for 1000 epochs.

For all training sets throughout this chapter, we used the validation to training data ratio as 1/32. We set the minibatch size for training to 32. We performed all the training and computation in the TensorFlow computational framework on a discrete server running CentOS 7 with twin Nvidia GTX 1080Ti GPUs equipped with 11 Gb of VRAM.

2.3 SIMULATION RESULTS

In this section, we present our simulation results on predicting the multi-timescale spiking dynamics exhibited by hippocampal CA1 pyramidal neurons over a long time-horizon using our developed deep LSTM neural network architecture described in Section 2.2. We trained 4 LSTM networks for making one timestep prediction (equivalently, 0.1 ms), 50 timesteps prediction (equivalently, 5 ms), 100 timesteps prediction (equivalently, 10 ms), and 200 timesteps prediction (equivalently, 20 ms). Figure 2.6 shows the training and validation loss for these 4 LSTM networks.

Using the iterative approach described in Section 2.2.1, we simulated the LSTM networks over 500 ms of time duration under different initial conditions and stimulating input currents between three different regimes of dynamical responses ("Regular Spiking" ($I \in [2.3, 3.0]$ nA), "Irregular Bursting" ($I \in [0.79, 2.3)$ nA), and "Regular Bursting" ($I \in [0.24, 0.79)$) nA) and compared the predicted state trajectories with the Hodgkin-Huxley model.

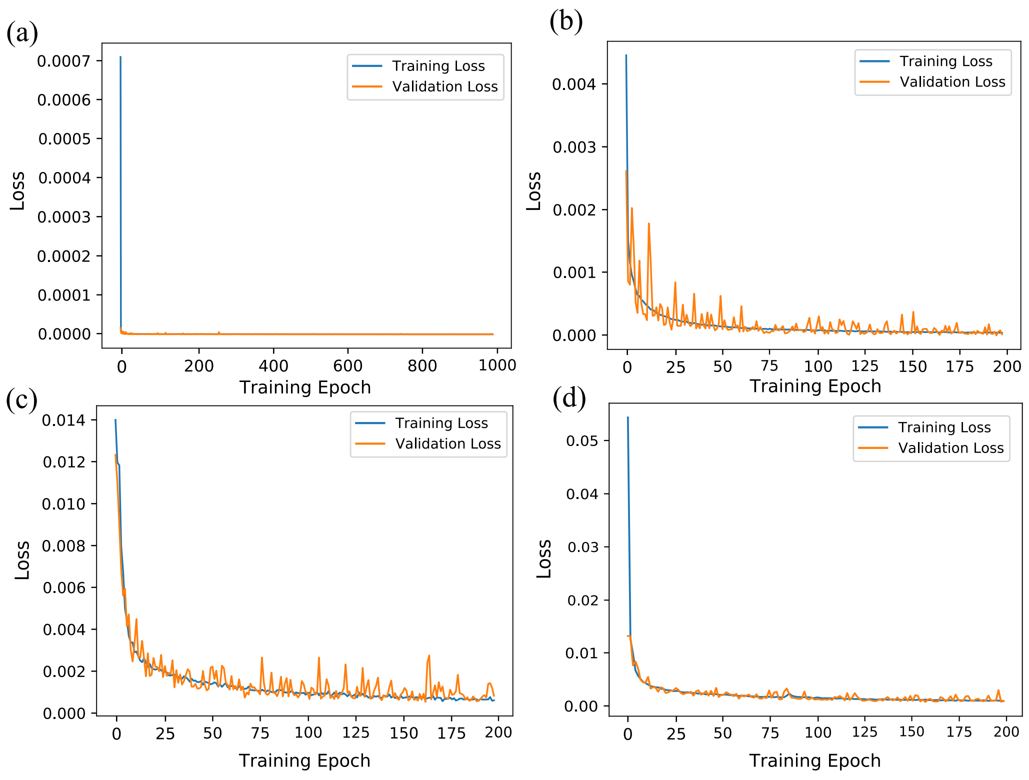


Figure 2.6: Training and validation loss for the deep long short term memory (LSTM) neural network with multi-timestep predictive horizon. (**a**) 1 timestep predictive horizon. (**b**) 50 timesteps predictive horizon. (**c**) 100 timesteps predictive horizon. (**d**) 200 timesteps predictive horizon.

2.3.1 *Regular Spiking*

In this section, we demonstrate the efficacy of our trained deep LSTM neural network over the range of external current between 2.3 nA and 3 nA in predicting the regular spiking dynamics shown by the biophysiological Hodgkin-Huxley model of CA1 pyramidal neuron in response to the external current $I \geq 2.3$ nA. In this range, we observe firing rates between approximately 165 Hz and 187 Hz. For clarity, we here show our results only for the membrane potential traces. We provide the complete set of simulation results on the LSTM network performance in predicting the dynamics of all the 9 states of the Hodgkin-Huxley model in Appendix 2.4.1 (see Figures 2.13–2.17).

Figure 2.7 shows a comparison of the membrane potential traces simulated using the Hodgkin-Huxley model and the 4 different predictive horizons of the LSTM network (i.e., 1 timestep, 50 timesteps, 100 timesteps and 200 timesteps, which we represent as $N_p = 1, 50, 100, 200$) for the external stimulating current $I = 3.0$ nA. Note that all the simulations are performed using the same initial condition as provided in Appendix 2.2.4. Since our LSTM network uses the initial sequence of outputs of appropriate predictive horizon (i.e., $N_p = 1, 50, 100, 200$) from the Hodgkin-Huxley model to make future time predictions, the LSTM network predictions (shown by dashed red line) start after 0.1 ms, 5 ms, 10 ms, and 20 ms in Figure 2.7a–d, respectively.

As shown in Figure 2.7, the iterative prediction of the membrane potential traces by the LSTM network did not differ significantly over a short time horizon (up to 200 ms) for $N_p = 1, 50, 100, 200$, but it significantly improved afterward with the increased predictive horizon of the LSTM network (i.e., $N_p = 1$ to $N_p = 200$). In particular, the LSTM network performance significantly improved in predicting the timing of the occurrence of spikes, but the magnitude of the membrane potential traces during spikes degraded as we increased N_p from 1 to 200. For clarity, we also computed the time-averaged root mean squared error (RMSE) of the membrane potential traces between the Hodgkin-Huxley model and the LSTM network for $N_p = 1, 50, 100, 200$ over 500 ms of simulation time. Figure 2.8a shows that the time-averaged RMSE decreased consistently with the increased predictive horizon of the LSTM network.

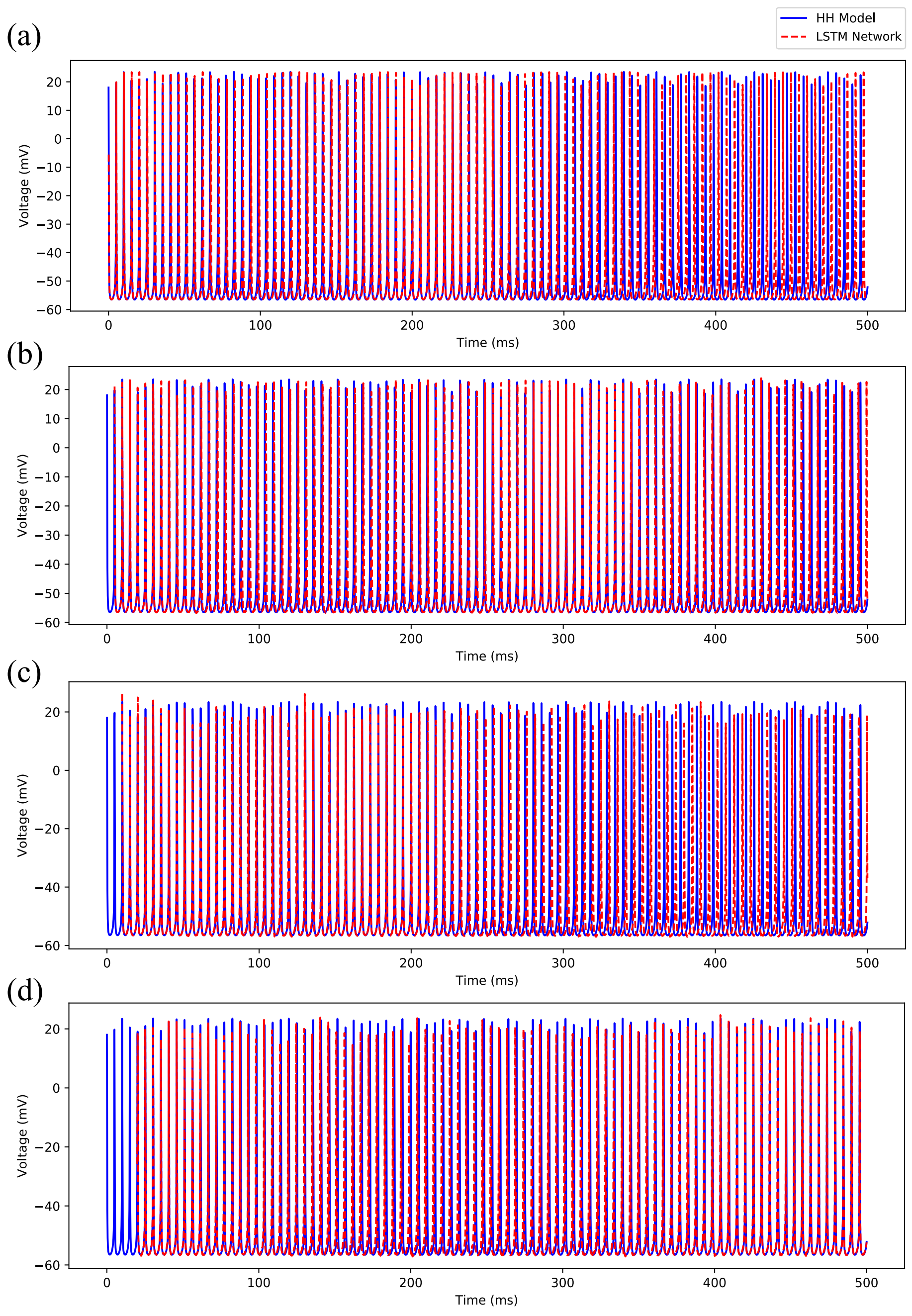


FIGURE 2.7: Comparison of predicted membrane potential traces by the deep LSTM neural network ("LSTM Network") to the regular spiking pattern exhibited by the Hodgkin-Huxley model ("HH Model") in response to the external stimulating current $I = 3.0$ nA. (**a**) Prediction using 1 timestep predictive LSTM network ($N_p = 1$). (**b**) Prediction using 50 timesteps predictive LSTM network ($N_p = 50$). (**c**) Prediction using 100 timesteps predictive LSTM network ($N_p = 100$). (**d**) Prediction using 200 timesteps predictive LSTM network ($N_p = 200$).

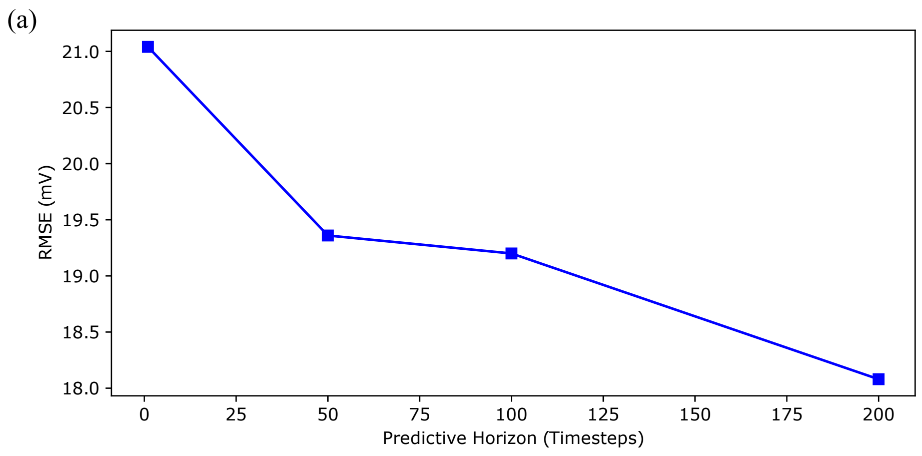
Overall, these results show that our LSTM network with a longer predictive horizon prefers to capture temporal correlations more accurately over the amplitude while an LSTM network with a shorter predictive horizon prefers to capture the amplitude more accurately over the temporal correlations.

To systematically evaluate whether the designed LSTM networks provide reasonable predictions of the membrane potential traces of the regular spiking dynamics across the range of external input currents between 2.3 nA and 3.0 nA, we performed simulations for 50 random stimulating currents drawn from a Uniform distribution $\mathcal{U}(2.3, 3.0)$. For each stimulating current, we chose 100 initial conditions drawn randomly from the maximum and minimum range of the Hodgkin-Huxley state variables (Note that the network was not trained over this wide range of initial conditions). Figure 2.8b shows the LSTM network performance, represented in terms of the root mean squared error vs time over 5000 realizations, for $N_p = 1, 50, 100, 200$. As shown in this figure, the root mean squared error decreased with the increased predictive horizon of the LSTM network for all time, which is consistent with the result shown in Figure 2.8a.

In conclusion, these results suggest that our deep LSTM neural network with a longer predictive horizon feature can predict the regular (periodic) spiking patterns exhibited by hippocampal CA1 pyramidal neurons with high accuracy over a long-time horizon.

2.3.2 *Irregular Bursting*

In this section, we demonstrate the efficacy of our trained deep LSTM neural network over the range of external current between 0.79 nA and 2.3 nA in predicting the irregular bursting dynamics shown by the biophysiological Hodgkin-Huxley model of CA1 pyramidal neuron in response to the external current $I \in [0.79, 2.3)$ nA. In this range, we observe firing rates between approximately 53 Hz and 164 Hz. For clarity, we here show our results only for the membrane potential traces. We provide the complete set of simulation results on the LSTM network performance in predicting the dynamics of all the 9 states of the Hodgkin-Huxley model in Appendix 2.4.2 (see Figures 2.18–2.22).



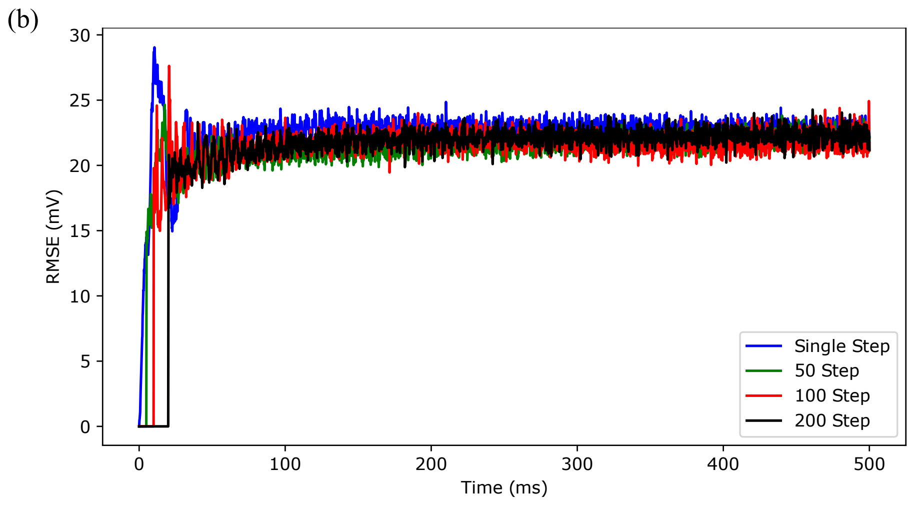


FIGURE 2.8: The effect of the length of predictive horizon of the deep LSTM neural network on the accuracy of regular spiking patterns prediction. (**a**) shows the time-averaged root mean squared error (RMSE) versus predictive horizon of the LSTM network ($N_p = 1, 50, 100, 200$) for the simulation results shown in Figure 2.7; (**b**) shows the RMSE versus simulation time for 5000 independent realizations, drawn from the predicted membrane potential trajectories of 50 randomly selected stimulating currents from a Uniform distribution $\mathcal{U}(2.3, 3.0)$ and 100 random initial conditions for each stimulating current.

Figure 2.9 shows a comparison of the membrane potential traces simulated using the Hodgkin-Huxley model and the 4 different predictive horizons of the LSTM network (i.e., $N_p = 1, 50, 100, 200$) for the external stimulating current $I = 1.5$ nA. Note that all the simulations are performed using the initial condition used for $I = 3.0$ nA in Figure 2.7. Since our LSTM network uses the initial sequence of outputs of appropriate prediction horizon (i.e., $N_p = 1, 50, 100, 200$) from the Hodgkin-Huxley model to make future time predictions, the LSTM network predictions (shown by dashed red line) start after 0.1 ms, 5 ms, 10 ms, and 20 ms in Figure 2.9a–d, respectively.

As shown in Figure 2.9, the LSTM performance significantly improved in predicting the timing of the occurrence of spikes up to 100 ms with the increased predictive horizon of the LSTM network from $N_p = 1$ to $N_p = 200$, but the performance degraded in capturing the magnitude of the membrane potentials during spiking with an increased value of N_p. Although the time-averaged root mean squared error of the membrane potential traces between the Hodgkin-Huxley model and the LSTM network for $N_p = 1, 50, 100, 200$ showed an improved performance with the increased value of N_p (see Figure 2.10a), none of the LSTM networks showed a reasonable prediction of the timing of the occurrence of spikes in this regime beyond 100 ms of the time-horizon.

To systematically evaluate whether the designed LSTM networks provide reasonable predictions of the membrane potential traces of the regular spiking dynamics across the range of external input currents between 0.79 nA and 2.3 nA, we performed simulations for 50 random stimulating currents drawn from a Uniform distribution $\mathcal{U}(0.79, 2.3)$. For each stimulating current, we chose 100 initial conditions drawn randomly from the maximum and minimum range of the Hodgkin-Huxley state variables (note that the network was not trained over this wide range of initial conditions). Figure 2.10b shows the LSTM network performance, represented in terms of the root mean squared error vs time over 5000 realizations, for $N_p = 1, 50, 100, 200$. As shown in this figure, the root mean squared error decreased with the increased predictive horizon of the LSTM network for all time, which is consistent with the result shown in Figure 2.10a.

In conclusion, these results suggest that our deep LSTM neural network with a longer predictive horizon feature can predict the irregular bursting patterns exhibited by hippocampal CA1 pyramidal neurons with high accuracy over only a short-time horizon.

2.3.3 *Regular Bursting*

In this section, we demonstrate the efficacy of our trained deep LSTM neural network over the range of external current between 0.24 nA and 0.79 nA in predicting the regular bursting dynamics shown by the biophysiological Hodgkin-Huxley model of CA1 pyramidal neuron in response to the external current $I \in [0.24, 0.79)$ nA. In this range, we observe firing rates between approximately 8 Hz and 52 Hz. For clarity, we here show our results only for the membrane potential traces. We provide the complete set of simulation results on the LSTM network performance in predicting the dynamics of all the 9 states of the Hodgkin-Huxley model in Appendix 2.4.3 (see Figures 2.23–2.27).

Figure 2.11 shows a comparison of the membrane potential traces simulated using the Hodgkin-Huxley model and the 4 different predictive horizons of the LSTM network (i.e., $N_p = 1, 50, 100, 200$) for the external stimulating current $I = 0.5$ nA. Note that all the simulations are performed using the initial condition used for $I = 3.0$ nA in Figure 2.7. Since our LSTM network uses the initial sequence of outputs of appropriate prediction horizon (i.e., $N_p = 1, 50, 100, 200$) from the Hodgkin-Huxley model to make future time predictions, the LSTM network predictions (shown by dashed red line) start after 0.1 ms, 5 ms, 10 ms and 20 ms in Figure 2.11a–d, respectively.

By analyzing the results shown in Figure 2.11, we found that the LSTM network performance in predicting the timing of spikes during bursts as well as tracking the subthreshold potential improved significantly from $N_p = 1$ to $N_p = 200$, but the performance substantially degraded in capturing the magnitude of the membrane potentials during spiking. In conclusion, the 200 timesteps prediction horizon based LSTM network (see Figure 2.11d) predicts the temporal dynamics with reasonable accuracy over the first 300 ms of prediction.

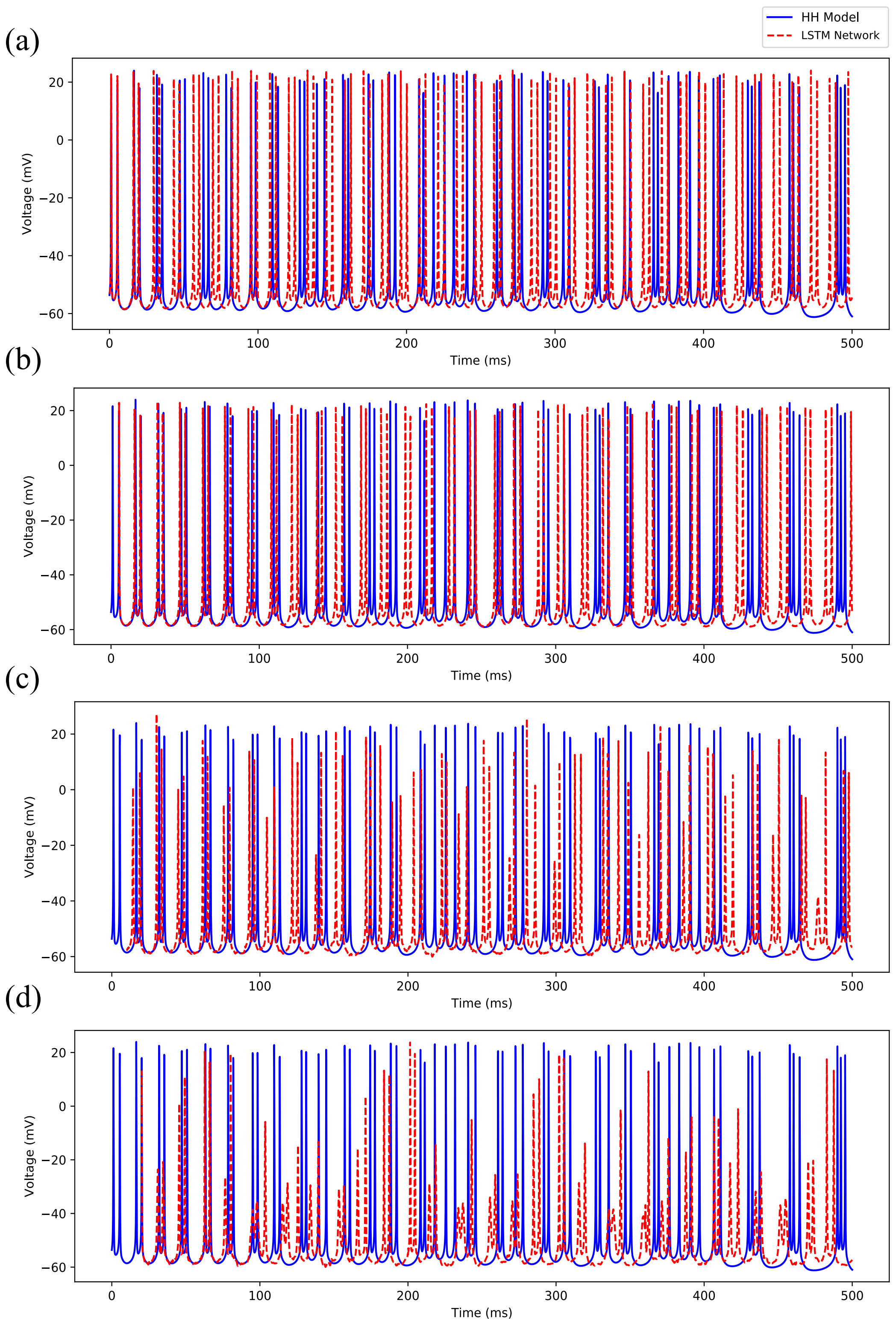


FIGURE 2.9: Comparison of predicted membrane potential traces by the deep LSTM neural network ("LSTM Network") to the irregular bursting spiking patterns exhibited by the Hodgkin-Huxley model ("HH Model") in response to the external stimulating current $I = 1.5$ nA. (**a**) Prediction using 1 timestep predictive LSTM network ($N_p = 1$). (**b**) Prediction using 50 timesteps predictive LSTM network ($N_p = 50$). (**c**) Prediction using 100 timesteps predictive LSTM network ($N_p = 100$). (**d**) Prediction using 200 timesteps predictive LSTM network ($N_p = 200$).

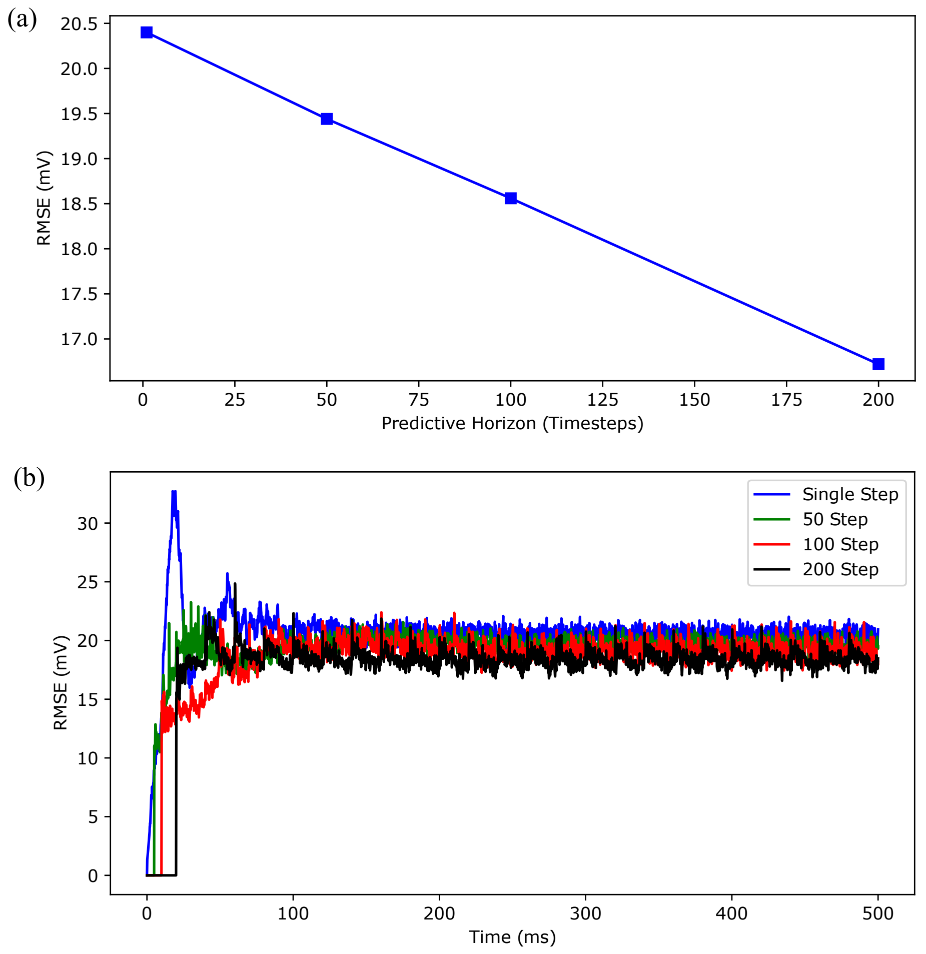


FIGURE 2.10: The effect of the prediction horizon of the deep LSTM neural network on the accuracy of irregular bursting dynamics prediction. (**a**) shows the time-averaged root mean squared error (RMSE) versus predictive horizon of the LSTM network ($N_p = 1, 50, 100, 200$) for the simulation results shown in Figure 2.9. (**b**) shows the RMSE versus simulation time for 5000 independent realizations, drawn from the predicted membrane potential trajectories of 50 randomly selected stimulating currents from a Uniform distribution $\mathcal{U}(0.79, 2.3)$ and 100 random initial conditions for each stimulating current.

Figure 2.12a shows the time-averaged root mean squared error of the membrane potential traces between the Hodgkin-Huxley model and the LSTM network for $N_p = 1, 50, 100, 200$. As noted in this figure, the root mean squared error decreased substantially between 100 timesteps and 200 timesteps prediction horizon compared to the regimes of regular spiking (Figure 2.8a) and irregular bursting (Figure 2.10a), which indicates that a longer predictive horizon based LSTM network is necessary to capture the two different timescales (i.e., short intraburst spiking intervals and long interburst subthreshold intervals) presented in these dynamics.

Figure 2.12b shows the LSTM networks performances, represented in terms of the root mean squared error vs time over 5000 realizations, for N_p = 1, 100, and 200 timestep prediction horizon LSTM network. As shown in this figure, the root mean squared error decreased with the increased predictive horizon of the LSTM network for all time, which is consistent with the result shown in Figure 2.12a. Note that we have excluded the simulation result for $N_p = 50$ as we found out in our detailed analysis that the trained LSTM network for $N_p = 50$ led to instability in predicting spiking responses for some of the initial condition values in this regime. The reason for this may be that the network may not have seen these initial conditions during the training.

2.4 SIMULATION RESULTS ON FULL STATE PREDICTIONS OF HODGKIN-HUXLEY MODEL

In Section 2.3, we showed our simulation results only for the membrane potential traces. Here, we provide the simulation results for all the 9 states of the Hodgkin-Huxley model of CA1 pyramidal neuron (HHCA1Py) predicted by the deep LSTM neural network over a long-time horizon and show the comparison between these predictions and the simulated dynamics from HHCA1Py.

2.4.1 *Regular Spiking*

In this section, we show the simulation results on predicting the dynamics of all the 9 states of HHCA1Py over a long-time horizon using the deep LSTM neural

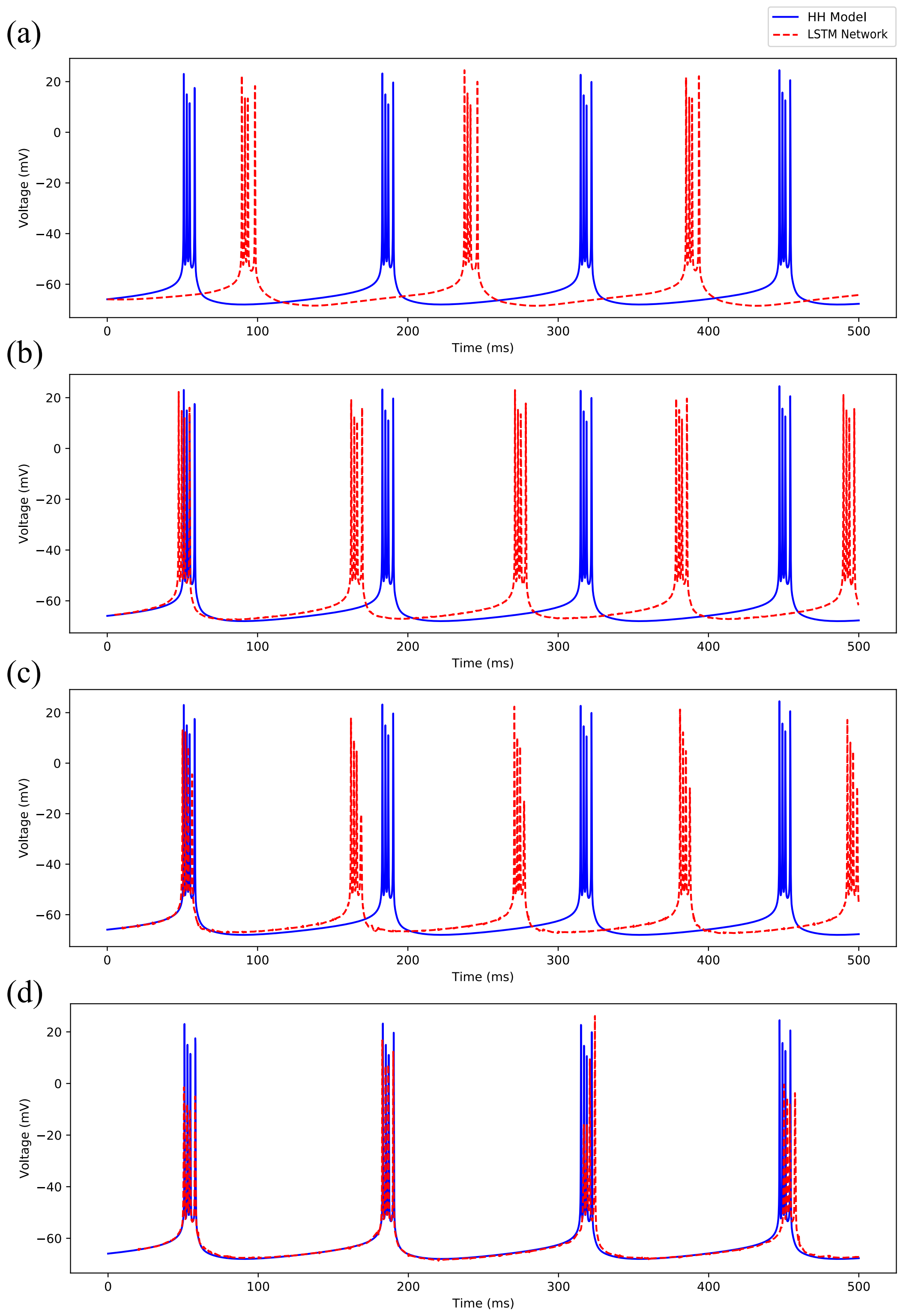


Figure 2.11: Comparison of predicted membrane potential traces by the LSTM network ("NN Prediction") to the irregular bursting spiking patterns exhibited by the Hodgkin-Huxley model ("HH Model") in response to the external stimulating current $I = 0.5$ nA. (**a**) Prediction using 1 timestep predictive LSTM network ($N_p = 1$); (**b**) Prediction using 50 timesteps predictive LSTM network ($N_p = 50$); (**c**) Prediction using 100 timesteps predictive LSTM network ($N_p = 100$); (**d**) Prediction using 200 timesteps predictive LSTM network ($N_p = 200$).

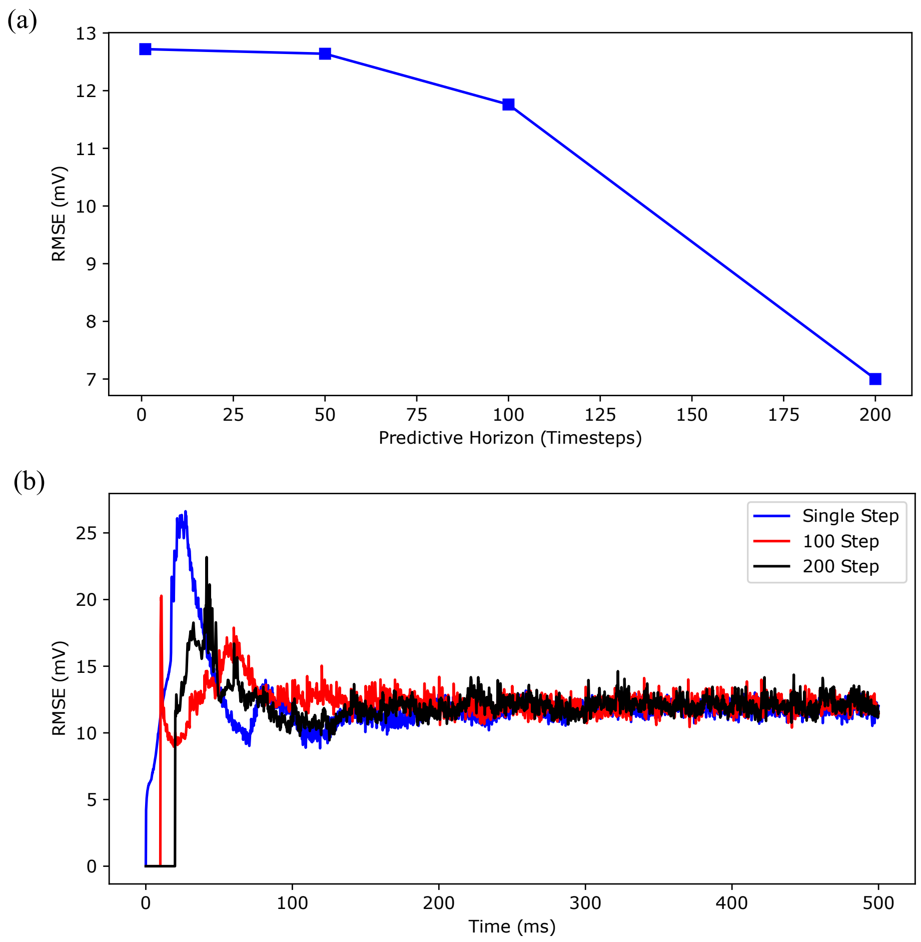


FIGURE 2.12: The effect of the prediction horizon of the multi-timestep LSTM network on the accuracy of regular bursting dynamics prediction. (**a**) shows the time-averaged root mean squared error (RMSE) versus predictive horizon of the LSTM network ($N_p = 1, 50, 100, 200$) for the simulation results shown in Figure 2.11. (**b**) shows the RMSE versus simulation time for 5000 independent realizations, drawn from the predicted membrane potential trajectories of 50 randomly selected stimulating currents from a Uniform distribution $\mathcal{U}(0.24, 0.79)$ and 100 random initial conditions for each stimulating current.

network for the regular periodic spiking regime ($I \in [2.3, 3.0]$ nA). Figures 2.13–2.16 show the comparison between the state's dynamics simulated using the Hodgkin-Huxley model and the deep LSTM neural network model developed for 1 timestep, 50 timesteps, 100 timesteps, and 200 timesteps (equivalently, $N_p = 1, 50, 100, 200$) predictive horizon, respectively.

As shown in these figures, the performance of the deep LSTM neural network model in predicting state dynamics significantly improved with the increased predictive horizon of the LSTM network (i.e., $N_p = 1$ to $N_p = 200$) for all the states except q_{sAHP} for which we found that the magnitude is comparable to the numerical precision of the performed simulations. Figure 2.17 shows the root mean squared error between the states of HHCA1Py and the deep LSTM neural network model as a function of simulation time over 5000 random realizations, for $N_p = 1, 50, 100, 200$. These results show that the root mean squared error decreases from $N_p = 1$ to $N_p = 200$.

2.4.2 *Irregular Bursting*

In this section, we show the simulation results on predicting the dynamics of all the 9 states of HHCA1Py over a long-time horizon using the deep LSTM neural network for the irregular bursting regime ($I \in [0.79, 2.3)$ nA). Figures 2.18–2.21 show the comparison between the state's dynamics simulated using the Hodgkin-Huxley model and the deep LSTM neural network model developed for 1 timestep, 50 timesteps, 100 timesteps, and 200 timesteps (equivalently, $N_p = 1, 50, 100, 200$) predictive horizon, respectively.

As shown in these figures, the deep LSTM neural network model provides a reasonable prediction of the dynamics of all the states except q_{sAHP} over the initial 100 ms of simulations. Moreover, the prediction improved from $N_p = 1$ to $N_p = 200$, which is consistent with the results for the regular spiking regime (see Figures 2.13–2.17). We found that the magnitude of q_{sAHP} was comparable to the numerical precision of our simulations, which hindered the capability of the LSTM network in making a reasonable prediction for this state.

Figure 2.22 shows the root mean squared error between the states of HHCA1Py and the deep LSTM neural network model as a function of simulation time over 5000 random realizations, for $N_p = 1, 50, 100, 200$. As shown here, the root mean squared error decreased with the increased predictive horizon of the LSTM network (i.e., $N_p = 1$ to $N_p = 200$).

2.4.3 *Regular Bursting*

In this section, we show the simulation results on predicting the dynamics of all the 9 states of HHCA1Py over a long-time horizon using the deep LSTM neural network for the regular bursting regime ($I \in [0.24, 0.79)$ nA). Figures 2.23–2.26 show the comparison between the state's dynamics simulated using the Hodgkin-Huxley model and the deep LSTM neural network model developed for 1 timestep, 50 timesteps, 100 timesteps, and 200 timesteps (equivalently, $N_p = 1, 50, 100, 200$) predictive horizon, respectively.

As shown in these figures, the performance of the deep LSTM neural network model in predicting state dynamics significantly improved between 1 timestep predictive horizon (Figure 2.23) and 200 timesteps predictive horizon (Figure 2.26) across all the states except q_{sAHP} for the similar reason we provided for the regular spiking and irregular bursting regimes. More importantly, the LTSM network predicted the temporal correlations with high accuracy over the time-horizon of 300 ms for $N_p = 200$. The extrapolation of these results suggest that increasing the predictive horizon beyond $N_p = 200$ could improve the prediction beyond 300 ms of time-horizon.

In Figure 2.22, we show the root mean squared error between the states of HHCA1Py and the deep LSTM neural network model as a function of simulation time over 5000 random realizations, for $N_p = 1, 50, 100, 200$. As shown here, the root mean squared error decreased with the increased predictive horizon of the LSTM network (i.e., $N_p = 1$ to $N_p = 200$), which is consistent with the results of the regular spiking and irregular bursting regimes.

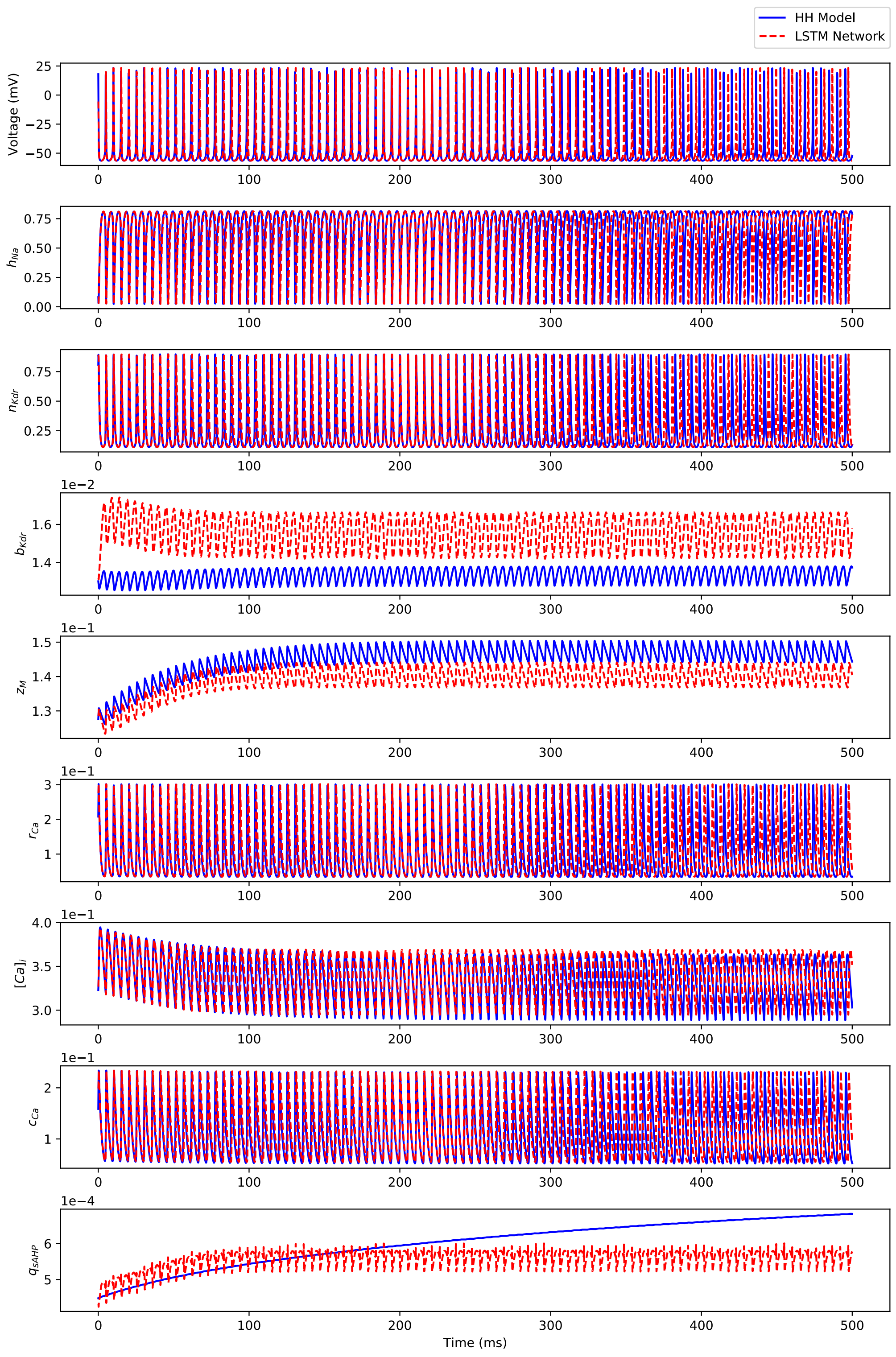


FIGURE 2.13: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 1 timestep predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 3.0$ nA.

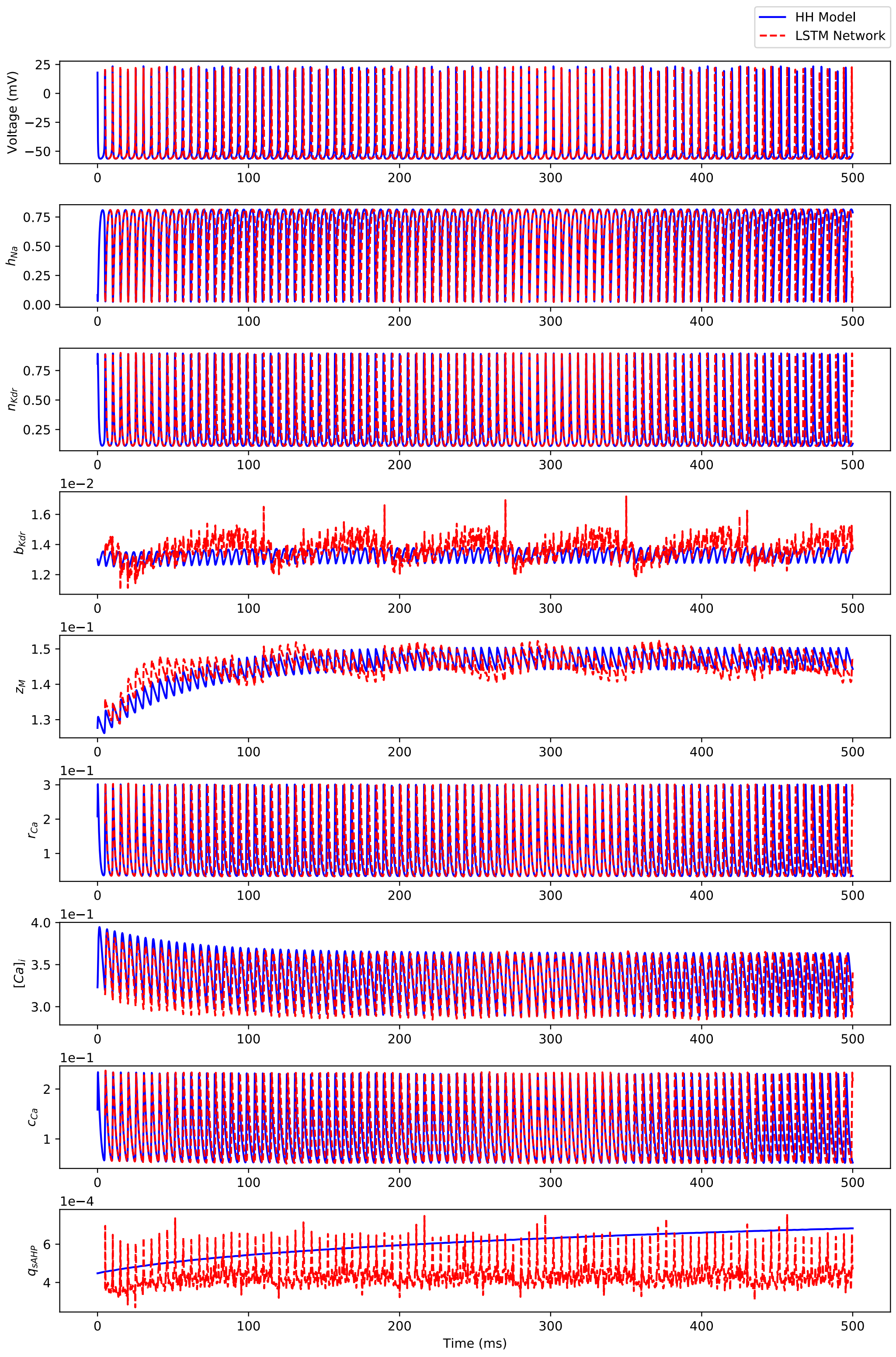


FIGURE 2.14: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 50 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 3.0$ nA.

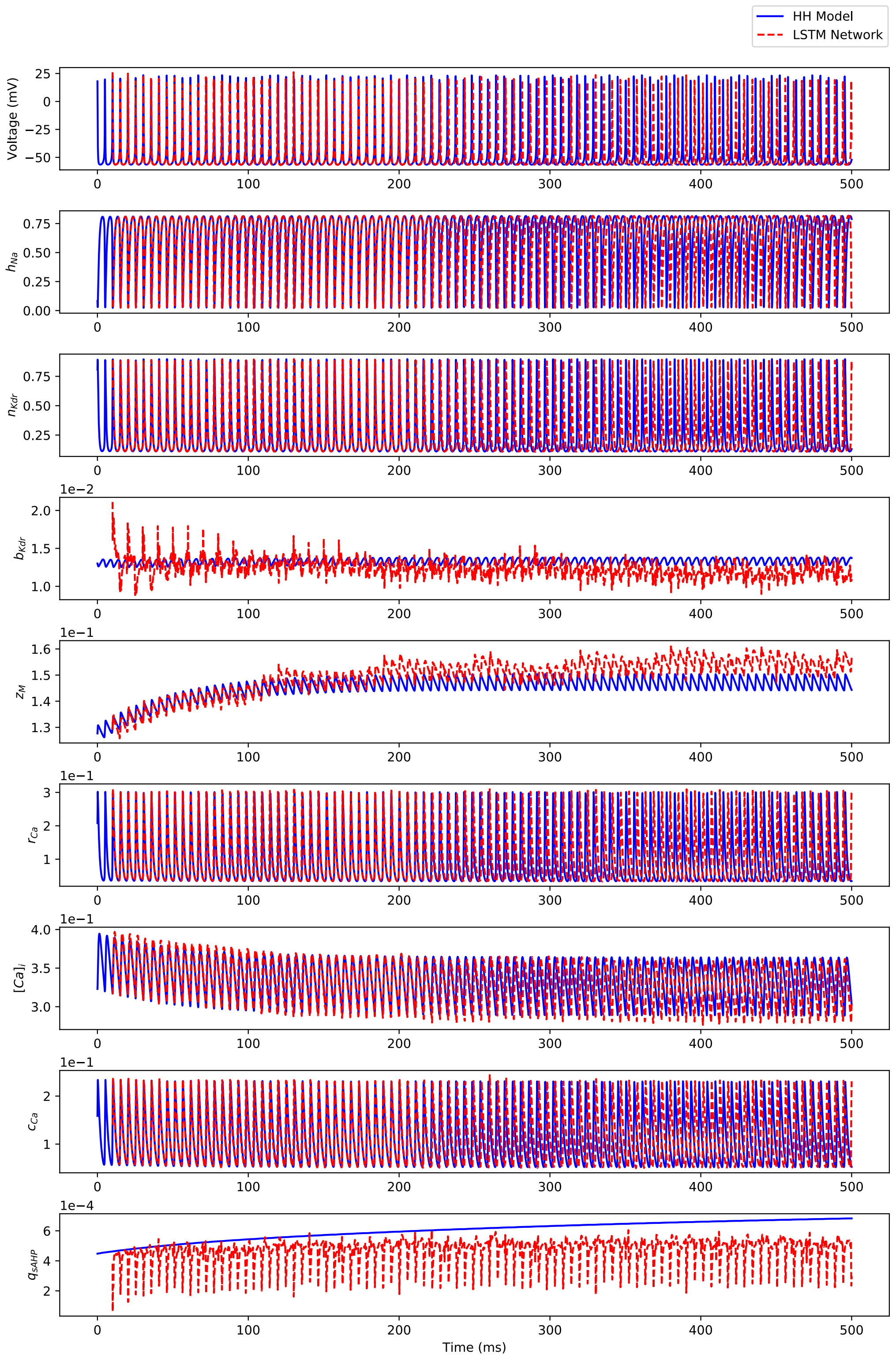


FIGURE 2.15: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 100 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 3.0$ nA.

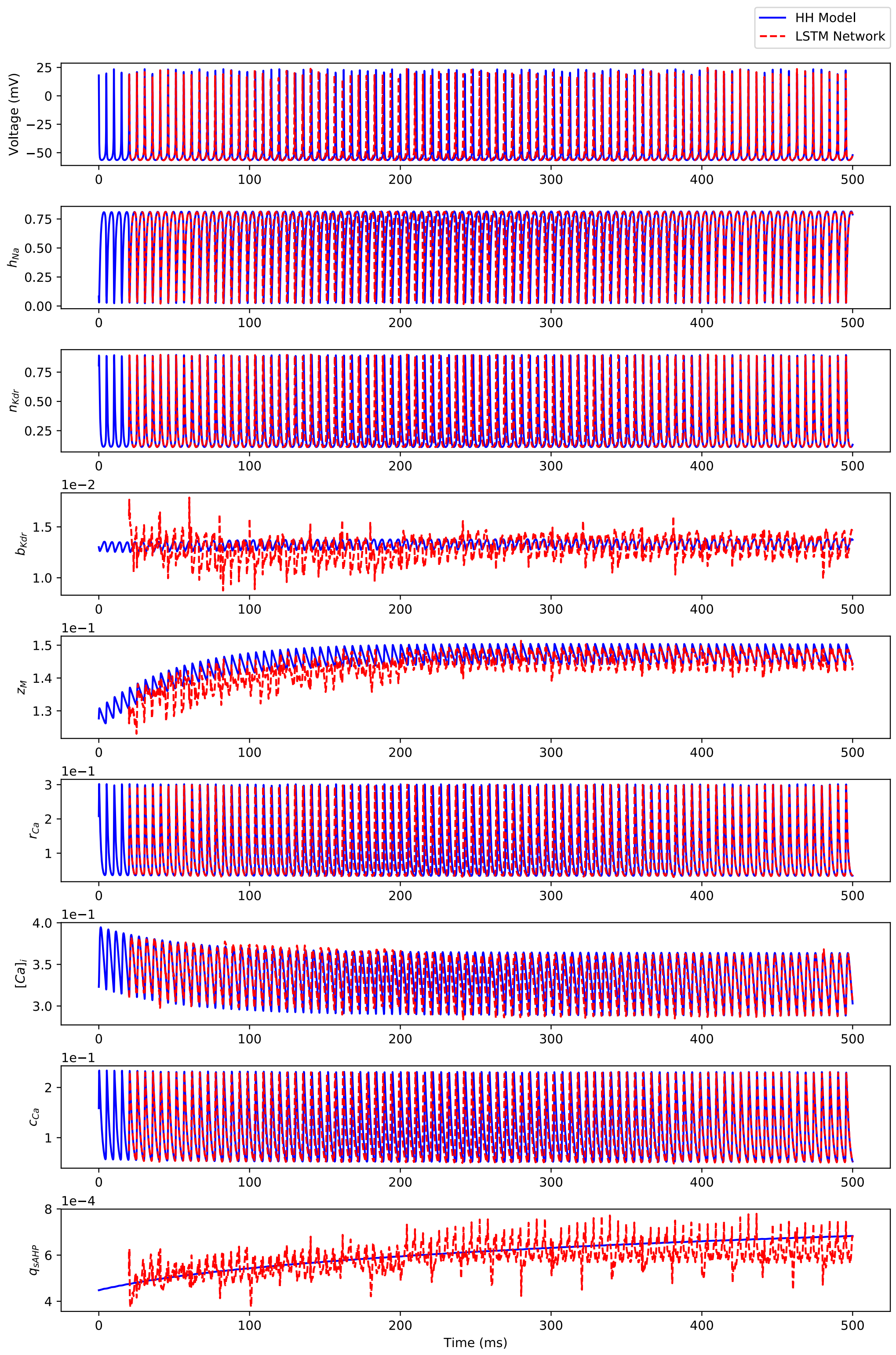


FIGURE 2.16: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 200 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 3.0$ nA.

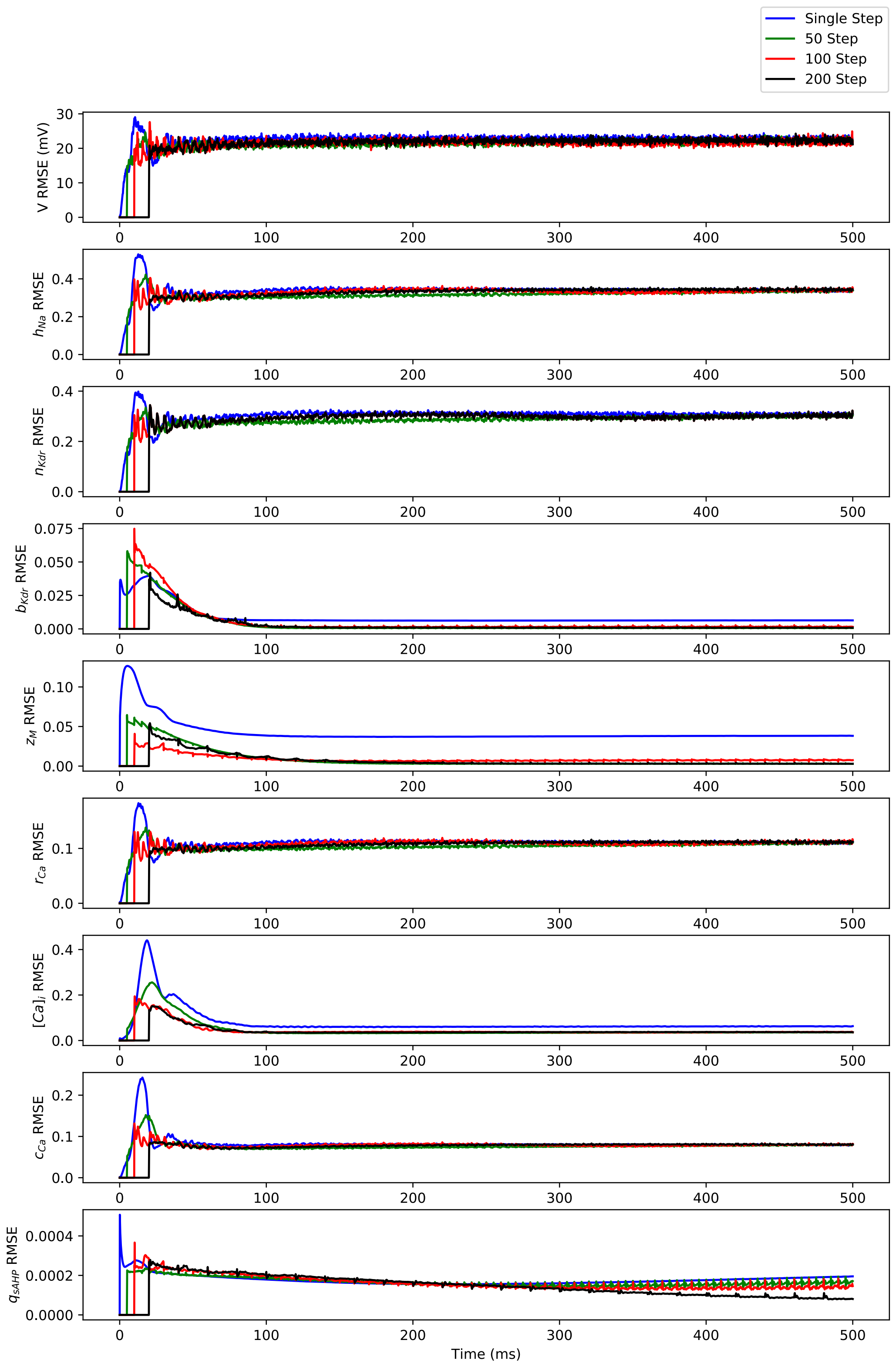


FIGURE 2.17: The root mean squared error (RMSE) versus simulation time for 5000 independent realizations, drawn from the predicted membrane potential trajectories of 50 randomly selected stimulating currents from a Uniform distribution $\mathcal{U}(2.3, 3.0)$ and 100 random initial conditions for each stimulating current.

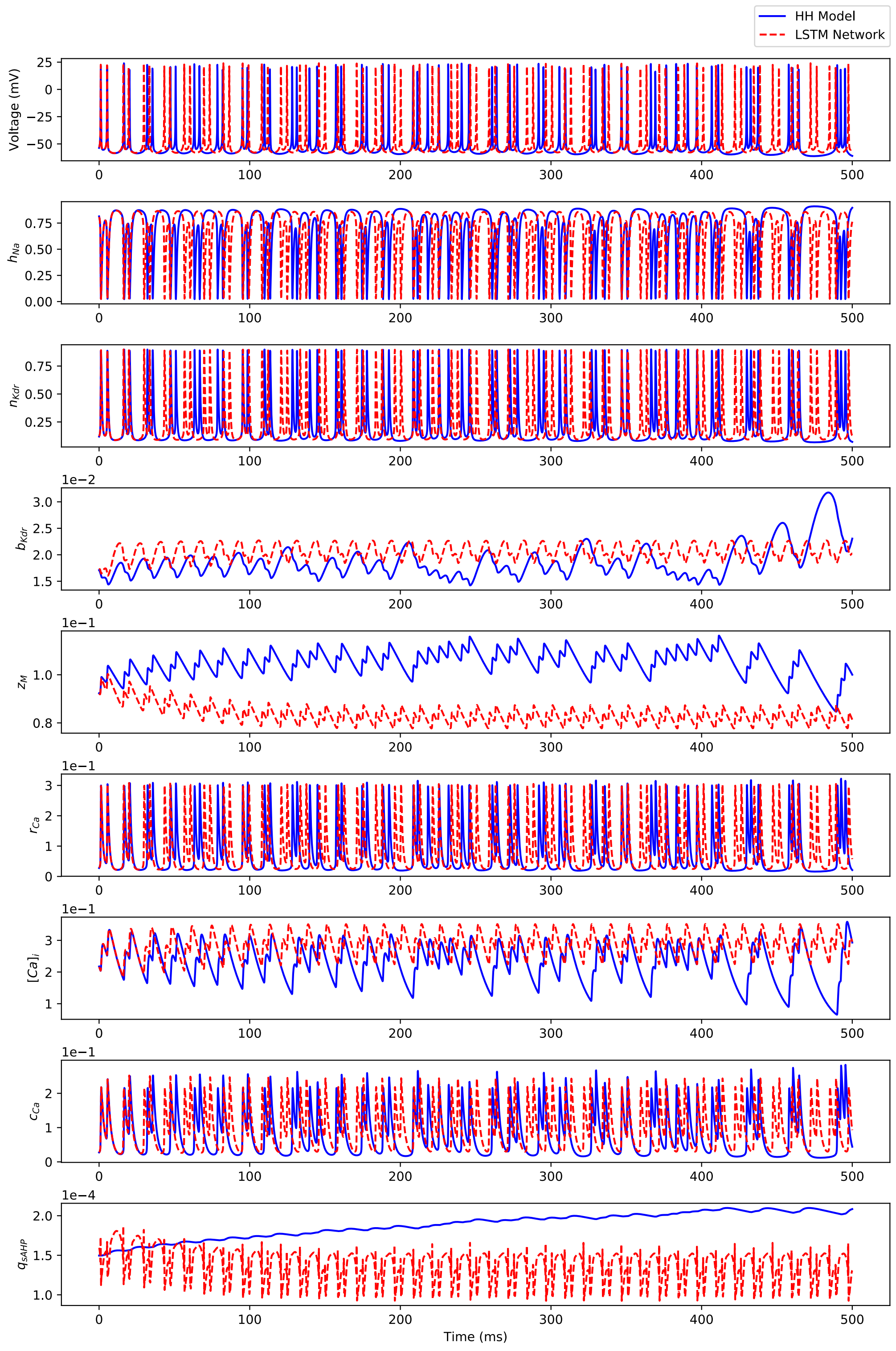


FIGURE 2.18: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 1 timestep predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 1.5$ nA.

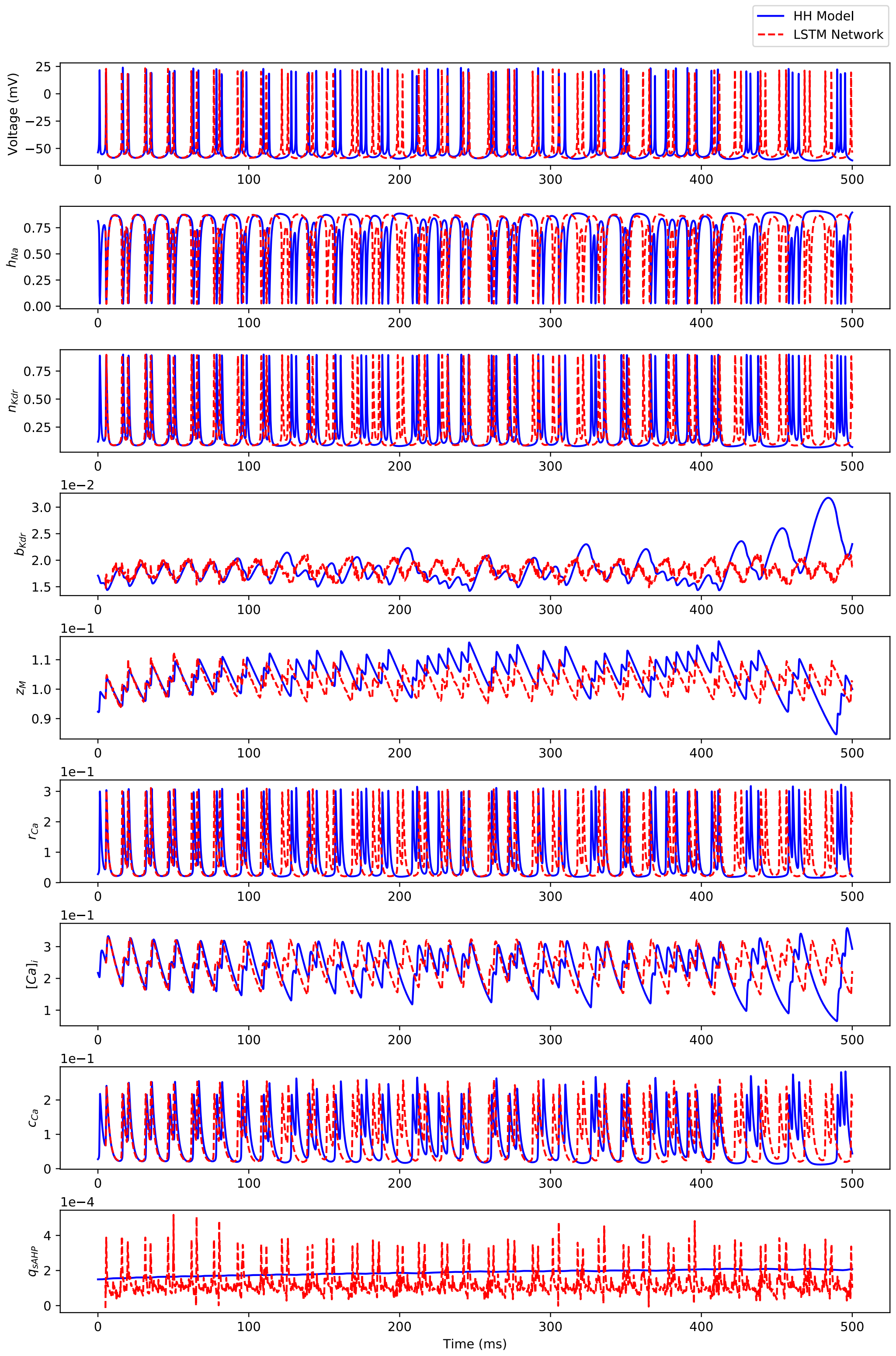


FIGURE 2.19: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 50 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 1.5$ nA.

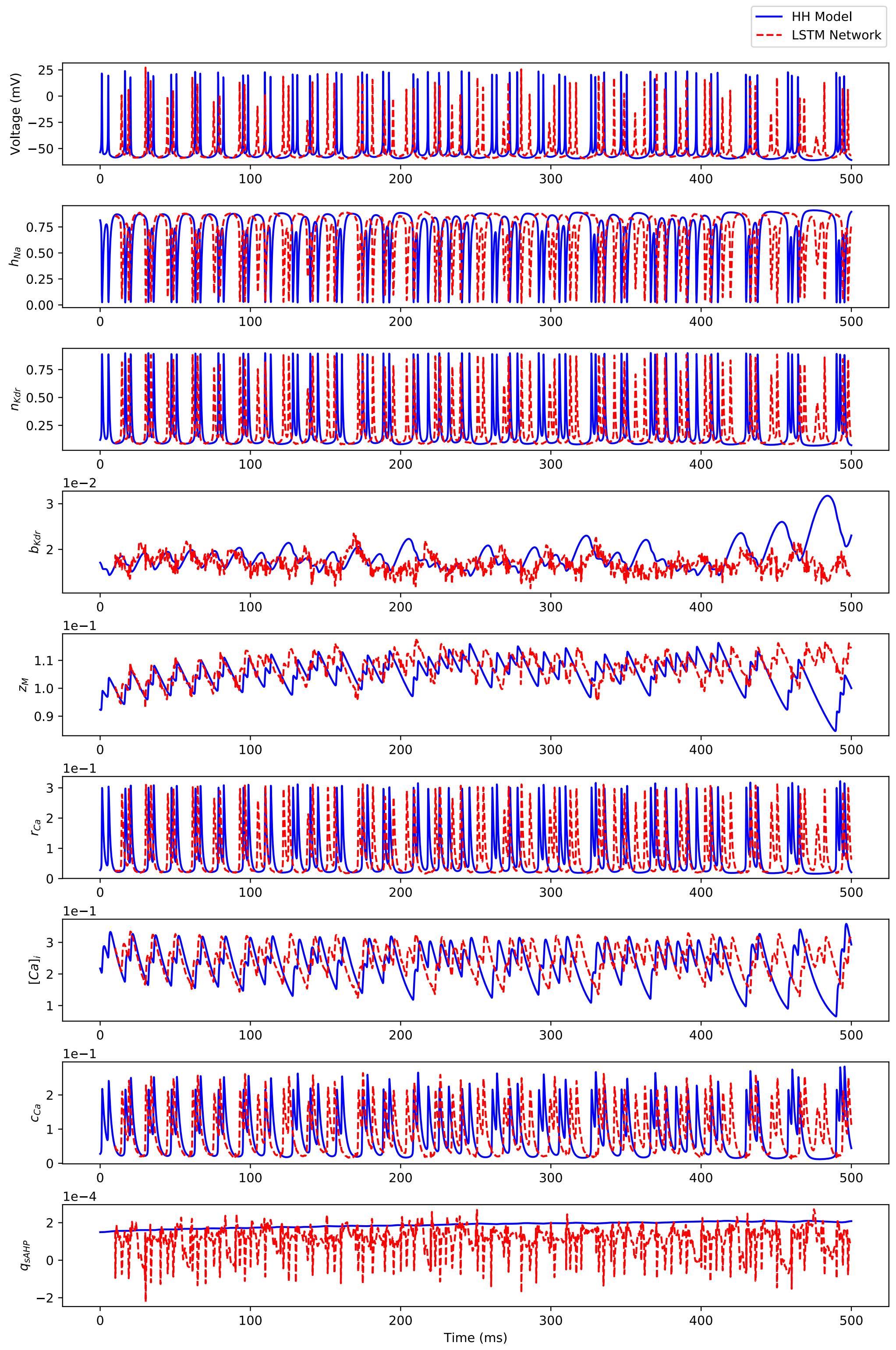


FIGURE 2.20: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 100 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 1.5$ nA.

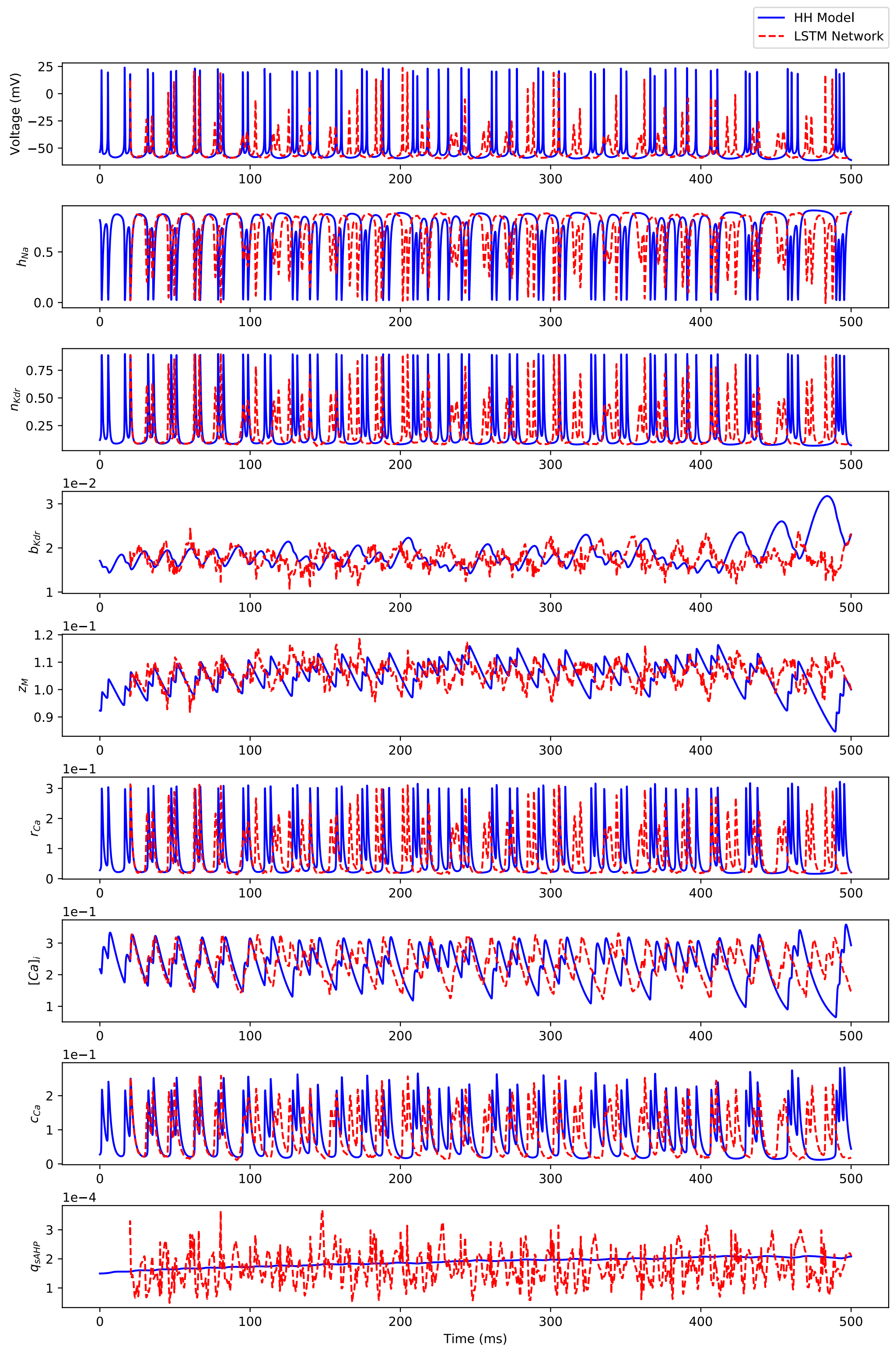


FIGURE 2.21: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 200 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 1.5$ nA.

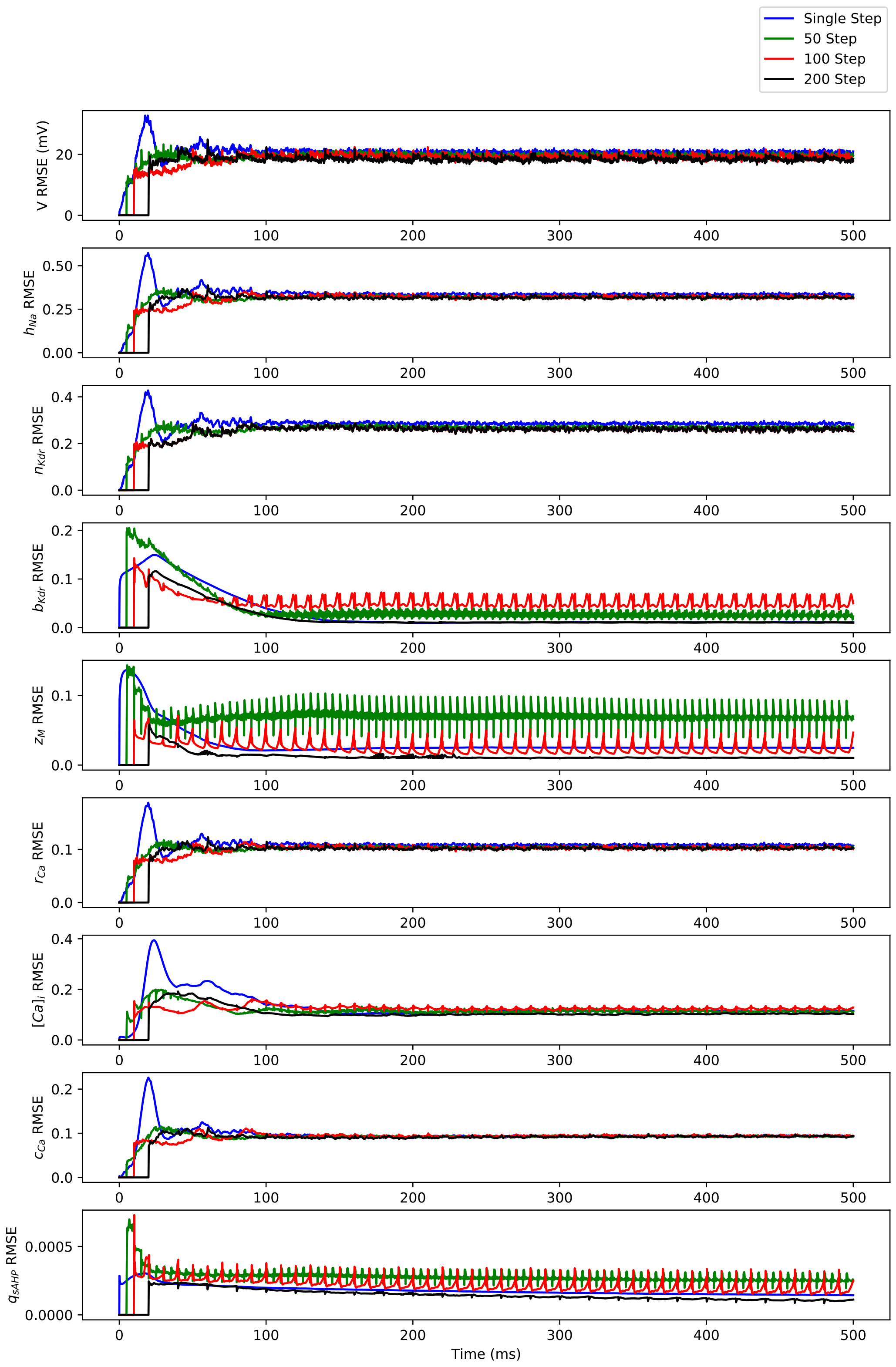


FIGURE 2.22: The root mean squared error (RMSE) versus simulation time for 5000 independent realizations, drawn from the predicted membrane potential trajectories of 50 randomly selected stimulating currents from a Uniform distribution $\mathcal{U}(0.79, 2.3)$ and 100 random initial conditions for each stimulating current.

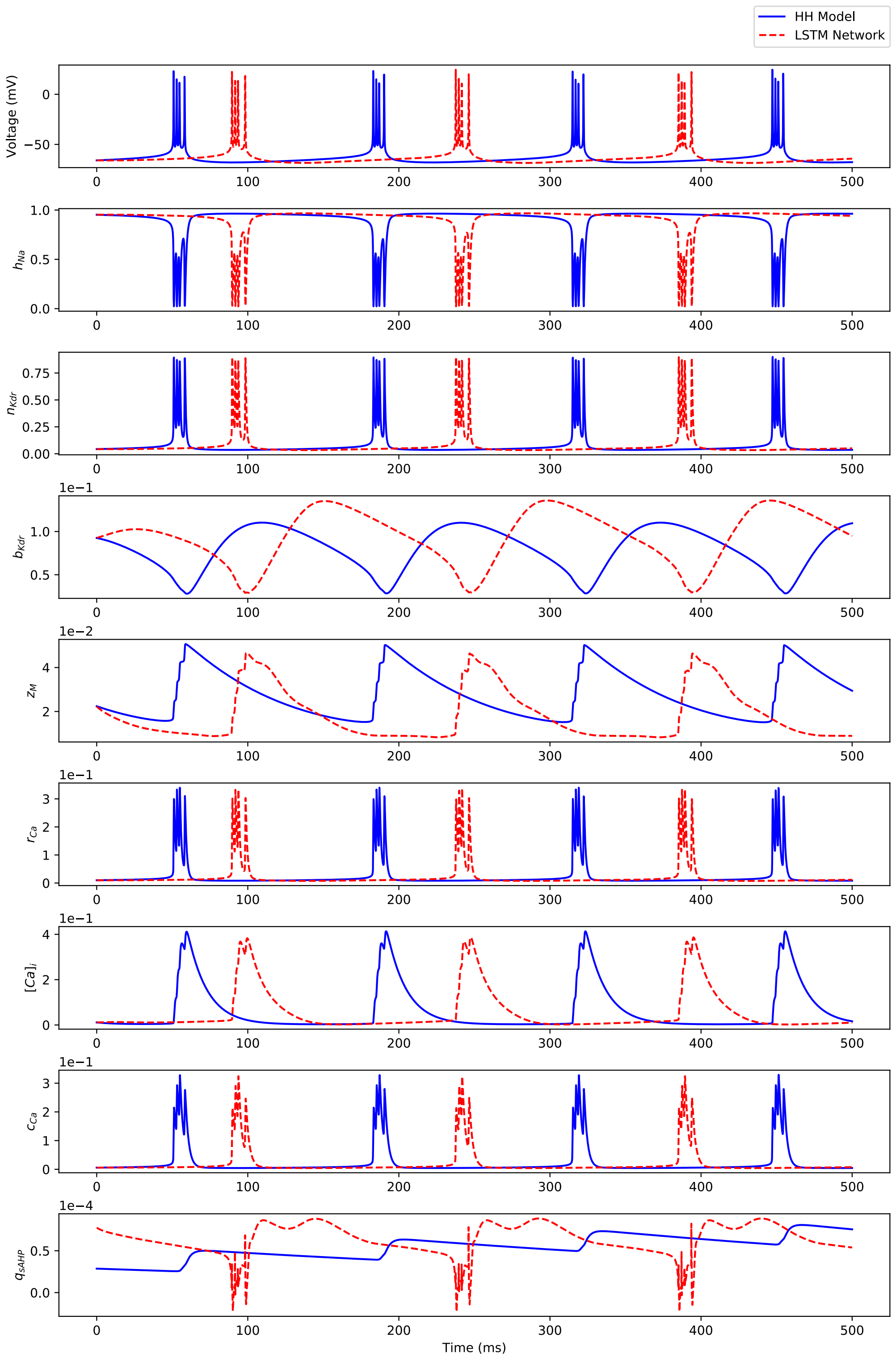


FIGURE 2.23: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 1 timestep predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 0.5$ nA.

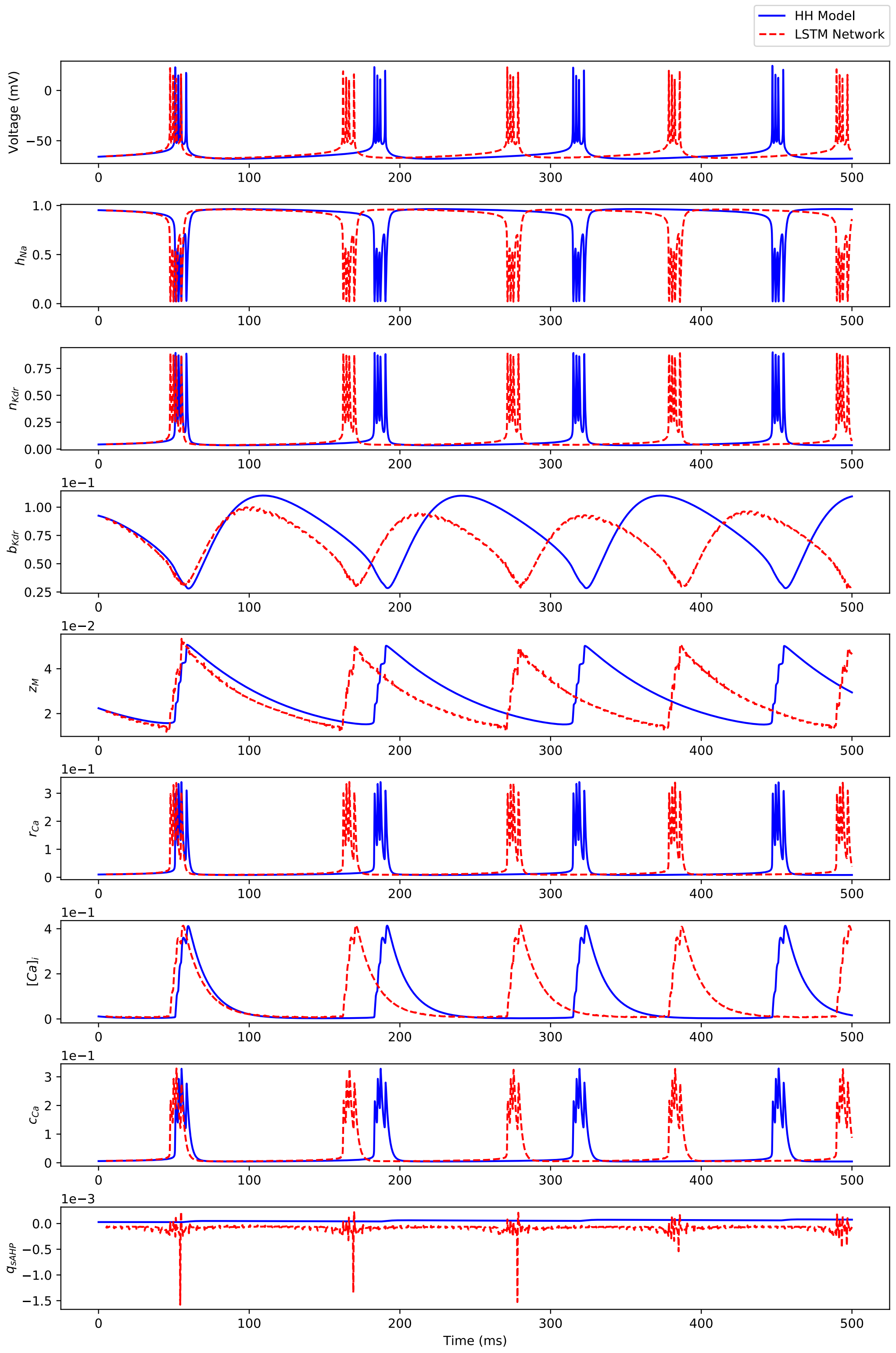


FIGURE 2.24: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 50 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 0.5$ nA.

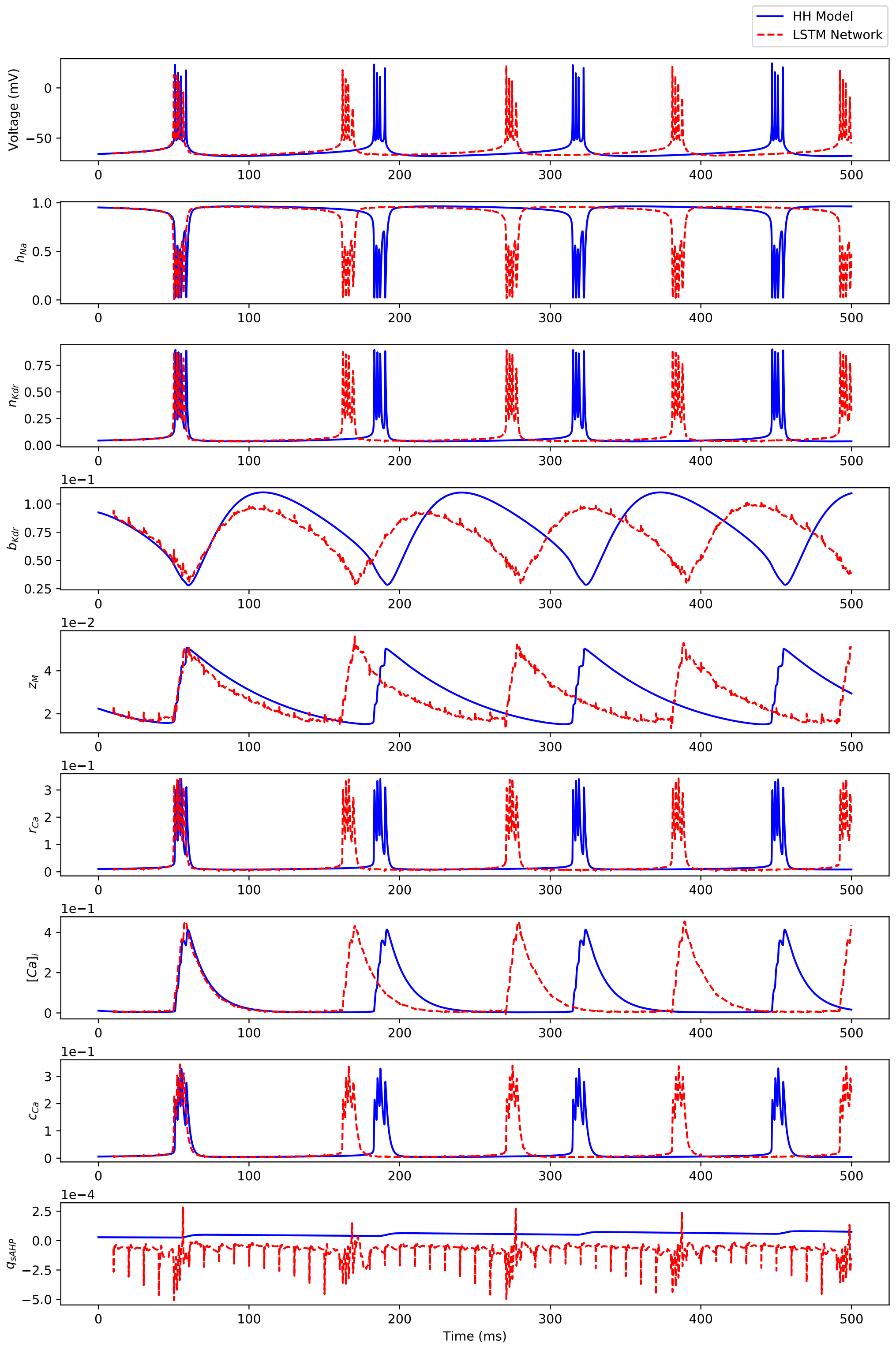


FIGURE 2.25: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 100 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 0.5$ nA.

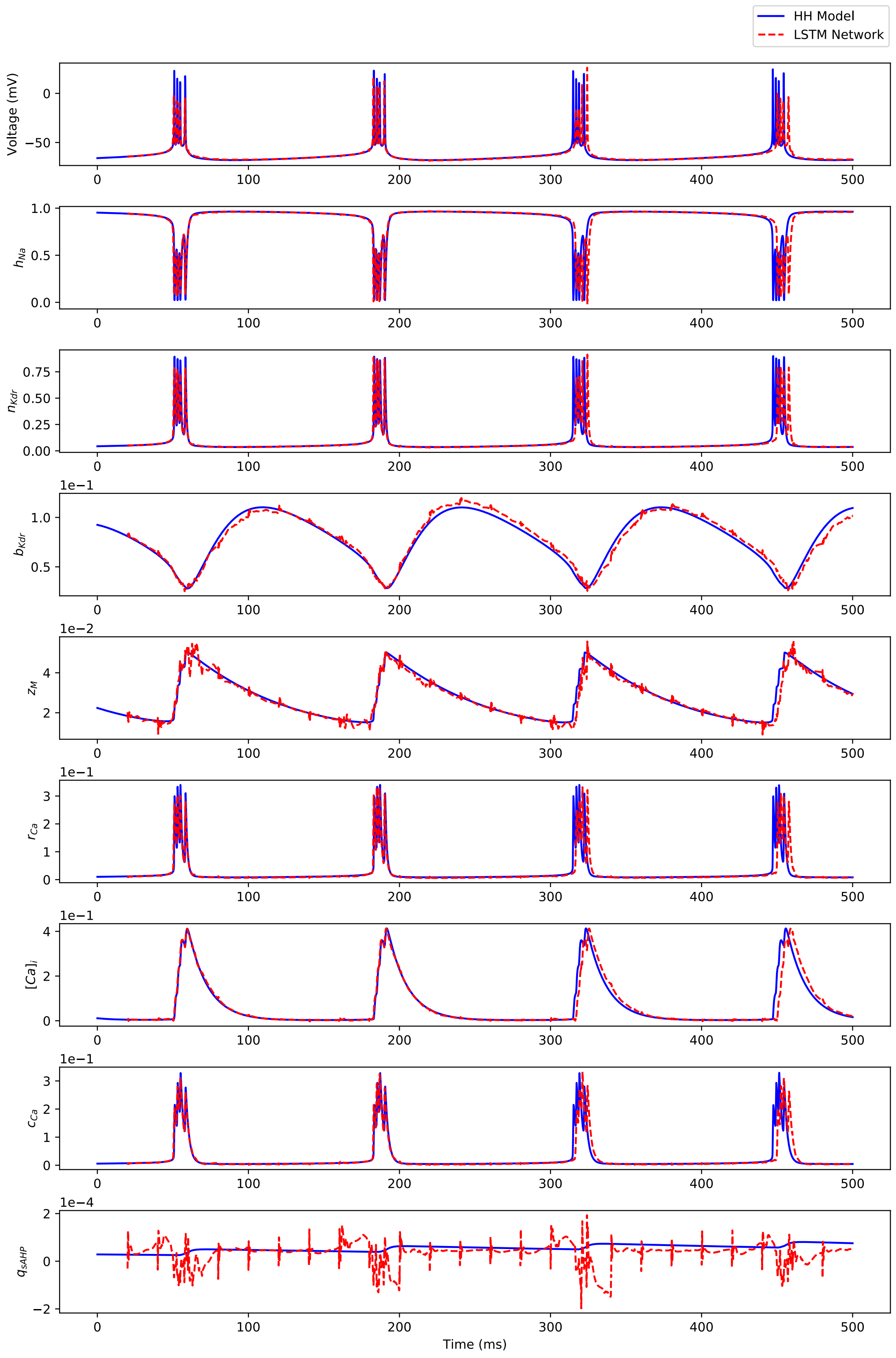


FIGURE 2.26: Comparison between the Hodgkin-Huxley model ("HH Model") states' dynamics and the iterative predictions of states' dynamics using the 200 timesteps predictive horizon-based deep LSTM neural network ("LSTM Network") in response to $I = 0.5$ nA.

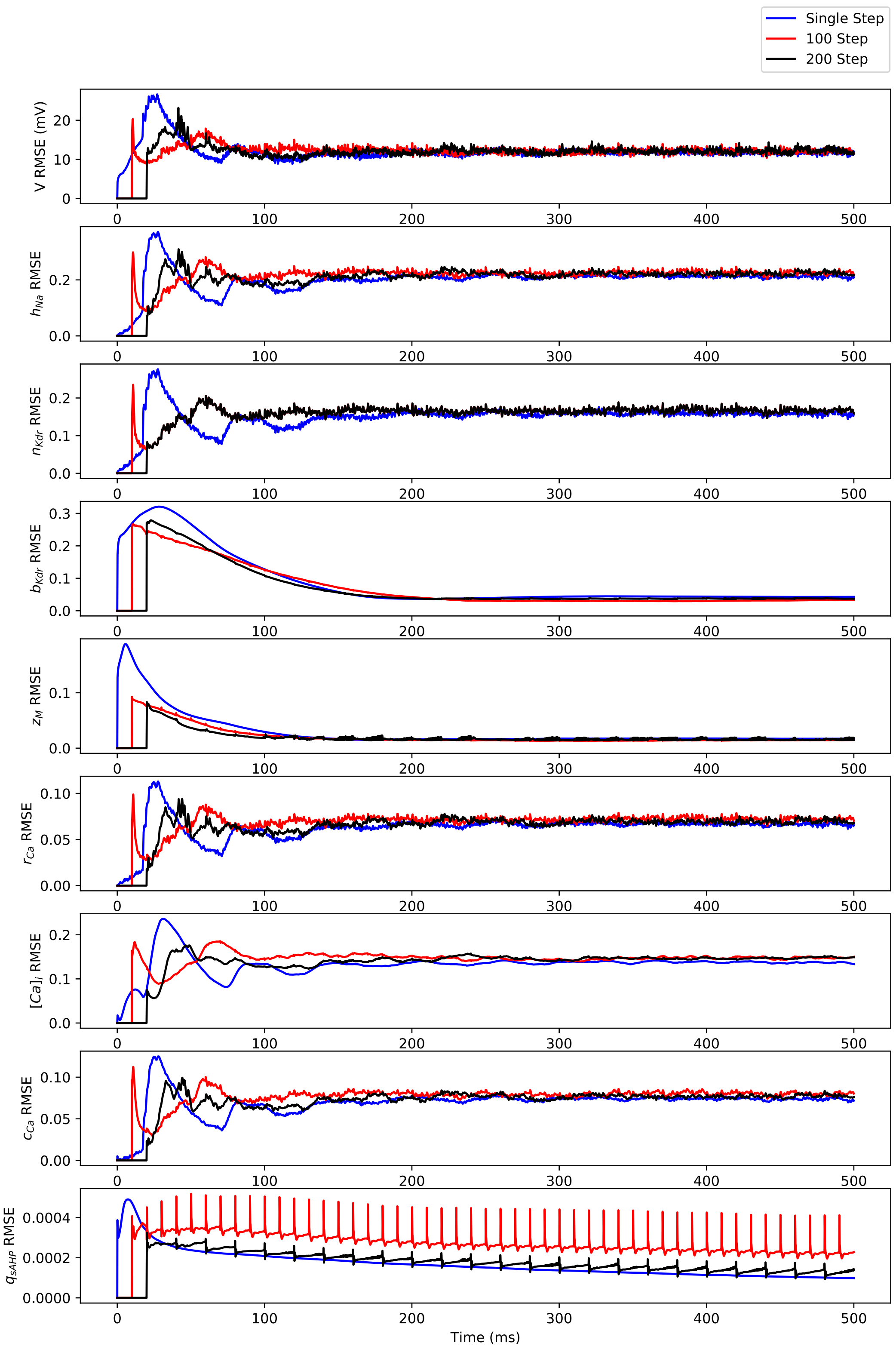


FIGURE 2.27: The root mean squared error (RMSE) versus simulation time for 5000 independent realizations, drawn from the predicted membrane potential trajectories of 50 randomly selected stimulating currents from a Uniform distribution $\mathcal{U}(0.24, 0.79)$ and 100 random initial conditions for each stimulating current.

2.5 DISCUSSION

In this chapter, we developed and presented a new data-driven long short-term memory (LSTM) based neural network (NN) architecture to predict the dynamical spiking patterns of single neurons. Compared to other LSTM-based NN architectures for forecasting dynamical systems behavior reported in the literature, our architecture incorporated a single dense feedforward output layer with an activation function and a reverse-order sequence-to-sequence mapping approach into traditional LSTM based neural networks to enable truly multi-timestep stable predictions of the dynamics over a long time-horizon. We demonstrated the efficacy of our architecture in predicting the multi-time scale dynamics of hippocampal CA1 pyramidal neurons and compared the predictions from our model with the ground truth synthetic data obtained from an experimentally validated biophysiological model of CA1 pyramidal neuron in the Hodgkin-Huxley formalism. Through simulations, we showed that (1) the presented architecture can learn multi-timescale dynamics; and (2) the predictive accuracy of the network increases with the increase in the predictive horizon of the LSTM network.

Our results for irregular bursting regime showed the limitation of the designed deep LSTM neural network architecture in making an accurate prediction of the timing of the occurrence of spikes over a long-time horizon compared to regular spiking and regular bursting regimes. A possible reason for this may be the architecture itself or the dataset used for training these networks, which requires further investigations by training the networks on the dataset explicitly generated from this regime. In addition, it has been shown that this regime of bursting exhibits chaotic dynamics (67), which may provide further explanation for the network's struggle to accurately predict this bursting behavior, as the system exhibits a high sensitivity to the initial conditions. Hybrid approaches that combine LSTM networks with mean stochastic models (MSM) have been explored for predictively modeling chaotic dynamical systems in Reference (50). However, this LSTM-MSM approach is limited to iterative single step prediction, and the application of this technique falls beyond the scope of this chapter.

Another limitation of our presented approach in modeling neuronal dynamics as currently constructed is the inclusion of the full state vector in both training and predictive evaluation. In experiment, it may be infeasible to have the entire state vector of the neuron measured for any given time. This should provide a valuable direction for future research, as partially observed systems or neuronal spike train recordings are much more feasible to measure in vivo or vitro and merit further consideration in combination with this approach.

In all dynamical regimes, our results showed a degraded performance of the deep LSTM neural network in predicting the amplitude of membrane potentials during the timing of the occurrence of spikes with the increased predictive horizon of the LSTM network. This issue may be related to the equally weighted norm-2 loss function used for training the networks. A further investigation is required by considering different loss functions, such as norm-1 or weighted norm-2, which we consider as our future work.

In addition, we make a note on the computational requirement of our presented approach. The computational cost of inference with an artificial neural network can effectively be boiled down to the number of multiplications and additions needed to complete a forward pass of information. The inference complexity for an LSTM is roughly $\mathcal{O}(d_i \cdot d \cdot h + d \cdot h^2)$, as described in Reference (50), where d_i is the dimensionality of each input, d is the number of inputs, and h is the dimensionality of the hidden states. Using this, we estimated the inference complexity of our LTSM network, represented by $\mathcal{O}(I)$, for the prediction horizon of 1, 50, 100, and 200. Using a single Nvidia GTX 1080Ti, we have also calculated the average computation time required for each of the predictive horizons used to predict 500 ms of state values from 1000 examples. We report these values in Table 2.1.

Although the data-driven approach developed in this chapter showed the ability of the designed LSTM-based neural network in learning multi-timescale dynamics, we note that the network struggles to accurately capture the dynamics of some state variables where the magnitude of the state variable is comparable to the numerical precision of our simulations. This can particularly be seen in Figures 2.14–2.16 and 2.26, where the network is not able to reconstruct the dynamics of the state variable q_{sAHP} with a reasonable accuracy. One possible way to alleviate this issue may be to increase

Table 2.1: Comparison of computational requirement for the iterative approach presented in this chapter.

N_P ($\Delta t = 0.1$ ms)	Predicted Time (ms)	Iterations	Computation Time (s)	$\mathcal{O}(I)$
1	500	5000	8.896	5.3×10^9
50	500	100	3.778	1.6×10^8
100	500	50	3.679	1.1×10^8
200	500	25	3.565	8.0×10^7

the tolerance of the numerical errors in simulations, which may increase the overall computational cost during training.

In conclusion, our results showed that a longer predictive horizon-based LSTM network can provide a more accurate prediction of multi-time scale dynamics, but at the expense of extensive offline training cost.

Chapter 3
Discovering Latent Dynamics Embedded In Large-Scale Neural Spiking Activity

3.1 Introduction

Neural populations encode information, at least partly, in the spatiotemporal patterns of spiking activity (68; 69; 70). Thus, understanding the spatiotemporal dynamics which generate these patterns is a critical step towards understanding neural coding. However, most of our current understanding of neural coding relies on the statistical modeling and analysis of neural spiking patterns (71; 72; 73) while ignoring the underlying dynamics which generate these patterns. Although computational modeling and analysis approaches in the framework of dynamical systems theory developed in computational neuroscience (74) have attempted to fill this gap, these approaches are often based on several underlying assumptions. More importantly, most of these approaches suffer from the issue of scalability to analyze large-scale neuronal populations. In the realm of our growing ability to record spiking activity from large neuronal populations in behaving animals over an extended time, we require scalable computationally efficient tools that can meet the ever-growing data with minimal assumptions on the underlying dynamics. In this chapter, we aim to take advantage of both worlds (i.e., dynamical systems theory and statistical analysis) by developing novel scalable data-driven dynamical systems modeling and analysis approaches to infer, and analyze the neural dynamics underlying large-scale spatiotemporal spiking patterns.

There is growing evidence in the brain-machine interface literature that there exist low-dimensional neural manifolds embedded in the high-dimensional neuronal spiking spatiotemporal patterns underlying reaching tasks (72). Such low dimensional neural dynamical structures have been found in various brain regions including the motor cortex (75), premotor cortex (76), thalamus (77), and the visual cortex (78). A wealth of single-trial spiking data analysis tools, such as Gaussian process factor analysis (GPFA) (79; 80) and latent factor analysis via dynamical systems (LFADS) (53), exists in the neuroscience literature to reveal these low-dimensional manifolds from the single-trial population spiking data. These tools utilize dimensionality reduction techniques to infer the low dimensional spatiotemporal neural structures from single-trial spiking patterns while providing a way to reconstruct the spiking patterns from these inferred neural structures. These tools not only allow us to gain insight into single-trial variability factors, such as change in the initial states or the stimulus, but also to visualize how these neural manifolds vary across different experimental trials or different tasks (81). However, these approaches lack to provide detailed systems-theoretic analysis of various spatial modes and temporal dynamics underlying the low-dimensional neural manifolds, which could provide a better understanding of the dynamics governing the observed spiking activity. More importantly, it is not possible to answer relevant questions on understanding neural dynamics such as what the governing dynamical equations of the low-dimensional neural manifolds are as well as how similar or different these dynamics are for behaviorally similar but different tasks using these existing approaches.

In this chapter, we develop novel data-driven approaches to infer the firing rates and latent neural manifolds from the high-dimensional spiking data such that the underlying high-dimensional neural spiking activity can be reconstructed. For this, we leverage techniques from deep learning and generative models. Figure 3.1 illustrates the overall framework for inferring firing rates from the high-dimensional spiking data.

As shown in Figure 3.1, our central idea is based on sequential variational autoencoders, a class of generative models based on the dimensionality reduction that allows the reconstruction of the high-dimensional data from the inferred latent dynamics. Sequential variational autoencoding approaches have found success in applications such as collaborative filtering (82), outlier detection and removal(83), and natural language processing (84). Here, we introduce an implementation of a sequential variational autoencoder for inferring smoothed firing rates of neuronal populations directly from binned spiking data in an unsupervised manner by generating smooth firing rates that maximize the likelihood of the reconstruction of the original data.

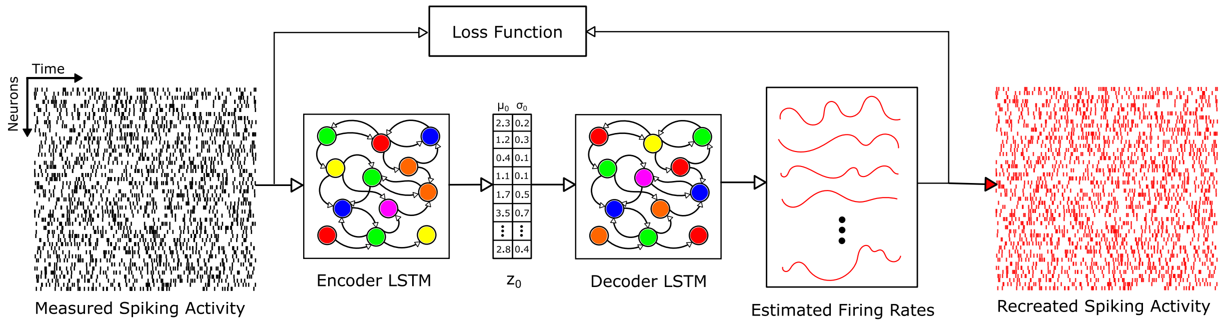


FIGURE 3.1: Illustration of general inference approach. Measured binned spikes are fed into the encoder LSTM network in array form. The bidirectional encoding LSTM returns a vector of Gaussian parameters, z_0, describing the initial condition of the network. A sample is drawn from the Gaussian distribution described by the parameters, and is fed into the decoder LSTM network, which returns the smoothed inferred firing rates of the neurons. These can then be compared in a probabilistic loss function to return the likelihood that the inferred firing rates generated the original measured spiking activity. This loss is then used to further optimize the network in iterative fashion via backpropagation.

This approach centers on several central assumptions: (1) we assume that the stochastic firing behavior of individual neurons can be modeled as a nonhomogeneous Poisson point process, (2) that underlying dynamics governing the firing rates of the neurons exist, and (3) that trial-specific initial conditions underlying the neural circuit exist and can be inferred. A major strength of this approach lies in the RNN autoencoder's ability to model highly non-linear temporal patterns, such as those expressed in neuronal circuits. This can be attributed to the general properties of ANNs as universal function approximators (7) (85). Their ability to learn highly-complex and otherwise intractable nonlinear functions governing the encoding and decoding of complex data structures forms the backbone of this autoencoding approach.

The remainder of this chapter is organized as follows. In Section 3.2, we provide background information relating to variational autoencoding approaches, and detail related approaches to the one developed in this Chapter. In Section 3.3, we provide technical details regarding formulation of the inference problem, and the development of our novel autoencoding approach adapted for sequential data. Additionally, we provide descriptions of the dynamical systems we use in simulation for generation of training and validation data, and detail the generation of our synthetic datasets used in training. We present the results from training, and quantify the inference ability of the trained autoencoders in Section 3.4. Finally, in Section 3.5, we provide discussion of results and limitations of the developed approach, and provide directions for future works and research.

3.2 RELEVANT BACKGROUND AND LITERATURE

In Section 3.2.1 we provide an overview of the background of variational autoencoding approaches. In Section 3.2.2 we detail related inference techniques and approaches to our own, which we will develop in detail in Section 3.3.

3.2.1 *Variational Autoencoders*

A Variational Autoencoder (VAE) is a class of generative neural network models, which are a probabilistic interpretation of the autoencoder, a model type that generally performs dimensional compression on high-dimensional input data by reducing it to a latent state. However, the VAE is a class of generative neural network model whose goal is not to learn to compress data into a deterministic latent state vector, but rather to compress it into parameters of a probabilistic distribution.

Our method can be considered as an RNN-based extension of a Variational Autoencoder (VAE) extended to processing sequential data. This autoencoding approach can be broken down into two main components: the encoder, E, and the decoder, D. The encoder transforms the input sequence of binned spike trains, $\mathbf{x}$, into a conditional distribution over some latent state $\mathbf{z}$, $E(\mathbf{z}|\mathbf{x})$. Typically, this is assumed as an independent diagonal Gaussian distribution, with each element $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$. This assumption is used in the construction of this Sequential VAE model.

Let $\hat{\mathbf{z}}$ denote a sample from the stochastic latent state, $\mathbf{z}$. The decoder accepts this sample and attempts to transform it back into the original data upon which the latent state is conditioned. That is, the output of the decoder can be represented as $D(\mathbf{x}|\hat{\mathbf{z}})$.

The autoencoder is comprised of these two different encoding and decoding networks being jointly trained on the same probabilistically formulated loss function, forming the variational autoencoding approach. If the autoencoder has been designed properly, $\hat{\mathbf{x}}$ should closely resemble the input data into the encoder portion of the network, $\mathbf{x}$.

Unlike traditional applications of variation autoencoding approaches, we are not interested in dimensionality reduction in this portion of the approach. We are only interested in inference of the N neuron's firing rates.

3.2.2 *Comparison With Existing Approaches for Inferring Latent Neural Manifolds from Single Trial Spiking Data*

This approach is not unique in its goal to discover low-dimensional neural trajectories from higher-dimensional neural population activity. Several previous approaches have been developed and have found success inferring smoothed firing activity from noisy higher-dimensional data, such as Gaussian-process factor analysis (GPFA)(79) and Latent factor analysis for dynamical systems (LFADS) (53).

GPFA is a novel method for extracting low-dimensional dynamical representations of high-dimensional behavior, constructed in a probabilistic framework that inherently allows for simultaneous smoothing and dimensionality reduction. The GPFA model is a set of factor analyzers at each time point, with a common Gaussian process prior. This introduction of a Gaussian process allows for time-correlation of data points, which in turn allows for inference of low-dimensional dynamical behavior. However, GPFA does not allow for the discovery of non-linear low-dimensional manifolds of activity, leaving it best used for exploratory studies of neural trajectories.

Similarly to the method we have devised, LFADS makes use of a sequential variational autoencoding approach to discover smoothed firing rates underlying recorded and binned neural spiking activity. This is done by use of a jointly-trained, encoding-decoding approach. In LFADS, the encoder is a bidirectional implementation of a gated recurrent unit (GRU)(86), a type of gated RNN whose internal gating structure is more complicated than a vanilla RNN, but not so complicated as the LSTM. This bidirectional GRU returns an initial condition of the system, which is passed on to the generator portion of the network. The generator is comprised of a forward-directional GRU network, with additional custom layer structure. This custom layer structure is comprised of a linear layer, which transforms the high-dimensional representation of the dynamics into a low-dimensional latent representation of the dynamics, which is then transformed linearly and exponentiated to recreate strictly positive firing rate dynamics. This differs from our approach, in which we seek to identify only smoothed firing rates from our implementation of a sequential variational autoencoder, and perform dimensionality reduction and latent-state dynamical identification separately in a downstream process. This process of encoding, decoding, dimensionality-reduction, and subsequent firing rate reconstruction from the low-dimensional representation makes the process of convergent learning extremely difficult, as there are a huge number of parameters to be learned. To overcome this, LFADS makes use of a highly tailored, complex custom learning algorithm.

In the coming sections, we will detail a statistical inference method that seeks to capture smoothed Poisson firing rates underlying binned spiking data, with the goal of being able to learn complex, non-linear dynamics while allowing for compatibility with standard optimization and training techniques.

3.3 METHODS

Here, we present the overview of the structure underlying the developed autoencoder architecture and dynamical inference approach. For notational simplicity in the coming sections, we denote the affine transformation $\mathbf{W}_i^T\mathbf{x} + \mathbf{b}_i$ as $W_i(\mathbf{x})$. In Section 3.3.1, we provide an overview of the encoder, which accepts input binned spike trains and returns the initial condition of the system. In Section 3.3.2, we delve into details regarding the generation of firing rates from the initial condition provided by the encoder. Section 3.3.3 details the development of the probabilistic loss function that our model is trained to maximize in an unsupervised learning framework. In Section 3.3.4, we provide details related to implementation and best practices in constructing the network. Specifics regarding the approach to further distill low-dimensional latent firing rate dynamics from the high-dimensional inferred firing rate data is given in Section 3.3.5. Finally, in Section 3.3.6, we provide descriptions of synthetic systems and detail the generation of synthetic neuronal spiking data from them to evaluate our approach.

3.3.1 *Encoder*

We consider data in the form of binned spike counts, $\mathbf{x}$, collected over T discrete time intervals, recorded from N individual neurons. As in many variational autoencoding approaches, we assume that the encoded latent state of the network is some diagonal Gaussian distribution, with form $\mathcal{N}_n(\boldsymbol{\mu}, \sigma\mathbf{I}_n)$. We wish to encode the entire sequence of these binned spike counts into an initial state vector, $\mathbf{z}_0$, which will be used to initialize the iterative prediction of the firing rate dynamics.

To accomplish this, we use a multi-layered, bidirectional LSTM network to run over the binned spiking data in both directions from $t = 1$ to $t = T$, and from $t = T$ to $t = 1$, returning a single vector with $2N$ elements, $\mathbf{C}$. The bidirectional nature of the LSTM encoder allows for the entire time series worth of input data to be encoded into a single vector, while minimizing bias attributed to directionality of sequential processing, and can be described as

$$\mathbf{C} = LSTM^{enc}(\mathbf{x}) \tag{3.1}$$

The initial condition vectors, $\boldsymbol{\mu_0}$ and $\boldsymbol{\sigma_0}$, are the output of two densely connected nonlinear transformations given by

$$\boldsymbol{\mu_0} = \mathbf{W}^{\mu_0}(\mathbf{C}) \tag{3.2}$$

$$\boldsymbol{\sigma_0} = exp\big(\mathbf{W}^{\sigma_0}(\mathbf{C})\big) \tag{3.3}$$

where $\mathbf{W}^{\mu_0}$ and $\mathbf{W}^{\sigma_0}$ are separate, trainable weight matrices. The sample from the encoded mean and variance vector is then taken as

$$\hat{\mathbf{z}}_0 \sim E(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0\mathbf{I}_n) \tag{3.4}$$

where each i^{th} element of the latent initial condition, $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$ is a element of a diagonal Gaussian distribution.

3.3.2 *Decoder*

This approach assumes that the observed binned spikes, $\mathbf{x}$, are samples from a Poisson process, with underlying firing rates, $\mathbf{r}$. We consider each neuron to have an corresponding independent Poisson firing rate, r_i. The overarching goal of this approach is to reconstruct the estimated firing rates, $\hat{\mathbf{r}}$ that maximize the probability of the input binned spikes into the system, $\mathbf{x}$. The rates are obtained as the exponentiated output of the final layer of the generative LSTM cell. The choice of the final non-linear exponentiation, $exp(\cdot)$, ensures that the estimated firing rate of the network remains positive. Additionally, $exp(\cdot)$ is the inverse canonical link function to the Poisson distribution, making it a natural choice in that regard.

The decoding process begins when a sample is drawn from the encoded initial state vector, $\hat{\mathbf{z}}_0 \sim E(\mathbf{z}|\mathbf{x})$. This is passed into the decoder LSTM cell, and the initial latent firing rate is generated as,

$$\hat{\mathbf{g}}_0 = LSTM^{gen}(\hat{\mathbf{z}}_0) \tag{3.5}$$

For each subsequent time step $t = 1, ..., T$, the subsequent latent firing rate is generated by accepting the previous time step's estimated latent rate as the input into the decoder cell

$$\hat{\mathbf{g}}_t = LSTM^{gen}(\hat{\mathbf{g}}_{t-1}) \tag{3.6}$$

Due to the nature of the LSTM cell, which typically necessitate the use of a hyperbolic tangent or sigmoidal output function, this constrains the output from the generative LSTM network to be on {-1,1} or {0,1}. Thus, we pass the output latent firing rates through a densely-connected output layer with an exponential activation. This allows for strictly positive firing rates, and $exp(\cdot)$ is the inverse canonical link function to the Poisson distribution, making it a natural choice for the output layer activation function. The firing rates are then output as

$$\hat{\mathbf{r}}_\mathbf{t} = exp\big(\mathbf{W}^{out}(\hat{\mathbf{g}}_\mathbf{t})\big) \tag{3.7}$$

where $\mathbf{W}^{out}$ represents a trainable parameter matrix. Once the firing rates have been generated, Poisson sampling can be used to approximately recreate the original input binned spike trains,

$$\hat{\mathbf{x}}_t \sim Poisson(\mathbf{x}_\mathbf{t}|\hat{\mathbf{r}}_\mathbf{t}) \tag{3.8}$$

where *Poisson* indicates that independent Poisson sampling is performed for each of the individual neuronal firing rates expressed as an element of the estimated rate vector, $\hat{\mathbf{r}}_t$. This final step is not typically done, as inference of the firing rates, rather than total reconstruction of the original binned spike trains, is the overarching goal of this developed inference technique.

3.3.3 *Autoencoder Loss Function*

The probability mass function of a Poisson distribution within a discrete time interval can be expressed as

$$P(k|\lambda) = \frac{e^{-\lambda}\lambda^k}{k!} \tag{3.9}$$

where k represents the number of events within some discrete time period, λ is the underlying event frequency, often denoted as the rate parameter, and e is Euler's number. The event frequency could also be described as the product of an instantaneous firing rate, $\mathbf{r}$ and the measure of the discrete time interval Δt.

This is easily translated to Poisson spiking neurons, where the observed event counts, k, are equivalent to the observed spike trains, $\mathbf{x}$, binned into discrete time intervals of length Δt, with the governing Poisson rate parameters, λ, equivalent to the estimated inferred firing rates from the autoencoder within each timestep, $\hat{\mathbf{r}}\Delta t$. Thus, for each timestep, t, the probability mass function can be expressed as

$$P(\mathbf{x}_t|\hat{\mathbf{r}}_t\Delta t) = \frac{e^{-\hat{\mathbf{r}}_t\Delta t}(\hat{\mathbf{r}}_t\Delta t)^{\mathbf{x}_t}}{\mathbf{x}_t!} \tag{3.10}$$

The autoencoder model is optimized such that it maximizes the log-likelihood of the original binned spike trains, given the estimated firing rates from the autoencoder, $logP(\mathbf{x}|\hat{\mathbf{r}}\Delta t)$. The log-likelihood is used as opposed to the likelihood to ensure numerical stability and avoid rounding errors in training. Thus at each time step, we compute the log-likelihood of the original binned spike trains, given the autoencoder's estimated firing rates as

$$logP(\mathbf{x}_t|\hat{\mathbf{r}}_t\Delta t) = \mathbf{x}_t log(\hat{\mathbf{r}}_t\Delta t) - \hat{\mathbf{r}}_t\Delta t - log(\mathbf{x}_t!) \tag{3.11}$$

Thus, for the entire timeseries of inferred firing rates over T timesteps, the total firing rate reconstruction loss can be computed as

$$\mathbf{L_r} = \sum_{t=1}^{T} logP(\mathbf{x}_t|\hat{\mathbf{r}}_t\Delta t) \tag{3.12}$$

However, the VAE framework centers around maximizing the lower evidence bound of the data,

$$logP(\mathbf{x}) \geq \mathcal{L} = \mathbf{L}_r - \mathbf{L}_{KL} \tag{3.13}$$

where $\mathbf{L}_r$ represents the aforementioned reconstruction cost of the firing rates given the original data, and $\mathbf{L}_{KL}$ represents a non-negative Kullback-Leibler Divergence term that serves to restrict the posterior initial condition to the decoder from deviating too far from the uninformative assumed Gaussian prior, and is defined as

$$\mathbf{L}_{KL} = \mathcal{D}_{KL}\big(\mathcal{N}_n(\mathbf{z}_0|\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0\mathbf{I}_n)|\mathcal{N}_n(0, \mathbf{I}_n)\big) \tag{3.14}$$

where $\mathcal{D}_{KL}$ represents the Kullback Leibler Divergence between the two distributions. This term serves as a regularizer on the posterior inferred from the network, ensuring that the distribution inferred remains 'near' the uninformative Gaussian prior.

We wish to maximize the lower evidence bound, $\mathcal{L}$, thus in practice we minimize the additive inverse, $-\mathcal{L}$.

3.3.4 *Further Details of Autoencoder Implementation*

Here, we provide details of implementation of the SVAE that were not mentioned in the more general functional description from sections 3.3.1, 3.3.2, and 3.3.3.

For all instances of an RNN being used throughout this work, we have used recurrent dropout, a form of regularization on the recurrent weights of the neural network architecture, to ensure generality of the solutions learned by the network. It is common to use L2 regularization as a means to achieve the same end, but tuning L2 regularization weight penalties by hand or automating the process becomes an extremely difficult problem in and of itself. Recurrent dropout ensures that at any given time step, a certain number of recurrent weights are masked, ensuring that the total set of recurrent weights is not overly dependent on any singular element.

Additionally, we employed dropout in the exponential output layer, ensuring that the network's weights transforming the generator's latent firing rate representation to the real-valued one remains generalized, and not over-reliant on a single particular weight within the layer.

3.3.5 *Identification of Latent Firing Dynamics*

In addition to inference of smoothed Poisson firing rate dynamics directly from the observed binned spiking data, we are also interested in the distillation of low-dimensional latent dynamics capable of describing their high-dimensional behavior. Towards this end, we employ deterministic autoencoders to recreate the original latent dynamical behavior. In practice, this deterministic autoencoder is simply a multi-layer perceptron, whose innermost layer, known as a "bottleneck", is comprised of a significantly fewer number of artificial neurons than both the input and output layers. This sharp decrease in the number of available neurons forces the data into a low-dimensional, high-information representation of the original data. It is this low-dimensional, 'latent' representation that we wish to ultimately distill, as this would enable identification techniques such as SINDY to relate this latent dynamical behavior to closed-form differential equation representation.

3.3.6 *Synthetic Datasets*

Lorenz Chaotic System — To examine the inference ability of this approach, we examined its performance on a variation of the well-characterized chaotic Lorenz system, with latent dynamics of the form

$$\dot{y}_1 = \sigma(y_2 - y_1) \tag{3.15}$$

$$\dot{y}_2 = y_1(\rho - y_3) - y_2 \tag{3.16}$$

$$\dot{y}_3 = y_1 y_2 - \beta y_3 \tag{3.17}$$

where σ, β, and ρ are the typical parameters used to induce chaotic behavior, and are valued at 10, 8/3, and 28, respectively. Euler integration was used with $\Delta t = 0.001$ s. The chaotic behavior exhibited by this system can be visualized in Figure 3.2. As done in (87), a linear readout matrix was used to transform this low-dimensional chaotic system into a high dimensional one, and firing rates were finally generated by exponentiating the linearly transformed system. For the linear readout matrix, all weights were selected randomly and independently from a uniform distribution on $(1,2)$. The resultant firing rate behaviors can be visualized in Figure 3.3.

To generate the data, we sampled 25 initial conditions, and integrated each response for 100 seconds, yielding timeseries of length $100,000$. The first $20,000$ steps were discarded to ensure the system's steady state behavior. The remaining $80,000$ timesteps were divided into sequences of $1,000$ timesteps (equaling 1 second), yielding 80 firing rate examples per initial condition. Poisson sampling was then performed to generated spiking events from synthetic firing rate data. The spikes were then binned at intervals of 20, 30, 40, and 50 milliseconds, yielding spiking activity vectors of length 50, 33, 25, and 20, respectively.

Chaotic Continuous-Time RNN System — To further test the ability of this generative framework, we We generated an N-dimensional, synthetic dataset from a continuous-time, nonlinear equation of the form

$$\tau \dot{\mathbf{r}}(t) = -\mathbf{r}(t) + \gamma \mathbf{W} tanh(\mathbf{r}(t)) \tag{3.18}$$

where $\mathbf{r}(t)$ represents the firing rates as a function of time, $\dot{\mathbf{r}}(t)$ represents the first order time derivative of the firing rates, γ is a parameter set to $1.5 - 2.5$ depending on the severity of the desired chaotic behavior, τ is a parameter set to 0.025s, and the elements of the matrix $\mathbf{W}$ are drawn independently from a normal distribution with mean zero and variance $1/N$, where N is the number of neuronal firing rates being described. For our applications, $N = 50$. For our purposes, we used $\gamma = 1.5$ for simulations. In this regime, the system exhibits chaotic dynamical behavior. Once all responses were generated, they were normalized, and subsequently multiplied by a scaling factor of 150, giving rise to instantaneous firing rates in the range of $0 - 150$ Hz. An example of the resulting firing rates for a single array of neurons can be seen in Figure 3.4.

Euler integration at $\Delta t = 0.001$ s was used. To generate the data, we sampled from random initial conditions with elements drawn from mean zero with unit variance. Simulation carried on for 100 timesteps, yielding firing rate vectors totaling 1 second.

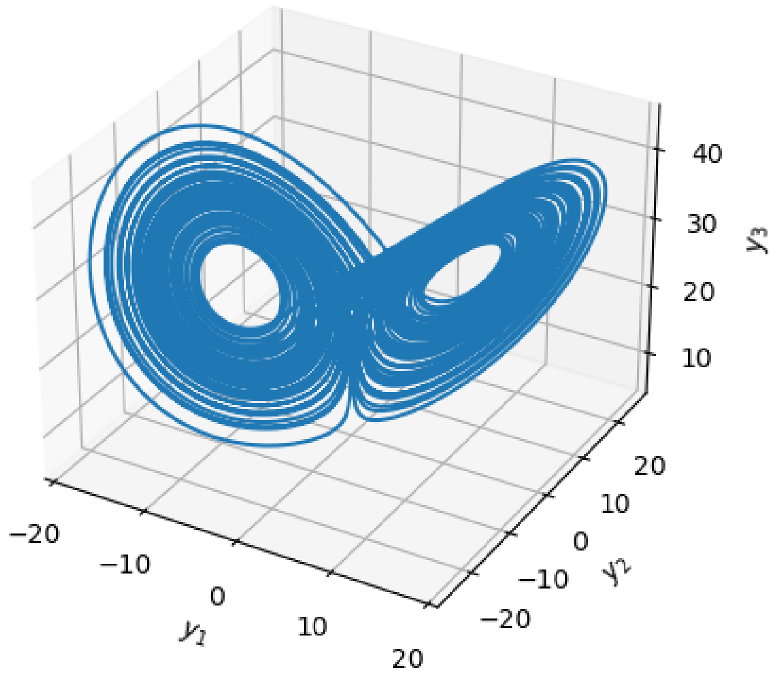


FIGURE 3.2: A sample solution of the Lorenz attractor dynamics when $\rho = 28$, $\beta = 8/3$, and $\sigma = 10$. The 'butterfly' orbit can be clearly visualized.

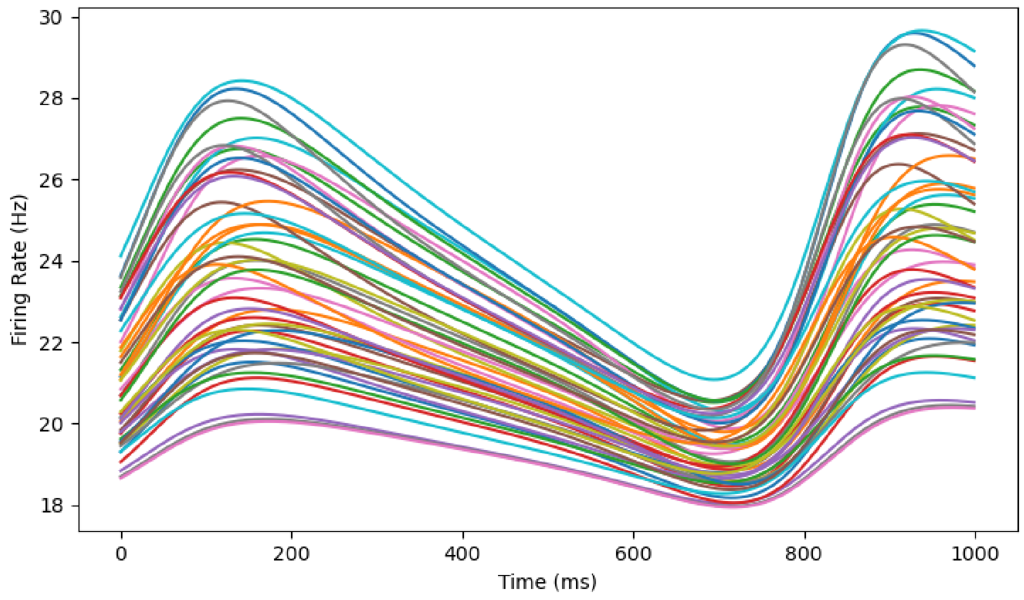


Figure 3.3: Illustration of linearly transformed and exponentiated Lorenz attractor dynamics.

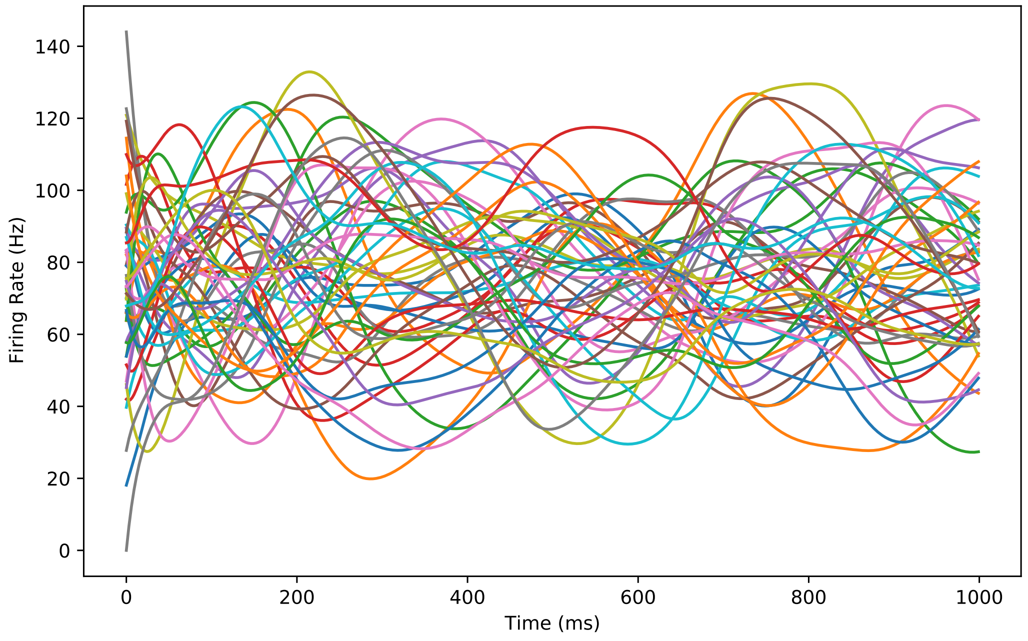


Figure 3.4: Example of chaotic firing rate dynamical behavior.

We generated 2000 examples of independent firing rates, and then performed Poisson sampling from those rates. These Poisson spike events were then binned at 20, 30, 40, and 50ms, yielding binned spike trains of length 50, 33, 25, and 20 time steps, respectively.

The training/testing split was set as 1/5, leading to 4000 examples used as the training set, and 1000 examples used as the test set. In training, we allowed for a randomly split 1/20 of the training set to be used as an on-line validation set during training.

3.3.7 *Optimization*

For all network training, the Adam (66) optimizer was used. Adam is a first-order gradient-based stochastic optimization algorithm, which makes use of lower-order moments of the gradients between layers to optimize the stochastic objective function. Given network parameters, $\theta^{(i)}$, and the objective function $\mathcal{L}(\theta^{(i)}$, the i^{th} update to the parameters is given by

$$m_\theta^{(i+1)} \leftarrow \beta_1 m_\theta^{(i)} + (1-\beta_1)\nabla_\theta \mathcal{L}^{(i)}, \tag{3.19}$$

$$v_\theta^{(i+1)} \leftarrow \beta_2 m_\theta^{(i)} + (1-\beta_2)(\nabla_\theta \mathcal{L}^{(i)})^2, \tag{3.20}$$

$$\hat{m}_\theta = \frac{m_\theta^{(i+1)}}{1-(\beta_1)^{i+1}}, \tag{3.21}$$

$$\hat{v}_\theta = \frac{v_\theta^{(i+1)}}{1-(\beta_2)^{(i+1)}}, \tag{3.22}$$

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \frac{\hat{m}_\theta}{\sqrt{\hat{v}_\theta} + \epsilon}, \tag{3.23}$$

where m_θ is the first moment of the weights in a layer, v_θ is the second moment of the weights in a layer, η is the learning rate, β_1 and β_2 are the exponential decay rates for the moment estimates, ∇ is the differential gradient operator, and ϵ is a small scalar term to help numerical stability. For all optimization in this Chapter, we used $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\eta = 0.001$ (66).

3.4 RESULTS

In this section, we present simulation results from both training and inference of the firing rates underlying the binned spiking activity given to the network, for both the non-chaotic and chaotic synthetic dynamical datasets. In Section 3.4.1, we present results from chaotic Lorenz dynamics with bin sizes of 50, 40, 30, and 20 milliseconds, and demonstrate the network's inference performance at each bin size. Additionally, we discuss the recovery of the latent 3-dimensiona chaotic Lorenz dynamics from the inferred high-dimensional firing rate data. In Section 3.4.2, we present results for inference of chaotic continuous-time RNN dynamical firing rates with bin sizes of 50, 40, 30, and 20 milliseconds.

3.4.1 *Chaotic Lorenz Dynamics*

50ms Bin — The network was trained for $5,000$ epochs to ensure thoroughness in training. This can be visualized in Figure 3.5. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained, the network's mean absolute percentage error in inferring the test set's original firing rates was 5.07%. Figure 3.6 demonstrates an example of the network's ability to infer firing rates underlying observed binned spiking activity.

Here we see the inference procedure has little trouble discovering the underlying firing rates on the chaotic Lorenz dataset binned at 50ms. Qualitatively, while there is mild error in timescale and magnitude, the dynamics inferred from the network closely resemble the ground truth firing rates on which the binned data was created. This behavior is characteristic of the inference approach's performance on this dataset. From this, we may conclude that the network successfully has learned to capture the dynamical behavior governing the firing rates of this system.

40ms Bin — The network was trained for $5,000$ epochs to ensure thoroughness in training. This can be visualized in Figure 3.7. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error

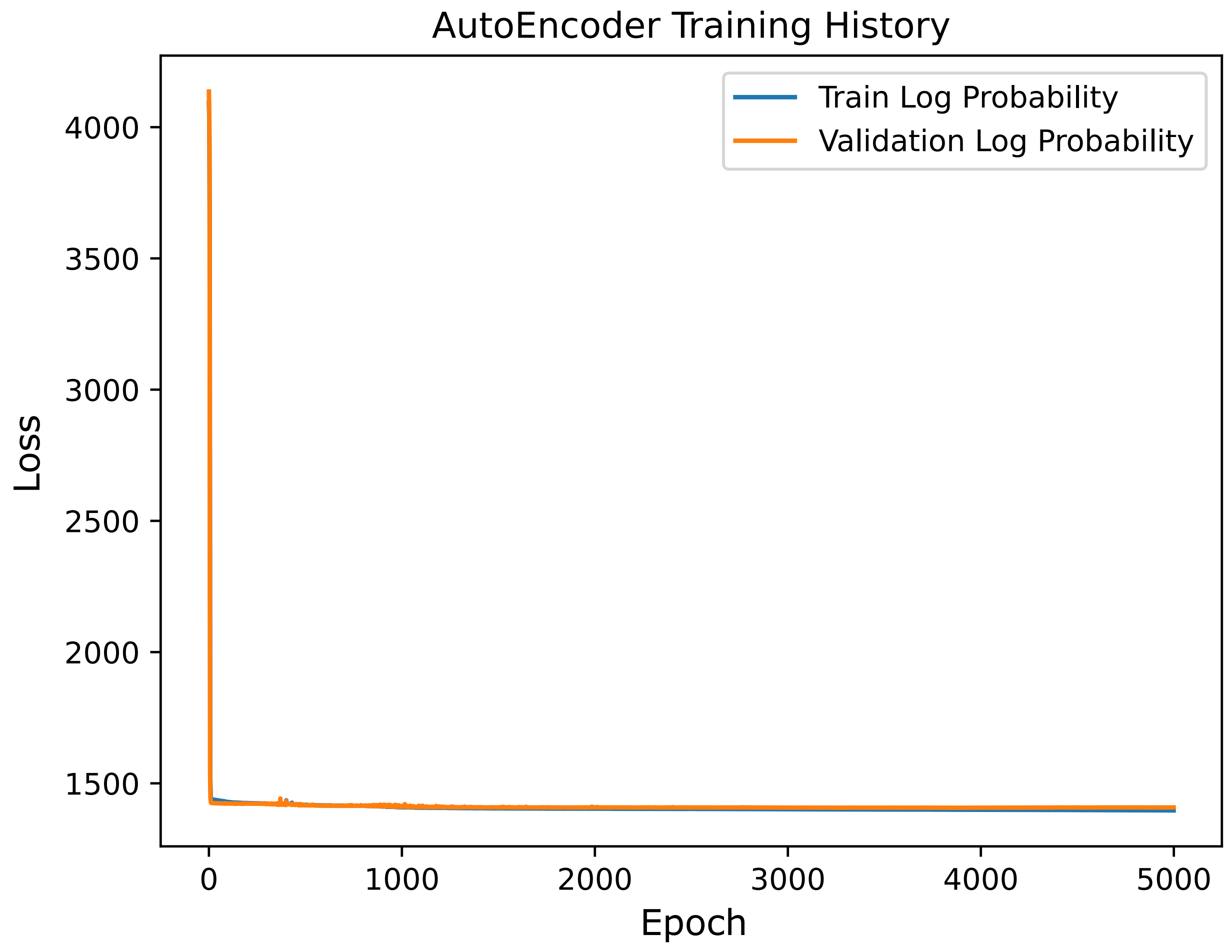


FIGURE 3.5: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer Lorenz spike trains binned at 50ms.

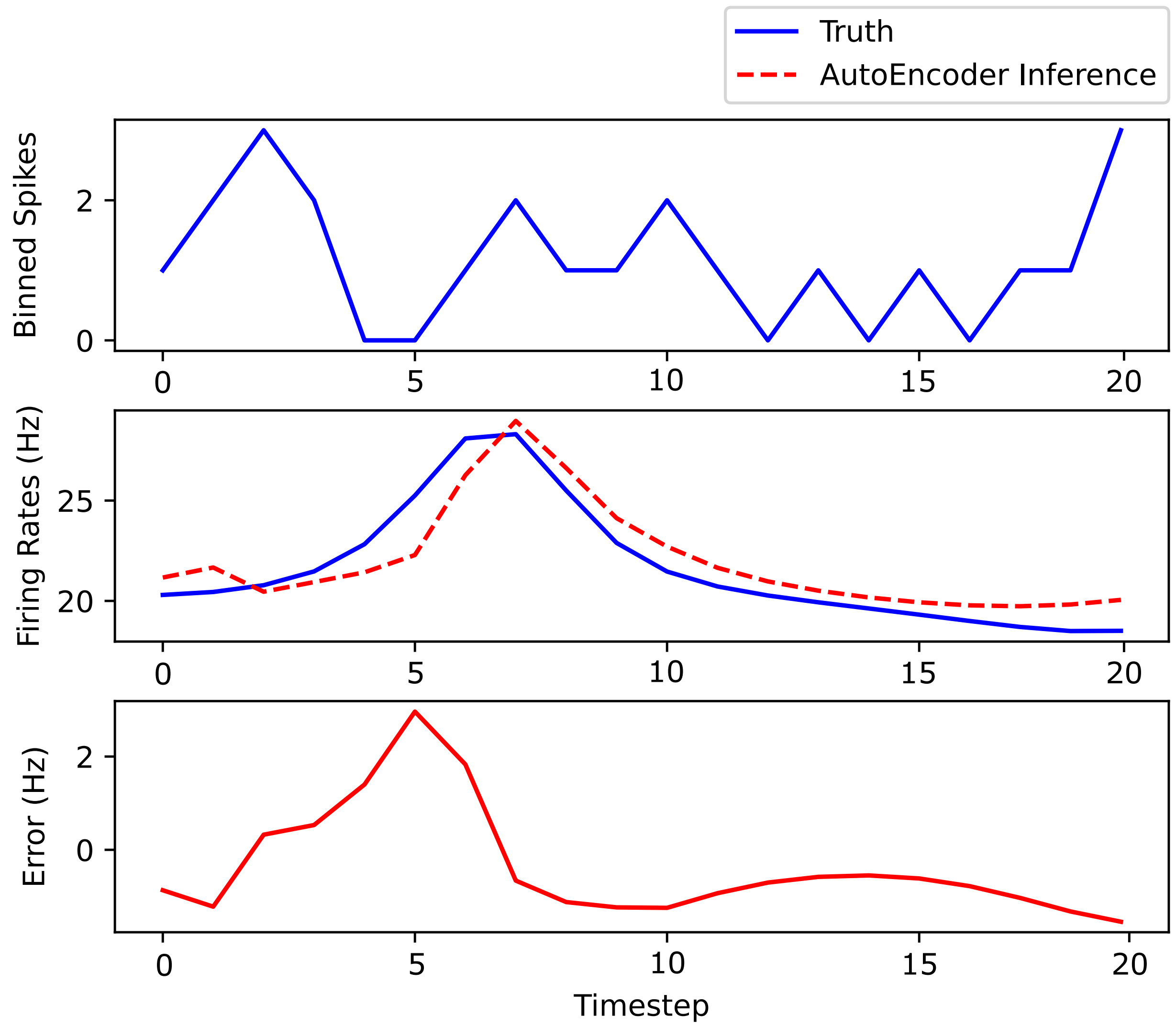


Figure 3.6: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on Lorenz dynamical spiking binned at 50ms. **Top:** Integer binned spike train input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between the prediction and ground truth, measured in Hz.

between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained, the network's mean absolute percentage error in inferring the test set's original firing rates was 5.86%. Figure 3.8 demonstrates an example of the network's ability to infer firing rates underlying observed binned spiking activity.

Here we see the inference procedure has little trouble discovering the underlying firing rates on the chaotic Lorenz dataset binned at 40ms. Qualitatively, while there are errant estimations in both timescale and magnitude, the dynamics inferred from the network closely resemble the ground truth firing rates on which the binned data was created, and error remains very low. This behavior is characteristic of the inference approach's performance on this dataset. From this, we may conclude that the network successfully has learned to capture the dynamical behavior governing the firing rates of this system.

30ms Bin — Once again, the network was trained for 5,000 epochs to ensure thoroughness in training. This can be visualized in Figure 3.9. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained, the network's mean absolute percentage error in inferring the test set's original firing rates was 6.02%. Figure 3.10 demonstrates an example of the network's ability to infer firing rates underlying observed binned spiking activity.

Here we see the inference procedure does well in discovering the underlying firing rates on the chaotic Lorenz dataset binned at 30ms. We note that there are errors in both the timescale and magnitude of the dynamics, but that ultimately the network's inferred rates closely resemble those of the ground truth. This behavior is characteristic of the inference approach's performance on this dataset. From this, we may conclude that the network successfully has learned to capture the dynamical behavior governing the firing rates of this system.

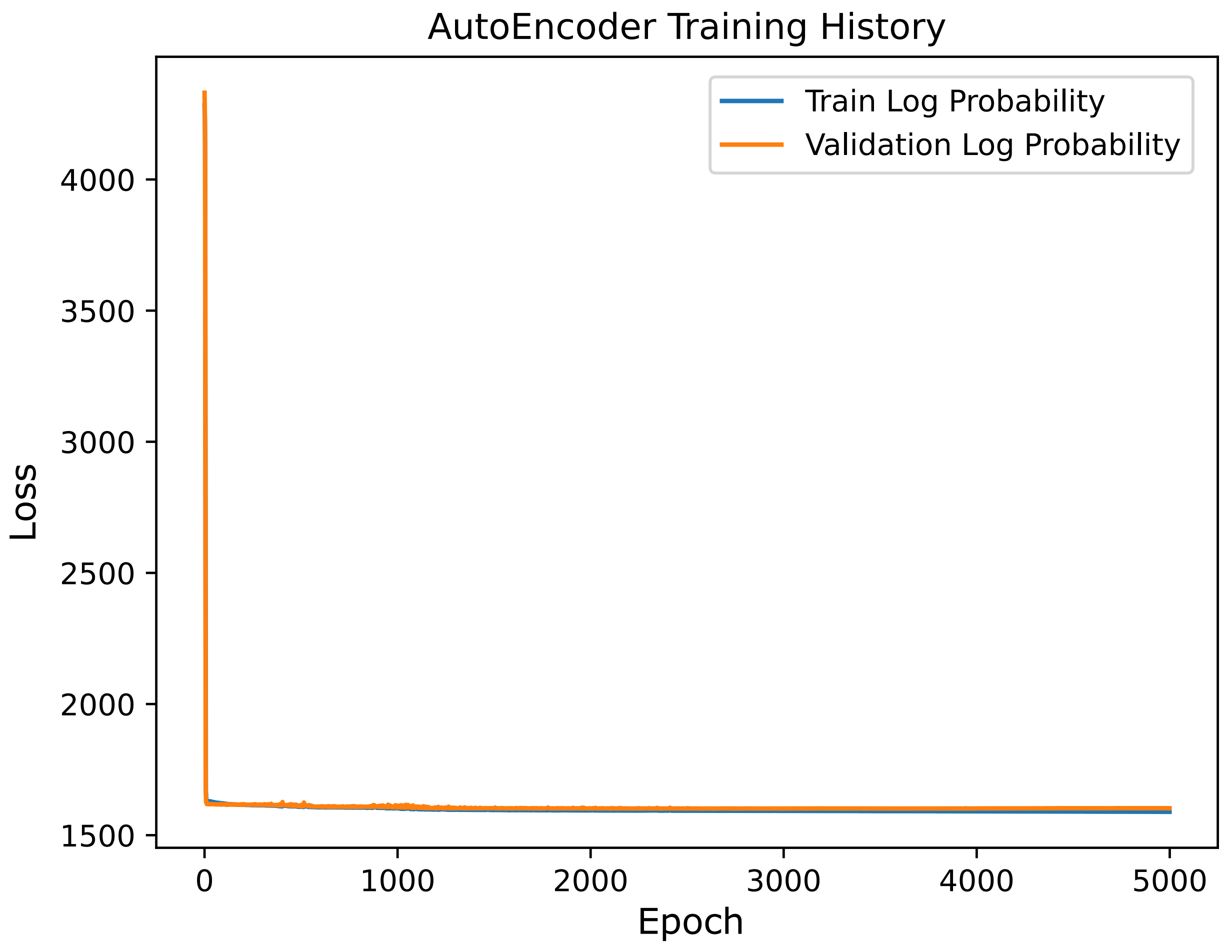


FIGURE 3.7: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer Lorenz spike trains binned at 40ms.

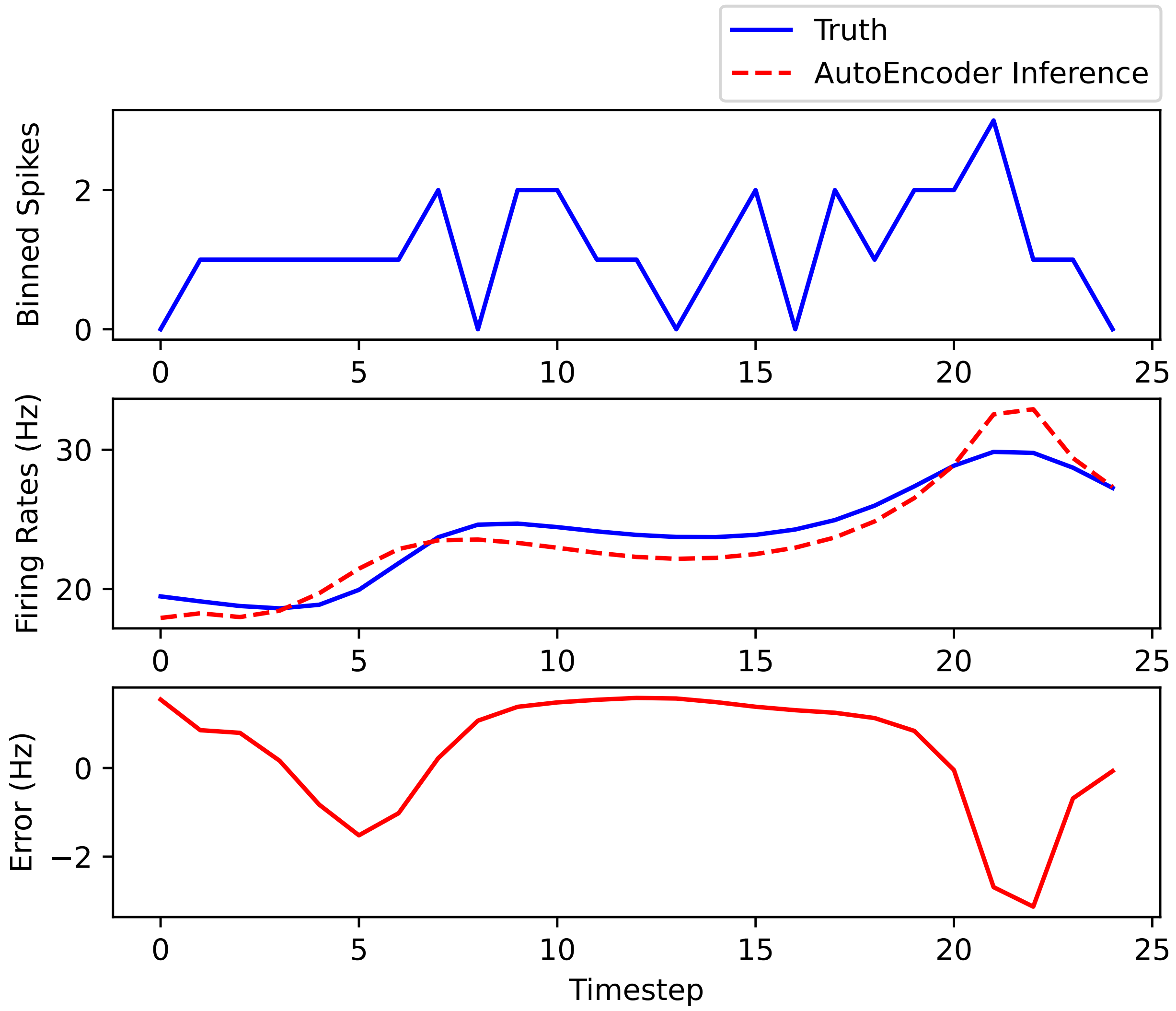


Figure 3.8: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on Lorenz dynamical spiking binned at 40ms. **Top:** Integer binned spike train input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between the prediction and ground truth, measured in Hz.

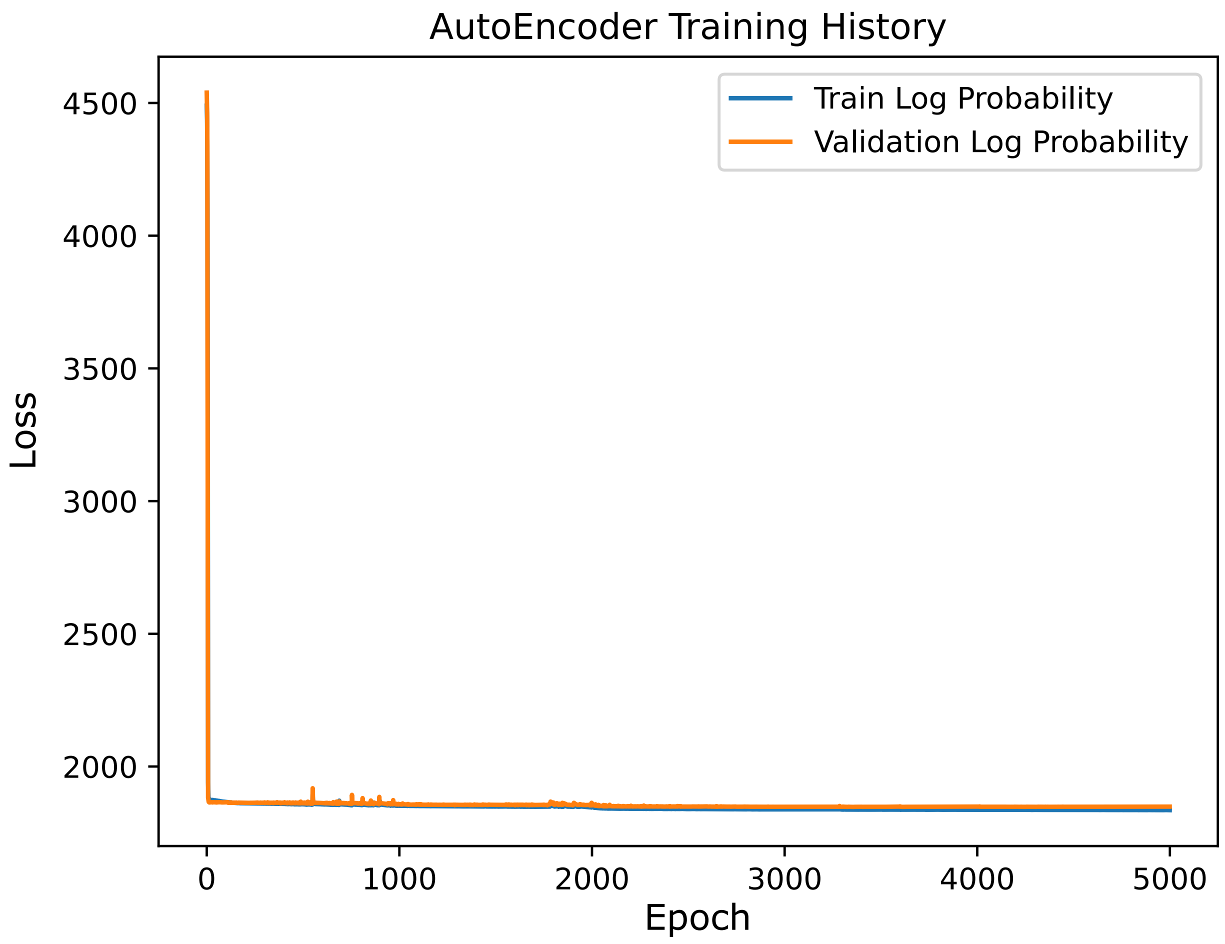


FIGURE 3.9: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer Lorenz spike trains binned at 30ms.

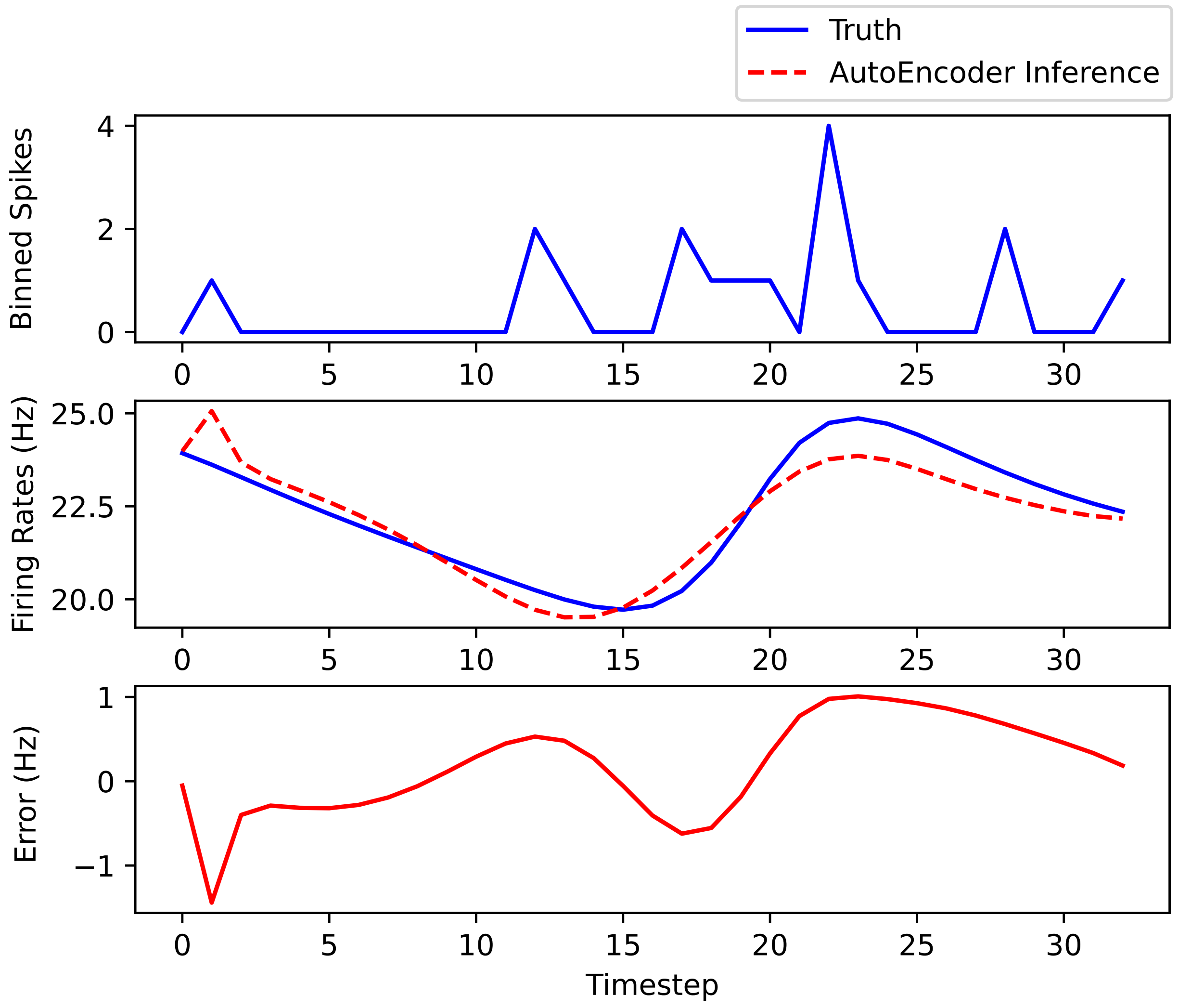


FIGURE 3.10: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on Lorenz dynamical spiking binned at 30ms. **Top:** Integer binned spike train input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between the prediction and ground truth, measured in Hz.

20ms Bin — The network was trained for 5,000 epochs to ensure thoroughness in training. This can be visualized in Figure 3.11. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained, the network's mean absolute percentage error in inferring the test set's original firing rates was 6.47%. Figure 3.12 demonstrates an example of the network's ability to infer firing rates underlying observed binned spiking activity.

We note that at the 20ms bin, performance begins to degrade, as can be seen in Figure 3.12. The network's inferred firing rate dynamics begin to err further in both magnitude and timescale of features exhibited, resulting in significantly higher error. While the inference procedure is certainly learning a representation of the dynamical firing rates underlying the chaotic binned spiking data, the representation is less accurate than at larger bin sizes.

Recovery of Governing Latent Dynamics — Finally, we must make a note on the use of deterministic autoencoders to recover governing latent dynamics, with the hopes of identifying their closed-form representations using SINDY regression techniques. These autoencoders were trained to reproduce the inferred firing rate data output from the sequential variational autoencoder for 2,000 epochs. The autoencoder used was a 3-layer densely connected network with widths of 50, 3, and 50 neurons, from input layer to output layer, respectively.

The deterministic autoencoder was able to reproduce the inferred firing rates from the network with exceptional accuracy in all cases of binning size, with mean absolute reconstruction error rates ranging from 0.2% to 0.55%, with mild variance depending on random weight initialization in the network. This behavior is consistent across all binning window sizes.

However, of more particular concern is the recreation of the original latent 3-dimensional Lorenz dynamics. If the deterministic autoencoder is properly trained and configured, the 3-dimensional encoded latent state from the autoencoder should closely resemble the 3-dimensional Lorenz dynamical data. However, as can be

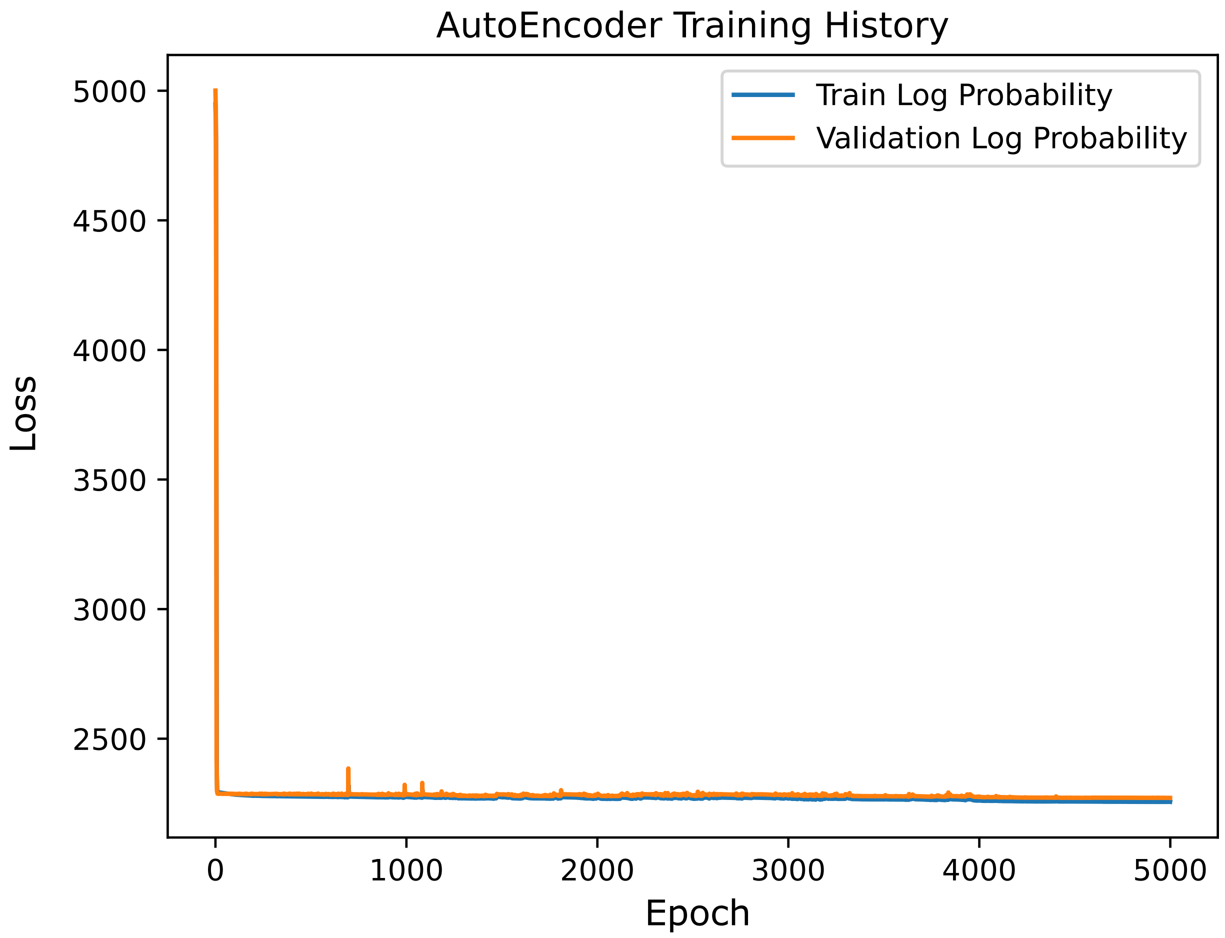


Figure 3.11: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer Lorenz spike trains binned at 20ms.

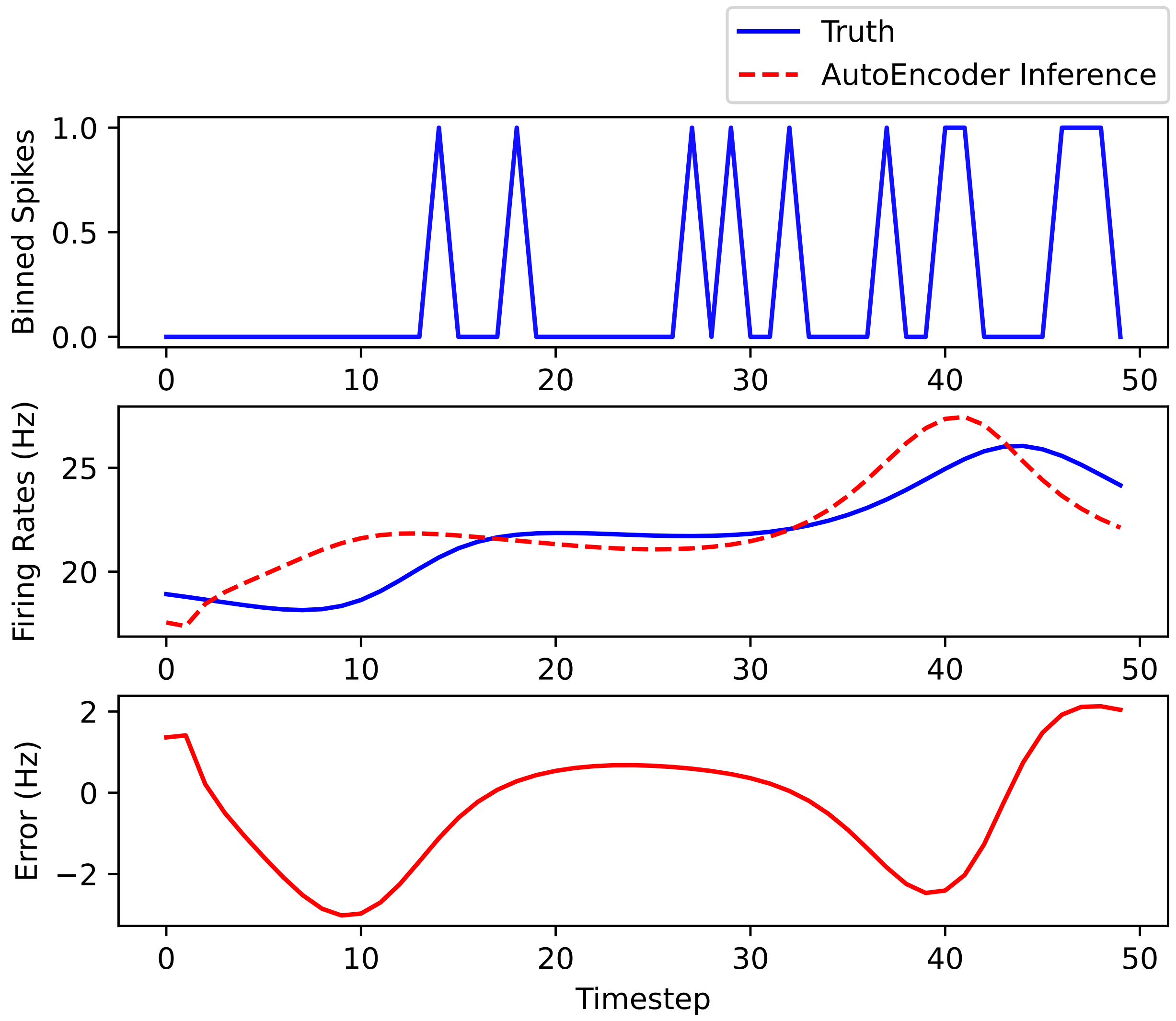


Figure 3.12: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on Lorenz dynamical spiking binned at 20ms. **Top:** Integer binned spike train input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between the prediction and ground truth, measured in Hz.

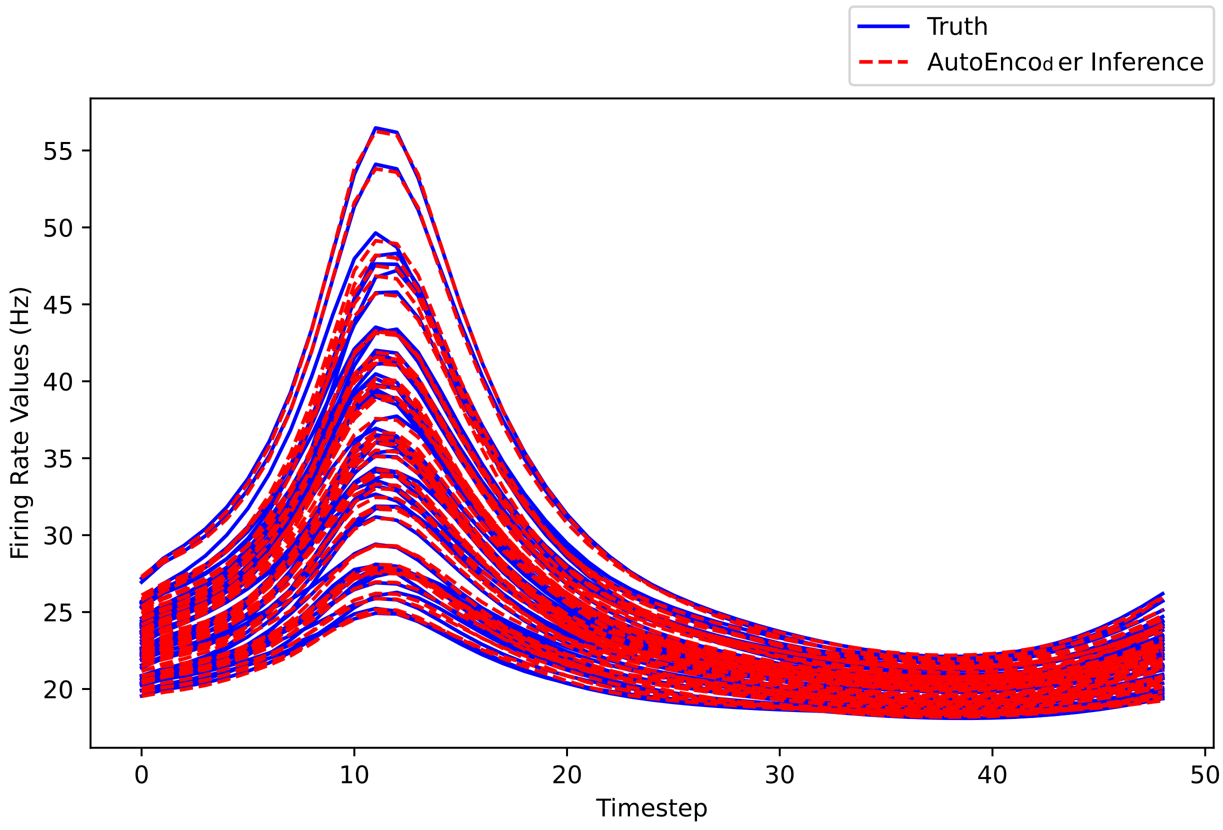


Figure 3.13: Illustration of recreated and originally inferred high-dimensional chaotic Lorenz firing rates from deterministic autoencoder. This example is from the 20ms binning case.

visualized in Figure 3.14, there are currently rather large issues in recovery of the true latent state used to generate the data, as the pictured dynamics do not resemble the Lorenz attractor of Figure 3.2. Without properly recovering the raw latent dynamics, application of the SINDY algorithm to discover their closed-form representation is extremely unlikely to succeed. This result provides a direction for future improvement and research, and constitutes our current most outstanding challenge in discovering latent dynamical representations embedded in high-dimensional spiking activity.

3.4.2 *Chaotic RNN Dynamics*

50ms Bin — The network was trained for 5,000 epochs to ensure thoroughness in training. This can be visualized in Figure 3.15. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained, the network's mean absolute percentage error in inferring the test set's original firing rates was 5.38%. Figure 3.16 demonstrates an example of the network's ability to infer firing rates underlying observed binned spiking activity.

Here we see the inference procedure has little trouble discovering the underlying firing rates on the chaotic dataset binned at 50ms. Qualitatively, while there is mild error, the dynamics inferred from the network closely resemble the ground truth firing rates on which the binned data was created. This behavior is characteristic of the inference approach's performance on this dataset. From this, we may conclude that the network successfully has learned to capture the dynamical behavior governing the firing rates of this system.

40ms Bin — The network was trained for 5,000 epochs to ensure thoroughness in training. This can be visualized in Figure 3.17. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained, the network's mean absolute percentage error in inferring the test set's original firing

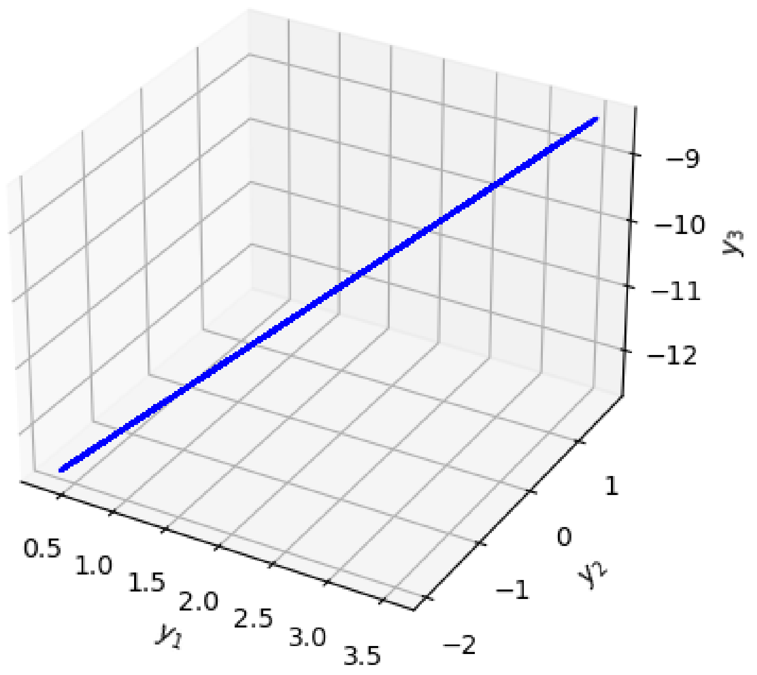


Figure 3.14: Illustration of recreated latent Lorenz chaotic firing rates from deterministic autoencoder. This example is from the 20ms binning case. It can be visualized that these recovered latent rates exhibit purely linear behavior, moving along a line in near perfect unison, instead of the classical Lorenz 'butterfly' attractor trajectories.

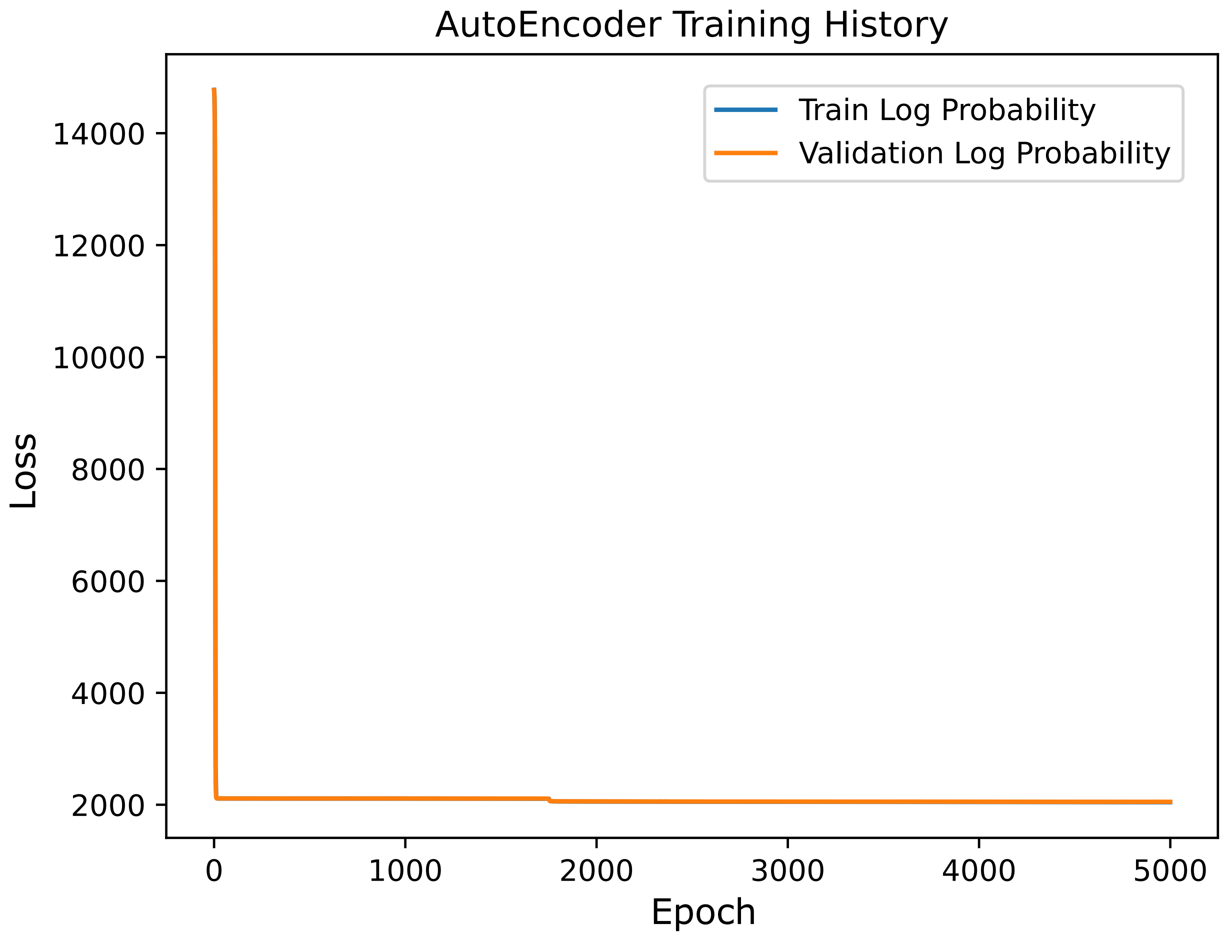


Figure 3.15: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer chaotic continuous-time RNN spike trains binned at 50ms.

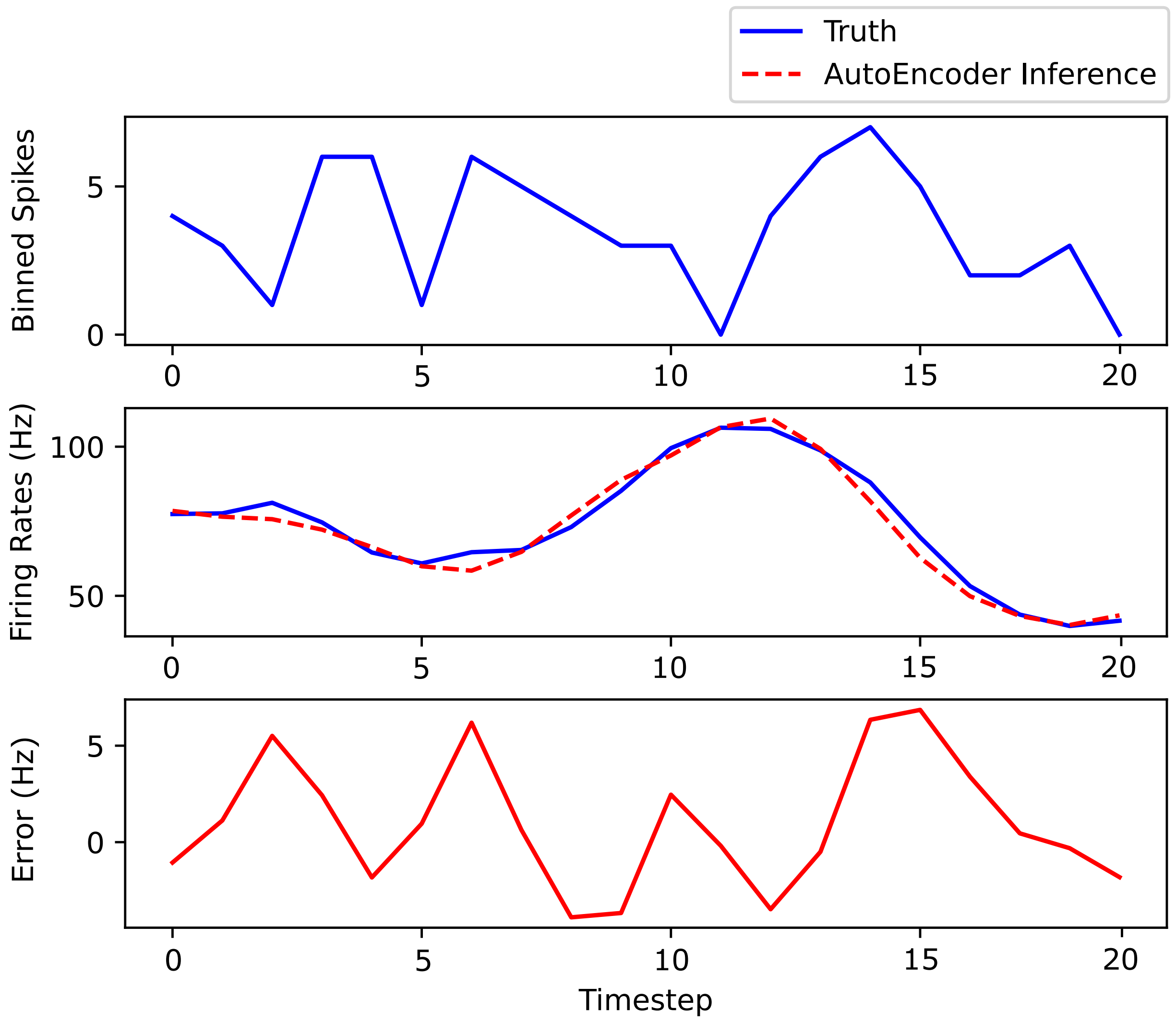


FIGURE 3.16: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on chaotic continuous-time RNN neuronal spiking binned at 50ms. **Top:** Integer binned spike trains input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between prediction and ground truth, measured in Hz.

rates was 6.14%. Figure 3.18 demonstrates an example of the network's ability to infer firing rates underlying binned spiking activity.

Similarly to the 50ms chaotic dataset, we observe that again the inference procedure of the network is able to discover firing rate dynamics that closely resemble the ground truth. This behavior is characteristic of the inference approach's performance on this dataset. From this, we conclude that the inference approach is able to successfully learn the dynamical behavior of the firing rates underlying the binned chaotic dataset.

30ms Bin — The network was trained for $5,000$ epochs to ensure thoroughness in training. This can be visualized in Figure 3.19. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained, the network's mean absolute percentage error in inferring the test set's original firing rates was 6.83%.

Here, we observe that the inference procedure is again able to reconstruct dynamical firing rate behavior that closely resembles the ground truth of the system. However, we must note that the network fails to capture all nuances of the dynamical behavior, often creating slightly overly-smoothed firing rates relative to the ground truth, as can be visualized in Figure 3.20. However, we can conclude that the network has successfully learned the dynamical behavior of the firing rates underlying the binned chaotic dataset.

20ms Bin — The network was trained for $7,500$ epochs to ensure thoroughness in training. The network was trained for the extra epochs to ensure that duration of training was not an issue, as this model struggled to generate dynamical firing rates. This can be visualized in Figure 3.21. Once trained, the network evaluated each of the 400 test set examples. From this, the mean absolute percentage error between the inferred firing rates and those used to generate the binned spiking data was used to quantify the inference performance of the approach. Once fully trained,

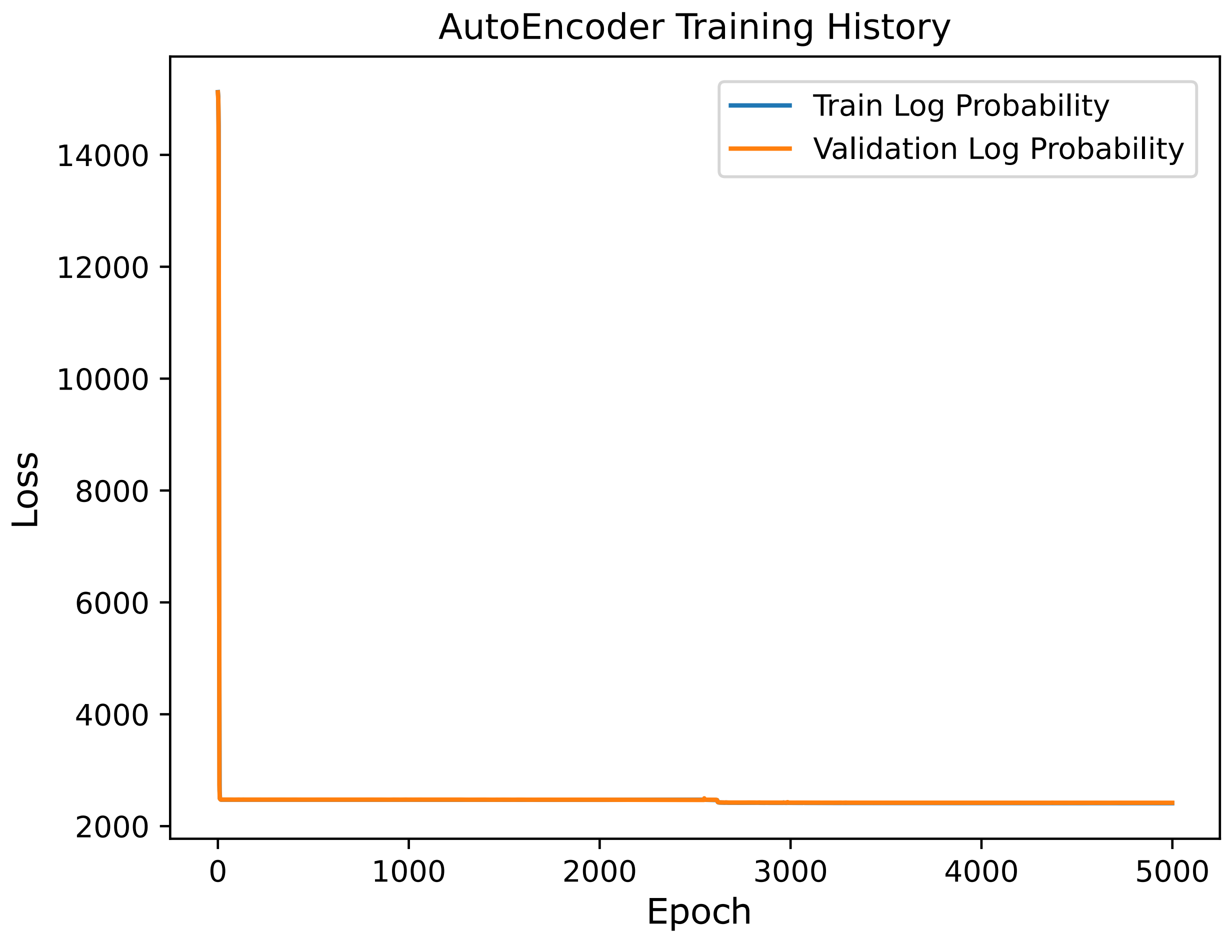


Figure 3.17: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer chaotic continuous-time RNN spike trains binned at 40ms.

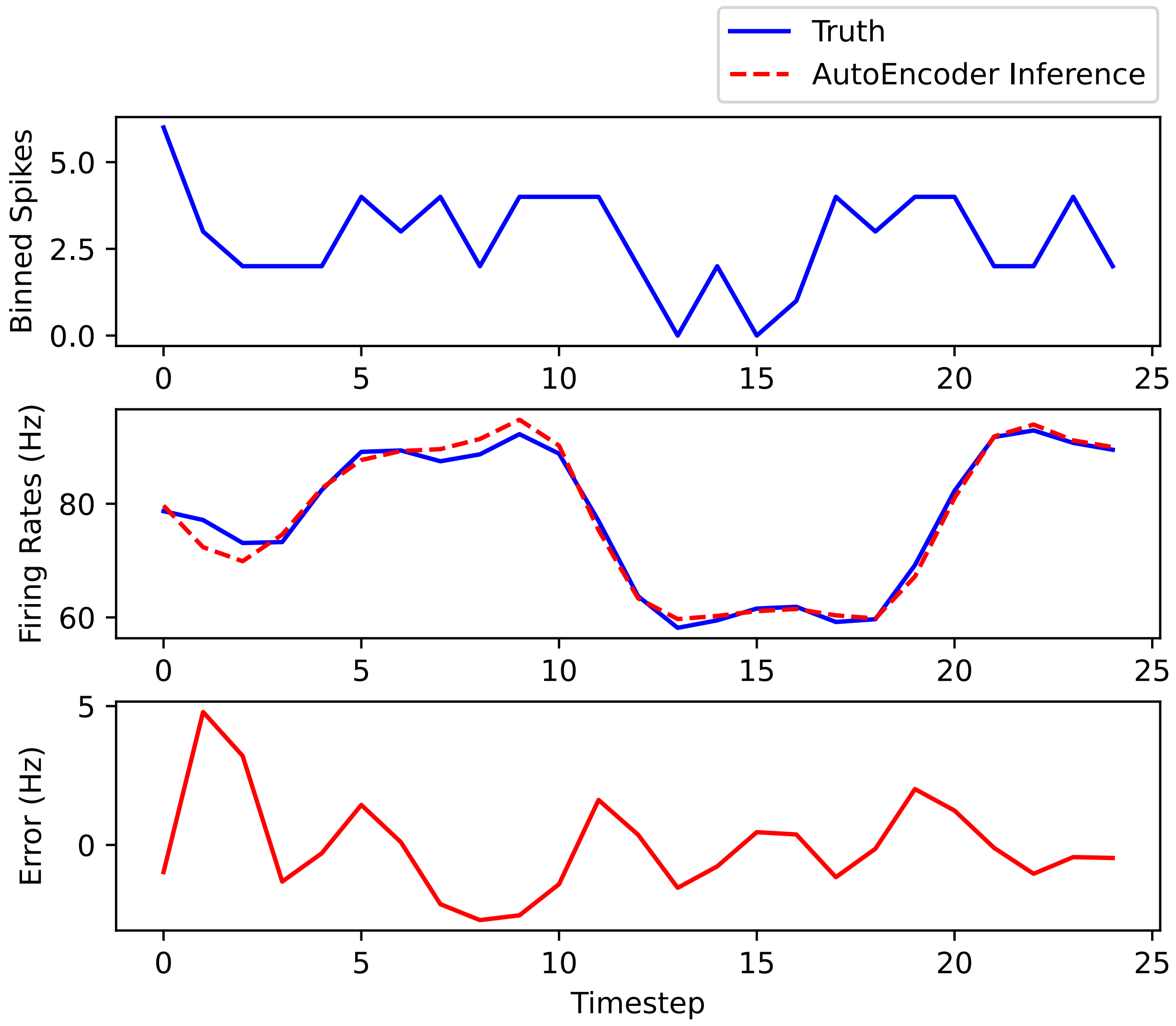


FIGURE 3.18: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on chaotic continuous-time RNN neuronal spiking binned at 40ms. **Top:** Integer binned spike trains input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between prediction and ground truth, measured in Hz.

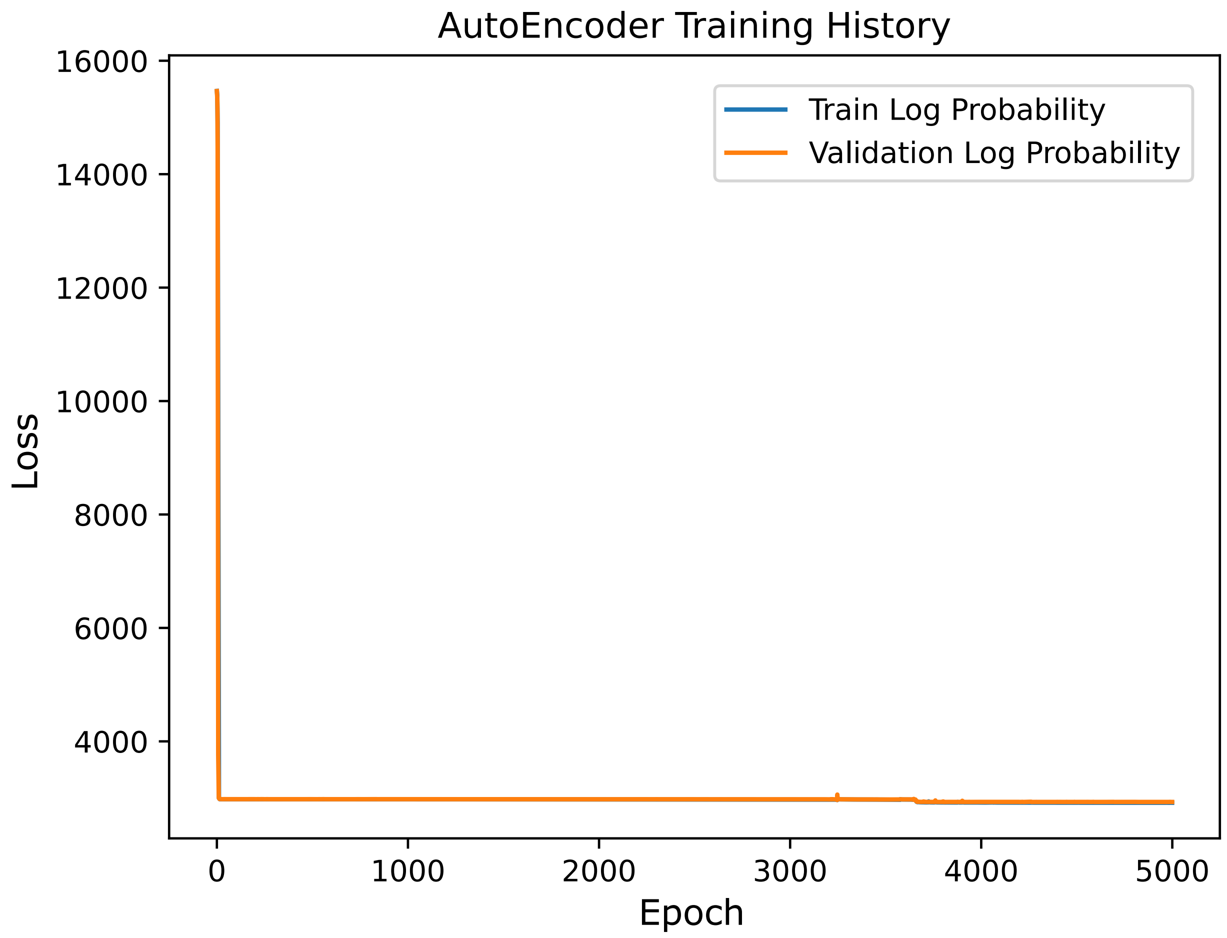


Figure 3.19: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer chaotic continuous-time RNN spike trains binned at 30ms.

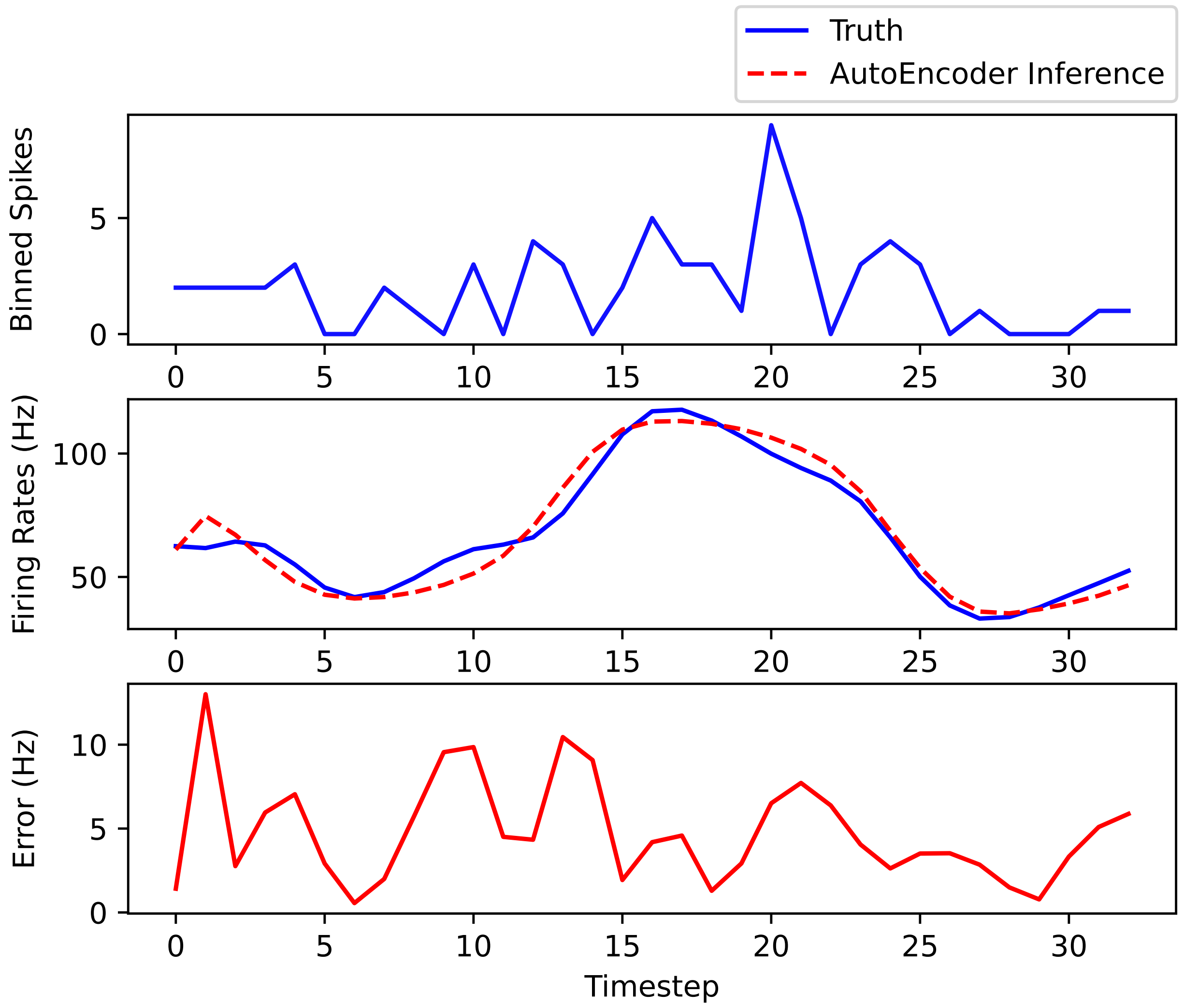


Figure 3.20: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on chaotic continuous-time RNN neuronal spiking binned at 30ms. **Top:** Integer binned spike trains input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between prediction and ground truth, measured in Hz.

the network's mean absolute percentage error in inferring the test set's original firing rates was 9.75%.

We note that at the 20ms bin, performance begins to degrade, as can be seen in Figure 3.22. The first several timesteps exhibit sharp, non-smooth behavior. From there, the network's inferred firing rate dynamics begin to err further in both magnitude and timescale of features exhibited, resulting in significantly higher error. While the inference procedure is certainly learning a representation of the dynamical firing rates underlying the chaotic binned spiking data, the representation is less accurate than at larger bin sizes.

3.4.3 *Impact of Binning Size*

Here, we briefly present summary graphs of mean absolute percentage error in prediction of the test sets binned at, 20, 30, 40, and 50 milliseconds, for both the chaotic Lorenz system as well as the chaotic continuous-time RNN system. We note that for both systems, we observe decreases in mean absolute error percentages between the ground truth firing rates and the network's inferred firing rates as the discrete-time binning window is increased. Figure 3.23 illustrates this for the network trained to infer firing rates from the chaotic Lorenz dynamical spiking data, while Figure 3.24 does so for the network trained to infer firing rates from the chaotic continuous-time RNN spiking data.

3.5 DISCUSSION

In this work, we have developed and presented a novel data-driven adaptation of a variational autoencoder to sequential data in order to approximately infer smoothed firing rate dynamics underlying discretely-observed Poisson event spiking. In comparison to other approaches which cannot learn highly nonlinear dynamics, do not necessarily scale well with increased system dimensionality, and make use of complicated, highly tailored, multi-phase learning algorithms, we make use of multi-layered LSTM networks, with a single dense output layer with an exponential activation paired with commonly-used and highly flexible stochastic gradient descent algorithms, such as

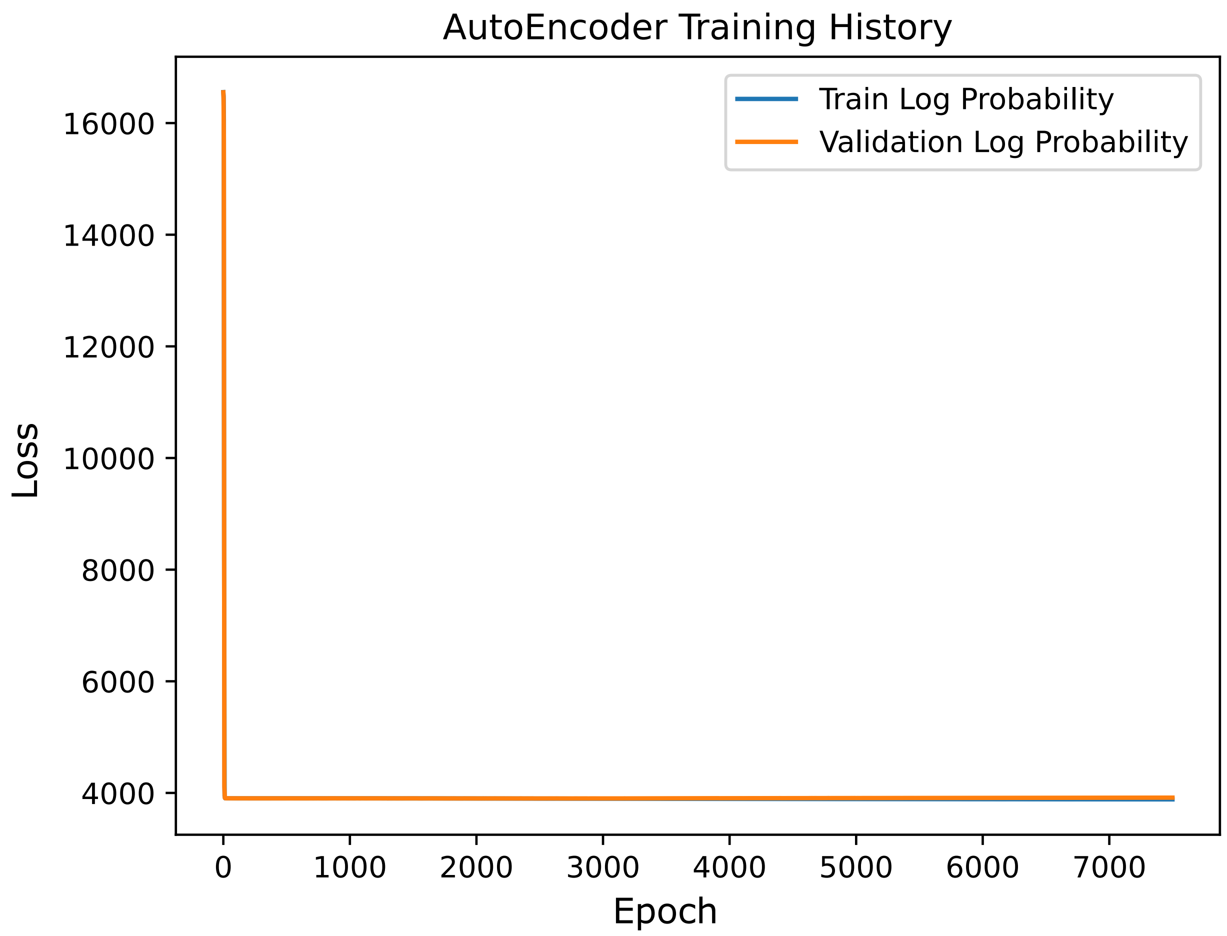


Figure 3.21: Training and validation loss for the SVAE inference network for inferring firing rates, for the network trained to infer chaotic continuous-time RNN binned spike trains of 20ms.

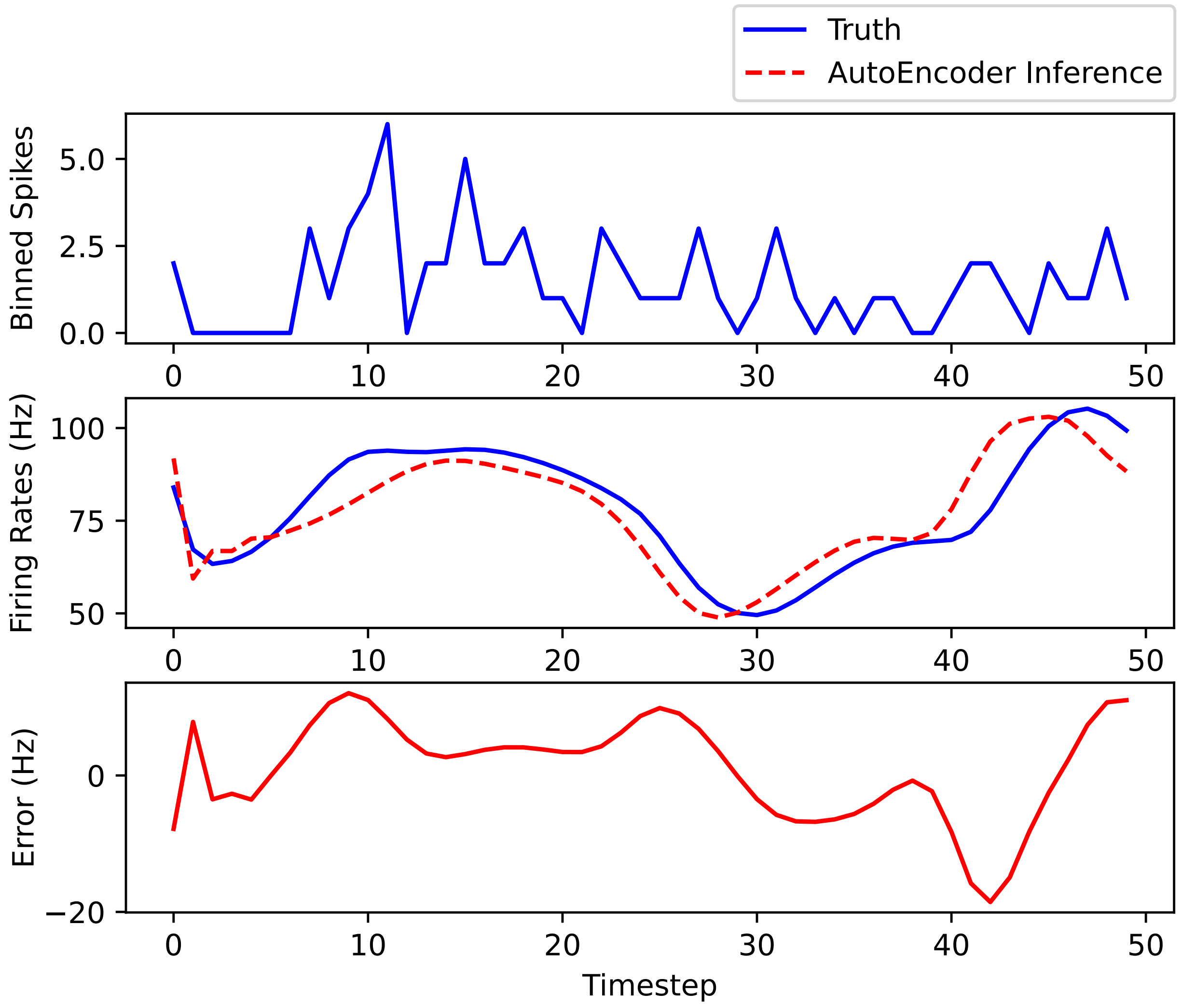


FIGURE 3.22: Example of firing rate inference for a randomly selected neuron, computed from a randomly selected test set example, for the network trained on chaotic continuous-time RNN neuronal spiking binned at 20ms. **Top:** Integer binned spike trains input from the neuron. **Middle:** Comparison of ground truth and inferred neuron firing rate underlying spikes shown above. **Bottom:** Error between prediction and ground truth, measured in Hz.

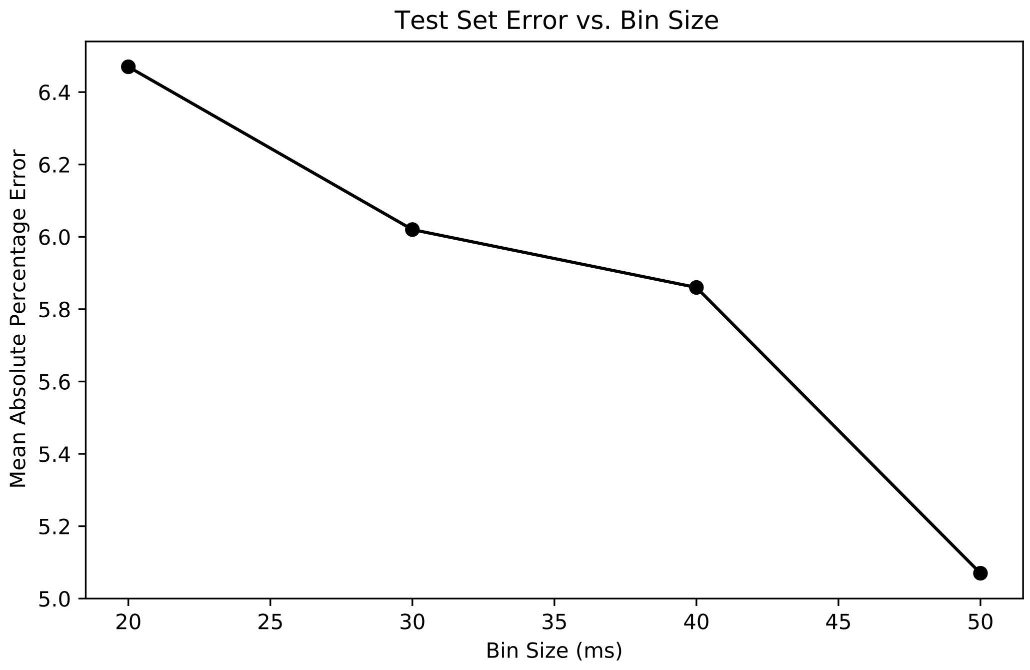


Figure 3.23: Mean absolute percentage error in inference of chaotic Lorenz firing rates for chaotic binned spiking activity plotted against bin size window, measured in milliseconds. We see that as bin size increases, the mean absolute percentage error of those predictions decreases.

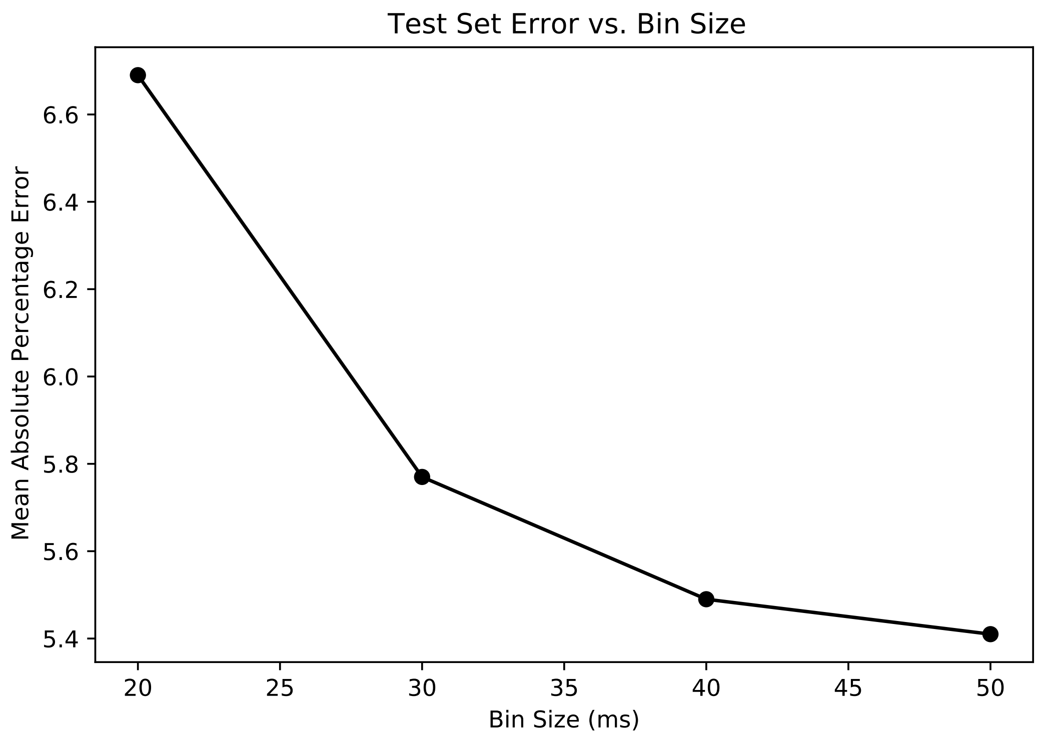


Figure 3.24: Mean absolute percentage error in inference of chaotic continuous-time RNN firing rates for chaotic binned spiking activity plotted against bin size window, measured in milliseconds. We see that as bin size increases, the mean absolute percentage error of those predictions decreases.

Adam. We demonstrate the efficacy of this approach to infer smoothed firing rates directly from discrete spiking data generated from simulations of a 50-neuron network of neurons with latent chaotic Lorenz dynamical behavior as well as a 50-neuron network with latent dynamics generated from a chaotic continuous-time RNN.

Through these simulations and training we have demonstrated that (1) the presented autoencoder architecture is capable of learning cohesive dynamical representations of smoothed firing rate data directly from discretely-observed event-based spiking data; and (2) that the predictive accuracy of the network increases as the discrete-time binning window is increased over an equivalent-time spiking sequence.

In both cases, the 20 millisecond binning window illustrated the limitation of this designed inference approach in inferring smoothed firing rates from the spiking data. At this binning window, degraded performance in prediction was observed with respect both to temporal features and raw magnitude. A possible reason for this may be the increased number of iterative predictions necessary due to the finer discrete time window at which data is binned. As the length of the predicted firing rate sequence increases, so does the necessary number of timesteps over which backpropagation must be carried out - this may result in weak gradients by the time that the iterative backpropagation reaches backwards to the initial timestep; however, this is merely a suspicion from related work in dynamical timeseries prediction with recurrent neural network architectures (88) (45) that merits further systematic investigation.

Another limitation of this approach as currently constructed is the assumption that all neurons within the synthetic systems devised above are fully observable. In practice, as the scale of the system increases, the ability to record spiking activity of each individual neuron within each circuit becomes increasingly less feasible, as neuronal circuits are comprised of many millions of neurons. There has been major progress made by prominent capital research ventures on this front, enabling recordings of many thousands of neurons in a potentially scalable fashion (89), but this remains a prohibitive and developing technology with much work to be done. Further work must be done to investigate the efficacy of this approach on partially observed spiking systems. This is of particular concern for real world applications, where data from in vitro and in vivo experiments are nearly always from partially observed systems.

Additionally, the topic of off-line training cost must be addressed. As is common to applications of RNNs to sequential datasets in sequence-to-sequence learning, as the length of the predictive horizon of the sequence increases, so does the offline computational cost of training said networks.

Finally, we must comment on the current limitations in distilling latent representations of the smoothed firing rate dynamics output from the SVAE. The current autoencoding-based method is not fully operational, as it is highly capable of recovering the high-dimensional smoothed firing rates, but is not capable of properly rediscovering the governing latent dynamics used to generate the high-dimensional firing rate data. This remains an outstanding challenge and poses a clear direction for future research endeavors.

In conclusion, our results showed that an increased discrete-time binning window in combination with our novel sequential variational autoencoder is highly capable of learning cohesive representations of firing rate dynamics underlying discretely-observed spiking data in a completely unsupervised variational autoencoding framework.

CHAPTER 4
SUMMARY AND DIRECTIONS FOR FUTURE WORK

4.1 SUMMARY

The approach of using data-driven neural networks to solve unique and otherwise intractable computational problems across a wide variety of fields has proved to be critical for the development of tools for modeling datasets previously thought infeasible. In this thesis, we have focused on the development and applications of novel deep neural network architectures for modeling and inferring neuronal dynamics at both the single-neuron and neuronal circuit level. Chapters 2 and 3 of this thesis cover the development of two separate ANN-centered techniques for modeling neuronal dynamics, in order to facilitate the development of computationally feasible optimal control systems, which are dependent upon a robust, multi-timestep predictive model of neuronal dynamics in order to properly function.

In Chapter 2, we present a novel deep LSTM architecture, which makes use of reversed-order sequence-to-sequence mapping, for use in predicting long-horizon simulated single-neuron responses to constant current stimulation. We used an experimentally validated, 9-dimensional Hodgkin-Huxley model with multiple dynamical bifurcations for use in generation of simulated data to be used in both training and evaluation of the network. These bifurcations resulted in three major dynamical behaviors: regular spiking, chaotic irregular bursting, and regular bursting. We systematically explored the impact of both architecture depth and length of predictive horizon used, and demonstrate that as the length of the predictive horizon of the LSTM used in prediction is increased so is the network's predictive accuracy. Simultaneously, we show that the on-line time required for inference also decreases as predictive horizon is increased, but at the expense of increased off-line training cost before use in inference. Additionally, we demonstrate that the use of reversed-order sequence-to-sequence mapping can increase accuracy of both early portions of predicted sequences, as well as improve overall network performance as a whole.

In conclusion, the developed LSTM architecture was demonstrated to be capable of learning and accurately predicting neuronal responses to externally applied input currents across a range of dynamical behaviors.

In Chapter 3, we develop a novel sequential adaptation of a variational autoencoder architecture for use in recovering smoothed Poisson firing rates directly from discrete, binned neuronal spiking data in an unsupervised learning framework and investigate the impact of binning size on the inference ability of the network. To do this, we created two synthetic chaotic datasets and quantitatively evaluated the performance of the networks on each. In both cases, chaotic firing rates were generated, binned spiking data was gathered through simulation of Poisson spiking activity from those firing rates, and the network was then allowed to discover firing rates that maximized the probability of the reconstruction of the data directly from the binned spiking activity. From this, performance of this approach was quantified in terms of mean absolute percentage error between the inferred firing rates and the ground truth firing rates used to generate the Poisson spiking data. We systematically investigate the impact of the binning window size on the performance of the network, and demonstrate that increased binning window size corresponds with decreased predictive error across both synthetic datasets. In conclusion, this approach is demonstrated to be capable of discovering cohesive firing rate dynamics directly from binned spiking behavior in an unsupervised framework.

4.2 FUTURE WORKS

In Chapter 2, we designed a novel deep LSTM architecture that was capable of learning long-horizon neuronal responses to externally applied inputs. It was demonstrated that as the predictive horizon of the network was increased, that so too was the predictive accuracy of the network. However, one cannot simply extend sequence lengths arbitrarily to many hundreds or even thousands of timesteps using traditional RNN architectures. Even the LSTM, which was specifically designed to mitigate the exploding gradient problem and allow for processing of lengthened sequences cannot extend beyond 200-300 timesteps at once without running into severe instability in training and degraded inference performance. Recent and highly novel sequence processing architectures, such as transformer networks with attention mechanisms (90), have completely revolutionized and reset the field of sequence-to-sequence mapping and sequential inference algorithms in natural language processing applications, as they do not suffer from degraded performance over sequence lengths considered infeasible for modern RNNs. Novel adaptations of these networks to become compatible with neuronal modeling problems is a challenge in an of itself, but it possesses the ability to once again revolutionize data-driven neuronal modeling techniques.

In Chapter 3, we designed an adaptation of a variational autoencoder for sequential data. This was used to infer underlying firing rate dynamics. However, there are multiple areas of potential improvement and expansion for this approach. To begin, there is no inclusion of external inputs to the neuronal network designed in the current iteration of this approach, which limits this autoencoding approach to a data-analysis tool. A novel method of accounting for inclusion of external inputs applied by a controller or experimentalist would greatly expand this approach in terms of applicability to in vivo and in vitro data, where external input is often applied and recorded. Secondly, this approach has thus far been used to recover smoothed firing rates in what is effectively a statistical inference problem - however, this approach could be modified for use in diagnostic applications, in which firing rates inferred by the network are fed into another sequential processing architecture to produce an estimate of pathological disease likelihood in a two-stage process. This can be visualized below in Figure 4.1.

Finally, there is work to be done towards distillation of low-dimensional latent firing-rate dynamics from the inferred firing rates output from the sequential variational autoencoder. Currently, while highly accurate reconstruction of the inferred firing rate dynamics has been achieved, there are significant challenges and deficiencies in recovery of the ground truth latent dynamics from them. Currently, this deterministic autoencoding approach to discover the latent dynamics makes use of non-recurrent multi-layer perceptrons; it is not unreasonable to suggest that use of recurrent architectures in lieu of multi-layer perceptrons could yield a more cohesive and realistic recreation of the latent firing dynamics, as RNNs must process sequences,

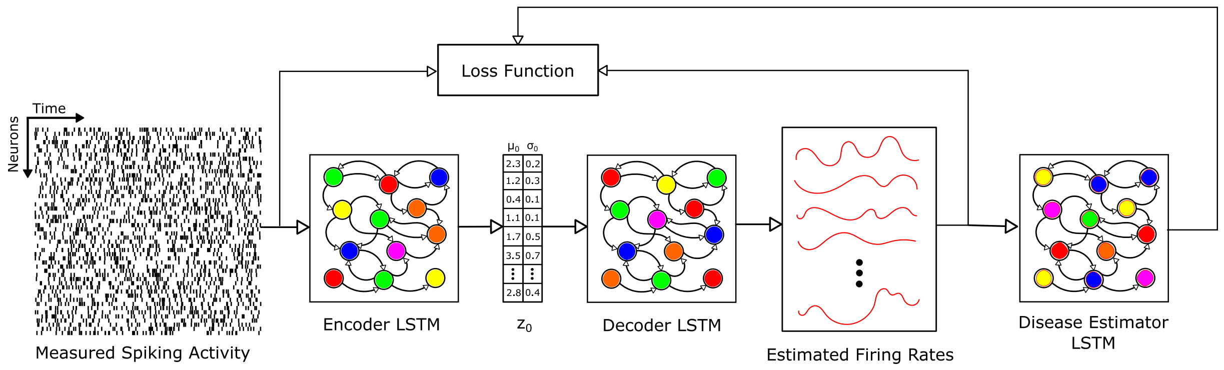


FIGURE 4.1: Illustration of potential diagnostic inference approach. Measured binned spikes are fed into the encoder LSTM network in array form. The bidirectional encoding LSTM returns a vector of Gaussian parameters, z_0, describing the initial condition of the network. A sample is drawn from the Gaussian distribution described the the parameters, and is fed into the decoder LSTM network, which returns the smoothed inferred firing rates of the neurons. These firing rates could then in theory be fed into a third LSTM network, which would return the probability of a certain disease, or the likelihood of a set of potential diagnoses. The firing rates and spikes are compared in a probabilistic loss function to return the likelihood that the inferred firing rates generated the original measured spiking activity, while the loss from the final diagnostic LSTM network may be some form of categorical accuracy loss. These losses would then be used to further optimize the network in iterative fashion via backpropagation.

instead of modeling static single-timestep dependencies as is done with the multi-layer perceptron. Additionally, these deterministic autoencoder structures are trained separately from the sequential autoencoder architecture. There is potential to greatly improve performance in this regard if the networks were to be jointly trained; however, the formulation of such a loss function, the relative weighting of each term expressed in the loss, and the implementation of such a function poses an arduous and difficult task.

Bibliography

[1] A. V. Herz, T. Gollisch, C. K. Machens, and D. Jaeger, "Modeling single-neuron dynamics and computations: a balance of detail and abstraction," *science*, vol. 314, no. 5796, pp. 80–85, 2006.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, 2008.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[7] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[8] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[9] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[10] B. Widrow *et al.*, *Adaptive" adaline" Neuron Using Chemical" memistors."*. 1960.

[11] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[15] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988.

[16] P. J. Werbos *et al.*, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[17] M. C. Mozer, "A focused backpropagation algorithm for temporal," *Backpropagation: Theory, architectures, and applications*, vol. 137, 1995.

[18] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[19] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[20] A. L. Hodgkin and A. F. Huxley, "The components of membrane conductance in the giant axon of loligo," *The Journal of physiology*, vol. 116, no. 4, p. 473, 1952.

[21] A. L. Hodgkin and A. F. Huxley, "Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo," *The Journal of physiology*, vol. 116, no. 4, p. 449, 1952.

[22] A. L. Hodgkin, A. F. Huxley, and B. Katz, "Measurement of current-voltage relations in the membrane of the giant axon of loligo," *The Journal of physiology*, vol. 116, no. 4, p. 424, 1952.

[23] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[24] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[25] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[26] D. Golomb, C. Yue, and Y. Yaari, "Contribution of persistent na+ current and m-type k+ current to somatic bursting in ca1 pyramidal cells: combined experimental and modeling study," *Journal of neurophysiology*, vol. 96, no. 4, pp. 1912–1926, 2006.

[27] K. D. Miller and F. Fumarola, "Mathematical equivalence of two common forms of firing rate models of neural networks," *Neural computation*, vol. 24, no. 1, pp. 25–31, 2012.

[28] D. Heeger, "Poisson model of spike generation," *Handout, University of Standford*, vol. 5, pp. 1–13, 2000.

[29] R. Salmelin, R. Hari, O. Lounasmaa, and M. Sams, "Dynamics of brain activation during picture naming," *Nature*, vol. 368, no. 6470, p. 463, 1994.

[30] M. D. Fox, A. Z. Snyder, J. L. Vincent, M. Corbetta, D. C. Van Essen, and M. E. Raichle, "The human brain is intrinsically organized into dynamic, anticorrelated functional networks," *Proceedings of the National Academy of Sciences*, vol. 102, no. 27, pp. 9673–9678, 2005.

[31] S. J. Kiebel, J. Daunizeau, and K. J. Friston, "A hierarchy of time-scales and the brain," *PLoS computational biology*, vol. 4, no. 11, p. e1000209, 2008.

[32] C. Siettos and J. Starke, "Multiscale modeling of brain dynamics: from single neurons and networks to mathematical tools," *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, vol. 8, no. 5, pp. 438–458, 2016.

[33] M. Breakspear, "Dynamic models of large-scale brain activity," *Nature neuroscience*, vol. 20, no. 3, p. 340, 2017.

[34] W. Gerstner and R. Naud, "How good are neuron models?," *Science*, vol. 326, no. 5951, pp. 379–380, 2009.

[35] S. Chen and S. Billings, "Neural networks for nonlinear dynamic system modelling and identification," *International journal of control*, vol. 56, no. 2, pp. 319–346, 1992.

[36] S. Purwar, I. Kar, and A. Jha, "Nonlinear system identification using neural networks," *IETE journal of research*, vol. 53, no. 1, pp. 35–42, 2007.

[37] J. G. Kuschewski, S. Hui, and S. H. Zak, "Application of feedforward neural networks to dynamical system identification and control," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 1, pp. 37–49, 1993.

[38] S. Pan and K. Duraisamy, "Long-time predictive modeling of nonlinear dynamical systems using neural networks," *Complexity*, vol. 2018, 2018.

[39] P. Gupta and N. K. Sinha, "Modeling robot dynamics using dynamic neural networks," *IFAC Proceedings Volumes*, vol. 30, no. 11, pp. 755–759, 1997.

[40] J. C. Patra, R. N. Pal, B. Chatterji, and G. Panda, "Identification of nonlinear dynamic systems using functional link artificial neural networks," *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, vol. 29, no. 2, pp. 254–262, 1999.

[41] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, IEEE, 2018.

[42] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.

[43] C. A. Bailer-Jones, D. J. MacKay, and P. J. Withers, "A recurrent neural network for modelling dynamical systems," *network: computation in neural systems*, vol. 9, no. 4, pp. 531–547, 1998.

[44] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control.," in *Robotics: Science and Systems*, Rome, Italy, 2015.

[45] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, 2013.

[46] N. Mohajerin and S. L. Waslander, "Multistep prediction of dynamic systems with recurrent neural networks," *IEEE transactions on neural networks and learning systems*, 2019.

[47] L. Lin, S. Gong, T. Li, and S. Peeta, "Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles," in *The Autonomous Vehicles Symposium*, vol. 2018, 2018.

[48] J. Gonzalez and W. Yu, "Non-linear system modeling using lstm neural networks," *IFAC-PapersOnLine*, vol. 51, no. 13, pp. 485–489, 2018.

[49] Y. Wang, "A new concept using LSTM neural networks for dynamic system identification," in *2017 American Control Conference (ACC)*, pp. 5324–5329, IEEE, 2017.

[50] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2213, p. 20170844, 2018.

[51] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[52] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," in *Advances in Neural Information Processing Systems*, pp. 1433–1443, 2018.

[53] C. Pandarinath, D. J. O'Shea, J. Collins, R. Jozefowicz, S. D. Stavisky, J. C. Kao, E. M. Trautmann, M. T. Kaufman, S. I. Ryu, L. R. Hochberg, *et al.*, "Inferring single-trial neural population dynamics using sequential auto-encoders," *Nature methods*, p. 1, 2018.

[54] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[55] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[56] E. C. McKiernan and D. F. Marrone, "CA1 pyramidal cells have diverse biophysical properties, affected by development, experience, and aging," *PeerJ*, vol. 5, p. e3836, 2017.

[57] J. Nowacki, H. M. Osinga, J. T. Brown, A. D. Randall, and K. Tsaneva-Atanasova, "A unified model of CA1/3 pyramidal cells: An investigation into excitability," *Progress in biophysics and molecular biology*, vol. 105, no. 1-2, pp. 34–48, 2011.

[58] K. A. Ferguson, C. Y. Huh, B. Amilhon, S. Williams, and F. K. Skinner, "Simple, biologically-constrained CA1 pyramidal cell models using an intact, whole hippocampus context," *F1000Research*, vol. 3, 2014.

[59] P. Poirazi, T. Brannon, and B. W. Mel, "Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell," *Neuron*, vol. 37, no. 6, pp. 977–987, 2003.

[60] M. Royeck, M.-T. Horstmann, S. Remy, M. Reitze, Y. Yaari, and H. Beck, "Role of axonal NaV1. 6 sodium channels in action potential initiation of CA1 pyramidal neurons," *Journal of neurophysiology*, vol. 100, no. 4, pp. 2361–2380, 2008.

[61] Y. Katz, V. Menon, D. A. Nicholson, Y. Geinisman, W. L. Kath, and N. Spruston, "Synapse distribution suggests a two-stage model of dendritic integration in CA1 pyramidal neurons," *Neuron*, vol. 63, no. 2, pp. 171–177, 2009.

[62] D. Bianchi, A. Marasco, A. Limongiello, C. Marchetti, H. Marie, B. Tirozzi, and M. Migliore, "On the mechanisms underlying the depolarization block in the spiking dynamics of CA1 pyramidal neurons," *Journal of computational neuroscience*, vol. 33, no. 2, pp. 207–225, 2012.

[63] A. Marasco, A. Limongiello, and M. Migliore, "Fast and accurate low-dimensional reduction of biophysically detailed neuron models," *Scientific reports*, vol. 2, no. 928, pp. 1–7, 2012.

[64] Y. Kim, C.-L. Hsu, M. S. Cembrowski, B. D. Mensh, and N. Spruston, "Dendritic sodium spikes are required for long-term potentiation at distal synapses on hippocampal pyramidal neurons," *Elife*, vol. 4, pp. (e06414) 1–30, 2015.

[65] M. J. Bezaire, I. Raikov, K. Burk, D. Vyas, and I. Soltesz, "Interneuronal mechanisms of hippocampal theta oscillations in a full-scale model of the rodent CA1 circuit," *Elife*, vol. 5, pp. (e18566) 1–106, 2016.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[67] R. Naud, N. Marcille, C. Clopath, and W. Gerstner, "Firing patterns in the adaptive exponential integrate-and-fire model," *Biological cybernetics*, vol. 99, no. 4-5, p. 335, 2008.

[68] D. N. Mastronarde, "Correlated firing of retinal ganglion cells," *Trends in neurosciences*, vol. 12, no. 2, pp. 75–80, 1989.

[69] M. N. Shadlen and W. T. Newsome, "The variable discharge of cortical neurons: implications for connectivity, computation, and information coding," *Journal of neuroscience*, vol. 18, no. 10, pp. 3870–3896, 1998.

[70] J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. Chichilnisky, and E. P. Simoncelli, "Spatio-temporal correlations and visual signalling in a complete neuronal population," *Nature*, vol. 454, no. 7207, pp. 995–999, 2008.

[71] S. Shinomoto, K. Shima, and J. Tanji, "Differences in spiking patterns among cortical neurons," *Neural computation*, vol. 15, no. 12, pp. 2823–2842, 2003.

[72] M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy, "Neural population dynamics during reaching," *Nature*, vol. 487, no. 7405, pp. 51–56, 2012.

[73] D. Kobak, W. Brendel, C. Constantinidis, C. E. Feierstein, A. Kepecs, Z. F. Mainen, X.-L. Qi, R. Romo, N. Uchida, and C. K. Machens, "Demixed principal component analysis of neural population data," *Elife*, vol. 5, p. e10989, 2016.

[74] K. V. Shenoy, M. Sahani, and M. M. Churchland, "Cortical control of arm movements: a dynamical systems perspective," *Annual review of neuroscience*, vol. 36, 2013.

[75] C. Pandarinath, V. Gilja, C. H. Blabe, P. Nuyujukian, A. A. Sarma, B. L. Sorice, E. N. Eskandar, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "Neural population dynamics in human motor cortex during movements in people with als," *Elife*, vol. 4, p. e07436, 2015.

[76] F. Carnevale, V. de Lafuente, R. Romo, O. Barak, and N. Parga, "Dynamic control of response criterion in premotor cortex during perceptual detection under temporal uncertainty," *Neuron*, vol. 86, no. 4, pp. 1067–1077, 2015.

[77] J. M. Shine, L. J. Hearne, M. Breakspear, K. Hwang, E. J. Müller, O. Sporns, R. A. Poldrack, J. B. Mattingley, and L. Cocchi, "The low-dimensional neural architecture of cognitive complexity is related to activity in medial thalamic nuclei," *Neuron*, vol. 104, no. 5, pp. 849–855, 2019.

[78] J. A. Goldberg, U. Rokni, and H. Sompolinsky, "Patterns of ongoing activity and the functional architecture of the primary visual cortex," *Neuron*, vol. 42, no. 3, pp. 489–500, 2004.

[79] B. M. Yu, J. P. Cunningham, G. Santhanam, S. Ryu, K. V. Shenoy, and M. Sahani, "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity," *Advances in neural information processing systems*, vol. 21, pp. 1881–1888, 2008.

[80] J. Luttinen and A. Ilin, "Variational gaussian-process factor analysis for modeling spatio-temporal data," in *Advances in neural information processing systems*, pp. 1177–1185, 2009.

[81] C. D. Harvey, P. Coen, and D. W. Tank, "Choice-specific sequences in parietal cortex during a virtual-navigation decision task," *Nature*, vol. 484, no. 7392, pp. 62–68, 2012.

[82] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," *arXiv preprint arXiv:1811.09975*, 2018.

[83] Y. Wang, B. Dai, G. Hua, J. Aston, and D. Wipf, "Recurrent variational autoencoders for learning nonlinear generative models in the presence of outliers," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 6, pp. 1615–1627, 2018.

[84] J.-T. Chien and C.-W. Wang, "Variational and hierarchical recurrent autoencoder," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3202–3206, IEEE, 2019.

[85] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators," in *International Conference on Artificial Neural Networks*, pp. 632–640, Springer, 2006.

[86] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[87] Y. Zhao and I. M. Park, "Variational latent gaussian process for recovering single-trial dynamics from population spike trains," *Neural computation*, vol. 29, no. 5, pp. 1293–1316, 2017.

[88] B. Plaster and G. Kumar, "Data-driven predictive modeling of neuronal dynamics using long short-term memory," *Algorithms*, vol. 12, no. 10, p. 203, 2019.

[89] E. Musk *et al.*, "An integrated brain-machine interface platform with thousands of channels," *Journal of medical Internet research*, vol. 21, no. 10, p. e16194, 2019.

[90] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, pp. 5998–6008, 2017.