Deep Learning Models for Irregularly Sampled and Incomplete Time Series

Satya Narayan Shukla
University of Massachusetts Amherst

DEEP LEARNING MODELS FOR IRREGULARLY SAMPLED AND INCOMPLETE TIME SERIES

A Dissertation Presented

by

SATYA NARAYAN SHUKLA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2021

College of Information and Computer Sciences

DEEP LEARNING MODELS FOR IRREGULARLY SAMPLED AND INCOMPLETE TIME SERIES

A Dissertation Presented

by

SATYA NARAYAN SHUKLA

Approved as to style and content by:

Benjamin M. Marlin, Chair

Madalina Fiterau Brostean, Member

Dan Sheldon, Member

Yan Liu, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

DEDICATION

In loving memory of my mother.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Benjamin Marlin for his unwavering support and for providing great guidance throughout my journey here. He has always been open to new ideas and provided valuable advice and criticism when it was needed. His high professional standards, penchant for rigor, and attention to detail have gradually shaped my logical thinking and problem solving skills. I could not have asked for a better mentor. Thank you, Ben!

I also thank my dissertation committee members Madalina Fiterau Brostean, Dan Sheldon, and Yan Liu for generously offering their time and for their insightful comments, valuable suggestions, and stimulating discussions during my proposal and dissertation defenses. I also thank Hong Yu for being an amazing advisor on my synthesis project.

I would also like to thank all the people who hosted and mentored me during internships at Microsoft Research, Bosch Center for AI, and Facebook: Kenneth Tran, Devin Willmott, Anit Kumar Sahu, Zico Kolter, and Erik Brinkman. I am also thankful to the lab members – Steve Li, Meet Vadera, Colin Samplawski, and Conrad Holtsclaw for fruitful discussions on several research problems throughout this journey.

I am grateful to have met many wonderful friends and housemates during my time here. Raghavendra Addanki, in particular, has been an exceptional friend, colleague, and housemate. He was also the first person I met when I came to Amherst. Rajvi Kapadia, Anirudh Sabnis, Sneha Aenugu, Niketh Murali, and Vikash Verma have also been great colleagues and amazing housemates. Utkarsh Srivastava, Rishi Mody, Nidhi Mundra, Ananya Suraj, and Neha Yadav have been fabulous friends. I have shared countless hours of meals, games, road trips, laughs, and discussions with them. Other wonderful friends with whom I have shared many great memories are Sainyam Galhotra, Nishant Yadav, Virat Shejwalkar, Manish Motwani, and Tomas Geffner. Thank all of you for making the past years memorable.

Finally, I would like to thank my brother, Krishna, for always encouraging me to shoot for the stars, my niece, Anshika, for always reminding me to have fun, and my sister-in-law, Chandani, for being there for me throughout this process. I deeply thank my father, Ram Kumar Shukla, for his unconditional trust in me and for providing endless love and encouraging me to reach for my dreams. I would like to thank my late grandparents for always believing in me. Last but definitely not least, I would like to thank my late mother, Prem Lata Shukla for her unlimited patience, love, encouragement, and unfaltering care for my happiness and well-being. Although she is no longer with us, she has been a constant source of inspiration to me. This one is for you.

Funding Acknowledgments: The work described in this thesis is partially supported by the National Science Foundation under Grant No. IIS-1350522 and the National Institutes of Health under award numbers 5U01CA229445 and 1P41EB028242.

ABSTRACT

DEEP LEARNING MODELS FOR IRREGULARLY SAMPLED AND INCOMPLETE TIME SERIES

SEPTEMBER 2021

SATYA NARAYAN SHUKLA

B.Tech., INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

M.Tech., INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Benjamin M. Marlin

Irregularly sampled time series data arise naturally in many application domains including biology, ecology, climate science, astronomy, geology, finance, and health. Such data present fundamental challenges to many classical models from machine learning and statistics. The first challenge with modeling such data is the presence of variable time gaps between the observation time points. The second challenge is that the dimensionality of the inputs can be different for different data cases. This occurs naturally due to the fact that different data cases are likely to include different numbers of observations. The third challenge is that different irregularly sampled instances have observations recorded at different times. This results in a lack of temporal alignment across data cases. There could also be a lack of alignment of observation time points across different dimensions in the same multivariate time series.

These features of irregularly sampled time series data invalidate the assumption of a coherent fully-observed fixed-dimensional feature space that underlies many basic supervised and unsupervised learning models.

In this thesis, we focus on the development of deep learning models for the problems of supervised and unsupervised learning from irregularly sampled time series data. We begin by introducing a computationally efficient architecture for whole time series classification and regression problems based on the use of a novel deterministic interpolation-based layer that acts as a bridge between multivariate irregularly sampled time series data instances and standard neural network layers that assume regularly-spaced or fixed-dimensional inputs. The architecture is based on the use of a radial basis function (RBF) kernel interpolation network followed by the application of a prediction network. Next, we show how the use of fixed RBF kernel functions can be relaxed through the use of a novel attention-based continuous-time interpolation framework. We show that using attention to learn temporal similarity results in improvements over fixed RBF kernels and other recent approaches in terms of both supervised and unsupervised tasks. Next, we present a novel deep learning framework for probabilistic interpolation that significantly improves uncertainty quantification in the output interpolations. Furthermore, we show that this framework is also able to improve classification performance. As our final contribution, we study fusion architectures for learning from text data combined with irregularly sampled time series data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

An irregularly sampled time series is a sequence of time-value pairs with non-uniform intervals between successive time points. Such time series data naturally occur in domains where the observation process is constrained to a degree that prohibits regular observation of continuously varying phenomena. This occurs in a number of scientific and industrial domains and is also a prominent feature of some types of data in the health domain (Eckner, 2014).

For example, observational studies of free-living animals in biology and ecology inevitably lack data points due to movement of subjects, weak transmitter reception, or poor weather and lighting conditions that hinder observation (Ruf, 1999). In astronomy, measurements of properties such as the spectra of celestial objects are taken at times determined by seasonal and weather conditions as well as the availability of time on required instruments (Scargle, 1982). In climate science, paleoclimate time series data are often sampled irregularly due to the difficulty in obtaining historical information (Schulz and Stattegger, 1997). In electronic health records data, vital signs and other measurements are recorded at time points determined by a number of factors that may include the type of measurement, the patient's state of health, and the availability of clinical staff (Marlin et al., 2012). Irregularly sampled time series data also commonly occur in a range of other areas with similar complex observation processes including geology (Rehfeld et al., 2011), finance (Manimaran et al., 2006), economics (Kling and Bessler, 1985), meteorology (Mudelsee, 2002), and traffic analysis (Ye et al., 2010).

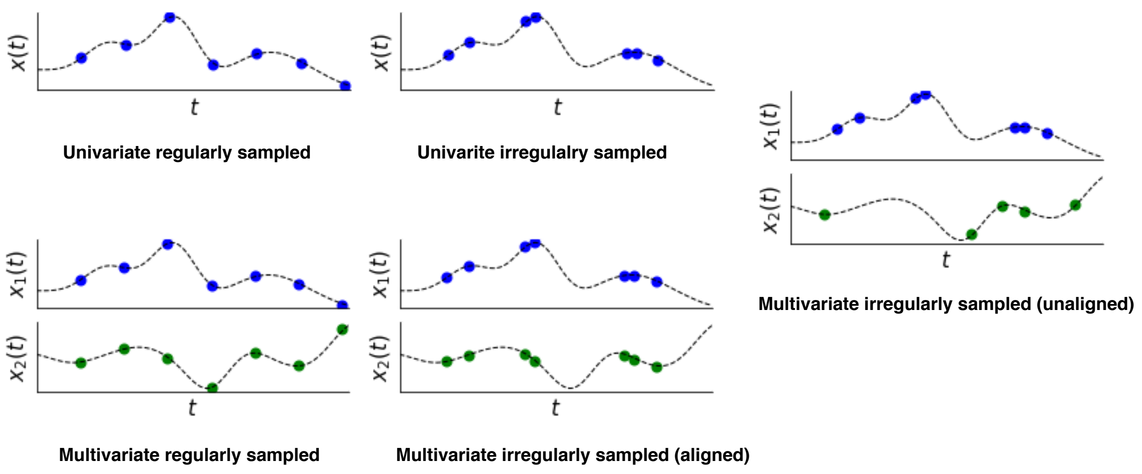


Figure 1.1: Illustration of regularly and irregularly sampled univariate and two-dimensional multivariate time series. Univariate irregularly sampled time series data are characterized by variable time intervals between observations. Multivariate irregularly sampled time series are also characterized by variable time intervals between observations, but can have aligned or unaligned observation times across dimensions. When observation time points are unaligned across dimensions, different dimensions also often contain different numbers of observations. Finally, while this figure only illustrates single univariate or multivariate time series, it is important to note that in data sets containing multiple univariate or multivariate time series as data cases, the observation times in different data cases are also typically unaligned and different data cases often also have different numbers of observations.

This thesis is motivated by problems in the analysis of electronic health records (EHRs). While the volume of electronic health records (EHR) data continues to grow, it remains rare for hospital systems to capture dense physiological data streams, even in the data-rich intensive care unit setting. Instead, it remains more common for the physiological time series data in electronic health records to be both sparse and irregularly sampled.

Irregularly sampled time series data (as shown in Figure 1.1) present fundamental challenges to many classical models from machine learning and statistics. To illustrate these challenges, consider the case of a supervised learning task where a model takes as input an irregularly sampled time series and must predict a scalar output. To learn such a model, we assume access to a data set $\mathcal{D}$ where each instance is a tuple

$(\mathbf{s}_n, y_n)$. Here $\mathbf{s}_n$ represents the irregularly samples time series and y_n represents the corresponding prediction target. There are a number of challenges in modeling such data:

(i) The first challenge with modeling such data is the presence of variable time gaps between the observation time points.

(ii) The second challenge is that the dimensionality of the inputs $\mathbf{s}_n$ can be different for different data cases n. This occurs naturally due to the fact that different data cases are likely to include different numbers of observations.

(iii) The third challenge is that even if all data cases contain the same number of observations, we would expect different irregularly sampled instances to have observations recorded at different times. This results in a lack of temporal alignment across data cases. For multivariate irregularly sampled time series data, we may also have a situation where different variables (or channels) within the same multivariate time series are observed at a different collection of time points.

These features of irregularly sampled time series data invalidate the assumption of a coherent fixed-dimensional feature space that underlies most basic supervised and unsupervised learning models including K-nearest neighbours (Altman, 1992), decision trees (Quinlan, 1986), linear and logistic regression (Hastie et al., 2001; Hosmer Jr et al., 2013), linear, polynomial and radial basis function kernel support vector machines (Cortes and Vapnik, 1995), multi-layer perceptrons (Hastie et al., 2001), K-means (Lloyd, 1982), mixtures models (with standard component distributions) (Mclachlan and Basford, 1988), factor analysis (Harman, 1976), PCA (Hotelling, 1933), autoencoders (Kramer, 1991), and more.

In this thesis, we focus on the development of deep learning models for the problems of supervised and unsupervised learning from irregularly sampled time series data. Below we provide an overview of the rest of the thesis and highlight our contributions.

1.1 Thesis Outline and Contributions

In **Chapter 2**, we present a survey of prior specialized models and architectures for learning from irregularly sampled multivariate time series data. The primary contributions of this chapter are listed below.

- We identify three underlying data representation for multivariate irregularly sampled time series that, while equivalent, expose different properties and suggest different approaches to modeling.
- Another categorization that we focus on is the set of inference tasks that a given approach is designed to solve. Carefully specifying different inference tasks is necessary to properly categorize models in terms of the tasks they can perform.
- We define modeling primitives to be the basic building blocks leveraged in larger and more complex models. We identify several such modeling primitives that specifically provide the interface between irregularly sampled time series data and more standard model components.
- We describe a large number of specific models and methods with respect to the model primitives they build on, the tasks they aim to solve, and their relative strengths and weaknesses.

In **Chapter 3**, we present a computationally efficient model architecture for supervised learning with multivariate sparse and irregularly sampled data: *Interpolation-Prediction Networks.* The architecture is based on the use of RBF kernel-based interpolation layers organized into an interpolation network, followed by the application of a prediction network that can leverage any standard deep learning model. Experiments show that this approach outperforms a range of baseline and recently proposed models on both classification and regression tasks with multivariate irregularly sampled time series. This approach achieved state-of-the-art results at the time of publication. The primary contributions of this chapter are listed below.

- We propose a novel method to handle irregularly sampled time series directly without any adhoc preprocessing.
- Our approach allows computing an explicit multi-timescale representation of an irregularly sampled time series.
- Our approach is fully modular that any standard deep learning network for fixed-length inputs can be used as the prediction network.

In **Chapter 4**, we show how the use of fixed RBF kernel functions can be relaxed through the use of a novel attention-based continuous-time interpolation framework: *Multi-Time Attention Networks*. Multi-Time Attention Networks are based on a time attention mechanism coupled with a learned continuous-time embedding function that replaces the use of a fixed similarity kernel. We show that using attention to learn temporal similarity provides significantly more representational flexibility and results in improvements over fixed RBF kernels and other recent approaches in terms of both supervised and unsupervised tasks. The primary contributions of this chapter are listed below.

- We provide a more flexible approach to modeling multivariate, sparse and irregularly sampled time series data (including irregularly sampled time series of partially observed vectors) by leveraging a time attention mechanism to learn temporal similarity from data instead of using fixed kernels.
- Our approach uses a temporally distributed latent representation to better capture local structure in time series data.

- Our approach provides interpolation and classification performance that is as good as current state-of-the-art methods or better, while providing significantly reduced training times.

In **Chapter 5**, we point out that Multi-Time Attention networks are not able to effectively reflect uncertainty due to variable input sparsity. In fact, they produce overly confident and incorrect imputations when faced with long segments with no observations. To address these issues, we present a novel encoder-decoder architecture for multivariate probabilistic time series interpolation that we refer to as the *Heteroscedastic Temporal Variational Autoencoder* or HeTVAE. HeTVAE consists of an input sparsity-aware encoder, parallel deterministic and probabilistic pathways for propagating input uncertainty to the output, and a heteroscedastic output distribution to represent variable uncertainty in the output interpolations. The primary contributions of this chapter are listed below.

- We introduce a novel Uncertainty-Aware Multi-Time Attention Network layer that encodes information about input uncertainty due to variable sparsity.
- We propose an augmented training objective to combat the presence of additional local optima that arise from the use of the heteroscedastic output structure.
- Our results show that the proposed model significantly improves uncertainty quantification in the output interpolations as evidenced by significantly improved log likelihood scores compared to several baselines and state-of-the-art methods.

In **Chapter 6**, we present architectures for combining time series and text data. Specifically, we show how we can leverage the content in text data and fuse the information they contain with irregularly sampled time series data. We build on the previously described frameworks for modeling sparse and irregularly sampled time series data. We study several embedding-based methods for representing the text data. We present fusion architectures that combine the different frameworks for time series with the embedding-based models for text data. Finally, we explore the predictive value of irregularly sampled time series data compared to text data on a real-world data set.

In **Chapter 7**, we conclude the thesis and discuss future directions for this work.

Bibliographic note: Chapter 2 is mainly based on Shukla and Marlin (2020b). Chapter 3 is based on Shukla and Marlin (2019). Chapter 4 is based on Shukla and Marlin (2021a). Chapter 5 is based on Shukla and Marlin (2021b). Chapter 6 is based on Shukla and Marlin (2020a).

CHAPTER 2
BACKGROUND AND RELATED WORK

There has been significant progress over the last decade on developing specialized models and architectures for learning from irregularly sampled multivariate time series data within the machine learning community. This chapter reviews many of these approaches with a focus on categorizing methods in terms of three key properties including the underlying data representation they use, the fundamental mechanism they use to accommodate irregular sampling, and the types of machine learning tasks to which they can be applied.

We identify three primary underlying data representations for irregularly sampled time series that we refer to as *series-based*, *vector-based*, and *set-based* representations. In the series-based representation, a multivariate irregularly sampled time series is viewed as consisting of a collection of univariate time series, each with its own collection of observation times and values. In the vector-based representation, a multivariate time series is viewed as a time series of vector-valued observations. When observations for different dimensions are not temporally aligned, the result is missing dimensions in the vector-valued observations. The set-based representation views a multivariate irregularly sampled time series as a set of tuples of the form (t, d, x) where t is a time point, d is a dimension, and x is the observed value of dimension d at time t. As we will see, different approaches are based on different of these representations, leading to methods with different capabilities and limitations.

Different approaches also leverage different "modeling primitives" for accommodating irregular sampling. We define modeling primitives to be the basic building

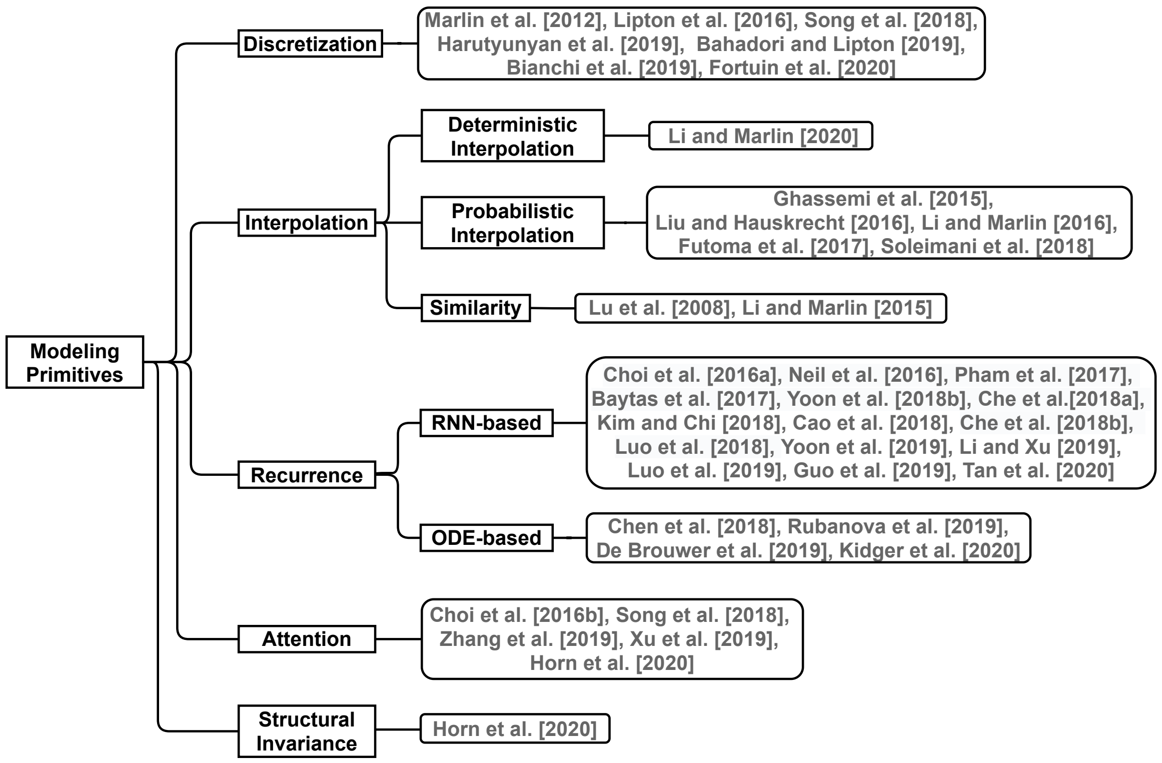

Modeling Primitives
Discretization
Marlin et al. [2012], Lipton et al. [2016], Song et al. [2018], Harutyunyan et al. [2019], Bahadori and Lipton [2019], Bianchi et al. [2019], Fortuin et al. [2020]
Interpolation
Deterministic Interpolation
Li and Marlin [2020]
Probabilistic Interpolation
Ghassemi et al. [2015], Liu and Hauskrecht [2016], Li and Marlin [2016], Futoma et al. [2017], Soleimani et al. [2018]
Similarity
Lu et al. [2008], Li and Marlin [2015]
Recurrence
RNN-based
Choi et al. [2016a], Neil et al. [2016], Pham et al. [2017], Baytas et al. [2017], Yoon et al. [2018b], Che et al.[2018a], Kim and Chi [2018], Cao et al. [2018], Che et al. [2018b], Luo et al. [2018], Yoon et al. [2019], Li and Xu [2019], Luo et al. [2019], Guo et al. [2019], Tan et al. [2020]
ODE-based
Chen et al. [2018], Rubanova et al. [2019], De Brouwer et al. [2019], Kidger et al. [2020]
Attention
Choi et al. [2016b], Song et al. [2018], Zhang et al. [2019], Xu et al. [2019], Horn et al. [2020]
Structural Invariance
Horn et al. [2020]

Figure 2.1: This figure illustrates a taxonomy of methods based on modeling primitives. We define modeling primitives as the fundamental approach that methods use to accommodate irregular sampling. We identify five high-level modeling primitives including discretization, interpolation, recurrence, attention and structural invariance.

blocks or modules leveraged in larger and more complex models. We identify several such modeling primitives that specifically provide the interface between irregularly sampled time series data and more standard model components. These modeling primitives fall into several categories including approaches based on temporal discretization, interpolation, similarity, recurrence, attention and structural invariance. We categorize approaches in terms of their use of such modeling primitives, which can sometimes be obscured in larger and more complex models. In Figure 2.1, we provide a categorization of different approaches under a taxonomy based on modeling primitives.

The third categorization that we focus on is the set of inference tasks that a given approach is designed to solve. These tasks include detection, prediction, filtering, smoothing, interpolation, and forecasting. Some approaches can be applied to multiple of these tasks, while others can not. Understanding the range of problems a given approach can be applied to in a valid way is obviously important, but can again be unclear for some larger and more complex models.

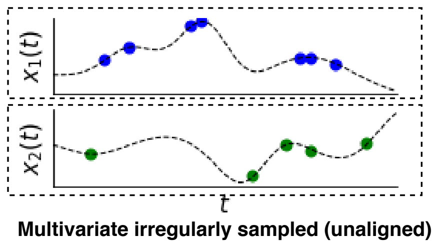
The rest of this chapter is organized as follows. In Section 2.1 we describe the categorization of representations for irregularly sampled time series. In Section 2.2, we define modeling tasks for irregularly sampled time series. In Section 2.3, we define modeling primitives for irregularly sampled time series. In Sections 2.4 to 2.8, we present a detailed discussion of specific models and approaches organized by their primary modeling primitives including temporal discretization, interpolation, similarity, recurrence, attention and structural invariance. In Section 2.10 we discuss data sets commonly used to evaluate models for irregularly sampled time series.

2.1 Data Representations for Irregularly Sampled Time Series

There are several possible data representations for multivariate irregularly sampled time series. While equivalent, they expose different properties and suggest different approaches to modeling. In all cases we will assume that a data set contains N data cases and that each data case consists of observations of D different variables through time. We being by describing what we will refer to as the series-based representation followed by the vector-based representation and the set-based representation.

2.1.1 Series-Based Representation

As shown in Figure 2.2, the series-based representation of a multivariate irregularly sampled time series data set views the data for a single data case n as a collection of

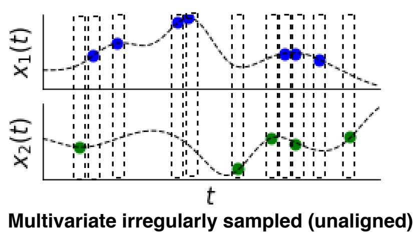


$$\begin{aligned}
\mathbf{s} &= [\mathbf{s}_1, \mathbf{s}_2] \\
\mathbf{s}_1 &= (\mathbf{t}_1, \mathbf{x}_1) \\
\mathbf{t}_1 &= [t_{11}, \ldots, t_{L_1 1}] \\
\mathbf{x}_1 &= [x_{11}, \ldots, x_{L_1 1}] \\
\mathbf{s}_2 &= (\mathbf{t}_2, \mathbf{x}_2) \\
\mathbf{t}_2 &= [t_{12}, \ldots, t_{L_2 2}] \\
\mathbf{x}_2 &= [x_{12}, \ldots, x_{L_2 2}]
\end{aligned}$$

Figure 2.2: Illustration of the series-based representation for a two-dimensional multivariate irregularly sampled time series. In the series-based representation, a d-dimensional multivariate irregularly sampled time series $\mathbf{s} = [\mathbf{s}_1, ..., \mathbf{s}_D]$ is represented as a collection of univariate irregularly sampled time series, one per dimension. Here $\mathbf{s}_d = (\mathbf{t}_d, \mathbf{x}_d)$ indicates the time series for dimension d. $\mathbf{t}_d$ indicates the collection of time points with observed values for dimension d while $\mathbf{x}_d$ indicates the corresponding collection of observed values.

D univariate irregularly sample time series with one item in the collection for each of the D dimensions. We define L_{dn} to be the number of observations of variable d for data case n. We define $\mathbf{t}_{dn} = [t_{1dn}, ..., t_{L_{dn}dn}]$ to be the collection of time points at which variable d is observed for data case n. We define $\mathbf{x}_{dn} = [x_{1dn}, ..., x_{L_{dn}dn}]$ to be the corresponding collection of observed values. We assume that the data are in time order (i.e., $t_{idn} < t_{jdn}$ for $i < j$). The data for time series n and variable d is then a univariate irregularly sampled time series $\mathbf{s}_{dn} = (\mathbf{t}_{dn}, \mathbf{x}_{dn})$. We define $\mathbf{s}_n = [\mathbf{s}_{1n}, ..., \mathbf{s}_{Dn}]$ to be the complete multivariate irregularly sampled time series for data case n.

We note that in this representation, there is no missing data. We represent only the observations available for each dimension. Further, this structure exposes the fact that in the fully general case of multivariate irregular sampling, different dimensions can be observed at different collections of time points with different total numbers of observations for the same data case. We also note that this representation can be wasteful in scenarios where all variables are observed at each time point. This motivates the vector-based representation, which we introduce next.



$$\mathbf{s} = (\mathbf{t}, \mathbf{x}, \mathbf{r})$$
$$\mathbf{t} = [t_1, \ldots, t_L]$$
$$\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_L]$$
$$\mathbf{r} = [\mathbf{r}_1, \cdots, \mathbf{r}_L]$$
$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} \quad x_{id} = \begin{cases} x_{id}, & \text{if } x_{id} \text{ is observed} \\ \text{NA}, & \text{otherwise} \end{cases}$$
$$\mathbf{r}_i = \begin{bmatrix} r_{i1} \\ r_{i2} \end{bmatrix} \quad r_{id} = \begin{cases} 1, & \text{if } x_{id} \text{ is observed} \\ 0, & \text{otherwise} \end{cases}$$

Figure 2.3: Illustration of the vector-based representation for multivariate irregularly sampled time series in the case of two dimensions with unaligned observation times. In this representation, there is a single collection of time points $\mathbf{t}$. At each time point t_i, we define a D-dimensional vector-valued observation $\mathbf{x}_i$. In the general case, not all dimensions of $\mathbf{x}_i$ are observed, leading to the need to explicitly represent which dimensions are observed and which are missing. Following Little and Rubin (2014), we introduce a D-dimensional binary response indicator vector $\mathbf{r}_i$ at each time point t_i to indicate which dimensions are observed and which are missing. The complete representation for the time series is thus $\mathbf{s} = (\mathbf{t}, \mathbf{x}, \mathbf{r})$ where $\mathbf{t}$ is the collection of observation times, $\mathbf{x}$ is the collection of vector-valued observations (with missing values), and $\mathbf{r}$ is the collection of response indicators.

2.1.2 Vector-Based Representation

As shown in Figure 2.3, the vector-based representation of a multivariate irregularly sampled time series data set views the data for a single data case n as a single series of D dimensional vectors. We define L_n to be the number of time points with observations for data case n. We define $\mathbf{t}_n = [t_{1n}, ..., t_{L_n n}]$ to be the collection of time points with observations for data case n. As in the series-based representation, the data are assumed to be in time order (i.e., $t_{in} < t_{jn}$ for $i < j$). Next, we define $\mathbf{x}_n = [\mathbf{x}_{1n}, ..., \mathbf{x}_{L_n n}]$ to be the corresponding collection of D-dimensional vector-valued observations. We again let x_{idn} be the observed value of dimension d at time point t_{in}. We define $\mathbf{s}_n = (\mathbf{t}_n, \mathbf{x}_n)$ to be the irregularly sampled time series for data case n. Under this representation, different data cases n can still have different observation time points as well as different numbers of observation time points. However, the observations across dimensions within a single data case are assumed to be aligned in time.

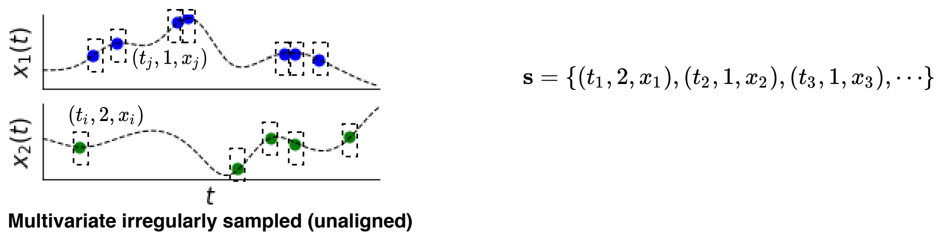

$x_1(t)$
$(t_j, 1, x_j)$
$x_2(t)$
$(t_i, 2, x_i)$
t
Multivariate irregularly sampled (unaligned)
$\mathbf{s} = \{(t_1, 2, x_1), (t_2, 1, x_2), (t_3, 1, x_3), \cdots\}$

Figure 2.4: Illustration of the set-based representation for multivariate irregularly sampled time series in the case of two dimensions. In this representation, a D-dimensional multivariate irregularly sampled time series is represented as a set of (time, dimension, value) tuples, one for each observation.

When the vector $\mathbf{x}_{in}$ is not fully observed at time t_{in}, the vector-based representation results in the need to represent missing data. Following Little and Rubin (2014), the standard approach to representing which elements of $\mathbf{x}_n$ are observed and which are missing is to introduce an auxiliary response indicator series $\mathbf{r}_n = [\mathbf{r}_{1n}, ..., \mathbf{r}_{L_n n}]$ where $r_{idn} = 1$ if the value of x_{idn} is observed and $r_{idn} = 0$ otherwise. The full vector-based representation of an irregularly sampled time series with incomplete observations is thus $\mathbf{s}_n = (\mathbf{t}_n, \mathbf{x}_n, \mathbf{r}_n)$. Compared to the series-based representation, the vector-based representation can thus be substantially more space efficient, but only if all vectors are completely observed.

2.1.3 Set-Based Representation

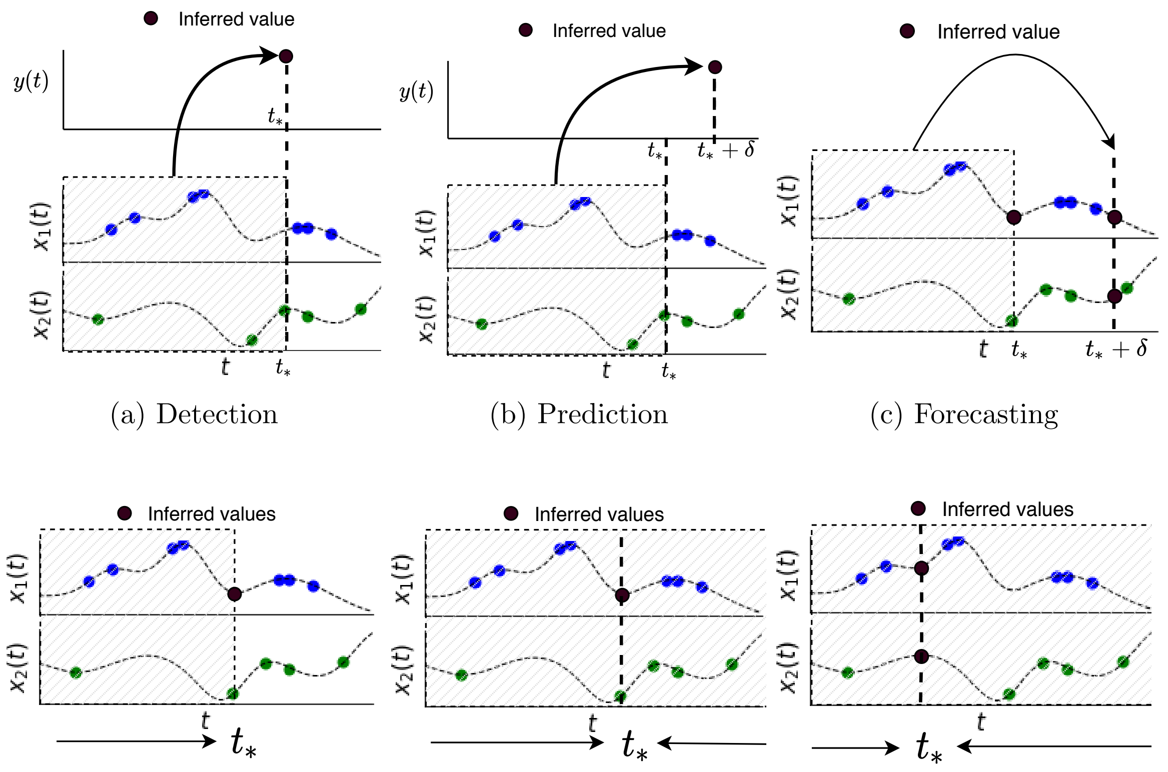
As shown in Figure 2.4, the set-based representation of a multivariate irregularly sampled time series data set views the data for a single data case n as set of time-dimension-value tuples of the form (t_{in}, d_{in}, x_{in}). Here we let L_n represent the total number of observations across all dimensions. The irregularly sampled time series for data case n is thus represented as $\mathbf{s}_n = \{(t_{in}, d_{in}, x_{in}) | 1 \leq i \leq L_n\}$. Like the series-based representation, the set-based representation does not require explicit representation of missing data. Unlike both the the series and vector-based representation, the time ordering of the data is not explicitly reflected in the structure of the set-based representation.

2.2 Inference Tasks

In this section, we define time series inference tasks. While these tasks are not specific to the case of irregularly sampled time series, carefully specifying inference tasks is necessary to properly categorize models for learning from irregularly sampled time series in terms of the tasks that they can perform. In the definition of these tasks, it will be important to specify the time ranges that are conditioned on for a given input irregularly sampled time series. We will use the notation $\mathbf{s}[:t]$ to denote all of the data contained in time series $\mathbf{s}$ that are observed up to and including time t. We will use $\mathbf{x}[t]$ to refer to the vector of observations available at time t, some (or all) of which may be missing. For brevity, we will use $\mathbf{x}^m[t]$ to indicate the sub-set of dimensions with values that are missing at time t. Some of the tasks we consider are supervised in the sense that they involve learning to infer values that are distinct from the input irregularly sampled time series. We denote these prediction targets at time t by $\mathbf{y}[t]$. Below we define and discuss six inference tasks shown in Figure 2.5.

Definition 1 Detection: *Inferring prediction target values* $\mathbf{y}[t_*]$ *at time* t_* *conditioning on the observations* $\mathbf{s}[:t_*]$ *available up to and including time* t_*.

The detection task is to infer the value of the prediction target variable at time t_*. All time series data observed up to and including time t_* can be conditioned on when making this inference. In machine learning, the inference for any quantity that is not known is often referred to as a "prediction," but in this context we reserve the term "prediction" to refer to a task where the inference is for the value of a variable at a time that is in the relative future of the time point t_* at which the inference is made, as we describe next.



(a) Detection (b) Prediction (c) Forecasting

(d) Filtering (e) Smoothing (f) Interpolation

Figure 2.5: This figure illustrates time series inference tasks. (a) The detection task involves predicting the target values $y[t_*]$ at time t_* conditioning on the observations available up to and including time t_*. (b) The prediction task requires inferring the prediction target value $y[t_* + \delta]$ at time $t_* + \delta$ for $\delta > 0$ by conditioning on the observations available up to and including time t_*. (c) The forecasting task requires inferring $\mathbf{x}[t_* + \delta]$ by conditioning on the observations up to an including time t_*. (d) The filtering task requires inferring missing variables at time t_* by conditioning on the observations up to an including time t_*. (e) The smoothing task requires inferring missing variables at time t_* using all observed data. (f) The interpolation task requires inferring the values of $x[t_*]$ at time t_* using all observed data.

Definition 2 Prediction*: Inferring prediction target values* $\mathbf{y}[t_* + \delta]$ *at time* $t_* + \delta$ *(for* $\delta > 0$*) conditioning on the observations* $\mathbf{s}[: t_*]$ *available up to and including time* t_**.*

In the prediction problem, t_* refers to the time point at which the prediction is made and $t_* + \delta$ refers to the time point at which the value of the target variable is to be predicted. We assume $\delta > 0$. The case $\delta = 0$ recovers the detection problem.

We note that in some supervised problems, the prediction target variables may not have time stamps explicitly associated with them. Such problems include whole time series regression and classification where an input time series $\mathbf{s}$ is associated with a single scalar output y. These problems are equivalent in structure to the prediction problem defined above where all available data are allowed to be conditioned on when inferring the target value.

Definition 3 Filtering: *Inferring missing variables* $\mathbf{x}^m[t_*]$ *at time* t_* *by conditioning on the observations* $\mathbf{s}[: t_*]$ *up to an including time* t_*.

The filtering task is the analog of the detection task but where the inference of interest is about the values of the multivariate time series itself. In the irregularly sampled setting, it may be that some dimensions of $\mathbf{x}[t_*]$ are observed at time t_*. In this case, these values and any values observed before this time point can be used to infer the unobserved values $\mathbf{x}^m[t_*]$.

Definition 4 Smoothing: *Inferring the values of* $\mathbf{x}^m[t_*]$ *at time* t_* *using the observed data in* $\mathbf{s}$.

The smoothing task is similar to the filtering task except that all observed data can be used to infer the unobserved values $\mathbf{x}^m[t_*]$. This includes observations in both the relative past and future of $\mathbf{x}^m[t_*]$.

Definition 5 Interpolation: *Inferring the values of* $\mathbf{x}[t_*]$ *at time* t_* *using the observed data in* $\mathbf{s}$.

The interpolation task is similar to the filtering task except that in the interpolation task, we infer the values of all variables at time t_*, not just those that are unobserved. This distinction is only relevant in scenarios when the time point t_* coincides exactly with an observation time point contained in the time series $\mathbf{s}$.

Definition 6 Forecasting: *Inferring* $\mathbf{x}[t_* + \delta]$ *(for* $\delta > 0$*) by conditioning on the observations* $\mathbf{s}[: t_*]$ *up to an including time* t_**.*

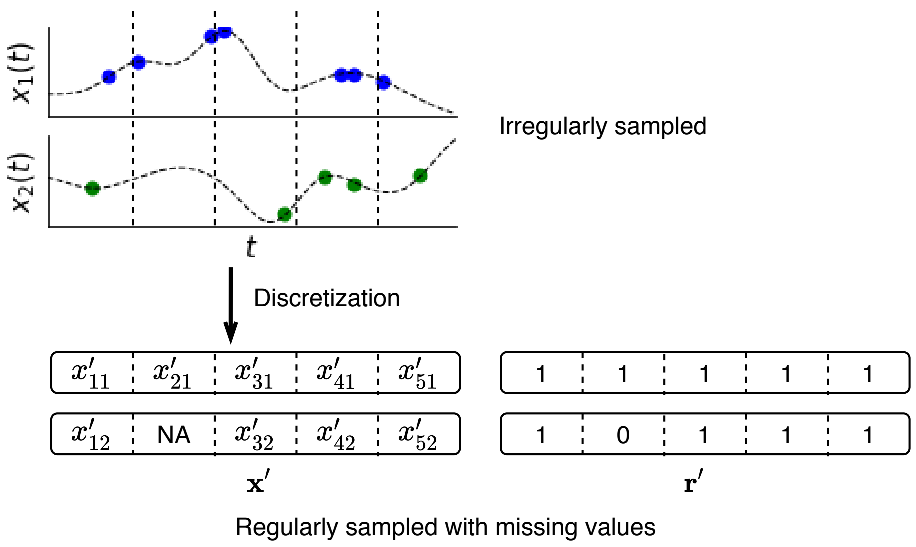
The forecasting task is the analog of the prediction task, but where we seek to predict the value of the time series itself at a future time point. In this problem, t_* again refers to the time point at which the forecast is made and $t_* + \delta$ for $\delta > 0$ refers to the time point about which the forecast is made. All observations up to and including time t_* can be used to compute a forecast at time t_*. δ is often referred to as the forecast horizon, the amount of time into the future we are forecasting the value of the time series.

2.3 Modeling Primitives for Irregularly Sampled Time Series

While there has been a significant volume of work in recent years on modeling sparse and irregularly sampled time series data in the context of different tasks and applications, the number of fundamental approaches for accommodating irregular sampling itself is much more limited. In this section, we describe several "modeling primitives" for accommodating irregular sampling. These modeling primitives are the basic building blocks or modules for dealing with irregular sampling that are leveraged in larger and more complex models and methods for solving specific tasks using irregularly sampled time series. We discuss a number of modeling primitives including discretization, interpolation, similarity, recurrence, attention and structural invariance. Figure 2.1 provides a taxonomy of modeling primitives and associated methods.

2.3.1 Discretization

Temporal discretization is a basic modeling primitive used to convert irregularly sampled time series data into a regularly sampled time series, as shown in Figure 2.6. To apply this primitive, one first defines a sequence of $K+1$ regularly spaced reference



$x_1(t)$
$x_2(t)$
t
Irregularly sampled
Discretization
x'_{11} x'_{21} x'_{31} x'_{41} x'_{51}
x'_{12} NA x'_{32} x'_{42} x'_{52}
1 1 1 1 1
1 0 1 1 1
$\mathbf{x}'$
$\mathbf{r}'$
Regularly sampled with missing values

Figure 2.6: This figure illustrate the discretization of irregularly sampled time series into regularly spaced time series with missing values. The discretization process requires dividing the time axis into equal sized non-overlapping intervals and defining a value within each time interval based on the observed values falling within that interval. In cases where a given interval contains no observations on a particular dimension, the result is missing data. We can again represent these missing values via auxiliary response indicator vectors.

time points $\tau_0, ..., \tau_K$. Given an irregularly sampled time series $\mathbf{s}_n$, we define a new regularly sampled time series in the vector-based representation with missing data indicators $\mathbf{s}'_n = (\mathbf{t}'_n, \mathbf{x}'_n, \mathbf{r}'_n)$. We let $\mathbf{t}'_n = [t'_1, ..., t'_K]$ for all n where $t'_i = (\tau_{i-1} + \tau_i)/2$.[1]

Next, we define a function that maps the observed values of $\mathbf{s}_n$ falling within the discretization window $[\tau_{i-1}, \tau_i)$ to the vectors $\mathbf{x}'_{in}$ and $\mathbf{r}'_{in}$. Assume there are $j > 0$ observations for dimension d in $\mathbf{s}_n$ within the interval $[\tau_{i-1}, \tau_i)$ and let their indices be $\mathbf{o} = [o_1, ..., o_j]$. We require a function of the form $\mathbf{x}'_{idn} = f([\mathbf{t}_{o_1n}, ..., \mathbf{t}_{o_jn}], [\mathbf{x}_{o_1dn}, ..., \mathbf{x}_{o_jdn}])$ that maps the set of observations within the discretization window to a single representative value. A common approach is to set this function to be the average of the

[1]We choose the midpoint of the interval $[\tau_{i-1}, \tau_i)$ to represent the interval, but the starting point or any other consistent choice could also be made.

j values. To complete the representation for this interval, we set $\mathbf{r}'_{idn} = 1$ to indicate that the interval has an observed value.

In the case where $j = 0$, the original time series has no observations of dimension d within the interval $[\tau_{i-1}, \tau_i)$. In this case we set $\mathbf{r}'_{idn} = 0$ to indicate no observations are available. The value set in $\mathbf{x}'_{idn}$ will typically be 0 or nan, but is not consequential as any correct algorithm for this representation will not use values in $\mathbf{x}'_{idn}$ when $\mathbf{r}'_{idn} = 0$ (Little and Rubin, 2014).

As we can see, the discretization primitive provides a reduction from the problem of modeling irregularly sampled time series to the problem of modeling regularly sampled time series that may contain missing data. As discretization windows become larger, the volume of missing data will generally decrease. However, more values will potentially also fall within the same discretization window leading to more aggregation. This may remove information needed to perform some tasks. As discretization windows become shorter, aggregation effects are reduced, but the length of the discretized time series increase as does the volume of missing data. Thus, the window size becomes an important hyperparameter of methods based on this approach.

We note that when discretization does result in significant volumes of missing data, the problem of dealing with the resulting missing data can itself be highly non-trivial as the presence of incomplete data also violates the assumptions of most standard discriminative machine learning methods. Imputation methods can be used as a final stage of pre-processing to address the problem of missing data, but care is needed when reflecting input uncertainty is important. A number of recent deep learning approaches have addressed the problem of providing flexible single and multiple imputation methods (Yoon et al., 2018a; Li et al., 2019; Mattei and Frellsen, 2019; Śmieja et al., 2018). Many simpler methods are also commonly used for time series including mean imputation, zero imputation and forward-filling (Lipton et al., 2016; Harutyunyan et al., 2019).

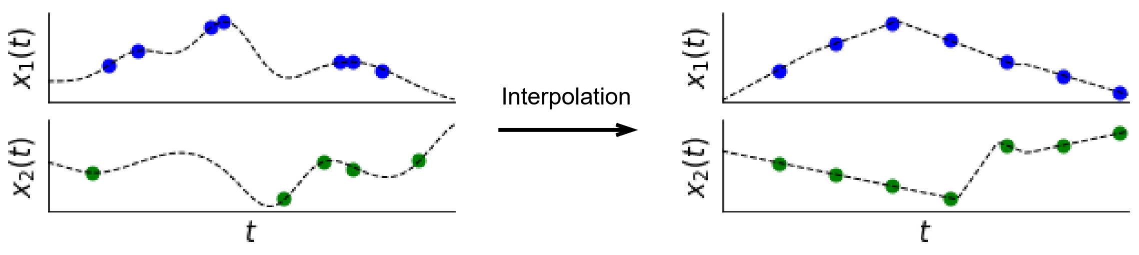


Figure 2.7: Illustration of interpolation in the case of a two-dimensional irregularly sampled time series. In this example, we illustrate the basic use of linear interpolation against a fixed set of reference time points.

2.3.2 Interpolation

Interpolation primitives provide an approach to accommodating irregular sampling that is closely related to discretization but provides improved flexibility with potentially fewer ad-hoc assumptions. Interpolation methods are often used to provide an interface between irregularly sampled time series data and models for either fixed-dimensional feature spaces or for variable length sequences.

Similar to discretization, we begin by defining a set of K reference time points $\boldsymbol{\tau} = [\tau_1, ..., \tau_K]$. In the case of a length L_n univariate irregularly sampled time series $\mathbf{s}_n = (\mathbf{t}_n, \mathbf{x}_n)$, the interpolated output $\mathbf{x}'_{in}$ at reference time point τ_i can be computed as shown in Equation 2.1. The interpolated time series is then given by $\mathbf{s}'_n = (\boldsymbol{\tau}, \mathbf{x}'_n)$. The function $\kappa_\theta()$ is a similarity kernel that puts higher weight on pairs of time points $(\tau_i, \mathbf{t}_{jn})$ that are closer together in time. This function can depend on a set of learnable parameters θ. The squared exponential kernel function $\kappa_\theta(t, t') = \alpha \exp(-\beta(t-t')^2)$ is a common choice in the literature for non-linear interpolation (with $\theta = [\alpha, \beta]$).

$$\mathbf{x}'_{in} = \frac{\sum_j \kappa_\theta(\tau_i, \mathbf{t}_{jn})\, \mathbf{x}_{jn}}{\sum_j \kappa_\theta(\tau_i, \mathbf{t}_{jn})} \tag{2.1}$$

For multivariate irregularly sampled time series, there are more possibilities in terms of the construction of interpolation methods as these methods can account for both correlation in time and correlation across different dimensions. One basic approach motivated by the series-based view of a multivariate time series is to separately interpolate each dimension of the time series using univariate interpolation as shown in Equation 2.1 and in Figure 2.7, but with the same set of reference time points used across all dimensions. This ignores cross-dimension correlations, but provides regularly spaced output to which further modeling can be applied to account for cross-dimensional correlations.

One potential issue when inter-observation times exhibit significant variability is variable uncertainty in interpolated values. The use of Gaussian process regression (GPR) models can provide a better alternative to deterministic interpolation methods in such cases (Rasmussen and Williams, 2006). The primary strength of GPR models is that they provide a joint posterior distribution over an arbitrary collection of reference time points $\boldsymbol{\tau} = [\tau_1, ..., \tau_K]$ conditioned on an input irregularly sampled time series $\mathbf{s}_n$. This posterior distribution is a joint Gaussian defined in terms of a mean $\mathbf{m}_n$ over $\boldsymbol{\tau}$ and a corresponding covariance matrix $\boldsymbol{S}_n$ (Rasmussen and Williams, 2006).

This approach is most straightforward to apply in the univariate case where $\mathbf{s}_n = (\mathbf{t}_n, \mathbf{x}_n)$. The model is defined in terms of a covariance function $\kappa_\theta(\cdot, \cdot)$ with parameters θ. Unlike the case of basic deterministic interpolation methods mentioned previously, GPR models require the covariance function to be a valid Mercer kernel (Rasmussen and Williams, 2006). The GPR model also includes a noise variance term σ^2 and a mean function $\mu_\phi()$. Under this model, the posterior probability density over the values $\mathbf{x}'_n = [x'_{1n}, ..., x'_{Kn}]$ is given by $p(\mathbf{x}'_n|\tau, \mathbf{s}_n, \theta, \phi, \sigma) = \mathcal{N}(\mathbf{x}'_n; \mathbf{m}_n, \mathbf{S}_n)$ where the conditional mean and covariance matrices $\mathbf{m}_n$ and $\mathbf{S}_n$ are shown below. $\mathbf{K}_{\mathbf{t},\mathbf{t}'}$ denotes the covariance matrix defined by $[\mathbf{K}_{\mathbf{t},\mathbf{t}'}]_{ij} = \kappa_\theta(\mathbf{t}_i, \mathbf{t}'_j)$.

$$\boldsymbol{m}_n = \mu_\phi(\boldsymbol{\tau}) + \mathbf{K}_{\boldsymbol{\tau},\mathbf{t}_n}(\mathbf{K}_{\mathbf{t}_n,\mathbf{t}_n} + \sigma^2\mathbf{I})^{-1}(\mathbf{x}_n - \mu_\phi(\mathbf{t}_n)) \tag{2.2}$$

$$\boldsymbol{S}_n = \mathbf{K}_{\boldsymbol{\tau},\boldsymbol{\tau}} - \mathbf{K}_{\boldsymbol{\tau},\mathbf{t}_n}(\mathbf{K}_{\mathbf{t}_n,\mathbf{t}_n} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{\mathbf{t}_n,\boldsymbol{\tau}} \tag{2.3}$$

One possible application of GPR to the interpolation problem is to define $\mathbf{s}'_n = (\boldsymbol{\tau}, \boldsymbol{m}_n)$. This approach simply uses the posterior mean on τ as the interpolated values. This approach has no particular advantage over deterministic interpolation methods using direct smoothing kernels introduced above as it discards all posterior uncertainty. A potentially better approach is to draw J samples of the form $\mathbf{x}'_{nj} \sim \mathcal{N}(\mathbf{x}'_n; \mathbf{m}_n, \mathbf{S}_n)$ for $1 \leq j \leq J$ and to construct J interpolated time series using $\mathbf{s}'_{nj} = (\boldsymbol{\tau}, \mathbf{x}'_{nj})$. The ensemble of interpolants $\{\mathbf{s}'_{nj}\}_{j=1:J}$ will exhibit greater cross-sample variation in regions where there are few observations in $\mathbf{s}_n$, reflecting natural uncertainty due to observation sparsity. Such an ensemble can be used in down-stream analyses in a way that is analogous to multiple imputations in a missing data problem (Little and Rubin, 2014).

However, as we can see, applying GPR in this way can be quite expensive as the naive computation of the matrix inverse in the posterior mean $\mathbf{m}_n$ and covariance matrix $\mathbf{S}_n$ computations requires time cubic in L_n. While faster approximations are available for these computations as well as for drawing samples from these posteriors, GPR-based approaches are typically still much slower than deterministic interpolation methods. We also note that as with deterministic interpolation methods, GPR has several parameters to estimate. These can be fit in a number of ways including the use of marginal likelihood maximization when given a data set consisting of irregularly sampled time series (Rasmussen and Williams, 2006).

The discussion of GPR thus far has focused only on the case of univariate irregularly sampled time series. Like with deterministic interpolation methods, GPR-based methods can also be applied to the case of multivariate irregularly sampled time series. The two basic approaches are again to leverage the series-based view and produce a posterior distribution or set of samples over each dimension separately, or to define a joint covariance function that applies both over time and across dimensions. A full GPR model over all dimensions (using a multi-task or multi-output Gaussian process) has the advantage of being able to leverage covariance structure across all dimensions while producing a posterior distribution for any individual dimension at any time point (Bonilla et al., 2008). However, defining flexible covariance functions for multivariate problems can be quite challenging due to the requirement of positive definiteness.

2.3.2.1 Similarity

The previous section discusses the use of similarity functions and kernels within individual irregularly sampled time series. These approaches essentially aim to interpolate a single time series by considering the local similarity of time points and/or dimensions. However, we can also consider applying similarity to pairs of multivariate irregularly sampled time series $\mathbf{s}$ and $\mathbf{s}'$, providing a similarity-based modeling primitive.

At a minimum, a similarity kernel between irregularly sampled time series $\mathcal{K}_\theta(\mathbf{s}, \mathbf{s}')$ needs to provide non-negative values and needs to encode the notion that when $\mathbf{s}$ and $\mathbf{s}'$ are more "similar" in some sense, $\mathcal{K}_\theta(\mathbf{s}, \mathbf{s}')$ takes larger values. Any such kernel function $\mathcal{K}_\theta(\mathbf{s}, \mathbf{s}')$ can then be used to solve machine learning tasks that can be formulated in terms of similarity between data points. Basic examples include K-nearest neighbor classification and regression methods (Altman, 1992). If the function $\mathcal{K}_\theta(\mathbf{s}, \mathbf{s}')$ also satisfies the requirements of a Mercer kernel function, a wider set of classical kernel methods can be applied including support vector classification (Cortes and Vapnik, 1995), support vector regression (Smola and Schölkopf, 2004), kernel ridge regression (Hastie et al., 2001), kernel logistic regression (Wahba, 1999), kernel principle components analysis (Schölkopf et al., 1998), etc.

A number of kernels have been proposed for irregularly sampled time series, some of which are valid Mercer kernels. For example, Lu et al. (2008) proposed an approach for the univariate case that can be thought of as constructing a latent function condi-

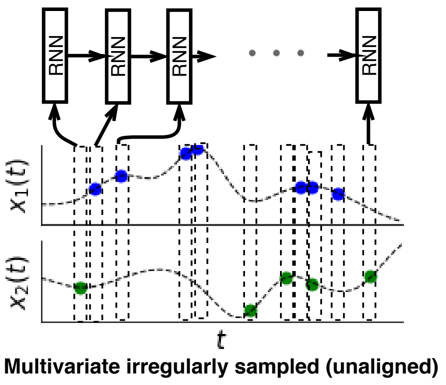


Figure 2.8: This figure illustrates the recurrence based modeling primitive for a two-dimensional irregularly sampled time series with unaligned observations. In this case, we show the use of recurrent neural network (RNN) cell that integrates the input at each time point with the latent state from the previous time point. Basic RNN models view time series as general sequences and require completely observed input vectors. However, many RNN-based models have been developed that explicitly represent time to deal with irregular sampling and integrate methods for dealing with missing data.

tioned on an observed time series $\mathbf{s}$ using a temporal similarity kernel. The construction of this latent function is identical to the construction of the Gaussian process posterior mean function described in the previous section. Lu et al. (2008) then show that a kernel between two such latent functions induced by two different irregularly sampled time series $\mathbf{s}$ and $\mathbf{s}'$ can be computed only in terms of the observed values and time points that define the two time series. This construction is quite elegant in that it avoids the need to materialize the latent functions at reference time points. However, since it measures the similarity between deterministic latent functions, it discards all uncertainty due to sparse sampling. Other approaches include kernels that consider alignment and warping when assessing similarity (Shimodaira et al., 2001; Cuturi, 2011).

2.3.3 Recurrence

As noted in the previous sections, one of the issues when dealing with time series in general is the possibility of different data cases containing different numbers of observations. This problem can be dealt with by defining models that appeal to primitives based on recurrence. The key property of such primitives is that they use a fixed, finite set of parameters to model sequences of arbitrary length. Examples include classical autoregressive structures used in Markov models, hidden Markov models (Rabiner, 1990), conditional random fields (Lafferty et al., 2001), and recurrent neural networks (Rumelhart et al., 1986). In this section, we will focus on recurrent neural network (RNN) models as an example recurrent primitive as shown in Figure 2.8.

An RNN provides a basic building block for modeling fully observed multivariate sequences $\mathbf{z}_n = [\mathbf{z}_{1n}, ..., \mathbf{z}_{L_n n}]$. The key is to define a cell whose hidden state $\mathbf{h}_i$ is updated based on the previous state $\mathbf{h}_{i-1}$ and the current input $\mathbf{z}_{in}$ through a non-linear function $f_\theta()$ with parameters θ. The model can also optionally output values $\hat{\mathbf{y}}_{in}$ via a second function $o_\phi()$ applied to the hidden state. The model is thus able to process sequences of arbitrary length due to the fact that it processes elements of the sequence one at a time and the parameters θ and ϕ are position-independent.

$$\mathbf{h}_i = f_\theta(\mathbf{h}_{i-1}, \mathbf{z}_{in}) \tag{2.4}$$

$$\hat{\mathbf{y}}_{in} = o_\phi(\mathbf{h}_i) \tag{2.5}$$

Due to the ability to process sequences of arbitrary length, RNNs are a useful building block for modeling time series in general. In the case of univariate irregularly sampled time series or multi-variate series with completely observed vector-valued observations, an RNN can be directly applied simply by processing the data in time order and discarding the values of the time points. Such an approach could be suitable for problems where the variation in inter-observation intervals is relatively small. When that is not the case, discarding the time values makes the model invariant to time gaps in a way that may be harmful to performance on tasks of interest.

A number of simple approaches can be taken to solve this problem. One approach is to append the time points or inter-observation intervals to the vector-valued observations yielding the sequence of values $\mathbf{x}'_{in} = [\mathbf{x}_{in}, t_{in}]$ or $\mathbf{x}'_{in} = [\mathbf{x}_{in}, t_{in} - t_{i-1n}]$. An RNN model can then be applied to this modified sequence. In theory, a sufficiently powerful non-linear mapping $f_\theta()$ should be able to account for irregular sampling based on this representation. In practice, this approach can under-perform other approaches to dealing with time (Che et al., 2018a).

Recent work on ordinary differential equation (ODE) models (Chen et al., 2018) in machine learning provides an alternative recurrence-based solution with better properties than traditional RNNs in terms of their ability to accomodate irregularly sampled data. In these models, ODEs are used to evolve the hidden state between continuous time observations. At each observation time-point, the hidden state is updated using a standard RNN update. These models are often referred to as ODE-RNNs. The hidden state update equations are defined below:

$$\mathbf{h}'_i = \text{ODESolve}(g_\gamma, \mathbf{h}_{i-1}, (t_{i-1n}, t_{in})) \tag{2.6}$$

$$\mathbf{h}_i = f_\theta(\mathbf{h}'_i, \mathbf{x}_{in}) \tag{2.7}$$

The function g_γ is a time-invariant function that takes the value at the current time step and outputs a gradient: $\frac{\partial \mathbf{h}(t)}{\partial t} = g_\gamma(\mathbf{h}(t))$. This function is parameterized using a neural network. The ODE defined by $g_\gamma(\mathbf{h}(t))$ is then solved at the given time point t_{in} to produce a new hidden state $\mathbf{h'}_i$. This hidden state is further updated to fold in the input value at time t_{in}, yielding the final hidden state value $\mathbf{h}_i$ at time t_{in}. This model does not explicitly depend on t_{in} or $t_{in} - t_{i-1n}$ when updating the hidden state, but does depend on time implicitly through the resulting latent state of the dynamical system.

The ODE-RNN model primitive is a substantially more elegant approach to dealing with irregular sampling than adaptations of discrete time RNNs, but it can also be

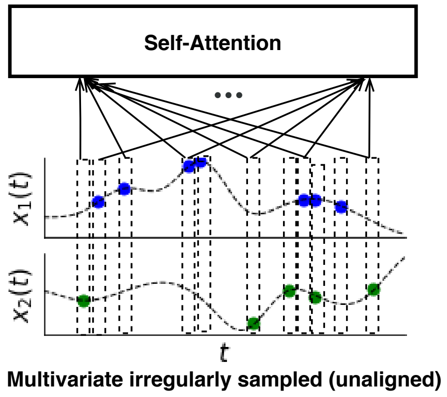


Figure 2.9: This figure illustrates the attention modeling primitive for two dimensional irregularly sampled time series. The attention-based primitive processes time series in parallel instead of sequentially as for an RNN. Attention models learn which regions of an input time series to attend to when computing outputs at different points in time by leveraging positional or time encodings.

substantially slower due to the need to repeatedly apply an ODE solver. Further, we note that when irregularly sampled time series are multivariate, but the vector-valued observations are not fully observed, the application of both standard RNN models and ODE-RNNs becomes significantly more challenging as both models require a fully observed $\mathbf{x}_{in}$ vector in order to update the hidden state.

A common baseline approach to missingness is to perform some form of imputation by defining a new sequence of values $\mathbf{x}'_{in} = \mathbf{r}_{in}\ \mathbf{x}_{in} + (1 - \mathbf{r}_{in})\ \tilde{\mathbf{x}}_{in}$ where $\tilde{\mathbf{x}}_{in}$ is the value to impute when the response indicator $\mathbf{r}_{in} = 0$. Forward filling, zero imputation, and mean imputation can sometimes be reasonable options when the volume of missing data is low.

2.3.4 Attention

Attention has become a key modeling component for many machine learning tasks including image caption generation, speech recognition and neural translation (Vaswani et al., 2017). In self-attention models, each element of a sequence learns which other elements of a sequence to attend to without relying on recurrent network structures. Self-attention models offer computational advantages over RNNs since sequence processing can be fully parallelized. Self-attention is a particular instance of scaled dot-product attention, which is defined as:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\,\mathbf{K}^T}{\sqrt{C}}\right)\mathbf{V} \tag{2.8}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ denote the query, key, value representation respectively, and C is the dimension of the key or query representation. In self-attention, all of the queries, keys and values come from the input sequence. Self-attention relies on positional encoding, a vector representation for each position in the sequence to recognize and capture the sequential ordering information. The above defined $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ matrices are typically linear projections of position-value joint representations.

To provide a primitive for irregular sampling (as shown in Figure 2.9), time values can be converted into a vector representation similar to positional encoding and concatenated with the observation value as described for RNNs. Missing values in vector-valued observations are also problematic for attention-based modules, which (like standard RNNs) expect fully observed vectors as input. Approaches based on attention thus also need to solve the problem of missing dimensions, and a variety of imputation solutions can be used as described previously.

2.3.5 Structural Invariance

While the modeling primitives presented to this point all attempt to represent the time ordering of observations in the structure of the model this is not strictly required. Leveraging such structural invariance has the potential to simplify learning problems.

Recently, set-based neural network approaches (as shown in Figure 2.10) have been employed to deal with irregularly sampled data and they exactly express this form

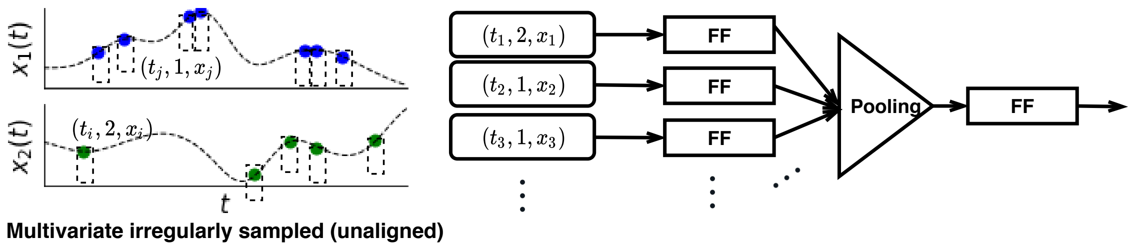


Figure 2.10: An illustration of the structural invariance-based modeling primitive. Inspired by the set-based view of a multivariate irregularly sampled time series, this approach processes individual (time, value, dimension) tuples via an encoding function and then pools over the output of all such tuples. The model structure thus does not reflect the time ordering of the data in any way.

of structural invariance. Set-based model architectures (Zaheer et al., 2017) support variable length sequences, partially observed vectors, sparse observations and irregular intervals between observation times while avoiding the need for temporal discretization, imputation and interpolation. These approaches leverage the set-based representation of an irregularly sampled time series data set $\mathbf{s}_n = \{(t_{in}, d_{in}, x_{in}) | 1 \leq i \leq L_n\}$. As shown below, such approaches produce an encoding of an input irregularly sampled time series by applying an initial encoder f_θ to individual time-dimension-value tuples.

$$\mathbf{h} = g_\phi(\text{pool}(\{f_\theta(t_{in}, d_{in}, x_{in}) | 1 \leq i \leq L_n\}) \tag{2.9}$$

The approach then performs a pooling operation over all initial encodings in a way that is completely invariant to the temporal structure of the data. The output of pooling is mapped through one additional set of encoding layers g_ϕ to produce the final representation. Commonly used pooling functions are max, mean and sum.

2.3.6 Summary

The discretization, interpolation, similarity, recurrence, attention and structural invariance modeling primitives that we have just described underlie essentially all of the recent work on models for irregularly sampled time series across all of the tasks described in Section 2.2. In the following sections, we survey over 40 papers categorized by the fundamental modeling primitive used and discuss the tasks to which each method can be applied.

2.4 Discretization-Based Approaches

In this section, we discuss approaches to modeling irregularly sampled time series that leverage discretization as their primary modeling primitive for accommodating irregular sampling.

Marlin et al. (2012), Lipton et al. (2016), Bahadori and Lipton (2019) and Harutyunyan et al. (2019) all apply discretization as a modeling primitive for dealing with irregular sampling. These papers all discretize time into consecutive, hour-long, non-overlapping intervals within a fixed time interval $[0, T]$ common to all data cases. The application of discretization under these conditions enables the use of models that operate on fixed-dimensional vectors. However, in all of the tasks considered in these papers, the application of discretization results in missing data that must be dealt with, as well as instances where aggregation is required within time intervals.

Marlin et al. (2012) make the assumption that the missing data is missing at random and apply probabilistic mixture models that can efficiently deal with missing data under this assumption. This allows models to be learned without requiring explicit imputation. Marlin et al. (2012) leverage this capability of the probabilistic mixture model framework to define a generative mixture of experts classifier for whole time series classification. The case of multiple values for a given dimension being defined within the same time interval is dealt with via averaging.

Lipton et al. (2016) apply discretization followed by an RNN to build a whole time series classification model. This approach requires explicit imputation of missing data created during the discretization step. They consider two basic approaches:

forward-filling and zero imputation. Missing values are replaced with zeros in the zero-imputation strategy. In the forward-filling approach, a missing value $\mathbf{x}_{idn}$ at time t_{in} on dimension d is set to the last observed value on dimension d (e.g., the value of $\mathbf{x}_{jdn}$ where j is such that t_{jn} is the largest time value satisfying $t_{jn} < t_{in}$ and $\mathbf{r}_{jdn} = 1$). In the absence of previous measurements (or if the variable is missing entirely), it is replaced with the median estimated over all measurements in the training data.

One important feature of missing data problems is the potential for the sequence of observation times to itself be informative (Little and Rubin, 2014). Since the set of response indicators $\mathbf{r}_{idn}$ is always observed, this information is easy to condition on as well. This approach was also used by Lipton et al. (2016) where they include as input to the RNN both the observed and imputed values of the multivariate time series and the values of the response indicator vector. In addition to the binary indicator variable, Lipton et al. (2016) add hand-engineered features derived from the response indicator time series such as mean and standard deviation of indicator variables for each time series.

Harutyunyan et al. (2019) also consider the application of discretization followed by RNNs. They apply averaging or selection of the last time point in an interval to deal with multiple observations in the discretization window, and use mean imputation or forward filling to deal with missing values. Harutyunyan et al. (2019) augment standard RNN models by predicting outputs for multiple tasks jointly. Unlike in standard RNNs where the hidden state from the last position in the sequence is decoded to predict supervised outputs, Harutyunyan et al. (2019) apply supervision at each time step. This framework has been applied to whole time series classification tasks, detection tasks and prediction tasks.

Song et al. (2018) follow Harutyunyan et al. (2019)[2] and also discretize the sparse and irregularly sampled time-series data into hour long intervals, and use similar imputations for dealing with missing data. This model has been applied to the tasks of whole time series classification, detection and prediction. However, Song et al. (2018) adopt the multi-head attention mechanism similar to Vaswani et al. (2017) instead of an RNN as the primary model structure. They use positional encoding to incorporate the temporal order into the representation learning. They use the dense interpolation technique (Trask et al., 2015) to obtain a unified representation for a sequence.

Several approaches have also employed an encoder-decoder framework for learning with missing data in time series, which can also be applied to irregularly sampled time series after discretization. Bianchi et al. (2019) proposed an autoencoder-based approach to learn representations of multivariate time series with missing data for the whole time series classification and smoothing tasks. They use a standard RNN as the decoder. For the encoder, they use a bi-directional RNN where they combine the hidden layer output of forward and backward RNNs using a fully-connected layer. In the presence of missing data, they replace the missing values with zero or the mean while encoding, and use the decoder as a final imputer. Bianchi et al. (2019) also introduce a kernel alignment procedure to preserve the pairwise similarities of the inputs in the learned representations. These pairwise similarities are encoded in a positive semi-definite matrix that is also passed as input to the model. Fortuin et al. (2020) proposed a variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) approach for the task of smoothing in multivariate time series with a Gaussian process prior in the latent space to capture temporal dynamics.

[2]Harutyunyan et al. (2019) had an arxiv version available since 2017.

In summary, temporal discretization provides a solution for learning from irregularly sampled time series that replaces the problem of irregular observation intervals with the problem of missing data. While convenient in some respects, this approach requires several ad-hoc choices including the width of the discretization windows and the aggregation function used within windows. In the next section, we turn to methods based on the use of interpolation as a modeling primitive. These approaches still require specifying a discrete set of interpolation points, but use more sophisticated global methods for defining values at the interpolation points.

2.5 Interpolation-Based Approaches

In this section we discuss methods for learning from irregularly sampled time series that are based on interpolation as a modeling primitive. Similar to discretization methods, interpolation methods require specifying a discrete set of reference time points. However, they materialize interpolants at these time points in a way that can leverage all available observations in the input if desired. This can make them much more powerful than discretization methods with ad-hoc local aggregation functions.

Li and Marlin (2020) proposed an encoder-decoder architecture for modeling irregularly sampled time series data. Encoder in this framework is based on a piecewise linear function, which learns an interpolation on a fixed set of reference time points. This framework uses a kernel smoother as a decoder to interpolate at arbitrary times.

One of the potential disadvantages of using deterministic kernel smoothing-based interpolation approaches is that they do not reflect uncertainty due to long intervals with no observations. An alternative line of work on interpolation-based approaches instead leverages Gaussian processes as a building block, enabling uncertainty propagation as described previously (Rasmussen and Williams, 2006). In response to the shortcomings of the kernel-based approach of Lu et al. (2008), Li and Marlin (2015) developed a two-step interpolation-based approach to modeling the similarity between sparse and irregularly sampled time series in a way that is sensitive to uncertainty. In the first step of this approach, they use marginal likelihood maximization to fit a Gaussian process regression model to a data set of irregularly sampled time series. Once the model is trained, they define a set of evenly spaced interpolation points and materialize the Gaussian process posterior for each time series at the same set of reference time points. In the second phase of the method, they define Mercer kernels between the Gaussian process posteriors at the reference time points. This approach allows for the use of irregularly sampled time series with classical supervised and unsupervised kernel methods such as support vector machines and kernel PCA.

In subsequent work, Li and Marlin (2016) showed how the materialized Gaussian process regression posterior representation could be extended for use as an uncertainty preserving interface between irregularly sampled time series and arbitrary deep learning model components that expect regularly spaced or fixed dimensional inputs. Further, this work showed how the Gaussian process regression model parameters and the deep learning model parameters can be learned jointly and end-to-end using the re-parameterization trick (Kingma et al., 2015). This model structure allows for solving a variety of tasks, but was primarily evaluated in the context of whole time series classification.

While the model of Li and Marlin (2016) was only applied to univariate irregularly sampled time series in the original work, in follow-up work the model was extended to multivariate time series using a multi-output Gaussian process regression model (Futoma et al., 2017). However, modeling multivariate time series within this framework is quite challenging due to the constraints on the covariance function used in the GP regression layer. Futoma et al. (2017) deal with this problem using a sum of separable kernel functions (Bonilla et al., 2008), which somewhat limits the expressiveness of the model. The work of Futoma et al. (2017) also focused on the whole time series classification as a task.

In terms of the tasks of interpolation and forecasting, Ghassemi et al. (2015) proposed a multi-task Gaussian process (MTGP) model for irregularly sampled physiological signals and clinical notes, which jointly transforms the measurements into a unified latent space. They showed that MTGP models provide better performance than single-task GP models for interpolation and forecasting. Liu and Hauskrecht (2016) combine state-space models with GPs for learning forecasting models from irregularly sampled multivariate clinical data. Soleimani et al. (2018) also employed multi-output GPs to jointly model multivariate physiological signals.

In summary, interpolation as a modeling primitive provides more sophisticated approaches than the more basic discretization-based methods. The two main families of approaches explored to date in this area are deterministic kernel smoothing methods and probabilistic Gaussian process-based methods. Both approaches naturally accommodate continuous time observations and can provide interfaces to other modeling building blocks such as kernel machines and neural networks. The main advantage of Gaussian process-based modeling primitives is the ability to represent and propagate uncertainty into downstream model components. Their main drawback is the significantly higher run times during both training and prediction, and the added complexity of needing to define valid mercer kernels for multivariate time series. Deterministic interpolation frameworks do not have the elegance of Gaussian processes in terms of representing uncertainty, but can be much more flexible in terms of how they express temporal and cross-dimensional relationships. Importantly, they are also one to two orders of magnitude more computationally efficient during training. In the next section we turn to the case of recurrence as a modeling primitive and explore methods based on recurrent neural networks.

2.6 Recurrence-Based Approaches

In this section, we discuss approaches to modeling irregularly sampled time series that leverage recurrence as their primary modeling primitive for accommodating irregular sampling. We begin with traditional discrete RNN models, and then discuss more recent work on ODE-based models. Importantly, methods in this section do not rely on discretization prior to the application of RNNs. This is the primary distinction relative to methods presented in the previous sections, some of which also leverage RNNs after discretization has been applied to accommodate irregular sampling.

2.6.1 RNN-Based Methods

Similar to Lipton et al. (2016), Che et al. (2018a) present several methods based on gated recurrent unit networks (GRUs, Chung et al. (2014)) for the whole time series classification task. Their approach is cast in the vector-based representation. Thus, even though they do not apply discretization, they still need to deal with the problem of missing data inherited from this view of the problem. In their baseline methods, Che et al. (2018a) deal with the issue of missing values in vectors observed at irregularly sampled time points using simple imputation methods including mean imputation and forward filling. They also consider augmenting inputs with binary response indicator values and per-dimension inter-observation time intervals. This last approach attempts to provide the RNN with information about how long it has been since a value on a given dimension was last observed. When applying forward filling as the imputation method, the inter-observation time interval values indicate how long it has been since a new value for the variable was last observed.

However, the primary contribution of Che et al. (2018a) is the GRU with decay (GRU-D) model that adds a temporal decay mechanism to the the input variables and the hidden states. The decay rate function is defined as:

$$\gamma(\delta) = \exp\{-\max(0, \mathbf{W}_\gamma \delta + \mathbf{b}_\gamma)\} \tag{2.10}$$

where $\mathbf{W}_\gamma$ and $\mathbf{b}_\gamma$ are learned jointly with GRU parameters at training time. δ represents the length of the inter-observation interval. They introduce input decay for missing variables to decay the previously observed value toward the empirical mean for a given dimension over time. For large inter-observation intervals, this can be a more sensible choice than continued forward filling as the previously observed value will become uncorrelated with the current value over sufficiently long intervals. This can make the overall mean for that dimension a better imputed value. We describe the GRU-D layer below:

$$\mathbf{x}'_{idn} = \mathbf{r}_{idn}\,\mathbf{x}_{idn} + (1 - \mathbf{r}_{idn})\,(\gamma(\delta)\mathbf{x}_{jdn} + (1 - \gamma(\delta))\bar{\mathbf{x}}_d) \tag{2.11}$$

where $\mathbf{x}_{jdn}$ is the last observation $(t_j < t_i)$ for dimension d, and $\bar{\mathbf{x}}_d$ is the empirical mean of the d^{th} dimension in multivariate time series. Importantly, while this approach can be thought of as a form of imputation, the parameters that control imputation are learned end-to-end with the rest of the model parameters based on a supervised objective.

As a further modification, Che et al. (2018a) also consider decaying the hidden states in the absence of observations. This has the effect of partially resetting the hidden state when there is a long interval with no observations. This is also a sensible modification as any accumulated hidden state may be irrelevant following a long time interval with no observations. Choi et al. (2016a) employed a similar method where they concatenate the time gaps between observations with the actual observations and supply it directly as input to a GRU model. Pham et al. (2017) proposed to capture time irregularity by modifying the forget gate a LSTM (Hochreiter and Schmidhuber, 1997) using a decay combined with a parametric function of the time gap between current and past measurement.

Neil et al. (2016) proposed the phased LSTM for supervised learning from irregularly sampled time series. The phased LSTM model introduced a time-dependent gating mechanism by adding a new time gate which regulates access to the hidden and cell state of the LSTM. The time-aware LSTM (T-LSTM) (Baytas et al., 2017) instead transforms time intervals into weights using a decay function and uses them to adjust the hidden state passed from the previous time step. While these approaches allow the network to handle event-based sequences with irregularly spaced observations, they do not support incomplete multivariate observations.

Li and Xu (2019) argue that the time difference δ between the current time step and last observation used in the dynamic imputation mechanism is not sufficient for the model to fully capture the overall pattern of missingness across a time series. They present VS-GRU, which also considers the observation rate of each dimension. They adopt a similar decay mechanism to that used by GRU-D. Li and Xu (2019) proposed VS-GRU-i, which adds a standard GRU layer on top of VS-GRU. They add a penalized mechanism at the input of the second GRU layer to detect if one variable is completely missing or there are only a few observed values.

Kim and Chi (2018) also propose an approach to dealing with missing data in an RNN using decay, which they refer to as a temporal belief memory. When an input is missing, the belief gate computes the belief of the last observation based on the time gap between the current time and the last observation time. If the belief is greater than a threshold, the model imputes using the last observation; otherwise, it sets the missing value to the mean value of observations for that dimension. The belief gate function $\gamma(\delta)$ for a given dimension d is defined as:

$$\gamma(\delta) = \begin{cases} 1 & \text{if } \exp(-\beta_d \delta / \tau) > \pi \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

where β_d is a tuneable decay parameter, τ is a window length, and δ is the time difference from the last observation to the current observation. Imputed input can then be defined using Equation 2.11.

One drawback of all of the approaches mentioned so far is that they only process an input time series in time order. This is clearly a disadvantage for RNN-based models that attempt to accommodate missing values under a vector-based representation as any imputation that occurs corresponds to the application of a filtering approach. This is overly restrictive when models are applied to problems like interpolation or whole time-series classification since in these problems it is permissible to integrate information from both the relative past and relative future of each time point. Indeed, methods like GRU-D make maximally uninformative imputations immediately prior to the next observation, whereas smoothing approaches would be informed by the next as well as the previously observed values in the time series.

A separate line of work has looked at using data from the future as well as from the past for imputation of missing data in RNNs. Yoon et al. (2019) and Yoon et al. (2018b) present an approach based on a multi-directional RNN (M-RNN) which operates across dimensions as well as within dimensions. It consists of two separate blocks for imputing missing values. They first construct an imputation function using a bi-directional RNN that operates within a given dimension. The second imputation block constructs an imputation function using a fully connected layer that operates only across streams at a given time point. They also concatenate the input with the response indicator vector and time gap from the last observed value to the current timestamp.

Cao et al. (2018) also proposed models for imputing the missing values for multiple correlated time series using smoothing-like methods. Their basic approach learns the missing values directly in a recurrent dynamical system based on the observed data. They combine the hidden decay mechanism of GRU-D (Che et al., 2018a) and the RNN missing data imputation methods proposed in Han-Gyu Kim et al. (2017). They also propose a bidirectional variant, BRITS-I, where they model the recurrent dynamics in the forward as well as backward direction. They introduce an additional loss in the bidirectional case to enforce consistency of predictions obtained in both directions. They extend BRITS-I for correlated recurrent imputation where the imputation is a weighted sum of history-based estimation and feature-based estimation. History-based estimation is simply uncorrelated recurrent imputation while feature-based estimation is computed based on only other features available at that time. They also concatenate the input with mask variables and time gaps from the last observed value to the current time point.

Tan et al. (2020) also introduce a time-aware structure based on a GRU to handle irregular time intervals. Similar to GRU-D, they handle irregular sampling by decaying the hidden state between the successive observations. However, instead of using exponential decay as in GRU-D, they apply inverse log decay, which they claim works better:

$$\hat{\mathbf{h}}_{t-1} = \frac{1}{\log(e + \delta_t)} \odot \mathbf{h}_{t-1} \tag{2.13}$$

Tan et al. (2020) use a dual attention mechanism for dealing with missing values caused by partially observed vectors or misalignment of observation time points on different dimensions. They perform imputations using Gaussian processes and a separate time-aware GRU model and combine them using an embedding layer to get the transformed input.

Luo et al. (2018) and Guo et al. (2019) employed generative adversarial networks (GANs) to impute missing values in irregularly sampled time series data by leveraging GRU-D style models to take into account irregular sampling. They propose a 2-stage method for imputation of missing data in time series. In the first stage, they train a GAN with GRU-D as generator and discriminator. In the second stage, they train the input “noise” of the generator of the GAN so that the generated time series is as close as possible to the original incomplete time series and the generated data has high probability of being real. They achieve this by minimizing a weighted sum of reconstruction loss and discriminative loss.

Luo et al. (2019) propose a single-stage generative model to impute missing values in multivariate time series without the need of the two-stage process as in Luo et al. (2018). They use a two-layer GRU-D network (Che et al., 2018a) and a fully connected layer as generator. They begin with a zero imputed input. After a recurrent processing of the input time series using GRU-D, the last hidden state is processed using a fully connected layer which outputs a compressed low-dimensional vector. The compressed low dimensional vector acts as the input to another fully connected layer which transforms it into the initial input for the second GRU-D layer. The second GRU-D layer then outputs the generated sample at every time step. The generator model is trained in a denoising autoencoder fashion where they add some noise to the input samples and reconstruct them at the output. The discriminator model is implemented using a GRU-D layer and a fully-connected layer. The fully connected layer takes the hidden state of GRU-D and outputs the probability of being true. The task of the discriminator is to distinguish between generated sample and the input/true sample.

Che et al. (2018b) proposed a deep generative model that uses a latent hierarchical structure to capture the temporal dependencies of multivariate time series with different sampling rates. This model captures the underlying data generation process by using a VAE-based approach and learns latent hierarchical structures using learnable switches and auxiliary connections. In particular, the switches use an update-and-reuse mechanism to control the updates of the latent states of a layer based on their previous states and the lower latent layers. The auxiliary connections between time series of different sampling rates and different latent layers help the model effectively capture short-term and long-term temporal dependencies. Their inference network consists of different RNN models for time series with different sampling rates. They jointly learn the parameters of the generative model and the inference network by maximizing the ELBO. In the case of irregularly sampled time series or missing data, missing values are interpolated by adding auxiliary connections using a model trained using only the observed data points. While training, missing data points are replaced with zero in the inference network and the corresponding auxiliary connections are removed in the generative model.

In summary, a great deal of work in the area of modeling and learning from irregularly sampled time series has been based on modifications of standard recurrent neural network models. RNN-based approaches have the primary advantage that they can deal with input sequences of different lengths. However, basic RNNs have no direct ability to deal with irregular sampling and no way to deal with missing values. GRU-D and related methods attempt to solve both problems using temporal decay, but the resulting models lack true continuous time semantics. In the next section, we turn to ODE-based recurrent models, which can provide a more elegant approach to modeling continuous time data at increased computational cost.

2.6.2 ODE Based Methods

Chen et al. (2018) proposed a variational auto-encoder (VAE) model (Kingma and Welling, 2014; Rezende et al., 2014) for continuous time data based on the use of a neural network decoder combined with a latent ordinary differential equation (ODE) model. They model time series data via a latent continuous-time function $\mathbf{z}(t)$. This function is defined via a neural network representation of its gradient field that takes the form $\frac{\partial \mathbf{z}(t)}{\partial t} = f_\theta(\mathbf{z}(t))$. The full generative process for a single data case sampled from the neural ODE latent time series model is shown below. Note that the generative process conditions on the set of observation time points $\mathbf{t} = t_0, ..., t_L$.

$$\mathbf{z}(t_0) \sim p(\mathbf{z}(t_0)) \tag{2.14}$$

$$\mathbf{z}(t_1), \cdots, \mathbf{z}(t_L) = \text{ODESolve}(\mathbf{z}(t_0), f, \theta, t_0, \cdots, t_L) \tag{2.15}$$

$$\mathbf{x}_i \sim p(\mathbf{x}_i | g_\phi(\mathbf{z}(t_i))) \tag{2.16}$$

While this model has many appealing properties as a continuous time generative model, VAE's also require the specification of a recognition network or encoder function to map observations into the latent space. Due to the deterministic nature of the ODE model, it suffices to identify the distribution over the latent space at any single time t. Chen et al. (2018) use a basic RNN model as the encoder/recognition network for computing the approximate posterior. In the case of irregularly sampled time series, Chen et al. (2018) propose to apply the encoder to the sequence of observed values backward in time from the end of the time series to the beginning.

However, the RNN encoder used did not account for variable inter-observation intervals at all and also did not account for partially observed input vectors. This means that the overall model has fairly asymmetric capabilities with the ODE used only in the generator and much more basic components applied in the encoder. Rubanova et al. (2019) subsequently proposed using an ODE-RNN model as an encoder for the latent ODE model. This yields a much more capable model with the intrinsic ability to accommodate irregularly sampled time series of fully observed vectors.

However, this model still has the limitation that it does not directly accommodate irregularly sampled time series of incompletely observed vectors. In addition, the encoder used by Rubanova et al. (2019) is also applied backwards in time only. While in theory all information in the input time series can be encoded into the latent state at the start of the time series, in practice this structure can be limiting as the encoder may not be able to adequately preserve information about structures in later sections of the time series.

De Brouwer et al. (2019) proposed a related method based on a combination of ODE and GRU components that they refer to as GRU-ODE-Bayes. This model is a continuous-time version of the Gated Recurrent Unit model (Chung et al., 2014) that can be applied to irregularly sampled time series. Instead of the encoder-decoder architecture where the ODE is decoupled from the input processing, GRU-ODE-Bayes provides a tighter integration by interleaving the ODE and the input processing steps. GRU-ODE-Bayes uses the ODE to propagate the hidden state between continuous-time observations. It updates the current hidden state whenever a new observation is available.

Contrary to the previous approaches in this subsection, GRU-ODE-Bayes can also handle partially observed vectors. Partially observed input in the RNN hidden update equation is replaced with a transformed input defined as a function (with learnable parameters) of previous hidden state, partially observed input and mask variable, similar to the approach used in the GRU-D model. However, due to the structure of the GRU-ODE-Bayes model, it only has access to information from the relative past of individual time points when accounting for missing data. This means that it suffers from the same problem as the GRU-D model where imputed values for missing dimensions have maximum uncertainty immediately before the next observation. This makes GRU-ODE-Bayes a poor choice for a smoothing or interpolation method, while the local structure of the hidden state updates make it better suited for filtering tasks. The final hidden states can also be used as input to prediction and detection tasks.

One of the disadvantages of the latent ODE models defined by Equation 2.15 is that once θ has been learned, the solution to Equation 2.15 is determined by the initial condition at z_0, and there is no direct mechanism for incorporating data or hidden state at a later time. The ODE-RNN model tries to solve this problem by using a RNN in conjunction with ODE which updates the hidden state in the presence of a new observation and then solves the ODE using this new hidden state as the initial point.

This leads to jumps in the hidden state values, which may not be suitable for defining continuous time dynamics. Kidger et al. (2020) solve this problem using Neural Controlled Differential Equations (CDEs). Neural CDEs are capable of processing incoming irregularly sampled data in a more principled way. Neural CDEs also have the capability to deal with incomplete observations by independently interpolating each channel.

In summary, ODE-based modeling primitives provide an interesting basis for learning from continuous time data. They solve the problem of accommodating irregularly sampled time series of fully observed vectors in a way that is significantly cleaner than standard RNNs. However, ODE models can be quite a bit slower during training and deployment due to the need to make repeated calls to an ODE solver. Finally, some of the model structures proposed to date are also limiting in their inability to cleanly accommodate incomplete observations. While Neural CDEs are a recent approach, they appear to have interesting advantages over the ODE-RNN framework and related models by providing an improved approach to integrating observations through time.

2.7 Attention-Based Approaches

Several recent models have leveraged attention mechanisms as their fundamental approach to dealing with irregular sampling (Horn et al., 2020; Song et al., 2018; Zhang et al., 2019). Most of these approaches are similar to Vaswani et al. (2017) where they replace the positional encoding with an encoding of time and model sequences using self-attention. Instead of adding the time encoding to the input representation as in Vaswani et al. (2017), they concatenate it with the input representation. These methods use a fixed time encoding similar to the positional encoding of Vaswani et al. (2017). Xu et al. (2019) learn a functional time representation and concatenate it with the input event embedding to model time-event interactions.

Zhang et al. (2019) proposed an attention-based time-aware approach for modeling the time irregularity between events. This approach adjusts the memory of an LSTM when accumulating previous information. Instead of reading the information from just one previous cell state, it combines values in previous cell states using attention weights and time gaps.

Choi et al. (2016b) learn an interpretable representation of irregularly sampled events using a two-level attention model. Attention vectors are generated by running RNNs backward in time. These models are subsequently used to compute a final representation using a weighted average of the event embedding with the corresponding attention weights. They showed that performance can be improved by concatenating the corresponding timestamps with the event embedding.

Xu et al. (2019) proposed a set of time embedding methods for functional time representation learning, and demonstrate their effectiveness when combined with self-attention in continuous-time event sequence prediction. The proposed functional forms are motivated from Bochner's (Loomis, 2011) and Mercer's (Mercer, 1909) theorem.

Horn et al. (2020) employed a transformer-based (Vaswani et al., 2017) approach for modeling irregularly sampled time series. This approach deals with irregular sampling by defining a time embedding to represent the time point of an observation. The time embedding defined here is a variant of positional encoding (Vaswani et al., 2017) which takes continuous time values as input. The transformer architecture was applied to the time series by concatenating the vectors of each time point with a mask variable indicating whether the variable is observed or missing and the corresponding time embedding. If the observation is missing, the input is set to zero for that dimension. The output of the transformer architecture is a sequence of embeddings, which can be aggregated into a final representation using mean-pooling for classification tasks.

2.8 Structural Invariance-Based Approaches

In this section we describe work in the area of using structural invariance as a modeling primitive. Such approaches leverage the set-based view of an irregularly sampled time series to build a model whose structure does not depend on the time order of the input time series.

In recent work, Horn et al. (2020) propose a set function-based approach for the task of classification and detection on time-series with irregularly sampled and unaligned observations. This approach can inherently handle irregular sampling and partially observed vectors. Similar to the positional encoding used in transformer models (Vaswani et al., 2017), Horn et al. (2020) define an embedding for continuous values of time. This approach represents the irregularly sampled time series data as a set of tuples consisting of a time embedding, value and dimension indicator to differentiate between different dimensions. Following the approach described in Section 2.3.5, tuples are independently transformed using a multi-layer feed-forward network. Horn et al. (2020) suggest a weighted mean approach to aggregate the encoded tuples in order to allow the model to decide which observations are relevant and which should be considered irrelevant. This is achieved by computing attention weights over the set of input elements, and subsequently, computing the weighted sum over all elements in the set corresponding to their attention weights.

2.9 Predictive Performance

In this section, we summarize evidence regarding the relative predictive performance of methods presented in the previous sections. We provide a brief introduction to the data sets that are commonly used to evaluate models for sparse and irregularly sampled time series in Section 2.10.

In terms of discretization-based methods, Lipton et al. (2016) showed that discretization combined with zero imputation and conditioning on response indicators outperformed LSTM models that do not condition on response indicators. Lipton et al. (2016) also showed that in presence of missing indicators, zero imputation performed better than forward filling based imputation. Harutyunyan et al. (2019) demonstrated the advantages of using channel-wise LSTMs and learning to predict multiple tasks using a single neural model. Song et al. (2018) showed that the performance of Harutyunyan et al. (2019) on the challenging MIMIC-III benchmark data set can be further improved by using self-attention instead of an LSTM. Bahadori and Lipton (2019) showed that a temporal clustering based approach achieves better performance than Song et al. (2018) and Harutyunyan et al. (2019) on classification and detection tasks.

However, discretization-based methods for irregular sampling are typically outperformed by models with the ability to directly use an irregularly sampled time series as input. Kim and Chi (2018) showed that a simple imputation based on the time gap between the current time and the last observed time achieved better performance than imputation methods based on mean and forward-filling on classification tasks. They also showed that using response indicators further improved performance. Choi et al. (2016b) showed that the classification performance of their attention model on irregularly sampled events can be improved by concatenating the corresponding timestamps with the event embedding.

Che et al. (2018a) and Pham et al. (2017) showed that introducing a decay function based on the time gap between current and past measurement inside an RNN model improves performance on classification and prediction tasks as compared to several off-the-shelf machine learning models and RNN baselines based on mean and forward filling imputation schemes as well as RNN models based on concatenating time gaps and missing indicators. Che et al. (2018a) also showed that for classification and prediction tasks, combining exponential decay of hidden states with an imputation scheme based on the weighted average of previous values and the empirical mean further outperforms RNNs that use exponential decay on hidden state only. The T-LSTM proposed by Baytas et al. (2017) improved on Pham et al. (2017) by transforming time gaps into weights using a decay function and using them to adjust the hidden state. Li and Xu (2019) improved on GRU-D (Che et al., 2018a) by considering the observation rate in addition to the exponential decay on hidden state and weighted imputation. Zhang et al. (2019) showed that using a time-aware attention model outperforms Choi et al. (2016b) and the T-LSTM on detection tasks.

Recurrent methods based on bidirectional RNNs typically achieve better performance than those based on single directional RNNs for both smoothing and classification tasks. Yoon et al. (2019) and Yoon et al. (2018b) proposed an MRNN based on bidirectional RNNs, which outperforms Che et al. (2018a), Futoma et al. (2017), Lipton et al. (2016) and Choi et al. (2016a) on both smoothing and detection tasks. On the smoothing (imputation) task, it also outperforms standard methods such as spline and cubic interpolation, MICE, Miss Forest, matrix completion and auto-encoder based approaches. Cao et al. (2018) show that another bidirectional RNN-based approach for learning the missing values in vector-based representations outperforms GRU-D and MRNN on both smoothing and classification tasks.

GAN-based methods for imputing missing values in irregularly sampled time series data (Luo et al., 2018, 2019) have been shown to achieve better classification performance than GRU-D. Luo et al. (2019) also outperforms Luo et al. (2018) and Yoon et al. (2018a) on classification and smoothing tasks and Cao et al. (2018) on classification tasks. The VAE-based approach of Fortuin et al. (2020) achieves better classification performance than Luo et al. (2018) and Cao et al. (2018).

ODE-based recurrent models typically achieve better classification, imputation, detection and prediction performance on irregularly sampled time series than discrete time RNN-based recurrent models. For example, Latent-ODE (Rubanova et al., 2019) outperforms GRU-D, an RNN with exponential decay on hidden state, an RNN with imputation based on the weighted average of previous values and the empirical mean, and an RNN with values concatenated with time gaps and response indicators on both classification and detection tasks. Similarly, another related method based on the combination on ODE and RNN components (De Brouwer et al., 2019) achieves better forecasting performance than GRU-D and T-LSTM. Neural CDE (Kidger et al., 2020) achieves better performance than De Brouwer et al. (2019), GRU-D and ODE-RNN (Rubanova et al., 2019) on whole time series classification tasks.

Among attention based methods, a dual-attention time-aware method introduced by Tan et al. (2020) outperforms T-LSTM and GRU-D on classification task. Horn et al. (2020) showed that a set-based and transformer-based approach for modeling irregularly sampled time series achieve comparable performance to GRU-D on classification tasks.

In Section 2.11, we provide a final distillation of the above evidence in terms of the strengths and weaknesses of different approaches.

2.10 Data Sets

The previous sections have described methods for modeling irregularly sampled time series that can be applied to a variety of inference tasks. In this section, we describe some of the data sets that have been widely used in the machine learning community to conduct experiments with irregularly sampled time series and describe some of the commonly used tasks associated with each data set.

2.10.1 MIMIC-III

The MIMIC-III data set (Johnson et al., 2016) is a de-identified data set collected at Beth Israel Deaconess Medical Center from 2001 to 2012. It consists of approximately 58,000 hospital admission records. This data set contains sparse and irregularly sampled physiological signals, medications, diagnostic codes, in-hospital mortality, length of stay, discharge summaries, progress notes, demographics information and more. Benchmark tasks on this data set include in-hospital mortality prediction, length of stay prediction and phenotype prediction. In-hospital mortality and length of stay prediction are whole time series classification and regression tasks while phenotype classification task is a multi-label classification task. The length of stay prediction task has also be framed as a classification task by discretizing the length of stay variable into a fixed number of buckets. MIMIC-III is available at `https://mimic.physionet.org/`.

2.10.2 PhysioNet 2012

The PhysioNet Challenge 2012 data set (Silva et al., 2012) consists of multivariate time series data with 37 variables extracted from intensive care unit (ICU) records. Each record contains sparse and irregularly spaced measurements from the first 48 hours after admission to ICU. The data set includes 4000 labeled instances and 4000 unlabeled instances. Benchmark tasks on this data set are in-hospital mortality prediction (whole time series classification) and interpolation. All 8000 instances can be used for interpolation experiments while only 4000 labeled instances are available for classification experiments. The PhysioNet dataset is available at `https://physionet.org/content/challenge-2012/`.

2.10.3 Human Activity

The Localization Data for Human Activity data set (Kaluža et al., 2010) consists of 3D positions of the waist, chest and ankles collected from five individuals performing several activities including walking, sitting, lying, standing, etc. Following the data preprocessing steps of Rubanova et al. (2019), a data set of $6,554$ sequences with 12 channels and 50 time points can be constructed. Labels are provided for each observation time point and denote the type of activity that the person is performing, such as walking, sitting, lying, etc. The data set consists of 11 classes.

A per-time point classification task (detection task) is the benchmark task on this data set. The data set is available at `https://archive.ics.uci.edu/ml/datasets/Localization+Data+for+Person+Activity`.

2.10.4 eICU Collaborative Research Data Set

The eICU Collaborative Database (Pollard et al., 2018) is a database relating to patients who were treated as part of the Philips eICU program across intensive care units in the United States. eICU consists of medical records from 200,859 patients collected from 208 critical care units in the United States between 2014 and 2015. The database is deidentified, and includes sparse and irregularly sampled vital sign measurements, care plan documentation, severity of illness measures, diagnosis information, treatment information, and more. The data set is available at `https://eicu-crd.mit.edu/`.

2.10.5 PhysioNet 2019

PhysioNet 2019 (Reyna et al., 2019) is a publicly available data set collected from two hospital systems. It contains 40,336 ICU patient admission records and 2,359 records of diagnosed sepsis cases. The data set consists of a set of multivariate time series that contains 40 variables including 8 vital signs, 26 laboratory values and 6 demographic variables. Each timestamp is labeled with a binary variable indicating whether the onset of sepsis has occurred. The benchmark task on this data set is the early detection of sepsis using physiological data.

2.10.6 UWave Dataset

UWave dataset is an univariate time series data consisting of simple gesture patterns divided into eight categories. The dataset has been split into 3582 train and 896 test instances. Each time series contains 945 observations. 10% of the observations points from each time series are randomly sampled to create a sparse and irregularly sampled data. The benchmark task on this data set is whole time series classification. The data set is available at `http://timeseriesclassification.com/description.php?Dataset=UWaveGestureLibraryAll`.

2.10.7 USHCN Climate Dataset

The U.S. Historical Climatology Network Monthly (USHCN) dataset (Menne et al., 2016) is a publicly available dataset consisting of daily measurements of 5 climate variables – daily maximum temperature, daily minimum temperature, whether it was a snowy day or not, total daily precipitation, and daily snow precipitation. It contains data from the last 150 years for $1,218$ meteorological stations scattered over the United States. Following the preprocessing steps of Che et al. (2018b), we extract daily climate data for 100 consecutive years starting from 1910 to 2009 from 54 stations in California. To get multi-rate time series data, we split the stations into 3 groups with sampling rates of 2 days, 1 week, and 1 month respectively. We divide the data into smaller time series consisting of yearly data and end up with a dataset of 100 examples each consisting of 270 features. Benchmark tasks on this dataset are interpolation and forecasting. The dataset is available for download at `https://cdiac.ess-dive.lbl.gov/ftp/ushcn_daily/`.

2.10.8 UCI Electricity Dataset

The UCI household electricity dataset contains measurements of seven different quantities related to electricity consumption in a household. The data are recorded every minute for 47 months between December 2006 and November 2010, yielding over 2 million observations. To simulate irregular sampling, we keep observations only at durations sampled from an exponential distribution with $\lambda = 20$. Following the preprocessing step of Binkowski et al. (2018), we also do random feature sampling where we choose one out of seven features at each time step. The probabilities of the features were chosen to be proportional to $[1.5^0, 1.5^1, \cdots 1.5^6]$ and randomly assigned to each feature before sampling. We divide the data into smaller time series consisting of monthly data and end up with a dataset of 1431 examples each consisting of 7 features. Benchmark tasks on this dataset are interpolation and forecasting. The dataset is available for download at `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption`.

2.11 Conclusion

In this chapter, we have discussed the relationship between the three fundamental representations of multivariate irregularly sampled time series (series-based, vector-based, and set-based), which motivate the use of different modeling primitives. We have introduced a categorization of approaches to modeling irregularly sampled time series data based on the fundamental modeling primitives used to accommodate irregular sampling. These modeling primitives include temporal discretization, interpolation, recurrence, attention and structural invariance. As we have shown, these modeling primitives underlie essentially all recent work on the modeling multivariate irregularly sampled time series.

Importantly, different modeling primitives are imbued with different strengths and weaknesses as a result of the representations that they leverage. A key example of this is the case of recurrent methods that, by leveraging the vector-based representation, must explicitly tackle the problem of incomplete vector-valued irregularly sampled observations that the interpolation and set-based primitives avoid by leveraging representations that do not yield explicit missing data when multi-dimensional observations are unaligned.

We have also defined a range of inference tasks that can be performed using irregularly sampled time series including detection, prediction, filtering, smoothing, interpolation and forecasting. As we have seen, there are also fundamental relationships between certain modeling primitives and the ability to successfully carry out particular tasks. Single-directional recurrent models and the interpolation/smoothing tasks are a prime example of this where the inherent structure of single-directional recurrent model typically results in maximally uncertain predictions immediately prior to the next observation. This is again an issue that is avoided by interpolation, attention and set-based methods that can inherently consider observations in the relative past and future of a given time point.

Finally, we have described a large number of specific models and methods categorized by the modeling primitives they build on. We believe this categorization is very useful as the models and methods based on particular modeling primitives inherit intrinsic strengths and weaknesses from the underlying modeling primitives. This includes strengths and weaknesses derived from the underlying time series representations and the relationships between modeling primitives and tasks. We summarize the identified strengths and weaknesses of each category of approaches below, as well as evidence regarding the relative predictive performance of approaches.

1. Discretization combined with imputation and any supervised model provides a simple, easy to implement, and modular baseline for solving whole time series classification and regression problems. The inclusion of response indicators as additional inputs has been found to help predictive performance under such an approach. However, the evidence indicates that such approaches are out-performed by methods based on other primitives.

2. As described above, due to the inherent filtering nature of single-directional discrete recurrent models, basic discrete RNNs are more suited for detection and prediction tasks than interpolation and smoothing problems. Bi-directional recurrent models typically out-perform single-directional models in terms of supervised tasks and can provide reasonable solutions to smoothing and interpolation problems. Using time stamps or time deltas as additional inputs appears to be helpful in overcoming the gap between representing a sequence

and representing an irregularly sampled time series. However, RNNs must also deal with the problem of incomplete vector-valued inputs. Here, more sophisticated approaches for end-to-end learning of model components (such as decay mechanisms) that enable imputation appear to be helpful.

3. ODE-based recurrent approaches provide a more elegant solution to dealing with irregular sampling than discrete RNNs and can be applied to all tasks described in Section 2.2. ODE-based approaches have been shown to outperform discrete RNN-based models on several tasks including whole time series classification, interpolation, detection, and prediction. However, ODE-RNN models still have limitations related to their nature as intrinsic filtering methods as well as their inability to accommodate incomplete vector-valued observations. The related neural CDE-based model family appears to have a number of advantages over ODE-RNN models in terms of its ability to incorporate observations through time.

4. Attention and structural invariance-based methods both break the sequential nature of recurrent approaches and have the potential for reductions in training time relative to recurrent models as a result. However, to date prior models have been able to meet but not exceed the performance of ODE and interpolation-based methods.

CHAPTER 3

INTERPOLATION PREDICTION NETWORKS

In this chapter, we present a new model architecture for supervised learning with multivariate sparse and irregularly sampled data: Interpolation-Prediction Networks. The architecture is based on the use of several semi-parametric interpolation layers organized into an interpolation network, followed by the application of a prediction network that can leverage any standard deep learning model. The interpolation network is an example of an interpolation primitive based model (Section 2.3.2). In this work, we use GRU networks (Chung et al., 2014) as the prediction network. We focus on whole time series regression and classification tasks (Section 2.2) with multivariate sparse and irregularly sampled time series.

The interpolation network allows for information contained in each input time series to contribute to the interpolation of all other time series in the model. The parameters of the interpolation and prediction networks are learned end-to-end via a composite objective function consisting of supervised and unsupervised components. The interpolation network serves the same purpose as the multivariate Gaussian process used in the work of Futoma et al. (2017), but remove the restrictions associated with the need for a positive definite covariance matrix.

Our approach also allows us to compute an explicit multi-timescale representation of the input time series, which we use to isolate information about transients (short duration events) from broader trends. Similar to the work of Lipton et al. (2016) and Che et al. (2018a), our architecture also explicitly leverages a separate information channel related to patterns of observation times. However, our representation uses a semi-parametric intensity function representation of this information that is more closely related to the work of Lasko (2014) on modeling medical event point processes.

Our architecture thus produces three output time series for each input time series: a smooth interpolation modeling broad trends in the input, a short time-scale interpolation modeling transients, and an intensity function modeling local observation frequencies.

We evaluate the proposed architecture on the MIMIC-III (Section 2.10.1) and UWave (Section 2.10.6) data set for both classification and regression tasks. Our approach outperforms a variety of simple baseline models as well as the basic and advanced GRU models introduced by Che et al. (2018a) across several metrics. We also compare our model with to the Gaussian process adapter (Li and Marlin, 2016) and multi-task Gaussian process RNN classifier (Futoma et al., 2017). Further, we perform full ablation testing of the information channels our architecture can produce to assess their impact on classification and regression performance.

3.1 Model Framework

In this section, we present the proposed modeling framework. We begin by presenting notation, followed by the model architecture and learning criteria.

3.1.1 Notation

We let $\mathcal{D} = \{(\mathbf{s}_n, y_n) | n = 1, ..., N\}$ represent a data set containing N data cases. An individual data case consists of a single target value y_n (discrete for classification and real-valued in the case of regression), as well as a D-dimensional, sparse and irregularly sampled multivariate time series $\mathbf{s}_n$. We follow the series-based data representation and notation (Section 2.1.1).

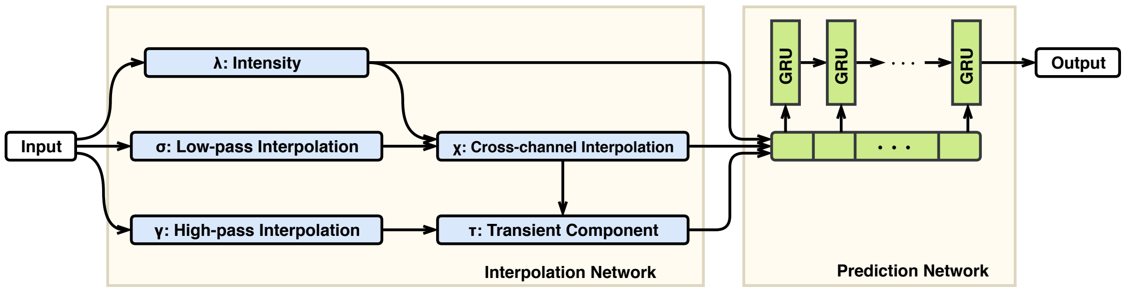


Figure 3.1: Architecture of the Interpolation-Prediction network.

3.1.2 Model Architecture

The overall model architecture consists of two main components: an interpolation network and a prediction network. The interpolation network interpolates the multivariate, sparse, and irregularly sampled input time series against a set of reference time points $\boldsymbol{\tau} = [\tau_1, ..., \tau_T]$. We assume that all of the time series are defined within a common time interval (for example, the first 24 or 48 hours after admission for MIMIC-III dataset). The T reference time points $\boldsymbol{\tau}$ are chosen to be evenly spaced within that interval. In this work, we propose a two-layer interpolation network with each layer performing a different type of interpolation.

The second component, the prediction network, takes the output of the interpolation network as its input and produces a prediction $\hat{y}_n$ for the target variable. We focus on whole time series classification and regression tasks (Section 2.2). The prediction network can consist of any standard supervised neural network architecture (fully-connected feedforward, convolutional, recurrent, etc). Thus, the architecture is fully modular with respect to the use of different prediction networks. In order to train the interpolation network, the architecture also includes an auto-encoding component to provide an unsupervised learning signal in addition to the supervised learning signal from the prediction network. Figure 3.1 shows the architecture of the proposed model. We describe the components of the model in detail below.

3.1.2.1 Interpolation Network

We begin by describing the interpolation network. The goal of the interpolation network is to provide a collection of interpolants of each of the D dimensions of an input multivariate time series defined at the T reference time points $\boldsymbol{\tau} = [\tau_1, ..., \tau_T]$. In this work, we use a total of $C = 3$ outputs for each of the D input time series. The three outputs (discussed in detail below) capture smooth trends, transients, and observation intensity information. We define $f_\theta(\boldsymbol{\tau}, \mathbf{s}_n)$ to be the function computing the output $\hat{\mathbf{s}}_n$ of the interpolation network. The output $\hat{\mathbf{s}}_n$ is a fixed-sized array with dimensions $(DC) \times T$ for all inputs $\mathbf{s}_n$.

The first layer in the interpolation network separately performs three semi-parametric univariate transformations for each of the D time series. Each transformation is based on a radial basis function (RBF) network to accommodate continuous time observations. The transformations are a low-pass (or smooth) interpolation $\boldsymbol{\sigma}_d$, a high-pass (or non-smooth) interpolation $\boldsymbol{\gamma}_d$ and an intensity function $\boldsymbol{\lambda}_d$. These transformations are computed at reference time point τ_k for each data case and each input time series d as shown in Equations 3.1, 3.2, 3.3 and 3.4.[1] The smooth interpolation $\boldsymbol{\sigma}_d$ uses a squared exponential kernel with parameter α_d, while the non-smooth interpolation $\boldsymbol{\gamma}_d$ uses a squared exponential kernel with parameter $\kappa\alpha_d$ for $\kappa > 1$.

[1]We drop the data case index n for brevity in the equations below.

$$Z(\tau, \mathbf{t}, \alpha) = \sum_{t \in \mathbf{t}} w(\tau, t, \alpha) \;, \quad w(\tau, t, \alpha) = \exp(-\alpha(\tau - t)^2) \tag{3.1}$$

$$\lambda_{kd} = h_\theta^\lambda(\tau_k, \mathbf{t}_d, \mathbf{x}_d) = Z(\tau_k, \mathbf{t}_d, \alpha_d) \tag{3.2}$$

$$\sigma_{kd} = h_\theta^\sigma(\tau_k, \mathbf{t}_d, \mathbf{x}_d) = \frac{1}{Z(\tau_k, \mathbf{t}_d, \alpha_d)} \sum_{j=1}^{L_{dn}} w(\tau_k, t_{jd}, \alpha_d)\, x_{jd} \tag{3.3}$$

$$\gamma_{kd} = h_\theta^\gamma(\tau_k, \mathbf{t}_d, \mathbf{x}_d) = \frac{1}{Z(\tau_k, \mathbf{t}_d, \kappa\alpha_d)} \sum_{j=1}^{L_{dn}} w(\tau_k, t_{jd}, \kappa\alpha_d)\, x_{jd} \tag{3.4}$$

The second interpolation layer merges information across all D time series at each reference time point by taking into account learnable correlations $\rho_{dd'}$ across all time series (Equation 3.5). This results in a cross-dimension interpolation $\boldsymbol{\chi}_d$ for each input dimension d. We further define a transient component $\boldsymbol{\eta}_d$ for each input dimension d as the difference between the high-pass (or non-smooth) interpolation $\boldsymbol{\gamma}_d$ from the first layer and the smooth cross-dimension interpolation $\boldsymbol{\chi}_d$, as shown in Equation 3.6.

$$\chi_{kd} = h_\theta^\chi(\tau_k, \mathbf{s}) = \frac{\sum_{d'} \rho_{dd'}\, \lambda_{kd'}\, \sigma_{kd'}}{\sum_{d'} \lambda_{kd'}} \tag{3.5}$$

$$\eta_{kd} = h_\theta^\eta(\tau_k, \mathbf{s}) = \gamma_{kd} - \chi_{kd} \tag{3.6}$$

In the experiments presented in the next section, we use a total of three interpolation network outputs per dimension d as the input to the prediction network. We use the smooth, cross-channel interpolants $\boldsymbol{\chi}_d$ to capture smooth trends, the transient components $\boldsymbol{\eta}_d$ to capture transients, and the intensity functions $\boldsymbol{\lambda}_d$ to capture information about where observations occur in time.

3.1.2.2 Prediction Network

Following the application of the interpolation network, all D dimensions of the input multivariate time series have been re-represented in terms of C outputs defined on the regularly spaced set of reference time points $\tau_1, ..., \tau_T$ (in our experiments, we use $C = 3$ as described above). Again, we refer to the complete set of interpolation network outputs as $\hat{\mathbf{s}}_n = f_\theta(\boldsymbol{\tau}, \mathbf{s}_n)$, which can be represented as a matrix of size $(DC) \times T$.

The prediction network must take $\hat{\mathbf{s}}_n$ as input and output a prediction $\hat{y}_n = g_\phi(\hat{\mathbf{s}}_n) = g_\phi(f_\theta(\boldsymbol{\tau}, \mathbf{s}_n))$ of the target value y_n for data case n. There are many possible choices for this component of the model. For example, the matrix $\hat{\mathbf{s}}_n$ can be converted into a single long vector and provided as input to a standard multi-layer feedforward network. A temporal convolutional model or a recurrent model like a GRU or LSTM can instead be applied to time slices of the matrix $\hat{\mathbf{s}}_n$. In this work, we conduct experiments leveraging a GRU network as the prediction network.

3.1.3 Learning

To learn the model parameters, we use a composite objective function consisting of a supervised component and an unsupervised component. This is due to the fact that the supervised component alone is insufficient to learn reasonable parameters for the interpolation network given the amount of available training data. The unsupervised component used corresponds to an autoencoder-like loss function. However, the semi-parametric RBF interpolation layers have the ability to exactly fit the input points by setting the RBF kernel parameters to very large values.

To avoid this solution and force the interpolation layers to learn to properly interpolate the input data, it is necessary to hold out some observed data points x_{jdn} during learning and then to compute the reconstruction loss only for these data points. This is a well-known problem with high-capacity autoencoders and past work has used similar strategies to avoid the problem of trivially memorizing the input data without learning useful structure.

To implement the autoencoder component of the loss, we introduce a set of masking variables m_{jdn} for each data point (t_{jdn}, x_{jdn}). If $m_{jdn} = 1$, then we remove the data point (t_{jdn}, x_{jdn}) as an input to the interpolation network and include the predicted value of this time point when assessing the autoencoder loss. We use the shorthand notation $\mathbf{m}_n \odot \mathbf{s}_n$ to represent the subset of values of $\mathbf{s}_n$ that are masked out, and $(1 - \mathbf{m}_n) \odot \mathbf{s}_n$ to represent the subset of values of $\mathbf{s}_n$ that are not masked out. The value $\hat{x}_{jdn}$ that we predict for a masked input at time point t_{jdn} is the value of the smooth cross-channel interpolant at that time point, calculated based on the un-masked input values: $\hat{x}_{jdn} = h_\theta^\chi(t_{jdn}, (1 - \mathbf{m}_n) \odot \mathbf{s}_n)$.

We can now define the learning objective for the proposed framework. We let ℓ_P be the loss for the prediction network (we use cross-entropy loss for classification and squared error for regression). We let ℓ_I be the interpolation network autoencoder loss (we use standard squared error). We also include ℓ_2 regularizers for both the interpolation and prediction networks parameters. δ_I, δ_P, and δ_R are hyper-parameters that control the trade-off between the components of the objective function.

$$\begin{aligned} \theta_*, \phi_* = \underset{\theta,\phi}{\arg\min} & \sum_{n=1}^{N} \ell_P(y_n, g_\phi(f_\theta(\mathbf{s}_n))) + \delta_I \|\theta\|_2^2 + \delta_P \|\phi\|_2^2 \\ & + \delta_R \sum_{n=1}^{N} \sum_{d=1}^{D} \sum_{j=1}^{L_{dn}} m_{jdn} \ell_I(x_{jdn}, h_\theta^\chi(t_{jdn}, (1 - \mathbf{m}_n) \odot \mathbf{s}_n)) \end{aligned} \tag{3.7}$$

3.1.4 Implementation Details

The model is learned using the Adam optimization method in TensorFlow with gradients provided via automatic differentiation. The multivariate time series representation used during learning is based on the union of all time stamps that exist in any dimension of the input time series. Undefined observations are represented as zeros and a separate auxiliary response indicator series is used to keep track of which time series have observations at each time point (Section 2.1.2). Equations 3.1 to 3.6 are modified such that data that are not available are not taken into account at all. This implementation is exactly equivalent to the computations described, but supports parallel computation across all dimensions of the time series for a given data case.

Finally, we note that the learning problem can be solved using a doubly stochastic gradient based on the use of mini batches combined with re-sampling the artificial missing data masks used in the interpolation loss. In practice, we randomly select 20% of the observed data points to hold out from every input time series.

For the time series missing entirely, our interpolation network assigns the starting point (time $t = 0$) value of the time series to the global mean before applying the two-layer interpolation network. In such cases, the first interpolation layer outputs the global mean for that channel, but the second interpolation layer performs a more meaningful interpolation using the learned correlations from other channels.

3.2 Experiments and Results

In this section, we present experiments based on both classification and regression tasks with sparse and irregularly sampled multivariate time series. In both cases, the input to the prediction network is a sparse and irregularly sampled time series, and the output is a single scalar representing either the predicted class or the regression target variable. We test the model framework on two publicly available real-world datasets: MIMIC-III (Section 2.10.1), and UWaveGesture (Section 2.10.6).

On MIMIC-III, we focus on predicting in-hospital mortality and length of stay using the first 48 hours of data. We extracted 12 standard physiological variables from each of the 53,211 records obtained after removing hospital admission records with length of stay less than 48 hours. There are 4310 (8.1%) patients with a $y_n = 1$ mortality label. Since the data set includes some very long stay durations, we let y_n represent the log of the length of stay in days for all models. We convert back from the log number of days to the number of days when reporting results. Table

3.1 shows the features, sampling rates (per hour) and their missingness information computed using the union of all time stamps that exist in any dimension of the input time series.

We use the MIMIC-III mortality and length of stay prediction tasks as example classification and regression tasks with multivariate time series. We use the UWave gesture classification task for assessing training time and performance relative to univariate baseline models.

Table 3.1: Features extracted from MIMIC III for our experiments

feature	#Missing	Sampling Rate
SpO2	31.35%	0.80
HR	23.23%	0.90
RR	59.48%	0.48
SBP	49.76%	0.59
DBP	48.73%	0.60
Temp	83.80%	0.19
TGCS	87.94%	0.14
CRR	95.08%	0.06
UO	82.47%	0.20
FiO2	94.82%	0.06
Glucose	91.47%	0.10
pH	96.25%	0.04

3.2.1 Baseline Models

We compare our proposed model to a number of baseline approaches including off-the-shelf classification and regression models learned using basic features, as well as more recent approaches based on customized neural network models.

3.2.1.1 Non-Neural Network Baselines

Following the discretization modeling primitive (Section 2.3.1), we create a fixed-size representation and use forward filling imputation to replace the missing values. We use this representation for non-neural network baselines. We evaluate Logistic

Regression (Hosmer Jr et al., 2013), Support Vector Machines (SVM) (Cortes and Vapnik, 1995), Random Forests (RF) (Breiman, 2001) and AdaBoost (Freund and Schapire, 1997) for the classification task.

For the length of stay prediction task, we apply Linear Regression (Hastie et al., 2001), Support Vector Regression (SVR), AdaBoost Regression (Drucker, 1997) and Random Forest Regression.

3.2.1.2 Neural Network Models

We compare to several existing deep learning baselines built on GRUs using simple interpolation or imputation approaches. In addition, we compare to prior state-of-the-art models for mortality prediction including the work of Che et al. (2018a). Their work proposed to handle irregularly sampled and missing data using recurrent neural networks (RNNs) by introducing temporal decays in the input and/or hidden layers (Section 2.6.1). We also evaluate the scalable end-to-end Gaussian process adapter (Li and Marlin, 2016) as well as multi-task Gaussian process RNN classifier (Futoma et al., 2017) for irregularly sampled univariate and multivariate time series classification respectively. The complete set of models that we compare to is as follows:

- **GP-GRU:** End-to-end Gaussian process with GRU as classifier.
- **GRU-M:** Missing observations replaced with the global mean of the variable across the training examples.
- **GRU-F:** Missing values set to last observed measurement within that time series (referred to as forward filling).
- **GRU-S:** Missing values replaced with the global mean. Input is concatenated with masking variable and time interval indicating how long the particular variable is missing.

- **GRU-D:** In order to capture richer information, decay is introduced in the input as well as hidden layer of a GRU. Instead of replacing missing values with the last measurement, missing values are decayed over time towards the empirical mean.

- **GRU-HD:** A variation of GRU-D where decay in only introduced in the hidden layer.

3.2.2 Empirical Protocols

The Logistic Regression model is trained with cross entropy loss with regularization strength set to 1. The support vector classifier is used with a RBF kernel and trained to minimize the soft margin loss. We use the cross entropy loss on the validation set to select the optimal number of estimators in the case of Adaboost and Random Forests. Similar to the classification setting, the optimal number of estimators for the regression task in Adaboost and Random Forests is chosen on the basis of squared error on the validation set.

3.2.2.1 MIMIC-III Experiments

We evaluate all models using a five-fold cross-validation estimate of generalization performance. In the classification setting, all the deep learning baselines are trained to minimize the cross entropy loss while the proposed model uses a composite loss consisting of cross-entropy loss and interpolation loss (with $\delta_R = 1$) as described in section 3.1.3. In the case of the regression task, all baseline models are trained to minimize squared error and the proposed model is again trained with a composite loss consisting of squared error and interpolation loss.

We follow the multi-task Gaussian process implementation given by Futoma et al. (2017) and treat the number of hidden units and hidden layers as hyper-parameters. For all of the GRU-based models, we use the already specified parameters (Che et al., 2018a). The models are learned using the Adam optimization. Early stopping is used on a validation set sub-sampled from the training folds. In the classification case, the final outputs of the GRU hidden units are used in a logistic layer that predicts the class. In the regression case, the final outputs of the GRU hidden units are used as input for a dense hidden layer with 50 units, followed by a linear output layer.

3.2.2.2 UWave Experiments

We independently tune the hyper-parameters of each baseline method. For GRU-based methods, the number of hidden units is searched over the range $\{2^5, 2^6, \cdots, 2^{11}\}$. Learning is done in same way as described above. We evaluate all the baseline models on the test set and compare the training time and accuracy. For the Gaussian process model, we use the squared exponential covariance function. We use the same number of inducing points for both the Gaussian process and the proposed model. The Gaussian process model is jointly trained with the GRU using stochastic gradient descent with Nesterov momentum. We apply early stopping based on the validation set (30% of the training set).

3.2.3 Results

In this section, we present the results of the classification and regression experiments, as well as the results of ablation testing of the internal structure of the interpolation network for the proposed model. We use the UWaveGesture dataset to assess the training time and classification performance relative to the baseline models. We use the standard train and test sets. We report the training time taken for convergence along with accuracy on test set.

For MIMIC-III, we report the results of a 5-fold cross validation experiment in terms of the average area under the ROC curve (AUC score), average area under the precision-recall curve (AUPRC score), and average cross-entropy loss for the classification task. For the regression task, we use average median absolute error and

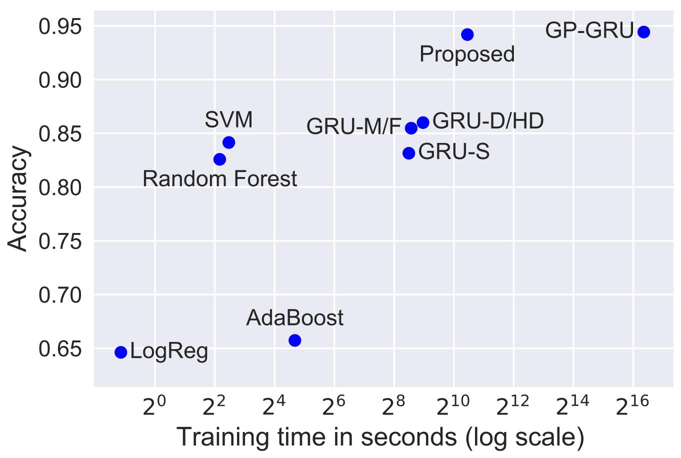


Figure 3.2: Classification performance on the UWaveGesture dataset. Models with almost same performance are shown with the same dot e.g. (GRU-M, GRU-F) and (GRU-D, GRU-HD).

average fraction of explained variation (EV) as metrics. We also report the standard deviation over cross validation folds for all metrics.

Figure 3.2 shows the classification performance on the UWaveGesture dataset. The proposed model and the Gaussian process adapter (Li and Marlin, 2016) significantly outperform the rest of the approaches. However, the proposed model achieves similar performance to the Gaussian process adapter, but with a 50x speed up (note the log scale on the training time axis). On the other hand, the training time of the proposed model is approximately the same order as other GRU-based models, but it achieves much better accuracy.

Table 3.2 compares the predictive performance of the mortality and length of stay prediction task on MIMIC-III. We note that in highly skewed datasets as is the case with MIMIC-III, AUPRC (Davis and Goadrich, 2006) can give better insight into classification performance compared to the AUC score. The proposed model consistently achieves the best average score over all metrics. We note that a paired t-test indicates that the proposed model results in statistically significant improvements over all baseline models ($p < 0.01$) with respect to all the metrics except median

Table 3.2: Performance on Mortality (classification) and Length of stay prediction (regression) tasks on MIMIC-III. Loss: Cross-Entropy Loss, MedAE: Median Absolute Error (in days), EV: Explained variance

Model	**Classification**			**Regression**	
	AUC	**AUPRC**	**Loss**	**MedAE**	**EV score**
Log/LinReg	0.772 ± 0.013	0.303 ± 0.018	0.240 ± 0.003	3.528 ± 0.072	0.043 ± 0.012
SVM	0.671 ± 0.005	0.300 ± 0.011	0.260 ± 0.002	3.523 ± 0.071	0.042 ± 0.011
AdaBoost	0.829 ± 0.007	0.345 ± 0.007	0.663 ± 0.000	4.517 ± 0.234	0.100 ± 0.012
RF	0.826 ± 0.008	0.356 ± 0.010	0.315 ± 0.025	3.113 ± 0.125	0.117 ± 0.035
GRU-M	0.831 ± 0.007	0.376 ± 0.022	0.220 ± 0.004	3.140 ± 0.196	0.131 ± 0.044
GRU-F	0.821 ± 0.007	0.360 ± 0.013	0.224 ± 0.003	3.064 ± 0.247	0.126 ± 0.025
GRU-S	0.843 ± 0.007	0.376 ± 0.014	0.218 ± 0.005	2.900 ± 0.129	0.161 ± 0.025
GRU-D	0.835 ± 0.013	0.359 ± 0.025	0.225 ± 0.009	$\mathbf{2.891 \pm 0.103}$	0.146 ± 0.051
GRU-HD	0.845 ± 0.006	0.390 ± 0.010	0.215 ± 0.004	$\mathbf{2.893 \pm 0.155}$	0.158 ± 0.037
GP-GRU	0.847 ± 0.007	0.377 ± 0.017	0.215 ± 0.004	$\mathbf{2.847 \pm 0.079}$	0.217 ± 0.020
Proposed	$\mathbf{0.853 \pm 0.007}$	$\mathbf{0.418 \pm 0.022}$	$\mathbf{0.210 \pm 0.004}$	$\mathbf{2.862 \pm 0.166}$	$\mathbf{0.245 \pm 0.019}$

absolute error. The version of the proposed model used in this experiment includes all three interpolation network outputs (smooth interpolation, transients, and intensity function).

3.2.4 Ablation Experiments

In this section, we address the question of the relative information content of the different outputs produced by the interpolation network used in the proposed model for the MIMIC-III dataset. Recall that for each of the $D = 12$ vital sign time series, the interpolation network produces three outputs: a smooth interpolation output (SI), a non-smooth or transient output (T), and an intensity function (I). The above results use all three of these outputs.

To assess the impact of each of the interpolation network outputs, we conduct a set of ablation experiments where we consider using all sub-sets of outputs for both the classification task and for the regression task.

Table 3.3: Performance of all subsets of the interpolation network outputs on Mortality (classification) and Length of stay prediction (regression) tasks. SI: Smooth Interpolation, I: Intensity, T: Transients, Loss: Cross-Entropy Loss, MedAE: Median Absolute Error, EV: Explained variance

Model	**Classification**			**Regression**	
	AUC	**AUPRC**	**Loss**	**MedAE**	**EV score**
SI, T, I	$\mathbf{0.853 \pm 0.007}$	$\mathbf{0.418 \pm 0.022}$	$\mathbf{0.210 \pm 0.004}$	2.862 ± 0.166	0.245 ± 0.019
SI, I	0.852 ± 0.005	0.408 ± 0.017	0.210 ± 0.004	2.745 ± 0.062	0.224 ± 0.010
SI, T	0.820 ± 0.008	0.355 ± 0.024	0.226 ± 0.005	2.911 ± 0.073	0.182 ± 0.009
SI	0.816 ± 0.009	0.354 ± 0.018	0.226 ± 0.005	3.035 ± 0.063	0.183 ± 0.016
I	0.786 ± 0.010	0.250 ± 0.012	0.241 ± 0.003	$\mathbf{2.697 \pm 0.072}$	0.251 ± 0.009
I, T	0.755 ± 0.012	0.236 ± 0.014	0.272 ± 0.010	2.738 ± 0.101	$\mathbf{0.290 \pm 0.010}$
T	0.705 ± 0.009	0.192 ± 0.008	0.281 ± 0.004	2.995 ± 0.130	0.207 ± 0.024

Table 3.3 shows the results from five-fold cross validation mortality and length of stay prediction experiments. When using each output individually, smooth interpolation (SI) provides the best performance in terms of classification. Interestingly, the intensity output is the best single information source for the regression task and provides at least slightly better mean performance than any of the baseline methods shown in Table 3.2. Also interesting is the fact that the transients output performs significantly worse when used alone than either the smooth interpolation or the intensity outputs in the classification task.

When considering combinations of interpolation network components, we can see that the best performance is obtained when all three outputs are used simultaneously in classification tasks. For the regression task, the intensity output provides better performance in terms of median absolute error while a combination of intensity and transients output provide better explained variance score. However, the use of the transients output contributes almost no improvement in the case of the AUC and cross entropy loss for classification relative to using only smooth interpolation and intensity. Interestingly, in the classification case, there is a significant boost in performance by

Table 3.4: Classification performance for in-hospital mortality prediction task on benchmark dataset

Model	AUC score	AUPRC score
Logistic Regression	0.8485	0.4744
LSTM	0.8547	0.4848
LSTM + Deep Supervision	0.8558	0.4928
Multitask LSTM	0.8607	0.4933
Interpolation Network + LSTM	**0.8610**	**0.5370**

combining smooth interpolation and intensity relative to using either output on its own. In the regression task, smooth interpolation appears to carry little information.

3.2.5 Additional Experiments

In this section, we compare the performance of the proposed model on a previous MIMIC-III benchmark dataset (Harutyunyan et al., 2019). This dataset only consists of patients with age > 18. Again, we focus on predicting in-hospital mortality using the first 48 hours of data. This yields training and test sets of size 17,903 and 3,236 records respectively.

We compare the proposed model to multiple baselines from Harutyunyan et al. (2019). In all the baselines, the sparse and irregularly sampled time-series data has been discretized into 1-hour intervals. If there are multiple observations in an interval, the mean or last observation is assigned to that interval, depending on the baseline method. Similarly, if an interval contains no observations, the mean or forward filling approach is used to assign a value depending on the baseline method. We compare with a logistic regression model and a standard LSTM network. In the multitask setting, multiple tasks are predicted jointly. Unlike the standard LSTM network where the output/hidden-state from the last time step is used for prediction, we provide supervision to the model at each time step. In this experiment, we use an

LSTM as the prediction network in the proposed model to match the baselines. Table 3.4 shows the classification performance of the methods in terms AUC and AUPRC score. The proposed model achieves the best score across both metrics.

3.3 Conclusion

In this chapter, we have presented a new framework for dealing with the problem of supervised learning in the presence of sparse and irregularly sampled time series. The proposed framework is fully modular. It uses an interpolation network to accommodate the complexity that results from using sparse and irregularly sampled data as supervised learning inputs, followed by the application of a prediction network that operates over the regularly spaced and fully observed, multi-channel output provided by the interpolation network. The proposed approach also addresses some difficulties with prior approaches including the complexity of the Gaussian process interpolation layers used in (Li and Marlin, 2016; Futoma et al., 2017), and the lack of modularity in the approach of Che et al. (2018a). Our framework also introduces novel elements including the use of semi-parametric, feed-forward interpolation layers, and the decomposition of an irregularly sampled input time series into multiple distinct information channels. Our results show statistically significant improvements for both classification and regression tasks over a range of baseline and state-of-the-art methods.

CHAPTER 4

MULTI-TIME ATTENTION NETWORKS

In this chapter, we show how the use of fixed RBF kernel functions can be relaxed through the use of a novel attention-based continuous-time interpolation framework: Multi-Time Attention Networks or mTANs. The primary innovation in mTANs is the inclusion of a learned continuous-time embedding mechanism coupled with a time attention mechanism that replaces the use of a fixed similarity kernel when forming representations from continuous time inputs. This gives mTANs significantly more representational flexibility than previous interpolation-based models.

Our approach re-represents an irregularly sampled time series at a fixed set of reference points similar to IP-Nets (Chapter 3). The proposed time attention mechanism uses reference time points as queries and the observed time points as keys. We propose an encoder-decoder framework for end-to-end learning using an mTAN module to interface with given multivariate, sparse and irregularly sampled time series inputs. The encoder takes the irregularly sampled time series as input and produces a fixed-length latent representation over a set of reference points, while the decoder uses the latent representations to produce reconstructions conditioned on the set of observed time points. Learning uses methods established for variational autoencoders (Rezende et al., 2014; Kingma and Welling, 2014).

The main contributions of the mTAN model framework are:

- It provides a flexible approach to modeling multivariate, sparse and irregularly sampled time series data (including irregularly sampled time series of partially observed vectors) by leveraging a time attention mechanism to learn temporal similarity from data instead of using fixed kernels.

- It uses a temporally distributed latent representation to better capture local structure in time series data.
- It provides interpolation and classification performance that is as good as current state-of-the-art methods or better, while providing significantly reduced training times.

4.1 The Multi-Time Attention Module

In this section, we present the proposed Multi-Time Attention Module (mTAN). The role of this module is to re-represent a sparse and irregularly sampled time series in a fixed-dimensional space. This module uses multiple continuous-time embeddings and attention-based interpolation. We begin by presenting the time embedding and attention components. We follow the series-based data representation (Section 2.1.1) and follow the notation introduced in Section 3.1.1.

4.1.1 Time Embedding

The time attention module is based on embedding continuous time points into a vector space. We generalize the notion of a positional encoding used in transformer-based models to continuous time. Time attention networks simultaneously leverage H embedding functions $\phi_h(t)$, each outputting a representation of size d_r. Dimension i of embedding h is defined as follows:

$$\phi_h(t)[i] = \begin{cases} \omega_{0h} \cdot t + \alpha_{0h}, & \text{if} \quad i = 0 \\ \sin(\omega_{ih} \cdot t + \alpha_{ih}), & \text{if} \quad 0 < i < d_r \end{cases} \tag{4.1}$$

where the ω_{ih}'s and α_{ih}'s are learnable parameters. The periodic terms capture the periodicity in the time series. In this case, ω_{ih} and α_{ih} represent the frequency and phase of the sine function. The linear term, on the other hand, can capture non-periodic patterns dependent on the progression of time. For a given difference Δ, $\phi_h(t+\Delta)$ can be represented as a linear function of $\phi_h(t)$.

Learning the periodic time embedding functions is equivalent to using a one-layer fully connected network with a sine function non-linearity to map the time values into a higher dimensional space. By contrast, the positional encoding used in transformer models is defined only for discrete positions. We note that our time embedding functions subsume positional encodings when evaluated at discrete positions.

4.1.2 Multi-Time Attention

The time embedding component described above takes a continuous time point and embeds it into H different d_r-dimensional spaces. In this section, we describe how we leverage time embeddings to produce a continuous-time embedding module for sparse and irregularly sampled time series. This *multi-time attention* embedding module $\text{mTAN}(t, \mathbf{s})$ takes as input a query time point t and a set of keys and values in the form of a D-dimensional multivariate sparse and irregularly sampled time series $\mathbf{s}$ (as defined in Section 2.1.1), and returns a J dimensional embedding at time t. This process leverages a continuous-time attention mechanism applied to the H time embeddings. The complete computation is described below.

$$\text{mTAN}(t, \mathbf{s})[j] = \sum_{h=1}^{H} \sum_{d=1}^{D} \hat{x}_{hd}(t, \mathbf{s}) \cdot U_{hdj} \tag{4.2}$$

$$\hat{x}_{hd}(t, \mathbf{s}) = \sum_{i=1}^{L_d} \kappa_h(t, t_{id})\, x_{id} \tag{4.3}$$

$$\kappa_h(t, t_{id}) = \frac{\exp\left(\phi_h(t) W V^T \phi_h(t_{id})^T / \sqrt{d_k}\right)}{\sum_{i'=1}^{L_d} \exp\left(\phi_h(t) W V^T \phi_h(t_{i'd})^T / \sqrt{d_k}\right)} \tag{4.4}$$

As shown in Equation 4.2, dimension j of the mTAN embedding $\text{mTAN}(t, \mathbf{s})[j]$ is given by a linear combination of intermediate univariate continuous-time functions

$\hat{x}_{hd}(t, \mathbf{s})$. There is one such function defined for each input data dimension d and each time embedding h. The parameters U_{hdj} are learnable linear combination weights.

As shown in Equation 4.3, the structure of the intermediate continuous-time function $\hat{x}_{hd}(t, \mathbf{s})$ is essentially a kernel smoother applied to the d^{th} dimension of the time series. However, the interpolation weights $\kappa_h(t, t_{id})$ are defined based on a time attention mechanism that leverages time embeddings, as shown in Equation 4.4. As we can see, the same time embedding function $\phi_h(t)$ is applied for all data dimensions. The form of the attention mechanism is a softmax function over the observed time points t_{id} for dimension d. The activation within the softmax is a scaled inner product between the time embedding $\phi_h(t)$ of the query time point t and the time embedding $\phi_h(t_{id})$ of the observed time point, the key. The parameters W and V are each $d_r \times d_k$ matrices where $d_k \leq d_r$. We use a scaling factor $\frac{1}{\sqrt{d_k}}$ to normalize the dot product to counteract the growth in the dot product magnitude with increase in the dimension d_k.

Learning the time embeddings provides our model with flexibility to learn more complex temporal kernel functions $\kappa_h(t, t')$. The use of multiple simultaneous time embeddings $\phi_h(t)$ and a final linear combination across time embedding dimensions and data dimensions means that the final output representation function $\text{mTAN}(t, \mathbf{s})$ is extremely flexible. Different input dimensions can leverage different time embeddings via learned sparsity patterns in the parameter tensor U. Information from different data dimensions can also be mixed together to create compact reduced dimensional representations. We note that all of the required computations can be parallelized using masking variables to deal with unobserved dimensions, allowing for efficient implementation on a GPU.

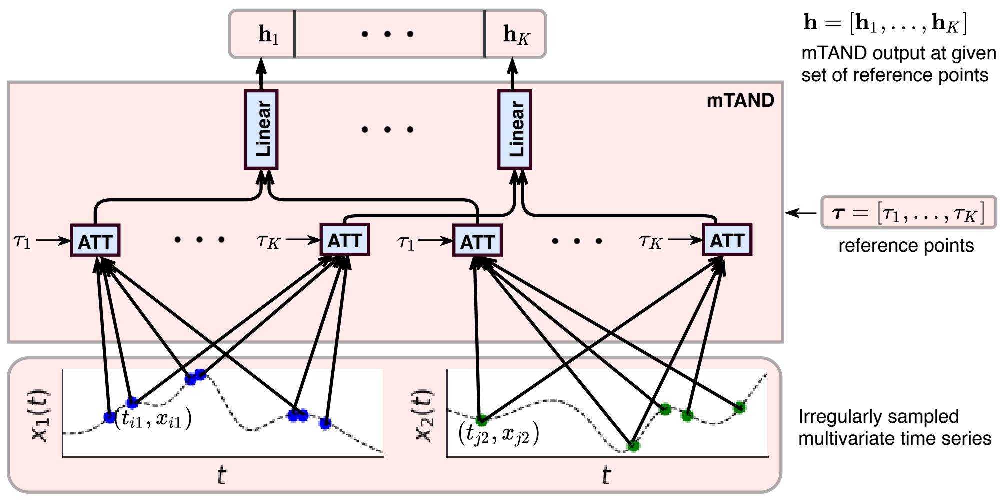
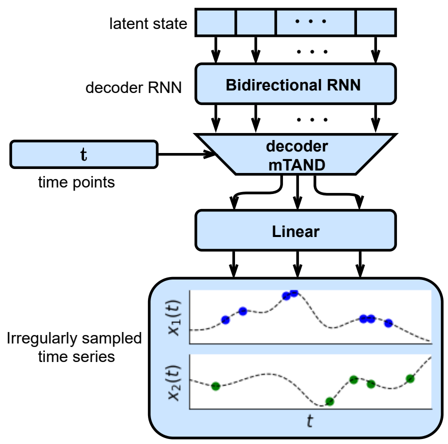

$\mathbf{h}_1$
$\mathbf{h}_K$
$\mathbf{h} = [\mathbf{h}_1, \ldots, \mathbf{h}_K]$
mTAND output at given set of reference points
mTAND
Linear
Linear
$\boldsymbol{\tau} = [\tau_1, \ldots, \tau_K]$
reference points
τ_1
ATT
τ_K
ATT
τ_1
ATT
τ_K
ATT
$x_1(t)$
(t_{i1}, x_{i1})
t
$x_2(t)$
(t_{j2}, x_{j2})
t
Irregularly sampled multivariate time series

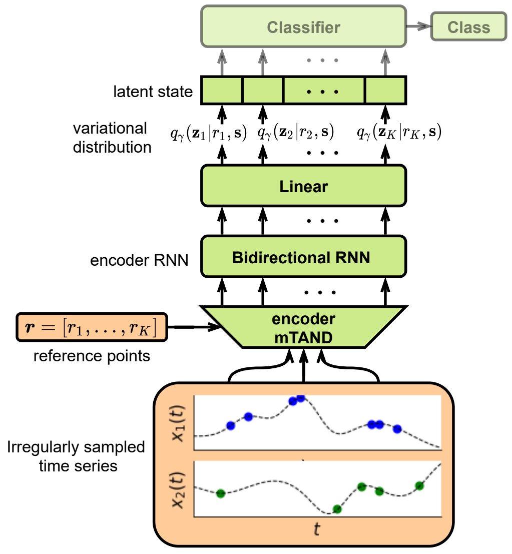
Figure 4.1: Architecture of the mTAND module. It takes irregularly sampled time points and corresponding values as keys and values and produces a fixed dimensional representation at the query time points. The attention blocks (**ATT**) perform a scaled dot product attention over the observed values using the time embedding of the query and key time points. Equation 4.3 and 4.4 defines this operation. Note that the output at all query points can be computed in parallel.

4.1.3 Discretization

Since the mTAN module defines a continuous function of t given $\mathbf{s}$, it can not be directly incorporated into neural network architectures that expect inputs in the form of fixed-dimensional vectors or discrete sequences. However, the mTAN module can easily be adapted to produce such an output representation by materializing its output at a set of reference time points $\boldsymbol{\tau} = [\tau_1, ..., \tau_K]$. In some cases, we may have a fixed set of such points. In other cases, the set of reference time points may need to depend on $\mathbf{s}$ itself. In particular, we define the auxiliary function $\rho(\mathbf{s})$ to return the set of time points at which there is an observation on any dimension of $\mathbf{s}$.

Given a collection of reference time points $\boldsymbol{\tau}$, we define the discretized mTAN module $\text{mTAND}(\boldsymbol{\tau}, \mathbf{s})$ as $\text{mTAND}(\boldsymbol{\tau}, \mathbf{s})[i] = \text{mTAN}(\tau_i, \mathbf{s})$. This module takes as input the set of reference time points $\boldsymbol{\tau}$ and the time series $\mathbf{s}$ and outputs a sequence of mTAN embeddings of length $|\boldsymbol{\tau}|$, each of dimension J. The architecture of the mTAND module is shown in Figure 4.1. The mTAND module can be used to inter-





Classifier
Class
latent state
variational distribution
$q_\gamma(\mathbf{z}_1|r_1, \mathbf{s})$
$q_\gamma(\mathbf{z}_2|r_2, \mathbf{s})$
$q_\gamma(\mathbf{z}_K|r_K, \mathbf{s})$
Linear
encoder RNN
Bidirectional RNN
$\boldsymbol{r} = [r_1, \ldots, r_K]$
reference points
encoder mTAND
Irregularly sampled time series
$x_1(t)$
$x_2(t)$
t

(a) Generative Model (Decoder) (b) Inference Network (Encoder)

Figure 4.2: Architecture of the proposed encoder-decoder framework **mTAND-Full**. The classifier is required only for performing classification tasks. The mTAND module is shown in Figure 4.1.

face sparse and irregularly sampled multivariate time series data with any deep neural network layer type including fully-connected, recurrent, and convolutional layers. In the next section, we describe the construction of a temporal encoder-decoder architecture leveraging the mTAND module, which can be applied to both classification and interpolation tasks.

4.2 Encoder-Decoder Framework

As described in the last section, we leverage the discretized mTAN module in an encoder-decoder framework as our primary model in this chapter. We develop the encoder-decoder framework within the variational autoencoder (VAE) framework in this section. The architecture for the model framework is shown in Figure 4.2.

4.2.1 Model Architecture

As we are modeling time series data, we begin by defining a sequence of latent states $\mathbf{z}_i$. Each of these latent states are iid-distributed according to a standard multivariate normal distribution $p(\mathbf{z}_i)$. We let the set of latent states be $\mathbf{z} = [\mathbf{z}_1, ..., \mathbf{z}_K]$ defined at K reference time points.

We define a three-stage decoder. First, the latent states are processed through an RNN decoder module to induce temporal dependencies, resulting in a first set of deterministic latent variables $\mathbf{h}^{dec}_{RNN} = [\mathbf{h}^{dec}_{1,RNN}, ..., \mathbf{h}^{dec}_{K,RNN}]$. Second, the output of the RNN decoder stage and the K time points $\mathbf{h}^{dec}_{RNN}$ are provided to the mTAND module along with a set of T query time points $\mathbf{t}$. The mTAND module outputs a sequence of embeddings $\mathbf{h}^{dec}_{TAN} = [\mathbf{h}^{dec}_{1,TAN}, ..., \mathbf{h}^{dec}_{T,TAN}]$ of length $|\mathbf{t}|$. Third, the mTAN embeddings are independently decoded using a fully connected decoder $f^{dec}()$ and the result is used to parameterize an output distribution. In this work, we use a diagonal covariance Gaussian distribution with mean given by the final decoded representation and a fixed variance σ^2. The final generated time series is given by $\hat{\mathbf{s}} = (\mathbf{t}, \mathbf{x})$ with all data dimensions observed. The full generative process is shown below. We let $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{t})$ define the probability distribution over the values of the time series $\mathbf{x}$ given the time points $\mathbf{t}$ and the latent variables $\mathbf{z}$. θ represents the parameters of all components of the decoder.

$$\mathbf{z}_k \sim p(\mathbf{z}_k) \tag{4.5}$$

$$\mathbf{h}^{dec}_{RNN} = \text{RNN}^{dec}(\mathbf{z}) \tag{4.6}$$

$$\mathbf{h}^{dec}_{TAN} = \text{mTAND}^{dec}(\mathbf{t}, \mathbf{h}^{dec}_{RNN}) \tag{4.7}$$

$$x_{id} \sim \mathcal{N}(x_{id}; f^{dec}(\mathbf{h}^{dec}_{i,TAN})[d], \sigma^2 \boldsymbol{I}) \tag{4.8}$$

For an encoder, we simply invert the structure of the generative process. We begin by mapping the input time series $\mathbf{s}$ through the mTAND module along with a collection of K reference time points $\boldsymbol{\tau}$. We apply an RNN encoder to the mTAND model that outputs $\mathbf{h}^{enc}_{TAN}$ to encode longer-range temporal structure. Finally, we construct a distribution over latent variables at each reference time point using a diagonal Gaussian distribution with mean and variance output by fully connected layers applied to the RNN outputs $\mathbf{h}^{enc}_{RNN}$. The complete encoder architecture is described below. We define $q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s})$ to be the distribution over the latent variables induced by the input time series $\mathbf{s}$ and the reference time points $\boldsymbol{\tau}$. γ represents all of the parameters in all of the encoder components.

$$\mathbf{h}^{enc}_{TAN} = \text{mTAND}^{enc}(\boldsymbol{\tau}, \mathbf{s}) \tag{4.9}$$

$$\mathbf{h}^{enc}_{RNN} = \text{RNN}^{enc}(\mathbf{h}^{enc}_{TAN}) \tag{4.10}$$

$$\mathbf{z}_k \sim q(\mathbf{z}_k|\boldsymbol{\mu}_k, \boldsymbol{\sigma}^2_k), \quad \boldsymbol{\mu}_k = f^{enc}_{\mu}(\mathbf{h}^{enc}_{k,RNN}), \quad \boldsymbol{\sigma}^2_k = \exp(f^{enc}_{\sigma}(\mathbf{h}^{enc}_{k,RNN})) \tag{4.11}$$

4.2.2 Unsupervised Learning

To learn the parameters of our encoder-decoder model given a data set of sparse and irregularly sampled time series, we follow a slightly modified VAE training approach and maximize a normalized variational lower bound on the log marginal likelihood based on the evidence lower bound or ELBO. The learning objective is defined below where $p_\theta(x_{jdn}|\mathbf{z}, \mathbf{t}_n)$ and $q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s}_n)$ are defined in the previous section.

$$\mathcal{L}_{\text{NVAE}}(\theta, \gamma) = \sum_{n=1}^{N} \frac{1}{\sum_d L_{dn}} \Big(\mathbb{E}_{q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}, \mathbf{t}_n)] - D_{\text{KL}}(q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s}_n)||p(\mathbf{z})) \Big) \tag{4.12}$$

$$D_{\text{KL}}(q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s}_n)||p(\mathbf{z})) = \sum_{i=1}^{K} D_{\text{KL}}(q_\gamma(\mathbf{z}_i|\boldsymbol{\tau}, \mathbf{s}_n)||p(\mathbf{z}_i)) \tag{4.13}$$

$$\log p_\theta(\mathbf{x}_n|\mathbf{z}, \mathbf{t}_n) = \sum_{d=1}^{D} \sum_{j=1}^{L_{dn}} \log p_\theta(x_{jdn}|\mathbf{z}, t_{jdn}) \tag{4.14}$$

Since irregularly sampled time series can have different numbers of observations across different dimensions as well as across different data cases, it can be helpful to normalize the terms in the standard ELBO objective to avoid the model focusing more on sequences that are longer at the expense of sequences that are shorter. The objective above normalizes the contribution of each data case by the total number of observations it contains. The fact that all data dimensions are not observed at all time points is accounted for in Equation 4.14. In practice, we use k samples from the variational distribution $q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s}_n)$ to compute the learning objective.

4.2.3 Supervised Learning

We can also augment the encoder-decoder model with a supervised learning component that leverages the latent states as a feature extractor. We define this component to be of the form $p_\delta(y_n|\mathbf{z})$ where δ are the model parameters. This leads to an augmented learning objective as shown in Equation 4.15 where the λ term trades off the supervised and unsupervised terms.

$$\mathcal{L}_{\text{supervised}}(\theta, \gamma, \delta) = \mathcal{L}_{\text{NVAE}}(\theta, \gamma) + \lambda \mathbb{E}_{q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s}_n)} \log p_\delta(y_n|\mathbf{z}) \tag{4.15}$$

In this work, we focus on classification as an illustrative supervised learning problem. For the classification model $p_\delta(y_n|\mathbf{z})$, we use a GRU followed by a 2-layer fully connected network. We use a small number of samples to approximate the required intractable expectations during both learning and prediction. Predictions are computed by marginalizing over the latent variable as shown below.

$$y^* = \underset{y \in \mathcal{Y}}{\arg\max}\ \mathbb{E}_{q_\gamma(\mathbf{z}|\boldsymbol{\tau}, \mathbf{s})}[\log p_\delta(y|\mathbf{z})] \tag{4.16}$$

4.3 Quantitative Experiments

In this section, we present interpolation and classification experiments using a range of models and three real-world data sets − Physionet Challenge 2012 (Section 2.10.2), MIMIC-III (Section 2.10.1), and the Human Activity dataset (Section 2.10.3). For MIMIC-III, we follow the data extraction process described in Section 3.2 and focus on predicting in-hospital mortality using the first 48 hours of data. On PhysioNet, we use all 8000 instances for interpolation experiments and the 4000 labeled instances for classification experiments. We focus on predicting in-hospital mortality. On the human activity dataset, we focus on classifying each time point in the sequence into one of eleven types of activities.

4.3.1 Experimental Protocols

We conduct interpolation experiments using the 8000 data cases in the PhysioNet data set. We randomly divide the data set into a training set containing 80% of the instances, and a test set containing the remaining 20% of instances. We use 20% of the training data for validation. In the interpolation task, we condition on a subset of available points and predict values for rest of the time points. We perform interpolation experiments with a varying percentage of observed points ranging from 50% to 90% of the available points. At test time, the values of observed points are conditioned on and each model is used to infer the values at the rest of the available time points in the test instance. We repeat each experiment five times using different random seeds to initialize the model parameters. We assess performance using mean squared error (MSE).

We use the labeled data in all three data sets to conduct classification experiments. The PhysioNet and MIMIC III problems are whole time series classification. Note that for the human activity dataset, we classify each time point in the time series. We treat this as a smoothing problem and condition on all available observations when producing the classification at each time-point (similar to labeling in a CRF). We use bidirectional RNNs as the RNN-based baselines for human activity dataset. We randomly divide each data set into a training set containing 80% of the time series, and a test set containing the remaining 20% of instances. We use 20% of the training set for validation. We repeat each experiment five times using different random seeds to initialize the model parameters. Due to class imbalance in the Physionet and MIMIC-III data sets, we assess classification performance using area under the ROC curve (the AUC score). For the Human Activity dataset, we evaluate models using accuracy.

4.3.2 Models

The model we focus on is the encoder-decoder architecture based on the discretized multi-time attention module (**mTAND-Full**). In the classification experiments, the hidden state at the last observed point is passed to a two-layer binary classification module for all models. For each data set, the structure of this classifier is the same for all models. For the proposed model, the sequence of latent states is first passed through a GRU and then the final hidden state is passed through the same classification module. For the classification task only, we consider an ablation of the full model that uses the proposed mTAND encoder, which consists of our mTAND module followed by a GRU to extract a final hidden state. The result is then passed to the classification module (**mTAND-Enc**). We compare to several deep learning models that expand on recurrent networks to accommodate irregular sampling. We also compare to several encoder-decoder approaches. The full list of model variants is briefly described below. We use a Gated Recurrent Unit (GRU) (Chung et al., 2014) module as the recurrent network throughout.

- **RNN-Impute:** Missing observations replaced with weighted average of last observed measurement within that time series and global mean of the variable across training examples (Che et al., 2018a).
- **RNN-Δ_t:** Input is concatenated with masking variable and time interval Δ_t indicating how long the particular variable has been missing.
- **RNN-Decay:** RNN with exponential decay on hidden states (Mozer et al., 2017; Che et al., 2018a).
- **GRU-D:** combining hidden state decay with input decay (Che et al., 2018a).
- **Phased-LSTM:** Captures time irregularity by a time gate that regulates access to the hidden and cell state of the LSTM (Neil et al., 2016) with forward filling to handle partially observed vectors.
- **IP-Nets:** Interpolation prediction networks, which use several semi-parametric RBF interpolation layers, followed by a GRU (Chapter 3).
- **SeFT:** Uses a set function based approach where all the observations are modeled individually before pooling them together using an attention based approach (Horn et al., 2020).
- **RNN-VAE:** A VAE-based model where the encoder and decoder are standard RNN models.
- **ODE-RNN:** Uses neural ODEs to model hidden state dynamics and an RNN to update the hidden state in presence of a new observation (Rubanova et al., 2019).
- **L-ODE-RNN:** Latent ODE where the encoder is an RNN and decoder is a neural ODE (Chen et al., 2018).
- **L-ODE-ODE:** Latent ODE where the encoder is an ODE-RNN and decoder is a neural ODE (Rubanova et al., 2019).

4.3.3 Architecture Details

Multi-Time Attention Network (mTAND-Full): In our proposed encoder-decoder framework (Figure 4.2), we use a bi-directional GRU as the recurrent model in both the encoder and decoder. In the encoder, we use a two-layer fully connected network with 50 hidden units and ReLU activations to map the RNN hidden state at each reference point to mean and variance. Similarly, in the decoder, mTAND embeddings are independently decoded using a two-layer fully connected network with 50 hidden units and ReLU activations. The result is used to parameterize the output distribution. For classification tasks, we use a separate GRU layer on top of the latent states followed by a 2-layer fully connected layer with 300 units and ReLU activations to output the class probabilities.

Multi-Time Attention Encoder (mTAND-Enc): As we show in the experiments, the proposed mTAN module can be used alone for classification tasks. The mTAND-Enc consists of a Multi-Time attention module followed by a GRU to extract the final hidden state, which is then passed to a 2-layer fully connected layer to output the class probabilities.

Loss Function: For computing the evidence lower bound (ELBO) during training, we use negative log-likelihood with fixed variance as the reconstruction loss. For all the datasets, we use a fixed variance of 0.01. For computing ELBO, we use five samples for the interpolation task and one sample for classification tasks. We use cross entropy loss for classification. For the classification tasks, we tune the λ parameter in the supervised learning loss function (Equation 15). We achieved best performance using λ equal to 100 and 5 for Physionet and MIMIC-III respectively. For the human activity dataset, we achieved the best results without using the regulaizer or ELBO component. We found that KL annealing with coeff 0.99 improved the performance of interpolation and classification tasks on Physionet.

4.3.4 Hyperparameters

For both interpolation and prediction, we select hyper-parameters on the held-out validation set using grid search, and then apply the best trained model to the test set.

Baselines: For the Physionet and Human Activity datasets, we use the reported hyperparameters for RNN baselines as well as ODE models from Rubanova et al. (2019). For the MIMIC-III dataset, we independently tune the hyperparameters of the baseline models on the validation set. We search for the number of GRU hidden units, the latent dimension, and the number of hidden units in the fully connected network for the ODE function in the recognition and generative models over the range $\{20, 32, 64, 128, 256\}$. For ODEs, we also searched for the number of layers in the fully connected network over the range $\{1, 2, 3\}$.

mTAN: We learn time embeddings of size 128. The number of embeddings $H \in \{1, 2, 4\}$. The linear projection matrices used for projecting the time embedding W are each $d_k \times d_k/h$ where d_k is the embedding size. We search for the latent dimension and GRU encoder hidden size over the range {32, 64, 128}. We keep the GRU decoder hidden size at 50. For the classification tasks, we use 128 reference points. For the interpolation task, we search for the number of reference points over the range $\{8, 16, 32, 64, 128\}$. We use the Adam optimizer for training the models. For classification, experiments are run for 300 iteration with learning rate 0.0001, while for the interpolation task experiments are run for 500 iterations with learning rate 0.001.

4.3.5 Results

Physionet Experiments: Table 4.1 compares the performance of all methods on the interpolation task where we observe $50\% - 90\%$ of the values in the test instances. As we can see, the proposed method (mTAND-Full) consistently and substantially

Table 4.1: Interpolation performance versus percent observed time points on PhysioNet

Model	**Mean Squared Error** ($\times 10^{-3}$)				
RNN-VAE	13.418 ± 0.008	12.594 ± 0.004	11.887 ± 0.005	11.133 ± 0.007	11.470 ± 0.006
L-ODE-RNN	8.132 ± 0.020	8.140 ± 0.018	8.171 ± 0.030	8.143 ± 0.025	8.402 ± 0.022
L-ODE-ODE	6.721 ± 0.109	6.816 ± 0.045	6.798 ± 0.143	6.850 ± 0.066	7.142 ± 0.066
mTAND-Full	$\mathbf{4.139 \pm 0.029}$	$\mathbf{4.018 \pm 0.048}$	$\mathbf{4.157 \pm 0.053}$	$\mathbf{4.410 \pm 0.149}$	$\mathbf{4.798 \pm 0.036}$
Observed %	50%	60%	70%	80%	90%

outperforms all of the previous approaches across all of the settings of observed time points. We note that in this experiment, different columns correspond to different setting (for example, in the case of 70%, we condition on 70% of data and predict the rest of the data; i.e., 30%).

Table 4.2 compares predictive performance on the PhysioNet mortality prediction task. The full Multi-Time Attention network model (mTAND-Full) and the classifier based only on the Multi-Time Attention network encoder (mTAND-Enc) achieve significantly improved performance relative to the current state-of-the-art methods (ODE-RNN and L-ODE-ODE) and other baseline methods.

We also report the time per epoch in minutes for all methods. We note that the ODE-based models require substantially more run time than other methods due to the required use of an ODE solver (Chen et al., 2018; Rubanova et al., 2019). These methods also require taking the union of all observation time points in a batch, which further slows down the training process. As we can see, the proposed full Multi-Time Attention network (mTAND-Full) is over 85 times faster than ODE-RNN and over 100 times faster than L-ODE-ODE, the best-performing ODE-based models.

MIMIC-III Experiments: Table 4.2 compares the predictive performance of the models on the mortality prediction task on MIMIC-III. The Multi-Time Attention network-based encoder-decoder framework (mTAND-Full) achieves better performance than the recent IP-Net and SeFT model as well as all of the RNN baseline models. While ODE-RNN and L-ODE-ODE both have slightly better mean AUC

Table 4.2: Classification Performance on PhysioNet, MIMIC-III and Human Activity dataset.

Model	**AUC Score**		**Accuracy**	**time**
	PhysioNet	**MIMIC-III**	**Human Activity**	**per epoch**
RNN-Impute	0.764 ± 0.016	0.8249 ± 0.0010	0.859 ± 0.004	0.5
RNN-Δ_t	0.787 ± 0.014	0.8364 ± 0.0011	0.857 ± 0.002	0.5
RNN-Decay	0.807 ± 0.003	0.8392 ± 0.0012	0.860 ± 0.005	0.7
RNN GRU-D	0.818 ± 0.008	0.8270 ± 0.0010	0.862 ± 0.005	0.7
Phased-LSTM	0.836 ± 0.003	0.8429 ± 0.0035	0.855 ± 0.005	0.3
IP-Nets	0.819 ± 0.006	0.8390 ± 0.0011	0.869 ± 0.007	1.3
SeFT	0.795 ± 0.015	0.8485 ± 0.0022	0.815 ± 0.002	0.5
RNN-VAE	0.515 ± 0.040	0.5175 ± 0.0312	0.343 ± 0.040	2.0
ODE-RNN	0.833 ± 0.009	$\mathbf{0.8561 \pm 0.0051}$	0.885 ± 0.008	16.5
L-ODE-RNN	0.781 ± 0.018	0.7734 ± 0.0030	0.838 ± 0.004	6.7
L-ODE-ODE	0.829 ± 0.004	$\mathbf{0.8559 \pm 0.0041}$	0.870 ± 0.028	22.0
mTAND-Enc	0.854 ± 0.001	0.8419 ± 0.0017	$\mathbf{0.907 \pm 0.002}$	**0.1**
mTAND-Full	$\mathbf{0.858 \pm 0.004}$	$\mathbf{0.8544 \pm 0.0024}$	$\mathbf{0.910 \pm 0.002}$	0.2

than mTAND-Full, the differences are not statistically significant. Further, as shown on the PhysioNet classification problem, mTAND-Full is more than an order of magnitude faster than the ODE-based methods.

Human Activity Experiments: Table 4.2 shows that the mTAND-based classifiers achieve significantly better performance than the baseline models on this prediction task, followed by ODE-based models and IP-Nets.

4.4 Qualitative Evaluation

In this section, we demonstrate the effectiveness of learning temporally distributed latent representations with mTANs on a synthetic dataset. We generate a synthetic dataset consisting of 1000 trajectories each of 100 time points sampled over $t \in [0, 1]$. We fix 10 reference points and use an RBF kernel ($\kappa(t, t') = \exp(-\gamma(t - t')^2)$ with $\gamma = 100$ for constructing local interpolations at 100 time points over $[0, 1]$. The values at the reference points are drawn from a standard normal distribution.

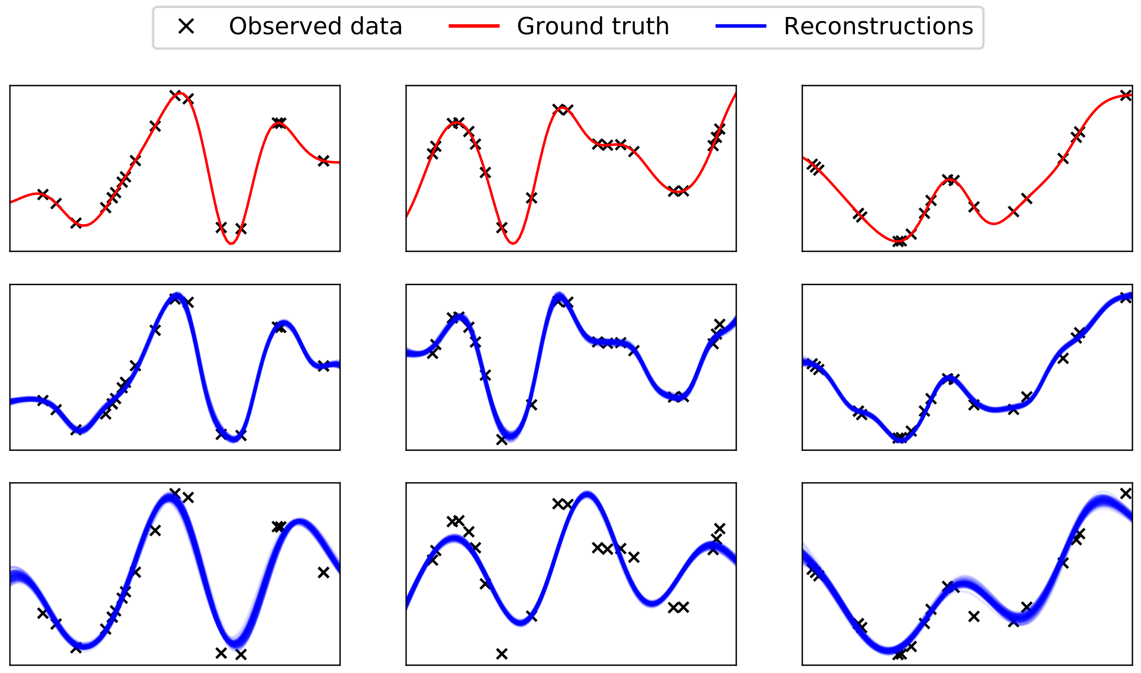


Figure 4.3: Interpolations on the synthetic interpolation dataset. The columns represent 3 different examples. First row: Ground truth trajectories with observed points, second row: reconstructions on the complete range $t \in [0, 1]$ using the proposed model mTAN, third row: reconstructions on the complete range $t \in [0, 1]$ using the Latent ODE model with ODE encoder.

We randomly sample 20 observations from each trajectory to simulate a sparse and irregularly sampled multivariate time series. We use 80% of the data for training and 20% for testing. At test time, the encoder conditions on 20 irregularly sampled time points and the decoder generates interpolations on all 100 time points. Figure 4.3 illustrates the interpolation results on the test set for the Multi-Time Attention Network and the Latent ODE model with ODE encoder (Rubanova et al., 2019). For both the models, we draw 100 samples from the approximate posterior distribution. As we can see from Figure 4.3, the ODE interpolations are much smoother and do not capture the local structure as well as mTANs. This property of mTANs helps to improve the interpolation performance in terms of mean squared error.

Table 4.3: Synthetic Data: Mean Squared Error

Latent Dimension	Model	Reconstruction	Interpolation
10	L-ODE-ODE	0.0209	0.0571
	mTAND-Full	**0.0088**	**0.0409**
20	L-ODE-ODE	0.0191	0.0541
	mTAND-Full	**0.0028**	**0.0335**

Table 4.3 compares the proposed model with best performing baseline Latent-ODE with ODE encoder (L-ODE-ODE) on the reconstruction and interpolation tasks. For both tasks, we condition on the 20 irregularly sampled time points and reconstruct the input points (reconstruction) and the whole set of 100 time points (interpolation). We report the mean squared error on the test set.

4.5 mTAN Ablation Study

In this section, we perform ablation experiments to show the performance gain achieved by learning the similarity kernel and time embedding. Table 4.4 shows the ablation results obtained by substituting a fixed positional encoding (Vaswani et al., 2017) in place of the learnable time embedding defined in Equation 4.1 in the mTAND-Full model on the PhysioNet and MIMIC-III data sets for the classification task. We report the average AUC score over 5 runs. As we can see from Table 4.4, learning the time embedding improves AUC score by 1% as compared to using fixed positional encodings.

Table 4.4: Ablation with time embedding

Dataset	Time Embedding	AUC Score
PhysioNet	Positional Encoding	0.845 ± 0.004
	Learned Time Embedding	$\mathbf{0.858 \pm 0.004}$
MIMIC-III	Positional Encoding	0.843 ± 0.001
	Learned Time Embedding	$\mathbf{0.854 \pm 0.002}$

Since mTANs are fundamentally continuous-time interpolation-based models, we perform an ablation study by comparing mTANs with the IP-Nets (Chapter 3). IP-Nets use several semi-parametric RBF interpolation layers, followed by a GRU to model irregularly sampled time series. In this framework, we replace the RBF kernel with a learnable similarity kernel using the mTAND module. This model corresponds to mTAND-Enc. Table 4.5 compares the performance of the two methods on the classification task on the PhysioNet, MIMIC-III and Human Activity data sets. We report the average AUC score over 5 runs. Table 4.5 shows that learning the similarity kernel using mTAND module performs as well or better than using a fixed RBF kernel.

Table 4.5: Comparing interpolation kernels

Dataset	**Model**	**AUC Score**
PhysioNet	IP-Nets	0.819 ± 0.006
	mTAND-Enc	$\mathbf{0.854 \pm 0.001}$
MIMIC-III	IP-Nets	$\mathbf{0.839 \pm 0.001}$
	mTAND-Enc	$\mathbf{0.842 \pm 0.001}$
Human Activity	IP-Nets	0.869 ± 0.007
	mTAND-Enc	$\mathbf{0.907 \pm 0.002}$

4.6 Visualizing Attention Weights

In this section, we visualize the attention weights learned by our proposed model. We experiment using the synthetic dataset (described in 4.4), which consists of univariate time series. Figure 4.4 shows the attention weights learned by the encoder mTAND module. The input shown in the figure is the irregularly sampled time series and the edges show how the output at reference points attends to the input time points. The final output can be computed by substituting the attention weights in Equation 4.3.

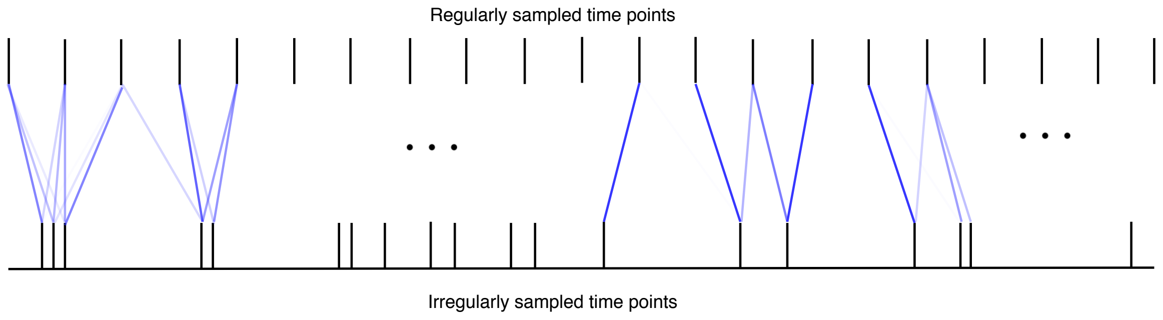

Regularly sampled time points
Irregularly sampled time points

Figure 4.4: Visualization of attention weights. mTAN learns an interpolation over the query time points by attending to the observed values at key time points. The brighter edges correspond to higher attention weights.

4.7 Conclusion

In this chapter, we have presented the Multi-Time Attention (mTAN) module for learning from sparse and irregularly sampled data along with a VAE-based encoder-decoder model leveraging this module. We show that the use of fixed RBF kernel functions used in Chapter 3 to learn temporal similarity can be relaxed through the use of a time attention mechanism coupled with a learned continuous-time embedding function. The proposed method provides significantly more representational flexibility and results in improvements over fixed RBF kernels. Our results also show that the resulting model performs as well or better than a range of baseline and state-of-the-art models on both the interpolation and classification tasks, while offering training times that are one to two orders of magnitude faster than previous state of the art methods.

CHAPTER 5

HETEROSCEDASTIC TEMPORAL VARIATIONAL AUTOENCODER

The mTAN framework introduced in Chapter 4 has been shown to provide state-of-the-art classification and deterministic interpolation performance on irregularly sampled time series. However, like many VAEs, the mTAN architecture produces a homoscedastic output distribution conditioned on the latent state. This means that the model can only reflect uncertainty due to variable input sparsity through variations in the VAE latent state. As we will show, this mechanism is insufficient to capture differences in uncertainty over time. On the other hand, Gaussian Process Regression-based (GPR) methods (Rasmussen and Williams, 2006) have the ability to reflect variable uncertainty through the posterior inference process. The main drawbacks of GPR-based methods are their significantly higher run times during both training and inference, and the added restriction of needing to define positive definite covariance functions for multivariate time series (Section 2.5).

In this chapter, we propose a novel encoder-decoder architecture for multivariate probabilistic time series interpolation that we refer to as the *Heteroscedastic Temporal Variational Autoencoder* or HeTVAE. HeTVAE aims to address the challenges described above by encoding information about input sparsity using an uncertainty-aware multi-time attention network (UnTAN), flexibly capturing relationships between dimensions and time points using both probabilistic and deterministic latent pathways, and directly representing variable output uncertainty via a heteroscedastic output layer.

The proposed UnTAN layer generalizes the previously introduced mTAN layer with an additional intensity network that can more directly encode information about input uncertainty due to variable sparsity. Similar to mTANs, the proposed UnTAN layer uses an attention mechanism to produce a distributed latent representation of irregularly sampled time series at a set of reference time points. The UnTAN module thus provides an interface between input multivariate, sparse and irregularly sampled time series data and more traditional deep learning components that expect fixed-dimensional or regularly spaced inputs. We combat the presence of additional local optima that arises from the use of a heteroscedastic output layer by leveraging an augmented training objective where we combine the ELBO loss with an uncertainty agnostic loss component. The uncertainty agnostic loss component helps to prevent learning from converging to local optima where the structure in the data is explained as noise.

We evaluate the proposed architecture on both synthetic and real data sets. The results show that our approach outperforms a variety of baseline models and recent approaches in terms of log likelihood, which is our primary metric of interest in the case of probabilistic interpolation. Finally, we perform ablation testing of different components of the architecture to assess their impact on interpolation performance.

5.1 Probabilistic Interpolation with the HeTVAE

In this section, we present the proposed architecture for probabilistic interpolation of irregularly sampled time series, the Heteroscedastic Temporal Variational Autoencoder (HeTVAE). HeTVAE leverages a sparsity-aware layer as the encoder and decoder in order to represent input uncertainty and propagate it to output interpolations. We begin by describing the architecture of the encoder/decoder network followed by the complete HeTVAE architecture. We follow the series-based data representation introduced in Section 2.1.1.

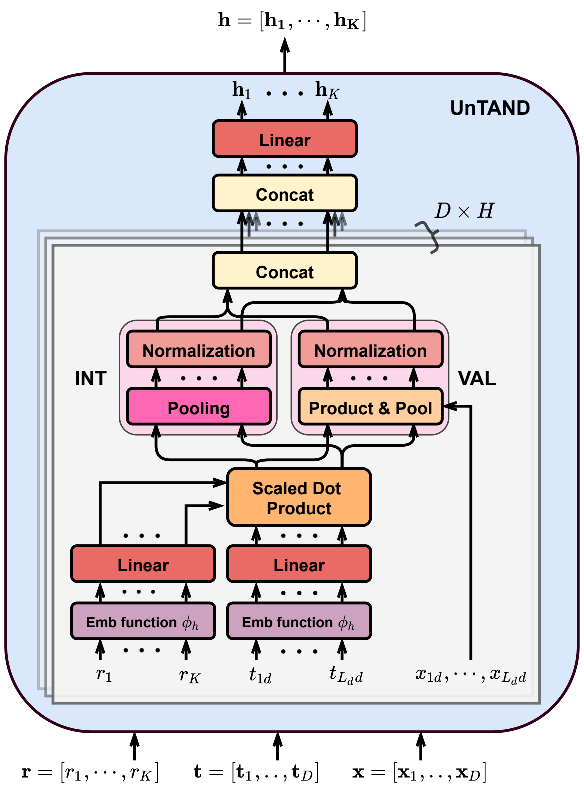

$\mathbf{h} = [\mathbf{h_1}, \cdots, \mathbf{h_K}]$
$\mathbf{h}_1 \ldots \mathbf{h}_K$
UnTAND
Linear
Concat
$D \times H$
Concat
Normalization
Normalization
INT
Pooling
Product & Pool
VAL
Scaled Dot Product
Linear
Linear
Emb function ϕ_h
Emb function ϕ_h
r_1
r_K
t_{1d}
$t_{L_d d}$
$x_{1d}, \cdots, x_{L_d d}$
$\mathbf{r} = [r_1, \cdots, r_K]$
$\mathbf{t} = [\mathbf{t}_1, .., \mathbf{t}_D]$
$\mathbf{x} = [\mathbf{x}_1, .., \mathbf{x}_D]$

Figure 5.1: Architecture of UnTAND module. This module takes D-dimensional irregularly sampled time points $\mathbf{t} = [\mathbf{t}_1, \cdots, \mathbf{t}_D]$ and corresponding observations $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_D]$ as keys and values and produces a fixed dimensional representation at the query time points $\mathbf{r} = [r_1, \cdots, r_K]$. Shared time embedding and attention function provide input to parallel intensity (Int) and value (Val) encoding networks, whose outputs are subsequently fused via concatenation and an additional linear encoding layer.

5.1.1 Representing Input Sparsity

As noted in the previous section, the mTAN encoder module does not represent information about input sparsity due to the normalization of the attention weights. To address this issue, we propose an augmented module that we refer to as an **Un**certainty Aware Multi-**T**ime **A**ttention **N**etwork (UnTAN). The UnTAN module is shown in Figure 5.1. This module includes two encoding pathways that leverage a shared time embedding function and a shared attention function. The first encoding pathway (the intensity pathway, INT) focuses on representing information about the sparsity of observations while the second encoding pathway (the value pathway, VAL) focuses on representing information about values of observations. The outputs of these two pathways are concatenated and mixed via a linear layer to define the final output of the module. The mathematical description of the module is given in Equations 5.1 to 5.3 and is explained in detail below.

$$\text{int}_h(t_q, \mathbf{t}_d) = \frac{\texttt{pool}(\{\exp(\alpha_h(t_q, t_{id})) \mid t_{id} \in \mathbf{t}_d\})}{\texttt{pool}(\{\exp(\alpha_h(t_q, t_{i'u})) \mid t_{i'u} \in \mathbf{t}_u\})} \tag{5.1}$$

$$\text{val}_h(t_q, \mathbf{t}_d, \mathbf{x}_d) = \frac{\texttt{pool}(\{\exp(\alpha_h(t_q, t_{id})) \cdot x_{id} \mid t_{id} \in \mathbf{t}_d, x_{id} \in \mathbf{x}_d\})}{\texttt{pool}(\{\exp(\alpha_h(t_q, t_{i'd})) \mid t_{i'd} \in \mathbf{t}_d\})} \tag{5.2}$$

$$\alpha_h(t, t') = \left(\frac{\phi_h(t)\mathbf{w}\mathbf{v}^T\phi_h(t')^T}{\sqrt{d_e}}\right) \tag{5.3}$$

Time Embeddings and Attention Weights: Similar to the mTAN module, the UnTAN module uses time embedding functions $\phi_h(t)$ to project univariate time values into a higher dimensional space. Each time embedding function is a one-layer fully connected network with a sine function non-linearity $\phi_h(t) = \sin(\omega \cdot t + \beta)$. We learn H time embeddings each of dimension d_e. $\mathbf{w}$ and $\mathbf{v}$ are the parameters of the scaled dot product attention function $\alpha_h(t, t')$ shown in Equation 5.3. The scaling factor $1/\sqrt{d_e}$ is used to normalize the dot product to counteract the growth in the dot product magnitude with increase in the time embedding dimension d_e.

Intensity Encoding: The intensity encoding pathway is defined by the function $\text{int}_h(t_q, \mathbf{t}_d)$ shown in Equation 5.1. ϕ_h refers to the h^{th} time embedding function. The inputs to the intensity function are a query time point t_q and a vector $\mathbf{t}_d$ containing all the time points at which observations are available for dimension d. The numerator of the intensity function exponentiates the attention weights between t_q and each time point in $\mathbf{t}_d$ to ensure positivity, then pools over the observed time points. The denominator of this computation is identical to the numerator, but the set of time points $\mathbf{t}_u$ that is pooled over is the union over all observed time points for dimension d from all data cases.

Intuitively, if the largest attention weight between t_q and any element of $\mathbf{t}_d$ is small relative to attention weights between t_q and the time points in $\mathbf{t}_u$, then the output of the intensity function will be low. Importantly, due to the use of the non-linear time embedding function, pairs of time points with high attention weights do not necessarily have to be close together in time meaning the notion of intensity that the network expresses is significantly generalized relative to IP-Net model (Section 3.1.2.1).

We also note that different sets could be used for $\mathbf{t}_u$ including a regularly spaced set of reference time points. One advantage of using the union of all observed time points is that it fixes the maximum value of the intensity function at 1. The two pooling functions applicable in the computation of the intensity function are *max* and *sum*. If the time series is sparse, *max* works well because using *sum* in the sparse case can lead to very low output values. In a more densely observed time series, either *sum* or *max* can be used.

Value Encoding: The value encoding function $\text{val}_h(t_q, \mathbf{t}_d, \mathbf{x}_d)$ is presented in Equation 5.2 in a form that highlights the symmetry with the intensity encoding function. The primary differences are that $\text{val}_h(t_q, \mathbf{t}_d, \mathbf{x}_d)$ takes as input both observed time points $\mathbf{t}_d$ and their corresponding values $\mathbf{x}_d$, and the denominator of the function pools over $\mathbf{t}_d$ itself. While different pooling options could be used for this function, in practice we use sum-based pooling. These choices lead to a function $\text{val}_h(t_q, \mathbf{t}_d, \mathbf{x}_d)$ that interpolates the observed values at the query time points using softmax weights derived from the attention function. The values of observed points with higher attention weights contribute more to the output value. This structure is equivalent to that used in the mTAN module when sum-based pooling is used. We can also clearly see that this function on its own can not represent information about input sparsity due to the normalization over $\mathbf{t}_d$. Indeed, the function is completely invariant to an additive decrease in all of the attention weights $\alpha'_h(t_q, t_{id}) = \alpha_h(t_q, t_{id}) - \delta$.

Module Output: The last stage of the UnTAN module concatenates the value and intensity pathway representations and then linearly weights them together to form the final J-dimensional representation that is output by the module. The parameters of this linear stage of the model are U^{int}_{hdj} and U^{val}_{hdj}. The value of the j^{th} dimension of the output at a query time point t_q is given by Equation 5.4.

$$\text{UnTAN}(t_q, \mathbf{t}, \mathbf{x})[j] = \sum_{h=1}^{H}\sum_{d=1}^{D} \begin{bmatrix} \text{int}_h(t_q, \mathbf{t}_d) \\ \\ \text{val}_h(t_q, \mathbf{t}_d, \mathbf{x}_d) \end{bmatrix}^T \begin{bmatrix} U^{int}_{hdj} \\ \\ U^{val}_{hdj} \end{bmatrix} \tag{5.4}$$

Finally, similar to the mTAN module, the UnTAN module can be adapted to produce fully observed fixed-dimensional discrete sequences by materializing its output at a set of reference time points. For a given set of reference time points $\mathbf{r} = [r_1, \cdots, r_K]$, the discretized UnTAN module UnTAND$(\mathbf{r}, \mathbf{t}, \mathbf{x})$ is defined as UnTAND$(\mathbf{r}, \mathbf{t}, \mathbf{x})[i] = $ UnTAN$(r_i, \mathbf{t}, \mathbf{x})$. This module takes as input the time series $\mathbf{s} = (\mathbf{t}, \mathbf{x})$ and the set of reference time points $\mathbf{r}$ and outputs a sequence of K UnTAN embeddings, each of dimension J corresponding to each reference point. As described in the next section, we use the UnTAND module to provide an interface between sparse and irregularly sampled data and fully connected MLP network structures.

5.1.2 The HeTVAE Model

In this section, we describe the overall architecture of the HeTVAE model, as shown in Figure 5.2. The HeTVAE consists of parallel deterministic and probabilistic pathways for propagating input information to the output distribution, including information about input sparsity. We begin by mapping the input time series $\mathbf{s} = (\mathbf{t}, \mathbf{x})$ through the UnTAND module along with a collection of K reference time points $\mathbf{r}$. In the probabilistic path, we construct a distribution over latent variables

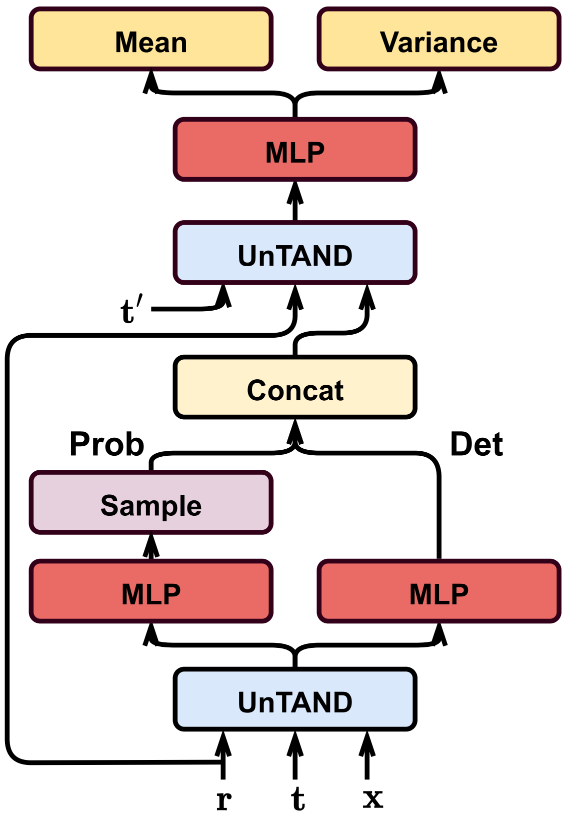


Figure 5.2: Architecture of HeTVAE consisting of the UnTAND module to represent input uncertainty, parallel probabilistic (Prob) and deterministic (Det) encoding paths, and a heteroscedastic output distribution that aims to reflect uncertainty due to input sparsity in the output distribution.

at each reference time point using a diagonal Gaussian distribution q with mean and variance output by fully connected layers applied to the UnTAND output embeddings $\mathbf{h}^{enc} = [\mathbf{h}_1^{enc}, \cdots, \mathbf{h}_K^{enc}]$ as shown in Equation 5.6. In the deterministic path, the UnTAND output embeddings $\mathbf{h}^{enc}$ are passed through a feed-forward network g to produce a deterministic temporal representation (at each reference point) of the same dimension as the probabilistic latent state.

The decoder takes as input the representation from both pathways along with the reference time points and a set of query points $\mathbf{t}'$ (Equation 5.8). The UnTAND module produces a sequence of embeddings $\mathbf{h}^{dec} = [\mathbf{h}_1^{dec}, \cdots, \mathbf{h}_{|\mathbf{t}'|}^{dec}]$ corresponding to each time point in $\mathbf{t}'$. The UnTAND embeddings are then independently decoded using a fully connected decoder f^{dec} and the result is used to parameterize the output distribution. We use a diagonal covariance Gaussian distribution where both the mean $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_{|\mathbf{t}'|}], \boldsymbol{\mu}_i \in \mathbb{R}^D$ and variance $\boldsymbol{\sigma}^2 = [\boldsymbol{\sigma}_1^2, \cdots, \boldsymbol{\sigma}_{|\mathbf{t}'|}^2], \boldsymbol{\sigma}_i^2 \in \mathbb{R}^D$ are predicted for each time point by the final decoded representation as shown in Equation 5.9. The final generated time series is sampled from this distribution and is given by $\hat{\mathbf{s}} = (\mathbf{t}', \mathbf{x}')$ with all data dimensions observed.

The complete model is described below. We define $q_\gamma(\mathbf{z}|\mathbf{r}, \mathbf{s})$ to be the distribution over the probabilistic latent variables $\mathbf{z} = [\mathbf{z}_1, \cdots, \mathbf{z}_K]$ induced by the input time series $\mathbf{s} = (\mathbf{t}, \mathbf{x})$ at the reference time points $\mathbf{r}$. We let $p_\theta^{het}(x'_{id} \,|\, \mathbf{z}^{\text{cat}}, t'_{id})$ define the final probability distribution over the value of time point t'_{id} on dimension d given the concatenated latent state $\mathbf{z}^{\text{cat}} = [\mathbf{z}_1^{\text{cat}}, \cdots, \mathbf{z}_K^{\text{cat}}]$. γ and θ represent the parameters of all components of the encoder and decoder respectively.

$$\mathbf{h}^{enc} = \text{UnTAND}^{enc}(\mathbf{r}, \mathbf{t}, \mathbf{x}) \tag{5.5}$$

$$\mathbf{z}_k \sim q_\gamma(\mathbf{z}_k \,|\, \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2), \;\; \boldsymbol{\mu}_k = f_\mu^{enc}(\mathbf{h}_k^{enc}), \;\; \boldsymbol{\sigma}_k^2 = f_\sigma^{enc}(\mathbf{h}_k^{enc}) \tag{5.6}$$

$$\mathbf{z}_k^{\text{cat}} = \text{concat}(\mathbf{z}_k, g(\mathbf{h}_k^{enc})) \tag{5.7}$$

$$\mathbf{h}^{dec} = \text{UnTAND}^{dec}(\mathbf{t}', \mathbf{r}, \mathbf{z}^{\text{cat}}) \tag{5.8}$$

$$p_\theta^{het}(x'_{id} \,|\, \mathbf{z}^{\text{cat}}, t'_{id}) = \mathcal{N}(x'_{id};\, \boldsymbol{\mu}_i\,[d],\; \boldsymbol{\sigma}_i^2\,[d]), \;\; \boldsymbol{\mu}_i = f_\mu^{dec}(\mathbf{h}_i^{dec}), \;\; \boldsymbol{\sigma}_i^2 = f_\sigma^{dec}(\mathbf{h}_i^{dec}) \tag{5.9}$$

$$x'_{id} \sim p_\theta^{het}(x'_{id} \,|\, \mathbf{z}^{\text{cat}}, t'_{id}) \tag{5.10}$$

Compared to the constant output variance used to train the mTAN-based VAE model proposed in Chapter 4, our proposed model produces a heteroscedastic output distribution that we will show provides improved modeling for the probabilistic interpolation task. However, the increased complexity of the model's output representation results in an increased space of local optima. We address this issue using an augmented learning objective, as described in the next section. Finally, we note that we can easily obtain a simplified homoscedastic version of the model with constant output variance σ_c^2 using the alternate final output distribution $p_\theta^c(x'_{id} \,|\, \mathbf{z}, t'_{id}) = \mathcal{N}(x'_{id};\, \boldsymbol{\mu}_i\,[d],\; \sigma_c^2)$.

5.1.3 Augmented Learning Objective:

To learn the parameters of the HeTVAE framework given a data set of sparse and irregularly sampled time series, we propose an augmented learning objective based on a normalized version of the evidence lower bound (ELBO) combined with an uncertainty agnostic scaled squared loss. We normalize the contribution from each data case by the total number of observations so that the effective weight of each data case in the objective function is independent of the total number of observed values.

We include the scaled squared loss term to counteract the propensity of the heteroscedastic model to become stuck in poor local optima where the mean is essentially flat and all of the structure in the data is explained as noise. Including a small contribution from the uncertainty insensitive squared loss component helps to regularize the model toward finding more informative parameters. As we will show in the experiments, the use of this augmented training procedure has a strong positive impact on final model performance. The augmented learning objective is defined below. $\boldsymbol{\mu}_n$ is the predicted mean over the test time points as defined in Equation 5.9. Also recall that the concatenated latent state $\mathbf{z}^{\text{cat}}$ depends directly on the probabilistic latent state $\mathbf{z}$.

$$\mathcal{L}_{\text{NVAE}}(\theta,\gamma) = \sum_{n=1}^{N} \frac{1}{\sum_d L_{dn}} \Big(\mathbb{E}_{q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)}[\log p_\theta^{het}(\mathbf{x}_n|\mathbf{z}_n^{\text{cat}}, \mathbf{t}_n)] - D_{\text{KL}}(q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)||p(\mathbf{z})) + \lambda\, \mathbb{E}_{q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)} \|\mathbf{x}_n - \boldsymbol{\mu}_n\|_2^2] \Big) \tag{5.11}$$

$$D_{\text{KL}}(q_\gamma(\mathbf{z}|\mathbf{r},\mathbf{s}_n)||p(\mathbf{z})) = \sum_{i=1}^{K} D_{\text{KL}}(q_\gamma(\mathbf{z}_i|\mathbf{r},\mathbf{s}_n)||p(\mathbf{z}_i)) \tag{5.12}$$

$$\log p_\theta^{het}(\mathbf{x}_n|\mathbf{z}_n^{\text{cat}}, \mathbf{t}_n) = \sum_{d=1}^{D} \sum_{j=1}^{L_{dn}} \log p_\theta^{het}(x_{jdn}|\mathbf{z}_n^{\text{cat}}, t_{jdn}) \tag{5.13}$$

5.2 Experiments

In this section, we present interpolation and classification experiments using a range of models and four real-world data sets consisting of multivariate, sparse and irregularly sampled time series data – Physionet Challenge 2012 (Section 2.10.2), MIMIC-III (Section 2.10.1), USHCN climate dataset (Section 2.10.7), and UCI electricity dataset (Section 2.10.8). We also show qualitative results on a synthetic dataset.

5.2.1 Datasets

5.2.1.1 Real Data

On PhysioNet, we use all 8000 instances for interpolation experiments and the 4000 labeled instances for classification experiments. For MIMIC-III, we follow the data extraction process described in Section 3.2 and focus on interpolation and classification experiments using all $53,211$ records. For climate and electricity dataset, we use all the records obtained after preprocessing steps described in Section 2.10.7 and 2.10.8 respectively and focus on interpolation task. We rescale time to be in $[0, 1]$ for all datasets. We also re-scale all dimensions. Specifically for PhysioNet and MIMIC-III, for each dimension we first remove outliers in the outer 0.1% percentile region. We then compute the mean and standard deviation of all observations on that dimension. The outlier detection step is used to mitigate the effect of rare large values in the data set from affecting the normalization statistics. Finally, we z-transform all of the available data (including the points identified as outliers). No data points are discarded from the data sets during the normalization process.

5.2.1.2 Synthetic Data

We also show qualitative results on a synthetic dataset to demonstrate the capabilities of our model. We generate a synthetic dataset consisting of 2000 trajectories each consisting of 50 time points with values between 0 and 1. We fix 10 reference time points and draw values for each from a standard normal distribution. We then use an RBF kernel smoother with a fixed bandwidth of $\alpha = 120.0$ to construct local interpolations over the 50 time points. The data generating process is shown below:

$$
\begin{aligned}
z_k &\sim \mathcal{N}(0,1), k \in [1, \cdots, 10] \\
r_k &= 0.1 * k \\
t_i &= 0.02 * i, i \in [1, \cdots, 50] \\
x_i &= \frac{\sum_k \exp(-\alpha(t_i - r_k)^2) \cdot z_k}{\sum_{k'} \exp(-\alpha(t_i - r_{k'})^2)} + \mathcal{N}(0, 0.1^2)
\end{aligned}
$$

We randomly sample $3-10$ observations from each trajectory to simulate a sparse and irregularly sampled univariate time series.

5.2.2 Experimental Protocols

We conduct interpolation experiments on all the datasets. We use the labeled data in Physionet and MIMIC-III to perform classification experiments. We randomly divide the real data sets into a training set containing 80% of the instances, and a test set containing the remaining 20% of instances. We use 20% of the training data for validation. In the interpolation task, we condition on a subset of available points and produce distributions over the rest of the time points. On the real-world datasets, we perform interpolation experiments by conditioning on 50% of the available points. At test time, the values of observed points are conditioned on and each model is used to infer single time point marginal distributions over values at the rest of the available time points in the test instance. In the case of methods that do not produce probabilistic outputs, we make mean predictions. In the case of the synthetic dataset where we have access to all true values, we use the observed points to infer the values at the rest of the available points. We repeat each real data experiment five times using different random seeds to initialize the model parameters. We assess performance using the negative log likelihood, which is our primary metric of interest. We also report mean squared and mean absolute error for interpolation experiments and area under ROC curve (AUC score) for classification experiments. For all experiments, we select hyper-parameters on the held-out validation set using grid search and then apply the best trained model to the test set. The hyper-parameter ranges searched for each model and dataset are fully described in Section 5.2.5.

5.2.3 Models

We compare our proposed model HeTVAE to several probabilistic and deterministic interpolation methods. We compare to two Gaussian processes regression (GPR) approaches. The most basic GP model for multivariate time series fits one GPR model per dimension. This approach is known as a single task GP model (**STGP**) (Rasmussen and Williams, 2006). A potentially better option is to model data using a Multi Task GP (**MTGP**) (Bonilla et al., 2008). This approach models the correlations both across different dimensions and across time by defining a kernel expressed as the Hadamard product of a temporal kernel (as used in the STGP) and a task kernel.

We also compare to several VAE-based approaches. These approaches use a homoscedastic output distribution with different encoder and decoder architectures. **HVAE RNN** employs a gated recurrent unit network (Chung et al., 2014) as encoder and decoder, **HVAE RNN-ODE** (Chen et al., 2018) replaces the RNN decoder with a neural ODE, **HVAE ODE-RNN-ODE** (Rubanova et al., 2019) employs a ODE-RNN encoder and neural ODE decoder. Finally, we compare to **HTVAE mTAN** (Shukla and Marlin, 2021a), a temporal VAE model consisting of multi-time attention networks producing homoscedastic output. For VAE models with homoscedastic output, we treat the output variance term as a hyperparameter and select the variance using log likelihood on the validation set. Architecture details for these methods can be found in Section 5.2.4. As baselines, we also consider deterministic mean and forward imputation-based methods. **Forward imputation** always predicts the last observed value on each dimension, while **mean imputation** predicts the mean of all the observations for each dimension.

5.2.4 Architecture Details

HeTVAE: Learnable parameters in the UnTAND architecture shown in Figure 5.1 include the weights of the three linear layers and the parameters of the shared time embedding functions. Each time embedding function is a one layer fully connected network with a sine function non-linearity. The two linear layers on top of embedding function are linear projections from time embedding dimension d_e to d_e/H where H is the number of time embeddings. Note that these linear layers do not share parameters. The third linear layer performs a linear projection from $2 \times D \times H$ to J. It takes as input the concatenation of the VAL encoder output and INT encoder output and produces an output of dimension J. d_e, H and J are all hyperparameters of the architecture. The ranges considered are described in the next section.

The HeTVAE model shown in the Figure 5.2 consists of three MLP blocks apart from the UnTAND modules. The MLP in the deterministic path is a one layer fully connected layer that projects the UnTAND output to match the dimension of the latent state. The remaining MLP blocks are two-layer fully connected networks with matching width and ReLU activations. The MLP in the decoder takes the output of the UnTAND module and outputs the mean and variance of dimension D and sequence length $|\mathbf{t}'|$. We use a softplus transformation on the decoder output to get the variance $\boldsymbol{\sigma}_i = 0.01 + \texttt{softplus}(f_\sigma^{dec}(\mathbf{h}_i^{dec}))$. Similarly, in the probabilistic path, we apply an exponential transformation to get the variance of the q distribution $\boldsymbol{\sigma}_k^2 = \exp(f_\sigma^{enc}(\mathbf{h}_k^{enc}))$. We use K reference time points regularly spaced between 0 and 1. K is considered to be a hyperparameter of the architecture. The ranges considered are described in the next section.

Baselines: For the HTVAE mTAN, we use a similar architecture as HeTVAE where we remove the deterministic path, heteroscedastic output layer and use the mTAND module instead of the UnTAND module (Shukla and Marlin, 2021a). We use the same architectures for the ODE and RNN-based VAEs as Rubanova et al. (2019).

5.2.5 Hyperparameters

HeTVAE: We fix the time embedding dimension to $d_e = 128$. The number of embeddings H is searched over the range $\{1, 2, 4\}$. We search for the number of reference points K over the range $\{4, 8, 16, 32\}$, the latent dimension over the range $\{8, 16, 32, 64, 128\}$, the output dimension of UnTAND J over the range $\{16, 32, 64, 128\}$, and the width of the two-layer fully connected layers over $\{128, 256, 512\}$. In the augmented learning objective, we search for λ over the range $\{1.0, 5.0, 10.0\}$. We use the Adam Optimizer for training the models. Experiments are run for $2,000$ iterations with a learning rate of 0.0001 and a batch size of 128. We use 100 samples from the probabilistic latent state to compute the evaluation metrics.

Ablations: We follow the same procedure as HeTVAE in tuning the hyperparamters for the several ablations. We note that the ablation HeTVAE - PROB - ALO consists of only the deterministic pathway and is trained only with the ELBO objective.

VAE Baselines: For VAE models with homoscedastic output, we treat the output variance term as a hyperparameter and select the variance over the range $\{0.01, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$. For HTVAE mTAN, we search for the corresponding hyperparameters over the same range as HeTVAE. For the ODE and RNN based VAEs, we search for GRU hidden units, latent dimension, the

Table 5.1: Interpolation performance on PhysioNet.

Model	**Negative Log Likelihood**	**Mean Absolute Error**	**Mean Squared Error**
Mean Imputation	–	0.7396 ± 0.0000	1.1634 ± 0.0000
Forward Imputation	–	0.4840 ± 0.0000	0.9675 ± 0.0000
Single-Task GP	0.7875 ± 0.0005	0.4075 ± 0.0001	0.6110 ± 0.0003
Multi-Task GP	0.9250 ± 0.0040	0.4178 ± 0.0007	0.7381 ± 0.0051
HVAE RNN	1.5220 ± 0.0019	0.7634 ± 0.0014	1.2061 ± 0.0038
HVAE RNN-ODE	1.4946 ± 0.0025	0.7372 ± 0.0026	1.1545 ± 0.0058
HVAE ODE-RNN-ODE	1.2906 ± 0.0019	0.5334 ± 0.0020	0.7622 ± 0.0027
HTVAE mTAN	1.2426 ± 0.0028	0.5056 ± 0.0004	0.7167 ± 0.0016
HeTVAE	$\mathbf{0.5542 \pm 0.0209}$	$\mathbf{0.3911 \pm 0.0004}$	$\mathbf{0.5778 \pm 0.0020}$

Table 5.2: Interpolation performance on MIMIC-III.

Model	**Negative Log Likelihood**	**Mean Absolute Error**	**Mean Squared Error**
Mean Imputation	–	0.7507 ± 0.0000	0.9842 ± 0.0000
Forward Imputation	–	0.4902 ± 0.0000	0.6148 ± 0.0000
Single-Task GP	0.8360 ± 0.0013	0.4167 ± 0.0006	0.3913 ± 0.0002
Multi-Task GP	0.8722 ± 0.0015	0.4121 ± 0.0005	0.3923 ± 0.0008
HVAE RNN	1.4380 ± 0.0049	0.7804 ± 0.0073	1.0382 ± 0.0086
HVAE RNN-ODE	1.3464 ± 0.0036	0.6864 ± 0.0069	0.8330 ± 0.0093
HVAE ODE-RNN-ODE	1.1533 ± 0.0286	0.5447 ± 0.0228	0.5642 ± 0.0334
HTVAE mTAN	1.0498 ± 0.0013	0.4931 ± 0.0008	0.4848 ± 0.0008
HeTVAE	$\mathbf{0.6662 \pm 0.0023}$	$\mathbf{0.3978 \pm 0.0003}$	$\mathbf{0.3716 \pm 0.0001}$

number of hidden units in the fully connected network for the ODE function in the encoder and decoder over the range $\{20, 32, 64, 128, 256\}$. For ODEs, we also search for the number of layers in the fully connected network in the range $\{1, 2, 3\}$. We use a batch size of 50 and a learning rate of 0.001. We use 100 samples from the latent state to compute the evaluation metrics.

Gaussian Processes: For the single task GP, we use a squared exponential kernel. In case of the multi-task GP, we experimented with the Matern kernel with different smoothness parameters, and the squared exponential kernel. We found that

Table 5.3: Interpolation performance on Climate Dataset.

Model	**Negative Log Likelihood**	**Mean Absolute Error**	**Mean Squared Error**
Mean Imputation	–	0.4539 ± 0.0000	0.8403 ± 0.0000
Forward Imputation	–	0.2979 ± 0.0000	0.8426 ± 0.0000
Single-Task GP	0.2478 ± 0.0016	$\mathbf{0.2738 \pm 0.0002}$	$\mathbf{0.4886 \pm 0.0001}$
Multi-Task GP	–	–	–
HVAE RNN	1.3666 ± 0.0674	0.4838 ± 0.0474	0.8587 ± 0.0863
HVAE RNN-ODE	1.1769 ± 0.0032	0.3514 ± 0.0067	0.6076 ± 0.0059
HVAE ODE-RNN-ODE	1.1766 ± 0.0053	0.3531 ± 0.0034	0.5953 ± 0.0051
HTVAE mTAN	0.9262 ± 0.0073	0.2916 ± 0.0046	0.5162 ± 0.0060
HeTVAE	$\mathbf{0.1287 \pm 0.0242}$	0.2813 ± 0.0034	0.5013 ± 0.0116

Table 5.4: Interpolation performance on Electricity Dataset.

Model	**Negative Log Likelihood**	**Mean Absolute Error**	**Mean Squared Error**
Mean Imputation	–	0.6765 ± 0.0000	1.0311 ± 0.0000
Forward Imputation	–	0.6163 ± 0.0000	1.2626 ± 0.0000
Single-Task GP	0.8972 ± 0.0009	0.5456 ± 0.0007	0.7827 ± 0.0005
Multi-Task GP	0.7767 ± 0.0033	0.5324 ± 0.0016	0.7782 ± 0.0006
HVAE RNN	1.3981 ± 0.0043	0.6267 ± 0.0055	0.9577 ± 0.0093
HVAE RNN-ODE	1.3947 ± 0.0054	0.6262 ± 0.0074	0.9469 ± 0.0111
HVAE ODE-RNN-ODE	1.4089 ± 0.0095	0.6453 ± 0.0042	0.9792 ± 0.0184
HTVAE mTAN	1.4040 ± 0.0148	0.6392 ± 0.0250	0.9724 ± 0.0366
HeTVAE	$\mathbf{0.7055 \pm 0.0103}$	$\mathbf{0.5049 \pm 0.0039}$	$\mathbf{0.7503 \pm 0.0162}$

the Matern kernel performs better. We use maximum marginal likelihood to train the GP hyperparameters. We search for the learning rate over the range $\{0.1, 0.01, 0.001\}$ and run for 100 iterations. We search for smoothness parameter over the range $\{0.5, 1.5, 2.5\}$. We search for the batch size over the range $\{32, 64, 128, 256\}$.

5.2.6 Interpolation Results

Tables 5.1, 5.2, 5.3 and 5.4 compare the interpolation performance of all the approaches on PhysioNet, MIMIC-III, Climate and Electricity dataset respectively. The proposed model HeTVAE outperforms the previous approaches in terms of log likelihood on all datasets. The Gaussian Process based methods – Single-Task GP and Multi-Task GP[1] achieve second and third best performance. We emphasize that while the MAE and MSE values for some of the prior approaches are close to those obtained by the HeTVAE model, the primary metric of interest for comparing probabilistic interpolation approaches is log likelihood, where the HeTVAE performs much better than the other methods. We also note that the MAE/MSE of the VAE-based models with homoscedastic output can be improved by using a small fixed variance during training. However, this produces even worse log likelihood values.

In terms of the number of parameters, GP-based approaches (STGP and MTGP) have very few learnable parameters compared to other deep learning based approaches. Hence, these approaches would have an advantage over the HeTVAE and other VAE-based approaches in very low data regimes.

5.2.7 Classification Results

Similar to mTANs, we can augment the HeTVAE model with a supervised learning component that leverages the latent states as a feature extractor (HeTVAE - Latent State). We also consider using the output interpolations as a feature extractor (HeTVAE - Output Interpolations). In this work, we focus on classification as an illustrative supervised learning problem. We use a one layer Transformer encoder (Vaswani et al., 2017) as the classification model. In order to perform classification, we use the standard approach of adding an extra learnable token whose state at the output of the Transformer encoder is passed through a two-layer fully connected layer to output the class probabilities.

We compare to several deep learning models that expand on recurrent networks to accommodate irregular sampling. We also compare to several encoder-decoder

[1]The current implementation of Multi-Task GP is not scalable to the Climate dataset (270 dimensions).

Table 5.5: Classification performance on PhysioNet.

Models	**AUC Score**	**Negative Log Likelihood**
GRU-D	0.818 ± 0.008	0.348 ± 0.002
Phased-LSTM	0.836 ± 0.003	0.330 ± 0.002
IP-Nets	0.819 ± 0.006	0.344 ± 0.005
SeFT	0.795 ± 0.015	0.363 ± 0.009
ODE-RNN	0.833 ± 0.009	0.333 ± 0.003
L-ODE-ODE	0.829 ± 0.004	0.336 ± 0.003
mTAND-Full	$\mathbf{0.858 \pm 0.004}$	$\mathbf{0.310 \pm 0.001}$
HeTVAE - Output Interpolation	$\mathbf{0.852 \pm 0.002}$	0.323 ± 0.003
HeTVAE - Latent State	$\mathbf{0.857 \pm 0.003}$	0.319 ± 0.004

Table 5.6: Classification performance on MIMIC-III.

Models	**AUC Score**	**Negative Log Likelihood**
GRU-D	0.8270 ± 0.0010	0.2213 ± 0.0010
Phased-LSTM	0.8429 ± 0.0035	0.2187 ± 0.0012
IP-Nets	0.8390 ± 0.0011	0.2216 ± 0.0006
SeFT	0.8485 ± 0.0022	0.2150 ± 0.0030
ODE-RNN	0.8561 ± 0.0051	0.2110 ± 0.0004
L-ODE-ODE	0.8559 ± 0.0041	0.2127 ± 0.0005
mTAND-Full	0.8544 ± 0.0024	0.2139 ± 0.0005
HeTVAE - Output Interpolation	0.8807 ± 0.0007	0.1992 ± 0.0004
HeTVAE - Latent State	$\mathbf{0.8874 \pm 0.0008}$	$\mathbf{0.1952 \pm 0.0005}$

approaches. The full list of model variants is described in Section 4.3.2. Table 5.5 and 5.6 compare the predictive performance of the models on the mortality prediction task on PhysioNet and MIMIC-III respectively. The HeTVAE classifier using the latent state significantly outperforms other models on the MIMIC-III dataset across both metrics, while on PhysioNet it achieves comparable performance to the mTAND-Full model introduced in Chapter 4. We can also see that the HeTVAE classifier using latent state achieves better performance than the classifier using output interpolations on the MIMIC-III, while their performance are similar on the PhysioNet dataset.

Table 5.7: Ablation Study of HeTVAE on PhysioNet.

Model	Negative Log Likelihood	Mean Absolute Error	Mean Squared Error
HetVAE	$\mathbf{0.5542 \pm 0.0209}$	$\mathbf{0.3911 \pm 0.0004}$	$\mathbf{0.5778 \pm 0.0020}$
HeTVAE - ALO	0.6087 ± 0.0136	0.4087 ± 0.0008	0.6121 ± 0.0063
HeTVAE - DET	0.6278 ± 0.0017	0.4089 ± 0.0005	0.5950 ± 0.0018
HeTVAE - INT	0.6539 ± 0.0107	0.4013 ± 0.0005	0.5935 ± 0.0015
HeTVAE - HET - ALO	1.1304 ± 0.0016	0.3990 ± 0.0003	0.5871 ± 0.0016
HeTVAE - DET - ALO	0.7425 ± 0.0066	0.4747 ± 0.0024	0.6963 ± 0.0031
HeTVAE - PROB - ALO	0.7749 ± 0.0047	0.4251 ± 0.0029	0.6230 ± 0.0040
HeTVAE - INT - DET - ALO	0.7866 ± 0.0029	0.4857 ± 0.0003	0.7120 ± 0.0007
HeTVAE - HET - INT - DET - ALO	1.2426 ± 0.0028	0.5056 ± 0.0004	0.7167 ± 0.0016

5.3 HeTVAE Ablation Study

Tables 5.7 and 5.8 show the complete results of ablating several different components of the HeTVAE model and training procedure with respect to all three evaluation metrics on PhysioNet and MIMIC-III respectively. The first row shows the results for the full proposed approach HeTVAE. We denote different components of the HeTVAE model as follows –

- HET: heteroscedastic output layer
- ALO: augmented learning objective
- INT: intensity encoding
- DET: deterministic pathway

The results show selected individual and compound ablations of these components and indicate that all of these components contribute significantly to the model's performance in terms of the negative log likelihood score. We provide detailed comments below.

5.3.1 Effect of Heteroscedastic Layer

Since the augmented learning objective is introduced to improve the learning in the presence of heteroscedastic layer, we remove the augmented learning objective

Table 5.8: Ablation Study of HeTVAE on MIMIC-III.

Model	**Negative Log Likelihood**	**Mean Absolute Error**	**Mean Squared Error**
HetVAE	$\mathbf{0.6662 \pm 0.0023}$	$\mathbf{0.3978 \pm 0.0003}$	$\mathbf{0.3716 \pm 0.0001}$
HeTVAE - ALO	0.6869 ± 0.0111	0.4043 ± 0.0006	0.3840 ± 0.0007
HeTVAE - DET	0.7478 ± 0.0028	0.4129 ± 0.0008	0.3845 ± 0.0009
HeTVAE - INT	0.7430 ± 0.0011	0.4066 ± 0.0001	0.3837 ± 0.0001
HeTVAE - HET - ALO	0.9272 ± 0.0002	0.4044 ± 0.0001	0.3765 ± 0.0001
HeTVAE - DET - ALO	0.9005 ± 0.0052	0.5177 ± 0.0004	0.5325 ± 0.0008
HeTVAE - PROB - ALO	0.7472 ± 0.0056	0.4049 ± 0.0006	0.3833 ± 0.0008
HeTVAE - INT - DET - ALO	0.9245 ± 0.0021	0.5208 ± 0.0009	0.5358 ± 0.0012
HeTVAE - HET - INT - DET - ALO	1.0498 ± 0.0013	0.4931 ± 0.0008	0.4848 ± 0.0008

(ALO) with the heteroscedastic layer (HET). This ablation corresponds to HeTVAE - HET - ALO. As we can see from both Table 5.7 and 5.8, this results in a highly significant drop in the log likelihood performance as compared to the full HeTVAE model on both datasets. However, it results in only a slight drop in performance with respect to MAE and MSE, which is sensible as the HET component only affects uncertainty sensitive performance metrics.

5.3.2 Effect of Intensity Encoding

HeTVAE - INT removes the intensity encoding pathway from the UnTAND module. It results in an immediate drop in performance on both datasets. We also compare the effect of intensity encoding after removing the deterministic pathway and the augmented learning objective. These ablations are shown in HeTVAE - DET - ALO and HeTVAE - INT - DET - ALO. The performance drop is less severe in this case because of the propensity of the heteroscedastic output layer to get stuck in poor local optima in the absence of the augmented learning objective (ALO).

5.3.3 Effect of Augmented Learning Objective

The HeTVAE - ALO ablation shows the result of removing the augmented learning objective and training the model only using only the ELBO. This results in an immediate drop in performance on PhysioNet. The performance drop is less severe on MIMIC-III. We further perform this ablation without the DET component and observe severe drops in performance across all metrics on both datasets. These ablations correspond to HeTVAE - DET and HeTVAE - DET - ALO. This shows that along with ALO component, the DET component also constrains the model from getting stuck in local optima where all of the structure in the data is explained as noise. We show interpolations corresponding to these ablations in Section 5.4.1.

5.3.4 Effect of Deterministic Pathway

HeTVAE - DET removes the deterministic pathway from the model, resulting in a performance drop on both MIMIC-III and PhysioNet across all metrics. We further compare the performance of both the probabilistic and deterministic pathways in isolation as shown by ablation HeTVAE - DET - ALO and HeTVAE - PROB - ALO. We observe that the deterministic pathway HeTVAE - PROB - ALO outperforms the probabilistic pathway HeTVAE - DET - ALO in terms of log likelihood on MIMIC-III while the opposite is true in case of PhysioNet. However, on both datasets using only the deterministic pathway (HeTVAE - PROB - ALO) achieves better MAE and MSE scores as compared to using only the probabilistic pathway (HeTVAE - DET - ALO).

5.4 Qualitative Evaluation

In this section, we show sample visualizations on PhysioNet and a synthetic dataset.

5.4.1 Interpolations on PhysioNet

Figure 5.3 shows example interpolations on the PhysioNet dataset. Following the experimental setting mentioned in Section 5.2, the models were trained using all dimensions and the inference uses all dimensions. We show interpolations corresponding to Heart Rate only as an illustration. As we can see, the STGP and HeTVAE

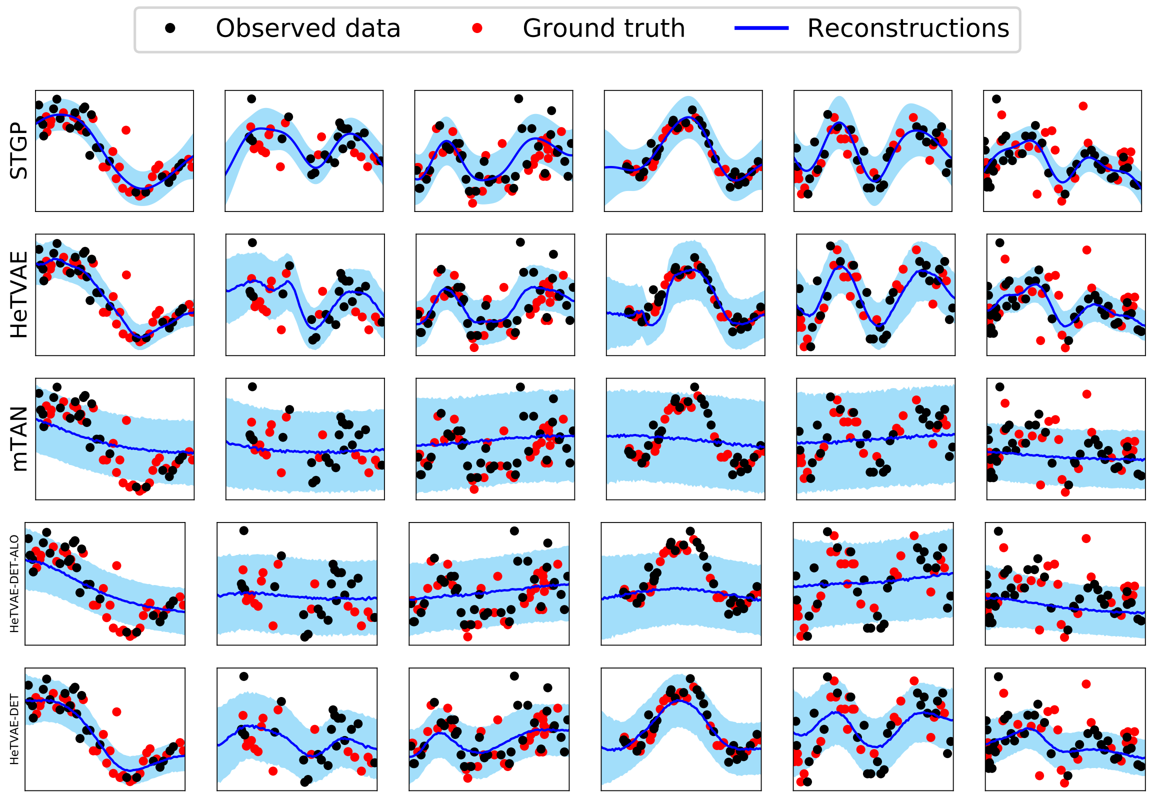


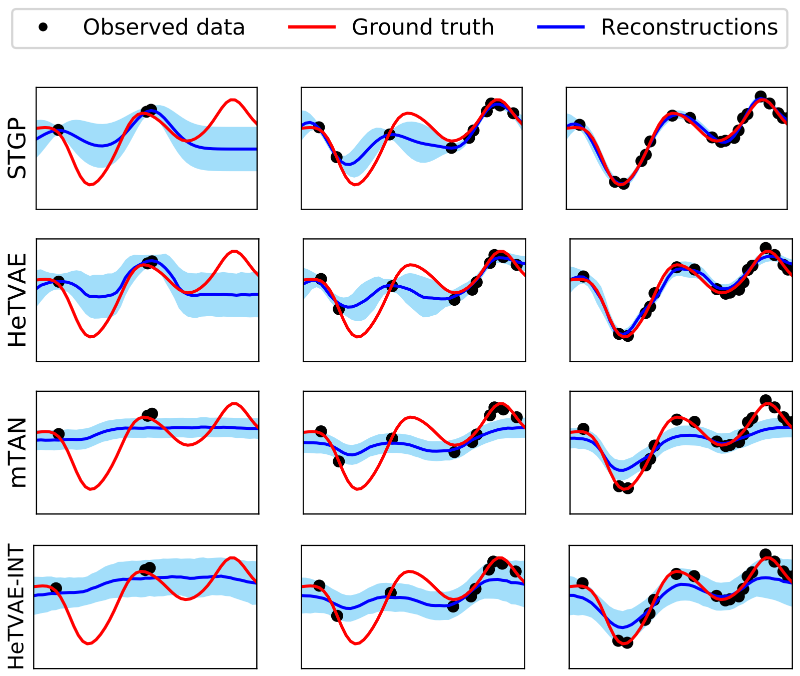
Figure 5.3: In this figure, we show example interpolations of one dimension corresponding to Heart Rate on the PhysioNet dataset. The columns correspond to different examples. The rows correspond to STGP, HeTVAE, HTVAE mTAN, HeTVAE-DET-ALO and HeTVAE-DET respectively. The shaded region corresponds to ± one standard deviation. STGP, HeTVAE and HeTVAE-DET exhibit variable output uncertainty and good fit while mTAN and HETVAE-DET-ALO does not.

models exhibit good fit and variable uncertainty on the edges where there are no observations. We can also see that mTAN trained with homoscedastic output is not able to produce as good a fit because of the fixed variance at the output (discussed in Section 5.2).

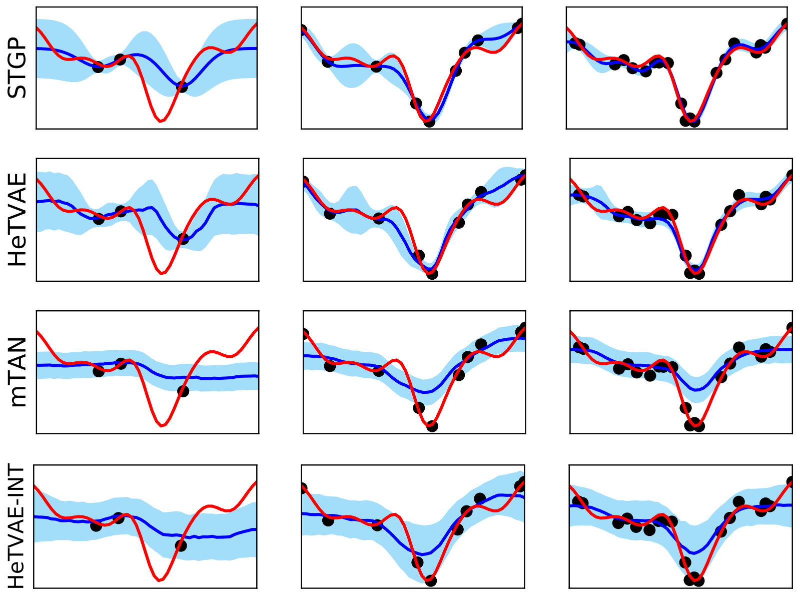
The most interesting observation is the performance of HeTVAE - DET - ALO, an ablation of HeTVAE model that retains heteroscedastic output, but removes the deterministic pathways and the augmented learning objective. This ablation significantly underfits the data and performs similar to mTAN. This is an example of local optima that arises from the use of a heteroscedastic output layer where the mean is excessively smooth and all of the structure in the data is explained as noise. We address this with the use of augmented learning objective described in Section 5.1.2. As seen in the Figure 5.3, adding the augmented learning objective (HeTVAE - DET) clearly improves performance.

5.4.2 Synthetic Data Visualizations: Sparsity

In this section, we show sample interpolation results on the synthetic dataset. The setting here is same as in Section 5.2. Figure 5.4 compares HTVAE mTAN, the single task Gaussian process STGP, the proposed HeTVAE model and an ablation of the proposed model without intensity encoding HeTVAE - INT. We vary the number of observed points $(3, 10, 20)$ and each model is used to infer the distribution over the remaining time points. We draw multiple samples from the VAE latent state for HeTVAE, HeTVAE - INT and HTVAE mTAN, and visualize the distribution of the resulting mixture. Figure 5.4 illustrates the interpolation performance of each of the models. As we can see, the interpolations produced by HTVAE mTAN have approximately constant uncertainty across time and this uncertainty level does not change even when the number of points conditioned on increases. On the other hand, both HeTVAE and STGP show variable uncertainty across time. Their uncertainty reduces in the vicinity of input observations and increases in gaps between observations. Even though the STGP model has an advantage in this experiment (the synthetic data were generated with an RBF kernel smoother and STGP uses an RBF kernel as the covariance function) the proposed model HeTVAE shows comparable interpolation performance. The HeTVAE-INT model performs slightly better than HTVAE mTAN model but it does not show variable uncertainty due to input sparsity like HeTVAE.



(a) Example 1.



(b) Example 2.

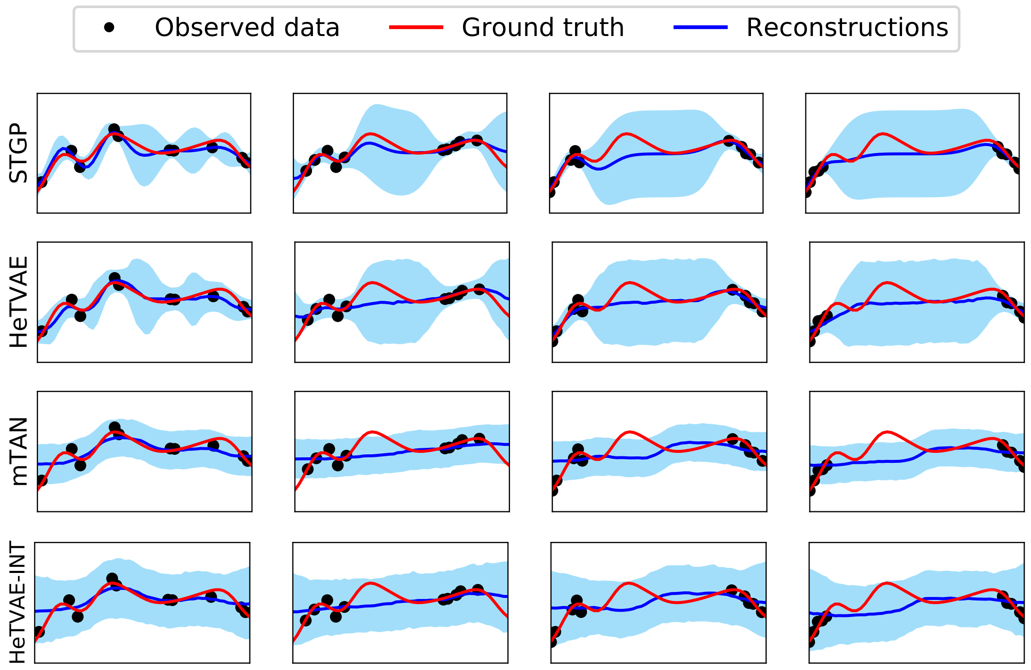
Figure 5.4: Sample interpolation results on the synthetic dataset. The 3 columns correspond to interpolation results with increasing numbers of observed points: 3, 10 and 20 respectively. The shaded region corresponds to $\pm$ one standard deviation. STGP and HeTVAE exhibit variable output uncertainty in response to input sparsity while mTAN and HeTVAE - INT do not.

5.4.3 Synthetic Data Visualizations: Inter-Observation Gap

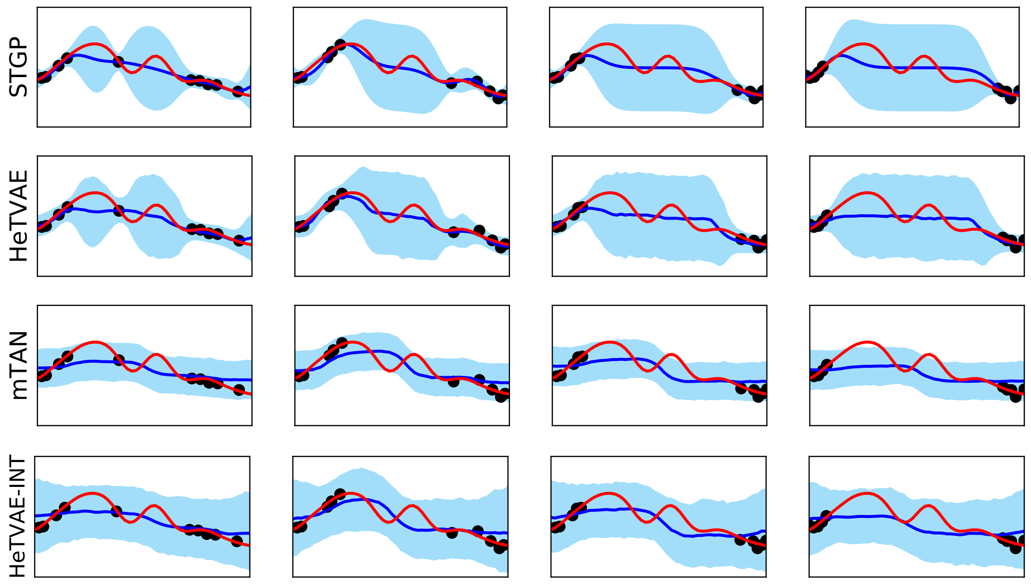
To demonstrate the effectiveness of intensity encoder (INT), we perform another experiment on the synthetic dataset where we increase the maximum gap between the observations. We follow the same training protocol as described in Section 5.2. At test time, we condition on 10 observed points with increasing maximum inter-observation gap. We vary the maximum inter-observation gap from 20% to 80% of the length of the original time series. Each model is used to infer single time point marginal distributions over values at the rest of the available time points in the test instance.

Figure 5.5 shows the interpolations with increasing maximum inter-observation gap. STGP and HeTVAE show variable uncertainty with time and the uncertainty increases with increasing maximum inter-observation gap. On the other hand, HT-VAE mTAN with homoscedastic output shows approximately constant uncertainty with time and also across different maximum inter-observation gaps. These results clearly show that HTVAE mTAN produces over-confident probabilistic interpolations over large gaps.

Furthermore, we show an ablation of the proposed model HeTVAE - INT, where we remove the intensity encoder and perform the interpolations. As we see from the figure, this leads to approximately constant uncertainty across time as well as different maximum inter-observation gaps. This shows that the HeTVAE model is not able to capture uncertainty due to input sparsity as effectively without the intensity encoder.



(a) Example 1.



(b) Example 2.

Figure 5.5: In this figure, we show example interpolations on the synthetic dataset with increasing maximum inter-observation gap. The columns correspond to an inter-observation gap of size 20%, 40%, 60% and 80% of the length of original time series. The rows correspond to STGP, HeTVAE, HTVAE mTAN and HeTVAE-INT respectively. The shaded region corresponds to the confidence region. STGP and HeTVAE exhibit variable output uncertainty while mTAN and HeTVAE-INT does not.

5.5 Conclusion

In this chapter, we have proposed the Heteroscedastic Temporal Variational Autoencoder (HeTVAE) for probabilistic interpolation of irregularly sampled time series data. HeTVAE consists of an input sparsity-aware encoder, parallel deterministic and probabilistic pathways for propagating input uncertainty to the output, and a heteroscedastic output distribution to represent variable uncertainty in the output interpolations. Furthermore, we propose an augmented training objective to combat the presence of additional local optima that arise from the use of the heteroscedastic output structure. Our results show that the proposed model significantly improves uncertainty quantification in the output interpolations as evidenced by significantly improved log likelihood scores compared to several baselines and state-of-the-art methods.

CHAPTER 6

FUSION MODELS

In this chapter, we present architectures for combining time series and text modality. Specifically, we show how we can leverage the content in text data and fuse the information they contain with irregularly sampled time series data. We build on the previously described frameworks for modeling sparse and irregularly sampled time series data. We study several methods for representing the text data, along with both early and late fusion approaches to integrating the two data modalities (Kiela and Bottou, 2014; Fiterau et al., 2017). Finally, we explore the predictive value of integrating irregularly sampled time series data and text into a unified prediction model.

We begin by presenting the proposed fusion approach. Next, we present mortality prediction experiments on the MIMIC-III data set (Johnson et al., 2016) demonstrating how the relative predictive value of clinical text and physiological data change during the first 48 hours after admission. We show that the late fusion approach can provide a significant improvement over using individual modalities in isolation.

6.1 Fusion Model Framework

In this section, we present the proposed fusion modeling framework. We begin by presenting a description of the models used for text followed by a discussion of fusion approaches. We follow the notation introduced in Section 3.1.1. Additionally, we let $\mathbf{v}_n$ represent the unstructured text data present in the clinical notes as a sequence of words.

6.1.1 Text Models

We consider several different approaches to modeling unstructured text including approaches based on bag-of-words and word embedding representations. We describe each text representation approach below.

- **TF-IDF:** Each text document is first represented using TF-IDF features computed from a bag-of-words representation. We remove stop words and select a vocabulary consisting of the top 6,000 most frequent remaining words. We apply a one hidden layer (1NN) fully connected network of size 128 on top of the TF-IDF inputs, followed by the rest of the prediction network.

- **Word Embedding (WE):** Each document is first represented as a matrix where rows are words in the document and columns are word embedding dimensions. Word embeddings are computed using a standard, pre-trained 300-dimensional GloVe model (Pennington et al., 2014). We then apply a convolutional neural network model with one 1D convolution and one pooling layer, followed by a fully-connected layer of size 128 connected to the rest of the prediction network. Stop words and words with no embeddings are removed. All documents are zero-padded to match the length of the longest document. We select the number of convolution kernels on a validation set.

- **Unweighted Sentence Embedding (USE):** Each document is first represented as a matrix where rows are sentences in the document and columns are the sentence embedding dimensions. The sentence embeddings are computed by averaging the GloVe embeddings (Pennington et al., 2014) of their constituent words. We then apply a GRU model (Chung et al., 2014) to the sequence of sentence embeddings, followed by a fully-connected layer of size 128 connected to the rest of the prediction network. We consider GRU models with between 32 and 512 hidden units and select the best on a validation set.

- **Weighted Sentence Embedding (WSE):** Each document is represented as a matrix where rows are sentences in the document and columns are are the sentence embedding dimensions. We compute the sentence embedding by weighting the GloVe word embeddings (Pennington et al., 2014) based on their unigram probability in the entire corpus as described in Arora et al. (2017). The remainder of this approach matches the unweighted case as described above.

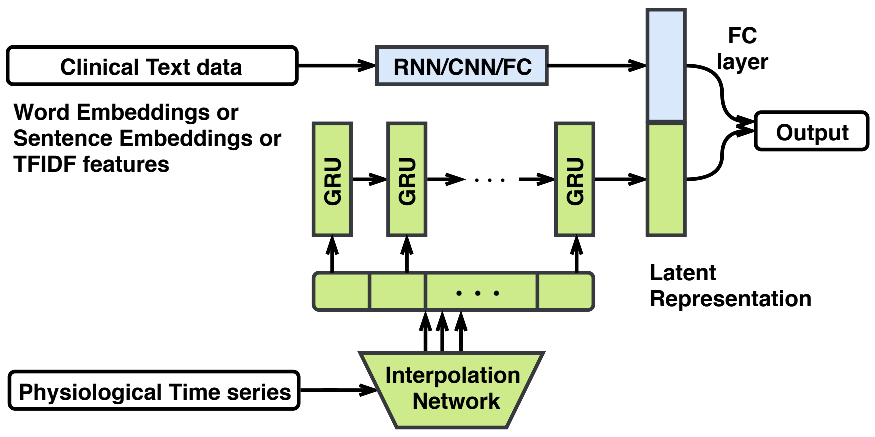
In all cases, the text representations described above are connected to the remainder of a prediction network via a 128-dimensional hidden layer. Recalling that $\mathbf{v}_n$ represents the raw, unstructured text data available as input for data case n, we can view each of the methods described above as a different approach to computing a fixed, 128-dimensional embedding $\hat{\mathbf{v}}_n = h_\phi(\mathbf{v}_n)$. To learn the parameters ϕ for each approach, we use a supervised pre-training approach. We directly connect the text embedding layer $\hat{\mathbf{v}}_n$ to the prediction target, and minimize a prediction loss. In this work, we focus on in-hospital mortality prediction and use binary cross entropy as the loss function during pre-training.

6.1.2 Fusion Approaches

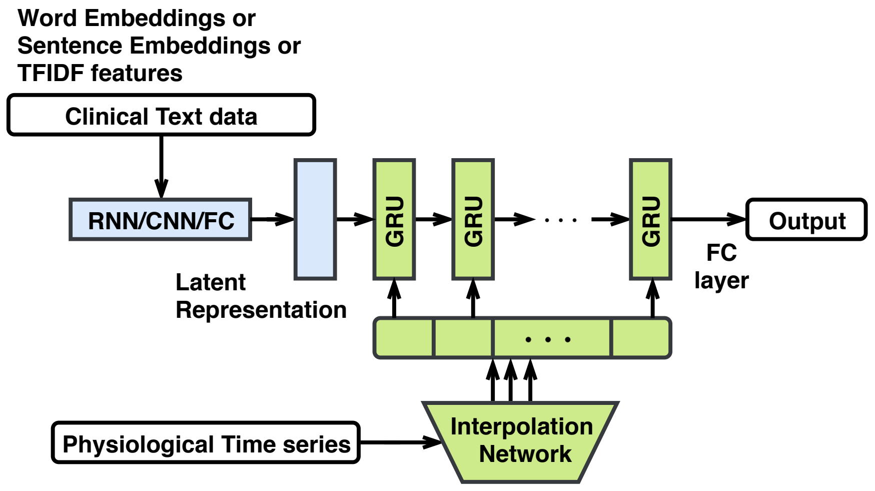
In this section, we present fusion architectures that combine the networks for irregularly sampled time series introduced in previous chapters with the embedding-based models for representing unstructured text described in Section 6.1.1. In particular, we present two fusion architectures that accept as input the representations produced by the interpolation network[1] and the text embeddings produced by the unstructured text models. Both architectures are shown in Figure 6.1 and are described below.

- **Late Fusion:** In this approach, the fusion architecture uses the GRU architecture to extract a fixed-dimensional latent representation of the physiological

[1]We refer to the architectures for irregularly sampled time series as interpolation network.



(a) Late Fusion



(b) Early Fusion

Figure 6.1: Fusion architectures (top: late fusion, bottom: early fusion) for combining time series and text information.

time series data. This representation is concatenated with the text embedding layer and the combined latent representation is connected to the prediction target using a linear layer. This architecture is shown in Figure 6.1a.

- **Early Fusion:** We also consider a deeper integration of the information contained in both physiological time series and clinical notes. In this method, our prediction network has access to the clinical text data prior to incorporating physiological time series data via a GRU layer, as shown in Figure 6.1b.

Since the output of the interpolation network is always a fully-observed fixed dimensional temporal representation, we can also replace the GRU architecture in fusion models with the recent Transformer (Vaswani et al., 2017) model. In both cases, the fusion architecture takes as input the time series interpolants $\hat{\mathbf{s}}_n = f_\theta(\mathbf{s}_n)$ (where f_θ denote the interpolation network) as well as the text embedding $\hat{\mathbf{v}}_n = h_{\phi_*}(\mathbf{v}_n)$ and outputs a prediction $\hat{y}_n = g_\omega(\hat{\mathbf{s}}_n, \hat{\mathbf{v}}_n) = g_\omega(f_\theta(\mathbf{s}_n), h_{\phi_*}(\mathbf{v}_n))$. As described above, we use supervised pre-training of all of the model parameters by training the interpolation and text embedding networks in isolation. During the fusion stage, we freeze the text embedding parameters ϕ to their optimal pre-trained values ϕ_*, and fine-tune the rest of the network parameters θ and ω.

The learning objective for the fusion framework requires specifying a loss ℓ_P for the prediction network (we use cross-entropy loss for classification). We let ℓ_I be the interpolation network loss. We also include ℓ_2 regularizers for all the network parameters. δ_F, δ_G, and δ_R are hyper-parameters that control the trade-off between the components of the objective function. The full objective is shown below.

$$\begin{aligned}\theta_*, \omega_* = \underset{\theta,\omega}{\arg\min} &\sum_{n=1}^{N} \ell_P(y_n, g_\omega(f_\theta(\mathbf{s}_n), h_{\phi_*}(\mathbf{v}_n))) \\ &+ \delta_R \sum_{n=1}^{N} \ell_I(\mathbf{s}_n, \hat{\mathbf{s}}_n) + \delta_F \|\theta\|_2^2 + \delta_G \|\omega\|_2^2 \end{aligned} \tag{6.1}$$

The interpolation network loss ℓ_I corresponds to autoencoder loss, ELBO and ELBO with augmented objective for IP-Nets, mTANs and HeTVAE respectively. Note again that we leverage a pre-trained text embedding model, thus the text embedding model parameters ϕ are fixed to their optimal pre-trained values ϕ_*. The parameters of the fusion model (as well as all other models used in this work) are learned using the Adam optimization method in TensorFlow with gradients provided via automatic differentiation.

6.2 Experiments

In this section, we present experiments and results. Our experiments focus on the relative predictive performance of text-only models, time series-only models, and fusion models for the problem of in-hospital mortality prediction. The prediction output is a single binary variable representing the occurrence of in-hospital morality more than 48 hours after admission. The time series inputs to the prediction task are sparse and irregularly sampled physiological time series. We consider making predictions using physiological time series data available between 6 and 48 hours after admission. The text inputs to the prediction task consist of text content known at the time of admission and progress notes available between 6 and 48 hours after admission. We begin by briefly describing the data set used, followed by the set of baseline and comparison models, the empirical protocols used, and finally the results.

6.2.1 Dataset

Our experiments are based on the publicly available MIMIC-III dataset (Section 2.10.1). We start with the dataset used in Section 3.2 which consists of hospital admission records with hospital admission-to-discharge length of stay more than 48 hours. From that dataset, we obtained 42,984 records for our experiments after removing newborns and hospital admission records containing no clinical notes. A hospital admission may correspond to zero or multiple ICU episodes. In this chapter, we only consider the data cases that were admitted to ICU at least once during their hospital stay. We use text data known at the time of admission such as chief complaints, past medical history and history of present illness. We take care in extracting this information from discharge summaries in order to avoid any information leak. We also extract progress notes from non-discharge reports such as respiratory, ECG, echo, radiology, and nursing reports. We use the date and time stamps on these reports to create a set of notes available between 6 and 48 hours after admission. Note that the physiological data and clinical notes are aligned in a conservative manner. If a clinical note has both a date and time associated with it, we assume that information was available at the specified time. For notes that have dates but not times available, we assume that information was available at the end of the indicated day. This happens for some ECG and Echo reports in the data set.

6.2.2 Baseline Models

We compare fusion models with a number of baseline approaches that model the physiological time series or the clinical text data individually. We use the models introduced in Chapter 3, 4 and 5 as baselines for time series. We use the pre-trained text-only models described in Section 6.1.1 to provide text-only baselines.

6.2.3 Empirical Protocols

Each unique hospital admission-to-discharge episode for a patient is assigned a unique ID in the MIMIC-III data set. The data in each episode are treated as being independent. In the train-test split, we divide the data based on the hospital admission ID (i.e. 80% (27510) of IDs are used for training and 20% (8597) are used for testing). We set aside another 20% (6877 data cases) from the training set to use as a validation set. Since we only use data from within individual hospital-to-discharge episodes, the data cases we construct are temporally non-overlapping. Again, this is consistent with how the MIMIC-III data set has been used in past research (Che et al., 2018a).

All models are trained to minimize the cross entropy loss. For all of the models, we independently tune the following hyper-parameters : number of hidden layers, hidden units, convolutional filters, filter-size, learning rate, dropout rates and regularization parameters on the validation set. For TF-IDF-based models, we also tune the number of TF-IDF features. The neural network models are learned using the Adam optimizer. Early stopping is used on the validation set. The final outputs of

Table 6.1: Text-only baselines.

Text Model	**Hours from Admission**	**AUC**
WE / CNN	0	0.6974
WSE / RNN	0	0.7364
USE / RNN	0	0.7473
TF-IDF / 1-NN	0	0.7965
TF-IDF / 1-NN	6	0.8035
TF-IDF / 1-NN	12	0.8173
TF-IDF / 1-NN	18	0.8263
TF-IDF / 1-NN	24	0.8410
TF-IDF / 1-NN	30	0.8454
TF-IDF / 1-NN	36	0.8503
TF-IDF / 1-NN	42	0.8554
TF-IDF / 1-NN	48	0.8627

the hidden layers are used in a logistic layer that predicts the class. We evaluate all the models using an estimate of generalization performance computed on the test set. We report the performance on the test set in terms of the area under the ROC curve (AUC score).

6.3 Results

In this section, we present the results of the mortality prediction experiments. We begin with text-only and time series-only baseline results, followed by fusion model results.

6.3.1 Text-Only Baselines

Table 6.1 shows the classification performance for the text-only models described in Section 6.1.1. We evaluate all models in the case of text data available at the time of admission. These results show that the TF-IDF-based model performs significantly better than the embedding methods. This may be due to the fact that health-specific concepts are not well represented in the standard Glove embeddings used. Another

Table 6.2: Performance Comparison of Time Series-Only baselines in terms of Area under ROC Curve (AUC Score).

Hours	**IP-Net**	**mTAN**	**HeTVAE**
6	0.7106	0.7255	0.7433
12	0.7380	0.7678	0.7762
18	0.7645	0.7880	0.7981
24	0.7759	0.7921	0.8137
30	0.7902	0.8085	0.8185
36	0.7958	0.8176	0.8319
42	0.8023	0.8221	0.8413
48	0.8245	0.8366	0.8467

possible reason could be the use of abbreviated terms, which are quite common in clinical notes. For this reason, we only consider the TF-IDF model when making predictions based on all the progress notes available after admission. We can see that prediction performance using the TF-IDF model increases significantly as more text data become available over time.

6.3.2 Time Series-Only Baseline

Table 6.2 assesses the predictive performance of the time series-only baseline. We consider all the three models introduced in this thesis – IP-Nets (Chapter 3), mTAN (Chapter 4) and HeTVAE (Chapter 5). As expected, predictive performance increases as the amount of observed physiological data increases. HeTVAE model achieves the best performance consistently across all the settings. We can also see that mTAN model always provides improvement over the IP-Net model. Comparing to the results in Table 6.1, we can see that the predictive value of the clinical text available at the time of admission exceeds that of the available physiological data until near the end of the 42 hour period following admission in case of IP-Nets, while in case of HeTVAE that period reduces to 18 hours. We note that the results reported here are different from that in their respective chapters because of additional data filtering required for

Table 6.3: Performance of *Early* and *Late* Fusion approach with notes available at admission time and increasing amount of physiological signals in terms of AUC score.

Hours	**IP-Net**		**mTAN**		**HeTVAE**	
	Early Fusion	**Late Fusion**	**Early Fusion**	**Late Fusion**	**Early Fusion**	**Late Fusion**
6	0.7850	0.8027	0.8028	0.8208	0.8019	0.8234
12	0.7916	0.8161	0.8145	0.8338	0.8171	0.8407
18	0.8046	0.8138	0.8251	0.8436	0.8243	0.8318
24	0.8126	0.8256	0.8262	0.8483	0.8327	0.8454
30	0.8284	0.8324	0.8358	0.8554	0.8442	0.8525
36	0.8291	0.8320	0.8390	0.8563	0.8495	0.8643
42	0.8380	0.8376	0.8469	0.8651	0.8523	0.8697
48	0.8427	0.8453	0.8521	0.8714	0.8612	0.8741

removing hospital admission records containing no clinical notes and neonates data. The next set of experiment aims to assess whether these two modalities can result in improved performance when fused.

6.3.3 Fusion Approaches with Admission Notes

Based on the observed success of the TF-IDF-based model in the text-only baseline experiments, we examine the performance of fusion approaches using the TF-IDF-based model to embed the clinical text data. We begin by assessing the performance of a fusion approach that only has access to the clinical text data available at the time of admission, but increasing amounts of physiological time series data up to the end of the 48 hour period following admission. Table 6.3 shows the classification performance of the early and late fusion models under this experimental scenario. Figure 6.2 shows the performance of early and late fusion relative to the time series-only and text-only baselines. We see that the late fusion approach achieves better performance than the early fusion approach while both significantly improve on the time series-only baseline. However, we see that all three models that incorporate physiological data increase in predictive performance as the amount of physiological

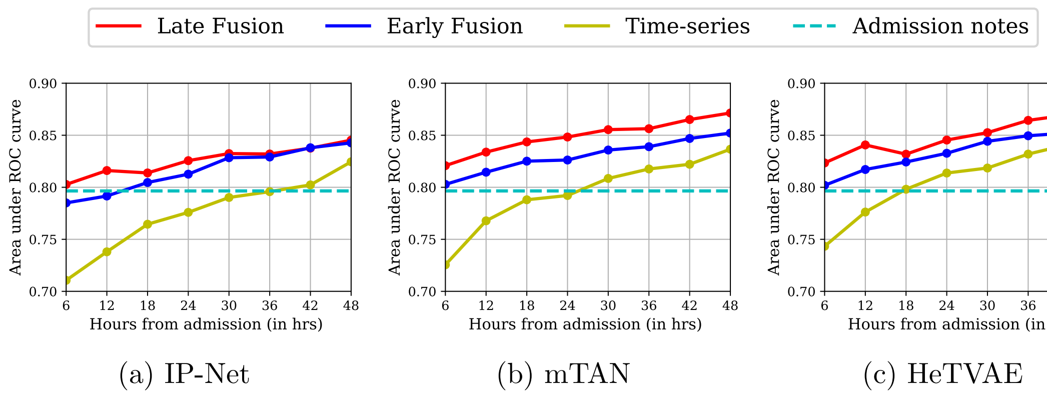


Figure 6.2: Performance comparison on the mortality prediction task with text available at admission only but increasing amounts of physiological time series.

data increases. Comparing performance across different models, we see that fusion approaches with attention-based models mTAN and HeTVAE achieve similar results while both performing better than that with IP-Net. Further, we see that the performance gap between fusion and time series-only models decreases over time, which indicates that the advantage provided by the initial fusion with text data available at time of admission decreases over time as that information becomes less relevant. Finally, we note that the late fusion model outperforms the text-only baseline at all times while the early fusion model with IP-Net initially exhibits lower performance than the text-only TF-IDF baseline, but goes on to match and then outperforms the text-only baseline.

6.3.4 Fusion Approaches with Progress Notes

Next, we consider the fusion process as increasing volumes of text data become available through time, as well as increasing volumes of physiological data. For this experiment, we consider only the TF-IDF-based text embedding model and limit the discussion to the late fusion approach as these models have achieved the best performance in our experiments to date. We consider incorporating text data known at admission at time 0, followed by the text of all notes known between 6 and 48 hours

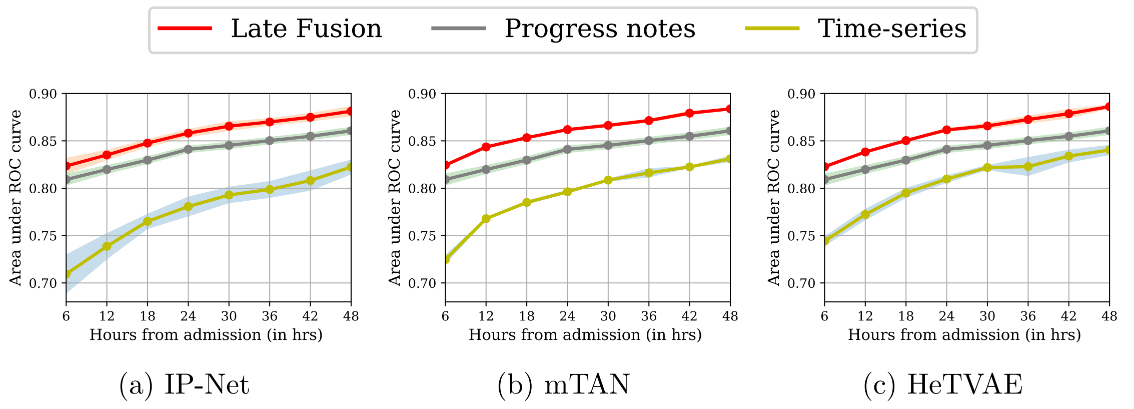


Figure 6.3: Performance comparison on the mortality prediction task with increasing amount of physiological time series and progress notes using 5 randomly generated train-validation-test splits. Bands correspond to 95% confidence interval around the mean.

following admission . The results of this experiment are shown in Figure 6.3 and Table 6.4. We can see the predictive performance of progress notes is significantly better than physiological time series data. We can also see that the performance of the fused model always exceeds that of the corresponding text-only baseline at a given point in time, with performance generally rising as additional physiological/text data become available. Table 6.4 shows that the trend is the same in case of all three interpolation network architectures. Furthermore, we see that with the availability of additional text data through time, the performance of all three late fusion models employing different interpolation network architectures is very similar, which indicates that the relevance of the time series data has reduced. By comparing with Figure 6.2, we can see that the addition of progress notes past admission results in a final fused model that significantly outperforms models that only have access to text data from the time of admission.

To verify the statistical significance of the gap between the late fusion approach and the single-modality approaches, we perform a five-fold random resampling assessment of the test AUC by randomly generating 5 train-validation-test splits. We

Table 6.4: Performance of *Late* Fusion approach with increasing amount of physiological signals and progress notes in terms of AUC score.

Hours	**IP-Net**	**mTAN**	**HeTVAE**
6	0.8235 ± 0.0099	0.8244 ± 0.0007	0.8227 ± 0.0019
12	0.8351 ± 0.0064	0.8435 ± 0.0018	0.8383 ± 0.0013
18	0.8477 ± 0.0049	0.8534 ± 0.0013	0.8503 ± 0.0022
24	0.8581 ± 0.0050	0.8619 ± 0.0021	0.8616 ± 0.0021
30	0.8655 ± 0.0058	0.8664 ± 0.0027	0.8658 ± 0.0037
36	0.8699 ± 0.0046	0.8714 ± 0.0025	0.8725 ± 0.0049
42	0.8749 ± 0.0052	0.8793 ± 0.0030	0.8787 ± 0.0058
48	0.8813 ± 0.0063	0.8838 ± 0.0010	0.8862 ± 0.0027

run the complete hyper-parameter selection and learning pipeline for each approach on each of the five data sets. Figure 6.3 shows the mean with 95% confidence interval for all the baselines.

6.4 Conclusion

In this chapter, we have developed methods for investigating the relative predictive value of the content of clinical notes and physiological time series data in ICU EHRs. We have considered models based on clinical text only, models based on physiological time-series only, and a novel fusion approach that combines both modalities. Our experiments have focused on using this methodology to assess the relative predictive value of clinical text and physiological data as a function of time since admission. We have focused on the task of predicting in-hospital mortality events which take place more than 48 hours after admission. We have performed experiments with all three frameworks for irregularly sampled time series introduced in this thesis. Our results show that the relative value of information in text records known at the time of admission decreases over time as more physiological data are observed. However, incorporating newly available text data can significantly boost predictive performance. Finally, our results strongly support the conclusion that fusing both data modalities results in the best overall predictive performance. The use of the more powerful mTAN and HeTVAE frameworks improves the performance of the time series-only and late fusion approaches when considering the notes available only at the time of admission. However, with the availability of additional text data through time, we see no advantage with the mTAN and HeTVAE framework indicating that the relevance of the time series data has reduced.

CHAPTER 7
CONCLUSIONS AND FUTURE DIRECTIONS

7.1 Summary

In this thesis, we have focused on the development of deep learning models for the problems of supervised and unsupervised learning from irregularly sampled time series data. We started by presenting a range of principles for learning from irregularly sampled time series data. We have provided a categorization of current approaches for this setting into groups based on the fundamental modeling primitives used to accommodate irregular sampling. These modeling primitives include temporal discretization, interpolation, recurrence, attention and structural invariance. We have discussed the relationship between the three fundamental representations of multivariate irregularly sampled time series (series-based, vector-based, and set-based) that, while equivalent, motivate the use of different modeling primitives and result in different specific models and methods. We have also defined a range of inference tasks that can be performed using irregularly sampled time series (detection, prediction, filtering, smoothing, interpolation and forecasting). Finally, we have described a large number of specific models and methods with respect to the model primitives they build on, the tasks they aim to solve, and their relative strengths and weaknesses.

Next, we present three methodological advances for unsupervised and supervised learning with irregularly sampled time series. These approaches are based on the interpolation modeling primitive. First we present Interpolation-prediction Networks, a computationally efficient architecture based on RBF-kernel interpolation layers resulting in state-of-the-art results on classification and regression tasks at the time of publication. Next, we show how the use of fixed RBF kernel functions can be relaxed through the use of a novel attention-based continuous-time interpolation framework: Multi-Time Attention Networks. This approach improves on interpolation-prediction networks and provides interpolation and classification performance better than the current state-of-the-art methods, while providing significantly reduced training times. However, these approaches are not able to reflect the input uncertainty due to sparsity and missingness in the output interpolations. To address this, we present HeTVAE, novel deep learning framework for probabilistic interpolation. HeTVAE significantly improves uncertainty quantification in the output interpolations compared to the current state-of-the-art methods. Furthermore, we show that this framework is also able to improve classification performance over the current state-of-the-art methods. Finally, we use these building blocks to present fusion architectures for integrated modeling of irregularly sampled time series and text data.

Overall, we have shown that interpolation-based models introduced in this thesis outperform current methods based on GPs, RNNs, and ODEs on several tasks including whole time series classification, interpolation, smoothing, and prediction. These models have also been shown to be substantially faster to train than Gaussian process regression and ODE-based models.

7.2 Limitations

We note that the proposed models in this thesis are focused on learning from time series generated from underlying continuously varying, real-valued univariate or multivariate functions. While these models can be used with binary and ordinal time series by treating them as real-valued, these models do not currently produce discrete outputs. This is an interesting direction for future work as approaches for dealing with irregularly sampled discrete data are also significantly lacking.

Another limitation is that the approaches discussed in this thesis are not applicable to the continuous-time data generating process that do not correspond to sampling continuously varying latent functions. Examples of such processes include various types of temporal point processes and marked temporal point processes. In particular, a marked temporal point process generates a sequence of time-value pairs (t, x) that is structurally identical to data generated by an irregular sampling of a continuously varying function; however, these two categories of generative processes are quite distinct. Point processes have time-to-event generative semantics and there is no notion of a value that is defined during inter-event intervals. As a result, temporal point process models focus on modeling the distribution of inter-event times and the corresponding modeling approaches are largely distinct from those considered here.

We also note that while the HeTVAE model can produce a probability distribution over an arbitrary collection of output time points, it is currently restricted to producing marginal distributions in the final layer. As a result, sampling from the model does not necessarily produce smooth trajectories as would be the case with GPR-based models. Augmenting the HetVAE model to account for residual correlations in the output layer is also an interesting direction for future work. Another possible direction would be to use more expressive posterior distribution on the temporal latent state to reflect correlations in time as compared to the fully factorized posterior used in HeTVAE.

Finally, we note that we have not evaluated the proposed models or baseline approaches from the perspective of algorithmic bias, an important consideration when learning models from observational healthcare data and another important direction for future work.

7.3 Future Directions

While in this thesis, we have focused on VAE-based encoder-decoder architecture, the proposed mTAN and UnTAN modules can also be used in GAN-based frameworks. The frameworks presented in this thesis can readily incorporate most techniques developed for generative models including recent advances in GANs and VAEs. Furthermore, recent advances in the development of improved attention techniques can also be easily incorporated in the mTAN and HeTVAE frameworks.

One of the goal of this thesis has been to develop efficient methods for learning from irregularly sampled time series. The proposed approaches are already substantially faster than other competitive methods based on Gaussian processes and ODEs. The standard scaled dot product attention used in the approaches introduced here has the run time complexity of $\mathcal{O}(TK)$ for an input time searies of length T and number of reference points K. The proposed approaches can become slow for longer length time series. We can further improve the run time complexity from $\mathcal{O}(TK)$ to $\mathcal{O}(wK)$ by using local attention (Beltagy et al., 2020) with the receptive field w. Investigating the effectiveness of this approach for irregularly sampled time series is an interesting direction for future work.

In this thesis, we mainly focused on irregularly sampled time series data but the methods introduced here, particularly Multi-Time Attention Network and HeTVAE, could easily be employed for modeling irregularly sampled spatial data. These approaches are also applicable for learning from regularly spaced data. The techniques introduced in Chapter 5, specifically heteroscedastic output layers with augmented learning objective, could also be useful for probabilistic imputation of other data modalities such as images.

In terms of the fusion models, our experiments showed that the TF-IDF based approach achieve better performance than the word embedding and sentence embedding based approaches. With the recent advances in NLP, BERT and Transformer based models pre-trained specifically on medical notes can be utilized to learn representation of the admission and progress notes. Investigating these approaches for text data and incorporating in the fusion approaches is another interesting direction for future work.

Based on the survey of prior methods for irregularly sampled time series, it is apparent that recent research in the machine learning community has focused most heavily on supervised problems, followed by interpolation and smoothing. There has been significantly less attention on the forecasting task. Indeed, most of the prior methods and the proposed methods in this thesis have not been applied to forecasting tasks. The development of methods for learning to forecast accurately from irregularly sampled inputs thus appears to be a significantly more open problem than the development of methods for the other tasks described. Investigating and extending the approaches introduced in this thesis for the forecasting task is an interesting future direction.

In terms of directions for future work considering the different modeling primitives, the attention and structural invariance modeling primitives have promising properties, but have been much less explored than other primitives including recurrence. The further development of attention and structural invariance-based approaches may lead to improved accuracy-speed trade-offs by leveraging the enhanced parallel computation that these primitives enable. Within the area of recurrent models, differential equation-based models for irregular time series have advanced the state-of-the-art over discrete RNNs. Neural CDEs appear to have interesting advantages over ODE-based models as noted above and are also an interesting area for further exploration. Recent advances in multi-task GP-based methods also provide an important future research direction for probabilistic time series interpolation.

BIBLIOGRAPHY

N. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician*, 46(3):175–185, Aug. 1992.

S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*, 2017.

M. T. Bahadori and Z. C. Lipton. Temporal-clustering invariance in irregular healthcare time series. *CoRR*, abs/1904.12206, 2019. Presented at the ACM Conference on Health, Inference, and Learning, Workshop Track, 2020.

I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou. Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 65–74, 2017.

I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

F. M. Bianchi, L. Livi, K. Mikalsen, M. Kampffmeyer, and R. Jenssen. Learning representations of multivariate time series with missing data. *Pattern Recognition*, 96, 2019.

M. Binkowski, G. Marti, and P. Donnat. Autoregressive convolutional neural networks for asynchronous time series. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 580–589, StockholmsmÃ€ssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/binkowski18a.html`.

E. V. Bonilla, K. M. Chai, and C. Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, pages 153–160, 2008.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li. BRITS: Bidirectional Recurrent Imputation for Time Series. In *Advances in Neural Information Processing Systems*, pages 6775–6785. 2018.

Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1), 2018a.

Z. Che, S. Purushotham, G. Li, B. Jiang, and Y. Liu. Hierarchical deep generative models for multi-rate multivariate time series. In *Proceedings of the 35th International Conference on Machine Learning*, pages 784–793, 2018b.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583. 2018.

E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun. Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. In *Proceedings of the 1st Machine Learning for Healthcare Conference*, pages 301–318, 2016a.

E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016b.

J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. Presented at the Deep Learning workshop at NIPS 2014.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

M. Cuturi. Fast global alignment kernels. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, page 929–936, 2011.

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240, 2006.

E. De Brouwer, J. Simm, A. Arany, and Y. Moreau. GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series. In *Advances in Neural Information Processing Systems*, pages 7379–7390. 2019.

H. Drucker. Improving regressors using boosting techniques. In *Proceedings of the 14th International Conference on Machine Learning*, pages 107–115, 1997.

A. Eckner. A framework for the analysis of unevenly spaced time series data. Technical report, 2014.

M. Fiterau, S. Bhooshan, J. A. Fries, C. Bournhonesque, J. L. Hicks, E. Halilaj, C. Ré, and S. L. Delp. ShortFuse: Biomedical Time Series Representations in the Presence of Structured Information. In *Proceedings of the 2nd Machine Learning for Healthcare Conference*, pages 59–74, 2017.

V. Fortuin, D. Baranchuk, G. Raetsch, and S. Mandt. GP-VAE: Deep Probabilistic Time Series Imputation. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, pages 1651–1661, 2020.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119–139, 1997.

J. Futoma, S. Hariharan, and K. A. Heller. Learning to detect sepsis with a multitask gaussian process RNN classifier. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1174–1182, 2017.

M. Ghassemi, M. A. Pimentel, T. Brennan, D. A. Clifton, P. Szolovits, T. Naumann, and M. Feng. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in ICU with sparse, heterogeneous clinical data. In *Proceedings of the National Conference on Artificial Intelligence*, 2015.

Z. Guo, Y. Wan, and H. Ye. A data imputation method for multivariate time series based on generative adversarial network. *Neurocomputing*, 2019.

Han-Gyu Kim, Gil-Jin Jang, Ho-Jin Choi, Minho Kim, Young-Won Kim, and Jaehun Choi. Recurrent neural networks with missing information imputation for medical examination data prediction. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 317–323, 2017.

H. H. Harman. *Modern Factor Analysis.* The University of Chicago Press, 1976.

H. Harutyunyan, H. Khachatrian, D. C. Kale, and A. Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6, 2019.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. 2001.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

M. Horn, M. Moor, C. Bock, B. Rieck, and K. Borgwardt. Set functions for time series. In *Proceedings of the 25th International Conference on Machine Learning*, 2020.

D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied logistic regression.* John Wiley & Sons, 2013.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, and M. Gams. An agent-based approach to care in independent living. In *Proceedings of the 1st International Joint Conference on Ambient Intelligence*, pages 177–186, 2010.

P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, 2020.

D. Kiela and L. Bottou. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 36–45, 2014.

Y. J. Kim and M. Chi. Temporal belief memory: Imputing missing data during rnn training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2326–2332, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583. 2015.

J. L. Kling and D. Bessler. A comparison of multivariate forecasting procedures for economic time series. *International Journal of Forecasting*, 1:5–24, 1985.

M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal*, 37:233–243, 1991.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, page 282–289, 2001.

T. A. Lasko. Efficient inference of gaussian-process-modulated renewal processes with application to medical event data. In *Uncertainty in artificial intelligence: proceedings of the... conference. Conference on Uncertainty in Artificial Intelligence*, volume 2014, page 469. NIH Public Access, 2014.

Q. Li and Y. Xu. VS-GRU: A Variable Sensitive Gated Recurrent Neural Network for Multivariate Time Series with Massive Missing Values. *Applied Sciences*, 2019.

S. C.-X. Li and B. Marlin. Learning from irregularly-sampled time series: A missing data perspective. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5937–5946, 2020.

S. C.-X. Li and B. M. Marlin. Classification of sparse and irregularly sampled time series with mixtures of expected Gaussian kernels and random features. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pages 484–493, 2015.

S. C.-X. Li and B. M. Marlin. A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In *Advances In Neural Information Processing Systems*, pages 1804–1812, 2016.

S. C.-X. Li, B. Jiang, and B. Marlin. Learning from Incomplete Data with Generative Adversarial Networks. In *International Conference on Learning Representations*, 2019.

Z. C. Lipton, D. Kale, and R. Wetzel. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Proceedings of the 1st Machine Learning for Healthcare Conference*, pages 253–270, 2016.

R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

Z. Liu and M. Hauskrecht. Learning Adaptive Forecasting Models from Irregularly Sampled Multivariate Clinical Data. pages 1273–1279, 2016.

S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

L. H. Loomis. *Introduction to abstract harmonic analysis*. Dover books on mathematics. 2011.

Z. Lu, T. K. Leen, Y. Huang, and D. Erdogmus. A Reproducing Kernel Hilbert Space Framework for Pairwise Time Series Distances. In *Proceedings of the 25th International Conference on Machine Learning*, pages 624–631, 2008.

Y. Luo, X. Cai, Y. Zhang, J. Xu, and X. Yuan. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1596–1607, 2018.

Y. Luo, Y. Zhang, X. Cai, and X. Yuan. E2GAN: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3094–3100, 2019.

P. Manimaran, J. C. Parikh, P. K. Panigrahi, S. Basu, C. M. Kishtawal, and M. B. Porecha. Modelling financial time series. In *Econophysics of Stock and other Markets*, pages 183–191. 2006.

B. M. Marlin, D. C. Kale, R. G. Khemani, and R. C. Wetzel. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 389–398, 2012.

P.-A. Mattei and J. Frellsen. MIWAE: Deep Generative Modelling and Imputation of Incomplete Data Sets. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4413–4423, 2019.

G. Mclachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*. Dekker, 1988.

M. J. Menne, C. N. Williams, Jr., and R. S. Vose. Long-term daily and monthly climate records from stations across the contiguous united states (u.s.historical climatology network) (ndp-019). 2016. doi: 10.3334/CDIAC/CLI.NDP019.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, 209:415–446, 1909.

M. C. Mozer, D. Kazakov, and R. V. Lindsey. Discrete Event, Continuous Time RNNs. *CoRR*, abs/1710.04110, 2017.

M. Mudelsee. TAUEST: A computer program for estimating persistence in unevenly spaced weather/climate time series. *Computers & Geosciences*, 28:69–72, 2002.

D. Neil, M. Pfeiffer, and S.-C. Liu. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. In *Advances in Neural Information Processing Systems*, pages 3882–3890. 2016.

J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

T. Pham, T. Tran, D. Phung, and S. Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of Biomedical Informatics*, 69:218–229, 2017.

T. Pollard, A. Johnson, J. Raffa, L. Celi, R. Mark, and O. Badawi. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific Data*, 5, 2018.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in Speech Recognition*, page 267–296. 1990.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. 2006.

K. Rehfeld, N. Marwan, J. Heitzig, and J. Kurths. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18:389–404, 2011.

M. Reyna, C. Josef, R. Jeter, S. Shashikumar, M. B. Westover, S. Nemati, and G. Clifford. Early prediction of sepsis from clinical data: The physionet/computing in cardiology challenge 2019. *Critical Care Medicine*, 48, 2019.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pages 5320–5330. 2019.

T. Ruf. The lomb-scargle periodogram in biological rhythm research: analysis of incomplete and unequally spaced time-series. *Biological Rhythm Research*, 30(2): 178–201, 1999.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. 1986.

J. D. Scargle. Studies in astronomical time series analysis. ii-statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, 263:835–853, 1982.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computing*, 10:1299–1319, 1998.

M. Schulz and K. Stattegger. Spectrum: Spectral analysis of unevenly spaced paleoclimatic time series. *Computers & Geosciences*, 23:929–945, 1997.

H. Shimodaira, K.-i. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in Neural Information Processing Systems*, pages 921–928, 2001.

S. N. Shukla and B. Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2019.

S. N. Shukla and B. M. Marlin. Integrating Physiological Time Series and Clinical Notes with Deep Learning for Improved ICU Mortality Prediction. *CoRR*, abs/2003.11059, 2020a. Presented at the ACM Conference on Health, Inference, and Learning, Workshop Track, 2020.

S. N. Shukla and B. M. Marlin. A Survey on Principles, Models and Methods for Learning from Irregularly Sampled Time Series: From Discretization to Attention and Invariance. *CoRR*, abs/2012.00168, 2020b. Presented at ML Retrospectives, Surveys & Meta-Analyses (ML-RSA) Workshop, NeurIPS 2020.

S. N. Shukla and B. M. Marlin. Multi-time attention networks for irregularly sampled time series. 2021a. International Conference on Learning Representations.

S. N. Shukla and B. M. Marlin. Heteroscedastic Temporal Variational Autoencoder for Irregularly Sampled Time Series. *CoRR*, abs/2107.11350, 2021b.

I. Silva, G. Moody, D. Scott, L. Celi, and R. Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. *Computing in cardiology*, 39:245–248, 2012.

M. Śmieja, L. u. Struski, J. Tabor, B. Zieliński, and P. a. Spurek. Processing of missing data by neural networks. In *Advances in Neural Information Processing Systems*, pages 2719–2729. 2018.

A. J. Smola and B. Schölkopf. A Tutorial on Support Vector Regression. *Statistics and Computing*, 14:199–222, 2004.

H. Soleimani, J. Hensman, and S. Saria. Scalable Joint Models for Reliable Uncertainty-Aware Event Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:1948–1963, 2018.

H. Song, D. Rajan, J. Thiagarajan, and A. Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4091–4098, 2018.

Q. Tan, M. Ye, B. Yang, S. Liu, A. J. Ma, T. C.-F. Yip, G. L.-H. Wong, and P. Yuen. DATA-GRU: Dual-Attention Time-Aware Gated Recurrent Unit for Irregular Multivariate Time Series. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 930–937, 2020.

A. Trask, D. Gilmore, and M. Russell. Modeling Order in Neural Word Embeddings at Scale. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2266–2275, 2015.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems*, page 6000–6010, 2017.

G. Wahba. Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In *Advances in Kernel Methods: Support Vector Learning*, page 69–88. 1999.

D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan. Self-attention with functional time representation learning. In *Advances in Neural Information Processing Systems*, pages 15915–15925. 2019.

Q. Ye, W. Szeto, and S. C. Wong. Short-term traffic speed forecasting based on data recorded at irregular intervals. *IEEE Transactions on Intelligent Transportation Systems*, 13:1727–1737, 2010.

J. Yoon, J. Jordon, and M. van der Schaar. GAIN: Missing data imputation using generative adversarial nets. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5689–5698, 2018a.

J. Yoon, W. R. Zame, and M. van der Schaar. Deep Sensing: Active Sensing using Multi-directional Recurrent Neural Networks. In *International Conference on Learning Representations*, 2018b.

J. Yoon, W. R. Zame, and M. van der Schaar. Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks. *IEEE Transactions on Biomedical Engineering*, 66:1477–1490, 2019.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401. 2017.

Y. Zhang, X. Yang, J. Ivy, and M. Chi. ATTAIN: Attention-based Time-Aware LSTM Networks for Disease Progression Modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4369–4375, 2019.