Learning from Irregularly-Sampled Time Series: A Missing Data Perspective

Steven Cheng-Xian Li [1]   **Benjamin M. Marlin** [1]

Abstract

Irregularly-sampled time series occur in many domains including healthcare. They can be challenging to model because they do not naturally yield a fixed-dimensional representation as required by many standard machine learning models. In this paper, we consider irregular sampling from the perspective of missing data. We model observed irregularly-sampled time series data as a sequence of index-value pairs sampled from a continuous but unobserved function. We introduce an encoder-decoder framework for learning from such generic indexed sequences. We propose learning methods for this framework based on variational autoencoders and generative adversarial networks. For continuous irregularly-sampled time series, we introduce continuous convolutional layers that can efficiently interface with existing neural network architectures. Experiments show that our models are able to achieve competitive or better classification results on irregularly-sampled multivariate time series compared to recent RNN models while offering significantly faster training times.

1. Introduction

Irregularly-sampled time series are characterized by non-uniform time intervals between successive measurements. Such data naturally occur in many real world domains. For example, in clinical data, an individual patient's state of health may be recorded only at irregular time intervals with different subsets of variables observed at different times. Further, different individuals typically have different numbers of observations for different subsets of variables observed at different time points, including after aligning to events like time of admission or disease onset.

These characteristics of irregularly-sampled time series data

create multiple challenges for classical machine learning models and algorithms that require data to be defined with respect to a fixed dimensional feature space. However, there has been significant recent progress on this problem. For example, the GRU-D model was proposed as a direct extension of discrete time RNNs to the case of continuous time observations (Che et al., 2018). The model uses exponential decay dynamics applied to either visible or latent states. Rubanova et al. (2019) proposed latent ordinary differential equation (ODE) models as a more natural way to model continuous dynamics. Latent ODEs extend the neural ODE model (Chen et al., 2018), which enables modeling of complex ODEs using neural networks. However, many of these models can be slow to learn due to their sequential nature.

The focus of this paper is on learning from a collection of irregularly-sampled time series that are observed over a fixed time span. The specific tasks we want to accomplish are: i) learning the distribution of the latent temporal process, ii) given a time series, inferring the distribution of the corresponding latent process, and iii) classification of time series. If we view each time series as observations sampled from a complete latent process defined over a time span $[0, T]$, this is essentially a missing data problem as we only have information about the latent process at a subset of points in time within $[0, T]$.

Learning complex distributions in the presence of missing data is a problem that has received substantial recent attention. For example, models have recently been proposed based on variational autoencoders (VAEs) (Kingma & Welling, 2014) such as partial VAEs (Ma et al., 2018; 2019) and MIWAE (Mattei & Frellsen, 2019). Implicit models based on generative adversarial networks (GANs) (Goodfellow et al., 2014) have also been recently proposed such as MisGAN (Li et al., 2019). However, these models only work for problems with finite dimensional data such as recommendation systems or image modeling. Neural processes (Garnelo et al., 2018a;b) can be seen as an extension of partial VAEs for the continuous space that model distributions over functions.

The main contribution of this paper is the development of a scalable framework for learning distributions from irregularly-sampled time series. We transform modeling such time series data into a general missing data problem

---

[1]University of Massachusetts Amherst. Correspondence to: Steven Cheng-Xian Li <cxl@cs.umass.edu>.

and introduce an encoder-decoder framework that unifies a number of previous approaches to modeling incomplete data based on variational autoencoders. In addition, we propose a GAN-based model for training this framework that we show outperforms the recently proposed MisGAN model. We then introduce continuous convolutional layers for handling irregularly-sampled time series to efficiently interface with existing neural network architectures. Experiments show that our framework is able to achieve competitive or better classification results on irregularly-sampled multivariate time series classification tasks compared to recent time series models such as Latent ODE, while can be trained faster by an order of magnitude.

2. Index Representation for Incomplete Data

Suppose we have data defined over an index set $\mathcal{I}$. We can represent a complete data case as a function $f : \mathcal{I} \to \mathbb{R}$ such that the value of the element associated with an index $t \in \mathcal{I}$ is $f(t)$. We use $\mathbb{R}^{\mathcal{I}}$ to denote the space of complete data. For example, for images of size $h \times w$, an element of the index set $t \in \mathcal{I}$ corresponds to the coordinates of a pixel and $f(t)$ is the corresponding pixel value. The index set $\mathcal{I}$ in this case is the collection of all possible coordinates, $\{1, \dots, h\} \times \{1, \dots, w\}$. For time series defined within an interval $[0, T]$, an index is a timestamp of an observation and the index set $\mathcal{I}$ is the continuous interval $[0, T]$.

In the incomplete data setting such as time series, we do not observe the entire f. Instead we have access to a set of values $\mathbf{x}$ of f associated with a set of indices $\mathbf{t}$ that is a subset of $\mathcal{I}$. Following Little & Rubin (2014), the generative process for an incomplete data case $(\mathbf{x}, \mathbf{t})$ in a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^n$ can be decomposed into three steps: i) sampling a complete data f from a distribution $p_{\mathcal{F}}(f)$ over $\mathbb{R}^{\mathcal{I}}$, ii) sampling a set of indices $\mathbf{t} = [t_i]_{i=1}^{|\mathbf{t}|}$ from a distribution $p_{\mathcal{I}}(\mathbf{t}|f)$ over the power set $2^{\mathcal{I}}$ conditioned on the sampled f, and iii) retaining the values of f at the sampled indices $\mathbf{t}$ to form a set of corresponding observed values $\mathbf{x} = [f(t_i)]_{i=1}^{|\mathbf{t}|}$.

We note that this representation of incomplete data is *permutation invariant*, that is, the incomplete data $(\mathbf{x}, \mathbf{t})$ is equivalent to $([x_{\pi(i)}]_{i=1}^{|\mathbf{t}|}, [t_{\pi(i)}]_{i=1}^{|\mathbf{t}|})$ for any permutation π of $\{1, \dots, |\mathbf{t}|\}$. We will later discuss why this property is important for constructing the encoder in Section 3.1.

The goal of this work is to model the complete data distribution $p_{\mathcal{F}}$ given only the incomplete observations contained in the dataset $\mathcal{D}$. We do not focus on learning the distribution $p_{\mathcal{I}}$ as this distribution is typically not the primary concern in the applications we focus on. For simplicity, we make the further assumption that f and $\mathbf{t}$ are independent, that is, the

generative process of an incomplete case $(\mathbf{x}, \mathbf{t})$ is given by

$$f \sim p_{\mathcal{F}}(f), \quad \mathbf{t} \sim p_{\mathcal{I}}(\mathbf{t}), \quad \mathbf{x} = [f(t_i)]_{i=1}^{|\mathbf{t}|}. \quad (1)$$

In Appendix D we will discuss the implications of this assumption and how to relax it. In the next section, we present models for finite index sets. In Section 4 we present models for continuous index sets.

3. Incomplete Data with a Finite Index Set

In this section, we focus on the case where the index set $\mathcal{I}$ is finite. We begin by describing a base encoder-decoder framework, which can be trained by models based on VAEs and GANs.

3.1. Encoder-Decoder Framework

We employ a general encoder-decoder framework for modeling incomplete data. For the decoder, we model the distribution of the complete data $p_{\mathcal{F}}(f)$ as a two-step procedure:

$$\mathbf{z} \sim p_z(\mathbf{z}), \quad f = g_\theta(\mathbf{z}) \quad (2)$$

where we first draw a latent code $\mathbf{z}$ from a simple distribution $p_z(\mathbf{z})$ such as a standard Gaussian. We then transform $\mathbf{z}$ into a complete sample $f \in \mathbb{R}^{\mathcal{I}}$ through a deterministic function $g_\theta(\mathbf{z})$.

The encoder, denoted $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$, aims to model the posterior distribution of the latent code associated with an incomplete example $(\mathbf{x}, \mathbf{t})$. Since the representation of incomplete data is permutation invariant as noted earlier, the encoder should also be permutation invariant (Zaheer et al., 2017). Below we define such a function $m(\mathbf{x}, \mathbf{t})$ that provides a simple construction of the encoder.

Definition 1. *The masking function $m(\mathbf{x}, \mathbf{t})$ maps an incomplete data case $(\mathbf{x}, \mathbf{t})$ to a masked form in $\mathbb{R}^{\mathcal{I}}$ with all missing entries replaced by zero. Specifically, let $\mathbf{v} = m(\mathbf{x}, \mathbf{t})$ then each entry of $\mathbf{v}$ has the form $v_t = \sum_{i=1}^{|\mathbf{t}|} x_i \mathbf{1}\{t_i = t\}$ for all $t \in \mathcal{I}$.*

The masking function serves as an interface that transforms an incomplete data case $(\mathbf{x}, \mathbf{t})$ with arbitrary size to the masked form $m(\mathbf{x}, \mathbf{t})$ of fixed dimension in $\mathbb{R}^{\mathcal{I}}$.

We can construct the encoder distribution to have the form of $q_\phi(\mathbf{z}|m(\mathbf{x}, \mathbf{t}))$, where the distribution is only parameterized by the fixed-dimensional masked data $m(\mathbf{x}, \mathbf{t})$. For example, we can use a Gaussian encoder, $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{v}), \Sigma_\phi(\mathbf{v}))$ where $\mathbf{v} = m(\mathbf{x}, \mathbf{t})$, with its mean μ_ϕ and diagonal covariance Σ_ϕ constructed using neural networks.

Note that in the presence of missing data, we cannot use a deterministic encoder as in standard autoencoders for complete data, because different incomplete samples may carry
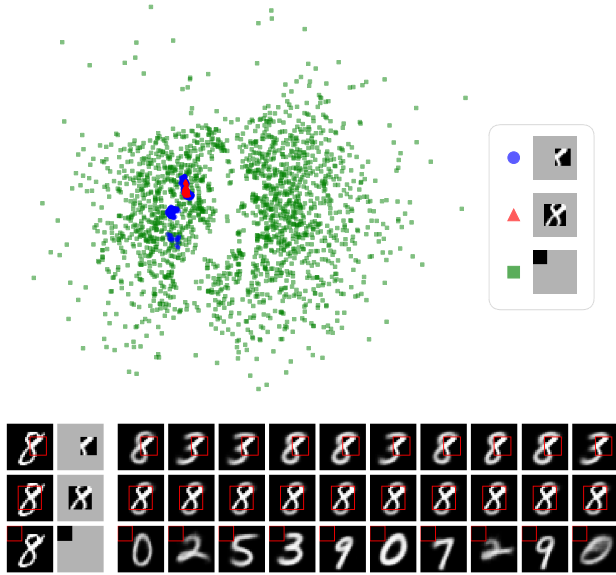
Figure 1. At the top we plot the 2D latent codes drawn from the encoder $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$ with three different incomplete MNIST examples. At the bottom, each row corresponds to one of the three examples we encode. All three cases come from the same MNIST image as the leftmost image in each row except we observe different rectangular regions on the image. The resulting incomplete images are shown as the second image in each row where the gray pixels corresponds to the missing entries. The ten images on the right are decoded from the random samples drawn from the encoder $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$. The pixels inside of each red box on those sampled images are the observed pixels and those outside are generated by the model described in Section 6.1. Note that the blurry completion is due to the insufficient capacity of 2D latent codes. The latent space plot shows that the second case (red) has relatively low uncertainty. In contrast, the third case (green) has high uncertainty whose encoded distribution looks similar to the Gaussian prior $p_z(\mathbf{z})$.

very different levels of uncertainty as shown in Figure 1. In other words, there could be many different latent codes $\mathbf{z}$ that can be decoded into a variety of complete samples that are consistent with the observed part of the data.

We next describe two training strategies for learning the encoder and decoder.

3.2. Partial Variational Autoencoder

To train the framework using maximum likelihood, we construct a proper density model by adding i.i.d. noise to each component of $g_\theta(\mathbf{z}, t_i)$ for all $t_i \in \mathbf{t}$, where $g_\theta(\mathbf{z}, t_i)$ denotes $f(t_i)$ with $f = g_\theta(\mathbf{z})$. For example, for real-valued data, the distribution $p(x_i|g_\theta(\mathbf{z}, t_i))$, or referred in short as $p_\theta(x_i|\mathbf{z}, t_i)$, could be a Gaussian $\mathcal{N}(x_i|f(t_i), \sigma^2)$ with a pre-defined variance σ^2. As a result, the joint distribution

of an incomplete data case $(\mathbf{x}, \mathbf{t})$ is

$$p(\mathbf{x}, \mathbf{t}) = \int p(\mathbf{z})p_\mathcal{I}(\mathbf{t}) \prod_{i=1}^{|\mathbf{t}|} p_\theta(x_i|\mathbf{z}, t_i)d\mathbf{z}.$$

Since this marginal is intractable, we instead maximize a variational lower bound on $\log p(\mathbf{x}, \mathbf{t})$ given by

$$\int q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t}) \log \frac{p_z(\mathbf{z})p_\mathcal{I}(\mathbf{t}) \prod_{i=1}^{|\mathbf{t}|} p_\theta(x_i|\mathbf{z}, t_i)}{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} d\mathbf{z}. \quad (3)$$

To learn the distribution of the data parameterized by $g_\theta(\mathbf{z})$, we only need to learn the parameters of $p_\theta(x|\mathbf{z}, t)$ and $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$, denoted by θ and ϕ respectively. Due to the assumed independence between $\mathbf{t}$ and $\mathbf{z}$, when taking the derivative of (3) with respect to θ and ϕ, the term $p_\mathcal{I}(\mathbf{t})$ can be dropped. As a result, the model can be equivalently learned by maximizing the variational lower bound on the conditional log-likelihood given below where $p_\mathcal{D}$ denotes the empirical distribution of the training dataset $\mathcal{D}$:

$$\mathbb{E}_{(\mathbf{x}, \mathbf{t}) \sim p_\mathcal{D}} \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \left[\log \frac{p_z(\mathbf{z}) \prod_{i=1}^{|\mathbf{t}|} p_\theta(x_i|\mathbf{z}, t_i)}{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \right]. \quad (4)$$

This training objective has been previously introduced as the Partial Variational Autoencoder (Ma et al., 2018; 2019), which we abbreviate as P-VAE. Neural processes (Garnelo et al., 2018a;b) and MIWAE (Mattei & Frellsen, 2019) also have the similar structure. All of these previous approaches are introduced as optimizing a conditional objective directly while here we come from the complete generative process that includes the point process $p_\mathcal{I}$.

Similar to VAEs, we can use reparameterizable distributions for the encoder $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$, such as Gaussians as we described in Section 3.1. There are various techniques to construct more expressive encoders that can also be used in our case. For example, we can apply inverse autoregressive flows (Kingma et al., 2016) to transform distributions or use semi-implicit variational inference (Yin & Zhou, 2018) to flexibly construct expressive encoders. Moreover, the objective (4) can also adopt importance weighted autoencoders (Burda et al., 2016; Mattei & Frellsen, 2019) to optimize a tighter variational bound.

3.3. Partial Bidirectional GAN

Unlike P-VAE, which requires specifying an explicit density, we can instead learn the distribution $p_\mathcal{F}(f)$ parameterized by (2) implicitly based on generative adversarial networks (GANs) (Goodfellow et al., 2014). Inspired by the Bidirectional GAN (BiGAN) (Donahue et al., 2017; Dumoulin et al., 2017), we propose a model that improves on Mis-GAN (Li et al., 2019) for modeling incomplete data. We

call the proposed approach the Partial Bidirectional GAN (P-BiGAN).

The overall structure of P-BiGAN is shown in Figure 2, which consists of a separate encoding and decoding part. Given an incomplete dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^n$, P-BiGAN aims to match the joint distribution of the incomplete data $(\mathbf{x}, \mathbf{t})$ sampled from $\mathcal{D}$ and the corresponding code $\mathbf{z}$ drawn from $p_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$ to the joint distribution of generated masked outputs $(g_\theta(\mathbf{z}', \mathbf{t}'), \mathbf{t}')$ where $\mathbf{z}'$ is a random latent code drawn from the prior $p_z(\mathbf{z}')$ and $\mathbf{t}'$ is a set of random indices separately sampled from $\mathcal{D}$.[1] Note that we use $g_\theta(\mathbf{z}, \mathbf{t})$ as shorthand notation for $[g_\theta(\mathbf{z}, t_i)]_{i=1}^{|\mathbf{t}|}$.

Specifically, P-BiGAN tries to solve the following minimax optimization problem:

$$\min_{\theta,\phi} \max_D L(D, \theta, \phi) \qquad (5)$$

where

$$L(D, \theta, \phi) = \mathbb{E}_{(\mathbf{x},\mathbf{t})\sim p_\mathcal{D}} \mathbb{E}_{\mathbf{z}\sim p_\phi(\mathbf{z}|\mathbf{x},\mathbf{t})} [\log D(\mathbf{x}, \mathbf{t}, \mathbf{z})]$$
$$+ \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})} \mathbb{E}_{(\mathbf{x},\mathbf{t})\sim p_\mathcal{D}} [\log(1 - D(g_\theta(\mathbf{z}, \mathbf{t}), \mathbf{t}, \mathbf{z}))].$$

P-BiGAN is compatible with many GAN variations. Other form of $L(D, \theta, \phi)$ such as the loss used by BigBiGAN (Donahue & Simonyan, 2019) can also be applied. The encoder of P-BiGAN can be constructed more flexibly than P-VAE as we don't need to evaluate the density of the drawn samples. For example, we can construct a distribution using the generative process shown below where the encoded samples are first drawn from a parameterized Gaussian followed by a transformation g_ϕ:

$$\mathbf{v} = m(\mathbf{x}, \mathbf{t}), \ \mathbf{u} \sim \mathcal{N}(\mu_\phi(\mathbf{v}), \Sigma_\phi(\mathbf{v})), \ \mathbf{z} = g_\phi(\mathbf{u}).$$

The discriminator of P-BiGAN takes as input an incomplete data sample $(\mathbf{x}, \mathbf{t})$ and its corresponding code $\mathbf{z}$. Following MisGAN (Li et al., 2019), the discriminator is constructed in the form of $D(m(\mathbf{x}, \mathbf{t}), \mathbf{z})$, which can also be parameterized by neural networks. Proposition 1 below justifies the use of $m(\mathbf{x}, \mathbf{t})$ when the data lies in a finite space, under the independence assumption described in Section 2.

Proposition 1. *(Adapted from Li et al. (2019, Theorem 2)) When the data space and the index set are both finite, given a distribution $p_\mathcal{I}(\mathbf{t})$, two distributions $p_\theta(f)$ and $p_{\theta'}(f)$ induce the same distribution of $m(\mathbf{x}, \mathbf{t})$ if and only if they have the same marginals $p_\theta(\mathbf{x}|\mathbf{t}) = p_{\theta'}(\mathbf{x}|\mathbf{t})$ for all $\mathbf{t}$ with $p_\mathcal{I}(\mathbf{t}) > 0$.*

Moreover, following Donahue et al. (2017), the global optimum of (5) is achieved if and only if the induced joint distribution over $\mathbf{x}$, $\mathbf{t}$ and $\mathbf{z}$ are identical for the encoder

---

[1] Here $\mathbf{t}'$ is essentially drawn from $p_\mathcal{I}(\mathbf{t})$, the marginal of $p_\mathcal{D}(\mathbf{x}, \mathbf{t}) \equiv p_\mathcal{D}(\mathbf{x})p_\mathcal{I}(\mathbf{t})$, due to the independence assumption.
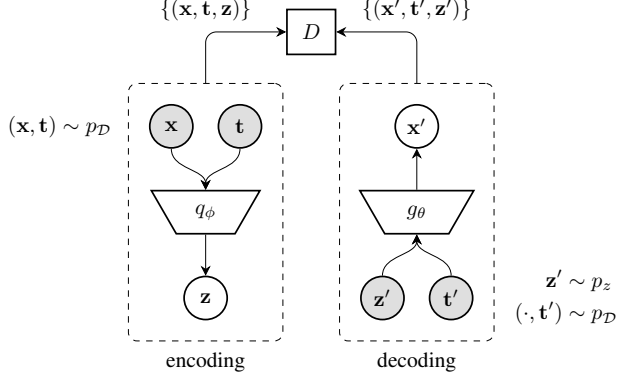

$\{(\mathbf{x}, \mathbf{t}, \mathbf{z})\}$ $\qquad$ $\{(\mathbf{x}', \mathbf{t}', \mathbf{z}')\}$

$(\mathbf{x}, \mathbf{t}) \sim p_\mathcal{D}$

$\mathbf{z}' \sim p_z$
$(\cdot, \mathbf{t}') \sim p_\mathcal{D}$

encoding $\qquad$ decoding

Figure 2. The structure of P-BiGAN.

$q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$ and decoder g_θ. We can show the following invertibility relationship between the encoder and the decoder when optimality is attained (see Appendix A for the proof).

Proposition 2. *When the optimally learned encoder and decoder achieve the same joint distribution over $(\mathbf{x}, \mathbf{t})$ and $\mathbf{z}$ by optimizing (5), for any $(\mathbf{x}, \mathbf{t})$ with non-zero probability, if $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$ we have $g_\theta(\mathbf{z}, \mathbf{t}) = \mathbf{x}$ almost surely.*

In practice, it is hard to achieve optimality with GAN training, and therefore we usually don't have a very good match between $g_\theta(\mathbf{z}, \mathbf{t})$ and $\mathbf{x}$ as described in Proposition 2. For applications that rely on the encoded representation $\mathbf{z}$ such as those that we will present later in Section 5, we found that further adding an autoencoding loss in addition to the original P-BiGAN loss $L(D, \theta, \phi)$ to enforce this consistency improves the results (see Appendix B). Specifically, when training the model, we instead use the following objective with some $\lambda \geq 0$ that controls the strength of the autoencoding term:

$$L(D, \theta, \phi) + \lambda \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x},\mathbf{t})} \left[\sum_{i=1}^{|\mathbf{t}|} \ell(x_i, g_\theta(\mathbf{z}, t_i)) \right] \quad (6)$$

where $\ell(x, x')$ is a loss function that measures the discrepancy between x and x' such as L_2 loss for real-valued data, which is analogous to the log likelihood term $\log p_\theta(x_i|\mathbf{z}, t_i)$ in P-VAE.

Finally, we point out that there are two main differences between P-BiGAN and MisGAN. First, P-BiGAN utilizes the independence assumption to sample $\mathbf{t}'$ directly from the training data instead of learning the distribution $p_\mathcal{I}$ as in MisGAN. This not only makes the training faster, but improves the quality of the resulting data generator when the distribution $p_\mathcal{I}$ is difficult to learn. Second, the imputer in MisGAN can only be applied to data with finite index set. Since P-BiGAN is an encoder-decoder framework, this not only greatly simplifies the model complexity but can be generalized to the case of continuous index sets as we discuss in the next section.

4. Irregularly-Sampled Time Series: The Continuous Index Set Case

For continuous time series defined over some time interval $[0, T]$, the index set $\mathcal{I} = [0, T]$ is no longer finite. In this section, we propose a computationally efficient encoder-decoder architecture for modeling irregularly-sampled time series data.

4.1. Decoder: Kernel Smoother

To model the distribution of continuous functions over the time interval $[0, T]$, we first use a standard convolutional neural network (CNN) decoder to generate a length-L output $v_1, \ldots, v_L$ as the reference values on a set of evenly-spaced locations $u_1, \ldots, u_L$ over $[0, T]$, and then construct the function as the smooth interpolation of those references. Here we use a kernel smoother to interpolate at arbitrary times. Specifically, we model irregularly-sampled time series as samples from a distribution over functions defined by the following generative process:

$$
\begin{aligned}
\mathbf{z} &\sim p_z(\mathbf{z}), \\
\mathbf{v} &= \text{CNN}_\theta(\mathbf{z}), \\
f(t) &= \frac{\sum_{i=1}^{L} K(u_i, t) v_i}{\sum_{i=1}^{L} K(u_i, t)}
\end{aligned}
\tag{7}
$$

where K is a smoothing kernel. We use the Epanechnikov kernel, $K(u, t) = \max(3/4(1 - (|u - t|/\beta)^2), 0)$, which has finite support so that each location is only influenced by a small number of its neighbors. Moreover, we can compute the kernel values among those neighbors only once in the beginning as those stay constant during training.

This kernel smoother layer can also be applied to multivariate time series by interpolating each channel independently using the kernel smoother on a CNN with multi-channel output.[2]

4.2. Encoder: Continuous Convolutional Layer

Inspired by CNNs, we adapt the convolutional layer in CNNs to accommodate irregularly-sampled time series. To mimic the locally-focused receptive field of standard convolutional layers, we generalize the discrete filter (or kernel) to a continuous function $w(t)$ defined over a fixed small interval, say $[0, \delta]$ with a tunable kernel width δ. That is, $w(t) = 0$ when $t \notin [0, \delta]$.

Similar to the convolutional layers in CNNs, we perform cross-correlation between the continuous filter $w(t)$ and the masked function $f(t) = \sum_{i=1}^{|\mathbf{t}|} x_i \mathbf{1}\{t_i = t\}$ induced by the

---

[2]For multivariate time series with C channels defined over the time interval $[0, T]$, the index set $\mathcal{I} = \{1, \ldots, C\} \times [0, T]$.

observations in time series as follows:

$$
\begin{aligned}
(w \star f)(r) &= \int w(t - r) \left(\sum_{i=1}^{|\mathbf{t}|} x_i \mathbf{1}\{t_i = t\} \right) dt \\
&= \sum_{i: t_i - r \in [0, \delta]} w(t_i - r) x_i.
\end{aligned}
$$

We apply this operation on L (need not be the same L for the decoder) evenly-spaced locations $r_1, \ldots, r_L$ spanning the time interval $[0, T]$ to transform non-uniform inputs to a length-L uniform representation $[(w \star f)(r_i)]_{i=1}^{L}$.

We construct the continuous filter $w(t)$ as a piecewise linear function parameterized by a small number of evenly-spaced knots over $[0, \delta]$. This is equivalent to a degree-1 B-spline (Piegl & Tiller, 2012) and backpropagation through such functions can be computed efficiently (Fey et al., 2018). We found that degree-1 B-splines already perform well comparing with more expensive higher-order B-spline interpolation.

In preliminary experiments, we compared this architecture with several alternatives. First, we use a multi-layer perceptron (MLP) to approximate an arbitrary function as in neural processes (Garnelo et al., 2018a;b). However, an MLP is not as parameter efficient as a piecewise linear function whose only parameters are the values of the knots. We found that we need many more parameters for an MLP to achieve similar performance to piecewise linear functions and the optimization is generally more difficult. We also compare with a kernel smoother similar to the decoder described in Section 4.1 to provide another parameter efficient choice. Although a kernel smoother gives roughly the same performance, it is about 20% slower then the piecewise linear function due to the expensive normalization. Note that although not as efficient as the convolutional structure, we can also construct the encoder with the attention mechanism such as in Kim et al. (2019) and Lee et al. (2019).

We can extend this operator to the case when there are C_{in} input channels and C_{out} output channels. Given a multi-channel incomplete example $(\mathbf{x}, \mathbf{t}) = \{(\mathbf{x}_c, \mathbf{t}_c)\}_{c=1}^{C_{\text{in}}}$, we define the continuous convolutional layer as

$$
\text{CONV}_k(r, \mathbf{x}, \mathbf{t}) = b_k + \sum_{c=1}^{C_{\text{in}}} \sum_{i: t_{c,i} - r \in [0, \delta]} w_{c,k}(t_{c,i} - r) x_{c,i}
$$

where we include a bias term b_k similar to standard convolutional layers. For each time series, the continuous convolutional layer produces a 2D output $\mathbf{V} \in \mathbb{R}^{C_{\text{out}} \times L}$ where $V_{kj} = \text{CONV}_k(r_j, \mathbf{x}, \mathbf{t})$, which can then be fed into a regular CNN encoder.

Similar to the kernel smoother, we can also precompute the distance to the neighboring reference points once in the beginning for the continuous convolutional layer. Note that the same architecture can also be used for the discriminator

in P-BiGAN. In fact, the continuous convolutional layer serves as a drop-in replacement for the masking function $m(\mathbf{x}, \mathbf{t})$ in Definition 1 for continuous time series.

5. Applications

In this section, we briefly describe two applications of our encoder-decoder model framework: missing data imputation and supervised learning.

5.1. Missing Data Imputation

Given an incomplete example $(\mathbf{x}, \mathbf{t})$, the goal of missing data imputation is to infer the values of the unobserved features $\mathbf{x}'$ that correspond to indices $\mathbf{t}' \subseteq \mathcal{I} \setminus \mathbf{t}$ according to $p(\mathbf{x}'|\mathbf{t}', \mathbf{x}, \mathbf{t})$. Once the model is trained, imputations can be drawn according to the distribution

$$p(\mathbf{x}'|\mathbf{t}', \mathbf{x}, \mathbf{t}) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \left[p_\theta(\mathbf{x}'|\mathbf{z}, \mathbf{t}') \right].$$

Since $p(\mathbf{x}'|\mathbf{z}, \mathbf{t}')$ is defined implicitly by (2), sampling from $p(\mathbf{x}'|\mathbf{t}', \mathbf{x}, \mathbf{t})$ can be done with the following steps:

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t}), \quad f = g_\theta(\mathbf{z}), \quad \mathbf{x}' = [f(t_i')]_{i=1}^{|\mathbf{t}'|}.$$

5.2. Supervised Learning

We can perform supervised learning when each incomplete data case has a corresponding prediction target. We focus on the classification case where the prediction target is a class label y. We assume y depends only on the latent representation $\mathbf{z}$ in the generative process (2).

For P-VAE, we augment the training objective to include the classification term $p(y|\mathbf{z})$ as follows:

$$\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \left[\log \frac{p_z(\mathbf{z}) p(y|\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{t})}{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \right] \quad (8)$$

$$= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \left[\log \frac{p_z(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{t})}{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \right]}_{\text{regularization}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} [\log p(y|\mathbf{z})]}_{\text{classification}}.$$

Note that we use the encoder that depends only on the incomplete data instead of the most general form $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t}, y)$, which includes the class label as well. This allows us to decompose (8) into two separate terms: a regularization term as in P-VAE and a classification term $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})}[\log p(y|\mathbf{z})]$. Therefore, we can either train the classifier $p(y|\mathbf{z})$ along with the pre-trained encoder $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$ or train the whole model jointly from scratch. Moreover, this decomposition allows us to do semi-supervised learning easily: we only include the classification term when the label is available.

Similarly, for P-BiGAN, we can train a classifier separately with the pre-trained encoder or add a classification loss

$-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})}[\log p(y|\mathbf{z})]$ into (6) to jointly train the classifier with P-BiGAN.

Once the model is trained, prediction can be performed efficiently with the expectation approximated using a small number of samples ($S = 1$ suffices in practice):

$$y^* = \underset{y}{\arg\max} \, \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})} \left[\log p(y|\mathbf{z}) \right]$$

$$\approx \underset{y}{\arg\max} \, \frac{1}{S} \sum_{s=1}^{S} \log p(y|\mathbf{z}_s), \text{ where } \mathbf{z}_s \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t}).$$

6. Experiments

In this section, we first evaluate the models on the finite index set case described in Section 2. We assess our framework using image modeling and completion experiments with controlled missingness on standard image benchmarks. Next, we evaluate the performance of our framework equipped with the continuous-time encoder/decoder using the multivariate irregularly-sampled time series classification task on a medical benchmark. Additional results on time series imputation and visualization of the learned temporal process on synthetic data are provided in Appendix C.

6.1. Image Modeling and Completion

MisGAN was previously shown to outperform a range of methods on the problem of learning the image distribution from incomplete data. We follow the experimental setup of MisGAN to quantitatively evaluate the performance of P-VAE and P-BiGAN on the imputation task using two image benchmarks, MNIST (LeCun et al., 2010) and CelebA (Liu et al., 2015). We train the models using incomplete images under two missing patterns: i) square observation where all pixels are missing except for a square occurring at a random location on the image, and ii) independent dropout where each pixel is independently missing with a given probability. For both missing patterns, we vary the missing rate from 10% to 90%.

To evaluate the quality of a model, we impute all the incomplete images with the observed pixels kept intact and use the Fréchet Inception Distance (FID) (Heusel et al., 2017) between the completed images and the original fully-observed dataset as the evaluation metric.[3]

For P-VAE and P-BiGAN, we use the same convolutional decoder architecture used in MisGAN. For P-VAE, we use an encoder $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})$ constructed by

$$\mathbf{z}_0 \sim \mathcal{N}(\mu_\phi(m(\mathbf{x}, \mathbf{t})), \Sigma_\phi(m(\mathbf{x}, \mathbf{t}))), \ \mathbf{z} = \text{IAF}_\phi(\mathbf{z}_0) \quad (9)$$

---

[3]Unlike FID that evaluates distributional discrepancy, metrics like RMSE that measure the discrepancy of imputation against the ground truth are not suitable here when the true posterior is highly multimodal, especially in the cases with high missingness. See Li et al. (2019, Appendix C) for more details.
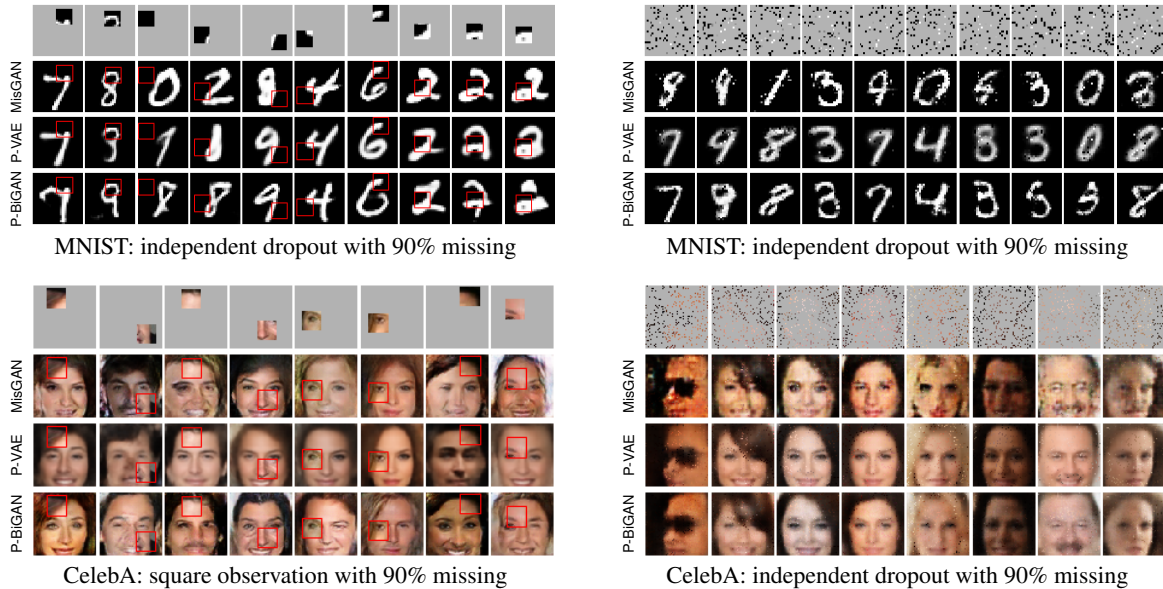
Figure 3. Imputation results on MNIST and CelebA under 90% missingness. The images in first row of each block are the incomplete images where gray pixels indicate missing data. For square observation cases on the left, the pixels inside of each red box are observed.
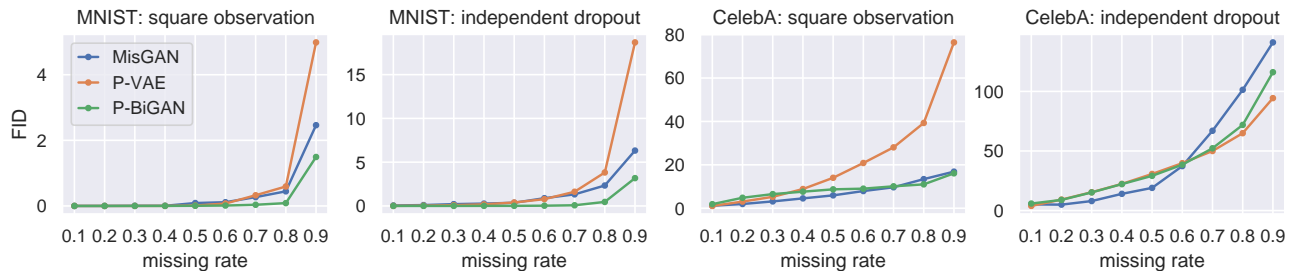

Figure 4. Comparison of FIDs (the lower the better) on MNIST and CelebA with different missing patterns and missing rates.

using convolutional μ_ϕ and Σ_ϕ. Here we use two-layers of inverse autoregressive flow (IAF) (Kingma et al., 2016). In addition, we use importance weighted autoencoders (IWAEs) with five importance weights.

For P-BiGAN, we use the same architecture as P-VAE including the IAF component except we do not compute its density. For the discriminator, we concatenate the embedding of $(\mathbf{x}, \mathbf{t})$ computed using the same convolutional architecture as the encoder and the embedding of $\mathbf{z}$ using a two-layer MLP. The concatenated embedding is then fed into another two-layer MLP to produce the score.

Figure 4 compares the FIDs of MisGAN, P-VAE and P-BiGAN under different missing patterns and missing rates. For MNIST, it shows that P-BiGAN performs slightly better than MisGAN due to the more expressive encoder architecture of P-BiGAN. P-VAE has the worst FID scores especially for high missing rates, which is reflected by the blurriness of the imputation results shown in Figure 3.

For square observations on CelebA, P-BiGAN and MisGAN perform about the same, while P-VAE has significantly worse FIDs also due to the blurriness. However, for the independent dropout case, P-VAE performs the best when the missing rate is high. It seems that GAN-based models are better at capturing spatial correlations when learning with convolutional networks, but when neighboring pixels rarely co-occur, they are not able to learn effectively. Because of the autoencoding regularization used in P-BiGAN, it shares the benefit of autoencoding when it comes to independent dropout and thus also outperforms MisGAN when the missing rate is high. However, for low missingness, MisGAN outperforms both P-VAE and P-BiGAN due to its U-Net imputer that allows the model to produce better imputation results when the images are almost fully observed.

Table 1 shows the per-epoch running time and the number of parameters of each model, where the running time is roughly proportional to the number of parameters. For MNIST, it shows that P-BiGAN and P-VAE have proportionally less

Table 1. The average per-epoch running time in minutes and the number of parameters of each model.

dataset	method	time	params
MNIST	MisGAN	1.72	8.67M
	P-VAE	0.84	4.70M
	P-BiGAN	1.38	6.01M
CelebA	MisGAN	39.47	40.35M
	P-VAE	11.93	11.32M
	P-BiGAN	14.78	16.71M

Table 2. AUC (mean $\pm$ std), per-epoch time in hours and the number of parameters of each model on MIMIC-III.

method	AUC (%)	time	params
M-RNN	83.87 ± 0.80	–	101.6K
GRU-D	83.88 ± 0.65	0.11	2.6K
Latent ODE	85.71 ± 0.38	2.62	154.7K
Cont Classifier	84.87 ± 0.18	0.03	30.5K
Cont P-VAE	85.52 ± 0.54	0.05	67.8K
Cont P-BiGAN	86.02 ± 0.38	0.22	73.2K

parameters then MisGAN, even if they both use a large encoder that roughly doubles the parameters of the decoder.

For CelebA, MisGAN uses a separate U-Net imputer trained with another discriminator, while P-BiGAN only utilizes an additional encoder to impute along with the decoder. Moreover, P-BiGAN does not model the missingness that requires an extra pair of generator and discriminator for the masks as in MisGAN. Therefore, the great reduction in model parameters makes P-BiGAN about 2.7x faster than MisGAN. On the other hand, P-VAE enjoys the simplest training procedure and the lowest model complexity without the need for learning separate discriminators. As a result, it is the fastest among the three models.

6.2. Classification of Irregularly-Sampled Time Series

In this section, we evaluate our framework on a healthcare multivariate time series dataset, MIMIC-III (Johnson et al., 2016), using the mortality prediction task.

MIMIC-III consists of about 53,000 data cases. We use 12 irregularly-sampled temporal variables that are recorded within 48 hours. If we discretize observations into 1-minute intervals, the overall missing rate is about 92% on average. Our task is to predict the in-hospital mortality as a binary classification problem. We use the area under the ROC curve (AUC) as the evaluation metric. We split the data into 64% for training, 16% for validation, and the remaining 20% for testing.

We evaluate both P-VAE and P-BiGAN equipped with the continuous encoder and decoder described in Section 4, which we denote Cont P-VAE and Cont P-BiGAN respectively. For the decoder, we use 128 evenly-spaced references over $[0, 1]$ for the kernel smoother, whose values are the output of a standard CNN decoder. We use the Epanechnikov kernel with the kernel bandwidth set to 3/128. For the continuous convolutional layer in the encoder, we use 64 output channels with 98 evenly-spaced references. The piecewise-linear convolutional kernel has width 2/98 with 7 knots. The output of the continuous convolutional layer is then fed into a standard IAF encoder as in (9).

For Cont P-VAE we use 8 importance weights for the IWAE objective. Both P-VAE and P-BiGAN are trained with a separate two-layer fully-connected classifier jointly. As an ablation study, we compare our models with a classifier, denoted Cont Classifier, that combines the same encoder and classifier used in Cont P-VAE and Cont P-BiGAN, but without adding extra Gaussian noise in the encoder. We compare our models with two recent methods designed for irregularly-sampled time series: GRU-D (Che et al., 2018) and Latent ODE (Rubanova et al., 2019). We also include a baseline model for learning time series with missing data, Multi-directional RNN (M-RNN) (Yoon et al., 2018). Since M-RNN does not work well with massive missingness, it is run on the modified data with observations quantized into 30-minute intervals.

Table 2 shows predictive performance, per-epoch training time and model sizes. The training time of M-RNN is omitted because it runs on the much smaller quantized dataset and the time is thus not comparable to other methods. The table shows that Cont P-BiGAN achieves the highest mean AUC of all of the methods, followed closely by Latent ODE and Cont P-VAE. Although the difference between Cont P-BiGAN and Latent ODE is not statistically significant, Cont P-BiGAN is over 10 times faster per training epoch. On the other hand, Cont P-VAE is over 60 times faster than Latent ODE.

These run time differences are due to the fact that the convolutional architectures used in the proposed approaches are highly parallelizable compared to the recurrent structure used by the baseline models. Moreover, our models directly parameterize temporal functions using (7); on the contrary, Latent ODE instead models the dynamics using ODEs, which requires expensive numerical integration. Meanwhile, Cont P-VAE is faster than Cont P-BiGAN because Cont P-BiGAN requires running continuous convolutional layers in both the encoder and discriminator, which is the most expensive computation during training that marshals time series of variable size.

We also note that Cont Classifier achieves better AUCs than M-RNN and GRU-D without generative modeling. This

shows that the continuous convolutional layer provides an effective intermediate representation for irregularly-sampled time series.

7. Conclusion

In this paper, we have presented the problem of modeling irregularly-sampled time series from the perspective of missing data. We introduced an encoder-decoder framework for modeling general missing data problems and introduced two model families leveraging this framework: P-VAE and P-BiGAN. We showed how to integrate this framework with a continuous convolutional layer to efficiently featurize irregularly-sampled time series for interfacing with standard neural network architectures. Our proposed models achieve comparable predictive performance to the recently-proposed Latent ODE model, while offering significantly faster training times.

References

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

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

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pp. 2980–2988, 2015.

Donahue, J. and Simonyan, K. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pp. 10542–10552, 2019.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Dumoulin, V., Belghazi, M. I. D., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. Adversarially learned inference. In *International Conference on Learning Representations (ICLR)*, 2017.

Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International Conference on Machine Learning (ICML)*, pp. 1690–1699, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6629–6640, 2017.

Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. In *International Conference on Learning Representations (ICLR)*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available:* http://yann.lecun.com/exdb/mnist, 2, 2010.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning (ICML)*, pp. 3744–3753, 2019.

Li, S. C.-X., Jiang, B., and Marlin, B. MisGAN: Learning from incomplete data with generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Ma, C., Gong, W., Hernández-Lobato, J. M., Koenigstein, N., Nowozin, S., and Zhang, C. Partial VAE for hybrid recommender system. In *NIPS Workshop on Bayesian Deep Learning. 2018*, 2018.

Ma, C., Tschiatschek, S., Palla, K., Hernandez-Lobato, J. M., Nowozin, S., and Zhang, C. EDDI: Efficient dynamic discovery of high-value information with partial VAE. In *International Conference on Machine Learning (ICML)*, pp. 4234–4243, 2019.

Mattei, P.-A. and Frellsen, J. MIWAE: Deep generative modelling and imputation of incomplete data sets. In *International Conference on Machine Learning (ICML)*, pp. 4413–4423, 2019.

Piegl, L. and Tiller, W. *The NURBS book*. Springer Science & Business Media, 2012.

Rubanova, Y., Chen, T. Q., and Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pp. 5321–5331, 2019.

Yin, M. and Zhou, M. Semi-implicit variational inference. In *International Conference on Machine Learning (ICML)*, pp. 5646–5655, 2018.

Yoon, J., Zame, W. R., and van der Schaar, M. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 66(5):1477–1490, 2018.

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