Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks

Jinsung Yoon, William R. Zame, and Mihaela van der Schaar, *Fellow, IEEE*

***Abstract*—Missing data is a ubiquitous problem. It is especially challenging in medical settings because many streams of measurements are collected at different—and often irregular—times. Accurate estimation of the missing measurements is critical for many reasons, including diagnosis, prognosis, and treatment. Existing methods address this estimation problem by interpolating within data streams or imputing across data streams (both of which ignore important information) or ignoring the temporal aspect of the data and imposing strong assumptions about the nature of the data-generating process and/or the pattern of missing data (both of which are especially problematic for medical data). We propose a new approach, based on a novel deep learning architecture that we call a Multi-directional Recurrent Neural Network that interpolates within data streams and imputes across data streams. We demonstrate the power of our approach by applying it to five real-world medical datasets. We show that it provides dramatically improved estimation of missing measurements in comparison to 11 state-of-the-art benchmarks (including Spline and Cubic Interpolations, MICE, MissForest, matrix completion, and several RNN methods); typical improvements in Root Mean Squared Error are between 35%–50%. Additional experiments based on the same five datasets demonstrate that the improvements provided by our method are extremely robust.**

***Index Terms*—Missing data, temporal data streams, imputation, recurrent neural nets.**

I. Introduction

MISSING data/measurements present a ubiquitous problem. The problem is especially challenging in medical settings which present time series containing many streams of measurements that are sampled at different and irregular times, and is especially important in these settings because accurate estimation of these missing measurements is often critical for accurate diagnosis, prognosis and treatment, as well as for accurate modeling and statistical analyses. This paper presents a new method for estimating missing measurements in time series data, based on a novel deep learning architecture. By comparing our method with current state-of-the-art benchmarks on a variety of real-world medical datasets, we demonstrate that our method is much more accurate in estimating missing measurements, and that this accuracy is reflected in improved prediction of outcomes.

The most familiar methods for estimating missing data follow one of three approaches, usually called *interpolation, imputation* and *matrix completion*. Interpolation methods such as [1], [2] exploit the correlation among measurements at different times *within each stream* but ignore the correlation across streams. Imputation methods such as [3]–[6] exploit the correlation among measurements at the same time *across different streams* but ignore the correlation within streams. Because medical measurements are frequently correlated both within streams *and* across streams (e.g., blood pressure at a given time is correlated both with blood pressure at other times and with heart rate), each of these approaches loses potentially important information. Matrix completion methods such as [7]–[10] do exploit correlations within and across streams, but assume that the data is static – hence ignore the temporal component of the data – or that the data is perfectly synchronized – an assumption that is routinely violated in medical time series data. Some of these methods also make modeling assumptions about the nature of the data-generating process or of the pattern of missing data. Our approach is expressly designed to exploit both the correlation within streams and the correlation across streams and to take into account the temporal and non-synchronous character of the data; our approach makes no modeling assumptions about the data-generating process or the pattern of missing data. (We do assume – as is standard in most of the literature – that the data is *missing at random* [1]. Dealing with data that is not missing at random [11] presents additional challenges and is left for future works.)

Our method relies on a novel neural network architecture that we call a *Multi-directional Recurrent Neural Network (M-RNN)*. Our M-RNN contains both an interpolation block and an imputation block and it trains these blocks *simultaneously*, rather than separately (See Fig. 1 and 2). Like a bi-directional RNN (Bi-RNN) [12], an M-RNN operates forward and backward

Manuscript received November 23, 2017; revised February 28, 2018, June 19, 2018, and September 9, 2018; accepted October 1, 2018. Date of publication October 8, 2018; date of current version April 19, 2019. This work was supported in part by the Office of Naval Research (ONR) and in part by the National Science Foundation under Grants ECCS1462245, ECCS1533983, and ECCS1407712. *(Corresponding author: Jinsung Yoon.)*

J. Yoon is with the Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: jsyoon0823@ucla.edu).

W. R. Zame is with the Department of Economics and Mathematics, University of California.

M. van der Schaar is with the Department of Engineering Science, University of Oxford and also with the Alan Turing Institute.

Digital Object Identifier 10.1109/TBME.2018.2874712

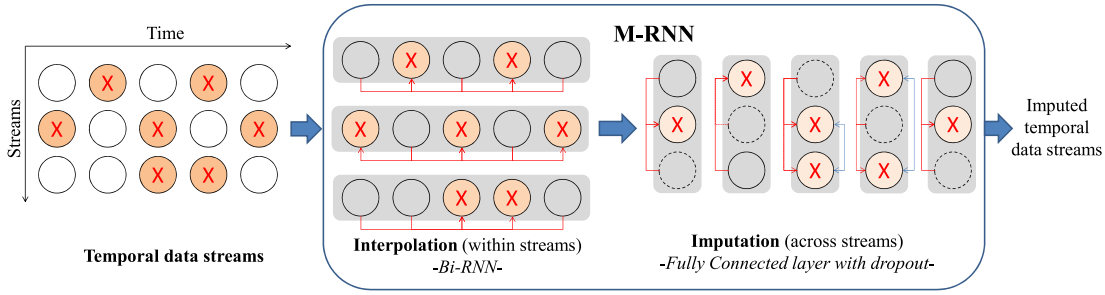

Fig. 1. Block diagram of missing data estimation process: X = missing measurements; red lines = connections between observed values and missing values in each layer; blue lines = connections between interpolated values; and dashed lines = dropout.

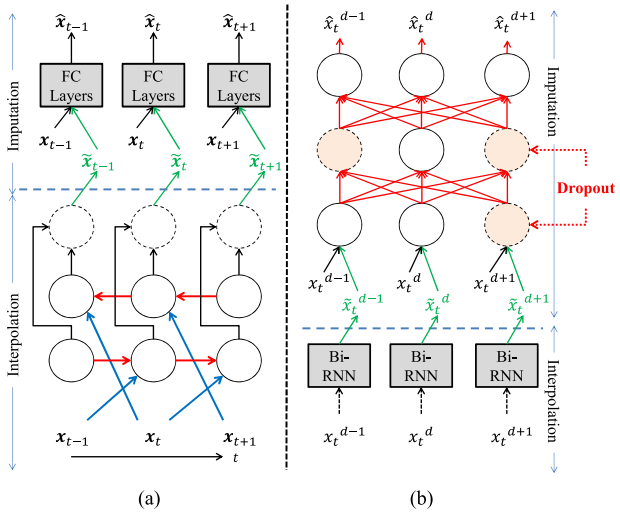

Fig. 2. M-RNN architecture. (a) Architecture in the time domain section. (b) Architecture in the feature domain section (Dropout is used for multiple imputations). Note that both $\tilde{\mathbf{x}}$ (the output of interpolation block) and $\mathbf{x}$ are inputs to the imputation block to construct $\hat{\mathbf{x}}$ (the output of imputation block).

within each data stream – in the *intra-stream directions*. An M-RNN also operates *across* different data streams – in the *inter-stream directions*. Unlike a Bi-RNN, the timing of inputs into the hidden layers of our M-RNN is lagged in the forward direction and advanced in the backward direction. As illustrated in Fig. 2, our M-RNN architecture exploits the 3-dimensional nature of the dataset.

An important aspect of medical data is that there is often enormous uncertainty in the measured data. As is well-known, although single imputation (SI) methods may yield the most plausible/most likely estimate for each missing data point [13], they do not capture the uncertainty in the imputed data [3]. Multiple imputation (MI) methods capture this uncertainty by sampling imputed values several times in order to form multiple complete imputed datasets, analyzing each imputed dataset separately and combining the results via Rubin's rule [3], [14], [15]. Capturing the uncertainty in the dataset is especially important in the medical setting, in which diagnostic, prognostic and treatment decisions must be made on the basis of the imputed values [16], [17]. In our setting, we use dropout [18] to produce multiple imputations; see Section IV-D.

To demonstrate the power of our method, we apply it to five different public real-world medical datasets: the MIMIC-III [19] dataset, the clinical deterioration dataset used in [20], the UNOS dataset for heart transplantation, the UNOS dataset for lung transplantation (both available at https://www.unos.org/data/), and the UK Biobank dataset [21]. We show that our method yields large and statistically significant improvements in estimation accuracy over previous methods, including interpolation methods such as [1], [2], imputation methods such as [3]–[6], RNN-based imputation methods such as [22]–[24] and matrix completion methods such as [7]. For the MIMIC-III and clinical deterioration datasets the patient measurements were made frequently (hourly basis), and our method provides Root Mean Squared Error (RMSE) improvement of more than 50% over all 11 benchmarks. For the UNOS heart and lung transplantation datasets and the Biobank dataset, the patient measurements were made much less frequently (yearly basis), but our method still provides RMSE improvement of more than 40% in most cases, and significant improvements in the other cases. We also show that this improvement in estimation yields (smaller) improvements in the predictions of outcomes (patients' future states). A number of experiments based on these same datasets show that the extent to which our method improves on outcomes depends on the method used for prediction, on the way in which our model is optimized in training, on the amount of data available (both in terms of the number of patients for whom we have data and on the amount of data available for each patient), and on the nature and extent of missing data. These results illustrate the important point that, as mentioned earlier, there are *many* reasons for imputing missing data [15], [25] – for the estimation of parameters (e.g., means or regression coefficients), for determination of confidence intervals and significance, as well as for prediction – and that no single method for imputing data can be expected to be superior *on all datasets* or *for all reasons*.

As [26] has emphasized, an extremely desirable aspect of any imputation method is that it be *congenial*; i.e., that it should produce imputed values in a manner that preserves the original relationships between features and labels. As we demonstrate using the complete Biobank dataset, our method is also more congenial than the best competing benchmarks; see Section VI-G.

II. Related Work

As we have noted, there are three standard and very widely-used methods for dealing with missing data: interpolation, imputation and matrix completion. Interpolation methods [1], [2]

attempt to reconstruct missing data by capturing the temporal relationship *within* each data stream but not the relationships *across* streams. Imputation methods [3]–[6] attempt to reconstruct missing data by capturing the synchronous relationships *across* data streams but not the temporal relationships *within* streams. Matrix completion methods [7]–[10] treat the data as static – ignoring the temporal aspect – or perfectly synchronized and assume a specific model of the data-generating process and/or the pattern of missing data.

There is also a substantial literature that uses Recurrent Neural Networks (RNNs) for prediction on the basis of time series with missing data. For example, [27] first replaces all the missing values with a mean value, then uses the feedback loop from the hidden states to update the imputed values and finally uses the reconstructed data streams as inputs to a standard RNN for prediction. [28] uses the Expectation-Maximization (EM) algorithm to impute the missing values and again uses the reconstructed data streams as inputs to a standard RNN for prediction. [29] uses a linear model to estimate missing values from the latest measurement and the hidden state within each stream followed by a standard RNN for prediction. In the first two of these papers, missing values are imputed by using only the synchronous relationships across data streams but not the temporal relationships within streams; in the third paper, missing values are interpolated by using only the temporal relationships within each stream but not on the relationships across streams.

A more recent literature extends these methods to deal with both missing data and irregularly sampled data [22]–[24], [30]. All of these papers use the sampling times to capture the informative missingness and time interval information to deal with irregular sampling, using the measurements, sampling information and time intervals as the inputs of an RNN. However, they differ in the replacements they use for missing values. [22], [23], [30] replace the missing values with 0, mean values or latest measurements – all of which are independent of either the intra-stream or inter-stream relationships or both. [24] imputes the missing values using only the most recent measurements, the mean value of each stream, and the time interval.

III. Problem Formulation

Our formulation and method are applicable to a wide variety of settings with missing data. However, for ease of exposition – and to facilitate the discussion of our application to medical datasets – it is convenient to adopt medical terminology throughout.

We consider a dataset consisting of N patients. For each patient, we have a multivariate time series data stream of length T (the length T and the other components of the dataset may depend on the patient n but for the moment we suppress the dependence on n) that consists of time stamps $\mathcal{S}$, measurements $\mathcal{X}$, and labels $\mathcal{Y}$, sampled from an (unknown) underlying distribution $\mathcal{F}$: $(\mathcal{S}, \mathcal{X}, \mathcal{Y}) \sim \mathcal{F}$.

For each t the *time stamp* $s_t \in \mathbb{R}$ represents the *actual time* at which the measurements x_t were taken. For convenience we normalize so that $s_1 = 0$ (so that we are measuring actual times for each patient beginning from the first observation for that patient); we assume actual times are strictly increasing: $s_{t+1} > s_t$ where $0 \leq t < T$. Note that the measurements may not be sampled regularly, so that the interval $s_{t+1} - s_t$ between successive measurements need not be constant.

There are D streams of measurements. We view each measurement as a real number, but it will typically be the case that not every stream is actually observed/measured at s_t. Hence we adopt notation in which the set of possible measurements at the t-th time stamp s_t is $\mathbb{R}_* = \mathbb{R} \cup \{*\}$. We interpret $x_t^d = *$ to mean that the stream d was not measured at s_t; otherwise $x_t^d \in \mathbb{R}$ is the actual measurement of stream d at s_t. (In computations with neural networks, we set $x_t^d = 0$ when the measurement x_t^d is missing. This guarantees that the missing measurement has no effect on the architecture.) For convenience, we scale all measurements to lie in the interval $[0, 1]$.

It is convenient to introduce some additional notation. For each t, define the index m_d^t to equal 0 if $x_t^d = *$ (i.e., the stream d was not measured at s_t) and to equal 1 if $x_t^d \in [0, 1]$ (the stream d was measured at s_t). We define δ_t^d to be the actual amount of time that has elapsed from s_t since the stream d was measured previously; δ_t^d can be defined by setting $\delta_1^d = 0$ and then proceeding recursively as follows:

$$\delta_t^d = \begin{cases} s_t - s_{t-1} + \delta_{t-1}^d & \text{if } t > 1, m_{t-1}^d = 0 \\ s_t - s_{t-1} & \text{if } t > 1, m_{t-1}^d = 1. \end{cases}$$

Write $\boldsymbol{\delta}_t$ for the vector of elapsed times at time stamp t and $\Delta = \{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, ..., \boldsymbol{\delta}_T\}$.

The label y_t represents the outcome realized at time stamp t (actual time s_t) such as discharge, clinical deterioration, death. $\mathcal{Y}$ is the vector of outcomes for this patient. Again, we scale so the labels (and eventually predictions) lie in the interval $[0, 1]$. Frequently the outcome is binary in which case $y_t = 0$ or $y_t = 1$.

The information available for a particular patient n is therefore a triple consisting of a sequence of time stamps, an array of measurements at each time stamp (with the above convention about missing measurements), and an array of labels at each time stamp. It is convenient to use functional notation to identify information about a particular patient, so $x_t^d(n)$ is the measurement of stream d at time stamp t for patient n, etc. The entire dataset consists of all the triples for all the patients $\mathcal{D} = \{(\mathcal{S}(n), \mathcal{X}(n), \mathcal{Y}(n)\}_{n=1}^N$.

Our objective is to find a function $\mathbf{f}$ that provides the best estimate of missing values; i.e., the estimate that minimizes the estimation loss. As is usually done, we measure loss as the squared error, so if x_t^d is an (unobserved) actual measurement (sampled from $\mathcal{F}$) and $\hat{x}_t^d = f_t^d(\mathcal{S}, \mathcal{X})$ is the estimate formed on the basis of observed data, then the squared loss for this particular measurement is $\mathcal{L}(\hat{x}_t^d, x_t^d) = (\hat{x}_t^d - x_t^d)^2$. Hence the formal optimization problem is to find a function $\mathbf{f}$ to solve:

$$\min_{\mathbf{f}} \mathbb{E}_{\mathcal{F}} \left[\sum_{t=1}^{T} \sum_{d=1}^{D} (1 - m_t^d) \mathcal{L}(\hat{x}_t^d, x_t^d) \right]$$

$$= \min_{\mathbf{f}} \mathbb{E}_{\mathcal{F}} \left[\sum_{t=1}^{T} \sum_{d=1}^{D} (1 - m_t^d) (f_t^d(\mathcal{S}, \mathcal{X}, \mathcal{Y}) - x_t^d)^2 \right]. \quad (1)$$

Note that the function $\mathbf{f}$ we seek depends on the particular d and t, and on the entire array of time stamps and measurements – but not on labels (which may not be observed). Also note that the formal problem asks to find an $\mathbf{f}$ that minimizes the loss with respect to the *true distribution*. Of course we do not observe the true distribution and cannot compute the true loss, so we will minimize the empirical loss.

IV. Multi-Directional Recurrent Neural Networks (M-RNN)

Suppose that stream d was not measured at time stamp t, so that $x_t^d = *$. We would like to form an estimate $\hat{x}_t^d$ of what the actual measurement would have been. As we have noted, familiar interpolation methods use only the measurements $x_{t'}^d$ of the fixed data stream d for other time stamps $t' \neq t$ (perhaps both before and after t) – but ignore the information contained in other data streams $d' \neq d$; familiar imputation methods use only the measurements $x_t^{d'}$ at the fixed time t for other data streams $d' \neq d$ – but ignores the information contained at other times $t' \neq t$. Because information is often correlated both *within and across* data streams, each of these familiar approaches throws away potentially useful information. Our approach forms an estimate $\hat{x}_t^d$ using measurements both *within the given data stream and across other data streams*. In principle, we could try to form the estimate $\hat{x}_t^d$ by using *all* the information in $\mathcal{D}$. However, this would be impractical because it would require learning a number of parameters that is on the order of the square of the number of data streams, and also because it would create a serious danger of over-fitting. Instead, we propose an efficient hierarchical learning framework using a novel RNN architecture that effectively allows us to capture the correlations both within streams and across streams. Our approach limits the number of parameters to be learned to be of the linear order of the number data streams and avoids over-fitting. See Fig. 1.

Our basic single-imputation M-RNN consists of 2 blocks: an Interpolation block and an Imputation block; see Fig. 2. (Our construction puts the Imputation block after the Interpolation block in order to use the outputs of the Interpolation block to improve the accuracy of the Imputation block; as we discuss later, it would not be useful to put the Interpolation block after the Imputation block.) To produce multiple imputations, we adjoin an additional dropout layer to the basic single-imputation M-RNN. (We defer the details until Section IV-D.) The entire source codes of M-RNN implementation are publicly available in the following link: http://github.com/jsyoon0823/MRNN/.

A. Error/Loss

As formalized above in Equation (1), our overall objective is to minimize the error that would be made in estimating missing measurements. Evidently, we cannot estimate the error of a measurement that was not made and hence is truly missing in the dataset. Instead we fix a measurement x_t^d that *was* made and is present in the dataset, form an estimate $\hat{x}_t^d$ for x_t^d using only the dataset with x_t^d removed (which we denote by $\mathcal{D} - x_t^d$), and then compute the error between the estimate $\hat{x}_t^d$ and the actual measurement x_t^d. As above, we use the squared error $(\hat{x}_t^d - x_t^d)^2$ as the loss for this particular estimate; as the total loss/error for the entire dataset $\mathcal{D}$ we use the *mean squared error* (MSE):

$$\mathcal{L}(\hat{\mathbf{x}}, \mathbf{x}) = \sum_{n=1}^{N} \left[\frac{\sum_{t=1}^{T_n} \sum_{d=1}^{D} m_t^d(n) \times (\hat{x}_t^d(n) - x_t^d(n))^2}{\sum_{t=1}^{T_n} \sum_{d=1}^{D} m_t^d(n)} \right]$$

Note that this is the empirical error, which only utilized actually achievable variables.

B. Interpolation Block

The Interpolation block constructs an interpolation function $\mathbf{\Phi}_d$ that operates *within* the d-th stream. To emphasize that the output $\tilde{x}_t^d$ of the interpolation block depends only on the d-th data stream with x_t^d removed, we write $\tilde{x}_t^d = \mathbf{\Phi}_d(\mathcal{D}^d - x_t^d)$, where $\mathcal{D}^d$ is the d-th stream of the entire dataset $\mathcal{D}$, and the notation $\mathcal{D}^d - x_t^d$ emphasizes that we have removed x_t^d. It is important to keep in mind that the construction uses only the data from stream d, not the data from other streams. We construct $\mathbf{\Phi}_d$ using a bi-directional recurrent neural network (Bi-RNN). However, unlike a conventional Bi-RNN [12], the timing of inputs into the hidden layer is lagged in the forward direction and advanced in the backward direction: at t, inputs of forward hidden states come from $t-1$ and inputs of backward hidden states come from $t+1$. (This procedure ensures that the actual value x_t^d is not used in the estimation of $\tilde{x}_t^d$.) Note that each data stream uses its own Bi-RNN architecture ($\mathbf{\Phi}_d$). The inputs of the Interpolation block consist of the feature vector x, the mask vector m, and the elapsed time vector δ (defined in Section III, and extracted from the original data streams). If we write $\mathbf{z}_t^d = [x_t^d, m_t^d, \delta_t^d]$ (note that we explicitly include δ_t^d as the additional input to deal with the irregular sampling procedures) then a more mathematical description is:

$$\begin{aligned}
\tilde{x}_t^d &= g(U^d[\overrightarrow{\mathbf{h}}_t^d; \overleftarrow{\mathbf{h}}_t^d] + \mathbf{c}_o^d) = g(\overrightarrow{U}^d \overrightarrow{\mathbf{h}}_t^d + \overleftarrow{U}^d \overleftarrow{\mathbf{h}}_t^d + \mathbf{c}_o^d) \\
\overrightarrow{\mathbf{h}}_t^d &= (1 - \overrightarrow{\mathbf{u}}_t^d) \circ \overrightarrow{\mathbf{h}}_{t-1}^d \\
&\quad + \overrightarrow{\mathbf{u}}_t^d \circ q(\overrightarrow{W}_h^d(\overrightarrow{\mathbf{r}}_t^d \circ \overrightarrow{\mathbf{h}}_{t-1}^d) + \overrightarrow{V}_h^d \mathbf{z}_{t-1}^d + \overrightarrow{\mathbf{c}}_h^d) \\
\overrightarrow{\mathbf{u}}_t^d &= \gamma(\overrightarrow{W}_u^d \overrightarrow{\mathbf{h}}_{t-1}^d + \overrightarrow{V}_u^d \mathbf{z}_{t-1}^d + \overrightarrow{\mathbf{c}}_u^d) \\
\overrightarrow{\mathbf{r}}_t^d &= \gamma(\overrightarrow{W}_r^d \overrightarrow{\mathbf{h}}_{t-1}^d + \overrightarrow{V}_r^d \mathbf{z}_{t-1}^d + \overrightarrow{\mathbf{c}}_r^d) \\
\overleftarrow{\mathbf{h}}_t^d &= (1 - \overleftarrow{\mathbf{u}}_t^d) \circ \overleftarrow{\mathbf{h}}_{t+1}^d \\
&\quad + \overleftarrow{\mathbf{u}}_t^d \circ q(\overleftarrow{W}_h^d(\overleftarrow{\mathbf{r}}_t^d \circ \overleftarrow{\mathbf{h}}_{t+1}^d) + \overleftarrow{V}_h^d \mathbf{z}_{t+1}^d + \overleftarrow{\mathbf{c}}_h^d) \\
\overleftarrow{\mathbf{u}}_t^d &= \gamma(\overleftarrow{W}_u^d \overleftarrow{\mathbf{h}}_{t+1}^d + \overleftarrow{V}_u^d \mathbf{z}_{t+1}^d + \overleftarrow{\mathbf{c}}_u^d) \\
\overleftarrow{\mathbf{r}}_t^d &= \gamma(\overleftarrow{W}_r^d \overleftarrow{\mathbf{h}}_{t+1}^d + \overleftarrow{V}_r^d \mathbf{z}_{t+1}^d + \overleftarrow{\mathbf{c}}_r^d)
\end{aligned}$$

(As can be seen from these equations, we are using a bidirectional GRU.) Here, g, q, γ are activation functions. (In principle, any activation functions, such as Rectified Linear Unit (ReLU), tanh, etc., could be used; here we use ReLU.) The arrows indicate forward/backward direction and $\circ$ indicates element-wise multiplication. As we have emphasized, in this interpolation block, we are only using/capturing the temporal correlation *within each data stream*. In particular, the parameters for each

data stream are learned separately, and the number of parameters that must be learned is linear in the number of streams D. Note that $\tilde{x}_t^d$ is not the final output of our M-RNN architecture and is not necessarily an estimate of x_t^d.

C. Imputation Block

The Imputation blocks constructs an imputation function $\mathbf{\Psi}$ that operates *across* streams. To again emphasize that the estimate $\hat{x}_t^d$ for x_t^d depends on the data with x_t^d removed, we write $\hat{x}_t^d = \mathbf{\Psi}(\mathcal{D}_t - x_t^d)$; again, keep in mind that now we are using only data at time stamp s_t, not data from other time stamps. ($\mathcal{D}_t$ represents the t-th time stamp of the entire dataset $\mathcal{D}$.) We construct the function $\mathbf{\Psi}$ to be independent of t, so we use fully connected layers; see the Imputation component of Fig 2. If we write $\mathbf{z}_t = [\tilde{\mathbf{x}}_t, \mathbf{m}_t]$ then a more mathematical description is:

$$\hat{\mathbf{x}}_t = \sigma(W\mathbf{h}_t + \alpha) \quad \mathbf{h}_t = \phi(U\mathbf{x}_t + V\mathbf{z}_t + \beta)$$

where σ, ϕ are activation functions. It is important to keep in mind that the diagonal entries of U are zero and the off-diagonal entries of W are zero (i.e., W is diagonal) so that we do not use x_t^d in the estimation of $\hat{x}_t^d$.

We learn the functions $\{\mathbf{\Phi}_d\}_{d=1}^D$ and $\mathbf{\Psi}$ *jointly* using the stacked networks of Bi-RNN and Fully Connected (FC) layers, using MSE as the objective function.

$$\mathbf{\Psi}^*, \{\mathbf{\Phi}_d^*\}_{d=1}^D = \arg\min_{\mathbf{\Phi}_d, \mathbf{\Psi}} \mathcal{L}\left(\left\{\mathbf{\Psi}\left(\left\{x_t^d, \mathbf{\Phi}_d\left(\{x_\tau^d, m_\tau^d, \delta_\tau^d\}_{\tau=1}^T\right), m_t^d\right\}_{d=1}^D\right)\right\}_{t=1}^T, \mathbf{x}\right) \tag{2}$$

Note that $\tilde{x}_t$ is the output of the interpolation block, and $\hat{x}_t$ is the final output of the entire M-RNN architecture.

D. Multiple Imputations

It is well-understood that to account for the uncertainty in estimating missing values, it is useful to produce multiple estimates and generate multiple imputed datasets. These multiple imputed datasets can each be analyzed using standard methods and the results can be combined using Rubin's rule [3]. In our case, we generate multiple imputed datasets using the well-known Dropout [18] approach driven from the Bayesian Neural Network framework [31]: we randomly select neurons in the fully connected layers and delete those neurons and all their connections. (The dropout probability $p \in (0, 1)$ is a hyperparameter to be chosen; the neurons to be dropped are chosen according to the Bernoulli distribution with parameter p.) In the training stage, we conduct joint optimization (Equation (2)) using the dropout process. We then generate multiple outputs $\mathbf{o}_t$ by sampling different dropout vectors $\mathbf{R}$ from the Bernoulli distributions. This yields multiple imputations (MI). (To construct a single imputation (SI) we proceed in precisely the same way but set the dropout probability to 0. For comparisons, we normalize the final output by multiplying by p.)

E. Overall Structure and Computation Complexity

We refer to the entire structure above as a *Multi-directional Recurrent Neural Network (M-RNN)*. We use the notations M-RNN (MI) and M-RNN (SI) to clarify whether we are producing multiple or single imputations. The entire training times for both M-RNN (MI) and M-RNN (SI) are less than 2 hours for all 6 datasets (described in Section V) on a computer with Intel Core i7-4770 (3.4 GHz) CPU with 32 GB RAM. With the same machine, the entire training time for Multiple Imputation with Chained Equation (MICE) [5] is around 11 hours for all 6 datasets.

V. Datasets

We use five datasets, the characteristics of which are summarized in Table I: more detailed descriptions are below.

A. MIMIC-III

The dataset MIMIC-III [19] contains data for patients monitored in intensive care units (ICUs) of various hospitals. Within the entire MIMIC-III dataset, we only used the data for the 23,160 patients whose measurements are recorded by Metavision (post 2008). We used the 20 vital signs (e.g., heart rate, respiratory rate, blood pressures, etc.) and 20 lab tests (e.g., creatinine, chloride, etc.) whose missing rates are the least. For these patients we have 40 physiological data streams in all. Vital signs were sampled roughly every hour; lab tests were sampled roughly every 12 hours. Each patient was followed until either death (1,320 patients (5.7%)) or discharge from ICU (21,840 patients (94.3%)). Note that because lab tests are sampled only 1/12 as often as vital signs, in effect 11/12 of lab test data is missing – even if every lab test for every patient was actually conducted and recorded. For the purpose of prediction, we take the goal as predicting at each time t whether the patient will die within the next 24 hours. Hence we assign the label $y_t = 1$ if the patient actually died within the 24 hrs following the time s_t and $y_t = 0$ otherwise.

B. Deterioration

The dataset described in [20] provides records for a cohort of 6,094 patients who were followed for potential clinical deterioration while hospitalized. Patients were monitored for 28 vital signs (heart rate, blood pressure, etc.) and 10 lab tests (creatinine, hemoglobin, etc.) so there are 38 physiological data streams in all. Vital signs were sampled roughly every 4 hours; lab tests were sampled roughly every 24 hours. Each patient was followed until either admission to ICU (306 patients (5.0%)) or discharge from hospital (5,788 patients (95.0%)). Again, because lab tests are sampled only 1/6 as often as vital signs, in effect 5/6 of lab test data is missing – even if every lab test for every patient was actually conducted and recorded. For the purpose of prediction, we take the goal as predicting at each time stamp s_t whether the patient will be admitted to the ICU (experienced clinical deterioration) within the next 24 hours. Hence we assign the label $y_t = 1$ if the patient was admitted to the ICU within the following 24 hours and $y_t = 0$ otherwise.

C. UNOS-Heart and UNOS-Lung

The UNOS (United Network for Organ Transplantation) dataset (available at https://www.unos.org/data/) provides

TABLE I
SUMMARY OF THE DATASETS (CONT: CONTINUOUS, CAT: CATEGORICAL, AVG: AVERAGE)

Datasets	MIMIC-III	Deterioration	UNOS-Heart	UNOS-Lung	Biobank
Number of Patients	23,160	6,094	69,205	32,986	3,902
Number of Dimensions (Cont, Cat)	40 (31, 9)	38 (16, 22)	34 (10, 24)	34 (10, 24)	113 (67, 46)
Label $(y = 1)$	1,320 (5.7%)	306 (5.3%)	4,844 (7.0%)	2,276 (6.9%)	195 (5.0%)
Avg number of samples	24.3	34.3	6.2	4.0	3.0
Avg missing rate	75.0%	61.4%	59.1%	58.5%	0.0%
Avg Measurement freq.	1 hr / 12 hrs	4 hrs / 24 hrs	1 year	1 year	2.3 years
Avg Correlation within streams	0.4122	0.3436	0.1213	0.1157	0.2424
Avg Correlation across streams	0.3127	0.3454	0.0875	0.0897	0.0506

yearly follow-up information for the entire U.S. cohort of 69,205 patients who received heart transplants and 32,986 patients who received lung transplants during the period 1985–2015. We view patients in the dataset as described by a total of 34 clinical features. (In fact, a total of 232 features are recorded in the UNOS dataset, but many features are not recorded for most patients. We therefore excluded the 198 features for which missing rates were higher than 80%; this is in keeping with standard medical statistical practice.) For each patient, a number of yearly follow-ups are recorded; the smallest number of yearly follow-ups is 1, the largest is 26; the median number of follow-ups for heart transplantation is 6 and the median number of follow-ups for lung transplantation is 4. (In the main text, we focus entirely on features of the patient, ignoring features of the donor. We do this for two reasons: (i) the features of the patient change over time but the features of the donor do not (because the donor is dead); (ii) the relevant features of the donor appear to be largely captured in the time series measurements of the patient. However, as we show in the Appendix, taking features of the donor into account seems to make little difference for either imputation or prediction.) For the purpose of prediction, we take the goal in each case as predicting at each follow up time s_t whether the patient will be dead one year later. Hence we assign the label $y_t = 1$ if the patient actually died within the following year and $y_t = 0$ otherwise.

D. Biobank

We used the UK Biobank dataset gathered from 21 assessment centers across England, Wales, and Scotland using standardized procedures from 2007 to 2014. (The UK Biobank protocol is available online.) UK Biobank recorded various patient information including baseline measurements, physical measurements, and evaluations of biological samples. For this paper, we excluded all the variables that were missing for more than 80% of the participants and all the static measurements and only used the 113 longitudinal measurements. Of the 4,096 total patients, we used only the data for the 3,902 patients who missed no admissions to assessment centers (and hence were assessed the maximum number of times, which was three) and for whom there are no missing measurements of these 113 variables; thus we have a complete dataset. A complete dataset is required for the congeniality experiments described in Section VI-G, and we can extract a complete dataset from the UK Biobank dataset. In the other datasets we consider, there is no single patient for whom the information is complete, and so a complete dataset cannot be extracted and these datasets cannot be used for congeniality experiments. For the purpose of prediction we take the goal to be the correct prediction of diabetes, so we assign the label $y_t = 1$ if diabetes is diagnosed at t and $y_t = 0$ otherwise.

VI. RESULTS AND DISCUSSIONS

A. Imputation Accuracy on the Given Datasets

We begin by comparing the performance of our method (using both multiple imputations and single imputation) on the given datasets against 11 benchmarks with respect to the accuracy of imputing missing values. The benchmarks against which we compare are: the algorithms proposed in [22]–[24]; Spline and Cubic Interpolation [1]; MICE [5]; MissForest [6]; EM [4]; the matrix completion algorithm of [7]; the Auto-Encoder algorithm proposed in [32]; and the Markov chain Monte Carlo (MCMC) method [33]. (The details of the implementations for the various benchmarks are presented in the Appendix.) As is common, we use root mean squared error (RMSE) as the measure of performance. In each experiment, we use 5-fold cross-validation. Table II shows the mean RSME for our method and benchmarks, and the percentage improvement of RMSE for M-RNN (MI) over the benchmarks. (Note that we are unable to provide results for the EM algorithm on the UNOS-Heart and UNOS-Lung datasets because – at least for the implementation we use – the EM algorithm requires at least one patient for whom data is complete, and the UNOS-Heart and UNOS-Lung datasets do not contain any such patient.)

As can be seen in Table II, M-RNN achieves better performance (smaller RMSE) than all of the benchmarks on all of the datasets (for all comparisons are possible). With a single exception (the comparison with MissForest on the UNOS-Lung dataset) the performance improvements are statistically significant at the 95% level (i.e., $p < 0.05$), and many of the improvements are very large. For instance, for the Deterioration dataset, M-RNN using multiple imputations achieves RMSE of 0.0105 (95% CI: 0.0071–0.0138), while the *best* benchmark (Spline interpolation) achieves RMSE of 0.0215 (95% CI: 0.0178–0.0255); this represents an improvement of 51.2%.

The performance comparisons across datasets are revealing, if not necessarily surprising. The interpolation benchmarks

TABLE II
PERFORMANCE COMPARISON FOR MISSING DATA ESTIMATION

Category	Algorithm	Mean RMSE (% Gain of M-RNN (Multiple Imputations))				
		MIMIC-III	Deterioration	UNOS-Heart	UNOS-Lung	Biobank
M-RNN	**M-RNN (MI)**	**0.0141 (-)**	**0.0105 (-)**	**0.0479 (-)**	**0.0606 (-)**	**0.0637 (-)**
	M-RNN (SI)	**0.0144 (-)**	**0.0108 (-)**	**0.0477 (-)**	**0.0609 (-)**	**0.0629 (-)**
RNN-based	[23]	0.0337 (58.2%)	0.0258 (59.3%)	0.1352 (64.6%)	0.1343 (54.9%)	0.0812 (21.6%)
	[24]	0.0295 (52.2%)	0.0241 (56.4%)	0.1179 (59.4%)	0.1264 (52.1%)	0.0801 (20.5%)
	[25]	0.0292 (51.7%)	0.0233 (54.9%)	0.1057 (54.7%)	0.1172 (48.3%)	0.0778 (18.1%)
Interpolation	Spline	0.0735 (80.8%)	0.0215 (51.2%)	0.1102 (56.5%)	0.1199 (49.5%)	0.0845 (24.6%)
	Cubic	0.0279 (49.5%)	0.0223 (52.9%)	0.1072 (55.3%)	0.1177 (48.5%)	0.0887 (28.2%)
Imputation	MICE	0.0611 (76.9%)	0.0319 (67.1%)	0.1147 (58.2%)	0.1151 (47.4%)	0.0915 (30.4%)
	MissForest	0.0293 (51.9%)	0.0264 (60.2%)	0.0489 (2.0%)	0.0652 (7.1%)	0.0892 (28.6%)
	EM	0.0467 (69.8%)	0.0355 (70.4%)	-	-	0.0978 (34.9%)
Others	Matrix Completion	0.0311 (54.7%)	0.0264 (60.2%)	0.0974 (50.8%)	0.0942 (35.7%)	0.0886 (28.1%)
	Auto-encoder	0.0412 (66.0%)	0.0309 (65.0%)	0.0589 (18.7%)	0.0712 (14.9%)	0.0805 (20.9%)
	MCMC	0.0437 (67.7%)	0.0364 (71.2%)	0.1091 (56.1%)	0.1124 (46.1%)	0.0936 (31.9%)

(such as Spline, Cubic and RNN-based methods) work best on datasets, such as MIMIC-III and Deterioration, for which measurements were more frequent (and more highly correlated within each stream (see Table I)); the imputation benchmarks work best on datasets, such as UNOS-Heart and UNOS-Lung, for which measurements were less frequent but for which there were many streams of data (many dimensions). The improvement of our method over all benchmarks is larger for the MIMIC-III and Deterioration datasets because those datasets have many streams of frequently sampled data, so that our method gains a great deal from exploiting both the correlations within each data stream and the correlations across data streams. Conversely, the improvement of our method is smaller for the UNOS-Heart and UNOS-Lung datasets, because streams in those datasets are infrequently sampled to that there is less to be gained by exploiting the correlations within data streams. (The performances of the benchmarks for the Biobank dataset are mixed, and don't quite fit this same pattern, perhaps because Biobank is a small dataset (less than 4,000 patients with complete temporal data streams)).

1) Multiple Imputations vs. Single Imputation: As we have noted, the purpose of conducting multiple imputations is to reduce uncertainty/shrink confidence intervals (rather than to improve average performance). As is illustrated in the box-plot in Fig. 3(a) which shows the comparison of M-RNN with multiple imputations and M-RNN with a single imputation against the best benchmark (Cubic interpolation) on the MIMIC-III dataset, our multiple imputations do achieve this purpose. (For discussion of Fig. 3(b), which illustrates the corresponding reduction in uncertainty for prediction, see below).

2) Combining Models of Interpolation and Imputation: As we have already discussed, standard interpolation algorithms cannot capture the patterns across streams and standard imputation algorithms cannot capture the patterns within the streams. However, it is possible to combine a standard interpolation algorithm and standard imputation algorithm in an attempt to capture *both patterns*, and it might be thought that such a combination would be a fairer benchmark against which to compare

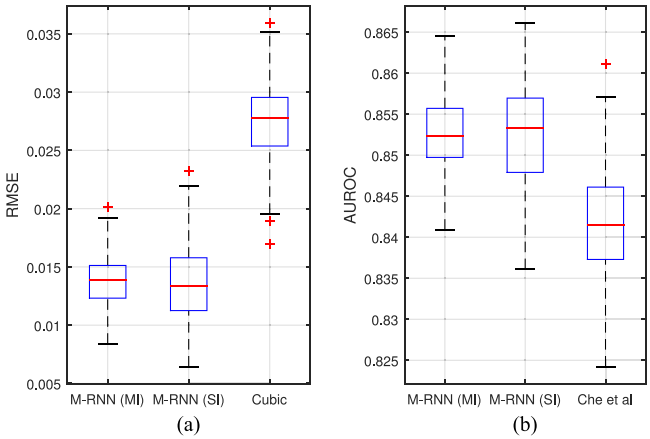

Fig. 3. Box-plot comparisons between M-RNN (MI), M-RNN (SI) and the best benchmark. (a) RMSE comparison using MIMIC-III dataset. (b) AUROC comparison using MIMIC-III dataset. Red crosses represents outliers.

our method. To put this idea to the test, we create a family of "joint algorithms" by first using an interpolation algorithm to interpolate the missing values, and then using the interpolated values as the initial points of an imputation algorithm to provide final imputed values. For this exercise, we use two standard interpolation methods (Cubic and Spline), and two standard imputation methods (MICE and MissForest) so that we have 4 interpolation-imputation combination models: Cubic + MICE, Cubic + MissForest, Spline + MICE, and Spline + MissForest.

As Table III shows, however, the performances of these interpolation-imputation combination models are very similar to those of the performance of the simple imputation model that is used. Indeed, the largest RMSE performance improvement is only 0.0018. The reason for this is that imputation methods use algorithms that operate iteratively until they converge, so that their performance is rather robust to the initialization. Hence, although the interpolation part of the joint models captures some of the inter-stream information, the iterative imputation part ignores most of what is captured.

TABLE III
PERFORMANCE COMPARISON FOR JOINT INTERPOLATION/IMPUTATION ALGORITHMS

Algorithm	Mean RMSE (% Gain from Imputation Algorithm)				
	MIMIC-III	Deterioration	UNOS-Heart	UNOS-Lung	Biobank
Spline + MICE	0.0602 (1.5%)	0.0320 (-0.3%)	0.1141 (0.5%)	0.1133 (1.7%)	0.0895 (2.2%)
Spline + MissForest	0.0291 (0.7%)	0.0259 (1.9%)	0.0491 (-0.4%)	0.0641 (1.4%)	0.0879 (4.1%)
Cubic + MICE	0.0605 (1.0%)	0.0315 (1.3%)	0.1137 (0.9%)	0.1138 (1.1%)	0.0901 (1.6%)
Cubic + MissForest	0.0289 (1.4%)	0.0261 (1.1%)	0.0493 (-0.8%)	0.0643 (1.4%)	0.0887 (3.2%)

TABLE IV
SOURCE OF GAIN OF M-RNN
(PERFORMANCE DEGRADATION FROM ORIGINAL M-RNN)

Datasets	M-RNN (Mean RMSE; % Gain)		
	Only Interp	Only Impute	Interp + Impute
MIMIC-III	0.0191 (26.2 %)	0.0312 (54.8 %)	0.0141 (-)
Deterioration	0.0133 (21.1 %)	0.0295 (64.4 %)	0.0105 (-)
UNOS-Heart	0.0897 (46.6 %)	0.0531 (9.8 %)	0.0479 (-)
UNOS-Lung	0.0998 (39.3 %)	0.0734 (17.4 %)	0.0606 (-)
Biobank	0.0794 (19.8 %)	0.0778 (18.1 %)	0.0637 (-)

B. Source of Gains

As illustrated in Fig. 2, our M-RNN consists of an Interpolation block and an Imputation block. To understand where the gains of our approach come from, we compare the performance of that is achieved when we use only the Interpolation block or only the Imputation block; the results are shown in Table IV.

The Interpolation block is intended to exploit the correlations within each data stream and the Imputation block is intended to exploit the correlations across streams, so it is to be expected that the largest gains of our M-RNN method should come from the Interpolation block for the datasets (MIMIC-III and Deterioration) which are frequently sampled and have large temporal correlations, and should come from the Imputation block for the datasets (UNOS-Heart and UNOS-Lung) which are infrequently sampled but have many data streams. As shown in Table IV, these intuitions are indeed supported by the experiments.

C. Additional Experiments

The experiments we have described above demonstrate that our method significantly outperforms a wide variety of benchmarks for the imputation of missing data on five somewhat representative datasets. However it is natural to ask how our method would compare in other circumstances. To get some understanding of this, we conducted four sets of experiments based on the MIMIC-III dataset: increasing the amount of missing data, reducing the number of data streams, reducing the number of samples, and reducing the number of measurements per patient. Within each set of experiments, we conducted 10 trials for each value of the parameter being studied (e.g., amount of missing data), and we report the average over these 10 trials. The results are described below and in Fig. 4. Although the results of these experiments are extremely suggestive, we caution the reader that these are only a specific set of experiments and that one should be careful about drawing general conclusions.

1) Amount of Missing Data (Fig. 4(a)): To evaluate the performance of M-RNN in comparison to benchmarks in settings with more missing data, we constructed sub-samples of the MIMIC-III dataset by randomly removing 10%, 20%, 30%, 40%, 50% of the actual data and carrying out the same estimation exercise as above on the smaller datasets that remain. (Recall that in the original MIMIC-III dataset, 75% of the data is already missing; hence removing 50% of the data present leads to an artificial dataset in which 87.5% of the data is missing.) The graph in Fig. 4(a) shows the performance of M-RNN against the *best* benchmarks of each type for these smaller datasets. As can be seen, M-RNN continues to substantially outperform the benchmarks. Note that as the amount of missing data increases the improvement of M-RNN over the imputation benchmark(s) increases, but the improvement over the interpolation benchmarks decreases.

2) Number of Data Streams (Fig. 4(b)): As we have noted, typical medical datasets contain many data streams (many feature dimensions). To evaluate the performance of M-RNN in comparison to benchmarks in settings with fewer data streams, we conducted experiments in which we reduced the number of data streams (feature dimensions) of MIMIC-III. In the original MIMIC-III dataset the number of data streams is $D = 40$; we conducted experiments with $D = 3, 5, 7, 10, 15, 20$ data streams. (In each case, we conducted 10 trials in which we selected data streams at random; we report the average of these 10 trials.) As expected, the performance of M-RNN degrades when there are fewer data streams, but as Fig. 4(b) shows, M-RNN still outperforms the benchmarks. (Note that interpolation methods are insensitive to the number of data streams because they operate only *within each data stream separately*.)

3) Number of Samples (Fig. 4(c)): The original MIMIC-III dataset has $N = 23,160$ samples (patients). To understand the performance of M-RNN in comparison to benchmarks in settings with fewer samples, we conducted experiments in which we used only subsets of all patients (samples) of sizes $N = 500, 1000, 2000, 4000, 8000, 16000$. Because M-RNN has to learn many parameters, it should come as no surprise that, as Fig. 4(c) shows, the performance of M-RNN degrades badly – and indeed is worse than that of (some) other benchmarks – when the number of samples is too small, but M-RNN outperforms all the benchmarks as soon as the number of training

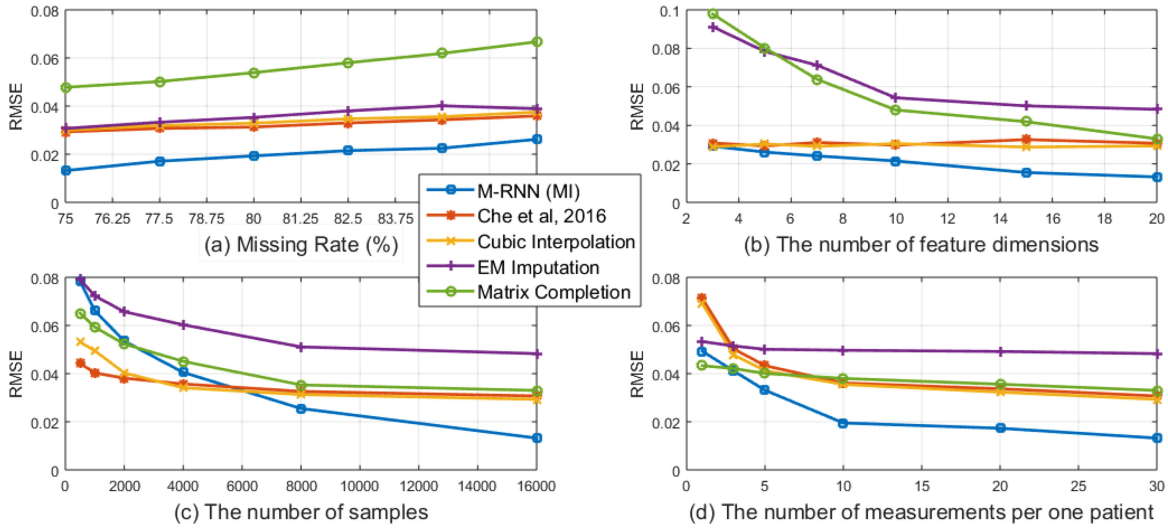

Fig. 4. Imputation accuracy for the MIMIC-III dataset with various settings. (a) Additional data missing at random. (b) Feature dimensions chosen at random. (c) Samples chosen at random. (d) Measurements chosen at random.

samples exceeds $N = 7,000$. (However, one should not necessarily take the figure $N = 7,000$ as representing a cut-off below which M-RNN should not be applied, because M-RNN outperforms the benchmarks on the Deterioration and Biobank datasets, which contain only 6,094 samples and 3,902 samples, respectively.)

4) Number of Measurements Per Patient (Fig. 4(d)): We have already noted that, in our datasets, MIMIC-III and Deterioration have many (relatively frequent) measurements per patient, while the other datasets have only a few (and infrequent) measurements per patient and that this leads to differences in performance of M-RNN. To further explore this effect, we created subsets of the MIMIC-III dataset with $T = 1, 3, 5, 10, 20, 30$ measurements per patient. As might be expected, and as Fig. 4(d) shows, having fewer measurements per patient degrades the performance of interpolation-based algorithms but has little effect on pure imputation-based methods; the performance of M-RNN is also degraded, but to a much lesser extent.

D. Prediction Accuracy

As we have noted, there are many reasons for imputing missing data; one such is to improve predictive performance. We therefore compare our method against the same 11 benchmarks with respect to the accuracy of predicting labels. (See the description of the datasets in Section V for labeling in each case.) For this purpose, we use Area Under the Receiver Operating Characteristic Curve (AUROC) as the measure of performance. (AUROC is defined as the area under the receiver operating characteristic curve, which is the graph of sensitivity (true positive rate) vs. 1-specificity (false positive rate).) To be fair to all methods of imputing missing values, we use the same predictive model (a simple 1-layer RNN) in all cases.

1) Prediction Accuracy on the Original Datasets: In this subsection, we evaluate the effects of the imputations on the prediction of labels (outcomes), which in the cases at hand correspond to prognoses.

Table V shows the mean and percentage performance gain of M-RNN (MI) in comparison with the benchmarks on all the datasets. M-RNN – which we have already shown to achieve the best imputation accuracy – also yields the best prediction accuracy. However, even in cases where the improvement in imputation accuracy is large and statistically significant, the improvements in prediction accuracy are sometimes smaller and not always statistically significant. For instance, on the Deterioration dataset, the AUROC of M-RNN (MI) is 0.7779 (95% CI: 0.7678–0.7868); the best benchmark is [24] with AUROC of 0.7593 (95% CI: 0.7478–0.7702). Similarly, on the UNOS-Heart dataset, the AUROC of M-RNN (MI) is 0.6855 (95% CI: 0.6781–0.6913); the best benchmark is MissForest, with AUROC of 0.6740 (95% CI: 0.6651–0.6817).

It should be noted that, by using mean squared error as the loss function, we have deliberately optimized M-RNN for *imputation accuracy*. If we want to optimize M-RNN for *prediction accuracy* we might do better by using a different loss function, such as cross-entropy. In Table XI of the Appendix we report an experiment in which we have done precisely that; the short summary is that optimizing for prediction accuracy does in fact improve the predictive performance of M-RNN but the improvement is marginal.

The Appendix also reports other experiments that help further our understanding of the M-RNN algorithm. Table VIII demonstrates that using a different predictive model (random forest, logistic regression or Xgboost [34], rather than a 1-layer RNN) for prediction after imputation leads to results similar to those obtained above. Tables IX and X demonstrate that accounting for donor features in the UNOS datasets makes little difference.

E. Prediction Accuracy With Various Missing Rates

As discussed above, we carried out experiments with increased rates of missing data in order to understand the implications for the accuracy of imputation. We also carried out

TABLE V
PERFORMANCE COMPARISON FOR PATIENT STATE PREDICTION WITH A 1-LAYER RNN (PERFORMANCE GAIN IS COMPUTED IN TERMS OF 1-AUROC)

Category	Algorithm	AUROC (Gain of M-RNN (MI) as % improvement in 1-AUROC)				
		MIMIC-III	Deterioration	UNOS-Heart	UNOS-Lung	Biobank
M-RNN	**M-RNN (MI)**	**0.8531 (-)**	**0.7779 (-)**	**0.6855 (-)**	**0.6762 (-)**	**0.8955 (-)**
	M-RNN (SI)	**0.8530 (-)**	**0.7783 (-)**	**0.6858 (-)**	**0.6759 (-)**	**0.8948 (-)**
RNN-based	[23]	0.8381 (9.3%)	0.7558 (9.0%)	0.6505 (10.0%)	0.6557 (6.0%)	0.8802 (12.8%)
	[24]	0.8402 (8.1%)	0.7551 (9.3%)	0.6574 (8.2%)	0.6561 (5.8%)	0.8748 (16.5%)
	[25]	0.8410 (7.6%)	0.7593 (7.7%)	0.6583 (8.0%)	0.6520 (7.0%)	0.8826 (11.0%)
Interpolation	Spline	0.8407 (7.8%)	0.7542 (9.6%)	0.6477 (10.7%)	0.6520 (7.0%)	0.8731 (17.7%)
	Cubic	0.8397 (8.4%)	0.7569 (8.6%)	0.6468 (11.0%)	0.6517 (7.0%)	0.8643 (23.0%)
Imputation	MICE	0.8377 (9.5%)	0.7571 (8.6%)	0.6397 (12.7%)	0.6509 (7.2%)	0.8850 (9.1%)
	MissForest	0.8368 (10.0%)	0.7578 (8.3%)	0.6740 (3.5%)	0.6587 (5.1%)	0.8767 (15.2%)
	EM	0.8312 (13.0%)	0.7531 (10.0%)	-	-	0.8794 (13.3%)
Others	Matrix Completion	0.8401 (8.1%)	0.7551 (9.3%)	0.6712 (4.3%)	0.6579 (5.3%)	0.8865 (7.9%)
	Auto-encoder	0.8399 (8.2%)	0.7488 (11.6%)	0.6633 (6.6%)	0.6574 (5.5%)	0.8785 (14.0%)
	MCMC	0.8298 (13.7%)	0.7512 (10.7%)	0.6417 (12.2%)	0.6512 (7.2%)	0.8667 (21.6%)

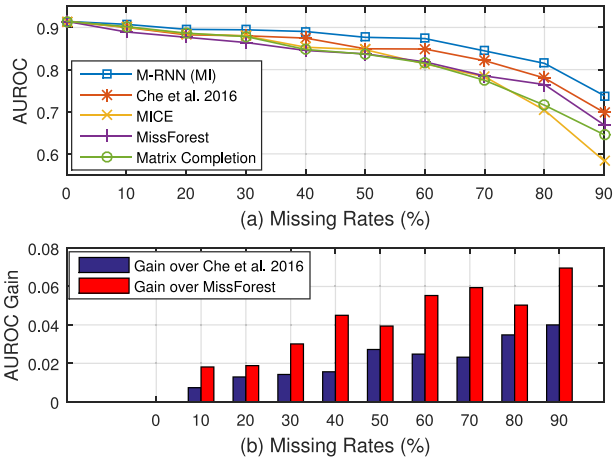

Fig. 5. (a) The AUROC performance with various missing rates. (b) The AUROC gain over the two most competitive benchmarks.

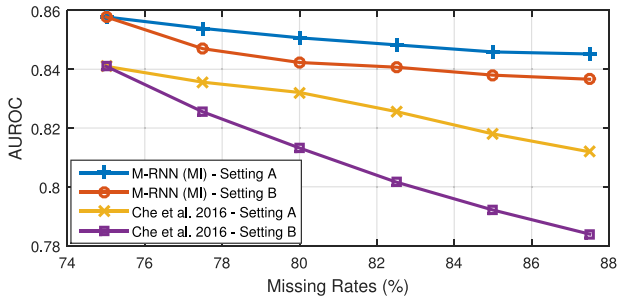

Fig. 6. AUROC comparisons in settings A and B using MIMIC-III dataset.

experiments with increased rates of missing data in order to understand the implications for the accuracy of prediction. To explore the predictive performance for a wide range of missing rates (from 0% to 90%), we begin with the Biobank dataset, which is a complete dataset. We randomly remove 10% to 90% of the measurements (with increments of 10%) to create multiple datasets with different missing rates. (In each case we use 80% of the data for training and 20% for testing.) As before, we use M-RNN and various benchmarks for imputing missing data and a 1-layer RNN as the predictive model. (In this setting we are predicting a clinical diagnosis of diabetes.) In each setting, we conducted 10 trials, and report the performance in terms of AUROC.

Fig. 5(a) illustrates the impact (in terms of AUROC) of increasing amounts of missing data for M-RNN and various benchmarks. As Fig. 5(a) shows, for M-RNN and all benchmarks, the prediction performance decreases as the amount of missing data increases. However, as Fig. 5(b) shows, M-RNN continues to outperform the benchmarks; indeed, the performance gap between M-RNN and the benchmarks widens when more data is missing. That is: the importance of accurate imputation is greater when more data is missing.

F. The Importance of Specific Features

To this point, we have treated all missing data as equally important and given the same weight to all errors. However, this is not always the right thing to do. In particular, it is clear that not all missing data is equally important for prediction. To understand the importance of missing data for purposes of prediction we conduct two experiments in parallel. For the first experiment (which we call Setting A: Purely Random Removal), we construct 5 sub-samples of the MIMIC-III dataset by randomly removing an *additional* 10%, 20%, 30%, 40%, 50% of the measurements for randomly chosen features. For the second experiment (which we call Setting B: Correlated Random Removal) we first identify the four features that are most highly correlated with the mortality label; those are anion gap, bicarbonate, systolic blood pressure, and pottasium. We then construct 5 sub-samples of the MIMIC-III dataset by removing an *additional* 10%, 20%, 30%, 40%, 50% of the measurements for *these specific features*. In both cases we repeat the exercise 10 times and report average results. We then compare the prediction performance of M-RNN (MI) with the best benchmarks; the results are shown visually in Fig. 6.

TABLE VI
CONGENIALITY OF IMPUTATION MODELS

Algorithm	Mean Bias ($\|\|\mathbf{w}-\hat{\mathbf{w}}\|\|_1$)	Root Mean Squared Error ($\|\|\mathbf{w}-\hat{\mathbf{w}}\|\|_2$)
M-RNN (MI)	0.0814 ± 0.0098	0.1229 ±0.0151
[25]	0.1097 ± 0.0104	0.1649 ± 0.0212
Cubic Interpolation	0.1169 ± 0.01075	0.1816 ± 0.0201
MissForest	0.0842 ±0.0103	0.1312 ± 0.0139
Matrix Completion	0.1001 ± 0.0125	0.1551 ± 0.0230

Fig. 6 shows that M-RNN outperforms the best benchmarks for every sub-sample and the improvement in performance is greater for the sub-samples for which more data is missing. The improvement in performance is statistically significant (p-value < 0.05) when an additional 30% or more of the measurements - i.e., a total of 82.5% of the measurements - (or of the most important features for Setting B) - are missing. In particular, the prediction performance of M-RNN is much less sensitive to the *amount of data that is missing* and to *which data is missing*.

G. Congeniality of the Model

As [26] has emphasized, an extremely desirable aspect of any imputation method is that it produce imputed values in a manner that is consistent and preserves the original relationships between features and labels; [26] refers to this as *congeniality*. Congeniality of an imputation model can be evaluated with respect to a particular model of the feature-label relationships by computing the model parameters for the true complete data and the imputed data and measuring the difference between parameters according to some specified metric. Of course no imputation method can be expected to be perfectly congenial, but we argue that our method is more congenial – i.e., better preserves the relationships between features and labels – than benchmarks. To see this, we exploit the Biobank dataset; this is a complete dataset, so that it is possible to compare the relationship between the actual (original) data and labels and the relationship between the the imputed data and labels.

In our particular experiment, we delete 20% of the data and impute the missing data using our M-RNN and the 4 best benchmarks (the method of [24], Cubic Interpolation, MissForest and Matrix Completion). As a model of the feature-label relationship, we use a logistic regression. As a metric of the difference between the logistic regression parameters $\mathbf{w}$ for the actual data and $\hat{\mathbf{w}}$ for the imputed data (which can be interpreted as a measure of the uncongeniality of the imputation) we report both the mean bias $\|\mathbf{w}-\hat{\mathbf{w}}\|_1$ and the root mean squared error $\|\mathbf{w}-\hat{\mathbf{w}}\|_2$.

As can be seen in Table VI, in comparison with the 4 best benchmarks, M-RNN achieves both smaller mean bias and small root mean squared error between the original and imputed representations of feature-label relationship. (With the exception of MissForest, all the performance improvements of M-RNN are statistically significant at the 95% level.) Thus our method is more congenial (to the logistic regression model) than the benchmarks.

TABLE VII
PERFORMANCE COMPARISON FOR MISSING DATA ESTIMATION FOR MCAR AND MAR SETTINGS ON THE BIOBANK DATASET

Algorithm	RMSE (% Gain of M-RNN (MI))	
	MCAR	**MAR**
M-RNN (MI)	**0.0637 (-)**	**0.1135 (-)**
[25]	0.0778 (18.1%)	0.1243 (8.7%)
Cubic Interpolation	0.0887 (28.2%)	0.1278 (11.2%)
MissForest	0.0892 (28.6%)	0.1359 (16.5%)
Matrix Completion	0.0886 (28.1%)	0.1331 (14.7%)

H. M-RNN When Data is Missing at Random

The experiments above are designed to demonstrate the superiority of the M-RNN framework in comparison to the benchmarks in settings where data is Missing Completely at Random (MCAR) [3] but it is important to understand the comparison in the settings where data is Missing at Random (MAR) - but *not* Missing Completely at Random. In this subsection, we show that M-RNN also outperforms the benchmarks when data is Missing at Random (MAR). (The assumption that data is Missing at Random is standard in the medical setting.)

To accomplish this, we again begin with the complete Biobank dataset and remove 20% of the data. However in this case we do not remove data completely at random; rather, we use the following procedure.[1] Using induction, we define the probability that the component i of sample n at time t is observed, conditional on the missingness and values (if observed) of the previous $i-1$ components at time t [3] to be

$$P_i^t(n) = \frac{p^m \cdot N \cdot e^{-\sum_{j<i} w_j m_j^t(n) x_j^t(n) + b_j (1-m_j^t(n))}}{\sum_{l=1}^{N} e^{-\sum_{j<i} w_j m_j^t(l) x_j^t(l) + b_j (1-m_j^t(l))}}$$

where p^m corresponds to the average missing rate (in our experiment, $p^m = 0.2$), and w_j, b_j are sampled from $\mathcal{U}(0,1)$ (but are only sampled once for the entire dataset). We sequentially sample $m_1^t, ..., m_D^t$ for each feature vector.

We compare the RMSE of M-RNN architecture against four competitive benchmarks: [24], Cubic Interpolation, MissForest, and Matrix Completion in both MCAR and MAR settings. As can be seen in Table VII, M-RNN outperforms other state-of-the-art imputation methods in both MCAR and MAR settings.

Additional experimental results and the details of the experiments, including standard deviations of the reported results, can be found in the Supplementary Materials: http://medianetlab.ee.ucla.edu/papers/TBME_MRNN_SM.pdf

VII. CONCLUSION

The problem of reconstructing/estimating missing data is ubiquitous in many settings – especially in longitudinal medical

[1] Other procedures are certainly possible.

TABLE VIII
PERFORMANCE COMPARISON FOR PATIENT STATE PREDICTION USING MIMIC-III DATASET

Category	Algorithms	AUROC (% Gain of M-RNN (Multiple Imputations))			
		RNN	Random Forest	Logistic Regression	Xgboost
M-RNN	**M-RNN (MI)**	**0.8531 (-)**	**0.8317 (-)**	**0.8389 (-)**	**0.8372 (-)**
	M-RNN (SI)	**0.8530 (-)**	**0.8331 (-)**	**0.8392 (-)**	**0.8351 (-)**
RNN-based	[23]	0.8381 (9.3%)	-	-	-
	[24]	0.8402 (8.1%)	-	-	-
	[25]	0.8410 (7.6%)	-	-	-
Interpolation	Spline	0.8407 (7.8%)	0.8260 (3.3%)	0.8367 (1.3%)	0.8328 (2.6%)
	Cubic	0.8397 (8.4%)	0.8173 (7.9%)	0.8296 (5.5%)	0.8223 (8.4%)
Imputation	MICE	0.8377 (9.5%)	0.8139 (9.6%)	0.8233 (8.8%)	0.8166 (11.2%)
	MissForest	0.8368 (10.0%)	0.8123 (10.3%)	0.8252 (7.8%)	0.8198 (9.7%)
	EM	0.8312 (13.0%)	0.8025 (14.8%)	0.8143 (13.2%)	0.8081 (15.2%)
Others	Matrix Completion	0.8401 (8.1%)	0.8118 (10.6%)	0.8301 (5.2%)	0.8240 (7.5%)
	Auto-encoder	0.8399 (8.2%)	0.8170 (8.0%)	0.8274 (6.7%)	0.8179 (10.6%)
	MCMC	0.8298 (13.7%)	0.8035 (14.4%)	0.8099 (15.3%)	0.8069 (15.7%)

TABLE IX
PERFORMANCE COMPARISON FOR MISSING VALUE ESTIMATION WITH/WITHOUT DONOR FEATURES IN UNOS DATASET

Category	Algorithms	Mean RMSE (% Gain of M-RNN (Multiple Imputations))			
		Heart (with)	Heart (w/o)	Lung (with)	Lung (w/o)
M-RNN	**M-RNN (MI)**	**0.0451 (-)**	**0.0479 (-)**	**0.0579 (-)**	**0.0606 (-)**
	M-RNN (SI)	**0.0453 (-)**	**0.0477 (-)**	**0.0583 (-)**	**0.0609 (-)**
RNN-based	[23]	0.1352 (66.6%)	0.1352 (64.6%)	0.1343 (56.9%)	0.1343 (54.9%)
	[24]	0.1179 (61.7%)	0.1179 (59.4%)	0.1264 (54.2%)	0.1264 (52.1%)
	[25]	0.1057 (57.3%)	0.1057 (54.7%)	0.1172 (50.6%)	0.1172 (48.3%)
Interpolation	Spline	0.1102 (59.1%)	0.1102 (56.5%)	0.1199 (51.7%)	0.1199 (49.5%)
	Cubic	0.1072 (57.9%)	0.1072 (55.3%)	0.1177 (50.8%)	0.1177 (48.5%)
Imputation	MICE	0.1067 (57.7%)	0.1147 (58.2%)	0.1015 (43.0%)	0.1151 (47.4%)
	MissForest	0.0465 (3.0%)	0.0489 (2.0%)	0.0627 (7.7%)	0.0652 (7.1%)
Others	Matrix Completion	0.0767 (41.2%)	0.0974 (50.8%)	0.0819 (29.3%)	0.0942 (35.7%)
	Auto-encoder	0.0544 (17.1%)	0.0589 (18.7%)	0.0668 (13.3%)	0.0712 (14.9%)
	MCMC	0.0934 (51.7%)	0.1091 (56.1%)	0.1017 (43.1%)	0.1124 (46.1%)

datasets – and is of enormous importance for many reasons, including statistical analysis, diagnosis, prognosis and treatment. In this paper we have presented a new method, based on a novel deep learning architecture, for reconstructing/estimating missing data that exploits both the correlation within data streams and the correlation across data streams. We have demonstrated on the basis of a variety of real-world medical datasets that our method makes large and statistically significant improvements in comparison with state-of-the-art benchmarks.

APPENDIX A
PREDICTIONS WITH ALTERNATIVE PREDICTIVE MODELS

Table V compares the implied predictive performance of M-RNN with the benchmarks when imputation is followed by prediction using a particularly simple predictive model. We now compare the implied predictive performance of M-RNN with the benchmarks when imputation is followed by prediction using other predictive models: in addition to the 1-layer RNN model used already, we use Random Forest [35], Logistic Regression and Xgboost [34]. (For implementations, we use the randomForest package, the glm package and the xgboost package, all in R.) We restrict our attention to the MIMIC-III dataset and continue to use AUROC as the performance metric. Table VIII shows the mean and performance gain (%) (in terms of AUROC) of M-RNN in comparison with the benchmarks with various predictive models.

As can be seen in Table VIII, the different predictive models do not yield much different prediction accuracy, and no one predictive model seems to consistently lead to more or less accurate predictions. (Because the RNN imputation methods [22]–[24] can only be combined with an RNN predictive model, there are no RF, LR, XG results for these methods.) Note that prediction accuracy is *not* perfectly correlated with imputation accuracy, but the differences are not statistically significant. As before, M-RNN leads to the most accurate predictions when followed by each of the various predictive models; again, not all the differences are statistically significant.

APPENDIX B
EFFECTS OF DONOR FEATURES IN UNOS DATASETS

The UNOS organ transplant dataset records features of both the recipient and the donor. In the main text we ignored donor features; here we expand the analysis to include these features, and compare the results obtained using only recipient features with the results obtained using both recipient and donor features. Table IX presents the comparison for imputations and Table X

TABLE X
PERFORMANCE COMPARISON FOR LABEL PREDICTION (AUROC) WITH/WITHOUT DONOR FEATURES IN UNOS DATASET

Category	Algorithm	AUROC (% Gain of M-RNN (Multiple Imputations))			
		Heart (with)	Heart (w/o)	Lung (with)	Lung (w/o)
M-RNN	**M-RNN (MI)**	**0.7153 (-)**	**0.6855 (-)**	**0.6883 (-)**	**0.6762 (-)**
	M-RNN (SI)	**0.7141 (-)**	**0.6858 (-)**	**0.6886 (-)**	**0.6759 (-)**
RNN-based	[23]	0.6830 (10.2%)	0.6505 (10.0%)	0.6575 (9.0%)	0.6557(6.0%)
	[24]	0.6892 (8.4%)	0.6574 (8.2%)	0.6651 (6.9%)	0.6561(5.8%)
	[25]	0.6962 (6.3%)	0.6583 (8.0%)	0.6728 (4.7%)	0.6520 (7.0%)
Interpolation	Spline	0.6817 (10.6%)	0.6477(10.7%)	0.6651 (6.9%)	0.6520 (7.0%)
	Cubic	0.6788 (11.4%)	0.6468 (11.0%)	0.6629 (7.5%)	0.6517 (7.0%)
Imputation	MICE	0.6785 (11.4%)	0.6397 (12.7%)	0.6567 (9.2%)	0.6509 (7.2%)
	MissForest	0.6981 (5.7%)	0.6740 (1.7%)	0.6754 (4.0%)	0.6587 (5.1%)
Others	Matrix Completion	0.6969 (6.1%)	0.6712 (2.2%)	0.6745 (4.2%)	0.6579 (5.3%)
	Auto-encoder	0.6966 (6.2%)	0.6633 (6.6%)	0.6741 (4.4%)	0.6574 (5.5%)
	MCMC	0.6772 (11.8%)	0.6417 (12.2%)	0.6717 (5.1%)	0.6512 (7.2%)

TABLE XI
PERFORMANCE COMPARISON FOR PREDICTION ORIENTED M-RNN

Algorithms	AUROC (% Gain of M-RNN (Multiple Imputations))				
	MIMIC-III	Deterioration	UNOS-Heart	UNOS-Lung	Biobank
M-RNN: Predict	**0.8578 (-)**	**0.7813 (-)**	**0.6897 (-)**	**0.6802 (-)**	**0.8987 (-)**
M-RNN: Original	**0.8531 (3.2%)**	**0.7779 (1.5%)**	**0.6855 (1.3%)**	**0.6762 (1.2%)**	**0.8955 (3.1%)**
[23]	0.8381 (12.2%)	0.7558 (10.4%)	0.6505 (11.2%)	0.6557 (7.1%)	0.8802 (15.4%)
[24]	0.8402 (11.0%)	0.7551 (10.7%)	0.6574 (9.4%)	0.6561 (7.0%)	0.8748 (19.1%)
[25]	0.8410 (10.6%)	0.7593 (9.1%)	0.6583 (9.2%)	0.6520 (8.1%)	0.8826 (13.7%)

TABLE XII
PERFORMANCE COMPARISON FOR MISSING DATA ESTIMATION FOR CATEGORICAL VARIABLES

Category	Algorithm	Mean PFC (% Gain of M-RNN (Multiple Imputations))				
		MIMIC-III	Deterioration	UNOS-Heart	UNOS-Lung	Biobank
M-RNN	**M-RNN (MI)**	0.0295 (-)	0.0225 (-)	0.0621 (-)	0.0718 (-)	0.1167 (-)
	M-RNN (SI)	0.0299 (-)	0.0229 (-)	0.0618 (-)	0.0722 (-)	0.1155 (-)
RNN-based	[23]	0.0471 (37.4%)	0.0308 (26.9%)	0.1459 (57.4%)	0.1371 (47.6%)	0.1376 (15.2%)
	[24]	0.0412 (28.4%)	0.0290 (22.4%)	0.1279 (51.4%)	0.1281 (44.0%)	0.1347 (13.4%)
	[25]	0.0406 (27.3%)	0.0279 (19.4%)	0.1153 (46.1%)	0.1199 (40.1%)	0.1298 (10.1%)
Interpolation	Spline	0.0394 (25.1%)	0.0258 (12.8%)	0.1199 (48.2%)	0.1219 (41.1%)	0.1419 (17.8%)
	Cubic	0.0389 (24.2%)	0.0278 (19.1%)	0.1163 (46.6%)	0.1195 (39.9%)	0.1492 (21.8%)
Imputation	MICE	0.0848 (65.2%)	0.0381 (40.9%)	0.1249 (50.3%)	0.1175 (38.9%)	0.1534 (23.9%)
	MissForest	0.1179 (75.0%)	0.0702 (67.9%)	0.0946 (34.4%)	0.1148 (37.5%)	0.2343 (50.2%)
	EM	0.1891 (84.4%)	0.0933 (75.9%)	-	-	0.2561 (54.4%)
Others	Matrix Completion	0.1246 (76.3%)	0.0698 (67.8%)	0.1900 (67.3%)	0.1642 (56.3%)	0.2327 (49.8%)
	Auto-encoder	0.1664 (82.3%)	0.0812 (72.3%)	0.1143 (45.7%)	0.1260 (43.0%)	0.2113 (44.8%)
	MCMC	0.1766 (83.3%)	0.0966 (76.7%)	0.2110 (70.6%)	0.1987 (63.9%)	0.2476 (52.9%)

presents the comparisons for prediction. As noted in the main text, the differences are not large, presumably because the donor information is completely static and the relevant aspects are captured implicitly in the recipient data. Note that the RNN-based Interpolation-based methods do not utilize information *across* features; hence, whether or not donor features are available, the imputation performance of these methods is unchanged (See Table IX).

As can be seen in Tables IX and X, the effects of donor features on imputation and prediction accuracy are not large. The RMSE of M-RNN (MI) improves from from 0.0479 to 0.0451 (5.8%) on the UNOS-Heart dataset and 0.0606 to 0.0579 (4.5%) on the UNOS-Lung dataset. In terms of prediction, AUROC of M-RNN (MI) improves from from 0.6855 to 0.7153 (9.5%) on the UNOS-Heart dataset and from 0.6762 to 0.6883 (3.7%) on UNOS-Lung dataset.

APPENDIX C
PREDICTION-ORIENTED M-RNN

Finally, we report the results of an experiment in which we optimized M-RNN for prediction accuracy rather than imputation accuracy. To do this, we trained M-RNN to minimize the cross-entropy $\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}[y_n \log \hat{y}_n + (1 - y_n)$

$\log(1-\hat{y}_n)]$, rather than the mean squared error. (We continue to evaluate performance in terms of AUROC.) As can be seen from Table XI, doing this does improve the predictions of M-RNN, but the improvement is marginal and not statistically significant. (However, doing this does have the advantage that it creates an end-to-end prediction algorithm that does not require any preprocessing or imputation steps.)

Appendix D
Categorical Value Imputation

As an additional performance metric for comparing the imputation quality of categorical features, we compute the proportion of falsely classified entries (PFC). If $\mathcal{S}_{cat}$ is the set of categorical variables, then PFC is defined as

$$PFC = \frac{\sum_{t=1}^{T}\sum_{n=1}^{N}\sum_{j\in\mathcal{S}_{cat}}(1-m_j^t(n))\mathbb{I}(\hat{x}_j^t(n)\neq x_j^t(n))}{\sum_{t=1}^{T}\sum_{n=1}^{N}\sum_{j\in\mathcal{S}_{cat}}(1-m_j^t(n))},$$

Keep in mind that *smaller* PFC means *better* imputation. As can be seen in Table XII, M-RNN outperforms all the benchmarks in all the datasets in terms of the PFC metric as well.

Appendix E
Implementations and Hyper-parameter Optimization

In all of our experiments, we set the depth of the networks for M-RNN and for other neural network benchmarks (including RNN-based benchmarks and auto-encoder) to 4. (In the case of M-RNN, the interpolation block uses 2 layers and the imputation block uses 2 layers.) For M-RNN, there are 4 hidden nodes in each layer in the interpolation block and D hidden nodes in each layer in the imputation block. For the benchmarks, in order to make a fair comparison, we adjusted the number of hidden nodes in each layer to match the model capacity (the number of parameters for all models) of M-RNN. The number of batches is 64 for both M-RNN and benchmarks.

For some of these algorithms, we are able to use off-the-shelf implementations. For Spline and Cubic Interpolation, we use the interp1 package in MATLAB; for MICE we use the mice package in R; for MissForest we use the MissForest package in R; for EM we use the Amelia package in R; for matrix completion we use the softImpute package in R.

Acknowledgment

The authors would like thank K. Poppe (University of Auckland) and A. Wood (University of Cambridge).

References

[1] D. M. Kreindler and C. J. Lumsden, "The effects of the irregular sample and missing data in time series analysis," *Nonlinear Dynamical Syst. Anal. Behavioral Sci.*, vol. 10, pp. 187–214, 2006.
[2] D. Mondal and D. B. Percival, "Wavelet variance analysis for gappy time series," *Ann. Inst. Statistical Math.*, vol. 62, no. 5, pp. 943–966, 2010.
[3] D. B. Rubin, *Multiple Imputation for Nonresponse in Surveys*, vol. 81. Hoboken, NJ, USA: Wiley, 2004.
[4] P. J. García-Laencina *et al.*, "Pattern classification with missing data: A review," *Neural Comput. Appl.*, vol. 19, no. 2, pp. 263–282, 2010.
[5] I. R. White *et al.*, "Multiple imputation using chained equations: Issues and guidance for practice," *Statist. Med.*, vol. 30, no. 4, pp. 377–399, 2011.
[6] D. J. Stekhoven and P. Bühlmann, "Missforest—non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2011.
[7] R. Mazumder *et al.*, "Spectral regularization algorithms for learning large incomplete matrices," *J. Mach. Learn. Res.*, vol. 11, pp. 2287–2322, 2010.
[8] H.-F. Yu *et al.*, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Proc Adv. Neural Inf. Process. Syst.*, 2016, pp. 847–855.
[9] T. Schnabel *et al.*, "Recommendations as treatments: Debiasing learning and evolution," in *Proc 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1670–1679.
[10] R. Mazumder *et al.*, "Spectral regularization algorithms for learning large incomplete matrices," *J. Mach. Learn. Res.*, vol. 11, pp. 2287–2322, 2010.
[11] A. M. Alaa *et al.*, "Learning from clinical judgments: Semi-Markov-modulated marked Hawkes processes for risk prognosis," in *Proc. 34th Int. Conf. Mach. Learn*, 2017, pp. 60–69.
[12] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5, pp. 602–610, 2005.
[13] A. R. T. Donders *et al.*, "A gentle introduction to imputation of missing values," *J. Clin. Epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.
[14] P. A. Patrician, "Multiple imputation for missing data," *Res. Nursing Health*, vol. 25, no. 1, pp. 76–84, 2002.
[15] S. Burgess *et al.*, "Combining multiple imputation and meta-analysis with individual participant data," *Statist. Med.*, vol. 32, no. 26, pp. 4499–4514, 2013.
[16] A. Mackinnon, "The use and reporting of multiple imputation in medical research–A review," *J. Internal Med.*, vol. 268, no. 6, pp. 586–593, 2010.
[17] J. A. Sterne *et al.*, "Multiple imputation for missing data in epidemiological and clinical research: Potential and pitfalls," BMJ 2009;338:b2393.
[18] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[19] A. E. Johnson *et al.*, "MIMIC-III, a freely accessible critical care database," *Sci. Data*, vol. 3, 2016, Art. no. 160035.
[20] A. M. Alaa *et al.*, "Personalized risk scoring for critical care prognosis using mixtures of gaussian processes," *IEEE Trans. Biomed. Eng.*, vol. 65, no. 1, pp. 207–218, Jan. 2018.
[21] L. J. Palmer, "UK biobank: Bank on it," *Lancet*, vol. 369, no. 9578, pp. 1980–1982, 2007.
[22] E. Choi *et al.*, "Doctor AI: Predicting clinical events via recurrent neural networks," *Mach. Learning Healthcare Conf.*, pp. 301–318, 2016.
[23] Z. C. Lipton *et al.*, "Directly modeling missing data in sequences with RNNs: Improved classification of clinical time series," *Mach. Learning Healthcare conf.*, pp. 253–270, 2016.
[24] Z. Che *et al.*, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, 2018, Art. no. 6085.
[25] Y. Deng *et al.*, "Multiple imputation for general missing data patterns in the presence of high-dimensional data," *Sci. Rep.*, vol. 6, 2016, Art. no. 21689.
[26] X.-L. Meng, "Multiple-imputation inferences with uncongenial sources of input," *Statistical Sci.*, vol. 9, pp. 538–558, 1994.
[27] F. Gingras and Y. Bengio, "Recurrent neural networks for missing or asynchronous data," in *Proc 8th Int. Conf. Neural Inf. Process. Syst.*, 1996, vol. 8, pp. 395–401.
[28] V. Tresp and T. Briegel, "A solution for missing data in recurrent neural networks with an application to blood glucose prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, 1998, pp. 971–977.
[29] S. Parveen and P. Green, "Speech recognition with missing data using recurrent neural nets," *in Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1189–1195.
[30] H.-G. Kim *et al.*, "Recurrent neural networks with missing information imputation for medical examination data prediction," in *Proc. IEEE Conf. Big Data Smart Comput.*, 2017, pp. 317–323.
[31] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.
[32] L. Gondara and K. Wang, "Multiple imputation using deep denoising autoencoders," 2017, arXiv:1705.02737.
[33] D. Schunk, "A Markov chain monte carlo algorithm for multiple imputation in large surveys," *AStA Adv. Statistical Anal.*, vol. 92, no. 1, pp. 101–114, 2008.
[34] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.
[35] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.