End-to-End Framework for Imputation and State Discovery in Longitudinal Energy Data

Alexander Ladd
ladd12@llnl.gov
Lawrence Livermore National Lab
Livermore, California, USA

Kwan Ho Ryan Chan
chan59@llnl.gov
Lawrence Livermore National Lab
Livermore, California, USA

Sam Nguyen
nguyen116@llnl.gov
Lawrence Livermore National Lab
Livermore, California, USA

Jose Cadena
cadenapico1@llnl.gov
Lawrence Livermore National Lab
Livermore, California, USA

Brenda Ng
ng30@llnl.gov
Lawrence Livermore National Lab
Livermore, California, USA

ABSTRACT

High-resolution signals from micro-phasor measurement units (μPMU) contain crucial information about the health and status of electric equipment in power grids. In this work, we provide an end-to-end framework for fault state discovery in μPMU data. Our proposed method uses a data-driven deep learning method called Latent Ordinary Differential Equations (LatentODE) for data imputation, followed by a Bayesian nonparametric method called distance-dependent Chinese Restaurant Franchise (dd-CRF) for unsupervised discovery of latent states of the energy grid. We applied our framework to analyze one month of μPMU data and were able to correctly binned 60% of the faults within our predicted faulty time segments.

We applied our framework to the task of identifying time segments when faults occur. faulty state of the time series. Our experiments show that our method produces comparably better results than traditional interpolation methods and interpretable regions for fault states in power electric systems.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by regression**; **Cluster analysis**; *Neural networks*; *Latent variable models*; • **Applied computing** → Mathematics and statistics.

ACM Reference Format:
Alexander Ladd, Kwan Ho Ryan Chan, Sam Nguyen, Jose Cadena, and Brenda Ng. 2021. End-to-End Framework for Imputation and State Discovery in Longitudinal Energy Data. In *The Twelfth ACM International Conference on Future Energy Systems (e-Energy '21), June 28-July 2, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3447555.3466588

e-Energy '21, June 28-July 2, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8333-2/21/06…$15.00
https://doi.org/10.1145/3447555.3466588

1 INTRODUCTION

A phasor measurement unit (PMU) is a device that measures electrical signals on an electrical grid using Global Positioning System (GPS) time for signal synchronization. PMU provides high-resolution, precise measurements of voltages and current phasors (both magnitude and phase angle) in real-time, thus playing an essential role in the monitoring, protection and feedback control of power networks [22].

The complexity of distribution networks, such as the large number of nodes, shorter distances between nodes requiring higher accuracy to disambiguate between nearby signals, faster dynamics due to transient behaviors in distribution assets [26], can benefit from information collected by micro-phasor measurement units (μPMU) [19]. A μPMU reports four fundamental measurements (voltage magnitude, voltage phase angle, current magnitude, and current phase angle per phase) on three phases, with a maximum reporting rate of 120 Hz [15, 17]. Together, this results in $3 \times 4 = 12$ channels of highly-resolved data.

The adoption of μPMU on distribution networks offers a ripe environment for large-scale machine learning to improve situation awareness of the distribution grid [11, 16]. [7] reviews the state-of-the-art research on μPMU data, in the context of monitoring, diagnostic and control applications.

Due to the sheer scale of μPMU data, labelling the data for faults is simply not scalable, so unsupervised learning is generally pursued to identify anomalies. The identified anomalies would then be cross-referenced with maintenance or outage reports, and/or presented to a subject matter expert, for verification as ground-truth faults for other downstream supervised learning tasks. In this work, we extend [2], which applies a nonparametric clustering method to partition a single channel of μPMU data into similarly behaving segments. This work extends this method to handle multiple channels, in order to capture correlations across channels to improve fault detection.

In most cases, when a particular channel has missing data, that period of missing data is generally omitted from analysis. Depending on statistical correlations inherent in the μPMU data, it may be possible to impute the missing values using a machine learning method that can handle irregularly-sampled data. In this work, we propose the use of LatentODE [14] to learn the continuous-time hidden dynamics that underlie the

μPMU data. The continuous-time hidden dynamics would be defined by ordinary differential equations (ODEs), which are used to predict the missing time points.

1.1 Contributions

Our contributions are summarized as follows:

(1) a nonparametric segmentation method on multi-channel μPMU data, inspired by [2] which builds on the Chinese Restaurant Process [21]; and
(2) an imputation method developed for μPMU data, which combines a general interpolation method with deep learning to capture transient behaviors inherent in μPMU data.

2 METHODS

Given longitudinal electrical measurements, such as μPMU data, our goal is to discover latent *operating states* of the underlying energy grid. Operating states could represent baseline operation or fault states associated with equipment failure, environmental causes, or voltage regulation device actions.

We model the longitudinal data as a multivariate time series $\mathbf{X} = \{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$, where $\mathbf{x}_t \in \mathbb{R}^C$ is a C-dimensional vector of observations captured at time $t \in \{0, \ldots, T\}$. In the case of μPMU data, if all the channels are used, then $C = 12$, since there are four measurements (voltage magnitude, voltage phase angle, current magnitude, and current phase angle) across each of the three phases.

We assume that the multivariate time series $\mathbf{X}$ can be partitioned into disjoint time segments, and each segment can be mapped to a particular latent state. Each latent state defines a statistical distribution over the observations within a particular time segment.

In what follows, we will explain our two-stage imputation procedure to estimate missing data, then describe our time series segmentation process. Given *lossy* data (i.e., a multivariate time series with missing time points), denoted by $\tilde{\mathbf{X}}$, we impute the missing values to arrive at the *full* data $\mathbf{X}$ (i.e., a regularly-sampled, multivariate time series). Then, we perform a two-level clustering on $\mathbf{X}$. At the first level, a set of contiguous measurements are grouped into M segments. At the second level, these M segments are clustered into K operating states. Nonparametric clustering is used to infer M and K from the data. Algorithm 1 and Figure 2 shows a quick overview of our method.

2.1 Imputation of Missing Data

Correlations between different channels of μPMU data may be leveraged to impute missing time points on any particular channel of the data. In this section, we describe a two-phase process (cf. Figure 1) in which we first learn a *global* trend of the signal, then learn transient behaviors that may be present, conditional on this global trend. The two-stage imputation process allows for multi-resolution imputation, which is crucial for highly-resolved data that exhibits volatile dynamics.

Algorithm 1

Input: C-channel time series $\tilde{\boldsymbol{X}} = \{\boldsymbol{x}_{t_i}\}$, where $\boldsymbol{x}_{t_i} \in \mathbb{R}^C$ with corresponding time points $\mathcal{T} = \{t_i\}_{i=0}^T$

 # Step 1: Global Interpolation
Interpolate data with cubic spline $\boldsymbol{X} = \text{CubicSpline}(\tilde{\boldsymbol{X}})$

 # Step 2: Local Interpolation with LatentODE
Compute latent states by passing inputs through an ODE-RNN: $z_0, z_1, z_2, \ldots, z_T = \text{ODE-RNN}(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$
Infer posterior for z_0 as a Gaussian distribution with parameter μ_{z_0}, σ_{z_0}: $q_\phi(z_0|\{\boldsymbol{x}_i, t_i\}_{i=0}^T) = \mathcal{N}(\mu_{z_0}, \sigma_{z_0})$
Use an ODESolver to infer hidden states based on the initial hidden state z_0': $z_1', z_2', \ldots, z_T' = \text{ODESolver}(z_0', f_\theta, \{t_1, t_2, \ldots, t_T\})$
Transform latent states back to data space via ODE-RNN: $\boldsymbol{x}_0', \boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_T' = \text{ODE-RNN}(z_0', z_1', z_2', \ldots, z_T')$

 # Step 3: State-discovery with dd-CRF
Divide $\boldsymbol{x}_0', \boldsymbol{x}_1', \boldsymbol{x}_2', \ldots, \boldsymbol{x}_T'$ into 1-hour blocks: $B_1, B_2, \ldots, B_H$
Obtain segments in each hour via Gibbs Sampling (Section 2.2)
Concatenate segments and infer CRP (Section 2.2)

Output: Assignments of timepoints $\boldsymbol{x}_i'$ to operating states c_k

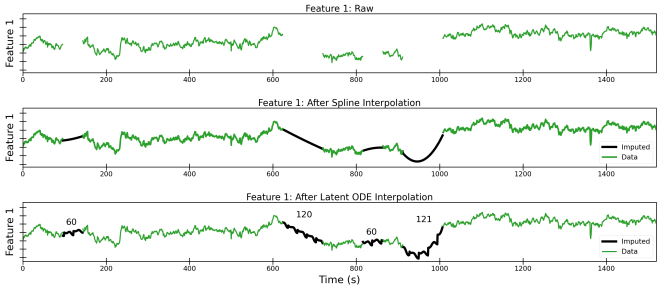


Figure 1: Imputation of missing longitudinal energy data. Measurements recorded by μPMU devices may contain large gaps (top). We first estimate the global trend of the data using univariate splines (middle). Then, we refine this estimate using LatentODE for the local trends (bottom).

2.1.1 Global learning. Imputation of missing data is achieved through a sequence of two phases: a global learning phase and a local learning phase. The global phase involves spline interpolation [6], fitting a one-dimensional piece-wise polynomial over univariate observations from regular time intervals $[t_0, t_1), [t_1, t_2), \ldots, [t_f - 1, t_f)$, with $t_0 = 0$ and $t_f = T$. Since splines are first- and second-order continuous, having well-defined derivatives at change points, this guarantees that the interpolation would not introduce any discontinuities to the imputed data. Let $\mathcal{T} = \{t_i\}_{i=0}^T$. For every single-channel measurement $\mathbf{X}^{(c)} = \left\{x_\tau^{(c)}\right\}_{\tau \in \mathcal{T}}$, we optimize the univariate spline

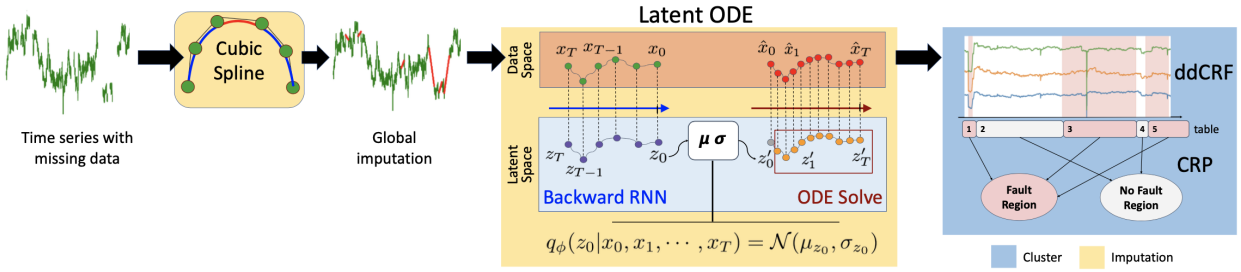


Figure 2: Schematic of our proposed end-to-end imputation and segmentation framework for μPMU data.

objective function individually, as in [8]:

$$\text{Spline}(Q, \lambda) = \frac{1}{T} \sum_{\tau \in \mathcal{T}} \left(q(\tau) - x_\tau^{(c)} \right)^2 + \lambda \int dt \left(q''(t) \right)^2, \quad (1)$$

where $q \in Q$ is the set of polynomial functions and λ is a tuned regularization hyper-parameter for polynomial curvature. Due to the continuous nature of the splines, it is well suited to capture the smooth global trend of the data.

2.1.2 Local learning. The interpolated data from the spline interpolation is partitioned into finely resolved time segments (30 seconds in our experiments) and these segments are put through a LatentODE [14] to learn the high-frequency signals present in the data.

LatentODE is a data-driven latent-variable time series model that parameterizes observations into latent states, such that dynamics are learned in a latent space instead of the data space. To simplify notation, we drop the subscript from our (irregular) time points, with the understanding that $\boldsymbol{x}_i := \boldsymbol{x}_{t_i}$. In a LatentODE, observations $\boldsymbol{x}_i$ are mapped to latent states $\boldsymbol{z}_i$ using an encoder ODE-RNN. An ODE-RNN is a specialized RNN in which the hidden states between observations can propagate in time based on an ODE as needed.

$$\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_T = \text{ODE-RNN}(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T) \quad (2)$$

Here, ODE-RNN takes in observations starting from $\boldsymbol{x}_T, \boldsymbol{x}_{T-1}, \ldots, \boldsymbol{x}_0$ then iteratively computes the latent states $\boldsymbol{z}_T, \boldsymbol{z}_{T-1}, \ldots, \boldsymbol{z}_0$ through an RNN. If observations $\boldsymbol{x}_j$ and $\boldsymbol{x}_{j-1}$ are too far apart, then it uses an ODESolver to approximate the intermediate latent states at the intermediate timesteps. Once $\boldsymbol{z}_0$ is obtained, it is passed through a variational autoencoder to learn the posterior q_ϕ, parametrized as a Gaussian distribution with mean $\mu_{\boldsymbol{z}_0}$ and standard deviation $\sigma_{\boldsymbol{z}_0}$:

$$q_\phi \left(\boldsymbol{z}_0 | \{\boldsymbol{x}_i, t_i\}_{i=0}^T \right) = \mathcal{N}(\mu_{\boldsymbol{z}_0}, \sigma_{\boldsymbol{z}_0}) \quad (3)$$

Samples are drawn from the posterior to get the initial *latent states* $\boldsymbol{z}'_0$. The other latent states at future time points are derived by solving an ODE initial-value problem with the forward model defined by the neural network f_θ:

$$\boldsymbol{z}'_1, \boldsymbol{z}'_2, \ldots, \boldsymbol{z}'_T = \text{ODESolver} \left(\boldsymbol{z}'_0, f_\theta, \{t_1, t_2, \ldots, t_T\} \right). \quad (4)$$

Finally, latent states are mapped from the latent space back to the data space via a RNN as a decoder network:

$$\boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_T = \text{ODE-RNN} \left(\boldsymbol{z}'_1, \boldsymbol{z}'_2, \ldots, \boldsymbol{z}'_T \right) \quad (5)$$

To optimize our neural networks f_θ and q_ϕ, we maximize the evidence lower bound (ELBO):

$$\begin{aligned} \text{ELBO}(\theta, \phi) = \mathbb{E}_{\boldsymbol{z}_0 \sim q_\phi(\boldsymbol{z}_0 | \{\boldsymbol{x}_i, t_i\}_{i=0}^T)} [\log p_\theta(\boldsymbol{x}_0, \ldots \boldsymbol{x}_T)] \\ - KL \left[q_\phi \left(\boldsymbol{z}_0 | \{\boldsymbol{x}_i, t_i\}_{i=0}^T \right) || p(\boldsymbol{z}_0) \right] \end{aligned} \quad (6)$$

We use Gated Recurrent Units (GRU) for our RNN. LatentODE is superior to other time series algorithms in that it is able to learn complex dynamics of irregularly sampled observations with missing values. Compared to traditional auto-regressive methods, LatentODE leverages data-driven learnable neural networks to capture nonlinear dynamics. LatentODE extends NeuralODE [3], and has been applied to multivariate time series prediction in other domains, such as Black Sigatoka infection [25] and virus outbreak prediction [12]. We discuss implementation details and compare against other imputation methods in Section 3.

2.2 Segmentation and State Discovery

In this section, we explain the two-level nonparametric clustering used to segment the full (imputed) data $\mathbf{X}$. At the first level, a set of contiguous measurements are grouped into M segments. At the second level, these M segments are clustered into K operating states. Both levels leverage the Chinese Restaurant Process for modeling the cluster distributions.

2.2.1 Chinese Restaurant Process. The Chinese Restaurant Process (CRP) [21] is a nonparametric Bayesian model that defines a distribution over clusterings of a dataset. The model was inspired by the following restaurant analogy: A set of customers (data) indexed by $i = 0, \ldots, T$ walk into a restaurant that has a set of infinite tables (clusters). When customer i enters, that person chooses to sit at a table with probability proportional to the popularity of the table, defined by the number of customers already sitting there. Formally, let t_i be the table chosen by customer i. Then, the probability that customer i chooses a certain

table j is given by:

$$p(t_i = j|\gamma) \propto \begin{cases} n_j, & n_j > 0 \\ \gamma, & j \text{ is a new, empty table} \end{cases} \tag{7}$$

where n_j is the number of customers from 1 to $i-1$ who have chosen table j so far, and γ is a *dispersion* parameter that controls the expected number of non-empty tables.

The cluster assignments inferred by the CRP exhibit a *rich-get-richer* structure, where a small number of tables contains most of the customers. This is a desirable property in our application, since we expect a majority of the observations to belong to a baseline operating state and a small minority to belong to the fault states. Unlike parametric clustering methods, such as k-means, the number of clusters is **not** a parameter to the model. The CRP infers the number of clusters directly from the data, which is a desirable property as a suitable number of states is difficult to determine *a priori*.

2.2.2 Distance-dependent Chinese Restaurant Process. Ideally, we want to enforce temporal continuity in the first-level clustering, so data from consecutive time points may be assigned to contiguous segments. Returning to the restaurant analogy, any customer may end up sitting with any other customer, whereas we want to find an assignment in which only customers with contiguous indices can sit together. In order to infer segments, we consider a CRP variant called the *distance-dependent* CRP (dd-CRP) [1]. The dd-CRP extends CRP by considering a distance function d, where $d(i, j)$ is the distance between a pair of customers. Upon entering the restaurant, a customer i either chooses to sit by themselves or to sit with a *friend*, some other customer j. This choice has probability inversely proportional to the distance between i and j. In our application, we choose d to be the index distance:

$$d(i, j) = |i - j|, \tag{8}$$

and we allow customers to sit together only if they have contiguous indices. Formally, let f_i denote the friend chosen by i; the probability that i chooses to sit with j is given by

$$p(f_i = j|d, \alpha) \propto \begin{cases} 1, & i \neq j \text{ and } d(i, j) = 1 \\ 0, & i \neq j \text{ and } d(i, j) > 1 \\ \alpha, & i = j \end{cases} \tag{9}$$

where "$i = j$" indicates that the customer chooses herself and α is a dispersion parameter. At the end of the process, the customer assignments induce a clustering of the data into M tables.

With the dd-CRP, we can infer a segmentation of the time series, and then we can apply a standard CRP to cluster these segments into states, with the property that segments in the same state will have the same statistical properties (i.e., they are generated from the same probability distribution with state-specific parameters). The combination of the dd-CRP and CRP results in a two-level clustering model called the distance-dependent Chinese Restaurant Franchise (dd-CRF) [9]. Cadena et al. [2] applied a univariate version of dd-CRF towards *fingerprint discovery* in univariate energy data. Our work extends dd-CRF to multivariate data.

In our proposed multivariate dd-CRF framework, we assume that each observation in the time series is sampled from a multivariate normal distribution with mean vector $\mu \in \mathbb{R}^C$ and covariance matrix $\Sigma \in \mathbb{R}^{C \times C}$. The full generative process of our proposed model is outlined as follows:

(1) For each customer $i = 1, \ldots, T$, select a friend f_i with probability $p(f_i|d, \alpha)$ defined in Equation 9. Let t_i be the table of i corresponding to the customer assignment.
(2) For each table $t = 1 \ldots, M$, select a state k_t with probability $p(k_t|\alpha)$ defined in Equation 7. Denote the state of customer i by $z_i = k_{t_i}$.
(3) For each state $k = 1 \ldots K$, sample the state parameters $\theta_k = (\mu_k, \Sigma_k)$.
(4) For each customer $i = 1, \ldots, T$, generate an observation $x_i \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$.

2.2.3 Inference. Here, we describe how to perform inference for the CRP and dd-CRP models. Inference is performed separately for CRP and dd-CRP in order to scale to long time series. Given a time series of μPMU data, we first divide the time series into 1-hour blocks and infer a dd-CRP in each block to obtain a segmentation. Then, we concatenate these segments and use it as input to the CRP model, which yields the operating states.

Inference for both models is performed via Gibbs sampling. We update the segment and cluster assignments of a single point in the time series by fixing all the assignments for the other points and sampling from the conditional distribution. For dd-CRP, let f_{i-} denote the current friend assignments for all customers except for i. Then, the friend of i is sampled from

$$p(f_i|f_{i-}, d, \alpha, \mathbf{X}) \propto p(f_i|d, \alpha)p(\mathbf{X}|f_i, f_{i-}), \tag{10}$$

where we obtain the first term from the fact that the friendship assignments depend only on the distance d and the dispersion α(cf. Equation 9). The second term is the likelihood of the data given all the friendship assignments. Suppose the friendship assignment induces M tables. Let t_i denote the table of customer i. Then, given our choice of multivariate normal distribution, the likelihood is given by:

$$\begin{aligned} p\left(\mathbf{X}|f_i, f_{i-}, \{\theta_m\}_{m=1}^M\right) &= p\left(\mathbf{X}|\{t\}_{i=1}^T, \{\theta_m\}_{m=1}^M\right) \\ &= \prod_{m=1}^{M} p(\mathbf{X}_m|\theta_m), \end{aligned} \tag{11}$$

where $\mathbf{X}_m = \{\mathbf{x}_j|t_j = m\}$ is the set of observations associated with table m. We infer the posterior distribution on the friend assignments by iteratively sampling the assignment for each customer over a pre-specified number of iterations.

Similarly, we infer the CRP by sampling from the conditional distribution of the table assignments:

$$p(c_i|c_{i-}, \gamma, \mathbf{X}) \propto p(c_i|\gamma)p(\mathbf{X}|c_i, c_{i-}). \tag{12}$$

In our application, each hour of data is processed in parallel via a "train" of dd-CRP models. The results of these dd-CRP

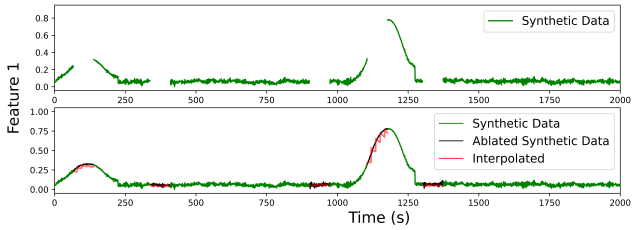


Figure 3: An example of our synthetic dataset (top), with imputation over two failures (bottom).

models are then concatenated as the input into CRP to obtain the final operating states.

3 EXPERIMENTS AND RESULTS

In this section, we present our findings on a synthetic dataset (to be released along with our code) and a proprietary μPMU dataset.

3.1 Hardware

We trained our models on computing nodes with IBM Power8 CPU 20 cores per node and 4 NVIDIA Tesla P100 SXM2 GPUs per node. Each node has 256 GB of CPU memory and 15 GB of GPU memory. The LatentODE model is trained in a distributed manner across 4 GPUs.

3.2 Data

3.2.1 μPMU Dataset. Our μPMU dataset is from a US-based utility company. For this study, we focused on one month of labelled μPMU data and performed our experiments using 3 channels of the data. In our plots, we refer to each channel as a "feature." The hourly data are partitioned as follows: Normal hours are used as training data for LatentODE, which is used to impute missing time points within faulty hours to form a *full* dataset of faulty hours. Then, the (full) faulty hours are concatenated together as input to dd-CRF. Each univariate measurement is z-scored normalized.

3.2.2 Synthetic Dataset. We generated a synthetic dataset in which a fixed number of random failures would occur at non-overlapping intervals. This dataset is constructed piecewise depending on whether the time points are within a fault window or not, as follows:

$$\begin{cases} f(\boldsymbol{x}_t) = (\mathbf{1} - s(t))f_0(\boldsymbol{x}_t) + s(t)(f_1(\boldsymbol{x}_t)) & t \in T_{\text{faulty}} \\ f(\boldsymbol{x}_t) = f_0(\boldsymbol{x}_t) & t \in T_{\text{normal}} \end{cases} \quad (13)$$

where $s(t) = \sigma(asin(ct)) \in [0, 1]$. $s(t)$ is periodic, so the sigmoid function allows smooth transitions between faulty states and normal states with length-scale defined by constants a and c. f_0 is the function generating values within normal intervals and $f_1(x)$ is the function generating values within failure intervals. There are many choices for these functions, but we chose to use stable multivariate autoregressive processes [20], AR(5). Formally, $f_0(\boldsymbol{x}_t) = \boldsymbol{a_0} + \boldsymbol{a}_1\boldsymbol{x}_{t-1} + \boldsymbol{a}_2\boldsymbol{x}_{t-2} + \ldots + \boldsymbol{a}_5\boldsymbol{x}_{t-5} + \boldsymbol{\epsilon}_t$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$ and $\boldsymbol{a}_0, ..., \boldsymbol{a}_5$ are constant vectors. Varying

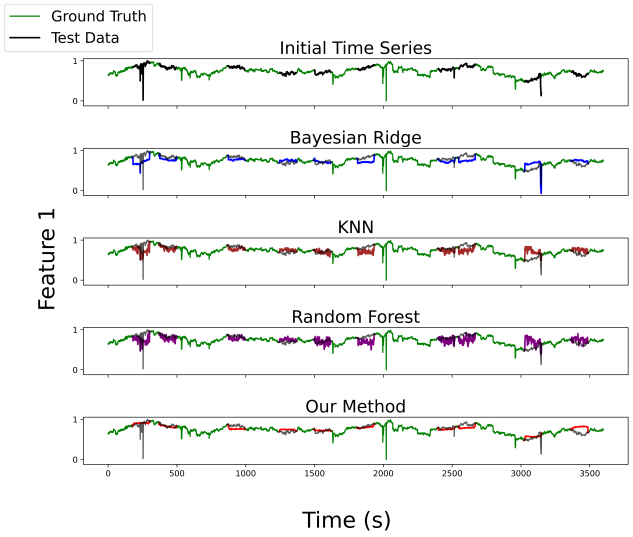


Figure 4: Comparison of imputation performance between our method and baselines, given the initial time series (top) as input to all methods.

the correlation within each vector $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_5$ results in channels that co-vary correspondingly. $f_1(\boldsymbol{x}_t)$ is defined using the same formula as $f_0(\boldsymbol{x}_t)$, but f_0 coefficients are $\boldsymbol{A}_0 \sim U[-.5, .5]$ and $\boldsymbol{A}_1$ coefficients are $\boldsymbol{A}_1 \sim U[0, 1]$. These constants define f_1 as a monotonically increasing AR process. The combination of f_1 and f_2, as defined in Equation 13, mimics a spiking electrical current. To generate the lossy versions of this data, we ablated fixed-length time segments based on these randomly-sampled start times.

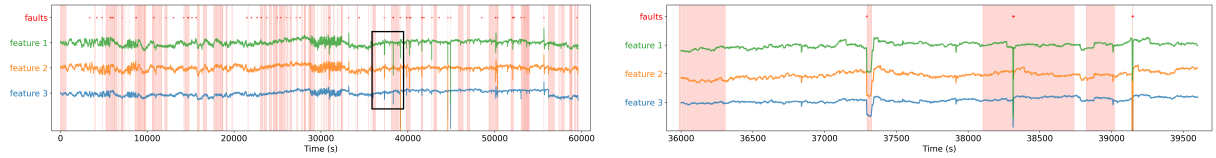
Table 1: Imputation results on μPMU data (Δt=60s).

Method	RMSE	MAE
Bayesian Ridge	0.0897 ± 0.0561	0.0647 ± 0.0476
KNN	0.1009 ± 0.0604	0.0690 ± 0.0509
Random Forest	0.1076 ± 0.0569	0.0712 ± 0.0472
Our Method	**0.0816 ± 0.0561**	**0.0572 ± 0.0476**

3.3 Imputation

Imputation of missing data is the first step of the experimental procedure. First, the global trend is learned using cubic spline (with regularization constant $\lambda = 7$) on the (non-missing) data. (The choice of cubic spline with this specific λ was chosen after hyper-parameter tuning.) The result of spline interpolation is a set of imputed points, which are then used for local trend learning via LatentODE.

To train the latentODE, we used the normal hours as training data. All together, the training data consisted of 40K time points, which were split into smaller sequences of 30 timesteps. In our LatentODE model, f_θ is a feed-forward neural network,



(a) Full data (b) Focused segment

Figure 5: Fault regions identified our framework. (a) Fault states across the entire μPMU data are highlighted in red. (b) An up-close view is presented.

with three 40-dimensional hidden layers and hyperbolic tangent as the activation function. For our ODE Solver, we used the `torchdiffeq` package [4, 5], and chose a fifth-order `dopri5` solver with relative tolerance 10^{-4} and absolute tolerance 10^{-3}. Our encoder network maps from data space to latent space using 2 generative layers with a latent dimension of 10. Our decoder network maps from latent space to data space using a one-layer feed-forward neural network with a 500-dimension hidden layer dimension. The batch size for training the entire LatentODE model is 10 and the learning rate is 0.01. The hyperparameters were tuned by performing a sweep on batch size and learning rate. Subsequently, we find that the performance of LatentODE is optimal when batch size is below 50 and the learning rate is below 0.03. To impute the missing points in the test data (i.e., the faulty hours), each missing 30 second interval (e.g., $t_j \dots t_{j+30}$) was passed through the trained LatentODE to obtain $x_j, \dots, x_{j+30}$. Finally, the imputed time points are then concatenated back to create a full dataset $\mathbf{X}$ to be passed to dd-CRP for segmentation.

To compare different interpolation methods, we tested our proposed method described in Section 3.3 against other baseline methods in an interpolation task. In this experiment, 10 60-second segments of data were held-out from 10 randomly selected time series and models are fit on the remaining data. Then, we used these models to impute the missing segments. We benchmarked our method against Bayesian Ridge, Random Forest and KNN. *Bayesian Ridge* [10] is a multivariate imputation strategy that infers missing values in each feature as a function of the other features . *Random Forest* is an iterative imputation strategy based on the MissForest algorithm [18]. *KNN* [13] is a k-Nearest-Neighbors based imputation strategy where the target is predicted by local interpolation of associated of the nearest neighbors. All baselines were implemented using the `scipy.interpolate` package [24]. The mean absolute error (MAE) and root mean squared error (RMSE) are shown in Table 1. Our method outperforms other benchmarks in this imputation task.

3.4 Segmentation

3.4.1 On μPMU Dataset. After imputation, we applied the dd-CRF model to the full μPMU data which is now regularly-sampled with any missing time points. We first fit a dd-CRP model to each hour of data that contained faults. Each dd-CRP model produced table assignments (i.e., segments) for its associated hours. We then used these table assignments as input to a CRP, as described in Section 2.2. Given the table assignments and the concatenated time series for the full dataset ($T = 59600$), the CRP model infers regions for the entire time series. For our experiments, we set the dispersion parameters to $\alpha = 10^{-6}$ and $\gamma = 10^{-4}$. This parameterization encourages the model to infer a small set of segments. Additionally, when performing clustering with CRP, we set the maximum number of regions to 2 to be able to discern between *baseline* regions and *faulty* regions. The choice of α, γ and maximum number of regions was carefully chosen after an extensive hyperparameter sweep.

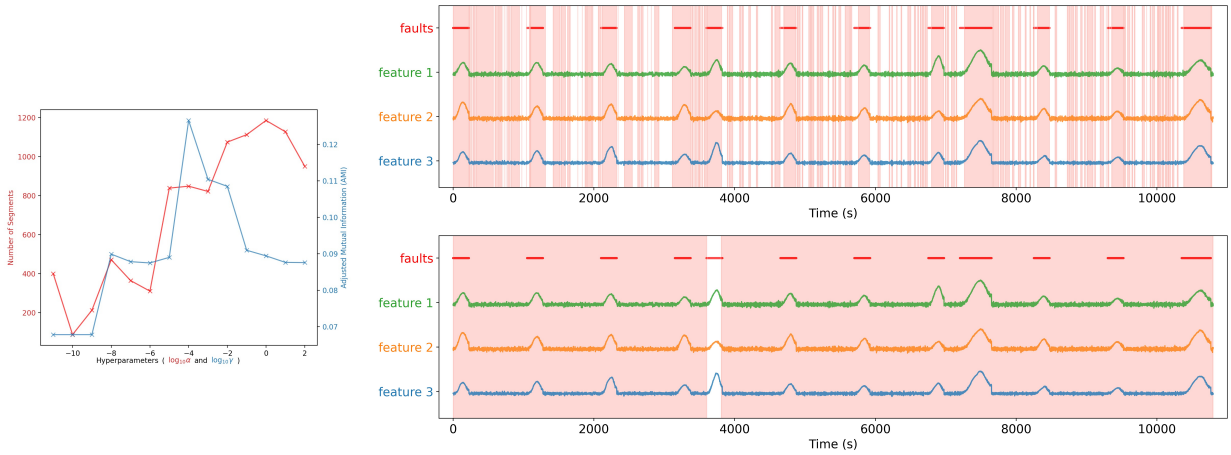
Table 2: Confusion matrix for real μPMU data.

		Prediction	
		Non-Fault	Fault
Actual	Non-Fault	42778	16632
	Fault	76	114

Figure 5(a) shows the concatenated data, along with the occurrences of faults and segmented regions predicted by the CRP. Here, our model is capable of capturing anomalies in the time series. We compare our segmentation results against the ground-truth labels received from our collaborators on this dataset. We take the red region to be the fault state and the non-colored region to be the normal state. The confusion matrix is shown as Table 2.

In the absence of ground-truth labels, one should consult with a subject matter expert to incorporate domain knowledge in the interpretation of these region assignments. By grouping anomalies together within similar time segments, experts can quickly attribute semantic meaning to these segments. Figure 5(b) shows, within the time window spanning $t = 36000$ to $t = 39600$, our model detects the sharp dips in all three features within the red region. This signifies that our model is capable of capture the unusual dynamics of our data and disambiguate those segments from normal behaviors.

3.4.2 On Synthetic Dataset. To better understand the effects of parameter α and γ, we perform a hyperparameter sweep and evaluate the performance of our dd-CRF model on the synthetic dataset. Here, we set $\boldsymbol{X}$ to be three hours of synthetic data, each with 3600 timepoints and 3 channels, totaling $3600 \times 3 = 10,800$ timepoints for testing. For both the dd-CRP and CRP, we used the same dataset and run the model configured with a specific set of hyperparameters, for 100 iterations.



(a) Hyperparameter sweep on α and γ.

(b) Regions discovered using $\alpha = 0.01$ and different γ. **(Top)** $\gamma = 10^{-4}$ **(Bottom)** $\gamma = 10^{-9}$

Figure 6: Hyperparameter sweep on three hours of synthetic data. (a) Parameter α and γ are tested on the same parameter range from 10^{-11} to 10^{2} (shown in log scale). The red line indicates the number of tables as α varies. The blue line indicates the Adjusted Mutual Information (AMI) as γ varies. (b) dd-CRP results on synthetic data based on different γ values. The γ on top is preferred over the γ on the bottom.

Table 3: Confusion matrix for synthetic data.

		Prediction	
		Non-Fault	Fault
Actual	Non-Fault	4172	3511
	Fault	394	2723

Impact of α. As mentioned in Section 2.2, α is a dispersion parameter in the dd-CRF that controls the probability that a new customer sits by herself rather than with a friend. As α increases, we expect to see more segments in the model, which is confirmed by our hyperparameter sweep results shown in Figure 6(a). As expected, we see an upward trend on the number of segments as α increases. We chose our α to provide a suitable number of tables to optimize our dd-CRP results.

Impact of γ. The relationship between the parameter γ in the CRP and the Adjusted Mutual Information (AMI) [23] is shown in Figure 6(a). AMI measures the agreement between two clusterings of the same data, and we use it as a score function to choose a suitable value for γ in our evaluation. We observe that AMI is highest when $\gamma = 10^{-4}$, as well as similar AMI values when $\gamma \in [10^{-4}, 10^{-2}]$, suggesting that there is a large range of values where performance is not affected by a particular choice of γ. The resulting clusterings for the best and the worst γ's (with respect to AMI) are shown in Figure 6(b). When $\gamma = 10^{-9}$, CRP combines most tables into the same state, indicating that the model has under-segmented the data and the selected parameter is too small to learn meaningful clusters.

4 SHORTCOMINGS AND FUTURE WORK

Our experiments on real and synthetic μPMU data show that dd-CRF is susceptible to noisy signals. In particular, noise in data and adversarial trends such as seasonality and electric load based on time-of-day can cause irregularities in data. This imposes a difficult challenge for our model, given that structured and unstructured noise can co-exist.

Since we are interested in the detection of incipient failures, the high number of false positives, if they correspond to time points that are pre-faults, is actually a desirable property of our method. If our method can detect that the system is entering a fault state before the occurrence of a real fault (as determined by state-of-the-art methods), then we can use this information to alert operators to take remedial actions in advance of noticeable degradation in system performance.

For future work, preprocessing methods that can further denoise the data and other formalisms that can incorporate domain knowledge about fault structures in μPMU's data would be promising directions to pursue.

5 CONCLUSION

Our proposed method deploys a two-phase imputation method to capture high-frequency fluctuations and low-frequency trends in μPMU data. Instead of simply casting out missing data, a robust imputation method allows for continuity between missing data, which allows for a wider class of machine learning or statistical methods to be applied for downstream analysis.

In this work, we presented an end-to-end framework that combines a data-driven deep learning technique of LatentODE with the Bayesian non-parametric clustering method dd-CRF. Our proposed method is capable of not only imputing realistic

trends and nonlinear dynamics in the data, but also inferring how these trends may be governed by underlying fault states. A predictive system built on these two capabilities can be used to provide advanced alerts to grid operators to improve situation awareness.

6 ACKNOWLEDGEMENT

We are grateful to: Hannah Burroughs (LLNL) for subject matter guidance; Indra Chakraborty (LLNL) and Pedro Sotorrio (LLNL) for assistance with data access; and Thomas Desautels for insight in synthetic data design.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and directly supported by the Grid Modernization Laboratory Consortium (GMLC) program.

REFERENCES

[1] David M Blei and Peter I Frazier. 2011. Distance dependent Chinese restaurant processes. *Journal of Machine Learning Research* 12, Aug (2011), 2461–2488.

[2] Jose Cadena, Priyadip Ray, and Emma Stewart. 2019. Fingerprint discovery for transformer health prognostics from micro-phasor measurements. ICML.

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366* (2018).

[4] Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. 2021. Learning Neural Event Functions for Ordinary Differential Equations. *International Conference on Learning Representations* (2021).

[5] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems* (2018).

[6] P. Dierckx. 1996. Curve and surface fitting with splines. In *Monographs on numerical analysis*.

[7] Emile Dusabimana and Sung-Guk Yoon. 2020. A survey on the micro-phasor measurement unit in distribution networks. *Electronics* 9, 2 (2020), 305.

[8] Nikolaj Ezhov and Svetozar Petrovic. 2018. Spline approximation, Part 1: Basic methodology. *Journal of Applied Geodesy* 12, 2 (2018), 139–155. https://doi.org/doi:10.1515/jag-2017-0029

[9] Dongwoo Kim and Alice Oh. 2011. Accounting for data dependencies within a hierarchical Dirichlet process mixture model. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 873–878.

[10] David J. C. MacKay. 1992. Bayesian Interpolation. *Neural Computation* 4, 3 (05 1992), 415–447. https://doi.org/10.1162/neco.1992.4.3.415 arXiv:https://direct.mit.edu/neco/article-pdf/4/3/415/812340/neco.1992.4.3.415.pdf

[11] H. Mohsenian-Rad, E. Stewart, and E. Cortez. 2018. Distribution Synchrophasors: Pairing Big Data with Analytics to Create Actionable Information. *IEEE Power and Energy Magazine* 16, 3 (2018), 26–34. https://doi.org/10.1109/MPE.2018.2790818

[12] Matías Núñez, Nadia Barreiro, Rafael Barrio, and Christopher Rackauckas. 2021. Forecasting virus outbreaks with social media data via neural ordinary differential equations. *medRxiv* (2021).

[13] L. E. Peterson. 2009. K-nearest neighbor. *Scholarpedia* 4, 2 (2009), 1883. https://doi.org/10.4249/scholarpedia.1883 revision #137311.

[14] Yulia Rubanova, Ricky TQ Chen, and David Duvenaud. 2019. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907* (2019).

[15] Alireza Shahsavari et al. 2017. A data-driven analysis of capacitor bank operation at a distribution feeder using micro-PMU data. In *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 1–5.

[16] Alireza Shahsavari, Mohammad Farajollahi, Emma M Stewart, Ed Cortez, and Hamed Mohsenian-Rad. 2019. Situational awareness in distribution grid using micro-PMU data: A machine learning approach. *IEEE Transactions on Smart Grid* 10, 6 (2019), 6167–6177.

[17] Alireza Shahsavari, Ashkan Sadeghi-Mobarakeh, Emma Stewart, and Hamed Mohsenian-Rad. 2016. Distribution grid reliability analysis considering regulation down load resources via micro-PMU data. In *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 472–477.

[18] Daniel J. Stekhoven and Peter Bühlmann. 2011. MissForest—nonparametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (10 2011), 112–118. https://doi.org/10.1093/bioinformatics/btr597 arXiv:https://academic.oup.com/bioinformatics/article-pdf/28/1/112/583703/btr597.pdf

[19] Emma Stewart, Anna Liao, and Ciaran Roberts. 2016. Open μPMU: A real world reference distribution micro-phasor measurement unit data set for research and application development. (2016).

[20] James H. Stock and Mark W. Watson. 2001. Vector Autoregressions. *Journal of Economic Perspectives* 15, 4 (December 2001), 101–115. https://doi.org/10.1257/jep.15.4.101

[21] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical Dirichlet Processes. *J. Amer. Statist. Assoc.* 101, 476 (2006), 1566–1581.

[22] Muhammad Usama Usman and M Omar Faruque. 2019. Applications of synchrophasor technologies in power systems. *Journal of Modern Power Systems and Clean Energy* 7, 2 (2019), 211–226.

[23] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research* 11 (2010), 2837–2854.

[24] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[25] Yuchen Wang, Matthieu Chan Chee, Ziyad Edher, Minh Duc Hoang, Shion Fujimori, Sornnujah Kathirgamanathan, and Jesse Bettencourt. 2020. Forecasting Black Sigatoka Infection Risks with Latent Neural ODEs. *arXiv preprint arXiv:2012.00752* (2020).

[26] Mohsen Ghalei Monfared Zanjani, Kazem Mazlumi, and Innocent Kamwa. 2018. Application of μPMUs for adaptive protection of overcurrent relays in microgrids. *IET Generation, Transmission & Distribution* 12, 18 (2018), 4061–4068.