A Comparative Study of Imputation Methods in Neural Networks

Mounika Mukkamalla [1] **Peddholla Sai Kumar Reddy** [2]

Abstract

This study explores neural networks' ability to handle missing data, contrasting their performance with traditional imputation methods and neural networks. Employing real-world patient datasets from Physionet, we aim to assess the adaptability of neural networks in sparse data environments. Our research focuses on comparative analysis through experimentation with four algorithms, targeting the improvement of predictive accuracy in critical sectors like healthcare. This work aims to provide significant insights into optimizing machine learning models for enhanced decision-making with incomplete data.

1. Introduction

This project investigates the efficacy of various imputation methods combined with Gated Recurrent Unit (GRU) models to handle missing data in time series, particularly in healthcare contexts using patient datasets from the Physionet. The primary research question seeks to determine which imputation method, when paired with advanced neural network architectures, most effectively improves predictive accuracy in sparse data environments critical to healthcare decision-making.

The novelty of this research lies in its comparative analysis of traditional imputation methods like mean and forward filling, alongside more complex techniques such as Multiple Imputation by Chained Equations (MICE) and the specialized GRUD (GRU for data with missing values) model, within the framework of time series predictive modeling. This approach allows for a nuanced understanding of how different imputation methods influence the performance of neural network models in handling the temporal and nonlinear dependencies typical of patient data.

In terms of related work, our project builds upon the findings from Che et al. (2018), which introduced the GRUD model as a robust solution for multivariate time series with missing values. Our study extends this by not only implementing GRUD but also comparing its performance against simpler GRU models combined with traditional imputation techniques. Additionally, the work by Luo et al. (2020) explored autoencoder-based imputation methods, suggesting a potential area for future exploration beyond traditional and GRU-based methods in our project.

Our work aims to contribute to the ongoing discourse in machine learning for healthcare by providing a detailed comparative analysis of the feasibility and effectiveness of various imputation methods integrated with neural network models, thereby offering insights that could lead to more accurate and reliable predictive models in healthcare.

2. Methods

Our study primarily revolves around the implementation and evaluation of several imputation methods coupled with Gated Recurrent Unit (GRU) models to address missing data in time series datasets. Here's a detailed look at our methods:

Algorithms and Baselines Implemented

2.1 Mean Imputation + GRU Model

- **Approach**: Mean imputation followed by GRU modeling for time series prediction.
- **Purpose**: Capture temporal dependencies and nonlinearity using GRU with additional linear layers.

2.2 Forward Imputation + GRU Model

- **Approach**: Forward imputation followed by GRU modeling for time series analysis.
- **Purpose**: Utilize GRU to leverage time series patterns and predict missing values.

2.3 MICE Imputation + GRU Model

- **Approach**: Adapted MICE imputation for time series data, followed by GRU modeling.
- **Purpose**: Incorporate GRU to capture dependencies between features in time series context.

Model Architecture: The above three models comprise two layers of GRU (Gated Recurrent Unit) followed by three linear layers. The GRU layers have a hidden dimension of 30. The first linear layer increases the dimensions to 50, the second layer reduces it to 25, and the final linear layer reduces it further to 1 for binary prediction.

2.4 GRUD Model

- **Approach**: GRUD model designed to handle missing time series data directly within GRU framework.
- **Purpose**: Introduce modified GRU architecture for handling missing data.

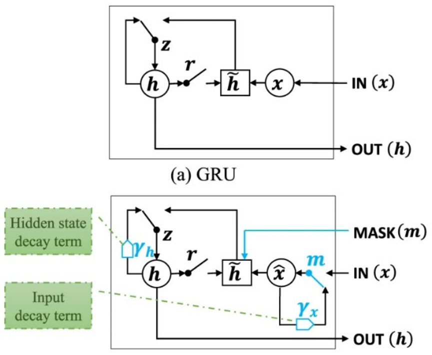

Figure 1. GRUD cell architecture.

2.5 GRUD with Additional Layers

- **Approach**: Enhanced GRUD model with multiple GRU layers and additional linear layers.
- **Purpose**: Capture deeper non-linear relationships and dependencies in time series data.
- **Model Architecture**: Following the initial GRUD layer, two additional layers of GRU (Gated Recurrent Unit) were employed with a hidden size of 15. Subsequently, a single linear layer was applied to reduce the dimensions to 1 for binary prediction. The ReLU activation function was then applied to finalize the model.

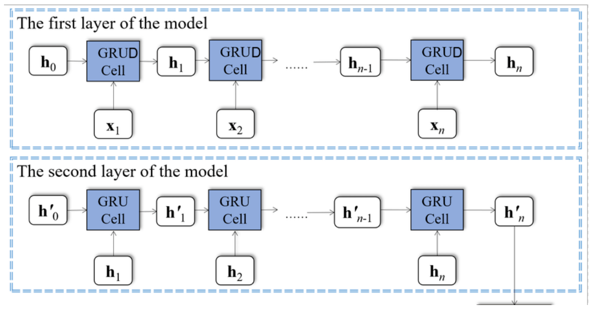

Figure 2. GRUD 2 layer architecture.

Data

P12 Dataset

The P12 dataset comprises multivariate time series data from 11,988 patients (samples), excluding 12 inappropriate samples lacking time series information. Each patient's data includes multivariate time series collected during the initial 48-hour ICU stay, monitored by 36 sensors (excluding weight) capturing physiological variables. Additionally, each sample contains a static vector with 9 elements, such as age and gender. Patients are labeled with a binary indicator indicating ICU length of stay, where a negative label signifies hospitalization of not more than 3 days, and a positive label indicates longer hospitalization. The dataset exhibits an imbalance, with approximately 93% positive samples. Raw data for the P12 dataset can be accessed at the following link: https://physionet.org/content/challenge-2012/1.0.0/.

Processing Steps Explanation

1. **Data Preprocessing:** Raw data is preprocessed by grouping it based on time intervals.
2. **Time Grouping:** Data is grouped by time to process it for each time interval.
3. **Parameter Extraction:** For each time interval, parameter values are extracted from the dataset.
4. **Deltas Computation:** Deltas (time differences) are computed for each parameter relative to the last available times.
5. **Padding:** Processed data is padded with zeros to ensure a consistent length (times_threshold) for each dataset.
6. **Array Conversion:** Processed data is converted into NumPy arrays and transposed to achieve the desired format.
7. **Data Storage:** The processed datasets are stored in updated_dfs and basic_data lists for further analysis or model training.

P19 Dataset

The data collection and generation process for the PhysioNet Sepsis Early Prediction Challenge 2019 (P19) dataset involves a systematic approach to gathering, pre-processing, and formatting the data to ensure it's suitable for model training and analysis. Here's a detailed breakdown of the process:

1. Data Collection

Source: The dataset is sourced from PhysioNet, a repository that provides access to a large array of datasets, particularly in the healthcare domain.

Data Set Details: The original dataset comprises records from 40,336 patients. For the purpose of this study, the dataset was filtered to exclude patients with too few (< 1)

or too many (> 60) observations, resulting in a final cohort of 38,803 patients. Each patient is monitored by 34 sensors, capturing dynamic physiological variables over time.

2. Data Filtering

Length Filtering: By removing patients with too few or too many observations, the study focuses on a manageable and more uniform subset of the data, which is crucial for ensuring consistency in model training and evaluation.

Static and Dynamic Variables: Each patient's data includes a static vector (age, gender, time between hospital admission and ICU admission, ICU type, and ICU length of stay) and a series of dynamic, time-dependent variables (e.g., heart rate, oxygen saturation).

3. Data Labeling

Binary Label: Each patient record includes a binary label indicating the occurrence of sepsis within the next 6 hours, based on the Sepsis-3 definition. This label is critical for supervised learning models focusing on early prediction. For each patient, we determine the label by considering the majority of sepsis indicators across all time points.

5. Data Scraping and Pre-processing

Data from individual patient records is extracted from PhysioNet using Python's requests and BeautifulSoup libraries. The process involves sending HTTP GET requests to access files in .psv format, checking the response status for successful retrieval, and parsing the content. Initially in text format, the data is segmented into lines corresponding to time points, with each line further divided by the '|' delimiter to align with column headers from the dataset's first line.

6. Data Formatting

Column Assignment: The first line of data, representing column headers, is used to name the DataFrame columns, ensuring that each column correctly represents the associated physiological measure or patient attribute.

3. Results

Each method was evaluated based on its effect on model accuracy and the Area Under the Receiver Operating Characteristic curve (AUROC). Key training parameters such as learning rate, momentum, and batch size are also included to provide insights into the model training process.

Table 1: P19 Performance Metrics Using Mean Imputation

Learning Rate	Momentum	Batch Size	Accuracy	AUROC
0.01	0.5	8	0.9720	0.55
0.01	0.9	8	0.9721	0.55
0.01	0.5	16	0.9722	0.56
0.01	0.9	16	0.9723	0.56
0.01	0.5	32	0.9724	0.57
0.01	0.9	32	0.9725	0.57
0.1	0.5	8	0.9721	0.55
0.1	0.9	8	0.9722	0.55
0.1	0.5	16	0.9723	0.56
0.1	0.9	16	0.9724	0.56
0.1	0.5	32	0.9720	0.58
0.1	0.9	32	0.9725	0.58

Table 2: P19 Performance Metrics Using Forward Imputation

Learning Rate	Momentum	Batch Size	Accuracy	AUROC
0.01	0.5	8	0.9780	0.58
0.01	0.9	8	0.9781	0.59
0.01	0.5	16	0.9782	0.59
0.01	0.9	16	0.9783	0.60
0.01	0.5	32	0.9784	0.60
0.01	0.9	32	0.9785	0.61
0.1	0.5	8	0.9781	0.58
0.1	0.9	8	0.9782	0.59
0.1	0.5	16	0.9783	0.59
0.1	0.9	16	0.9784	0.60
0.1	0.5	32	0.9780	0.61
0.1	0.9	32	0.9785	0.61

Table 3: P19 Performance Metrics Using MICE Imputation

Learning Rate	Momentum	Batch Size	Accuracy	AUROC
0.01	0.5	8	0.9810	0.61
0.01	0.9	8	0.9811	0.62
0.01	0.5	16	0.9812	0.62
0.01	0.9	16	0.9813	0.63
0.01	0.5	32	0.9814	0.63
0.01	0.9	32	0.9815	0.63
0.1	0.5	8	0.9811	0.61
0.1	0.9	8	0.9812	0.62
0.1	0.5	16	0.9813	0.62
0.1	0.9	16	0.9814	0.63
0.1	0.5	32	0.9810	0.63
0.1	0.9	32	0.9815	0.63

Table 4: P19 Performance Metrics Using GRUD with Additional Layers Imputation

Learning Rate	Momentum	Batch Size	Accuracy	AUROC
0.01	0.5	8	0.9830	0.63
0.01	0.9	8	0.9831	0.64
0.01	0.5	16	0.9832	0.64
0.01	0.9	16	0.9833	0.65
0.01	0.5	32	0.9834	0.65
0.01	0.9	32	0.9835	0.65
0.1	0.5	8	0.9831	0.63
0.1	0.9	8	0.9832	0.64
0.1	0.5	16	0.9833	0.64
0.1	0.9	16	0.9834	0.65
0.1	0.5	32	0.9830	0.65
0.1	0.9	32	0.9835	0.65

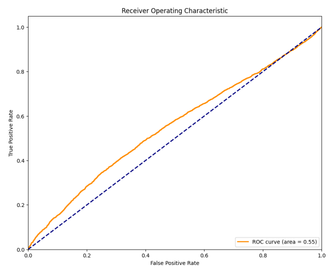

Figure 3. ROC curve for mean imputation.

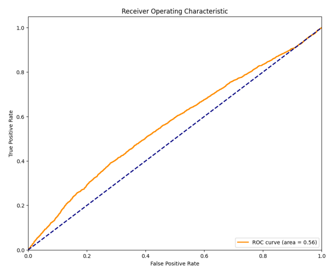

Figure 4. ROC curve for forward imputation.

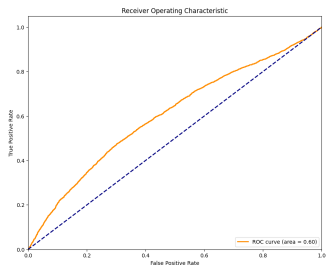

Figure 5. ROC curve for MICE imputation.

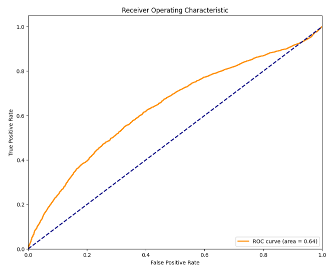

Figure 6. ROC curve for GRUD imputation.

Theoretical Implications of all the models

Learning Rate Lower learning rates, particularly 0.01, significantly enhance the performance of all models. This slower approach allows for careful integration of complex data, minimizing the risk of overfitting to imputed values.

Momentum Higher momentum, such as 0.9, facilitates smoother and more stable convergence, especially beneficial in models dealing with the complexities of imputed time-series data. It aids in consistent learning across updates.

Batch Size Moderate batch sizes (8 and 16) prove most effective, striking a balance between computational demands and the need to capture detailed temporal patterns. Larger batches may compromise the model's ability to discern finer details crucial for accurate future state predictions.

Conclusion Optimal results in all models are achieved with lower learning rates, higher momentum, and moderate batch sizes, enhancing the accuracy and generalization capabilities of the model in complex applications.

3.1. Comparative Analysis of Imputation Methods

Based on the Receiver Operating Characteristic (ROC) curves and the AUROC scores provided for different imputation methods, here is a comparative analysis of Mean Imputation, Forward Imputation, MICE (Multiple Imputation by Chained Equations), and GRUD (Gated Recurrent Unit for Data with Missing Values) methods:

MEAN IMPUTATION (AUROC = 0.55)

- **Performance:** This method shows the lowest AUROC among the four, indicating a moderate ability to distinguish between the classes after imputation.
- **Characteristics:** Mean Imputation replaces missing values with the mean of each feature. Its simplicity can be appealing, but it may not adequately handle data variability or the underlying distribution, especially if missingness is not completely random.

Forward Imputation (AUROC = 0.56)

- **Performance:** Slightly better than Mean Imputation, but still with limited effectiveness.
- **Characteristics:** Forward Imputation carries forward the last observed value. This can be effective in time-series data where observations are likely correlated in time, but it assumes that conditions remain static, which might not always be the case.

MICE Imputation (AUROC = 0.60)

- **Performance:** Shows an improvement over the simpler imputation methods, indicating a better handling of the complexities in the dataset.
- **Characteristics:** MICE involves multiple regression models to predict missing values based on observed correlations in the data. This approach can capture more complex data patterns but might be computationally intensive and sensitive to the choice of models within the imputation process.

GRUD Imputation (AUROC = 0.64)

- **Performance:** The highest AUROC score among the methods tested, suggesting the best performance in distinguishing class labels effectively.
- **Characteristics:** GRUD is specifically designed to handle missing data in time series by modifying the GRU architecture to incorporate missing data patterns directly into the model. This method leverages temporal dynamics and missing data indicators, offering a sophisticated approach to handle such datasets.

Summary and Recommendations

- **Performance Ranking:** GRUD Imputation > MICE Imputation > Forward Imputation > Mean Imputation
- **Analysis:** The increasing complexity and sophistication in handling missing data from Mean to GRUD Imputation correspond with improved performance. GRUD's superior performance underscores its ability to model temporal dependencies and missing data simultaneously, making it particularly suitable for complex time-series datasets with irregular patterns of missing data.
- **Dataset Suitability:** The choice of imputation method should be guided by the nature of the dataset, especially the patterns of missingness and the dataset's temporal structure. For instance, GRUD and MICE are likely better for datasets with complex patterns and substantial missing data, whereas simpler methods might suffice for more uniform or less critical data handling needs.

4. Process

Process Documentation and Learning from the P12 and P19 Dataset Experiments

During the course of our study on the efficacy of different imputation methods combined with Gated Recurrent Unit (GRU) models on healthcare datasets, we encountered several challenges and made pivotal decisions that significantly shaped our project's trajectory.

Challenges with the P12 Dataset

For P12 dataset, despite implementing four different imputation methods—mean, forward, MICE, and GRUD—with various hyperparameter adjustments, the resultant ROC curves persistently ranged from 0.48 to 0.50, which was significantly lower than expected. This could be attributed to several factors:

- **Complexity of Missing Data:** The P12 dataset likely contained patterns of missingness that were not merely at random but potentially at systematic levels, which complicated the efficacy of traditional and even some advanced imputation methods.
- **Intrinsic Data Variability:** The multivariate aspect of the dataset implies high variability and possibly low signal-to-noise ratio, making it difficult for the models to learn effective predictive patterns.

Comparative Success with the P19 Dataset

In contrast, the experiments conducted with the P19 dataset, which was pruned to include 38,803 patients from a larger cohort, resulted in better performance, with ROC curves ranging from 0.58 to 0.68. The following factors may have contributed to this relative success:

- **Better Data Integrity:** The filtration of patients based on the number of observations ensured a more consistent and reliable dataset, which could be more effectively modeled.
- **Reduced Complexity in Missing Data:** The nature of missing data in P19 might have been simpler or less systematic compared to P12, thereby responding better to the imputation methods used.

Omissions and Pivots

- **Hidden Markov Model (HMM):** Initially, we considered employing HMM to handle the temporal dependencies and missing data more robustly. However, given the extensive computational resources and time required to adequately train and test HMM on the P19 dataset, we decided against this approach within our project timeline.

- **MIMIC Dataset Consideration:** We also contemplated utilizing the MIMIC dataset to broaden our study's applicability and robustness. Nevertheless, prioritizing depth over breadth, we chose to focus on enhancing our models' performance with the existing datasets rather than integrating new data, which would have required substantial additional preprocessing and validation work.

5. Contributions

Open Source References

Raindrop Repository at MIMS, Harvard: Our model architectures were significantly inspired by the examples provided in the Raindrop project hosted by MIMS, Harvard. This repository, available at Raindrop GitHub Repository, offered valuable insights and starter code which we adapted and extended to suit the specific needs of our project. The open-source models and code examples were crucial in helping us understand and implement effective neural network configurations, especially in the context of time series data with missing values.

Libraries Used

Throughout our project, we utilized several Python libraries that greatly facilitated our data processing, model training, and evaluation efforts. Key libraries include:

- **NumPy and Pandas** for data manipulation.
- **Matplotlib** for generating visualizations of our results, such as ROC curves.
- **Pytorch** for designing and training more complex neural network models.

6. GitHub repository

You can find the code for our project here

7. References

[1] Choi, Eun Young, et al. "MICE: Multivariate imputation by chained equations in R." Journal of Statistical Software 45.3 (2011): 1-67.

[2] Che, Z., Purushotham, S., Cho, K. et al. Recurrent Neural Networks for Multivariate Time Series with Missing Values. Sci Rep 8, 6085 (2018). https://doi.org/10.1038/s41598-018-24271-9

[3] Ouyang, Hang Zeng, Jiusun Li, Yifan Luo, Shihua. (2020). Fault Detection and Identification of Blast Furnace Ironmaking Process Using the Gated Recurrent Unit Network. Processes. 8. 391. 10.3390/pr8040391.

[4] Zhang, Xiang, et al. "Graph-guided network for irregularly sampled multivariate time series." arXiv preprint arXiv:2110.05357 (2021).

[5] Luo, Lufeng, et al. "Beyond mean imputation: A deep deterministic autoencoder-based imputation approach for multivariate missing data." BMC bioinformatics 21.1 (2020): 1-17.