2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) | 979-8-3503-1855-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/SmartGridComm60555.2024.10738098

A Deep Learning Approach for In-Network Synchrophasor Missing Data Recovery Using Programmable Network Switches

Xiaoyu Guo*, Yanfeng Qu*, Gong Chen†, Dong Jin*
*Electrical Engineering and Computer Science, University of Arkansas, Fayetteville, AR, USA
{xsguo, yqu, dongjin}@uark.edu
†Computer Science, Illinois Institute of Technology, Chicago, IL, USA
gchen31@hawk.iit.edu

***Abstract*—Phasor measurement unit (PMU) networks deliver accurate and timely measurements, which is essential for managing today's electric power systems. To ensure data quality and enhance the cyber-resilience of PMU networks against malicious attacks and data errors, this study presents an online PMU missing data recovery scheme by leveraging P4 programmable switches. The data plane incorporates a customized PMU protocol parser that abstracts the necessary payload data for recovery. Recovery processes are executed in the control plane using a pre-trained machine learning model. Both traditional and advanced ML models, such as transformer and TimeGPT, are explicitly employed for data prediction. This approach ensures rapid and precise data recovery. Performance evaluations focus on recovery speed and accuracy, using a real dataset from a campus microgrid. With 20% missing PMU data, the mean absolute percentage error for voltage magnitude is 0.0384%, and the phase angle error discrepancy is approximately 0.4064%.**

***Index Terms*—Phasor Measurement Unit, Machine Learning, Programmable Network, P4, Smart Grid**

I. Introduction

A Phasor Measurement Unit (PMU) is a device used in power systems to measure voltage, current, and frequency with high precision and time synchronization. The measurements from PMUs are pivotal for dynamic event analysis and disturbance identification [1], crucial for power grid state estimation, significantly impacting critical security and electricity market decisions [2]. However, missing PMU data often arises from various error sources, including device failures, communication faults, GPS synchronization issues, electromagnetic interference, configuration errors, and cyber attacks. The missing data can significantly degrade the accuracy of grid state estimation due to reduced system observability. Therefore, precise prediction and recovery of missing values is vital for optimal state estimation. Additionally, given the high temporal resolution of PMU measurements, the speed of the recovery solution is also a crucial factor to consider.

Many existing work on PMU data recovery utilizes mathematical approaches, such as matrix completion [3], tensor decomposition [4], and cubic spline interpolation [5]. These recovery schemes take a centralized approach and are implemented in the application layer at the receiving end, typically the substation phasor data concentrator (PDC) or the control center. To significantly enhance recovery speed, we aim to adopt an *in-network* and *decentralized* approach, ideally within communication network devices, to inspect packets and recover missing data in real time. However, conventional network devices such as routers, switches, and substation gateways are limited to processing standard protocols like TCP/IP only and lack the capability to comprehend the semantics of PMU headers and data payloads.

To address these challenges, we employ programmable networks to develop an efficient PMU missing data recovery scheme. Our approach is decentralized, utilizing the programmable network devices to deploy the recovery scheme. Specifically, we harness the capabilities of P4, or programming protocol-independent packet processors, enabling *real-time* and *customizable* control over the packet processing pipeline for rapid detection and recovery of missing packets [6]–[8]. Furthermore, our approach integrates deep learning models with P4 switches to achieve high accuracy.

Utilizing programmable P4 switches offers several advantages: (1) The fully programmable pipeline allows for the design of parsers tailored to PMU protocols, providing fine-grained control over PMU packet processing behavior. (2) Data packets are transmitted and processed at a line rate, minimizing latency impact, while detection and mitigation occur within the communication network rather than at the receiving end, significantly reducing recovery time. (3) The solution necessitates only incremental changes to existing PMU networks, simply involving the addition of P4 switches at aggregators like PDCs.

We leverage the capabilities of programmable P4 switches to recover missing PMU data during packet transmission in the communication network. P4 switches parse incoming PMU packets in real time, extracting information from both packet headers and payloads, including power magnitude, angle measurements, and timestamps. Our design features a bi-layer architecture on P4 switches: packet parsing and miss data detection are handled on the data-plane, while the missing data recovery scheme is managed on the control-

Authorized licensed use limited to: University of Arkansas. Downloaded on April 29,2025 at 00:02:46 UTC from IEEE Xplore. Restrictions apply.

plane. This recovery scheme incorporates pre-trained deep learning models for rapid prediction of missing values within milliseconds.

In conventional terms, data recovery involves reconstructing missing data using mathematical or statistical methods based on existing data points. In this paper, the machine learning models we investigate are predictive in nature, meaning they use available data to forecast missing values. For clarity, we use the terms missing data "prediction" and "recovery" interchangeably. Several studies have explored PMU missing data recovery using machine learning models, including decision trees [9], Seq2Seq, and LSTM [10]. Our recovery algorithm is based on a transformer [11] trained on a dataset from a campus microgrid. This transformer model can swiftly make predictions, achieving millisecond-level speeds without compromising accuracy. By incorporating attention layers, the transformer effectively captures relationships between features and encodes positional information for each input data. Our experiments demonstrate that the Vanilla Transformer model outperforms other models trained from scratch, and the pre-trained TimeGPT model achieves the best overall accuracy despite the long inference time.

We implement the missing data recovery scheme on a software-based P4 switch named BMv2 [12] and evaluate the scheme in the Mininet testbed [13], [14]. Our evaluation utilizes data from a real PMU network deployed on a campus microgrid. The results demonstrate remarkable efficiency, with our in-network approach achieving an average recovery and delivery time of 2.5 milliseconds (ms) for missing PMU packets, compared to 1.2 ms for non-missing packets. Additionally, our scheme exhibits high accuracy, successfully recovering missing data in all test scenarios with minimal error, e.g., even with 20% missing PMU data, the mean absolute percentage error for voltage magnitude is 0.0384%, and the phase angle error is approximately 0.4064%.

The main contributions of this work are summarized as follows:

- Leveraging P4 programmable switches, we enabled real-time and early recovery of missing data within the PMU communication network itself.
- We conducted a comparative evaluation of different deep learning models for missing data recovery, identifying the Vanilla Transformer model as the top performer when implemented on P4 switches.
- Our evaluation demonstrates the effectiveness of our scheme in accurately predicting missing values and maintaining consistent performance across packet loss rates ranging from 1% to 20%.

The remainder of the paper is structured as follows. In Section II, we describe the related works on time-series missing data prediction. Section III presents the design of our AI-assisted PMU missing data recovery scheme on P4 switches. In Section IV, we explain the various deep learning models for PMU missing data prediction. Section V presents the experimental evaluation results, and Section VI concludes the paper with future works.

II. Related Work

PMU data is structured as a time series comprising a sequence of data points indexed chronologically, with measurements taken at regular intervals. Consequently, we can generalize the PMU missing data recovery problem as a time series forecasting problem.

One method of time-series forecasting is the Kalman filter proposed in 1960 [15]. This recursive algorithm utilizes historical measurements, accounting for noise and inaccuracies, to generate estimated one-step-ahead predictions. Kalman's pioneering work has paved the way for various state space time-series forecasting models, such as dynamic linear models [16] and balanced state space models [17].

Another method for time-series forecasting is autoregressive (AR) models [18]. AR models assume that present values are influenced by past values, relying solely on previous data to anticipate future trends. A basic example of an AR model is the linear regression model.

In recent years, neural network-based time-series forecasting models have become increasingly popular. Lim et al. conducted a comprehensive survey evaluating various deep learning models for time-series forecasting [3]. Deep neural networks have the advantage of automatically learning data feature representations without manual feature engineering. Additionally, these models typically offer open-source frameworks, simplifying the training process and allowing for potential customization of network components, such as the loss function.

Specific methods are proposed for PMU missing data. Liao et al. proposed ADMM, or alternating direction method of multipliers, which is a low-rank matrix-completion based approach [19]. However, ADMM uses the Lagrangian multiplier approach to solve a matrix optimization problem, and the computational complexity of ADMM is very high. In contrast, our recovery model avoids complex matrix operations, resulting in significantly lower computational complexity than ADMM. Another approach involves tensor decomposition, as utilized by Osipov et al., who organize PMU data into three-dimensional tensors based on time, location, and variable type [4]. Similar to ADMM, Osipov et al. decompose multiple three-dimensional tensors, which requires substantial computational resources. Moreover, the computational time required for tensor decomposition ranges from 3 to 6 seconds, making it slower than the deep learning model we employ.

Machine learning approaches have also been introduced to solve the PMU missing data problem. Yang et al. proposed a C4.5 decision tree based approach [9]. Their approach is implemented in the data center, so the model has all the necessary historical PMU data. In contrast, our approach is restricted within the P4 data plane of the communication network. Cheng et al. proposed a forecasting model that leverages Seq2Seq and LSTM with a prior knowledge matrix integrated into the attention mechanism, which preserves the correlations within the PMU data [10]. They also introduced the technique of magnitude trend decoupling of the residual forecast, making the model more resistant to

Authorized licensed use limited to: University of Arkansas. Downloaded on April 29,2025 at 00:02:46 UTC from IEEE Xplore. Restrictions apply.

noisy signals. The model Cheng et al. proposed includes two different LSTM networks, one measures the magnitude component and the other measures the trend component.

III. AI-assisted PMU Missing Data Recovery Scheme Design on P4 Switches

We leverage the capabilities of P4 switches to develop a real-time missing data recovery mechanism during packet transmission in the communication network. Figure 1 illustrates the bi-layer design architecture of our AI-assisted PMU missing data recovery scheme on P4 switches. The PMU packet parsing and miss data detection is developed on the data-plane of the P4 switches and the missing data recovery scheme is developed on the control-plane of the P4 switches.

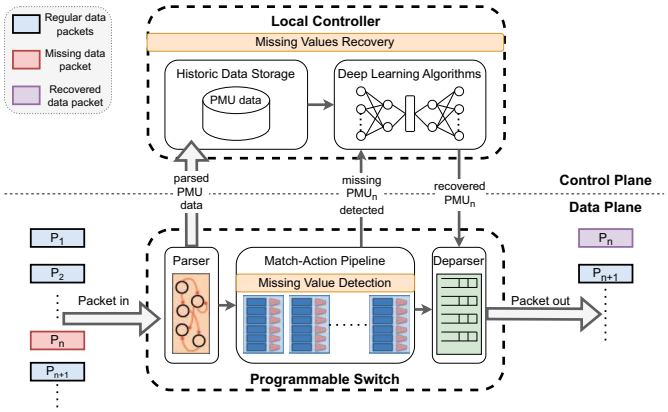

Fig. 1. AI-assisted PMU missing data recovery scheme on P4 switches.

The programmability of P4 switches enables the design of custom parser and deparser modules tailored to the format of PMU data packets. As shown in Figure 1, every packet arriving at input ports is parsed at the line rate, extracting features not only from the packet headers (e.g., IP addresses, port numbers) but also the payload (e.g., power magnitude, angle measurements, and timestamp). Because our deep learning based recovery algorithm requires 24 previous data points for each PMU data flow as input, we continuously aggregate and transmit the parsed angles and magnitudes data to the control plane, adopting a first-in, first-out method to ensure that only the most recent 24 data points are retained.

Following packet parsing, our match-action pipeline enables real-time detection of missing data. Utilizing the TimeTag in accordance with the IEEE C37.118 protocol [20], each synchrophasor measurement is timestamped, enabling a threshold-based detection algorithm. If the arrival time difference between two consecutive packets exceeds a predefined window, a missing data event is generated, prompting communication with the control-plane for data recovery initiation. While the paper focuses on missing data recovery, we opted for a straightforward detection mechanism for proof of concept. However, our modular design allows for future deployment of more advanced detection models within the data plane, such as adaptive thresholds or decision trees, to enhance detection accuracy.

Once triggered, the recovery mechanism prompts the data-plane to transmit a digest message to the control-plane, indicating the number of missing packets. Leveraging historical data aggregated from the parser, the control-plane inputs them into pre-trained deep learning models to predict the missed PMU data. The predicted values are then formatted into a PMU data packet with the appropriate timestamp and transmitted back to the data-plane via the deparser, effectively recovering the missing data with a new packet (i.e., the purpose packet in Figure 1). Contrasted with the P4 data-plane, the local controller boasts a general-purpose CPU, larger memory, and a Linux OS kernel, enabling the execution of advanced recovery algorithms. We have evaluated various deep learning models for the recovery process. The detailed design of these recovery algorithms will be discussed in the next section.

By integrating both the detection and recovery algorithms within the P4 switch, our architecture significantly minimizes communication overhead between these phases. This modular approach offers a flexible framework for P4-based solutions, with the detection algorithm operating in the data plane and the more complex recovery algorithm handled in the control plane. This allows for parallel processing, thereby enhancing the overall packet processing speed.

Using P4 programmable switches enables an in-network solution to address missing data recovery challenges. Through a decentralized approach that separates the detection and recovery phases and integrates all algorithms within the P4 switch, our solution minimizes communication overhead and achieves low latency. Pre-trained deep learning models deployed in the P4 network enable rapid prediction of missing values in just a few milliseconds. Furthermore, P4's support for user-defined protocols allows for direct customization of packet headers within the network, eliminating the need for parsing and analyzing PMU data packets at the receiving end, like a PDC or control center.

IV. Missing Data Prediction Models

Our architecture includes two phases. First, missing data is detected in the P4 data plane. Once missing data is detected, the P4 switch communicates with the control plane via digest messages, and the control plane uses the pre-trained model to predict the values of the missing data packet. Before we move to the BMv2 implementation of our design, we want to test the effectiveness of different AI models for missing data prediction.

A. Feature Preprocessing

By observing the distribution of the PMU data, we discover that the distributions of the angles and magnitudes follow a very similar shape. To investigate the data distribution further, we perform two statistical tests of correlation.

First, we compute the Spearman's rank correlation coefficients [21], which measures the rank coefficient between two variables. The coefficient depicts if there exists a monotonic relationship between the two variables, where the variables move in the relatively same direction, but not necessarily at

Authorized licensed use limited to: University of Arkansas. Downloaded on April 29,2025 at 00:02:46 UTC from IEEE Xplore. Restrictions apply.

a constant rate. We compute the Spearman's coefficient between every two magnitude measurements, and we see that all coefficients are in the range of $(0.98, 1)$, meaning that every single magnitude measurement has a strong monotonic relationship with all the other magnitude measurements in the dataset. That is, all magnitude measurements move in the same relative direction.

Furthermore, we compute the Pearson's correlation coefficient [22] between the magnitudes. This correlation shows the linear relationship between two variables, namely if the two variables move in the relative same direction at a constant rate. Similar to Spearman's coefficients, Pearson's coefficients between every two magnitude measurements fall into the range of $(0.98, 1)$, meaning there exists a strong linear relationship between any two magnitude measurements within our dataset.

Lastly, because all the angle measurements are periodic with a relatively similar shape, we transform the different angle measurements by a few time steps, and the shapes of the angle measurements overlap each other. After the transformation, we then compute both Spearman's and Pearson's correlation of every two angle measurements, and all coefficients fall in the range of $(0.99, 1)$, meaning that all angle measurements after the time-step transformation have strong monotonic and linear relationships with each other.

Using this knowledge, we can train the machine learning model with one pair of (magnitude, angle). In the inference phase, for each pair of missing PMU values, we can simply run the model multiple times with the appropriate input. This approach saves training time and decreases the size of the model, making it more suitable for in-network implementation.

B. Data Normalization

Because the dataset does not contain extreme outliers, we normalize our training data based on the z-score to ensure that the feature distributions have a mean of 0 and a standard deviation of 1. The mean and the standard deviation are calculated based on the training dataset. The normalization equation is defined as follows:

$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean value of the whole training dataset, and σ is the standard deviation.

C. Baseline model

For the baseline model, we choose to predict the missing value to have "no change" from the previous value. That is, if data x is missing, we predict the values of x to be the same values as data $x - 1$. This will naturally produce good results for single-step prediction, because the value differences between continuous PMU data are very small. If multiple values are missing consecutively, the performance of the baseline model will significantly decrease.

D. Deep Learning Models

Table I lists the deep learning models we have explored and the number of trainable parameters for each model, as well as the inference time for a single prediction by each model. The numbers of parameters determine the training time of each model. Despite the transformer model having more parameters than the LSTM model, training the transformer is quicker than training the LSTM due to the former's parallelizability. It can utilize GPUs during the training phase to process more data in a shorter time period. All the models are trained with 2 GPUs, while the testing is done with the Intel Xeon Gold 5220 CPUs at 2.20GHz.

- Multi-Layer Perceptron: fully connected neural network with three hidden layers.
- LSTM [23]: recurrent network with a cell state that retains long term memory.
- ResLSTM: LSTM inside a residual learning framework [24] that is fitted to the residual mapping.
- Transformer [11]: network with a self-attention layer and positional embedding.
- ResTransformer: transformer inside a residual learning framework that is fitted to the residual mapping.
- Autoformer [25]: transformer with a decomposition layer and replaces multi-head attention with an auto-correlation mechanism.

E. Transformer

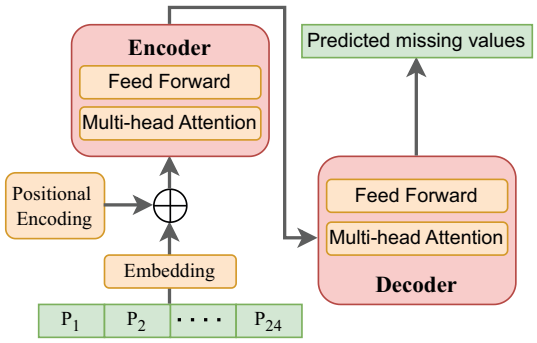

Fig. 2. High-Level architecture of the transformer model.

We choose to implement the Vanilla Transformer model in the software P4 switch, BMv2. The high-level architecture is shown in Figure 6. The transformer proposed by Vaswani et al. [11] introduces a self-attention layer and positional embedding. The transformer model, unlike RNN and LSTM, processes the input dataset as a whole, avoiding the issue of long dependencies. Vaswani et al. [11] introduced scaled dot-product attention and multi-head attention in their paper. These mechanisms, along with positional embedding, capture relationships between data from different time steps. Furthermore, the transformer's architecture allows for parallel computation, reducing training time by avoiding recursive computation.

One important feature of a transformer is the positional encoding, which is concatenated to the embedded input.

Authorized licensed use limited to: University of Arkansas. Downloaded on April 29,2025 at 00:02:46 UTC from IEEE Xplore. Restrictions apply.

Unlike RNNs, where input features are processed sequentially, the transformer treats input features with no positional order. Positional encoding ensures the correct order of input data, facilitating parallel processing and faster computation compared to LSTM models.

The transformer also has a self-attention mechanism, where it can capture long-term dependencies, meaning that historical information is less likely to be lost. The MLP model only takes the previous three data points as the input, so it retains no long-term dependencies. The LSTM models retain some long-term dependencies using the memory cells. However, they can still lose these long-term dependencies when processing extremely long sequences due to the forget gate becoming overly dominant with long input sequences. We theorize that the transformer will perform best from all the models due to its positional encoding and self-attention mechanism, which makes the transformer capable of retaining long-term dependencies.

F. TimeGPT

We also investigate TimeGPT [26], a generative pre-trained transformer developed by Garza et al. Pre-trained on extensive time-series data, TimeGPT possesses zero-shot learning capabilities, enabling predictions without additional training. The authors recommend fine-tuning with two seasonal historical data for optimal results. Employing TimeGPT for single time-step prediction, using the previous three time-steps as input, we find that post-fine-tuning, TimeGPT surpasses all other AI models tested and achieves performance comparable to the baseline.

However, the current version of TimeGPT has several limitations. Firstly, a portion of the TimeGPT architecture is proprietary and not open-sourced, hindering full transparency and customization. Additionally, parameters after fine-tuning cannot be saved, necessitating fine-tuning for each dataset submission to achieve optimal performance. Moreover, there is a cost quota associated with using the model, calculated based on input, output, and fine-tuning token usage. Once the quota is exhausted, accurate billing management is necessary to continue using the model. Lastly, our experiments revealed that predicting a single time-step using TimeGPT averages around 3 seconds, which does not meet the speed requirements for real-time data recovery. While we recognize TimeGPT's great potential and anticipate improvements in its limitations over time, we opt against implementing it in our P4 framework for evaluation.

V. Evaluation

A. PMU Datasets

This research utilizes a dataset derived from real-time measurements collected by PMUs within the IIT campus microgrid [27]. The dataset includes parameters, such as time (UTC), voltage magnitude, phasor angle, frequency, and frequency deviation, with measurements recorded approximately every 17 milliseconds. It encompasses data from 12 PMUs. For the evaluation, data from the first PMU was used for both training and testing.

B. Evaluation Metrics

For model evaluation, we chose the mean absolute percentage error (MAPE) metric to evaluate the predicted magnitudes. We choose MAPE because it tells us how much the predicted values deviate from the ground truth in percentage terms relative to the ground truth values. The metric is defined below:

$$\text{MAPE}(\%) = \frac{1}{n}\sum_{t=1}^{n}\frac{|A_t - F_t|}{|A_t|} * 100\% \tag{2}$$

We use the mean absolute error (MAE) metric to evaluate the predicted angles. We choose MAE because the angles can have a ground truth of 0 degrees, meaning the MAPE metric is no longer suitable due to denominators of zero. The metric is defined below:

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|A_t - F_t| \tag{3}$$

In both equations 2 and 3, A_t indicates the ground truth values and F_t indicates the forecast or predicted values.

C. Experimental Results and Analysis

1) Speed Analysis: We focus on the inference time for each model as they are pre-trained before implementation on the P4 switches. Models like Transformers and LSTMs, with more trainable parameters, exhibit longer inference times than MLP, ranging from 0.5 to 1 milliseconds. The Autoformer model requires about 1 to 2 milliseconds per prediction. TimeGPT, however, relies on communication with a cloud server provided by Nixtla, resulting in an inference time of up to 3 seconds per prediction.

TABLE I
COMPARISON OF TRAINABLE PARAMETERS AND INFERENCE TIME FOR SINGLE PREDICTIONS ACROSS DEEP LEARNING MODELS.

Models	# of Parameters	Inference Time (ms)
Baseline	0	0.0008
MLP	5,900	0.0505
LSTM	11,308	0.5151
ResLSTM	11,308	0.5183
Transformer	25,343	0.5344
ResTransformr	25,343	0.5350
Autoformer	25,944	1.0322
TimeGPT	N/A	3043.2

Table I reports the prediction time and complexity of each machine learning model. ResNets maintain the total number of trainable parameters due to residual learning, which adds an identity mapping layer requiring no additional training. Inference time generally correlates with parameter count, with most models averaging around 0.5 milliseconds. The Autoformer exhibits longer inference times, comparable to TimeGPT, due to its incorporation of the decomposition layer and auto-correlation mechanism, demanding increased computational resources during prediction.

The number of trainable parameters for TimeGPT is not publicly disclosed, and training the model is unnecessary for making predictions. Each prediction requires establishing a communication channel with the TimeGPT server

Authorized licensed use limited to: University of Arkansas. Downloaded on April 29,2025 at 00:02:46 UTC from IEEE Xplore. Restrictions apply.

and transmitting the requisite fine-tuning data. However, the parameters of the fine-tuned model cannot be stored locally, necessitating fresh fine-tuning for each prediction. Additionally, users are initially granted free credits upon TimeGPT sign-up, with costs contingent on input, output, and fine-tuning token usage. Once these credits are depleted, users must provide billing information for future token usage. While TimeGPT offers powerful time-series forecasting capabilities, its integration into our architecture would compromise processing speed.

The detection and recovery algorithms are executed within BMv2, a software-based P4 switch utilized for testing packet-processing behaviors. Initially, the P4 program undergoes compilation into a JSON representation, distinct from the static BMv2 source code, which is then loaded during runtime. The missing data prediction performance is identical to the evaluation from Section V-C2.

BMv2 serves as a tool for developing, testing, and debugging P4 data plane and control plane programs. Although the millisecond latency in this paper is sufficient for most real-time PMU applications, the applications running on P4 hardware switches are much faster than BMv2. As a result, we have not conducted end-to-end speed analysis in this paper. For future endeavors, we aim to implement our algorithms directly on P4 hardware to demonstrate further speed and accuracy improvement.

2) Accuracy Analysis: We report the MAPE of the magnitudes in Figure 3 and the MAE evaluation for angles in Figure 4. Our models are trained with the data collected by the first PMU within the IIT campus microgrid. The dataset contains 2,393,026 data packets, and we use a 70-30 train-test split.

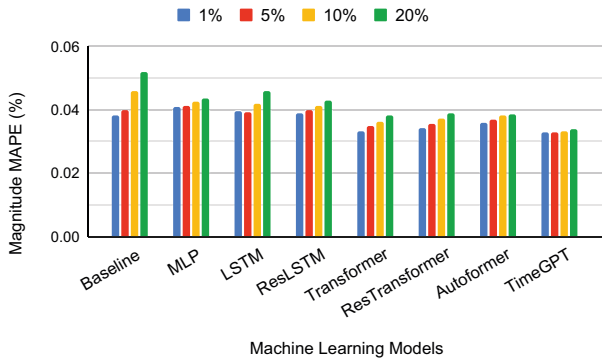

Fig. 3. MAPE evaluation of the predicted magnitudes under 1%, 5%, 10%, and 20% missing data rates.

Based on our evaluation results, the Vanilla Transformer model demonstrates superior performance compared to all other models except TimeGPT. With average magnitude MAPEs of 0.0357% and angle MAEs of 2.9381, we chose to implement the Transformer model in the P4 control plane. Notably, TimeGPT exhibits the highest performance across all models, with average magnitude MAPEs of 0.0331% and angle MAEs of 2.4333 despite its long inference time as reported earlier.

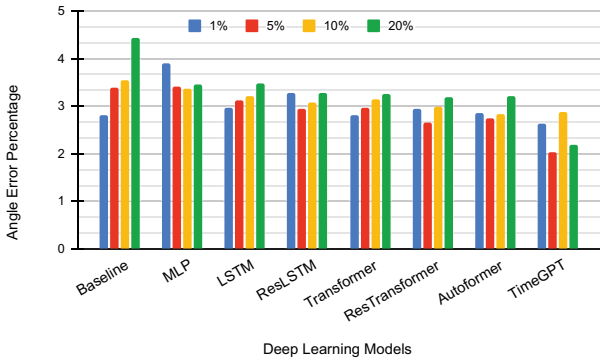

Fig. 4. MAE evaluation of the predicted angles under 1%, 5%, 10%, and 20% missing data rates.

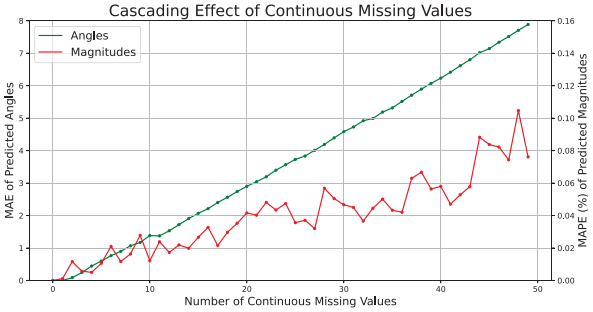

Fig. 5. Cascading effect when using predicted values for future prediction, evaluated from 1 to 50 continuous missing values.

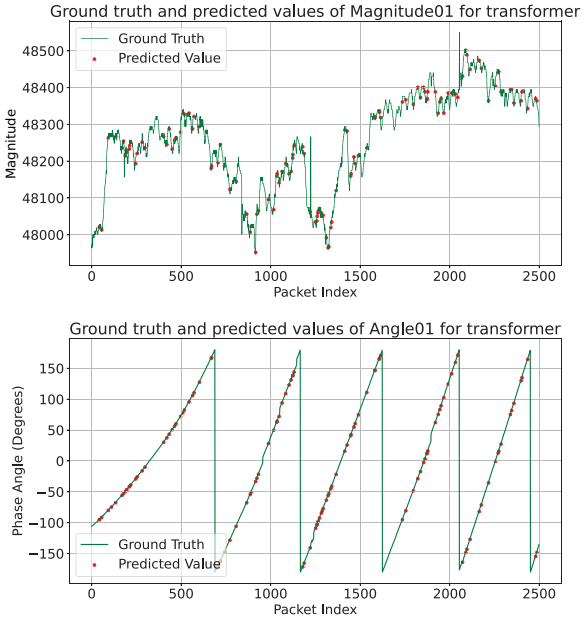

Fig. 6. Comparison of the predicted values and the ground truth values with 5% missing packets rate.

We also note that as the percentage of missing packets increases, both MAPE and MAE values rise. This increase is attributed to a larger portion of the input data being predicted rather than actual ground truth data, thereby introducing

Authorized licensed use limited to: University of Arkansas. Downloaded on April 29,2025 at 00:02:46 UTC from IEEE Xplore. Restrictions apply.

more error into the model. This trend is evident in Figure 5, where the cumulative MAPEs and MAEs increase with the number of continuous missing data packets. However, the variance of the predicted values remains low as the errors increase, meaning that the model's predictions are relatively consistent across different subsets of the testing dataset. By analyzing the occurrences of real missing packets from our dataset, we found that the majority involve fewer than 24 consecutive missing data packets. As shown in Figure 5, the MAPE for 24 missing values closely aligns with the average MAPE performance depicted in Figures 3 and 4.

In Figure 6, we compare the predicted values to the ground truth values in a scenario where 5% of the data is missing. Out of 2500 packets, we randomly designated 5% as missing, with the red dots indicating the values predicted by the transformer model. We plot the values of the first magnitude and the first phase angle from the PMU dataset. It is evident that the predicted values closely align with the ground truth values, maintaining the same distribution trend. This close correspondence demonstrates the effectiveness of our scheme in accurately addressing the PMU missing data recovery problem.

VI. Conclusion and Future Works

In this paper, we propose a deep learning based model on P4 programmable network switches for in-network PMU missing data recovery. We combine the advantages of the high-speed data plane for missing packet detection and the transformer model in the control plane for missing data prediction. We implemented the solution in the BMv2 P4 software switch, and our evaluation showed that the Vanilla Transformer model outperformed other machine learning models.

For future work, we aim to explore more specialized machine learning models tailored for time-series forecasting. Additionally, we aspire to implement our architecture on P4 hardware, enabling a comprehensive evaluation of the entire recovery process's speed. Furthermore, we intend to develop a decision tree-based algorithm to enhance the detection of missing PMU data packets.

Acknowledgment. The authors are grateful for the support of the National Science Foundation (NSF) under Grant CNS-2247721 and EEC-2113903, the U.S. Department of Energy, Office of Cybersecurity, Energy Security, and Emergency Response (CESER) under Award Number DE-CR00000031, and the Chancellor's Fund for Innovation and Collaboration at the University of Arkansas.

References

[1] Yang Chen, Le Xie, and PR Kumar. Dimensionality reduction and early event detection using online synchrophasor data. In *Proceedings of 2013 IEEE Power & Energy Society General Meeting*, 2013.

[2] Arun G. Phadke., James S Thorp, Reynaldo F. Nuqui, and Ming Zhou. v. In *Proceedings of 2009 IEEE/PES Power Systems Conference and Exposition*, 2009.

[3] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 2021.

[4] Denis Osipov and Joe H. Chow. PMU missing data recovery using tensor decomposition. *IEEE Transactions on Power Systems*, 2020.

[5] Shruthi Thangaraj, Vik Tor Goh, and Timothy Tzen Vun Yap. Modified recurrent equation-based cubic spline interpolation for missing data recovery in phasor measurement unit (PMU). *F1000Research*, 2022.

[6] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 2014.

[7] Zhiyao He, Yanfeng Qu, Gong Chen, Reuben Samson Raj, Hui Lin, and Dong Jin. Towards secure and resilient synchrophasor networks using p4 programmable switches. In *Proceedings of the 2024 IEEE Green Technology Conference*, 2024.

[8] Jack Norris, Zhiyao He, Yanfeng Qu, Gong Chen, Christine Hertzog, and Dong Jin. An in-network approach for PMU missing data recovery with data plane programmability. In *Proceedings of 2024 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*, 2024.

[9] Zhiwei Yang, Hao Liu, Tianshu Bi, Zikang Li, and Qixun Yang. An adaptive PMU missing data recovery method. *International Journal of Electrical Power & Energy Systems*, 2020.

[10] Yuanbin Cheng, Brandon Foggo, Koji Yamashita, and Nanpeng Yu. Missing value replacement for pmu data via deep learning model with magnitude trend decoupling. *IEEE Access*, 2023.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[12] p4Lang. p4lang/behavioral-model: The reference P4 software switch. https://github.com/p4lang/behavioral-model.

[13] Gong Chen, Zheng Hu, and Dong Jin. Enhancing fidelity of p4-based network emulation with a lightweight virtual time system. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2023.

[14] Mininet: An instant virtual network on your laptop (or other pc). http://www.mininet.org/.

[15] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[16] Andrew Harvey. Forecasting with unobserved components time series models. *Handbook of Economic Forecasting*, 2006.

[17] Stefan Mittnik. Macroeconomic forecasting experience with balanced state space models. *International Journal of Forecasting*, 1990.

[18] Granville Tunnicliffe Wilson. Time Series Analysis: Forecasting and Control, 5th Edition. *Journal of Time Series Analysis*, 2016.

[19] Mang Liao, Di Shi, Zhe Yu, Zhehan Yi, Zhiwei Wang, and Yingmeng Xiang. An alternating direction method of multipliers based approach for PMU data recovery. *IEEE Transactions on Smart Grid*, 2018.

[20] IEEE standard for synchrophasor data transfer for power systems. *IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)*, 2011.

[21] C Spearman. The proof and measurement of association between two things. *International Journal of Epidemiology*, 2010.

[22] Karl Pearson. VII. Note on regression and inheritance in the case of two parents. *The Royal Society of London*, 1895.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016.

[25] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 2021.

[26] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1, 2024.

[27] Microgrid at IIT. Online. Available: http://iitmicrogrid.net/.

Authorized licensed use limited to: University of Arkansas. Downloaded on April 29,2025 at 00:02:46 UTC from IEEE Xplore. Restrictions apply.