Programmable Data Plane Approaches for ML-Based Anomalous Event Classification in Grid Sensor Networks

Zhiyao He[‡], Reuben Samson Raj[‡], Yanfeng Qu, Dong Jin
Department of Electrical Engineering and Computer Science
University of Arkansas, Fayetteville, AR, USA
{the, rs077, yqu, dongjin}@uark.edu

***Abstract*—Grid-sensor networks, such as Phasor Measurement Units (PMU) systems, generate high-resolution, time-synchronized measurement data that are critical for detecting voltage and frequency disturbances. Conventional machine learning (ML) classification methods for detecting anomalous voltage events often rely on centralized processing, introducing latency that limits real-time responsiveness. In this work, we explore an in-network classification framework that leverages P4 programmability to support early classification of voltage events in synchrophasor data. We propose two complementary approaches: a data plane (DP)-driven approach that compiles ML logic directly into switch match-action tables for low-latency real-time classification, and a control plane (CP)-driven approach where parsed features by the network switch are exported for more accurate, although slightly higher-latency model inference. Using 27,000 PMU measurements from a campus microgrid, injected with 5% anomalous events, we evaluated multiple ML models, with the DP- and CP-driven approaches achieving on average 96% and 98% accuracy, respectively. Additionally, a hardware-based evaluation of the DP-driven approach using Intel's Tofino programmable switch demonstrated processing latency between 577 and 610 nanoseconds. Our work illustrates the feasibility of leveraging P4 for real-time, in-network deployment of ML-based security analytics in smart-grid communications.**

***Index Terms*—Grid-sensor Networks, Phasor Measurement Unit, Power System Monitoring, Event Detection and Classification, Programmable Network, P4, Smart Grid**

I. Introduction

Today's power grids employ sensor networks, such as Phasor Measurement Unit (PMU) networks, as a key component for situational awareness. PMUs provide time-synchronized measurements of voltage, frequency, and phase angle at high resolution (called Synchrophasors [1]) that enable grid operators to observe dynamics and respond to any disturbances in real time. Communication-wise, these Synchrophasor streams serve as application-layer telemetry flows.

Traditional grid-monitoring aggregates PMU data at Phasor Data Concentrators (PDC) and forwards synchrophasor streams to regional and central control centers, where event detection and analytics are performed [2]. This introduces additional network and processing hops before alerts or control actions. To overcome this challenge, modern programmable networks such as P4 [3] offer the possibility to offload event detection close to the communication network itself. The integration of P4 with PMU networks offers three key advantages: (i) Deep packet inspection that enables direct extraction and processing of PMU measurements, (ii) Line-rate processing of packets that can facilitate real-time analysis of power system events, and (iii) Flexible interfacing of P4 switches with general-purpose compute controller devices.

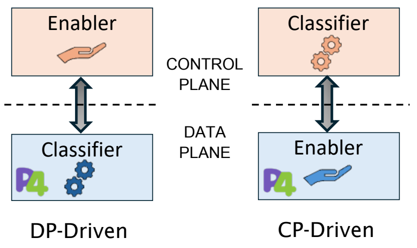

Fig. 1: P4-based power system event classification approaches

In this paper, we explore communication-first architectures that embed ML-based event classification in the network data path. We propose two complementary paradigms for in-network classification as shown in Fig. 1. In the **data-plane (DP)** driven approach, ML models are compiled into P4 artifacts - such as parsing and match-action tables (using tools such as Planter [4]) for the switch to perform line-rate inference. Here, the data-plane acts as the inference classifier, while the control-plane plays the role of an enabler, providing the data-plane with the required artifacts to perform the classification.

In the **control-plane (CP)** driven approach, the P4 switch extracts key features from PMU packets and exports them to the control-plane, where a trained ML model performs classification. The P4 switch can extract features at line rate, while the control plane offers higher model accuracy at the cost of a modest increase in processing delay. Thus, in this approach, the roles are flipped, with the switch data-plane acting as an enabler by feeding in features, while the control-plane acts as the main classifier.

Importantly, in this work, we do not seek to propose new ML models for PMU classification. Instead, our focus is on exploring the practical deployment of existing classifier models in programmable switches and controllers, supported by tools like Planter. Using PMU data from a campus microgrid containing 5% anomalous events, we evaluated both approaches on a range of existing ML models, including Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), and Principal Component Analysis (PCA). While the CP-

[‡]Equal contribution.

driven approach was evaluated using a software P4 switch, the DP-driven evaluation was evaluated for accuracy and switch processing latency in both software and hardware P4 switches.

To the best of our knowledge, this is the first work to demonstrate the use of programmable network switches in integrating ML models for power systems event classification. The main contributions of this work are summarized as follows:

- Developed a PMU-specific feature extraction and event classification workflow using P4's programmable parser and match-action tables. The feature extraction is done on the data plane, while the classification inference is offloaded to the control plane (CP-driven).
- Built a PMU-specific data loader program and integrated it with the Planter module. The data loader and Planter run on the control plane, while the event classification is performed in-network, on the data plane (DP-driven).
- Evaluated both the above approaches on real PMU data obtained from a campus microgrid environment, comparing classification accuracy for DT, RF, KNN, and PCA models. With a dataset of 27,000 packets containing 5% anomalous events, both DP-driven and CP-driven implementations achieved similar accuracy (96% and 98% respectively), with 1-2% difference across models.
- Assessed the run-time processing latency for the DP-driven approach on a hardware P4 (Tofino) switch target, which achieved a classification time between 577-610 nanoseconds per packet, while the CP-driven approach on the BMv2 software switch achieved an average processing latency of 3.65 milliseconds per packet, across different models.

II. Background and Related Work

A. Programmable Networks and P4

Over the past decades, communication networks have evolved from fixed-function services to more programmable entities, with Software-Defined Networking (SDN) marking the first such milestone. SDN introduced a major shift in viewing the network as two layers of abstraction: (i) control-plane (CP) that manages forwarding behavior and (ii) data plane (DP) that deals with the actual forwarding of packets. While SDN pioneered a programmable control plane, P4 [3] is the next evolution that has brought programmability to the switch data plane.

P4 allows creation of a custom packet processing pipeline through programmable parsers, match-action tables (for fast look-ups on parsed fields), along with several other 'extern' functions such as registers, meters, and counters. P4 programs can run on a variety of targets, including software switches like bmv2 [5] and hardware switches such as Intel's Tofino ASIC. P4's programmability has resulted in a number of applications that can be offloaded in-network, such as telemetry, traffic classification, routing, and security [6].

B. P4 Limitations for ML and Planter

In addition to the use cases listed above, recent works have shown a significant interest in the integration of machine learning with programmable data planes [7]. However, in-network ML is particularly challenging for several reasons, namely – limited on-switch memory (on-chip SRAM is allocated to fast look-ups and only small register arrays are available for stateful work), no native floating-point arithmetic, conventional loops are absent, and some ASICs only expose reduced-precision integer math operations.

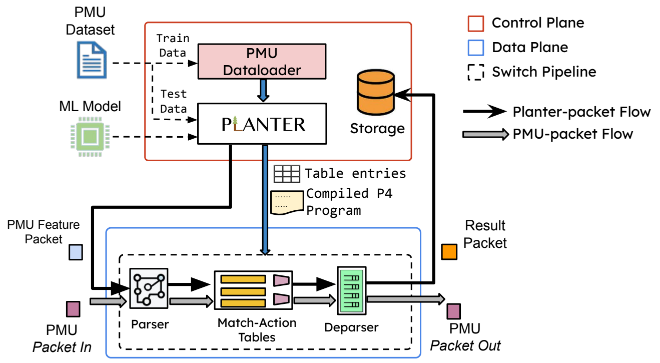

Fig. 2: Data-plane (DP) driven event classification

Planter [4] is a framework that addresses these constraints by introducing a novel end-to-end workflow that converts ML models into deployable P4 artifacts. It trains a user-specified ML model on the control-plane and generates equivalent P4 artifacts, installable onto the switch. This effectively transforms the switch into a classification device, capable of line-rate inference. While Planter is domain-agnostic, we build on this idea and extend its functionality to the power-system domain, specifically in our DP-driven approach.

C. Related Work

Traditional classification approaches rely on fixed thresholds applied to voltage, frequency, or ROCOF signals. These rule-based methods are simple and efficient but lack robustness to noise and cannot capture complex event dynamics. More advanced tools like FMEA and DFMEA are structured but manual and not suitable for automation. Machine learning methods offer improvements: Niazazari et al. [8] apply CNNs trained on real PMU data for accurate classification without manual feature engineering, while Liu et al. [9] use low-rank localization and signal-shape features with Random Forests for interpretable, resilient classification.

The above conventional methods, while robust in terms of the models themselves, are typically implemented as offline classification workflows on centralized infrastructure, which can cause delays and limit real-time applicability. Building on our previous work on real-time in-network power event detection [10], our proposed approaches are decentralized and run in-network, which enables real-time PMU event classification. In the following section, we present the system design for our two proposed approaches.

III. System Design

A. Data-Plane (DP) Driven Event Classification

Fig. 2 illustrates the workflow of our proposed DP-driven event classification system. Below, we describe the overall process, along with the role of each component in the system:

- **Data Loader**: The Data Loader program we developed in Python processes the dataset CSV, performing PMU-aware

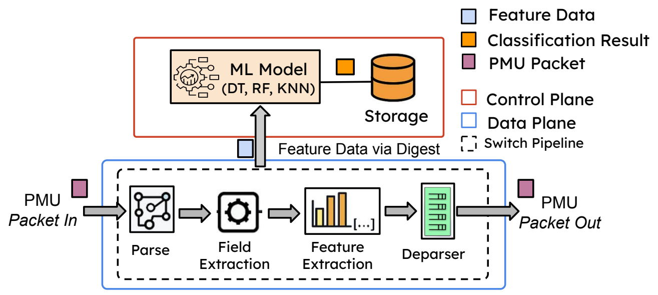

Fig. 3: Control-plane (CP) driven event classification

pre-processing steps such as voltage and frequency scaling, filling missing data, and creating engineered features. This serves as input to the Planter module.

- **Planter**: The planter module takes as input, the processed dataset and a user-specified ML model, runs the training locally, and automatically generates the required switch artifacts - P4 program and match-action table entries. Note that these artifacts are ML model-specific, thus essentially converting an ML model to its equivalent P4 program.
- **Switch setup**: To set up the switch data plane, Planter also compiles the P4 program and installs the compiled code and table entries onto the switch target, which can be a software switch (BMv2) or hardware (Tofino).
- **Test and Store Result**: To test the model, Planter converts each entry from the dataset into custom feature packets that align in structure with the generated P4 program. For each PMU test packet sent to the switch, the P4 pipeline performs parsing, table lookups, and appends the classification label onto the Result packet. The classification result is forwarded back to a storage unit running in the control plane.

In this approach, the CP takes the role of an *enabler* (data loading, feature extraction, p4 code generation) while the main classification inference is performed on the switch data plane. These roles are reversed in the CP-driven approach, as discussed in the next section.

B. Control-Plane (CP) Driven Event Classification

Fig. 3 illustrates the proposed CP-driven event classification workflow, with components described below:

- **Data Training**: The training data [11] is used to learn a supervised model from labeled PMU measurements. Feature vectors are constructed from voltage magnitude, phase angle, and frequency values over fixed time windows. During training, the model learns decision boundaries that minimize classification error on the training set. After training converges, the learned parameters are fixed and applied during runtime inference. As new labeled data becomes available, the model can be retrained to update the learned decision boundaries and adapt to changes in system behavior.
- **Parsing**: The switch identifies the PMU packet and determines the header and payload layout.
- **Field Extraction**: The switch extracts measurement values from the payload, including voltage magnitude, phase angle, and frequency. These values are stored in metadata fields for temporary use within the pipeline.
- **Feature Extraction**: Relevant features are selected from the extracted fields. Examples include voltage magnitude, frequency, and rate of change. These features are compactly encoded into a structured format and transmitted to the control plane using digest messages over a Nanolog socket.
- **Classification**: The control plane applies a pre-trained machine learning model (e.g., DT, RF, KNN, PCA) to classify events, whose results are stored in a database for further analysis.

This is in contrast to the DP-driven approach, where the data plane now acts as the enabler (parsing and feeding features to the CP-driven model for inference), while concurrently forwarding the original PMU packets to the intended destination.

In both approaches, the stored labels can then be used by grid operators (or downstream analytics) for real-time alerting and post-event analysis, for instance, correlating event types with specific buses to identify problematic locations and recurring disturbances.

IV. Event Classification

Power system event classification is the process of identifying abnormal electrical conditions from real-time PMU measurements. Disturbances such as sudden voltage or frequency shifts get manifested as deviations in PMU data. Table I summarizes the voltage-event rules used in this study, classifying disturbances by their per-unit magnitude V and duration D into six condition types: Sag, Swell, Interruption, Sustained Interruption, Under-Voltage, and Over-voltage based on the work in [12]. While the voltage can be directly extracted through parsing, the duration D is determined by calculating the time difference between the start and end of the voltage deviation, using the timestamp in consecutive PMU packets. The classification system uses a feature vector that is derived from each PMU packet, such as voltage, frequency, SOC timestamp, and phase angle [1], and outputs the class label as defined in Table I.

TABLE I: Voltage event rules

Voltage V (pu)	Duration D	Condition Type
$0.1 < V < 0.9$	0.5 cycles $\leq D \leq$ 1 min	Voltage Sag
$1.1 < V < 1.8$	0.5 cycles $\leq D \leq$ 1 min	Voltage Swell
$V < 0.1$	$D \leq 2$ min	Interruption
$V < 0.1$	$D > 2$ min	Sustained Interruption
$V \leq 0.9$	$D > 1$ min	Under-Voltage
$V \geq 1.1$	$D > 1$ min	Over-Voltage

A. Illustrative Example: Decision Tree Based Classifier

While several models were evaluated in our study, we use the Decision Tree (DT) model here to illustrate the classification process. To formalize the behavior of the decision tree across platforms, we consider a PMU packet p_i with a feature vector:

$$\mathbf{X}_i = \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(d)}\right),$$

where each component $x_i^{(j)}$ represents a measurement such as voltage, frequency deviation, duration, or angle. The DT applies a sequence of threshold-based comparisons of the form:

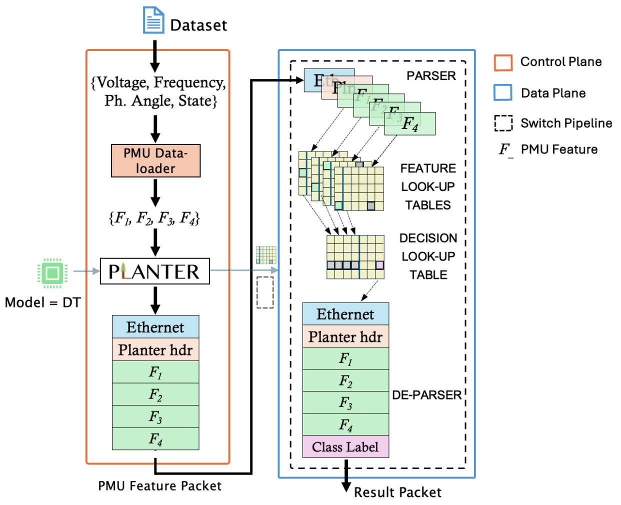

Fig. 4: DP-driven Decision Tree classification

$$x_i^{(j)} \leq \theta_j$$

Each internal node checks one such condition to determine whether the traversal proceeds to the left or right sub-tree. This continues until a leaf node is reached, at which point the classifier assigns an event class label. The classification output is denoted as:

$$C_i = \text{Tree}_\theta(\mathbf{X}_i), \quad \text{where } C_i \in \{\text{Normal}, \text{Voltage Events}\}$$

The DT function $\text{Tree}_\theta(\cdot)$ can be defined recursively as:

$$\text{Tree}_\theta(\mathbf{X}_i) = \begin{cases} C_\ell, & \text{if a leaf node is reached} \\ \text{Tree}_{\theta_{\text{left}}}(\mathbf{X}_i), & \text{if } x_i^{(j)} \leq \theta_j \\ \text{Tree}_{\theta_{\text{right}}}(\mathbf{X}_i), & \text{otherwise} \end{cases}$$

where C_ℓ denotes the class label stored at a given leaf node ℓ. This formulation ensures that the same decision logic is preserved across software, hardware, and CP-driven environments, allowing classification decisions to be made consistently regardless of the execution environment. In the following subsection, we present how the DT classifier is implemented in each of the proposed two approaches.

B. DT workflow in DP and CP-driven approaches

Fig. 4 depicts a sample classification workflow sequence in the DP-driven for the DT model. Note that Fig. 4 and Fig. 2 are similar, except that Fig. 4 shows the P4 switch pipeline components, along with input and output packet formats. In the control plane, our PMU dataloader pre-processes the PMU dataset and feeds the generated features to Planter. Using these features and the specified input model as DT, Planter trains the model and generates an equivalent P4 pipeline, and match-action table entries to be installed onto the switch data plane. The combination of this specific pipeline and table entries effectively mimics a DT flow. Each of the extracted features is first used to perform a feature look-up table to fetch a metadata code. This code is then used to perform another look-up on a final decision table to fetch the event class label, which the de-parser then packs onto the result packet before emitting it out of the switch.

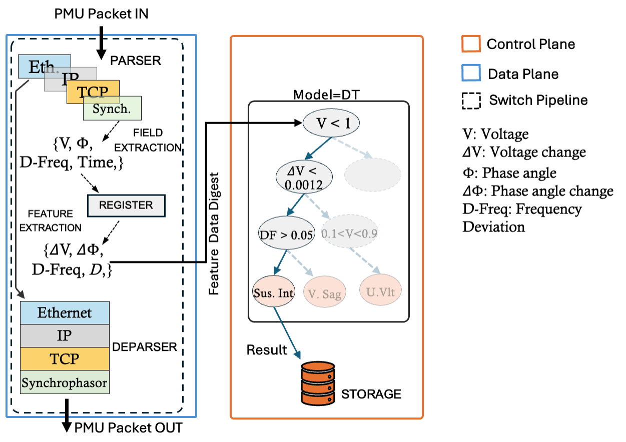

Fig. 5: CP-driven Decision Tree classification

Similarly, Fig. 5 presents the low-level classification workflow of Fig. 3 for a sample PMU packet in the CP-driven approach. In the data plane, incoming PMU packets are parsed to extract voltage, phase angle, frequency deviation, and timestamp. These fields are buffered using registers to compute features such as voltage change, angle change, and frequency deviation by comparing current and previous packets. The extracted features are sent as digest messages to the control plane via the Nanolog socket. The control plane then walks through the pre-trained DT model by evaluating each node with the received feature values. For example, starting from the root, it checks if the normalized voltage value is less than 1; if so, it then evaluates whether the voltage change is under 0.0012, and finally whether the frequency deviation exceeds 0.05, at which point the path terminates in a leaf that labels the event as a Sustained Interruption.

V. Evaluation

A. Experimental Setup and Dataset

We constructed a network topology in the Mininet emulator [13] that includes PMU and PDC communication through a P4 switch. The dataset, collected from an IIT campus microgrid, contains 27000 measurements from 12 PMUs, including magnitude, phase angle, SOC timestamp, frequency, and other related parameters. To emulate grid disturbances, we synthetically inject 5% anomalous events into the dataset, using voltage dips and phase shifts. Each set of experiments was executed five times. Code and dataset available at [11].

We performed three evaluations:

- CP-driven on software switch (BMv2)
- DP-driven on software switch (BMv2)
- DP-driven on hardware switch (Intel Tofino)

In all three evaluations, the same set of models - DT, RF, KNN, and PCA were evaluated to ensure direct comparison of performance metrics. The evaluated categories include the events listed in Table I.

B. CP-Driven - BMv2

Next, we present the accuracy and heatmap comparisons for CP-driven results. Fig. 6 presents a model-wise comparison

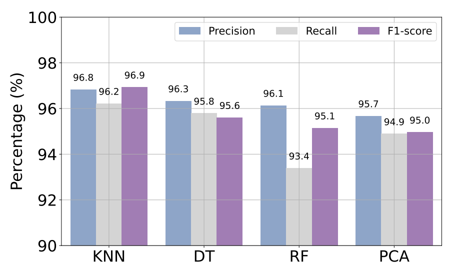

Fig. 6: Perf. metrics for CP-driven

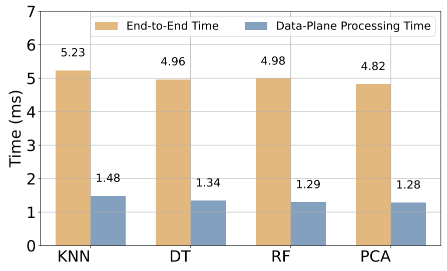

Fig. 7: CP-driven Execution Time

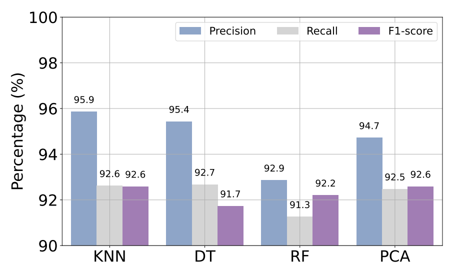

Fig. 8: Perf. metrics for DP-driven

of precision, recall, and F1-score for the CP-driven approach. All models achieved high and balanced performance, with precision, recall, and F1-score values generally above 95%. Notably, KNN attained the highest precision at 96.8%, while PCA demonstrated slightly lower but still competitive results.

Fig. 9 (left) shows a heatmap of precision scores across event types and models. Most models maintain precision above 94%. KNN achieves 98.5% on normal events and stays above 97% for others. DT and RF also perform strongly, even in harder cases such as interruptions and voltage deviations. The accuracy and heat map comparisons indicate that our P4-based CP-driven approach performs with robustness and consistency, not only at the model level but also when broken down by event type.

In Fig. 7, the average end-to-end processing time per packet is shown across different models. The blue bars indicate the data-plane processing time, which is consistently low, ranging from 1.28 ms to 1.48 ms and standard deviations between 0.09 and 0.12 ms. The small variation across models highlights the stability of the approach, and this stability underscores the robustness of the approach in supporting real-time performance.

C. *DP-Driven - BMv2*

As shown in Fig. 8, KNN achieved the highest average precision at 96.2%, demonstrating strong accuracy in identifying power events. PCA and DT followed closely, while RF was less accurate overall. Both KNN and PCA reached 92.5% recall and 92.6% F1-score, confirming their robustness. DT and RF showed a small drop in recall and F1-score, indicating less stable performance across event types. DT performed better than RF due to its compact, sequential match-action rule translation in the data plane, whereas RF, as an ensemble of trees, requires parallel decision paths that lead to fragmented and less efficient rule representations, reducing classification accuracy during in-network execution.

As shown in Fig. 9 (right), KNN demonstrated consistently high precision across all classes, achieving 96% on Normal Events, 94.5% on Voltage Sag, and 95.2% on Voltage Swell. For more complex event types such as Interruption, Undervoltage, and OverVoltage, KNN maintained a stable precision of 93%, highlighting its strong generalization and robustness to different events.

In comparison, the DT and RF models exhibited slightly reduced performance on complex event types. The DT model attained 97.4% precision on Normal Events but showed lower precision for Voltage Swell (93%), Interruption (91%), and Undervoltage (91%). RF followed a similar trend with 92% to 93% precision across most categories and dipped to 90%

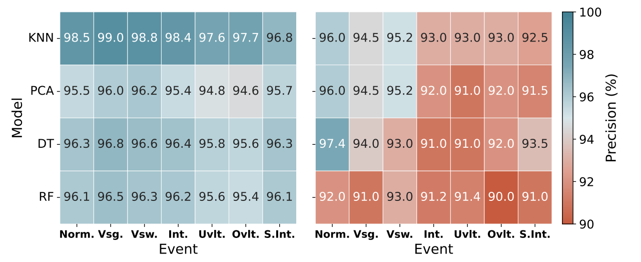

Fig. 9: Classification precision for each model across voltage event classes in the CP-Driven(left) and Dp-Driven(right). Abbreviations: Norm = Normal, Vsg. = Voltage Sag, Vsw. = Voltage Swell, Int = Interruption, Uvlt. = Undervoltage, Ovlt. = OverVoltage, S.Int. = Sustained Interruption.

for OverVoltage events. These results suggest that tree-based models, while accurate on average, may struggle with subtle variations in event characteristics and could be prone to overfitting or reduced sensitivity in noisier classifications.

Overall, KNN and PCA not only achieve high average precision but also maintain balanced performance across all event types, making them strong candidates for deployment in real-time PMU event classification systems.

D. *DP-Driven - Hardware P4*

Setup. Intel Tofino programmable ASICs typically deliver terabit-per-second (Tbps) switching performance. In our experiments, we used the Netberg Aurora 610 switch, which integrates a Tofino ASIC and provides an aggregate throughput of ∼2Tbps across its ports. Fig. 10 shows the setup with the Tofino switch connected to an Ubuntu PC with 10G NIC interfaces. PMU test traffic containing Planter-generated feature data was replayed from the PC to the switch's 25G ports.

Since Planter does not provide a testable P4 compilation workflow for Tofino by default due to Intel's confidentiality agreement, we manually verified the accuracy by replaying the same test traffic used in the BMv2 evaluation. Planter produces near-identical P4 code and match-action tables for both BMv2 and Tofino targets. Thus, on comparison of output packets from both systems using identical input PMU feature data packets, no differences in event classification accuracy were observed from the BMv2 implementations. We therefore exclude the hardware accuracy results here to avoid redundancy.

Measurement and Analysis. To measure the processing latency, we modified Planter's P4 code generation module to embed key timestamps onto the classification result packet. Tofino can read and embed nanosecond-precision timestamps

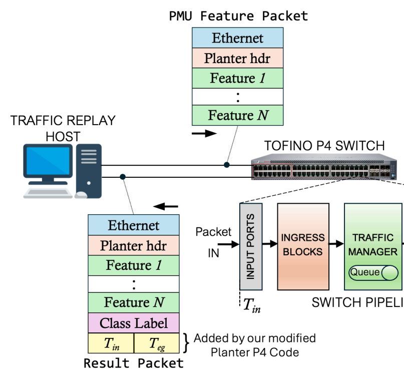

Fig. 10: Hardware P4 evaluation showing setup, packet structure, and timestamp insertion points in the switch pipeline

at multiple stages as packets traverse the switch pipeline. Our modified P4 code appends a custom header (yellow header in Fig. 10) containing two timestamps into every classification result packet: (i) time of packet entering the switch ingress MAC (T_{in}), and (ii) time of packet exiting the egress de-parser (T_{eg}). The host extracts the timestamp header from the result packet and computes processing latency as $T_{eg} - T_{in}$.

We measured the latency for input traffic replayed at varying bit rates, from 1 Mbps to 150 Mbps, for each of the four models (KNN, PCA, DT, and RF). Note that these bit rates are much higher than typical PMU traffic, which is usually a few hundred Kbps [14]. The hardware switch pipeline consumes between 577 and 610 nanoseconds per packet, with negligible deviation across input traffic rates (Table II).

Model complexity seemed to be the primary influence on latency, while an increase in input traffic rate had little effect. RF and DT models required a higher number of table look-ups per packet, resulting in slightly higher latency. Also, increasing the tree depth did not affect latency because the number of table look-ups remained the same; it only resulted in a larger number of entries per table.

E. Discussion

Stage and Memory constraints. Programmable switch pipelines have limited stages and on-chip memory that can place a lot of constraints on in-network ML mappings. Planter [4] (used in our DP-driven approach) explicitly addresses these constraints by using mapping strategies that reduce the number of stages used and instead trade computation for table look-ups. For instance, direct mapping of tree models can require a number of sequential tables proportional to tree depth, whereas Planter's encode-based RF mapping parallelizes tree evaluation. Also, it uses a small constant number of logical stages (feature tables sharing a stage, tree/code tables sharing a stage, followed by a voting table). Thus, increasing depth primarily affects the number of table entries (memory) rather than pipeline depth.

Scalability. Our framework scales with input traffic volume and model complexity rather than the number of PMU nodes, as latency is directly influenced by received PMU packet rate and feature size. In particular, the BMv2 results should be interpreted as a functional prototype and a conservative software baseline. Since BMV2 is host-CPU bound and does not reflect hardware switching pipelines, it provides an upper-bound worst-case estimate of end-to-end latency in a limited compute environment. Nevertheless, the measured processing times remain in the few-millisecond range, which is still within typical smart-grid monitoring latency budgets.

In contrast, Tofino hardware P4 sustains line-rate inference with near-constant per-packet latency across input traffic rates. The observed hardware latency is orders of magnitude below the typical smart-grid budget (hundreds of microseconds to tens of milliseconds). This indicates sufficient scalable capacity for hundreds of PMU nodes while maintaining predictable latency and throughput.

Accuracy Comparison. The two approaches resulted in comparable accuracy (1-2% difference). One contributing factor to this increased accuracy of the CP-driven approach is the use of full floating-point precision during the model training and inference, whereas the DP-driven approach loses some precision due to some approximations that P4 introduces. The increased resolution enables more fine-grained distinctions between event types in the CP-driven approach.

Trade-offs and Future Extensions. The two approaches we presented have their own benefits and limitations. The CP-driven approach offers the flexibility to run complex models due to the presence of general-purpose CPUs/GPUs, which are challenging in a DP-driven approach due to constraints on switch hardware. However, while the feature-extraction and feeding to CP can be done in real-time and at line-rate, the inference speed is still limited by the CP hardware.

The DP-driven approach is able to produce a faster, line-rate classification (in the HW setup). However, the pipeline currently performs inference on custom PMU feature packets, instead of native PMU packets. Therefore, a logical future extension to our current DP-driven approach is to modify Planter's workflow for supporting native PMU packets. This would require an encoder-decoder system to convert packets from Synchrophasor to Planter format and back.

TABLE II: Average Processing Latency and St. Dev (*in nanoseconds*) of Tofino for Different ML Models and Bit Rates

Bit Rate / ML Model	1 Mbps	10 Mbps	50 Mbps	150 Mbps
KNN	577.37	577.37	577.37	577.38
	1.78	1.78	1.78	1.78
PCA	580.81	580.81	580.82	580.81
	1.71	1.71	1.71	1.71
DT (Tree Depth = 4)	597.21	597.20	597.20	597.20
	1.68	1.67	1.67	1.68
DT (Tree Depth = 15)	597.19	597.20	597.20	597.20
	1.74	1.72	1.72	1.72
RF (# Trees = 5, Depth = 4)	610.28	610.28	610.28	610.27
	1.72	1.73	1.72	1.72

VI. Conclusion

Programmable data planes offer novel possibilities for offloading applications to the communication network. In this paper, we explored two complementary approaches using P4 for ML-based anomalous event classification of grid-sensor data. Our evaluations in both software and hardware switch platforms demonstrate that P4 can flexibly deploy ML models into a packet-processing workflow applicable to power systems. Even with these customizations, both the CP-driven and DP-driven approaches achieve reasonably high accuracy. The presented tradeoffs can better inform the choice of P4 deployment for a utility network administrator, and also motivate future research in expanding this work towards limiting anomalous PMU traffic and extending Planter's workflow to support truly real-time PMU data event classification.

Acknowledgment. The authors are grateful for the support of the National Science Foundation (NSF) under Grant CNS-2247721 and the U.S. Department of Energy, Office of Cybersecurity, Energy Security, and Emergency Response (CESER) under Award Number DE-CR00000031.

References

[1] "IEEE standard for synchrophasor data transfer for power systems," *IEEE Std C37.118.2-2011*, pp. 1–53, 2011.

[2] "IEEE guide for phasor data concentrator requirements for power system protection, control, andmonitoring," *IEEE Std C37.244-2013*, 2013.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[4] C. Zheng, M. Zang, X. Hong, L. Perreault, R. Bensoussane, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "Planter: Rapid prototyping of in-network machine learning inference," *SIGCOMM Comput. Commun. Rev.*, Aug. 2024.

[5] p4Lang, "p4lang/behavioral-model: The reference P4 software switch." github.com/p4lang/behavioral-model.

[6] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A survey on data plane programming with p4: Fundamentals, advances, and applied research," *Journal of Network and Computer Applications*, vol. 212, p. 103561, 2023.

[7] C. Zheng, X. Hong, D. Ding, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "In-network machine learning using programmable network devices: A survey," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 1171–1200, 2023.

[8] I. Niazazari, Y. Liu, A. Ghasenikhani, S. Biswas, H. Livani, L. Yang, and V. A. Centeno, "Pmu-data-driven event classification in power transmission grids," in *Proceedings of IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2021, pp. 1–5.

[9] Y. Liu, L. Yang, A. Ghasemkhani, H. Livani, V. A. Centeno, and P.-Y. Chen, "Robust event classification using imperfect real-world pmu data," *IEEE Internet of Things Journal*, vol. 10, pp. 7429–7438, 2023.

[10] Z. He, Y. Qu, and D. Jin, "Real-time power system event detection on programmable network switches with synchrophasor data," in *IEEE International Conference on Communications (ICC)*, 2025.

[11] Z. He, "Event classification," https://github.com/timhealltheway/Event-Classification---ICC, 2026, gitHub repository.

[12] M. H. J. Bollen, *Understanding Power Quality Problems: Voltage Sags and Interruptions*. New York, NY, USA: Wiley-IEEE Press, 2000.

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

[14] G. Hataway, B. Flerchinger, and R. Moxley, "Synchrophasors for distribution applications," in *Proceedings of the 2013 Power and Energy Automation Conference, Spokane, Washington, March*, 2013, pp. 26–28.