Interpretable Models for Workflow Differentiation in High-Performance Scientific Networks

Anna Giannakou*, Syed Asif Raza Shah§, Wenji Wu*, Bruce Mah*, Philip Demar§, Oliver Gutsche§, Chin Guok*

*Lawrence Berkeley National Lab, Email: {agiannakou, wenji, bmah, chin}@lbl.gov
§Fermi National Lab, Email: {sshah, demar, gutsche}@fnal.gov

***Abstract*—Scientific workflows in high-performance networks spawn hundreds of interdependent flows that must be managed collectively—yet existing network classifiers treat each flow in isolation, leading to fragmented QoS decisions and missed inter-flow patterns. We present a novel traffic classification solution that operates at the *workflow level*, distinguishing entire file-transfer operations from streaming analytics by capturing how concurrent flows interact and burst together. We introduce a workflow identification window (WIW) that ingests raw packet headers from parallel flows into unified tensors, preserving the spatial-temporal patterns that differentiate scientific workflows. This approach achieves 98.7% accuracy using CNN, LSTM, and hybrid architectures, while maintaining 84% accuracy on production traffic collected a week later—demonstrating robustness to temporal drift. By integrating SHAP and Grad-CAM explainability, we reveal that early-packet timing patterns and cross-flow correlations drive classification decisions, providing operators with interpretable insights. Our system enables coherent workflow-level QoS enforcement and dynamic bandwidth allocation in scientific networks, eliminating manual per-flow configuration while maintaining classification latency at millisecond level.**

I. Introduction

High-performance scientific networks like the Energy Sciences Network (ESnet) [1] are increasingly called upon to support data-intensive experiments such as the Compact Muon Solenoid (CMS) [2] experiment at the Large Hadron Collider (LHC) [3]. Meeting performance demands requires both high-resolution visibility into network traffic and accurate interpretation of network conditions. Visibility and interpretability are essential not only for diagnosing performance issues and re-engineering affected traffic in real time, but also for proactively applying Quality of Service (QoS) policies to ensure workflow-specific performance requirements are consistently satisfied. The CMS experiment features distributed workflows* that generate hundreds, even thousands, of flows† containing latency-sensitive streaming and throughput-intensive bulk transfers. As scientific research shifts toward self-driving labs and greater experiment automation, data processing requirements are growing exponentially, intensifying the need for intelligent, real-time traffic visibility via zero-touch operations, alongside advanced decision-making capabilities. CMS network operators require precise tools to distinguish between streaming and file transfer traffic to optimize bandwidth, allocate resources efficiently, and maintain workflow performance. However, while scientific networks have a plethora of tools and solutions capable of providing high-resolution, real-time visibility into network traffic (e.g., sFlow [4], High-Touch [5], NetFlow [6]), accurate and timely interpretation of the collected data remains challenging. First, existing solutions primarily rely on offline, post-mortem analysis, which delays identification of performance issues and impedes proactive network management.

Second, many research works propose novel classification models—from custom CNN variants to self-supervised transformers [7] [8] [9] [10] [11] [12]—but rarely make it into production: volatile traffic patterns and protocol heterogeneity cause brittle performance [13] [14] and undermine real-time reliability [15]. In addition, existing ML solutions for scientific networks [16] [17] [18] treat each flow in isolation, and thus cannot capture the inter-flow correlations inherent in complex scientific workflows. Differentiating workflows rather than individual flows is challenging because (i) flows from the same workflow start and stop asynchronously, (ii) multiple workflows often overlap on the same bottleneck links, and (iii) flows from the same workflow are not grouped by a simple 5-tuple pattern: they often originate from different hosts and ports. The only observable evidence that they belong together is that they experience the same congestion and QoS effects at the same time—for example, aligned bursts, simultaneous TCP window backoff, or correlated queueing delay across multiple flows. Per-flow methods miss these cross-flow effects and thus fragment QoS decisions across flows that actually belong to the same workflow. Compounding these challenges, the opaque "black-box" nature of most models further hinders operator trust in critical environments [19]. Machine learning for scientific network management must go beyond accuracy on static test sets—it needs to be fully interpretable and maintain robust performance under live, heterogeneous traffic conditions so it can reliably distinguish entire workflows (e.g., bulk transfers vs. streaming) rather than isolated flows enabling per-workflow QoS enforcement and dynamic resource allocation. We address this gap through adapting CNN, LSTM, and hybrid

*A "workflow" is a coordinated, end-to-end sequence of data and compute tasks where datasets (files or streams) move across heterogeneous storage and compute tiers according to the experiment's logic

†A "flow" is defined by a set of characteristics that allow packets to be grouped together. The most commonly used set of characteristics for defining a flow is the 5-tuple that consists of SrcIP, DstIP, SrcPort, DstPort, and Protocol

architectures to scientific traffic by feeding them raw-header tensors derived from overlapping multi-flow windows tailored to diverse CMS workflows—rather than isolating individual flows—and integrate SHAP and Grad-CAM explainability techniques to achieve rapid, transparent, per-workflow traffic differentiation under real-world operational conditions.

In this paper, we introduce a multi-flow sliding-window solution that delivers low-latency, interpretable traffic differentiation in high-performance science networks under real production load conditions. Rather than designing new AI models, we adapt CNN, LSTM, and hybrid architectures to learn directly from raw packet-header tensors extracted through our *workflow identification window (WIW)*—thereby eliminating the biases and blind spots of manual feature engineering [20]. Autonomous feature selection enables our models to discover complex interactions and nuances that human-designed features might miss, ultimately leading to a classification solution that performs reliably across varying network conditions. We make the following contributions:
(1) We develop an end-to-end, production-validated solution for low-latency deep learning-based traffic classification with robust flow interdependency mapping. Our goal is to accurately identify and differentiate scientific workflows, instead of individual flows. Our approach includes *workflow identification window*, a novel sliding window mechanism that synchronizes overlapping time intervals across concurrent flows. Unlike prior time-window techniques that only bin packets within a single flow [21] [22], WIW always groups a target flow together with all other flows active in the same interval, and encodes them jointly thereby preserving both individual packet characteristics and dynamic inter-flow dependencies. Our adapted models reach 99% accuracy on held-out training windows and still achieve 84% accuracy on unseen, CMS production traffic. Our solution is seamlessly integrated with existing scientific networking monitoring services (e.g., High-Touch). (2) Enhanced model interpretability through SHAP [23], Grad-CAM [24] and temporal packet analyses. Our solution precisely identifies the spatial and positional patterns among packets that influence classification decisions.

The rest of the paper is organized as follows: Section II provides the necessary backgrounds. Section III details our end-to-end pipeline, encompassing data processing, model selection, and overall data flow. Section IV presents the experimental results. Finally, we conclude with key observations and suggestions for future work in Section V.

II. Background and Related Work

CMS experimental workflows. CMS workflows are designed to support large-scale data processing, analysis, and simulation tasks in High Energy Physics research. They generate many interdependent TCP flows across geographically distributed sites and can be broadly categorized into two major types:

- ***File transfer workflows*** managed via FTS/Rucio [25] atop multiple servers, replicating large datasets at sustained rates —spawning hundreds of parallel flows to move the data. For example, a typical CMS file-transfer workflow replicates multi-terabyte datasets from Tier-0 (CERN) to Tier-1 (Fermilab) using hundreds of parallel 10 Gb/s flows, sustaining near-line-rate transfers over minutes to hours [26].
- ***Streaming workflows*** leverage the AAA ("Any Data, Anytime, Anywhere") service [27] to deliver user-requested data for interactive analysis under strict latency constraints. Such workflows rely on XRootD's [28] remote-read interface to fetch only the requested file segments on demand, without replicating the whole file to save network and storage resources. A typical streaming request—such as fetching a 13 GB analysis dataset—spawns 20–50 concurrent 10 Gb/s flows through XRootD's remote-read interface, delivering only the requested segments in under 6 seconds. The two workflows exhibit fundamentally different traffic patterns: file transfers generate long-lived flows with sustained data rates, while streaming workflows spawn hundreds of short-lived, bursty flows for interactive data access. This stark difference necessitates *workflow-level* classification—identifying and grouping all flows from a single file transfer or streaming session—to ensure QoS policies apply consistently across the entire workflow rather than fragmenting decisions across individual flows.

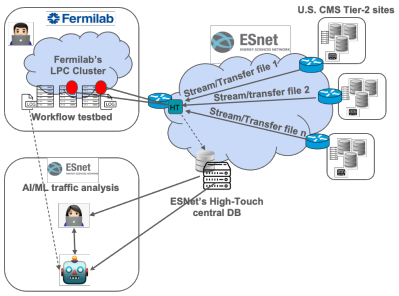

Fig. 1. Experimental Setup

Experimental setup. As illustrated in Figure 1, a local computing cluster serves as a workflow testbed while ESnet's High-Touch service, deployed at Fermi lab's border routers, captures all network traffic and aggregates network flows and performance metrics in a central database. File transfer and streaming workflows, executed on cluster nodes, move data between Fermilab and geographically distributed sites. For the file transfer CMS workflow, we used xrdcp [29].For the streaming workflow, we leveraged multiple remote sites for data access while utilizing the CMS cluster at Fermilab. Streaming uses *uproot* and *awkward* [30] to stream and analyze ROOT [31] files stored on CMS servers across the US CMS without requiring local file storage.

High-Touch platform. High-Touch is a system developed by ESnet that uses programmable hardware, combined with software running on commodity compute servers, to provide high-fidelity visibility into network traffic. It provides flow- and packet-level information on observed traffic. High-Touch is capable of operating at line rate on multiple 100Gbps and 400Gbps links without resorting to data reduction techniques such as packet sampling. ESnet has deployed High-Touch servers at a majority of its network edges, including Fermilab.

Related work. Prior network classifiers have almost exclu-

sively treated each flow in isolation, building features or images over complete flows (e.g., FlowPic's per-flow histograms [22], NeutRM's byte-sequence models [10]). None of these encodings introduce an explicit dimension for multiple concurrent flows: they build 2D representations *per flow* (e.g., packet index vs. size or time) and classify each flow independently. In contrast, WIW adds a flow axis F and aligns flows by time intervals, so the model can directly see cross-flow bursts and shared congestion signatures among flows that belong to the same workflow. Early packet-level methods for real-time use [32], [33] reduce latency but still operate on single flows. More recent work applies CNNs and LSTMs to network traces [34], [35], yet these studies focus on post-hoc analytics rather than low-latency inference in scientific network environments. A few efforts target Science DMZ "elephant" flows [16], but still rely on per-flow heuristics or non-overlapping time bins, thus missing the cross-flow burst correlations that emerge under heavy concurrency. Scientific workflows orchestrate parallel flows with correlated behavior patterns that require collective classification—a limitation that leads to fragmented QoS decisions and missed inter-flow patterns in production deployments. In contrast, our pipeline ingests multiple active flows into overlapping windows (created by our *WIW* mechanism—stacking raw-header bytes and timestamps into a single tensor—enabling workflow-level classification that captures inter-flow dynamics while maintaining low latency under operational load.

Classical ML baselines and deep packet inspection. Classical ML traffic classifiers are typically built on hand-crafted features computed independently for each 5-tuple flow, without explicitly modeling interactions among concurrent flows [36] [37]. Single-flow ML solutions do not address the core challenge we target: accurately and in-time identifying and differentiating scientific workflows that consist of hundreds, even thousands, of concurrent flows. Therefore we do not include them as comparison baselines. We do not compare our approach with DPI solutions because our captures retain only packet headers without payloads. Effective DPI requires full or near-full payload and flow reassembly, which are not available in our monitoring setup.

III. Flow Classification Pipeline

In this section, we present our end-to-end pipeline for traffic classification and detail individual steps like data preprocessing, sliding-window segmentation, and the training of interpretable deep learning models.

Pipeline Overview. Figure 2 presents our end-to-end pipeline for classifying network traffic using packets from multiple concurrent flows. The pipeline starts with the *Data Processing* stage, in which raw packet header captures from ESnet's High-Touch servers are processed in the following steps: flow reconstruction, packet extraction/truncation, and sliding-window segmentation. Initially, TCP flows are reconstructed by grouping packets sharing the same 4-tuple (source IP, source port, destination IP, destination port). We opt for flow reconstruction because using raw traffic and interleaving

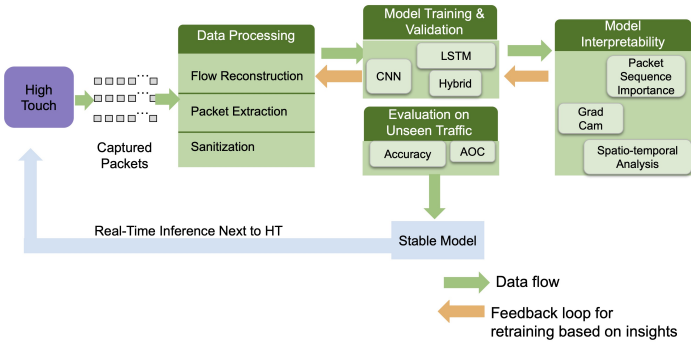

Fig. 2. End-to-end pipeline for traffic classification in high-performance scientific networks, where raw packet captures are used to train deep learning models for differentiating between data streaming and file transfer flows.

packets would dilute meaningful temporal and spatial patterns crucial for accurate classification. Each flow is then truncated to the first 100 packets. For flows with fewer than 100 packets, we pad the sequence by appending additional packets where each packet's values are set to 0, ensuring that every flow maintains a uniform length of 100 packets. Depending on the sliding-window configuration, we analyze up to the first 100 packets of each flow—a prefix we found to preserve enough temporal and spatial structure for accurate classification without introducing prohibitive inference latency IV. Figure 3 illustrates this preprocessing sequence, from raw packet captures to structured input segments ready for model ingestion. We propose a sliding window representation called

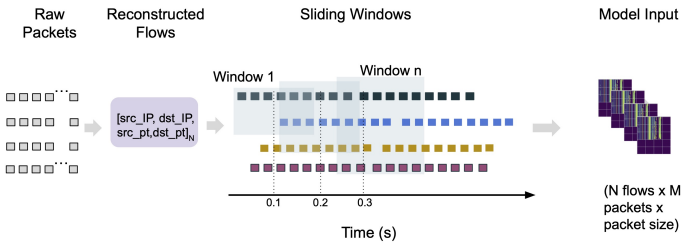

Fig. 3. The *workflow identification window (WIW)* mechanism capturing multiple parallel flows: raw packets are first grouped by flow, then segmented into overlapping time intervals that preserve inter-flow dynamics before feeding them into our deep learning models.

the Workflow Identification Window (WIW) that captures flow interdependencies crucial for real-world scientific workflows where flows overlap and interact, affecting packet timing, resource allocation, and congestion. By aggregating packets from a target flow with those from concurrently active flows within a 0.1-second interval, our approach enables models to extract local spatial features (e.g., short-term bursts in inter-arrival times and TCP window values that appear simultaneously across multiple flows in the same 0.1-s window) and longer-range temporal dependencies (e.g., how those burst patterns persist or evolve over the first tens of packets of a flow). Next, during the *Model Training and Validation* stage, we construct each input using our *WIW* approach as a 4-D tensor of shape

$$(C, F, P, B)$$

where

- $C = 2$ is the number of channels (raw-header bytes *and* normalized packet timestamps),
- F is the max number of concurrent "flows" per window,
- P ("packets") is the max packets per flow (e.g. 100),
- B ("bytes") is the header length (e.g. 100 bytes).

In our notation this is the "2×F×P×B" representation (often called 2×N×B when F=1). For example, a 2×10×100×100 tensor encodes up to 10 flows, each with 100 packets of 100 bytes, and two channels: one for the raw byte values and one for the time channel. We then train three architectures on the *WIW*-obtained windows: (1) CNN: treats the 4-D tensor as a multi-channel image, convolving across the (F×P) spatial dimensions to extract local byte-pattern and inter-flow burst features. (2) LSTM: flattens each flow's P packets into a sequence (time channel + bytes) and propagates hidden states, aiming to capture long-range temporal dependencies. (3) Hybrid CNN-LSTM: first applies convolutional feature maps over (F×P), then sequences those features through an LSTM block for combined spatial-temporal modeling.

In *Evaluation on Unseen Traffic* we assess our trained models on new, previously unseen traffic, that was collected several days after the traffic used during training and validation phases. We utilize standard performance metrics such as accuracy, precision, recall and F-1 score [38] [24], ensuring that our models generalize effectively under realistic scenarios. To enhance transparency in the model's decision-making process, we employ interpretability tools like Grad-CAM and SHAP to identify not only which packet fields influence classification but also how spatial and temporal patterns in network traffic drive the model's predictions. The stable model can be deployed adjacent to HT servers as shown by the blue arrow in Figure 2, enabling real-time inference on captured packets. A feedback loop continuously triggers retraining when new traffic patterns emerge. For example, if interpretability analyses (e.g., SHAP or Grad-CAM) reveal shifts in the importance of specific packet fields, or if classification accuracy drops below a predefined threshold (e.g., 90%), the loop initiates retraining so the model can adapt. This ongoing adjustment—triggered by both performance metrics and interpretability insights—keeps the classification system robust, accurate, and responsive to dynamic network conditions.

Traffic generation through workflow execution on dedicated testbed. Our training, testing and validation datasets are captured through running the two HEP workflows II on our dedicated testbed II and installing subnet-based filtering rules on the HT servers that are attached to the border routers. For each workflow, we separately record approximately 2 GB of streaming flows and 5 GB of file-transfer flows, ensuring that real traffic patterns (e.g., bursty streaming vs. more continuous file transfer) are reflected in the dataset. Table I provides a summary of our training data. The collected dataset contains almost three times the number of streaming flows compared to file transfer flows leading to a class imbalance favoring streaming flows. To address imbalance, we apply downsampling to the streaming class and use weighted loss functions during training. To prevent bias during training and evaluation stages we perform a sanitization check that detects and removes duplicate flows, ensuring that the training and test sets contain distinct instances.

TABLE I
SUMMARY OF OUR TRAINING DATASET.

Workflow	Label	Number of Flows	Data Volume
Data Streaming	0	6,510	2 GB
File Transfer	1	2,670	5 GB

Implementation and hardware setup. Our pipeline is implemented in Python using Scapy for PCAP parsing, scikit-learn for dataset partitioning and classical baselines, and TensorFlow/Keras for deep models, trained with Adam and binary cross-entropy. All experiments ran on a dual-socket Intel Xeon server with 768 GiB RAM and 10/100 GbE connectivity under Ubuntu 22.04. ‡

IV. RESULTS

Models configuration. We evaluate our three selected architectures—LSTM, CNN, and Hybrid—for early traffic classification between file transfer and streaming flows under various window length scenarios obtained through our *WIW* mechanism. All tests use the WIW 2×F×P×B window representation defined in Section III. We sweep two parameters in our *WIW* windows: the *number of concurrent flows* $F \in \{1, 3, 5, 10, 50, 100\}$ and the *number of packets per flow* $P \in \{20, 50, 100\}$. Each window is represented as a two-dimensional grid of size $F \times P$, where each **column** contains the packet headers for one flow (up to P packets in sequence) and each **row** aligns the same packet index across all F flows. Because we truncate each flow to its first 100 packets, a window with $P = 50$ covers two full segments per flow (i.e., two "passes" through the initial 100 packets), whereas $P = 20$ yields five shorter segments—enabling more frequent classification decisions with reduced per-window context. While larger windows capture more detailed flow information, they may delay classification and undermine timely resource allocation. We tuned the batch size and learning rate through preliminary experiments to balance model complexity, stability, and accuracy. Table II summarizes the hyperparameter configurations across model types.

Table III and Table IV compare CNN and LSTM performance on our 2×N×B packet-header windows. Across all settings, the CNN clearly outperforms the LSTM: for example, with 100 packets per window and 100 concurrent flows the CNN maintains 98.7% accuracy, whereas the LSTM plateaus around 83.9%. The performance gap stems from how each model ingests our representation. CNNs apply localized 2D filters over the "image" of raw bytes (first channel) and normalized timestamps (second channel), picking up spatially co-occurring patterns across different flows—such as simultaneous bursts that directly encode inter-flow interactions. In

‡Upon publication, we will release our data-processing scripts and model configuration files to facilitate reproducibility.

contrast, an LSTM unfolds the same data as a flat time series, updating its hidden state packet by packet. When multiple flows interleave, the strict packet order signal is diluted and the LSTM's sequential gates cannot easily disentangle overlapping streams; as a result, it fails to learn the spatial "hot-spots" that the CNN detects. Table V reports our CNN–LSTM hybrid performance.At low concurrency (3 flows) the hybrid peaks at 98.9%, slightly above the standalone LSTM's 97%—showing that the CNN front-end can pre-extract useful spatial features that the LSTM then sequences effectively. However, as concurrency ramps to 50–100 flows, the hybrid's accuracy collapses to 83.9%, mirroring the LSTM's dropoff.

Evaluation on unseen traffic. To evaluate generalization, we collected a fresh 2 GB capture of 4670 previously unseen streaming flows one week after our original training run. We then applied our largest-window configurations (100 packets/flow with 50 and 100 concurrent flows).Although these large-window settings did not yield the highest accuracy in our initial search, they allow us to determine whether the models can sustain high performance in operational environments with multiple concurrent flows. Although our new dataset contains exclusively streaming flows, it still reflects a critical, real-world scenario where only one of the two workflows is executed on available resources.

TABLE II
HYPERPARAMETER CONFIGURATIONS FOR LSTM AND CNN MODELS.

Window Size	Stride	Concurrent Flows	Layers/Filters	Batch Size	Learning Rate
LSTM					
10	5	{1,3,5,10,50,100}	[64, 32]	16	0.001
20	10	{1,3,5,10,50,100}	[128, 64]	32	0.0005
30	10	{1,3,5,10,50,100}	[256, 128]	32	0.0005
40	10	{1,3,5,10,50,100}	[256, 128]	32	0.0005
50	10	{1,3,5,10,50,100}	[256, 128, 64]	16	0.0001
100	10	{1,3,5,10,50,100}	[256, 128, 64]	16	0.0001
CNN					
10	5	{1,3,5,10,50,100}	[32, 64]	16	0.001
20	10	{1,3,5,10,50,100}	[64, 128]	32	0.0005
30	10	{1,3,5,10,50,100}	[64, 128]	32	0.0005
40	10	{1,3,5,10,50,100}	[64, 128]	32	0.0005
50	10	{1,3,5,10,50,100}	[64, 128, 64]	16	0.0001
100	10	{1,3,5,10,50,100}	[64, 128, 64]	16	0.0001

TABLE III
CNN PERFORMANCE RESULTS.

Max Packets	Max Flows	Accuracy	Precision	Recall	F1 Score
20	3	0.98	0.99	0.99	0.99
20	5	0.97	0.99	0.97	0.98
20	10	0.98	0.99	0.99	0.98
50	3	0.98	0.98	0.99	0.99
50	5	0.98	0.98	0.99	0.99
50	10	0.98	0.98	0.99	0.99
100	3	0.98	0.98	0.99	0.99
100	5	0.98	0.99	0.99	0.99
100	10	0.98	0.98	0.99	0.99
100	50	0.98	0.99	0.98	0.98
100	100	0.98	0.99	0.99	0.97

TABLE IV
LSTM PERFORMANCE RESULTS.

Max Packets	Max Flows	Accuracy	Precision	Recall	F1 Score
20	3	0.97	0.98	0.99	0.98
20	5	0.83	0.83	1.00	0.91
20	10	0.83	0.83	1.00	0.91
50	3	0.92	0.92	0.98	0.95
50	5	0.83	0.83	0.98	0.90
50	10	0.83	0.83	1.00	0.91
100	3	0.83	0.83	0.98	0.90
100	5	0.83	0.81	1.00	0.91
100	10	0.83	0.81	1.00	0.91
50	100	0.83	0.83	1.00	0.91
100	100	0.83	0.83	1.00	0.91

TABLE V
HYBRID (CNN+LSTM) PERFORMANCE RESULTS.

Max Packets	Max Flows	Accuracy	Precision	Recall	F1 Score
20	3	0.98	0.99	1.00	0.99
20	5	0.98	0.99	0.99	0.99
20	10	0.97	0.99	1.00	0.99
20	50	0.84	0.84	1.00	0.91
20	100	0.84	0.84	1.00	0.91
50	3	0.98	0.99	0.99	0.99
50	5	0.99	0.99	0.99	0.99
50	10	0.99	0.99	0.99	0.99
50	50	0.84	0.84	1.00	0.91
50	100	0.84	0.84	1.00	0.91
100	3	0.98	0.99	0.99	0.99
100	5	0.99	0.99	0.99	0.99
100	10	0.99	0.99	0.99	0.99
100	50	0.84	0.84	1.00	0.91
100	100	0.84	0.84	1.00	0.91

TABLE VI
PERFORMANCE ON UNSEEN STREAMING TRAFFIC FOR CNN AND LSTM.

Model	Max Packets	Max Flows	Accuracy	F1 Score
CNN	100	50	0.83	0.84
CNN	100	100	0.84	0.84
LSTM	100	50	0.74	0.74
LSTM	100	100	0.73	0.74

Table VI summarizes the classification results for the two largest models evaluated on the new dataset. Under the 50-flow configuration, the CNN achieves an accuracy of 0.83, while the LSTM attains only 0.74—a 9% improvement for the CNN. With 100 concurrent flows, the CNN's accuracy increases to 0.84, compared to 0.73 for the LSTM (an 11% difference). We hypothesize that streaming flows often display bursty, irregular packet arrivals. When multiple flows are interleaved, these irregularities produce distinct spatial patterns. Since our unseen dataset comprises only streaming flows, the CNN—which is designed to capture local spatial features—achieves significantly better performance. In contrast, LSTMs are optimized for modeling clear sequential dependencies within a single time series. The 84% accuracy on unseen streaming-only traffic, while lower than our 98% training accuracy, remains highly suitable for production deployment for several reasons: First, this represents a worst-case scenario—our model was trained on file-transfer and streaming workflows but tested exclusively on streaming workflows collected a week later. Despite this distribution shift, the CNN maintains 84% accuracy demonstrating robustness to temporal drift and class imbalance. Second, our multi-flow approach provides inherent resilience through workflow-level consensus:

Since CMS workflows spawn tens to hundreds of parallel flows, classifying at the workflow level means aggregating predictions across multiple flows. Even with 84% per-window accuracy, the majority vote across a workflow's constituent flows yields higher effective accuracy for QoS decisions. A file-transfer workflow with 100 parallel flows is unlikely to be misclassified when 84 of its flows are correctly identified. Third, in production environments, our feedback loop (shown in Figure 2) would trigger retraining when accuracy drops, preventing such degradation. The 84% accuracy represents performance without any adaptation—with continuous learning, we expect to maintain higher accuracy.

Inference latency. We measured the end-to-end per-window classification latency—including workflow identification window (WIW) build time and model forward pass—on our CPU-only testbed (described in III). For each model we ran 1,000 single-window forward passes. Across all model and concurrency settings—CNN and LSTM inference remains firmly in the tens of milliseconds (roughly 20 – 50 ms per window). All latencies remain well below our 0.1s window duration. If more powerful AI computing systems will be available, the inference delay can be further reduced. These results demonstrate that our solution can be used to differentiate workflows in near real-time on CPU-only servers under production load.

General applicability. Although we evaluated on CMS workflows, our packet-level sliding-window pipeline makes no assumptions beyond having two labeled traffic classes; by simply retraining on new packet-header captures and adjusting the sliding-window parameters F, P, the same end-to-end pipeline can be repurposed for other binary classification tasks on any mix of latency-sensitive vs. bulk workflows.

Interarrival time delay analysis. Figure 4 shows the distribution of interarrival times between the first packet and subsequent packets for file-transfer and streaming flows. File-transfer flows have tight interarrival times, peaking around 1 ms, reflecting steady, continuous transmission. Streaming flows exhibit a much wider spread, with delays up to tens of milliseconds, consistent with bursty behaviour. This clear separation means that early timing alone is a strong discriminative signal: tight, regular interarrivals indicate file transfers, whereas highly variable intervals suggest streaming.

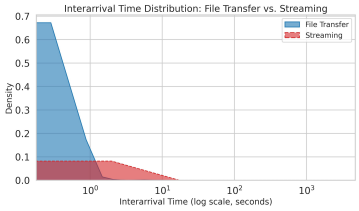

Fig. 4. Distribution of interarrival times for file-transfer and streaming flows.

SHAP. To elucidate how our model differentiates traffic types, we performed interpretability analysis on an LSTM instance using a 50-packet window. Figure 5 shows the average absolute SHAP values for each packet position (aggregated over all fields and samples), clearly indicating that the earliest packets hold the most predictive importance. Emphasis on early packets confirms that decisions are driven by early-flow behaviour rather than late packets. This aligns with our classical-ML baselines, where early interarrival-time statistics and TCP window mean/std emerge as the most informative features: the model relies primarily on timing and congestion-control behaviour, not opaque byte patterns.

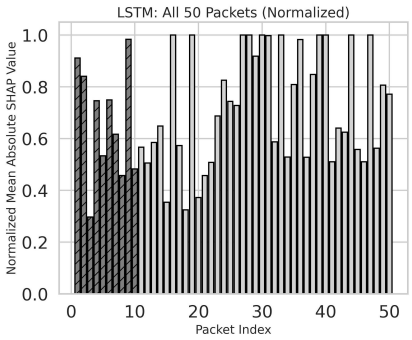

Fig. 5. Normalized SHAP values of an LSTM model across the first 50 flow packets. The first 10 packets consistently exhibit higher predictive importance.

Grad-CAM visualization for interpreting CNN predictions. To shed light on our CNN model's decision-making, we employ Grad-CAM to highlight the regions in the input that most strongly influence predictions. In Figure 6, each pixel in the heatmap represents a packet–byte position—warmer colors (reds/yellows) denote high importance, while cooler hues (blues) indicate minimal influence. Notably, the heatmap reveals bright "hot spots" concentrated in the early packets indicating that the model primarily relies on early-flow timing behaviour (e.g., inter-arrival times) to make decisions. These areas underscore the key differences in the initiation of flows: file transfers typically exhibit rapid, steady packet sequences (with distinctive early patterns), whereas streaming applications display more varied, sporadic arrivals.

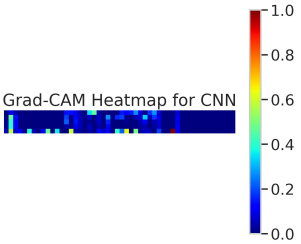

Fig. 6. Grad-CAM heatmap from the CNN final convolutional layer.

V. Conclusion & Future Work

In this paper, we introduced a production-validated, multi-flow sliding-window classification solution that ingests raw packet headers from traffic captures under operational load—preserving inter-flow burst correlations in a single tensor. By feeding these windows to CNN, LSTM, and hybrid models, we demonstrated 98% accuracy even with 100 concurrent flows and maintained low inference latency in real network conditions. When applied to a fresh, streaming-only capture collected days later, our pipeline sustained >84% accuracy, confirming it learns robust cross-flow patterns rather than overfitting to a single workload. Finally, integrated SHAP and Grad-CAM analyses pinpoint the precise packet-byte and timing features driving each decision.

We acknowledge that our evaluation used controlled CMS traces with solely two workflow types; we will expand our solution to include background traffic, finer-grained application classes, and diverse network environments. As a next step, we will benchmark *WIW* against alternative deep-learning and

multi-window baselines that also exploit multi-flow context, in order to isolate how much of the gain comes from the *WIW* representation versus the choice of model architecture. Future work will also focus on: adaptive ML techniques that recalibrate models on the fly as traffic patterns evolve without reconstructing individual flows, deployment and scaling across multiple CMS Tier-2 sites to optimize resource allocation end to end, integrating our inference engine directly into SDN/NFV controllers for closed-loop, real-time network orchestration. Together, these advances pave the way for truly self-driving scientific networks capable of guaranteed QoS and dynamic throughput optimization in low-latency environments.

REFERENCES

[1] Lawrence Berkeley National Laboratory, "Energy Sciences Network (ESnet)," https://www.es.net, 2025, u.S. Department of Energy's high-performance network serving the research community.

[2] CMS Collaboration, "The cms experiment at the cern lhc," *JINST*, 2008.

[3] "Large hadron collider," https://home.cern/science/accelerators/large-hadron-collider, 2025, CERN.

[4] "sflow.org," https://sflow.org, accessed: 2025-03-16.

[5] B. Mah, R. Cziva, C. Guok, and Y. Kumar, "ESnet6 High Touch Services ," https://sc20.supercomputing.org/app/uploads/2020/12/06_sc20_xnet_mah_cziva_kumar_esnet6.pdf, 2020, presented at SC20 Conference, virtual event, November 13, 2020.

[6] B. Claise, "Rfc 3954: Cisco systems netflow services," USA, 2004.

[7] I. Koukoulis, I. Syrigos, and T. Korakis, "Self-supervised transformer-based contrastive learning for intrusion detection systems," in *IFIP 2025*.

[8] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "Flowtransformer: A transformer framework for flow-based network intrusion detection systems," *arXiv preprint arXiv:2304.14746*, 2023.

[9] Z. Hou, "A cnn-based encrypted network traffic classifier," *arXiv preprint*, 2021. [Online]. Available: https://arxiv.org/abs/2102.12345

[10] A. Azzouni and G. Pujolle, "Neutm: A neural network-based framework for traffic matrix prediction in sdn," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018.

[11] J. Feng, X. Chen, R. Gao, M. Zeng, and Y. Li, "Deeptp: An end-to-end neural network for mobile cellular traffic prediction," *IEEE Network*, vol. 32, no. 6, pp. 108–115, 2018.

[12] Q. He, A. Moayyedi, G. Dán, G. P. Koudouridis, and P. Tengkvist, "A meta-learning scheme for adaptive short-term network traffic prediction," *IEEE Journal on Selected Areas in Communications*, vol. 38, 2020.

[13] M. A. Abdulraheem *et al.*, "Software defined networking based network traffic classification using machine learning techniques," *Scientific Reports*, vol. 14, no. 1, p. 20367, Aug 2024.

[14] M. Pandey, M. Fischer-Abaigar *et al.*, "Mitigating domain shifts: An introduction to address domain shifts through different adaptation paradigms," in *Proceedings of the Workshop on Domain Adaptation*. Berlin, Germany: ZUSE School RELAI, 2025. [Online]. Available: https://zuseschoolrelai.de/blog/mitigating-domain-shifts/

[15] A. Azab, M. Khasawneh, S. Alrabaee, K. K. R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges," *Digital Communications and Networks*, vol. 8, no. 6, pp. 936–947, 2022.

[16] J. Chen, J. Breen, J. M. Phillips, and J. Van der Merwe, "Practical and configurable network traffic classification using probabilistic machine learning," *Cluster Computing*, vol. 25, pp. 2839–2853, 2022.

[17] J. Waczyńska, E. Martelli, S. Vallecorsa, E. Karavakis, and T. Cass, "Convolutional lstm models to estimate network traffic," *arXiv preprint arXiv:2107.02496*, 2021, available: https://arxiv.org/abs/2107.02496.

[18] I. Mahmud, G. Papadimitriou, C. Wang, M. Kiran, A. Mandal, and E. Deelman, "Elephants sharing the highway: Studying tcp fairness in large transfers over high throughput links," in *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 806–818. [Online]. Available: https://doi.org/10.1145/3624062.3624594

[19] B. Gyevnar, N. Ferguson, and B. Schafer, *Bridging the Transparency Gap: What Can Explainable AI Learn from the AI Act?* IOS Press, Sep. 2023. [Online]. Available: http://dx.doi.org/10.3233/FAIA230367

[20] A. K. Sahu, S. Sharma, M. Tanveer, and R. Raja, "Iot network traffic classification using machine learning algorithms: An experimental analysis," *IEEE Internet of Things Journal*, vol. 8, no. 6, 2021.

[21] C.-S. Duong, H.-A. Tran, and T. X. Tran, "Continuous select-and-prune incremental learning for encrypted traffic classification in distributed sdn networks," in *2024 IEEE Conference on Local Computer Networks (LCN)*, 2024. [Online]. Available: https://doi.org/10.1109/LCN.2024.10639717

[22] T. Shapira and Y. Shavitt, "Flowpic: A generic representation for encrypted traffic classification and applications identification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, 2021.

[23] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017. [Online]. Available: https://arxiv.org/abs/1705.07874

[24] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7

[25] E. Vaandering, "Transitioning cms to rucio data management," in *24th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2019)*, ser. EPJ Web Conf., vol. 245, 2020, p. 04033.

[26] M. Barisits, T. Beermann, F. Berghaus, and *et al.*, "Rucio: Scientific data management," *Computing and Software for Big Science*, vol. 3, 2019.

[27] K. Bloom, T. Boccali, B. Bockelman, and *et al.*, "Any data, any time, anywhere: Global data access for science," *IEEE Data Eng. Bull.*, vol. 38, no. 4, 2015, arXiv:1508.01443.

[28] F. Furano, V. Garonne, W. Hanlon, V. Khotilovich, B. Holzman, V. Ocone, and J. Shank, "The xrootd data server for the cms experiment," in *Proceedings of the 2007 International Conference on Computing in High Energy and Nuclear Physics (CHEP 2007)*, 2007. [Online]. Available: https://cds.cern.ch/record/1023171

[29] "xrdcp(1) manual page," https://www.huge-man-linux.net/man1/xrdcp.html, 2025, accessed: March 31, 2025.

[30] J. Pivarski, I. Osborne, I. Ifrim, H. Schreiner, A. Hollands, A. Biswas, P. Das, S. Roy Choudhury, N. Smith, and S. Goyal, "Uproot: ROOT I/O in pure Python and NumPy," 2024. [Online]. Available: https://github.com/scikit-hep/uproot5

[31] R. Brun and F. Rademakers, "ROOT—an object oriented data analysis framework," *Nuclear Instruments and Methods in Physics Research Section A*, vol. 389, pp. 81–86, 1997, proceedings AIHENP'96 Workshop, Lausanne, September 1996.

[32] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.

[33] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of ACM SIGMETRICS*, 2005.

[34] P. Jurkiewicz, B. Kadziołka, M. Kantor, J. Domżał, and R. Wójcik, "Machine learning-based elephant flow classification on the first packet," *IEEE Access*, vol. 12, pp. 105 744–105 760, 2024.

[35] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, 2020.

[36] J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang, "An accurate traffic classification model based on support vector machines," *Netw.*, vol. 27, no. 1, p. n/a, Jan. 2017. [Online]. Available: https://doi.org/10.1002/nem.1962

[37] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018.

[38] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.