Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

CAPE-Net: Conformal, Adaptive, and Drift-Aware Early-Exit Classification of Encrypted Traffic at ISP Scale

Farzam Rezaei

farzam.rezaei@uam.es

Universidad Autónoma de Madrid

Jorge E. López de Vergara

Universidad Autónoma de Madrid

Luis de Pedro

Universidad Autónoma de Madrid

Iván González

Universidad Autónoma de Madrid

Research Article

Keywords: Encrypted traffic classification, Early-exit neural networks, Conformal prediction, Out-of-distribution detection, Concept drift, Network measurement

Posted Date: August 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10360175/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

CAPE-Net: Conformal, Adaptive, and Drift-Aware Early-Exit Classification of Encrypted Traffic at ISP Scale

Farzam Rezaei*, Jorge E. López de Vergara, Luis de Pedro, Iván González

Departamento de Tecnología Electrónica y de las Comunicaciones, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain.

*Corresponding author(s). E-mail(s): farzam.rezaei@uam.es;
Contributing authors: jorge.lopez_vergara@uam.es; luis.depedro@uam.es; ivan.gonzalez@uam.es;

Abstract

The encryption of nearly all Internet traffic through TLS, encrypted client-hello, and QUIC has pushed encrypted-traffic classification onto packet-sequence metadata—the sizes, directions, and inter-arrival times of first packets in a flow. For a network operator, however, accuracy alone is insufficient: a classifier deployed on a backbone link must decide early, carry a quantifiable per-class error guarantee, recognise applications it was never trained on, and remain maintainable as network traffic drifts. To deal with these problems, we present CAPE-Net, an early-exit classifier that treats its exit structure not only as a mere efficiency device but also as a multi-purpose inference instrument. Built on a compact recurrent backbone with eighteen early-exit checkpoints, CAPE-Net (i) certifies a per-class confidence threshold by a Learn-Then-Test procedure, yielding a finite-sample, distribution-free precision guarantee; (ii) abstains on unknown applications via a free-energy score with a confidence-first gate; and (iii) detects per-class concept drift with cumulative-sum monitors and repairs flagged classes through a bias-only update that cannot harm the rest of the model. On the 180-application CESNET-TLS-Year22 corpus, the accuracy guarantee holds for about 98% of classes on a held-out week while the typical flow is classified from roughly seven to eight packets—and the most confident classes from as few as three—and the maintenance loop slows drift degradation over a ten-week horizon at a cost of seconds per update. To assess the generability of the system, it has also been retrained from scratch on a year-long QUIC dataset (CESNET-QUICEXT-2025). The same design transfers with

the guarantee intact, committing later in line with QUIC's longer, padded flows. CAPE-Net shows that early, calibrated, and maintainable classification at ISP scale is achievable within a single architecturally disciplined model.

Keywords: Encrypted traffic classification, Early-exit neural networks, Conformal prediction, Out-of-distribution detection, Concept drift, Network measurement

1 Introduction

Internet service providers classify network traffic to allocate capacity, enforce quality-of-service policies, plan infrastructure, and detect threats. For decades this rested on port numbers or payload inspection, but the near-universal adoption of TLS, the arrival of encrypted client-hello, and the rapid growth of QUIC have closed that window: the bytes an operator most wants to read are now encrypted, and increasingly even the handshake metadata is obscured. Modern encrypted- traffic classification (ETC) therefore works from what remains observable without decryption—the sizes, directions, and inter-arrival times of the first packets of a flow. A substantial body of recent work has shown that these packet-sequence features are surprisingly discriminative, supporting accurate classification with byte-segment models [1], convolutional neural networks [2], graph neural networks [3–5], transformers [6, 7], multi-modal fusion [8, 9], and automated architecture search [10].

Accuracy on a held-out test split, however, is not the property a network operator most needs. A classifier deployed on a live backbone link faces three demands at once. First, it must decide *early*: a verdict that arrives only after a flow completes is of little use for real-time quality policing or routing, so the system should commit on as few packets as possible. Second, it must be *reliable* in a quantifiable sense—not merely accurate on average, but carrying a controlled, per-class error rate, because the cost of a confident misclassification falls unevenly across applications, and an aggregate accuracy figure hides that. Third, it must remain usable as network traffic *drifts*: application versions, provider behaviour, and the very mix of applications change over weeks and months, and retraining a large model from scratch at ISP scale is expensive and disruptive. To these three, we add a fourth, often-ignored reality: a deployed classifier will find flows from applications it was never trained on, and forcing every such flow into a known class is worse than admitting ignorance.

Existing works address these demands in isolation rather than together. The line closest to ours pursues earliness: FastFlow [11] classifies flows from a minimal number of time-series packets, TINIEE [12] resolves easy flows at shallow stages of an in-network model, and a range of elephant-flow methods commit from the first packet or a small prefix [13–15]. Yet, these systems commit on a single global confidence or volume threshold; none calibrates its commit point per class against a precision target, and none certifies the precision of the early decisions it emits. A second line hardens classifiers against perturbation—packet reordering [16], sequence noise [17], adversarial inputs [18], and distributional variation through data augmentation [19]—but robustness to perturbation is not the same as a guarantee on decision quality, and that work does not provide one. A third line confronts the non-stationarity of traffic, notably through transfer and domain-recognition pretraining on large, long- horizon QUIC data [20] and through network-level study of newly emerging applications [21], but

treats drift as a representation- learning problem rather than as an online maintenance problem with a bounded compute budget. In summary, to the best of our knowledge, no existing system delivers early commitment, per-class certified precision, built-in unknown-application detection, and lightweight drift maintenance within a single model—still less at the scale of a real ISP application taxonomy of well over a hundred classes.

To solve these matters, this paper presents CAPE-Net (Conformal Adaptive Packet-Efficient Network), a system built on a single observation: the early- exit structure of a multi-checkpoint classifier is not merely an efficiency device but a multi-purpose inference instrument. The same per-exit heads that let a flow commit early can also carry a calibrated per-class precision guarantee, an abstention score for unknown traffic, and—through their per-class exit-depth and precision statistics—a signal of concept drift. Built around a compact recurrent backbone over packet-sequence metadata with eighteen early-exit checkpoints, CAPE-Net realises this view through three coupled mechanisms, evaluated on the full 180-application CESNET-TLS-Year22 corpus and validated later on a year-long QUIC dataset (CESNET-QUICEXT-2025):

1. **Per-class conformal precision calibration.** Rather than committing on a global confidence threshold, CAPE-Net certifies a per-class confidence threshold by a Learn-Then-Test procedure with multiplicity correction, yielding a finite-sample, distribution-free guarantee that the precision of committed flows meets a target floor. On the held-out week, the guarantee is achieved for roughly 98% of the 180 classes, lifting the precision of committed flows to approximately 0.99 while exposing the coverage cost.
2. **Native unknown-application detection.** A free-energy score read at a fixed checkpoint lets the model abstain—emitting an "unknown" verdict—on flows from applications outside its training set, at a controlled in-distribution false-positive rate. A confidence-first gate orders this abstention correctly relative to early commitment, so that the typical confident flow is never delayed for the unknown test.
3. **Drift-aware, safe maintenance.** Per-class cumulative-sum monitors track shifts in exit depth and realised precision and flag degrading classes; flagged classes receive a bias-only update that, by construction, cannot alter the decision function for any non-flagged class. Over a ten-week drift horizon, this recovers a small, consistent margin each week at a cost of seconds per update—a maintenance net rather than a costly retraining.

Underpinning all three is the early-exit backbone itself, whose fine-grained checkpoint set lets the typical TLS flow be classified from roughly its first seven to eight packets, with the most confident classes certified to commit on as few as three—near the accuracy of a full thirty-packet model at a fraction of the observation cost. Retrained from scratch on QUIC, the same design classifies a held-out future week at comparable accuracy with the precision guarantee intact, while committing later in line with QUIC's longer, padded flows.

The remainder of this paper is organised as follows. Section 2 surveys related work on early, robust, and operational ETC. Next, Section 3 describes the CESNET-TLS-Year22 corpus and our sampling and splitting protocol. Then, Section 4 presents the CAPE-Net backbone and the three mechanisms described above. After this, Section 5 reports the in-distribution and drift evaluations, and Section 6 the QUIC study. Later, Section 7 discusses implications for operators, Section 8 states limitations and future work, and finally, Section 9 concludes the paper.

2 Related Work

We organise prior work into four strands that together motivate CAPE-Net: early and adaptive encrypted-traffic classification, the feature representations and model families used for the task, robustness, and calibration of traffic classifiers, and the open problems of drift and unknown-application handling. We deliberately position our work against the methods most comparable in goal— early commitment and operational reliability—rather than against the full breadth of intrusion-detection literature.

2.1 Early-Exit and Early-Commitment Traffic Classification

The idea of classifying a flow from as few packets as possible, rather than waiting for it to complete, is the closest line of work to ours. FastFlow [11] is the most directly comparable system: it classifies network flows from a minimal number of time-series packets using per-packet sequential decisions, and is explicitly designed for robustness, reporting a mean of around eight packets per decision. It commits, however, on a single global confidence criterion rather than per-class calibrated thresholds, and is evaluated on a small label space. TINIEE [12] pursues early exit from a different angle, embedding an early-exit strategy inside in-network inference so that "easy" flows are resolved at shallow stages of a programmable data plane. Li et al. [22] propose a multi-model serial framework that targets early classification by chaining models of increasing depth. Several works approach earliness through the lens of elephant-flow detection, where the goal is to identify a small number of high-volume flows quickly: classification on the very first packet with neural networks [13], random-forest detection under programmable data-plane constraints [14], and online elephant-flow prediction for switch-based load balancing [15]. Early-stage convolutional feature extraction for mining-traffic detection [23] likewise emphasises decisions from limited observation.

What unites these methods is that earliness is treated as a single budget—commit once a global confidence or volume threshold is met. None calibrates the commit point per class against a precision target, and none provides a finite-sample guarantee on the precision of the resulting early decisions. This is the gap our per-class conformal calibration addresses.

2.2 Feature Representations and Model Families

A second strand concerns how encrypted flows are represented and which model families are applied. Byte- and packet-segment models such as EBSNN [1] learn directly from byte segments of packets, while the two-phase approach of Wang et al. [24] balances speed and accuracy by cascading a fast first stage with a more accurate second stage. Graph-based representations have become prominent: EC-GCN [3] applies multi-scale graph convolution to encrypted-traffic flows, WFF-EGNN [4] classifies "weaved" flow fragments with an ensemble of graph neural networks, BPF-DAG [5] fuses byte-, packet-, and flow-level features through a dynamic attributed graph, and graph-convolutional analysis has also been applied to compressed-traffic detection [25]. Transformer and attention-based models form another family, including system-level denoising transformers for behaviour identification [6], feature-free transformer classification [7], and decision-tree/transformer hybrids [26]. Multi-modal and context-aware designs fuse heterogeneous views of a flow [8, 9], and a lightweight method using contextual information [27] targets efficiency directly. Self-supervised and contrastive

objectives have been explored to reduce label dependence, including contrastive intrusion detection [28] and next-packet prediction as a non-contrastive pretext task [29]. At the extreme of automation, AutoML4ETC [10] performs neural architecture search for real-world encrypted-traffic classification, and hybrid deep-learning/automata-learning classifiers [30] combine learned and symbolic structure.

These works establish the representational backdrop for CAPE-Net. Our model deliberately retains a compact recurrent backbone over packet-sequence metadata rather than a heavier graph or transformer stack, because the early-exit and per-class calibration machinery—not raw representational capacity—is the locus of our contribution.

2.3 Robustness and Calibrated Decisions

Because traffic classifiers are deployed in adversarial and noisy conditions, a growing body of work targets robustness. The packet-sequence permutation-aware approach of Jiang et al. [16] explicitly hardens classification against packet reordering, and SimSeq [17] pursues robust TLS classification under sequence perturbation. Adversarial training has been applied to VPN traffic classification [18], data augmentation has been used to improve classification efficiency and resilience [19], and TransGraphNet [31] combines transformer and graph models for robust detection of malicious encrypted traffic. FastFlow [11] is again relevant here, as its time-binned representation is motivated partly by robustness to packet loss.

What is largely absent from this strand is calibration in the formal sense: a guarantee that the confidence a classifier attaches to a committed decision corresponds to a controlled error rate. Existing robustness work hardens models against perturbation but does not certify the precision of the decisions they emit. CAPE-Net's use of a per-class Learn-Then-Test procedure to certify a precision floor is, relative to this corpus, a distinct contribution: it turns an empirical confidence threshold into a finite-sample, distribution-free guarantee.

2.4 Drift, Unknowns, and Operational Deployment

The final strand concerns the operational realities of long-lived deployment: traffic evolves, new applications appear, and models must remain serviceable. Within this corpus, the work closest to the drift problem comes from the QUIC domain-recognition pretraining of Luxemburk et al. [20], which studies transfer across time and domains on large CESNET QUIC data, and from recent network-level studies of newly emerging application traffic such as the analysis of large-language-model assistants [21]; both underscore that the application mix a classifier faces is non-stationary. Self-supervised and augmentation-based methods [19, 29] are often justified partly by their potential resilience to such shift, though they do not measure it directly. On the deployment side, several works target the efficiency constraints of real infrastructure—in-network early exit [12], energy-efficient offloading of detection to the edge [32], and classification under programmable data-plane limits [14]—which is the operating setting CAPE-Net's lightweight, bias-only maintenance loop is designed for.

Complementary evidence comes from Ayodele and Buttigieg [33], who evaluate how deep traffic classifiers generalise across virtualised and physical network environments and find that a model trained in one domain cannot simply be transferred to the other without recalibration—with the divergence concentrated in timing-derived features such as inter-arrival times and jitter. This is directly pertinent to our setting, since our per-packet representation is built on

precisely those features, and it corroborates the recalibration-centred view of maintenance we adopt.

Handling genuinely unknown applications—traffic from classes never seen in training—is treated only implicitly in most of this corpus, typically by assuming a closed world. This motivates the explicit abstention mechanism in CAPE-Net, which emits an "unknown" verdict rather than forcing every flow into a known class.

2.5 Positioning with Respect to the State of the Art

Taken together, these four research strands illustrate that encrypted-traffic classification, uncertainty-aware prediction, unknown-application detection, and model maintenance have largely evolved as separate research directions. While early classification, rich feature representations, and robustness have each been studied extensively, their integration into a unified operational framework that simultaneously provides per-class calibrated precision, built-in unknown detection, and lightweight drift-aware maintenance at ISP scale remains largely unexplored. CAPE-Net is designed precisely to bridge these previously disconnected capabilities within a single operational framework, aimed at real-world network management.

Table 1 summarises this positioning. Rather than comparing prediction accuracy across heterogeneous datasets, it highlights the operational capabilities that distinguish CAPE-Net from representative approaches in the literature. Each comparable method occupies part of the operational space—earliness, robustness, or scale— but none combines adaptive early exit with a per-class precision guarantee, built-in unknown detection, and lightweight drift maintenance at a large application-label scale. CAPE-Net is designed to fill exactly that intersection, with the one caveat that drop-robustness, where FastFlow and the permutation-aware methods are strongest, remains future work (Section 8).

3 Dataset and Sampling

All experiments in this work have been initially conducted on a single, year-long, real-world encrypted-traffic corpus (CESNET-TLS-Year22). We deliberately restrict ourselves to one dataset rather than a patchwork of smaller benchmarks: the contributions we make— (1) a per-class precision guarantee, (2) an unknown-application detector, and (3) a temporal drift-maintenance loop—are only meaningful when evaluated against a label space and a time horizon large enough for them to matter. In this section, we describe the corpus (§ 3.1), the stratified flow-sampling protocol and the resulting train/validation/test/drift splits (§ 3.2), and the in-distribution/out-of-distribution (ID/OOD) class partition used for Contribution 2 (§ 3.3).

3.1 The CESNET-TLS-Year22

We use CESNET-TLS-YEAR22, a publicly available dataset of TCP/TLS encrypted-traffic flows collected over the full calendar year of 2022 from the national research and education network in Czesk Republic. The corpus is labelled with 180 application classes (the `APP` field); each flow additionally carries a coarser `CATEGORY` label, which we do not use as a classification target. The 180-class label space is the central reason we adopt this corpus: it is

Table 1: Positioning of CAPE-Net against representative encrypted-traffic classification methods. Capability columns: ✓ provided, ∼ partial or implicit, ✗ not provided. The *Reported performance* column quotes each method's headline figure *as reported on its own dataset and label space*; because datasets, class counts, and metrics differ, these figures are **not directly comparable** and are given only to characterise the landscape. "n/r": no single headline figure reported in the source. "Scale" is the approximate number of application/traffic classes evaluated.

Method	Early exit	Per-class precision	Unknown detect.	Drift maint.	Robust perturb.	Reported performance	#Classes
FastFlow [11]	✓	✗	∼	✗	✓	>91%/96% acc. (apps/prov.)	∼7–40
TINIEE [12]	✓	✗	✗	✗	✗	−34.7% traffic, −17.8% FCT	≤15
Serial ETC [22]	✓	✗	✗	✗	✗	n/r (best vs. 3 baselines)	≤20
Two-phase [24]	∼	✗	✗	✗	✗	n/r (acc. at reduced wait)	≤20
Elephant-flow [13]	∼	✗	✗	✗	✗	first-packet detection	binary
EBSNN [1]	✗	✗	✗	✗	✗	n/r	∼20
EC-GCN [3]	✗	✗	✗	✗	∼	+5–20% acc. vs. SOTA	≤15
BPF-DAG [5]	✗	✗	✗	✗	∼	n/r	≤20
AutoML4ETC [10]	✗	✗	✗	✗	✗	n/r	∼40
Perm.-aware [16]	✗	✗	✗	✗	✓	n/r	≤20
SimSeq [17]	✗	✗	✗	✗	✓	F1 ≈0.9x (CESNET-TLS22)	≤20
QUIC-domain [20]	✗	✗	∼	∼	✗	n/r (transfer pretraining)	∼100+
CAPE-Net (this work)	✓	✓	✓	✓	∼	96% acc., 95% macro-F1, precision@ τ 0.99	**180**

roughly an order of magnitude larger than the label spaces of most early-exit and encrypted-traffic classifiers in the literature, and it is precisely at this scale that per-class calibration and class-level drift behaviour become non-trivial.

Each bidirectional flow in this dataset is represented by a *Per-Packet-Information* (PPI) sequence, together with a set of flow-level fields (a flow identifier, `APP`, `CATEGORY`, `TIME_FIRST`, `PPI_LEN`, the PPI sequence itself, `BYTES`, `PACKETS`, and `DURATION`). In our case, the PPI sequence is the only input to our models, given that this would be the case when classifying encrypted traffic in real time. It describes up to the first 30 packets of a flow, and each packet is encoded by four channels: inter-packet time (`IPT`), direction (`DIR`, +1 for client→server and −1 for server→client), transport payload size (`SIZE`), and the TCP `PSH` flag (0/1). These four channels are metadata only: no packet payload bytes are used at any stage. This is a deliberate design choice, as the progressive deployment of TLS 1.3, Encrypted Client Hello, and QUIC is steadily eroding the availability of payload-based features and making packet-sequence metadata the more durable signal.

Formally, each flow is mapped to a raw input tensor of shape $(N, 30, 4)$, where N is the number of flows in a split. Flows with fewer than 30 observed packets are zero-padded to the fixed length `MAX_PPI_LEN` = 30. This fixed-length representation is what allows the recurrent backbone to be read out at any of the early-exit checkpoints described in Section 4. In a real case, if the flow has only few packets, it might not be accurately classified with this method.

The four PPI channels are normalized independently using a per-channel z-score scaler. Crucially, the per-channel mean and standard deviation are estimated over all (flow, timestep) pairs in the *training split only*, and the resulting scaler is then applied unchanged to the validation, test, and drift splits; this prevents any information from the evaluation periods leaking into the normalisation statistics. A single *shared* scaler is used for both the GRU and

Table 2: Temporally ordered flow splits used in all experiments. Each split is drawn from a distinct, non-overlapping period of 2022 and covers all 180 application classes. The training split is capped at 30M flows; the calibration, in-distribution test, and drift test splits are single weeks.

Split	Period (2022)	Flows	Unique apps
Train	W1 – W38	30 000 000	180
Validation / calibration	W39	4 014 507	180
In-distribution test (Setting 1)	W40	6 000 000	180
Drift test (Setting 2)	W50	—[†]	180

[†] The W50 drift week is evaluated per class rather than as a single pooled split; its per-class flow counts vary widely because many applications are only marginally present in any single week ten weeks past calibration. We therefore report W50 results only *within* the week (before/after adaptation) and never pool its macro-F1 against the W40 figures (see Section 5).

LSTM backbones so that the two architectures are compared on identical inputs. The 180 application strings are mapped to integer class identifiers by a label encoder fitted on the training split.

3.2 Stratified Sampling and Temporal Splits

Because the full corpus spans a year and contains far more flows than is practical to train on directly, we sample it into four temporally ordered splits chosen so that each evaluation setting answers a distinct question, as shown in Table 2.

The training split is drawn from approximately the first nine months of 2022 (January through roughly week 36) and is capped at 30 000 000 flows with random undersampling. The sampling is stratified so that all 180 application classes are represented; the nine-month horizon is long enough to expose the model to seasonal and operational variation in the backbone traffic without collapsing the distinction between the training period and the later evaluation weeks.

The validation split is the single week `2022-W39`, containing 4 014 507 flows over all 180 classes. This week plays a dual role: it is the calibration set on which the per-class conformal thresholds τ_c are certified (Contribution 1), and it provides the reference baseline against which later temporal drift is measured (Contribution 3). Placing calibration on a held-out week immediately *after* the training horizon, rather than on a random subset of training flows, ensures that the precision guarantee is certified under a mild but genuine temporal shift.

The in-distribution test split is the week `2022-W40`, one week after validation, containing 6 000 000 flows. Being only a week past calibration, W40 is still essentially in-distribution; it is used to evaluate Contributions 1 and 2 on a healthy, non-drifting model.

The drift test split is the week `2022-W50`, approximately ten weeks past validation. This horizon is long enough for natural temporal drift to accumulate, and the split is used exclusively to evaluate the drift-detection and adaptation loop (Contribution 3). The two-setting separation—a near-term in-distribution week and a far-term drift week—lets us report calibration and OOD behaviour on a healthy model without conflating it with the distinct question of long-horizon maintainability.

3.3 In-Distribution / Out-of-Distribution Class Partition

To evaluate unknown-application detection (Contribution 2), the 180 application classes are partitioned into three disjoint groups: 150 *in-distribution* (ID) classes that the classifier is trained to recognise, 20 *OOD-train* classes used only to fine-tune the energy-based detector and never presented as in-distribution labels, and 10 *OOD-eval* classes held out entirely and used only to measure out-of-distribution (OOD) detection performance. This three-way split ensures that the classes on which the OOD detector is tuned are disjoint from those on which it is reported, so the AUROC figures in Section 5 reflect genuine generalisation to applications unseen during both classifier and detector training.

Table 3 lists the exact ID and OOD flow pools used for threshold calibration (on the validation week W39) and for reporting (on the in-distribution test week W40). The headline in-distribution figures reported in Section 5 are computed over the 5 714 882 ID flows of W40; the 212 776 OOD-evaluation flows enter only into the unknown-rate and AUROC measurements. Note that the ID and OOD-eval pools do not sum to the full 6 000 000 W40 split, because lows belonging to the 20 OOD-train classes are excluded from the W40 evaluation pools by construction.

Table 3: In-distribution (ID) and out-of-distribution (OOD) evaluation pools for the energy-based detector. The ID/OOD-eval pools are used for OOD reporting; threshold θ is calibrated on the W39 pools and reported on the W40 pools.

Role	Week	ID flows	OOD-eval flows
Threshold calibration	W39	3 829 784	138 399
Reporting	W40	5 714 882	212 776

Class partition: 150 in-distribution / 20 OOD-train / 10 OOD-eval, drawn from the 180 application classes. OOD-train classes are used only to fine-tune the detector and never appear as in-distribution labels; OOD-eval classes are used only for measurement.

4 The CAPE-Net Method

CAPE-Net is built around a single recurrent early-exit backbone that we augment with three add-on mechanisms. The organising idea is that the early-exit structure is not merely an efficiency mechanism but a multi-purpose inference instrument: the same per-exit heads that let the model commit early also carry a calibrated precision guarantee, an abstention score for unknown traffic, and a temporal drift signal. Figure 1 gives an overview of the resulting system. We first describe the base model and its adaptive early-exit mechanism (§ 4.1), then the three mechanisms in turn—per-class conformal calibration (§ 4.2), energy-based OOD detection (§ 4.3), and drift-aware bias-only retraining (§ 4.4)—and close with the two-setting evaluation protocol that ties them together (§ 4.5).

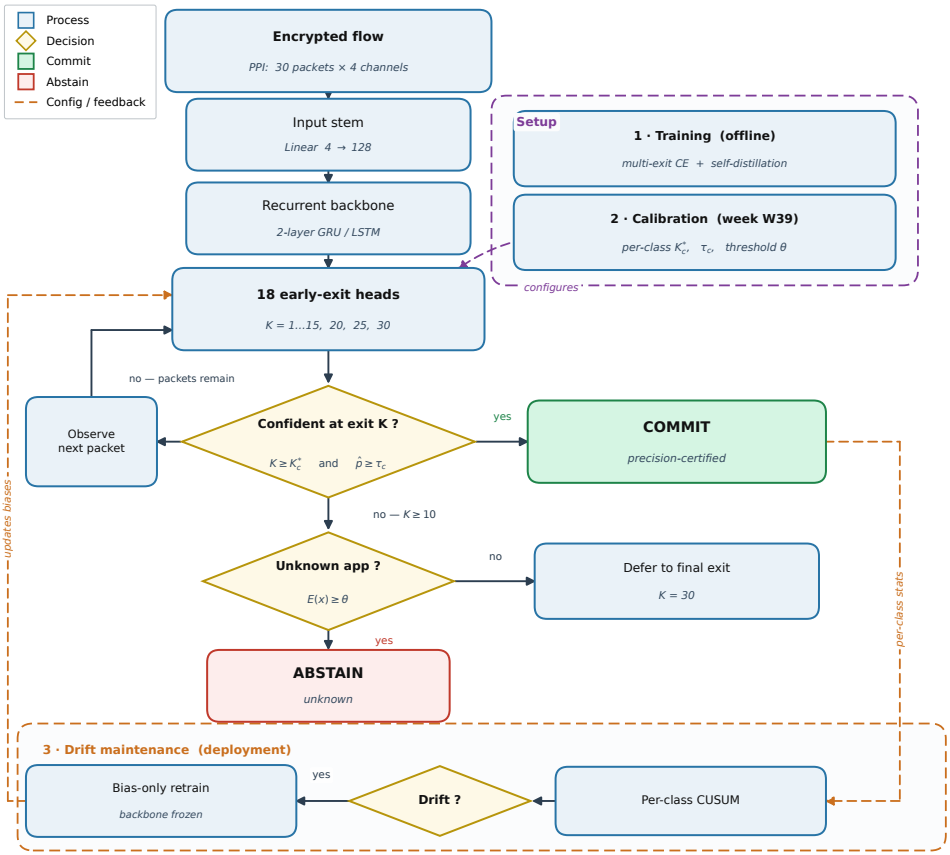


Fig. 1: Overview of the CAPE-Net method. A flow is processed packet by packet through the shared recurrent backbone and its eighteen early-exit heads. At each checkpoint the commit rule either certifies an early decision, abstains as unknown via the energy gate, or defers to a deeper exit. Offline multi-exit training and per-class calibration on the validation week configure the heads; in deployment, per-class CUSUM monitors on the committed flows trigger a bias-only retrain that leaves the backbone frozen.

4.1 Base Model and Adaptive Early Exit

The model maps a PPI sequence of shape $(30, 4)$ to a sequence of per-packet embeddings through a shared input stem and then processes it with a recurrent backbone read out at multiple checkpoints. The input stem is a single linear projection Linear$(4 \to 128)$ followed by a per-feature normalisation, producing a 128-dimensional embedding for each packet. These embeddings are consumed by a two-layer recurrent backbone with hidden size 256. We compare two backbone variants head-to-head under otherwise identical conditions: a two-layer GRU and a two-layer LSTM, both with hidden size 256 (the input-to-hidden weight matrices accordingly contain 3×256 and 4×256 gate blocks respectively). Keeping the stem, hidden

width, head structure, scaler, and training horizon fixed across the two backbones isolates the effect of the recurrent cell itself.

The classifier is the set of *early-exit heads* attached to the recurrent trunk. We use the *extended-K* checkpoint :

$$K = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 30\}, \tag{1}$$

i.e. 18 exit heads in total. At checkpoint K, the recurrent hidden state after the first K packets is read out by head head_K, which is a LayerNorm(256) followed by Linear($256 \rightarrow 180$) and produces 180-class logits. The exit heads are the only component that grows with K; the recurrent trunk is shared across all checkpoints, so the additional cost of supporting many exit points is a set of small linear heads rather than a deeper network.

An earlier configuration of the backbone used 14 exit heads beginning at $K = 5$, so the earliest possible commitment was after five packets. The *extended-K* set in Eq. (1) adds four very early checkpoints ($K \in \{1, 2, 3, 4\}$), giving the model the option to commit on genuinely easy flows after as few as one to four packets. Everything else—dataset, training horizon, and the three contributions—is held fixed, so this change isolates the effect of finer early-exit granularity. As Section 5 shows, the practical consequence is that the realised per-class commit point can drop as low as three packets at no in-distribution accuracy cost, giving operators finer latency/accuracy control rather than a coarse five-packet floor.

All 18 heads are trained jointly. Each head receives a standard cross-entropy loss against the true label at its checkpoint, so that every exit is individually a competent 180-class classifier on the prefix it observes. In addition, the shallow heads are trained with self-distillation from the deepest head ($K = 30$): the $K = 30$ head, which sees the full available prefix, acts as a teacher whose softened predictions guide the earlier heads. This encourages the early checkpoints to anticipate the eventual deep-exit decision rather than merely fitting their own truncated view, and it is what makes early commitment reliable enough to be worth certifying.

Our base model does not commit at a fixed checkpoint. Instead, for each class c a calibration pass on the validation week (W39) selects an *earliest reliable commit point* K_c^*: the smallest checkpoint in K at which the class's head is accurate enough, under an F1-ratio criterion measured across checkpoints. At inference, a flow is processed packet by packet, and at each checkpoint K the model commits to the current argmax class $\hat{c}$ if and only if:

$$K \geq K_{\hat{c}}^* \quad \text{and} \quad \hat{p} \geq \tau_{\hat{c}}, \tag{2}$$

where $\hat{p}$ is the head's confidence in $\hat{c}$ and $\tau_{\hat{c}}$ is the per-class confidence threshold defined in §4.2. A flow that satisfies neither condition waits for more packets; any flow still uncommitted after the last checkpoint falls back to the $K = 30$ head. We refer to the checkpoint at which a flow actually commits as its realised exit K_{used}. Under the extended-K set, the per-class commit points K_c^* range from 3 to 20, with mean K^* of 7.59 for the GRU backbone and 7.30 for the LSTM backbone, so that several classes are certified to commit on just three packets while the hardest classes defer toward the middle of the sequence.

4.2 Per-Class Conformal Precision Calibration

The commit rule in Eq. (2) is only as trustworthy as the thresholds τ_c it relies on. A naive choice—a single global confidence threshold, or per-class thresholds tuned to maximise validation accuracy—gives no guarantee on the precision actually realised at deployment, particularly in the long tail of a 180-class label space. Our first contribution replaces heuristic thresholds with per-class thresholds that come with a finite-sample, distribution-free precision guarantee.

For each class c we certify τ_c using the Learn-Then-Test (LTT) procedure, framing threshold selection as a family of hypothesis tests on the calibration week. Concretely, for a grid of candidate thresholds we test the null hypothesis that committing class c at that threshold violates the target precision, and we retain the threshold only if the test rejects at the chosen risk level. Because the same procedure is applied across all 180 classes simultaneously, we control the family-wise error with the Holm [34] correction; we compare Holm against the more conservative Bonferroni [35] correction and adopt Holm, which certifies the same set of classes at higher coverage. The target configuration is a per-class precision floor of 0.95 with $\alpha = 0.05$, $\delta = 0.05$, and a minimum support of 50 calibration samples per class.

A small number of classes have as well few calibration samples for LTT to issue a certificate at the required confidence. For these classes, we fall back to the original empirical threshold rather than forcing an uncertified guarantee. As reported in Section 5, 176 of 180 classes are certified under Holm on both backbones, with only four classes reverting to the empirical threshold.

It is worth being explicit about the semantics of the guarantee. The certificate is a *per-class* statement about the precision of *committed* flows: among flows that the model commits to class c under Eq. (2), the fraction that truly belong to c is at least 0.95 with finite-sample validity. It is not a marginal accuracy figure, and it is purchased at the cost of *coverage*: flows whose confidence never clears τ_c are deferred or fall back to the deep exit rather than being committed. Reporting realised precision and coverage together—rather than a single headline accuracy—is therefore central to how we evaluate this contribution.

4.3 Energy-Based Out-of-Distribution Detection

A classifier trained on a fixed set of applications will, by construction, assign every flow to one of its known classes, including flows from applications it has never seen. For a network operator this is a liability: a confident label on genuinely unknown network traffic is worse than an explicit abstention (e.g. giving this traffic high priority when it is not really needed). Our second contribution equips the model to emit an *unknown* verdict.

We read a free-energy score from the logits,

$$E(x) \;=\; -\,\mathrm{logsumexp}\big(\mathrm{logits}(x)\big), \tag{3}$$

at a fixed readout checkpoint $K_{\mathrm{OOD_MIN}} = 10$. The energy detector is fine-tuned with an energy-margin objective on the 20 held-out OOD-train classes (§3.3), pushing in-distribution flows to low energy and out-of-distribution flows to high energy. A decision threshold θ is then set so that the in-distribution false-positive rate on the validation week is at most 0.05, i.e. at most 5% of known-application flows are mistakenly flagged unknown.

The interaction between OOD gating and adaptive early exit requires care. Because the typical flow commits around $K_{\text{used}} \approx 7$–$8$, which is *below* the energy readout point $K_{\text{OOD_MIN}} = 10$, a naive gate that applied the energy test first would only ever see the hardest, slowest flows— exactly the ones most likely to look anomalous—and would abstain on many legitimate in-distribution flows. We therefore invert the natural order and use a *confidence-wins* gate: a flow that confidently classifies under Eq. (2) (i.e. $K \geq K_c^*$ and confidence $\geq \tau_c$) is committed immediately, and only flows that fail to classify confidently are passed to the energy gate, where they may be abstained as unknown. This ordering is essential to the detector's usefulness and is, to the best of our knowledge, a novel adaptation of energy-based OOD detection to the streaming early-exit setting.

4.4 Drift-Aware Bias-Only Retraining

A classifier deployed on a live ISP link ages: application behaviour, versions, and traffic mixes shift, and per-class accuracy degrades over weeks and months. Full retraining at this scale is expensive and disruptive. Our third contribution is a lightweight, always-safe maintenance loop that detects per-class drift and repairs only the affected classes.

On the deployed model, we run two cumulative-sum (CUSUM) monitors per class, updated daily. The first tracks *upward* drift in the mean realised exit K_{used} relative to the calibrated K_c^*, with control parameter $h = 5.0$; an increase signals that the class is becoming harder to decide early. The second tracks *downward* drift in realised precision relative to τ_c, with $h = 3.0$; a decrease signals that the class's precision guarantee is eroding. Either monitor crossing its threshold flags the class for adaptation. A class is considered to have genuinely *degraded* if its F1-score on the drift week drops by more than five percentage points relative to its W39 baseline; this definition is the ground truth against which detection precision and recall are measured in Section 5.

When a class is flagged, we perform a deliberately conservative repair. All model weights are frozen except the final-linear *bias* terms of the exit heads, and the gradients are masked so that only the biases of the *flagged* classes are updated. Because no other parameter can change, the update cannot alter the decision function for any non-flagged class—the maintenance loop can never damage a class it was not asked to repair. The retraining uses the chronologically first 80% of the flagged classes' flows in the drift week, with the remainder reserved for measurement, so that no future information leaks into the update. An energy-hinge regulariser on the OOD-train flows is added during retraining to preserve the OOD behaviour established in § 4.3. As § 5 reports, this update completes in roughly 16–19 seconds in the used hardware, making it lightweight enough to run as routine maintenance.

4.5 Evaluation Protocol: Two Settings

Because the contributions answer different questions, we evaluate them under two distinct temporal settings rather than a single pooled test.

- *Setting 1* uses the in-distribution week W40 and ablates the model in stages—the baseline (V1), then adding conformal calibration (V2), then adding OOD detection (V3)—to measure Contributions 1 and 2 on a healthy, non-drifting model.

- *Setting 2* uses the drift week W50, ten weeks past calibration, to measure the detection and recovery behaviour of Contribution 3.

For completeness, we also report a full 1+2+3 configuration (V4) on W40, but we do not treat it as the W40 headline: dropping a drift-calibrated model into a non-drifted week can only match or slightly hurt its W40 numbers, and calibrating the drift loop on W40 itself would constitute leakage. The W40 headline operating point is therefore V3.

We report, over committed in-distribution flows (abstentions excluded), accuracy and macro-F1 (averaged over the ID classes present in the split); the realised precision of committed flows against the 0.95 target; the exit-packet count (mean/median/p_{90} of K_{used}); the energy-score AUROC at $K_{\text{OOD_MIN}}$ on the held-out OOD-eval classes; and the unknown-rate, the fraction of all test flows abstained as unknown. The macro-F1 of the drift week is only ever compared *within* W50 (before versus after adaptation) and never against the W40 figures, for the reason noted in Section 3.

4.6 Implementation and Experimental Setup

All models are implemented in PyTorch (v2.9.1) and trained and evaluated on a single high-performance server equipped with two AMD EPYC 7F72 CPUs (24 cores / 48 threads each at 3.2 GHz), 1 TB of DDR4 memory, and NVIDIA A100-SXM4 GPU (40 GB HBM2). This configuration provides the CPU parallelism and large in-memory capacity needed to stream the multi-million-flow weekly splits, together with the GPU acceleration required for large-scale multi-exit training and high-throughput inference.

5 Results

We organise the results around the two evaluation settings of Section 4.5. Setting 1 (the in-distribution week W40) reports the staged ablation that isolates Contributions 1 and 2 on a healthy, non-drifting model (§ 5.1–§ 5.3). Setting 2 (the drift week W50, ten weeks past calibration) reports the detection and recovery behaviour of Contribution 3 (§ 5.4). We then summarise early-exit efficiency (§ 5.5) and place the extended-K configuration in context against the standard checkpoint set (§ 5.6). All in-distribution figures are computed over the 5 714 882 committed in-distribution flows of W40; unknown abstentions are excluded from accuracy and macro-F1, and macro-F1 is averaged over the in-distribution classes present in the split.

5.1 Setting 1: In-Distribution Variant Ablation

Table 4 reports the staged ablation on W40 for both backbones, adding one contribution at a time: the baseline model (V1), then per-class conformal calibration (V2), then energy-based OOD detection (V3), and finally the full configuration including the drift-adaptation loop (V4). Figure 2 visualises the macro-F1 column.

The headline observation is that the contributions stack monotonically on both backbones: accuracy and macro-F1 increase at every step from V1 to V3. Conformal calibration (C1) is the single largest lever, lifting macro-F1 from 0.9251 to 0.9463 on the GRU (+2.1 pp)

and from 0.9314 to 0.9576 on the LSTM (+2.6 pp), while raising the realised precision of committed flows to roughly 0.99—the 95% guarantee met with comfortable margin. This gain is purchased with latency: the mean exit point rises (e.g. GRU 7.73 → 9.47 packets), because flows whose confidence does not clear the per-class threshold defer to later checkpoints. Adding OOD detection (C2) then both improves accuracy and *reduces* the mean exit point relative to V2 (GRU 9.47 → 8.35), because uncertain flows are abstained as "unknown" rather than made to wait; it also introduces an abstention option, flagging roughly 5.5% of all flows (including held-out OOD traffic) as unknown. The best in-distribution operating point is therefore V3 on both backbones. As anticipated in Section 4.5, V4 sits about 1–1.5 pp below V3 on W40: this is the expected, harmless consequence of running a drift-calibrated model on a week that has not drifted, and is why V3, not V4, is the W40 headline.

Figure 3 makes the latency/accuracy interaction explicit by tracing each backbone through (mean K, accuracy) space: the V1→V2 step moves up and to the right (more accurate, later commit), while V2→V3 moves back to the left (OOD abstention recovers the latency that calibration spent).

Table 4: Setting 1 (W40, in-distribution) staged ablation over 5 714 882 committed in-distribution flows. Best operating point (V3) in **bold**. *prec@*τ is the realised precision of committed flows against the 0.95 target; *unk* is the fraction of all test flows abstained as unknown.

BB	Variant	acc	macF1	meanK	p90K	prec@τ	OOD AUROC
GRU	V1 baseline	0.9354	0.9251	7.73	11	0.9510	—
	V2 +Conformal	0.9529	0.9463	9.47	20	0.9906	—
	V3 +OOD	**0.9599**	**0.9502**	**8.35**	**12**	**0.9840**	**0.8728**
	V4 full	0.9451	0.9291	8.23	12	0.9536	0.8729
LSTM	V1 baseline	0.9408	0.9314	7.25	10	0.9528	—
	V2 +Conformal	0.9597	0.9576	8.62	13	0.9911	—
	V3 +OOD	**0.9640**	**0.9556**	**7.67**	**11**	**0.9837**	**0.6874**
	V4 full	0.9538	0.9409	7.61	10	0.9670	0.6871

5.2 Conformal Precision Compliance

Table 5 details the per-class conformal certification on W40. Under the Holm correction, 176 of 180 classes receive a finite-sample 0.95-precision certificate on both backbones, with only four low-support classes reverting to their empirical threshold. On the calibration week (W39) the guarantee holds for 100% of classes by construction; one week out, on W40, it generalises to 98.3% of classes for the GRU and 97.8% for the LSTM, with mean realised precision of $\approx$ 0.99 on both. This generalisation is purchased at a mean coverage (commit rate) of about 0.76: roughly a quarter of flows are deferred or abstained rather than committed, which is the cost of a per-class precision floor. The mechanism behind this precision gain is visible in Figure 4, which contrasts the per-class thresholds chosen by the two schemes. The baseline model's empirical thresholds concentrate near zero, so classes commit on weak

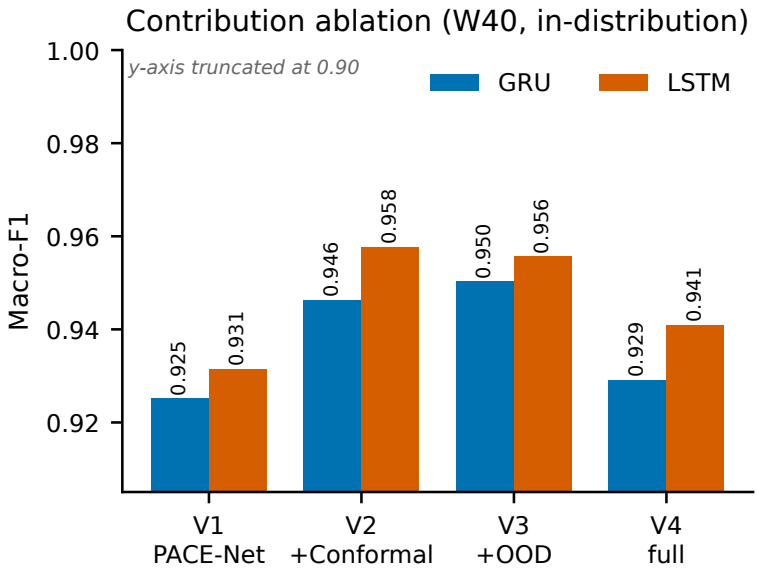


Fig. 2: Macro-F1 across the staged ablation (V1–V4) for the GRU and LSTM backbones on W40. Contributions stack monotonically through V3; the V4 dip is the expected effect of a drift-calibrated model on a non-drifted week.

Table 5: Contribution 1: per-class conformal compliance on W40 (Holm correction, $\alpha = 0.05$, $\delta = 0.05$, min. support 50).

	GRU	LSTM
Classes	180	180
Certified (Holm)	176	176
Fell back to empirical τ	4	4
Frac. classes ≥ 0.95 on val (W39)	1.000	1.000
Frac. classes ≥ 0.95 on test (W40)	0.9833	0.9778
Mean precision (W40)	0.9906	0.9903
Mean coverage (commit rate)	0.7560	0.7723

evidence; the LTT procedure instead pushes the bulk of the thresholds toward 0.85–0.90, requiring high confidence before any commitment is certified. The price of this shift is the reduced coverage noted above, but it is what converts an uncontrolled empirical rule into a finite-sample guarantee.

Figure 5 plots realised W40 precision for all 180 classes against the 0.95 target line, showing the small minority of classes that fall below the floor under temporal shift.

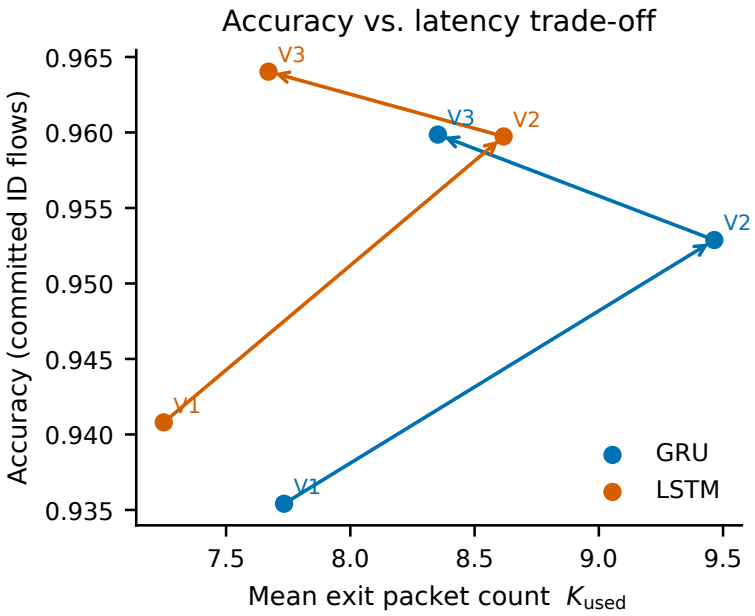


Fig. 3: Accuracy versus mean exit packet K as each contribution is added. Conformal calibration (V1→V2) trades latency for accuracy; OOD abstention (V2→V3) recovers latency.

5.3 Out-of-Distribution Detection

Table 6 reports the energy detector at the operating threshold that fixes the in-distribution false-positive rate at 0.05, and Figure 6 shows the corresponding ROC curve for the GRU backbone. The curve rises steeply away from the chance diagonal, reaching an AUROC of 0.858; at the fixed 5% false-positive operating point ($\theta_{\text{OOD}} = -17.96$) the detector recovers 42.8% of held-out unknown flows. The GRU is the stronger OOD backbone in this extended-K configuration, clearly outperforming the LSTM (AUROC 0.687). We note explicitly that this ordering is the *reverse* of the standard nine-month configuration (§5.6), in which the LSTM led on OOD; we therefore report OOD performance per backbone and do not claim a backbone-independent OOD advantage.

5.4 Setting 2: Temporal Drift Detection and Recovery

We now turn to the second evaluation setting: the behaviour of the drift-detection and bias-only recovery loop (Contribution 3) over the ten held-out weeks W41–W50, spanning roughly two to eleven weeks past the W39 calibration. The model is run unmodified each week; per-class CUSUM monitors flag drifting classes; each flagged class receives a bias-only, energy-hinge-preserving retrain on the chronologically first 80% of that week's flows; and macro-F1 is measured before and after the update. A class is counted as genuinely *degraded* if its weekly F1 has fallen by more than five percentage points relative to its W39 baseline.

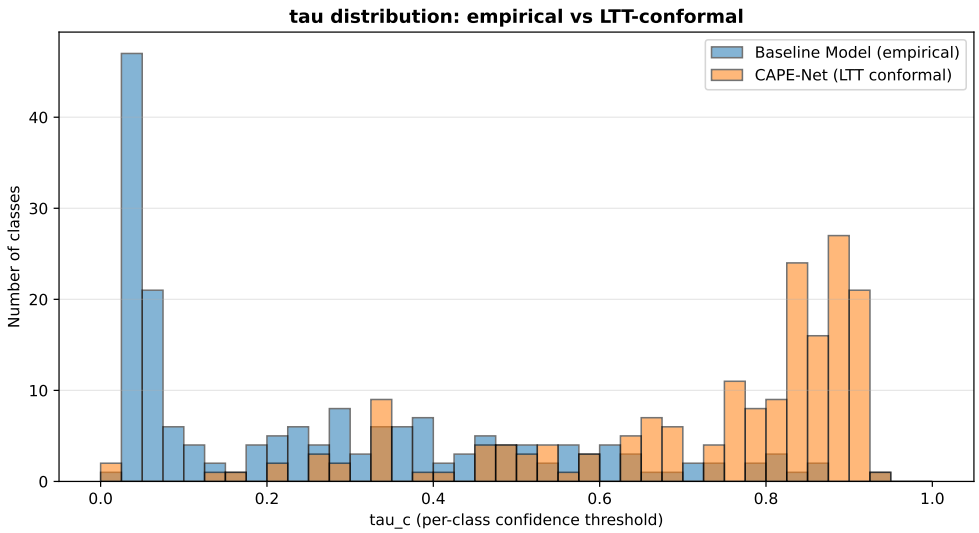


Fig. 4: Distribution of per-class confidence thresholds τ_c over the 180 classes: the baseline model's empirical thresholds versus CAPE-Net's LTT-conformal thresholds. Empirical thresholds cluster near zero, committing on weak confidence, whereas LTT shifts the mass toward 0.85–0.90, demanding substantially higher confidence before a class commits—the mechanism behind the rise in realised precision to ≈ 0.99.

Table 6: Contribution 2: energy-based OOD detector at the FPR ≤ 0.05 operating point. Held-out pools: 5 714 882 ID and 212 776 OOD-eval flows; energy read at $K_{\text{OOD_MIN}} = 10$.

	GRU	LSTM
Energy threshold θ	−17.955	−17.920
AUROC @ K=10	0.8728	0.6874
FPR @ θ	0.0500	0.0500
TPR @ θ	0.4285	0.2403
Overall unknown-rate	0.0183	0.0227
Mean K_{used}	7.629	7.184

Figure 7 shows the full temporal picture, and Table 7 gives the per-week numbers. Two trends are visible and mutually reinforcing. First, macro-F1 declines monotonically as the test week recedes from calibration, from 0.750 at W41 to 0.619 at W50—clear, sustained concept drift over roughly two and a half months. Second, as drift accumulates, both the number of genuinely degraded classes and the CUSUM detection precision rise together: degraded classes grow from 26 at W41 to 101 at W50, while detection precision climbs from 0.13 to 0.82 over the same span. This is exactly the behaviour a network operator wants from a traffic monitor. Early on, when week-to-week variation is dominated by sampling noise, the monitor raises few alarms and those it raises are uncertain; once real drift sets in, its alarms become

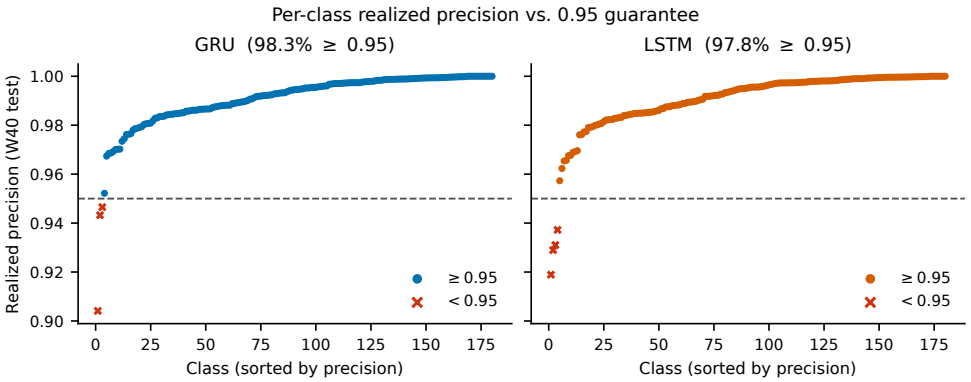
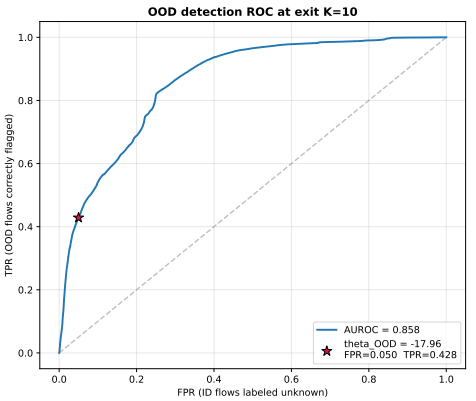
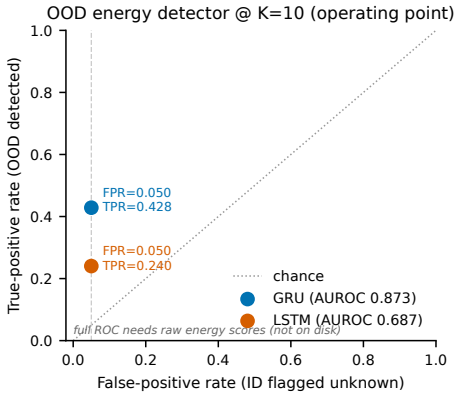


Fig. 5: Per-class realised precision on W40, sorted ascending, for each backbone. The horizontal line marks the 0.95 target; the guarantee holds for ≈ 98% of classes one week past calibration.



(a) Single-backbone detail (GRU): operating point at FPR = 0.05, $\theta_{\text{OOD}} = -17.96$, TPR = 0.428, AUROC = 0.858.



(b) Both backbones: GRU leads in this extended-K configuration (AUROC 0.873 vs 0.687), reversing the standard-run ordering.

Fig. 6: ROC of the energy-based OOD detector, with the energy score read at exit $K = 10$ and the FPR = 0.05 operating point marked.

highly trustworthy. The monitor is therefore conservative by construction—it does not cry wolf during the noisy early weeks—and grows confident only as genuine degradation emerges.

The bias-only retrain recovers a small but consistent margin in every week, between +0.002 and +0.010 macro-F1, at a cost of only 15–25 seconds per update with the used hardware. Because the update modifies only the final-linear bias terms of the flagged classes, with all other parameters frozen, it cannot alter the decision function for any non-flagged class. The recovery is thus best understood not as a one-shot correction for severe drift, but

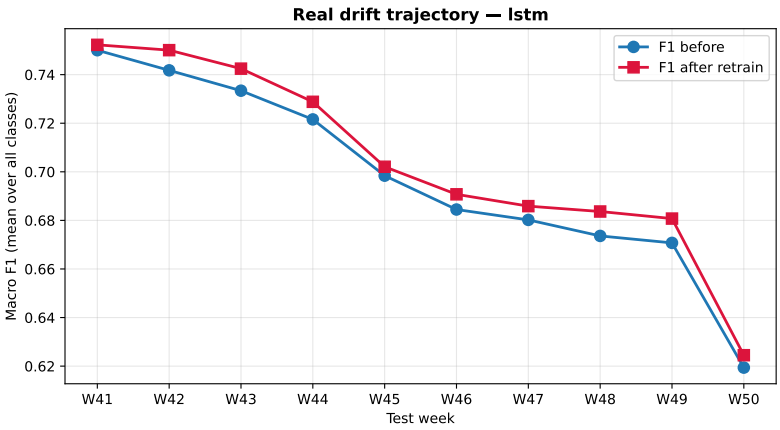


Fig. 7: Concept-drift trajectory over the ten held-out weeks W41–W50 (LSTM backbone), two to eleven weeks past the W39 calibration. Macro-F1 declines monotonically as the test week moves away from calibration; the bias-only drift retrain (upper line) consistently recovers a small margin over the unmodified model (lower line). Per-week macro-F1 is a mean over all 180 classes and is not comparable to the in-distribution W40 figures (Section 3).

Table 7: Setting 2: per-week drift trajectory (LSTM backbone), W41–W50. Macro-F1 before and after the bias-only retrain, number of classes flagged by CUSUM, number actually degraded (> 5 pp F1 drop vs the W39 baseline), and detection precision. Values are within-track means over all 180 classes and are not comparable to the W40 in-distribution F1.

Week	F1 before	F1 after	Flagged	Degraded	Det. prec.
W41	0.750	0.752	24	26	0.125
W42	0.742	0.750	27	30	0.407
W43	0.733	0.742	29	35	0.448
W44	0.722	0.729	28	48	0.536
W45	0.698	0.702	34	63	0.676
W46	0.684	0.691	38	67	0.711
W47	0.680	0.686	34	69	0.676
W48	0.674	0.684	38	69	0.737
W49	0.671	0.681	42	81	0.762
W50	0.619	0.624	44	101	0.818

as a lightweight, always-safe maintenance mechanism that slows degradation between full retrainings. On the most-drifted week (W50, eleven weeks past calibration), the loop flags 44 classes at a detection precision of 0.82 and lifts within-week macro-F1 from 0.619 to 0.624. We stress that the absolute macro-F1 values along this trajectory ($\approx$ 0.62–0.75) are single-week means over all 180 classes, many of which are only sparsely present in any given week; they are therefore *not* comparable to the in-distribution W40 test figures of Section 5.1. The before/after and week-to-week comparisons are meaningful only within this drift track.

5.5 Early-Exit Efficiency

We first characterise the base model policy's efficiency, independent of the conformal and OOD layers. Figure 8 places the adaptive policy on the latency–accuracy plane against fixed-K baselines that commit every flow at the same packet count. The fixed-K curve rises steeply and then saturates: macro-F1 climbs from 0.061 at $K = 1$ to 0.752 at $K = 5$, 0.872 at $K = 7$, and 0.928 at $K = 10$, after which it flattens toward 0.957 at the full 30 packets. The adaptive policy (red star) attains macro-F1 0.919 while observing only 7.29 packets per flow on average—matching the fixed-K baseline near $K = 9$ but committing earlier, and landing within 0.04 F1 of the full 30-packet model at roughly a quarter of the packets. In other words, the adaptive policy sits on the knee of the curve, capturing almost all of the attainable accuracy at a fraction of the latency, with clearly diminishing returns beyond $K \approx 12$–15. (The macro-F1 reported here, 0.919, is for the base adaptive policy in isolation; the conformal and OOD layers of Section 5.1 raise the operating-point accuracy further.)

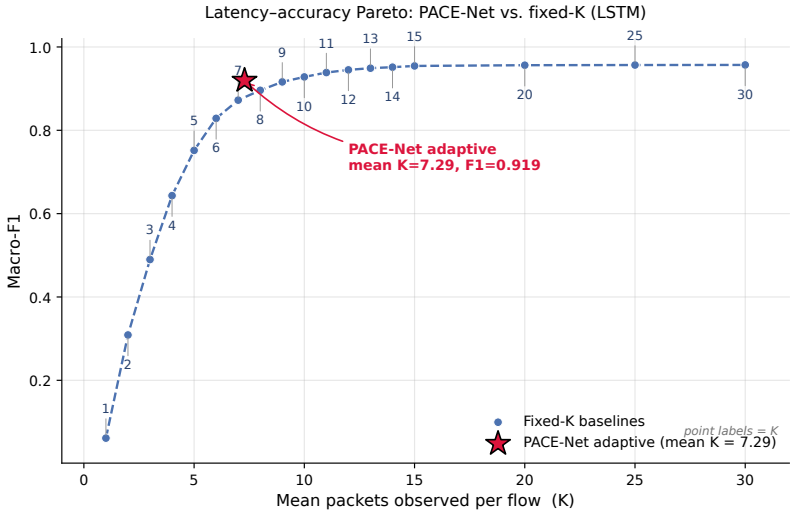


Fig. 8: Latency–accuracy trade-off (LSTM backbone). Dashed curve: fixed-K baselines committing all flows at a common packet count K (point labels give K). Red star: the base model adaptive policy (mean $K = 7.29$, macro-F1 0.919), which sits on the knee of the curve—near the accuracy of the full 30-packet model at about a quarter of the packets.

Figure 9 explains *why* the adaptive policy is efficient: the per-class commit points K_c^* are strongly heterogeneous. Across both backbones, the distribution ranges from 3 to 20 packets, with mean 7.59 (GRU) and 7.30 (LSTM) and a median of 7; roughly 70% of classes commit by $K^* = 8$, while a small tail of hard classes requires up to 20 packets. This heterogeneity is precisely what a fixed-K classifier cannot exploit and what motivates per-class adaptive exit: easy applications are identified from a handful of packets, while the deeper exits are reserved for the minority of classes that genuinely need them. Inference throughput reinforces this efficiency at the implementation level. On a single NVIDIA A100, a full forward pass over

the entire 6M-flow W40 test set (all eighteen exits evaluated) runs at 56 934 flows/s for the GRU backbone and 14 882 flows/s for the LSTM—a roughly 3.8× throughput advantage for the GRU at essentially the same accuracy (cf. Table 4). These figures are an upper bound on the work the adaptive policy performs: because most flows commit well before the deepest checkpoint, adaptive inference evaluates fewer exits per flow than the all-exits pass measured here. Training is comparably inexpensive on both backbones, at 2 639 s (GRU) and 2 739 s (LSTM) for twenty epochs on the same hardware.

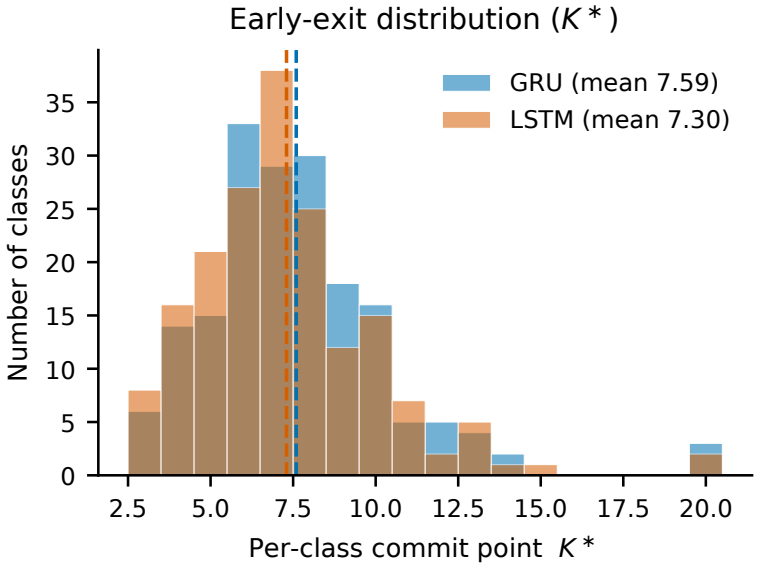


Fig. 9: Distribution of per-class commit points K_c^* over the 180 classes for both backbones. K^* is the smallest checkpoint at which a class reaches 95% of its per-class F1 ceiling. The range is 3–20 packets (mean 7.59 GRU, 7.30 LSTM; median 7), with ≈ 70% of classes committing by $K^* = 8$, motivating adaptive rather than fixed early exit.

Packet efficiency is only one axis of cost; Figure 10 plots the complementary one, measured inference throughput, against accuracy for each variant. Two patterns emerge. On the GRU, the mechanisms trace a genuine Pareto trade-off: throughput falls monotonically from 1.42M flows/s at the baseline (V1) to 1.35M with conformal calibration (V2) and 1.23M with the added energy detector (V3), a total cost of roughly 13%, purchased for a 2.4 pp accuracy gain (0.9354 → 0.9599). On the LSTM, by contrast, throughput is essentially flat across the three variants (1.17M, 1.21M, 1.18M flows/s), so its 2.3 pp accuracy gain (0.9408 → 0.9640) comes at almost no throughput cost. The two backbones are much closer in speed than this per-variant behaviour might suggest: at the V3 operating point the GRU sustains 1.23M flows/s against the LSTM's 1.18M, a difference of about 5%. In practical terms, both process the full six-million-flow test week in a few seconds on a single Nvidia A100 GPU, so the choice between them turns on accuracy and unknown-detection behaviour rather than on speed.

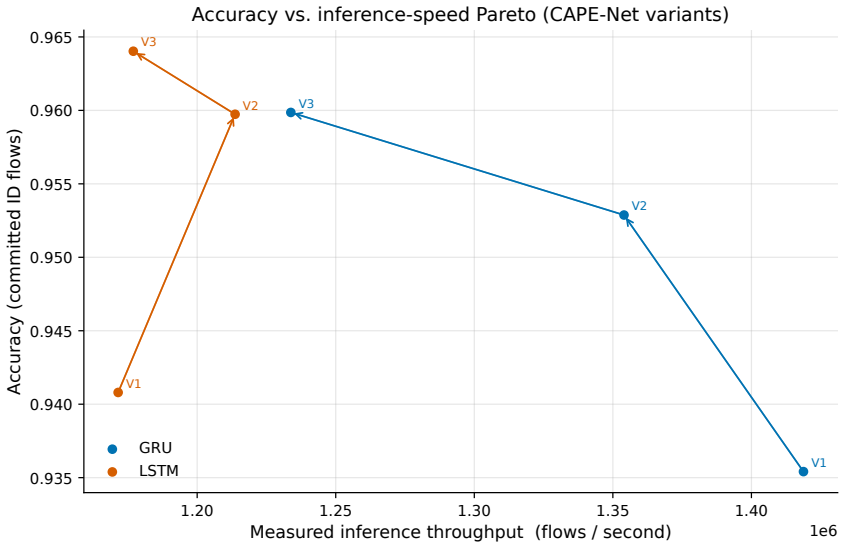


Fig. 10: Accuracy versus measured inference throughput for the three CAPE-Net variants on both backbones (W40, single A100). Arrows trace V1→V2→V3. On the GRU each added mechanism buys accuracy at a throughput cost; on the LSTM throughput is essentially flat across variants, so the accuracy gains are close to free.

5.6 Extended-K versus the Standard Checkpoint Set

Table 8 compares the extended-K configuration used throughout this paper (exits at $K \in \{1, \ldots, 15, 20, 25, 30\}$, 18 heads) against the standard nine-month set (exits from $K = 5$, 14 heads), at the V3 operating point. Adding the four very-early exits ($K = 1$–4) preserves accuracy and macro-F1—and slightly improves the LSTM—while enabling certified commitment as early as three packets, i.e. finer latency/accuracy control at no in-distribution cost. As already noted, OOD AUROC is sensitive to exit granularity and backbone (the ordering reverses between the two configurations), reinforcing that OOD results should be read per backbone rather than as a single claim.

Table 8: Extended-K (this work, 18 exits) versus the standard nine-month set (14 exits) at the V3 operating point on W40.

BB	Config	acc	macF1	OOD AUROC
GRU	Standard (14)	0.9591	0.9527	0.83
	Extended (18)	0.9599	0.9502	0.8728
LSTM	Standard (14)	0.9618	0.9551	0.91
	Extended (18)	0.9640	0.9556	0.6874

5.7 Implementation Details and Hyperparameters

For reproducibility, Table 9 lists the complete configuration used for all reported results. We emphasise that these values were fixed *a priori*: no hyperparameter search was performed for CAPE-Net, and a single configuration was used across both the TLS and QUIC studies, with only the small set of QUIC-specific overrides noted in the table. The TLS and QUIC runs are thus directly comparable in every respect except the protocol-driven differences (the zeroed `PSH` channel, a re-fitted scaler, and the reduced class and OOD-split counts). Our implementation is in PyTorch, with mixed-precision (AMP) training.

6 Generalisation to QUIC: CESNET-QUICEXT-25

The results so far established CAPE-Net on TLS-over-TCP traffic. Because QUIC now carries a large and growing share of encrypted traffic and behaves quite differently at the packet level—it runs over UDP, opens with a large padded Initial packet, and exposes no TCP `PSH` flag—it is a demanding second test of whether the same classification architecture and the same two method components (conformal precision calibration and energy-based OOD detection) carry over to a distinct encrypted protocol. We therefore retrain CAPE-Net from scratch on CESNET-QUICEXT-25 and evaluate it under a strict temporal split. The method is unchanged from Section 4; here we describe only what is QUIC-specific and report the QUIC numbers.

6.1 Dataset and Temporal Split

CESNET-QUICEXT-25 is a public, ISP-scale collection of QUIC (UDP) encrypted-traffic flows spanning a full year (June 2024–May 2025), organised into 53 ISO weeks; this section uses weeks WQ1–WQ42. As with the TLS corpus, each bidirectional flow is again represented by its Per-Packet-Information (PPI) sequence of up to 30 packets together with flow metadata. The QUIC PPI uses three per-packet channels—inter-packet time (`IPT`), direction (`DIR`), and payload size (`SIZE`)—since the TCP `PSH` flag does not exist in QUIC. To keep the input format identical to the TLS model, the fourth channel is set to zero, giving the same $(N, 30, 4)$ input tensor with zero-padding for sequences shorter than 30 packets. A fresh per-channel standardiser is fitted on the QUIC training flows only; the TLS scaler is not reused, because QUIC feature statistics differ substantially.

QUIC flows carry no native application labels. We therefore transfer labels from the CESNET-TLS-Year22 application taxonomy through the TLS Server Name Indication (SNI): an SNI-to-application lookup table built from TLS-Year22 (1 914 453 unique exact SNI keys, with a registered-domain fallback) is applied to each QUIC flow's `QUIC_SNI` in three tiers—exact SNI match, registered-domain match, or unmatched (treated as out-of-distribution). Supervised experiments use the exact and domain tiers, restricted to applications present in the taxonomy. Per-week label coverage is approximately 70–86%, highest early in the period and declining over the year as newer applications appear that the 2022 SNI table does not cover; every mapped application falls within the known taxonomy.

The split is strictly temporal and chronologically ordered, so the test week is genuinely future data the model never observed (Table 10). Training draws on weeks WQ1–WQ40, validation (used for K^* and τ calibration) on WQ41, and testing on WQ42, roughly two weeks

Table 9: Complete hyperparameter configuration for CAPE-Net. Values are shared between the TLS (CESNET-TLS-Year22) and QUIC (CESNET-QUICEXT-25) runs, unless a QUIC override is listed. No hyperparameter search was performed; this single configuration is used throughout.

Component	Hyperparameter	Value
Backbone	Input channels	4 ([IPT, DIR, SIZE, PSH])
	Max PPI length	30 packets (zero-padded)
	Input projection d_{model}	128 (Linear $\rightarrow$ LN $\rightarrow$ GELU)
	Recurrent cell	GRU / LSTM, hidden 256, 2 layers
	Dropout	0.10
	Exit checkpoints	$\{1{-}15, 20, 25, 30\}$ (18 heads)
	Per-exit head	LN(256) $\rightarrow$Drop$\rightarrow$Linear$(256, C)$
Training	Optimiser	AdamW, weight decay 10^{-4}
	Learning rate	3×10^{-3}, OneCycleLR
	Epochs	20
	Batch size (train / infer)	16,384 / 65,536
	Precision	AMP (autocast + GradScaler)
	Label smoothing	0.10
	Exit weights	linear over 18 exits, mean 1
	Self-distillation weight	0.20 (KL from K=30)
	Distillation temperature	2.0
Early-exit calibration	K_c^* criterion	smallest K: $F1_K(c) \geq 0.95 \max_K F1_K(c)$
	K_c^* fallback	30 (if never reached)
	τ_c target precision	0.95
	τ_c min. support	20 val. predictions/class
Conformal (LTT)	Target precision $1 - \alpha$	0.95 ($\alpha = 0.05$)
	Risk level δ	0.05
	Correction	Holm (all class–exit pairs)
	Min. support	50 (else empirical τ)
OOD	Score	$E(x) = -\text{logsumexp}(\text{logits})$
	Held-out classes	20/10 (QUIC: 8/4)
	Finetune	5 epochs, LR 10^{-4}
	Energy margins $m_{\text{in}}/m_{\text{out}}$	-25.0 / -7.0
	Energy loss weight λ	0.1
	$K_{\text{OOD_MIN}}$	10
Drift (Contrib. 3)	CUSUM mean-K	$h = 5.0$, slack $k = 1.0$
	CUSUM precision	$h = 3.0$, slack $k = 0.02$
	Precision baseline	0.95
	Min. committed/day	30
	Retrain	bias-only, LR 5×10^{-4}, 3 epochs
	Retrain batch / split	4096 / first 80% chronological
	Energy-hinge λ	0.05

after the training period ends—a near-future, in-distribution-but-unseen evaluation. We retain the 47 application classes with at least 5 000 in-encoder flows in the training weeks and sample

Table 10: Strict temporal split of CESNET-QUICEXT-25. Train, validation, and test are disjoint, chronologically ordered weeks; the test week WQ42 is unseen future traffic. 47 application classes.

Split	Week(s)	Flows	Share
Train	WQ1–WQ40	21 002 955	80.9%
Validation	WQ41	2 850 487	11.0%
Test	WQ42	2 119 984	8.2%
Total		25 973 426	100%

flows stratified by application (per-class floor 5k, cap 1.0M) so that a few dominant providers do not dominate training. We note that, because the test set is a single week, it constitutes about 8% of flows rather than a nominal 20%: a single week simply cannot supply a fifth of the data against 40 training weeks.

6.2 QUIC-Specific Model Notes

We use the LSTM backbone (two layers, hidden size 256, input projection to 128; 1.16M parameters), trained from scratch on QUIC. The adaptive early-exit policy uses the same 18 checkpoints $K \in \{1, \ldots, 15, 20, 25, 30\}$, with per-class commit point K^* and confidence threshold τ calibrated on the validation week. Per-class τ is certified by Learn-Then-Test with Holm correction at a target precision of 0.95 ($\alpha = \delta = 0.05$). The energy detector sets its threshold θ at FPR ≤ 0.05 on the in-distribution validation flows, with held-out classes acting as out-of-distribution traffic (35 ID / 8 OOD-train / 4 OOD-eval).

6.3 Results on the Held-Out Future Week

Variant ablation

Table 11 reports the staged ablation on the held-out week WQ42, over committed in-distribution flows. The two method components behave as they do on TLS. Conformal calibration lifts macro-F1 from 0.8344 to 0.8450 and raises the realised precision of committed flows from 0.952 to 0.976—the finite-sample precision guarantee realised on unseen data—at a small latency cost (mean K 15.0 $\rightarrow$ 15.4). Adding the energy detector raises accuracy to 0.9384 and introduces an abstention option, flagging 9.5% of all test flows as unknown, while slightly lowering the mean exit point as uncertain flows are abstained rather than made to wait. Macro-F1 dips marginally at this step because abstention removes some hard-but-classifiable flows from the committed set.

Conformal compliance

On WQ42 the per-class realised precision has mean 0.947, and 35 of the 47 classes (74.5%) meet the 0.95 target one-to-two weeks out, at a mean coverage (commit rate) of 0.656. Compliance is therefore somewhat lower than on TLS, which is consistent with the harder single-future-week test and QUIC's longer flows; the roughly one quarter of classes that miss the floor are deferred or abstained rather than committed below target.

Table 11: QUIC staged ablation on the held-out future week WQ42 (LSTM backbone). Metrics over committed in-distribution flows; macro-F1 and precision@τ average over ID classes present in the test week.

Variant	acc	macF1	meanK	prec@τ	AUROC	unk
1. Baseline model	0.9254	0.8344	15.0	0.9519	—	0%
2. +Conformal	0.9293	0.8450	15.4	0.9761	—	0%
3. +OOD	0.9384	0.8248	14.9	0.9687	0.7555	9.5%
4. Full (1+2)	0.9384	0.8248	14.9	0.9687	0.7555	9.5%

OOD detection

The energy detector reaches an AUROC of 0.7555 at the $K = 10$ readout, with an operating point of FPR = 0.049 / TPR = 0.379 ($\theta = -19.78$) and a 9.0% in-distribution unknown-rate. This AUROC should be read with a sampling caveat: the four held-out OOD-eval classes are rare in any single week, so only about 1 400 OOD-eval flows are present in WQ42, making this estimate noisier than the in-distribution metrics.

Early-exit efficiency

Figure 11 plots fixed-K accuracy and macro-F1 against the adaptive policy on WQ42. The adaptive policy reaches accuracy 0.925 at a mean of 15.0 packets—about half the full 30-packet sequence—with per-class commit points K^* ranging from 2 to 30 (mean 15.7, median 12). The most informative comparison with TLS is here: QUIC commits markedly later (mean exit $K \approx 15$ versus $\approx$ 7–8 on TLS). This is a genuine protocol effect rather than a modelling artefact—QUIC flows open with a large padded Initial packet and run longer, so discriminative information accrues later in the flow. (For comparability with the rest of the paper, the adaptive macro-F1 quoted in Table 11 averages over classes present in the test week; a mean over all 47 classes, including those barely present in WQ42, is lower at 0.733.)

Extended K version

Because QUIC commits around $K \approx 15$, the main model's checkpoint set $\{1{-}15, 20, 25, 30\}$ is sparse in precisely the 15–30 region where most QUIC decisions fall. To resolve this region we retrained the same base model with an exit head at every integer $K \in \{1, \ldots, 30\}$ (30 exits; conformal, OOD, and drift components are not applied). Figure 12 shows the resulting fixed-K curve, which now climbs smoothly and without gaps from accuracy 0.866 at $K = 15$ to 0.943 at $K = 30$. Two effects follow from the finer grid. First, the adaptive policy commits slightly *earlier* for the same accuracy—mean exit falls from 15.01 to 14.26 packets at essentially unchanged accuracy (0.925)—because each class can now stop at its true sufficiency point rather than at the next available sparse checkpoint. Second, full-sequence ($K = 30$) accuracy nudges up from 0.9387 to 0.9433, as the additional exit heads provide extra supervision during training (Table 12). We report accuracy and weighted-F1 (0.925) as the stable metrics here; the adaptive macro-F1 differs slightly (0.693 versus 0.733) but, being a mean over all 47 classes including those barely present in the single WQ42 test week, it is noisy and we do not read into the difference. The practical cost of the denser grid is only a set of additional—and individually cheap—linear classification heads.

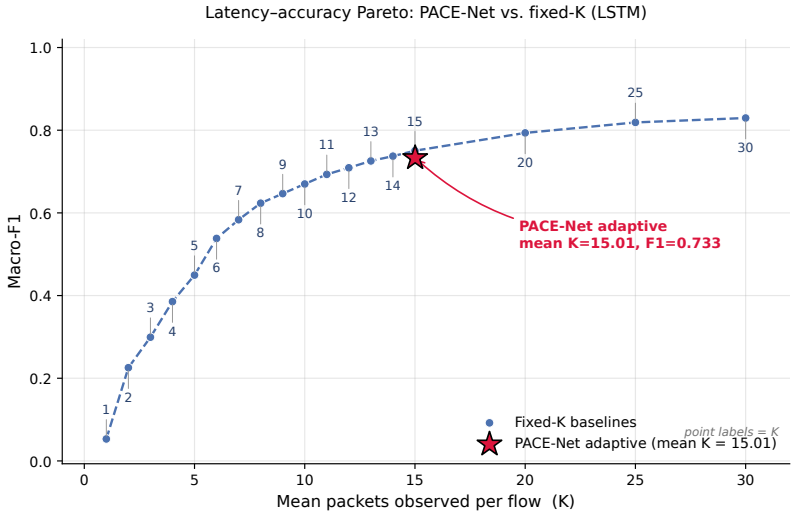


Fig. 11: Latency–accuracy trade-off on the QUIC held-out week WQ42 (LSTM backbone). Fixed-K baselines (curve) versus the adaptive policy (marked point, mean $K = 15.0$). QUIC commits about twice as late as TLS, reflecting its longer, padded flows.

Table 12: QUIC early-exit grid: the main 18-exit model versus a dense 30-exit variant (base model, WQ42 test). A finer grid lets the adaptive policy commit marginally earlier at equal accuracy and slightly raises full-sequence accuracy.

Exit grid	Adaptive mean K	Adaptive acc.	Fixed-K=30 acc.
18 exits (main)	15.01	0.9247	0.9387
30 exits (extended)	14.26	0.9251	0.9433

6.4 Temporal Drift on Held-Out QUIC Weeks

To mirror the TLS drift study on QUIC, we evaluate the model on the eight consecutive future weeks WQ43–WQ50—weeks the model, trained on WQ1–WQ40 and validated on WQ41, never observed. As in Section 5.4, the model is run unmodified each week, per-class CUSUM monitors flag drift on mean K_{used} and realised precision, and flagged classes receive the bias-only head retrain. A class counts as degraded if its per-week F1 falls more than five points below its WQ41 baseline (mean per-class F1 0.7265); each week is a 400k-flow sample for tractability. Table 13 reports the trajectory under both accuracy and macro-F1.

Reporting the two metrics together is essential, because they tell different stories. Over the eight weeks, accuracy falls from 0.9214 to 0.9053—a loss of only 1.6 points—while macro-F1 falls from 0.7194 to 0.6614, a loss of 5.8 points. The gap is explained by the class distribution: accuracy is dominated by a handful of very large applications (chiefly Google and Meta traffic) whose behaviour remains stable, whereas macro-F1 weights every application equally and therefore exposes the drift, which concentrates in the smaller and mid-sized applications. Accuracy alone would substantially understate the degradation, and we report both for that

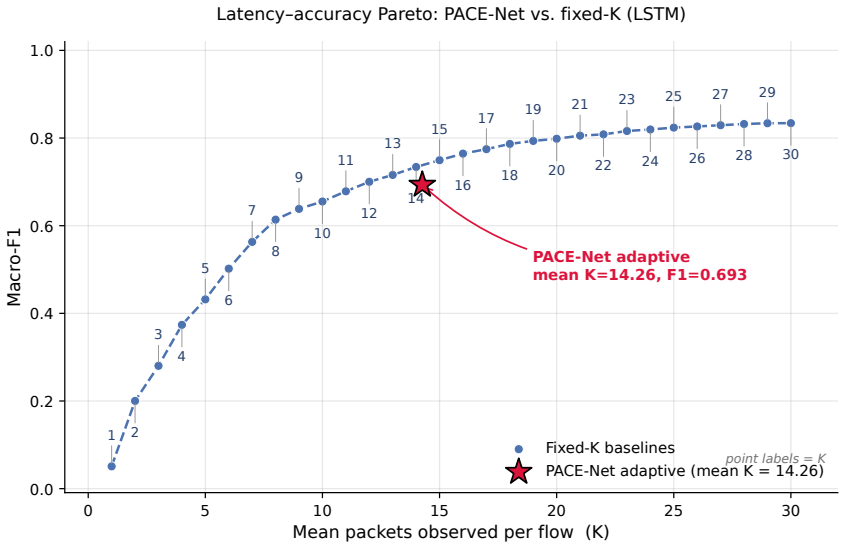


Fig. 12: Latency–accuracy trade-off on QUIC (WQ42, LSTM) with an exit head at every integer $K \in \{1, \ldots, 30\}$. The dense grid resolves the 15–30 region smoothly; the adaptive policy (star) commits at a mean of 14.26 packets.

reason. The supporting signals move consistently: the number of degraded classes grows from 5 at WQ43 to 13–15 from WQ46 onward, and the mean exit point creeps up from 15.9 to 16.5 packets as the model requires more evidence to remain confident on shifting traffic.

As on TLS, the drift is broad but mild: although 13–15 classes each lose more than five points, each individually stays below the CUSUM alarm thresholds, so detection recall is low (0–0.17) and only one to six classes are flagged per week; when the monitor does fire, it fires on genuinely degraded classes (precision up to 0.33).

The more instructive finding concerns the repair step. Unlike the bias-only update's small but consistent gains on TLS, here it recovers essentially nothing—Δaccuracy and Δmacro-F1 are both $\leq +0.0002$ in every week—despite running in only 2.7–3.3 seconds. The reason is diagnostic rather than incidental: a bias-only update can only rebalance per-class priors, whereas QUIC's drift is *representational*, meaning the learned feature-to-class mapping itself shifts. Rebalancing priors cannot recover a representational shift. We therefore read this as an negative result for the lightweight rung of the adaptation ladder on QUIC: the CUSUM monitor still detects drift precisely, but the cheap bias-only fix that suffices on TLS does not transfer, and periodic retraining of the classification heads (or the backbone) is the heavier but effective adaptation step this protocol requires.

6.5 Positioning Relative to the TLS Results

Retrained from scratch, CAPE-Net becomes a QUIC-native model that classifies its traffic at accuracy comparable to the TLS setting, with the conformal and unknown-detection mechanisms stacking much as they did before. What changes is the timing. QUIC commits later—a mean exit of roughly fifteen packets against seven or eight on TLS—because its flows open

Table 13: QUIC temporal drift over eight held-out future weeks WQ43–WQ50 (LSTM, 47 classes). Accuracy is measured over all flows of the week; macro-F1 is a mean over the classes present that week. "Degr." counts classes whose F1 fell more than 5 pp below the WQ41 baseline; "Flag." counts CUSUM alarms. Accuracy drifts by only 1.6 pp while macro-F1 drifts by 5.8 pp, because the degradation concentrates in smaller classes. Values are within-track per-week figures and are not comparable to the WQ42 in-distribution results of Section 6.3.

Week	Accuracy		macro-F1 (before)	Degr.	Flag.	Prec.	Rec.
	before	after					
WQ43	0.9214	0.9221	0.7194	5	1	0.00	0.00
WQ44	0.9153	0.9159	0.7273	5	4	0.00	0.00
WQ45	0.9123	0.9128	0.7211	6	5	0.20	0.17
WQ46	0.9085	0.9089	0.6848	13	6	0.33	0.15
WQ47	0.9063	0.9068	0.6761	15	5	0.00	0.00
WQ48	0.9087	0.9094	0.6642	14	6	0.17	0.07
WQ49	0.9092	0.9098	0.6686	13	5	0.00	0.00
WQ50	0.9053	0.9058	0.6614	13	5	0.00	0.00
Δ	−1.6 pp		−5.8 pp				

with a large padded Initial packet and run longer, so discriminative structure simply accrues further into the sequence. Conformal compliance and unknown-detection AUROC fall somewhat as well, which is consistent with those longer flows and with a harder test: a single future week, fewer classes, and a label set transferred through SNI whose coverage erodes as new applications appear. The practical lesson is that a latency budget calibrated on TLS cannot be transplanted to QUIC unchanged.

Drift tells a subtler story, and it is the one most worth dwelling on. Detection transfers cleanly. On QUIC, as on TLS, accuracy erodes steadily across the held-out weeks; the drift is broad but mild, with many classes each losing a little rather than a few collapsing, so relatively few cross the CUSUM thresholds —and when the monitor does fire, it fires on classes that have genuinely degraded. Repair is another matter. The bias-only update that recovers a small, consistent margin every week on TLS recovers essentially nothing on QUIC, barely +0.0002 accuracy per week, even though it costs only seconds to run. The reason is structural rather than incidental: adjusting output biases can only rebalance per-class priors, and QUIC's drift is representational—the learned mapping from features to classes has itself shifted, and no reweighting of the final layer will undo that.

This asymmetry refines rather than undermines the maintenance loop. It tells us that the lightweight rung of the adaptation ladder is protocol-dependent: an always-safe net against the prior-shift-like drift we observe on TLS, but not a substitute for retraining the classification heads when the representation itself has moved, as it does on QUIC. The monitoring machinery remains valuable in both cases, and arguably becomes more so on QUIC, since it is precisely what signals to an operator that the cheaper intervention has been exhausted and a heavier one is due. Read together, the two protocols suggest a maintenance policy rather than a single mechanism—monitor continuously, repair cheaply where cheap repair works, and retrain at the head level when it does not.

A final observation runs alongside all of this. Because QUIC's decisions cluster around fifteen packets, the exit grid is best matched to where they actually fall: densifying the checkpoints in that region lets each class stop at its true sufficiency point, and moving from eighteen exits to thirty lowers the mean exit from 15.01 to 14.26 packets at unchanged accuracy. The grid, in other words, is a design knob to be tuned per protocol, not a constant of the architecture.

7 Discussion

The results across both datasets support a single conceptual claim: the early-exit structure is most valuable when treated not only as an efficiency trick but also as a multi-purpose inference instrument. The same per-exit heads that let a flow commit early also carry a calibrated per-class precision guarantee, an abstention score for unknown traffic, and—through their per-class exit-depth and precision statistics—a signal of concept drift. Below we draw out what this means in practice for an operator, and when each design choice pays off.

Per-class conformal calibration proves to be the single most reliable lever across both TLS and QUIC. It lifts macro-F1 and pushes the realised precision of committed flows to roughly 0.97–0.99, turning an uncontrolled empirical threshold into a finite-sample guarantee. The cost is twofold and should be stated plainly. First, calibration raises the mean exit point, because committing only on high confidence necessarily means waiting longer on borderline flows. Second, the guarantee is purchased with coverage: a meaningful fraction of flows (about a quarter on TLS, a third on QUIC) are deferred or abstained rather than committed. For an operator, this is the right trade when a confident wrong label is more costly than a deferral—as in traffic policing, billing, or security triage—and the wrong trade when every flow must receive a label regardless. The mechanism is opt-in precisely so that this decision rests with the operator rather than being baked into the model.

The choice between the GRU and LSTM backbones is more subtle than the in-distribution numbers alone suggest. The two are close on accuracy and macro-F1, with the LSTM marginally ahead at the best TLS operating point. The sharper distinction lies in unknown-application detection, where the two backbones reverse their ordering between configurations: the GRU leads decisively in the extended-K TLS run (AUROC 0.87 versus 0.69), whereas the standard run favours the LSTM. We do not read this as evidence that one cell is intrinsically better at detecting unknowns; rather, it is a caution that energy-based detection performance is sensitive to exit granularity and backbone in ways that in-distribution accuracy is not. The practical implication is that unknown-detection behaviour must be validated for the specific backbone and checkpoint set an operator intends to deploy, not assumed to carry over from the classification metrics.

The drift loop, read correctly, behaves exactly as an operational monitor should. Over the ten-week TLS trajectory it stays quiet during the noisy early weeks, when week-to-week variation is mostly sampling noise, and its alarms become trustworthy—detection precision rising to 0.82—only once genuine degradation accumulates. The bias-only repair recovers a small, consistent margin each week at a cost of tens of seconds, and because it adjusts only the biases of flagged classes, it cannot harm the rest of the model. It is therefore best understood as a lightweight, always-safe maintenance mechanism that slows degradation between full retrainings, not as a substitute for periodic retraining once drift becomes severe. An operator

should expect it to extend the usable lifetime of a deployed model, not to hold accuracy constant indefinitely.

Finally, that the same architecture and the same mechanisms carry over to a QUIC-native model trained from scratch is itself informative: the approach is not tied to TCP-specific features. The one robust behavioural difference is that QUIC commits roughly twice as late (mean exit $K \approx 15$ versus $\approx$ 7–8 on TLS). This is a genuine protocol effect—QUIC opens with a large padded initial packet and its flows run longer, so discriminative structure accrues later—and it means latency budgets calibrated on TLS cannot be transplanted to QUIC unchanged. The lower conformal compliance and unknown-detection AUROC on QUIC are consistent with its longer flows and with a harder evaluation: a single future week, fewer classes, and a label set transferred via SNI whose coverage decays as new applications appear. Taken together, the two studies suggest that early, calibrated, and maintainable classification is achievable across protocols within one disciplined design, provided its latency expectations are recalibrated to the protocol at hand.

8 Limitations and Future Work

Several limitations bound the scope of our claims and point to concrete next steps.

Datasets and label provenance. Our evidence rests on two CESNET corpora. While these are large and both span a full year, both originate from a similar operational vantage point, and the QUIC labels are transferred from the TLS taxonomy through SNI rather than independently ground-truthed; SNI coverage declines over the year as new applications appear. Replicating the approach on independently labelled traffic from a different network, and on additional protocols, would strengthen external validity.

Robustness to packet loss and reordering. We evaluate on the packet sequences as recorded and do not test robustness to dropped or retransmitted packets. Because the PPI representation is order-sensitive, a flow whose early packets are lost or reordered presents a structurally different prefix than the model saw in training. Drop-robust training—random packet masking and jitter, or a permutation-tolerant readout—is a natural and inexpensive extension that we leave to future work.

OOD estimation under sparse unknowns. The OOD evaluation is limited by how few held-out unknown flows appear in any single test week—on QUIC, only on the order of a thousand—which makes the AUROC estimate noisier than the in-distribution metrics. Pooling unknown traffic across multiple weeks, or curating a dedicated unknown-application benchmark, would give a more stable estimate. The backbone/granularity sensitivity of the energy detector noted in Section 7 also merits dedicated study.

Drift recovery and labelling burden. The bias-only repair is deliberately conservative; it recovers a small margin per week and is not intended to counter severe drift. It also presupposes a supply of recent labelled flows for the flagged classes, which carries an operational labelling cost. Quantifying that cost, and exploring lighter-weight or partially self-supervised label sources for the repair step, are worthwhile directions. More ambitiously, the per-class exit-depth distribution could itself be developed into a richer drift descriptor than the precision and mean-K signals we currently monitor.

Single-backbone QUIC study. The QUIC evaluation uses only the LSTM backbone; a GRU counterpart, and the drift loop applied to QUIC's full year, are left for future work.

Backbone architecture. We restrict this work to compact recurrent backbones, whose constant-time per-packet update suits a streaming model read out at eighteen checkpoints; our calibration and abstention procedures are, however, agnostic to how the per-exit logits are produced. Extending CAPE-Net to a transformer backbone—with causal, cached attention to keep a fine-grained exit grid affordable—is a natural next step, particularly for QUIC, where discriminative information accrues later in the flow.

9 Conclusion

We presented CAPE-Net (Conformal Adaptive Packet-Efficient Network), an early-exit classifier for encrypted traffic that treats its exit structure as a shared instrument for efficiency, calibrated precision, abstention, and drift sensing. On the 180-class CESNET-TLS-Year22 corpus, per-class conformal calibration delivers a finite-sample 0.95 precision guarantee that holds for roughly 98% of classes a week past calibration; an energy-based detector abstains on unknown applications; and a safe, bias-only drift loop slows degradation over a ten-week horizon at a cost of seconds per update. The typical flow is classified from about its first seven to eight packets, with the most confident classes certified to commit on as few as three, outperforming previous state of the art [11]. Retrained from scratch on QUIC (CESNET-QUICEXT-25), the same design classifies a held-out future week at 0.94 accuracy with the precision guarantee intact, while committing later in line with QUIC's longer, padded flows. Although the lightweight drift-maintenance mechanism substantially slows degradation on TLS, its impact on QUIC is more modest, suggesting that different encrypted protocols may require protocol-specific maintenance strategies. Taken together, the results indicate that early, reliable, adaptive, and maintainable encrypted-traffic classification at ISP scale is achievable within a single architecturally disciplined model—and that the early-exit structure, properly calibrated, is what makes those goals compatible rather than competing.

Author Contributions. Conceptualization, F.R., J.E.L.d.V, L.d.P.; methodology, F.R., J.E.L.d.V, L.d.P.; software, F.R., I.G.; validation, J.E.L.d.V, L.d.P.; formal analysis, F.R., J.E.L.d.V, L.d.P.; investigation, F.R., J.E.L.d.V, L.d.P.; resources, I.G.; data curation, F.R.; writing—original draft preparation, F.R., J.E.L.d.V, L.d.P.; writing—review and editing, F.R., J.E.L.d.V, L.d.P., I.G.; visualization, F.R.; supervision, J.E.L.d.V, L.d.P.; project administration, J.E.L.d.V. All authors have read and agreed to the published version of the manuscript.

Data availability. Datasets have been made publicly available by CESNET at: https://zenodo.org/records/10608607 (CESNET-TLS-Year22) and https://zenodo.org/records/17249078 (CESNET-QUICEXT-25).

Code availability. All scripts and models are available at: https://github.com/farzamrezaei/cape-net-traffic-classification

Acknowledgement. This work was supported in part by a scholarship of the Department of Electronic and Communications Technologies, Universidad Autónoma de Madrid, and by

the R&D activity program with reference TEC-2024/COM-504 and acronym RAMONES-CM, granted by the Comunidad de Madrid through the Directorate General for Research and Technological Innovation via Order 5696/2024.

Competing interests. The authors declare no competing interests.

References

[1] Xiao, X., Xiao, W., Li, R., Luo, X., Zheng, H., Xia, S.: EBSNN: Extended Byte Segment Neural Network for Network Traffic Classification. IEEE Transactions on Dependable and Secure Computing **19**(5), 3521–3538 (2022) https://doi.org/10.1109/TDSC.2021.3101311

[2] Shapira, T., Shavitt, Y.: Flowpic: Encrypted internet traffic classification is as easy as image recognition, pp. 680–687 (2019). https://doi.org/10.1109/INFCOMW.2019.8845315

[3] Diao, Z., Xie, G., Wang, X., Ren, R., Meng, X., Zhang, G., Xie, K., Qiao, M.: EC-GCN: A encrypted traffic classification framework based on multi-scale graph convolution networks. Computer Networks **224** (2023) https://doi.org/10.1016/j.comnet.2023.109614

[4] Chen, Z., Cheng, G., Niu, D., Qiu, X., Zhao, Y., Zhou, Y.: WFF-EGNN: Encrypted Traffic Classification Based on Weaved Flow Fragment via Ensemble Graph Neural Networks. IEEE Transactions on Machine Learning in Communications and Networking **1**, 389–411 (2023) https://doi.org/10.1109/TMLCN.2023.3323915

[5] Shi, Y., Li, G., Wu, J., Li, J., Fang, H.: BPF-DAG: Byte-Packet-Flow Features Fusion via Dynamic Attributed Graph for Reliable Encrypted Traffic Classification. IEEE Transactions on Information Forensics and Security **21**, 197–211 (2026) https://doi.org/10.1109/TIFS.2025.3643127

[6] Zhao, Y., Zhu, Y., Li, X., Chen, R., Obaidat, M.S., Vijayakumar, P.: SD-Transformer: A System-Level Denoising Transformer for Encrypted Traffic Behavior Identification. In: Proceedings - IEEE Global Communications Conference, GLOBECOM, pp. 5153–5158 (2023). https://doi.org/10.1109/GLOBECOM54140.2023.10436868

[7] Miyamoto, K., Han, C., Ban, T., Takahashi, T., Takeuchi, J.: Intrusion Detection Simplified: A Feature-free Approach to Traffic Classification Using Transformers. In: Proceeding - 2024 Annual Computer Security Applications Conference Workshops, ACSACW 2024, pp. 20–29 (2024). https://doi.org/10.1109/ACSACW65225.2024.00011

[8] Li, S., Huang, Y., Gao, T., Yang, L., Chen, Y., Pan, Q., Zang, T., Chen, F.: FusionTC: Encrypted App Traffic Classification Using Decision-Level Multimodal Fusion Learning

of Flow Sequence. Wireless Communications and Mobile Computing **2023** (2023) https://doi.org/10.1155/2023/9118153

[9] Pang, B., Fu, Y., Ren, S., Shen, S., Wang, Y., Liao, Q., Jia, Y.: A Multi-Modal Approach For Context-Aware Network Traffic Classification. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, vol. 2023-June (2023). https://doi.org/10.1109/ICASSP49357.2023.10095124

[10] Malekghaini, N., Akbari, E., Salahuddin, M.A., Limam, N., Boutaba, R., Mathieu, B., Moteau, S., Tuffin, S.: AutoML4ETC: Automated Neural Architecture Search for Real-World Encrypted Traffic Classification. IEEE Transactions on Network and Service Management **21**(3), 2715–2730 (2024) https://doi.org/10.1109/TNSM.2023.3324936

[11] Babaria, R.J., Lyu, M., Batista, G., Sivaraman, V.: FastFlow: Early Yet Robust Network Flow Classification using the Minimal Number of Time-Series Packets. Proceedings of the ACM on Measurement and Analysis of Computing Systems **9**(2) (2025) https://doi.org/10.1145/3727115

[12] Kim, H., Yoon, S., Bae, C., Lee, S., Pack, S.: TINIEE: Traffic-Aware Adaptive In-Network Intelligence via Early-Exit Strategy. IEEE Transactions on Networking **34**, 5037–5052 (2026) https://doi.org/10.1109/TON.2026.3689743

[13] Kadziolka, B., Jurkiewicz, P., Wojcik, R., Domzal, J.: Elephant Flow Classification on the First Packet With Neural Networks. IEEE Access **12**, 65298–65309 (2024) https://doi.org/10.1109/ACCESS.2024.3398065

[14] Jurkiewicz, P., Kadziołka, B., Kantor, M., Wójcik, R., Domżał, J.: Elephant Flow Detection With Random Forest Models Under Programmable Network Dataplane Constraints. IEEE Access **12**, 158561–158578 (2024) https://doi.org/10.1109/ACCESS.2024.3485588

[15] Xie, S., Hu, G., Xing, C., Liu, Y.: Online Elephant Flow Prediction for Load Balancing in Programmable Switch-Based DCN. IEEE Transactions on Network and Service Management **21**(1), 745–758 (2024) https://doi.org/10.1109/TNSM.2023.3318752

[16] Jiang, Y., Wang, X., Lai, Y., Wang, Y.: A Packet Sequence Permutation-Aware Approach to Robust Network Traffic Classification. IEEE Networking Letters **6**(3), 203–207 (2024) https://doi.org/10.1109/LNET.2024.3435723

[17] Cheng, J., Zeng, F.: SimSeq: A Robust TLS Traffic Classification Method. In: Lecture Notes in Computer Science, vol. 16186 LNCS, pp. 69–88 (2026). https://doi.org/10.1007/978-3-032-08124-7_4

[18] Tawfeeq, T.M., Nickray, M.: Adversarial Training for Improved VPN Traffic Classification Using EfficientNet-B0 and Projected Gradient Descent. International Journal of Intelligent Engineering and Systems **18**(1), 1200–1215 (2025) https://doi.org/10.22266/ijies2025.0229.87

[19] Shin, C.-Y., Choi, Y.-S., Kim, M.-S.: Data Augmentation-Based Enhancement for Efficient Network Traffic Classification. IEEE Access **13**, 6006–6028 (2025) https://doi.org/10.1109/ACCESS.2024.3525000

[20] Luxemburk, J., Hynek, K., Plný, R., Čejka, T.: Universal Embedding Function for Traffic Classification via QUIC Domain Recognition Pretraining: A Transfer Learning Success. IEEE Transactions on Network and Service Management **23**, 1647–1663 (2026) https://doi.org/10.1109/TNSM.2025.3642984

[21] Montieri, A., Nascita, A., Pescapè, A.: From prompts to packets: A view from the network on ChatGPT, Copilot, and Gemini. Computer Networks **281** (2026) https://doi.org/10.1016/j.comnet.2026.112237

[22] Li, J., Cheng, G., Chen, Z.: Early Classification and Improved Performance: A Multi-Model Serial Encrypted Traffic Classification Framework. In: Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom, pp. 2843–2851 (2025). https://doi.org/10.1109/Trustcom66490.2025.00337

[23] Sun, P., Lyu, M., Li, H., Yang, B., Peng, L.: An early stage convolutional feature extracting method using for mining traffic detection. Computer Communications **193**, 346–354 (2022) https://doi.org/10.1016/j.comcom.2022.06.044

[24] Wang, Y., He, H., Lai, Y., Liu, A.X.: A Two-Phase Approach to Fast and Accurate Classification of Encrypted Traffic. IEEE/ACM Transactions on Networking **31**(3), 1071–1086 (2023) https://doi.org/10.1109/TNET.2022.3209979

[25] Fu, N., Cheng, G., Su, X.: Accurate compressed traffic detection via traffic analysis using Graph Convolutional Network based on graph structure feature. Computer Communications **207**, 128–139 (2023) https://doi.org/10.1016/j.comcom.2023.04.031

[26] Guo, S., Wu, R., Hu, N.: DAT: A Two-Stage Encrypted Traffic Classification Method Based on C4.5 Decision Tree and Transformer. In: Proceedings - 2025 IEEE 10th International Conference on Data Science in Cyberspace, DSC 2025, pp. 216–223 (2025). https://doi.org/10.1109/DSC67331.2025.00036

[27] Ma, X., Wang, Z., Zheng, Z., Jiang, M., Zhang, X., Liu, X.: A lightweight encrypted traffic classification method using contextual information. Computer Networks **274** (2026) https://doi.org/10.1016/j.comnet.2025.111798

[28] Yue, Y., Chen, X., Han, Z., Zeng, X., Zhu, Y.: Contrastive Learning Enhanced Intrusion Detection. IEEE Transactions on Network and Service Management **19**(4), 4232–4247 (2022) https://doi.org/10.1109/TNSM.2022.3218843

[29] Chen, J., Zhou, Y.: Next-Packet Prediction with Temporal Transformer Networks: A Non-Contrastive Self-Supervised Approach for AI-Powered Cybersecurity Threat Detection. In: 2025 5th International Conference on Network Communication and Information

Security, ICNCIS 2025, pp. 123–127 (2025). https://doi.org/10.1109/ICNCIS67521.2025.11296104

[30] Sabahi-Kaviani, Z., Ghassemi, F.: An encrypted traffic classifier via combination of deep learning and automata learning. Soft Computing **28**(23), 13443–13460 (2024) https://doi.org/10.1007/s00500-024-10383-0

[31] Shi, Q., Shi, D., He, L., Xiao, Q., Zheng, L., Ma, J., He, J., Zhang, Q.: TransGraphNet: Robust detection of malicious encrypted network traffic via transformer and graph neural models. PeerJ Computer Science **11** (2025) https://doi.org/10.7717/peerj-cs.3353

[32] Simioni, J.A., Viegas, E.K., Santin, A.O., Matos, E.: An Energy-Efficient Intrusion Detection Offloading Based on DNN for Edge Computing. IEEE Internet of Things Journal **12**(12), 20326–20342 (2025) https://doi.org/10.1109/JIOT.2025.3544060

[33] Ayodele, B., Buttigieg, V.: Evaluating Cross-Domain Generalisation of Deep Learning Models in Network Traffic Classification: A Statistical and Experimental Analysis of VNF and PNF. Journal of Network and Systems Management **34**(3) (2026) https://doi.org/10.1007/s10922-026-10049-2

[34] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6**(2), 65–70 (1979)

[35] Dunn, O.J.: Multiple comparisons among means. Journal of the American Statistical Association **56**(293), 52–64 (1961) https://doi.org/10.1080/01621459.1961.10482090

Farzam REZAEI is assistant researcher at Universidad Autónoma de Madrid (Spain), where he is pursuing his PhD since 2025. Previously, He received his M.Sc. degree in Computer Engineering (Computer Networks) from Sharif University of Technology (Iran) in 2023, and his B.Sc. degree from the University of Tehran (Iran) in 2019. Before beginning his doctoral studies, he worked for several years as a backend and DevOps engineer and technical team lead, and as a research assistant in computer-architecture, network and cybersecurity laboratories. His research interests include network traffic analysis and monitoring, and network security.

Jorge E. LÓPEZ DE VERGARA is associate professor at Universidad Autónoma de Madrid (Spain) since 2007 and accredited as full professor since 2023. He was a founding partner from 2009 to 2023 of Naudit HPCN, a spin-off company devoted to high-performance traffic monitoring and analysis, and is currently a founding partner of Qarina HPC, another spin-off company, devoted to quantum computing and artificial intelligence. He received his M.Sc. (Honors) and Ph.D. (Honors) degrees in Telecommunication Engineering from Universidad Politécnica de Madrid (Spain) in 1998 and 2003, respectively, where he also held an FPU-MEC research grant. During his Ph.D., he stayed for 6 months in 2000 at HP Labs in Bristol,

U.K. He studies network and service management and monitoring, and has coauthored more than 100 scientific papers on topics related to this field.

Luis de PEDRO is associate professor at Universidad Autónoma de Madrid (Spain) since 1994, and has been president and founding partner from 2009 to 2023 of Naudit HPCN, a spin-off of the UAM, devoted to high performance traffic monitoring and analysis, and is currently a founding partner of Qarina HPC, another spin-off company, devoted to quantum computing and artificial intelligence. He received his M.Sc. (Honors) and Ph.D. (Honors) degrees in Telecommunication Engineering from Universidad Politécnica de Madrid (Spain) in 1987 and 1992, respectively. He has been a Hewlett-Packard Executive from 1986 to 2019. He currently researches on statistical models for network traffic.

Ivan GONZÁLEZ received his M.Sc. degree in Computer Engineering in 2000 and his Ph.D. in Computer Engineering in 2006, both from UAM, Spain. From October 2002 to October 2006, he was a teaching assistant at the Computer Engineering Department of UAM. From November 2006 to January 2008, he was a postdoctoral research scientist at the High-Performance Computing Laboratory (HPCL), Electrical & Computer Engineering Department, George Washington University (Washington, DC). He was a faculty member of the NSF Center of High-Performance Reconfigurable Computing (CHREC) at George Washington University. He is currently associate professor at UAM, where he is teaching computer-architecture-related courses. He was also partner of Naudit HPCN. His main research interests are heterogeneous computing (with GPUs, FPGAs, etc.), parallel algorithms, and performance tuning. Other interests include big data, machine learning and data analytics.