Challenge Draft: Application Inference from Packet Flows

Challenge at a Glance

PRIMARY TASK	MAIN METRIC	LEAKAGE CONTROL	TWO TASKS
Flow classification	**Macro F1**	**No DNS/SNI/IP**	**Static + realtime**
Infer applications from encrypted flows	Balances minority classes	Direct label hints are removed	50 points each

Public Challenge Information

- Challenge website: https://competition.aiforgood.itu.int/web/challenges/challenge-page/495/overview
- Participant-facing repository: https://github.com/FG-AI4H/application-inference-packet-flows-public
- Public release date: 1 August 2026
- Submission deadline: 23 October 2026
- Incentives: USD 1,000 prize money, winner certificates, and an additional supplementary prize from TTC are planned.

Background

The rapid diversification of mobile applications has made bandwidth, latency, and reliability requirements highly application-dependent. Video streaming, social networking services, web browsing, messaging, map services, and notification processing exhibit different traffic patterns. When traffic from multiple applications shares the same network path, application-agnostic control can cause congestion, delay, and packet loss, degrading users' Quality of Experience (QoE).

If network operators can identify application categories at an early stage, they can apply more appropriate QoS control, traffic prioritization, network slicing, bandwidth allocation, and anomaly detection. However, much of today's mobile traffic is encrypted by TLS and QUIC, making conventional identification based on direct content inspection difficult. Therefore, techniques that infer applications from statistical packet characteristics, such as packet length, inter-arrival time, traffic direction, and early-flow behavior, without relying on payload inspection are increasingly important.

Real-time operation and resource constraints are also central issues in mobile networks. For applications where many packets arrive shortly after connection establishment, delayed identification reduces the effectiveness of QoS control. At the same time, applying expensive feature extraction and inference to every packet increases processing delay and energy consumption, creating practical deployment costs. Therefore, application identification methods must not only be accurate, but also capable of early

inference from limited information, robust against premature misclassification, and efficient in their use of computational resources.

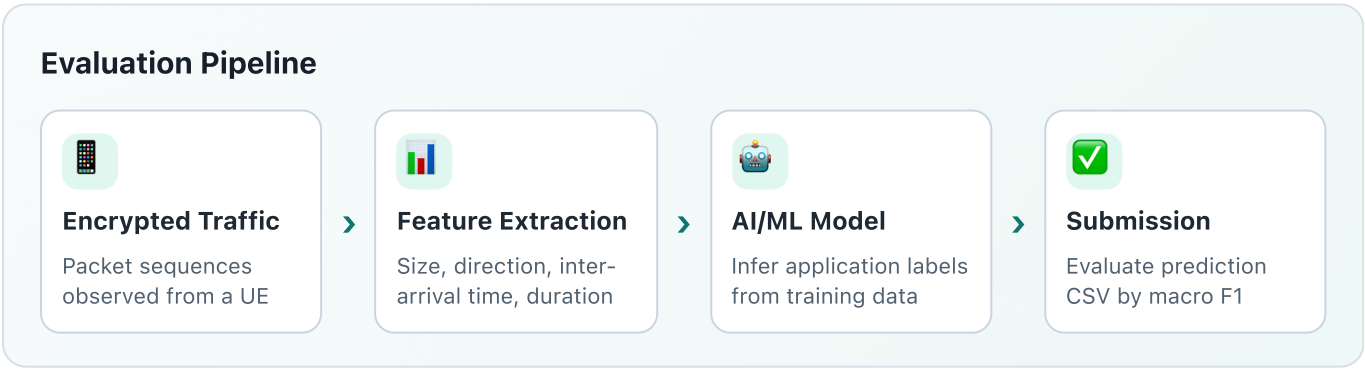
Objective

The objective of this challenge is to develop an AI/ML model that estimates the application associated with each communication flow using flow-level features extracted from encrypted mobile traffic. Participants will design methods that identify applications from statistical packet-sequence characteristics without directly relying on payload contents or domain names.

Participants will design models using the provided training data and predict application labels for flows in the evaluation data. Information directly tied to labels, such as DNS names, TLS SNI, HTTP Host, and IP addresses, is excluded from the participant feature files. Participants are therefore expected to use traffic volume, direction, packet size, inter-arrival time, burstiness, flow duration, and related features for inference.

This challenge expects methods that consider practical deployment in real networks, rather than focusing only on classification accuracy. Important perspectives include early identification from the first few packets, deferring decisions when confidence is low, lightweight model design, and learning methods robust to class imbalance. Through this task, the challenge aims to promote resource-efficient application identification techniques that can support real-time QoS control and application-aware network management.

Approach



Participants may freely design the following components.

- Feature selection
- Feature transformation, normalization, and dimensionality reduction
- AI/ML model selection
- Handling of class imbalance
- Generalization methods for applications with few samples

Possible approaches include the following.

1. Flow-statistics-based approach

This approach builds a classification model using statistics such as packet counts, byte counts, traffic direction, flow duration, packet size distribution, and inter-arrival times.

2. Time-series-based approach

This approach treats packet sequences as time-series data and models packet size sequences, direction sequences, inter-arrival time sequences, and related information.

3. Hybrid approach

This approach combines flow statistics and time-series features, or combines domain-knowledge-based feature engineering with machine learning models.

Tasks

Figure 1: Conceptual overview of application inference from encrypted packet flows

This challenge has two tasks with the same target application labels. Task 1 is static inference using full-flow features, and Task 2 is real-time inference using only information available at the beginning of a flow.

Task 1: Static Application Classification

For each flow in the evaluation data, participants predict one of the known application labels using statistical features computed from the full flow.

Dataset to use:

- `dataset/task1_static/`

Task 2: Real-Time Application Classification

For each flow in the evaluation data, participants predict one of the known application labels using only features computed from the first 5 packets of the flow.

Dataset to use:

- `dataset/task2_realtime_prefix5/`

For Task 2, participants must not use full-flow features from Task 1. Task 2 predictions must be generated only from files in `task2_realtime_prefix5/`.

Main target labels:

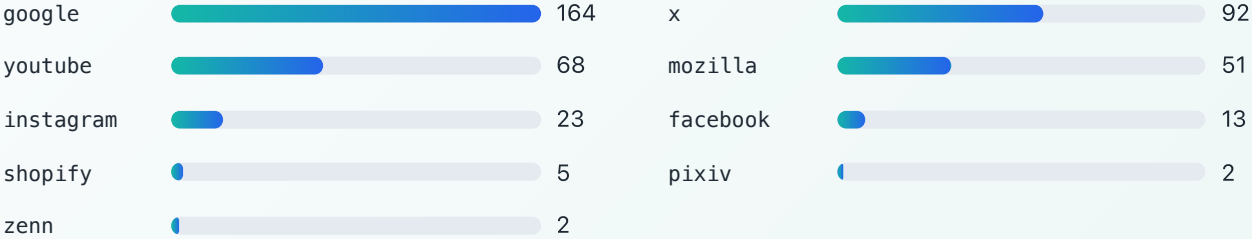
- `facebook`
- `google`
- `instagram`
- `mozilla`
- `x`
- `youtube`

Dataset

Dataset Structure

Task	Use	Train	Validation	Test	Points
`task1_static`	Static inference: full-flow features	247	83	81	50
`task2_realtime_prefix5`	Real-time inference: first-5-packet features	247	83	81	50

Closed-Set Label Distribution



The source data is a pcapng file containing approximately 44 minutes of traffic captured from a single UE. The organizers extract flow-level features from the pcapng file and provide them to participants in CSV format.

Each task directory contains the following CSV files.

- `train_features.csv`: flow-level features for training
- `train_labels.csv`: ground-truth labels for training flows
- `validation_features.csv`: flow-level features for validation
- `validation_labels.csv`: ground-truth labels for validation flows
- `test_features.csv`: flow-level features for evaluation
- `sample_submission.csv`: submission template

Organizer-only file:

- Final test ground-truth labels are held only by the organizers and are not distributed to participants.

The feature files include information such as the following.

- Flow duration
- Transport protocol
- Server-side port
- Uplink/downlink packet counts
- Uplink/downlink byte counts
- Mean and standard deviation of packet sizes
- Mean, standard deviation, minimum, and maximum of packet inter-arrival times
- Number of bursts
- Uplink/downlink ratios

The following information is not included in the participant feature files.

- IP addresses
- DNS query names
- TLS SNI
- HTTP Host
- URLs or domain strings
- Payload

Evaluation

> **Primary evaluation:** Macro F1 is used as the main metric, with accuracy reported as a secondary metric, to reduce the impact of class imbalance.

Participants submit predicted application labels for both tasks. The recommended submission format on the challenge platform is a JSON file with two top-level objects, `task1_static` and `task2_realtime_prefix5`.

Recommended JSON submission format:

```
{
  "task1_static": {
    "flow_000001": "youtube",
    "flow_000002": "instagram"
  },
  "task2_realtime_prefix5": {
    "flow_000001": "youtube",
    "flow_000002": "instagram"
  }
}
```

The evaluation script also accepts a ZIP file containing two CSV files:

- `task1_static.csv`
- `task2_realtime_prefix5.csv`

Each CSV file must contain predicted labels for the corresponding task.

```
flow_id,app
flow_000001,youtube
flow_000002,instagram
```

The primary evaluation metric is macro F1. Because the number of samples differs across classes, macro F1 evaluates performance on minority classes more appropriately than accuracy alone.

Accuracy is also calculated as a secondary metric.

The official score is 100 points. Task 1 and Task 2 are each worth 50 points, and each task score is based on macro F1.

- Task 1: 50 points
- Task 2: 50 points

Participants can use the public `validation_labels.csv` files to evaluate their methods locally. Final test labels are held only by the organizers and are not distributed to participants.

Conditions

- Ground-truth labels for evaluation data must not be used for training.
- Final test labels must not be distributed to participants.
- Full-flow features from Task 1 must not be used to generate Task 2 predictions.
- If information other than the provided features is used, it must be clearly described in the submitted report.

- If the pcapng file is additionally distributed, methods that directly infer labels from DNS names, TLS SNI, HTTP Host, or similar metadata should be treated as a separate track because they differ from the main objective of this challenge.

Submission

Participants submit the following two items.

1. Prediction file

A JSON file, or a ZIP file containing task-specific CSV files, with predicted labels for the evaluation flows.

2. Source code

Code used for training, inference, and submission generation. The organizers check the code to confirm that private labels and disallowed features are not used.

3. Technical report

A PDF file including the following items.

- Features used
- Model architecture
- Training method
- Handling of class imbalance
- Evaluation results
- Novelty of the proposed method
- Computational cost

Evaluation Criteria

Evaluation criterion #1: Estimation accuracy

Macro F1 is used as the primary metric. Accuracy is also checked as a reference value.

Evaluation criterion #2: Generalization performance

Methods that provide stable estimation for minority classes and low-volume flows are evaluated highly.

Evaluation criterion #3: Model design

Feature engineering, model architecture, handling of class imbalance, overfitting prevention, and related design choices are evaluated comprehensively.

Evaluation criterion #4: Practicality

Methods with low computational cost and high applicability to real network operations are evaluated highly.