NSDI '26

CascadeNet

Generating Network Traffic with High-Fidelity Temporal Patterns

Runwei Lu Yanran Deng Ruixuan Li Jinting Liu Yuejie Wang Xinyu Li Deming Xu Han Tian Kai Chen Guyue Liu

NYU Shanghai · Peking University · CMU · USTC · HKUST

Artifact Evaluated: Available · Functional · Reproduced

Why Synthetic Network Traces?

The Problem

Real network traces are scarce
Privacy concerns and proprietary restrictions limit access to real-world data

Critical for research
Traffic classification, anomaly detection, behavior analysis all need realistic traces

Synthetic Traces

ML-based generation is promising
Learn patterns from data — less manual effort than simulation or model-based methods

But a key gap remains
Existing methods fail to capture temporal dynamics — how traffic evolves over time

Existing Generators Distort Temporal Patterns

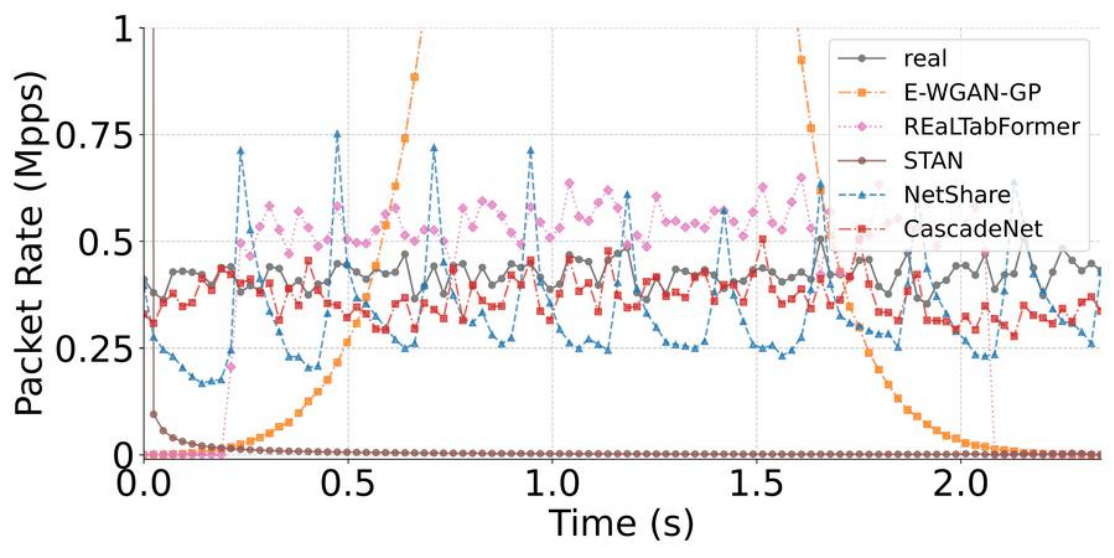


Figure 2: Packet rate — real vs. synthetic traces (CAIDA)

Key Observations

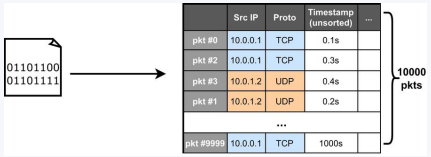
E-WGAN-GP & STAN
Completely distort the packet rate

NetShare
Learns a periodic pattern that contradicts reality

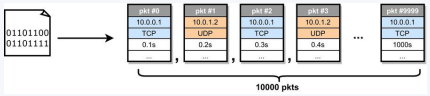
CascadeNet
Closely follows the real trace

Root cause:
Data representation fails to expose temporal patterns to models

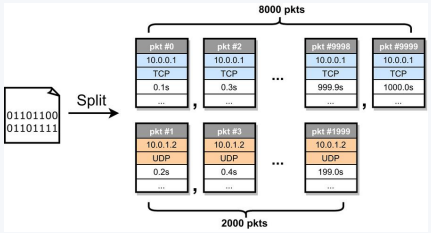
Data Representation: The Root Cause



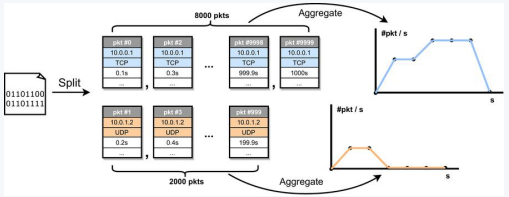
Approach 1: Collection of Records
E-WGAN-GP, CTGAN — ignores order



Approach 2: Sequence of Records
STAN — hard to learn long deps



Approach 3: Flows → Record Seqs
NetShare — still too long



Ours: Flows as Time Series
Explicit temporal patterns!

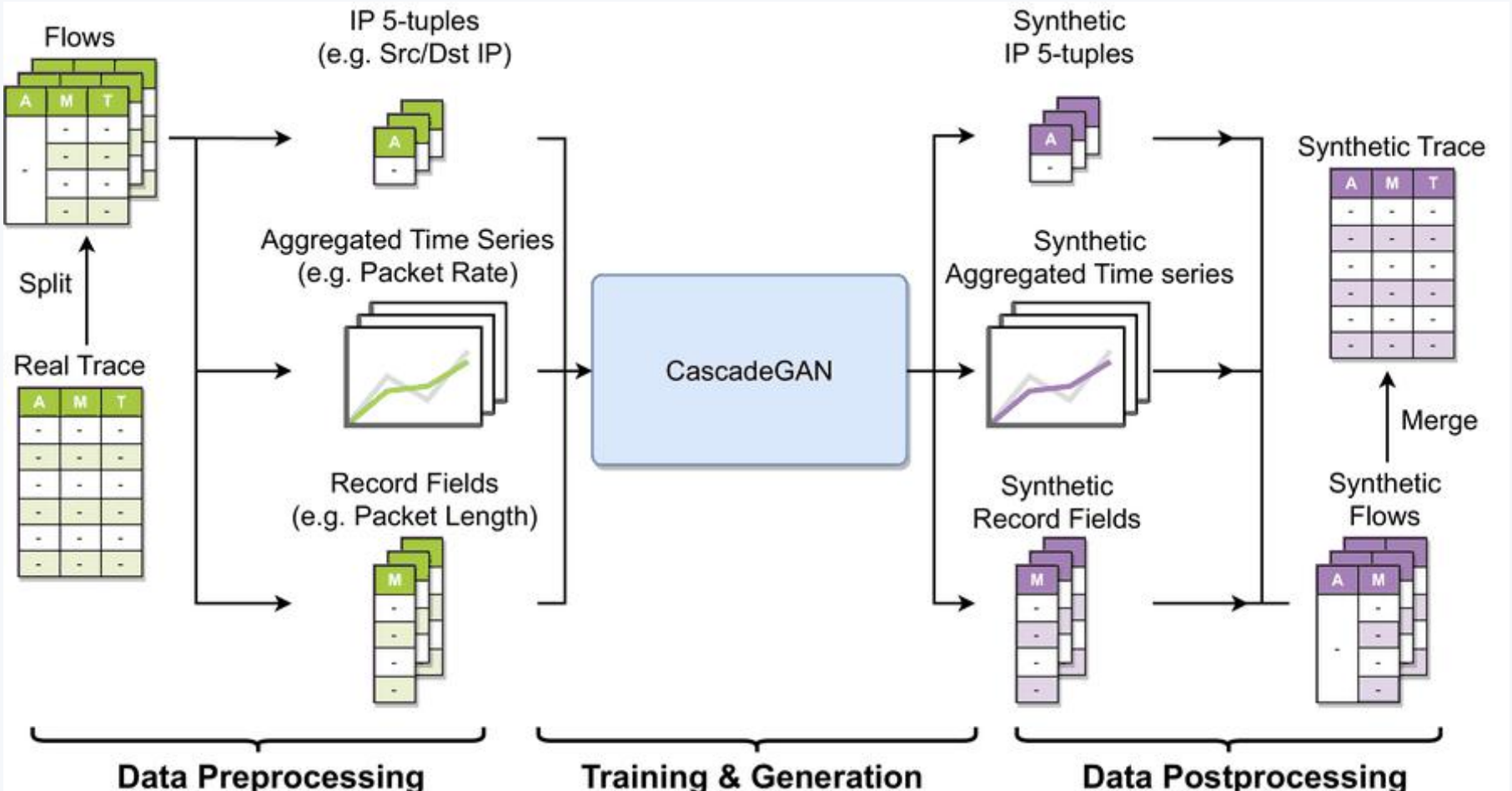
Our Insight

Temporal patterns (packet rate, throughput) are aggregated features
→ Models should learn them directly as time series, not infer from individual records

Advantages:

- Fidelity: Explicit temporal patterns + fixed-length series avoids long-sequence problems
- Scalability: Streamlined info → reduced sequence length → faster training & generation

CascadeNet: End-to-End Workflow



CascadeGAN: Hierarchical Generative Model

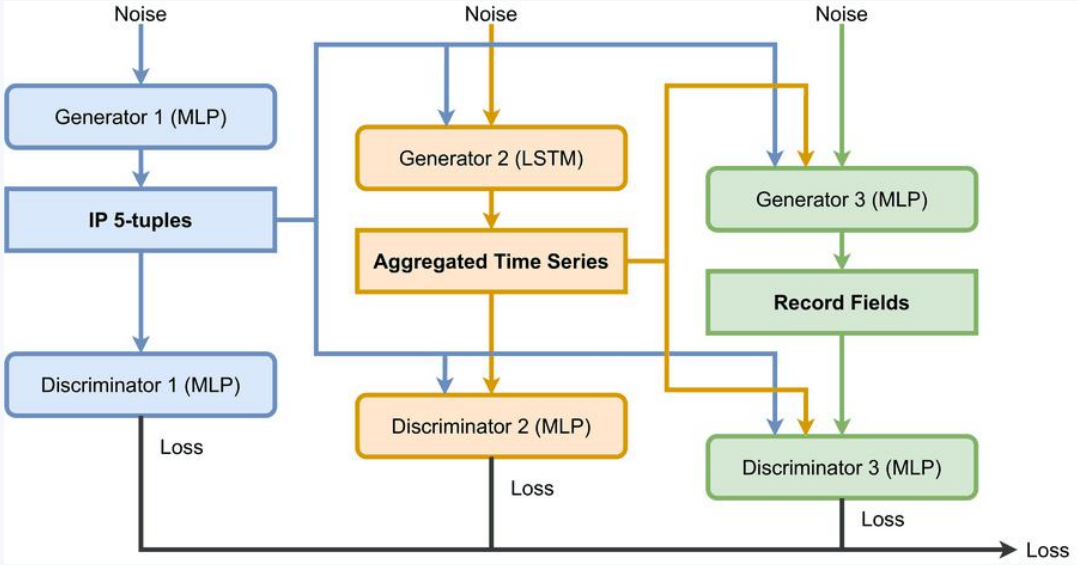


Figure 5: CascadeGAN architecture

Design Highlights

Gen 1 (MLP) → IP 5-tuples
Coarsest granularity — flow identifiers

Gen 2 (LSTM) → Time Series
Conditioned on 5-tuples; captures temporal dynamics

Gen 3 (MLP) → Record Fields
Conditioned on 5-tuples + time series (sliding window)

Key idea:
Convert joint distribution → conditional distributions (coarse-to-fine cascade)

Training: Two-phase strategy
1. Pretrain each GAN pair separately
2. Finetune all jointly ($L = \alpha L_1 + \beta L_2 + L_3$)

Avoids "cascading divergence"

Timestamp Generation & Optimizations

Timestamp Recovery

EQ **Equidistant**
Evenly space timestamps within each time step

SP **Sub-period**
Learn sub-period; space within active window

ML **ML-based ✓**
GAN 3 learns relative timestamps per time step

Optimization Techniques

Zero-Inflation Method

Time series are sparse (up to 88% zeros). Extra binary feature signals active vs. inactive time steps.

Conditional Input Strategy

Flow-level features (duration, max pkt rate) as conditional inputs → prevents mode collapse between mouse and elephant flows.

Both validated in ablation studies — each significantly improves temporal fidelity (Appendix G)

Evaluation Setup

4 Diverse IPv4 Header Traces

Dataset	Source	#Flows	#Packets	Duration
CAIDA	Backbone ISP	96K	999K	2.3s
DC	University DC	50K	1M	273s
CA	Cyber Defense	540K	1M	557s
TON_IoT	IoT Sensors	4.3K	445K	9802s

6 Baselines

- CTGAN (tabular GAN)
- E-WGAN-GP (flow GAN)
- STAN (auto-regressive)
- NetShare (SOTA)
- NetDiffusion (diffusion)
- REaLTabFormer (GPT-2)

Evaluation Dimensions

Temporal Fidelity

IT, FIT, FD, AT, BU, LD
(Earth Mover's Distance)

Statistical Fidelity

SA, DA, SP, DP, PR, PL,
TTL, NPF, NBF (JSD + EMD)

Downstream Tasks

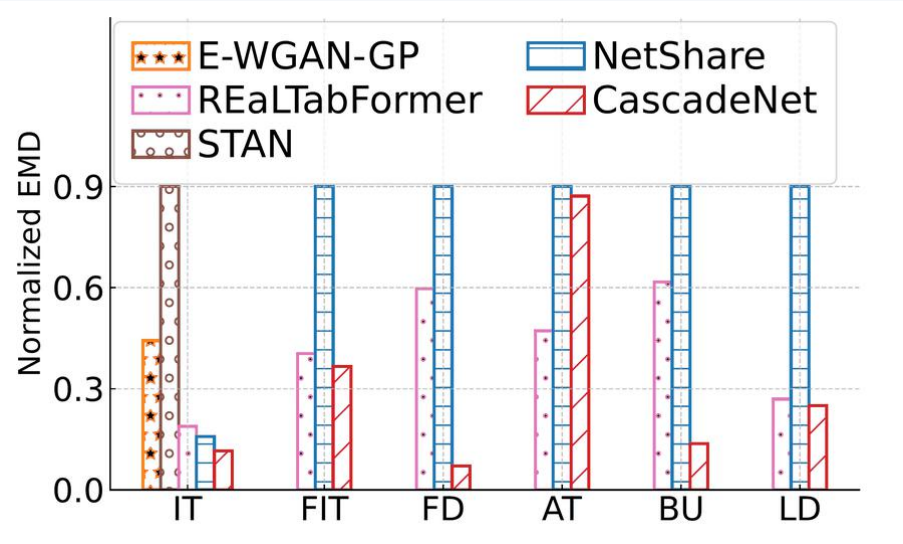
Burst analysis, throughput prediction, anomaly detection

Diversity & Scale

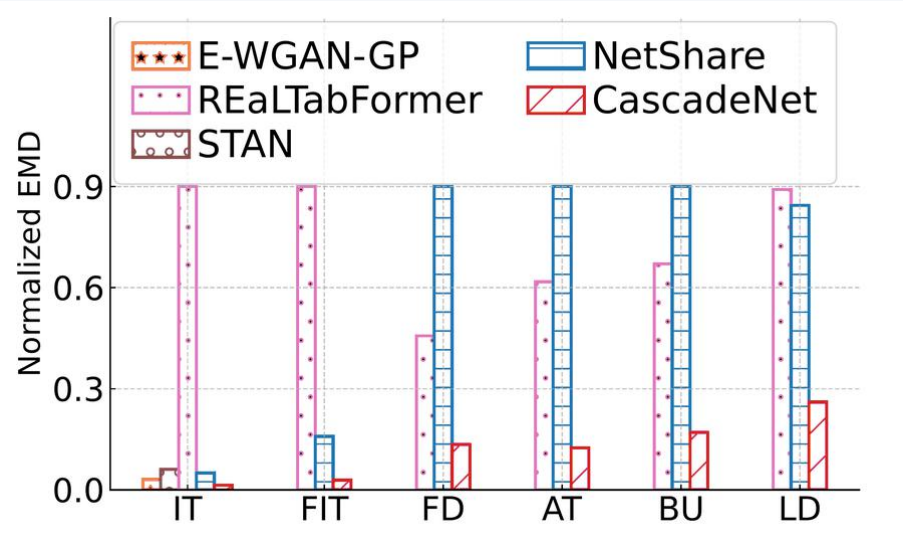
Record uniqueness, training & generation time

Temporal Fidelity: 41%~76% Improvement

Normalized EMD (↓) between temporal patterns of real and synthetic traces



(a) CAIDA

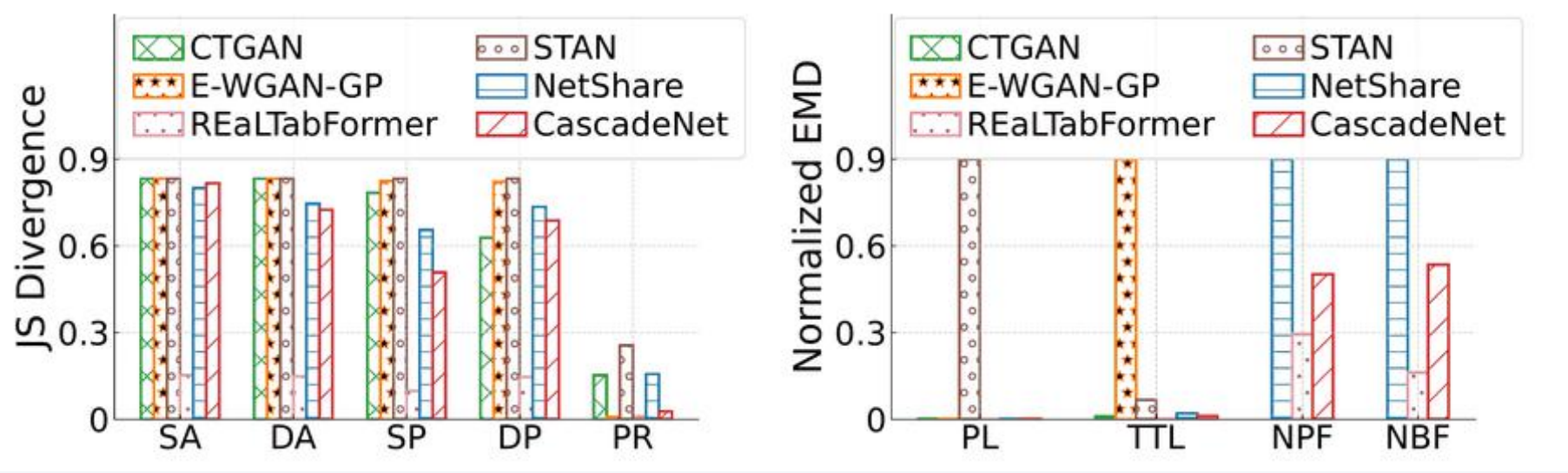


(b) DC

CascadeNet achieves 41%~76% improvement on all 6 temporal metrics

vs. baselines (excl. REaLTabFormer which memorizes 11~57% of training data). Consistent across all 4 datasets.

Statistical Fidelity: Comparable to Baselines

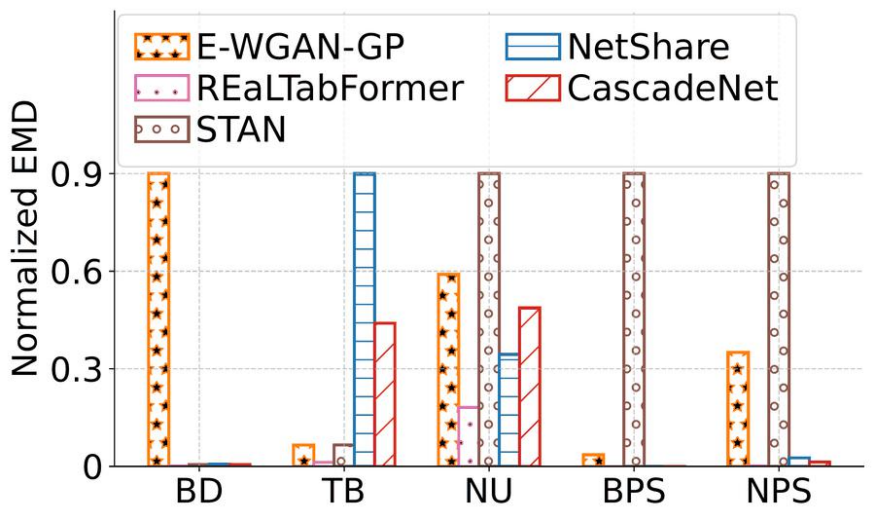


CAIDA — JSD (↓) for categorical features; Normalized EMD (↓) for numerical features

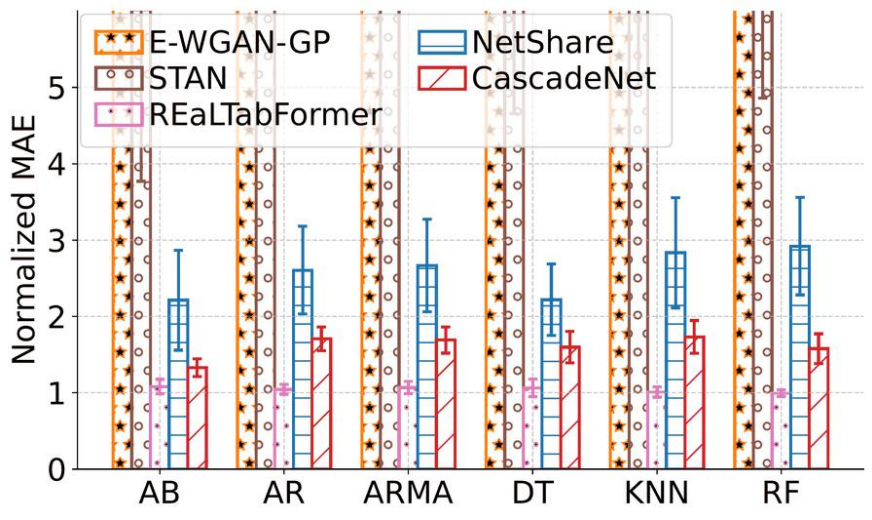
CascadeNet performs comparably to baselines on all 9 statistical features

Our focus on temporal patterns does not come at the cost of statistical accuracy.
SA, DA, SP, DP, PR (JSD) and PL, TTL, NPF, NBF (EMD) — competitive across all datasets.
Results for DC, CA, and TON_IoT in Figures 7b and 13 (Appendix A).

Downstream: Burst Analysis & Throughput Prediction



Burst Analysis — CAIDA (Fig 8a)



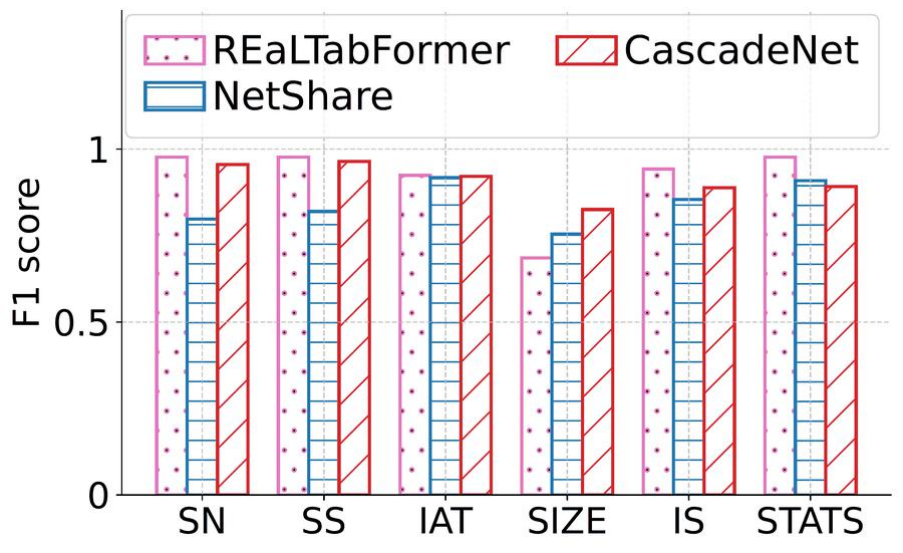
Throughput Prediction — CAIDA (Fig 9a)

Burst Analysis: Best or near-best on all 5 burst metrics (BD, TB, NU, BPS, NPS).

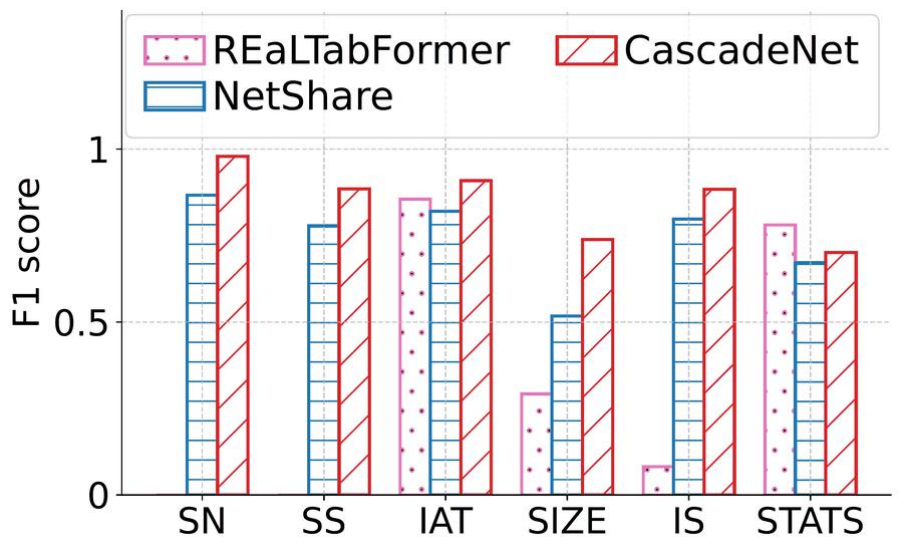
Throughput Prediction: Lowest normalized MAE across 6 ML models (AR, ARMA, KNN, DT, RF, AB).

CascadeNet's temporal fidelity directly translates to more reliable downstream results.

Downstream: ML-Based Anomaly Detection



(a) CAIDA — OCSVM (Fig 10a)



(b) DC — OCSVM (Fig 10b)

Higher F_1 than NetShare across most flow representations (IAT, SIZE, IS, STATS, SN, SS).

Consistent across 6 anomaly detectors: OCSVM, AutoEncoder, GMM, IForest, KDE, PCA (Appendix J).

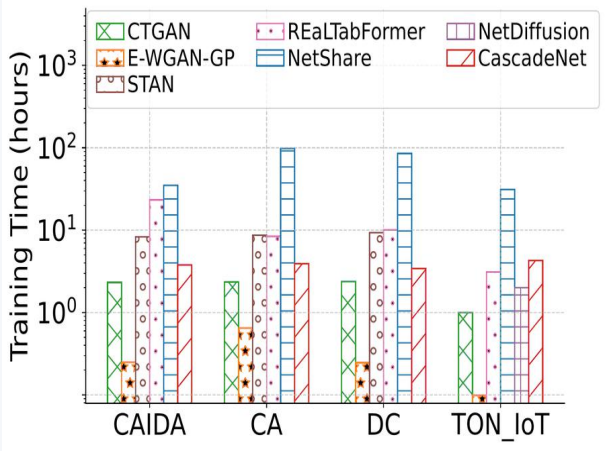
Diversity & Scalability

Diversity

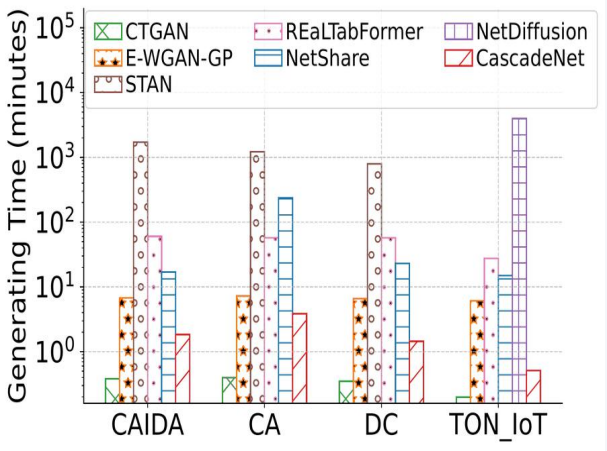
99.9%+

records differ from training data

REaLTabFormer memorizes
1.7~56.5% of training records



(a) Training Time



(b) Generating Time

7.3×~25× faster training
9.2×~62× faster generation
vs. NetShare (state-of-the-art)

Why so fast? Aggregated time series have fixed, short lengths (≤200) regardless of packet count.

NetShare must process elephant flows with 10K+ record sequences — inherently much slower.

Conclusion

Novel data representation:

Aggregate flows into time series → explicitly expose temporal patterns to ML models

CascadeGAN:

Hierarchical model (3G + 3D) learns IP 5-tuples, time series, record fields from coarse to fine

High fidelity:

41%~76% better on temporal metrics; comparable statistical fidelity; superior downstream performance

Scalable:

7.3×~25× faster training, 9.2×~62× faster generation vs. NetShare

Code: github.com/lurw2000/CascadeNet

Thank you! Questions?