Practical GAN-based Synthetic IP Header Trace Generation using NetShare

Yucheng Yin
Carnegie Mellon University
Pittsburgh, PA
yyin4@andrew.cmu.edu

Zinan Lin
Carnegie Mellon University
Pittsburgh, PA
zinanl@andrew.cmu.edu

Minhao Jin
Carnegie Mellon University
Pittsburgh, PA
minhaoj@andrew.cmu.edu

Giulia Fanti
Carnegie Mellon University
Pittsburgh, PA
gfanti@andrew.cmu.edu

Vyas Sekar
Carnegie Mellon University
Pittsburgh, PA
vsekar@andrew.cmu.edu

Presented for UO CS607 on 2023-01-10 by Chris Misa

Problem Formulation

- Traces (i.e., lists of what traffic was observed passing through a network) are critical for many networking tasks.
  - We'll consider *packet-level* and *flow-level* traces.
  - In particular, **lack of access to high-quality data is a road block for traffic monitoring system design, ML-based event/anomaly detection, etc.**
- *The people with interesting traces will never share them directly.*

	Fidelity	Flexibility	Privacy	Effort
Raw	High	✗	✗	Low
Anonymized	Depends	✗	Depends	Low
Synthetic	Possible	High	Possible	High

Table 1: Trade-offs for data holders sharing Raw vs. Anonymized vs. Synthetic traces

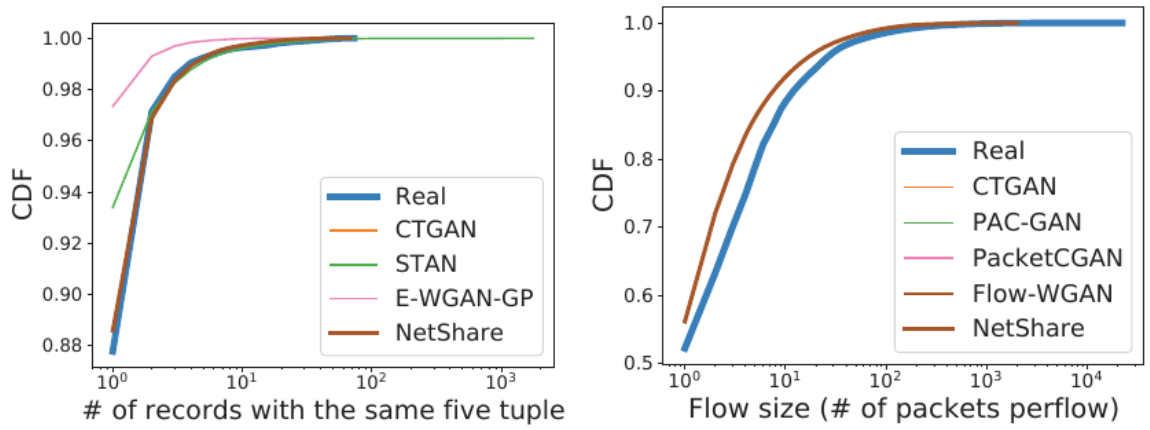
Problem Formulation

- Given: unsampled trace of IPv4 headers.
- Goal: learn a generative model that satisfies *fidelity* metrics.
  - ***Header-level*** distributional properties like popularity of distinct values.
  - ***Flow-level*** properties like flow size or flow duration.
  - ***Use-case specific*** properties like accuracy preservation or order preservation.

- *(Later)* If we can do this, then we can get people with interesting traces to train these models and use them to generate new traces with same properties but no private information so they'll be more comfortable sharing them.
  - *(Don't share the model cause we really don't know what it's learned!)*

What are the technical challenges?

Challenge 1 (C1): ***Baselines do not accurately capture header correlations of packets/flows, e.g., flow length.***

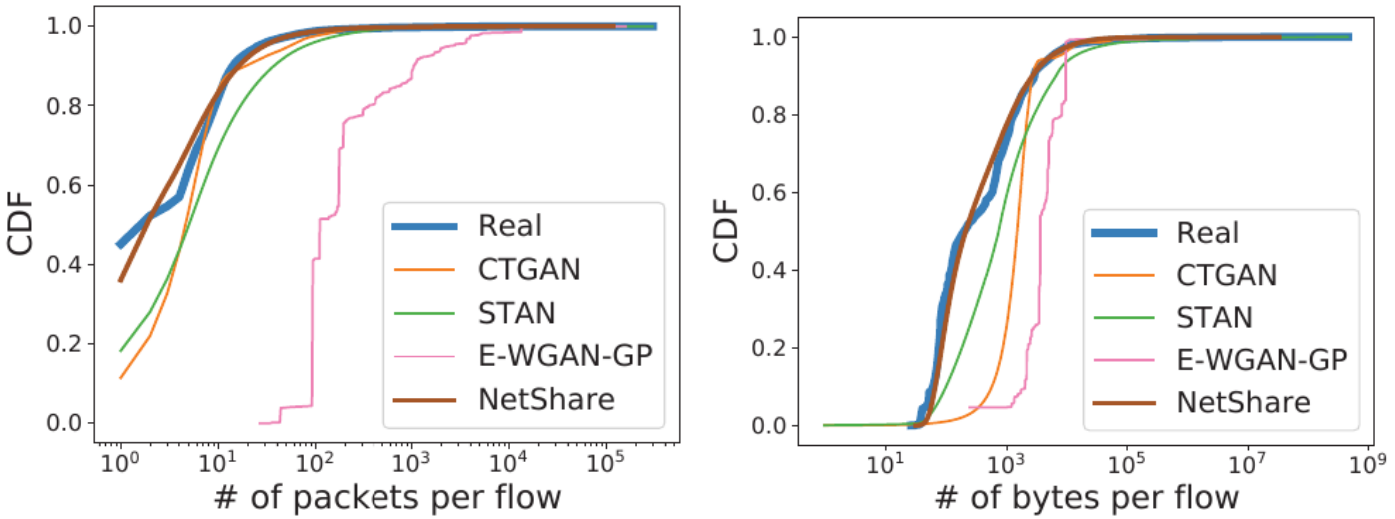


(a) CDF of NetFlow records with same five tuples (UGR16).

(b) CDF of flow size (# of packets) on CAIDA.

Figure 1: Distribution of # of records/packets with the same five tuples on UGR16 (NetFlow, *left*) and CAIDA (PCAP, *right*). All baselines are missing in Fig. 1b as they don't generate flows with > 1 packet.

Challenge 2 (C2): ***Baselines struggle to accurately capture the distributions for fields with large support.***

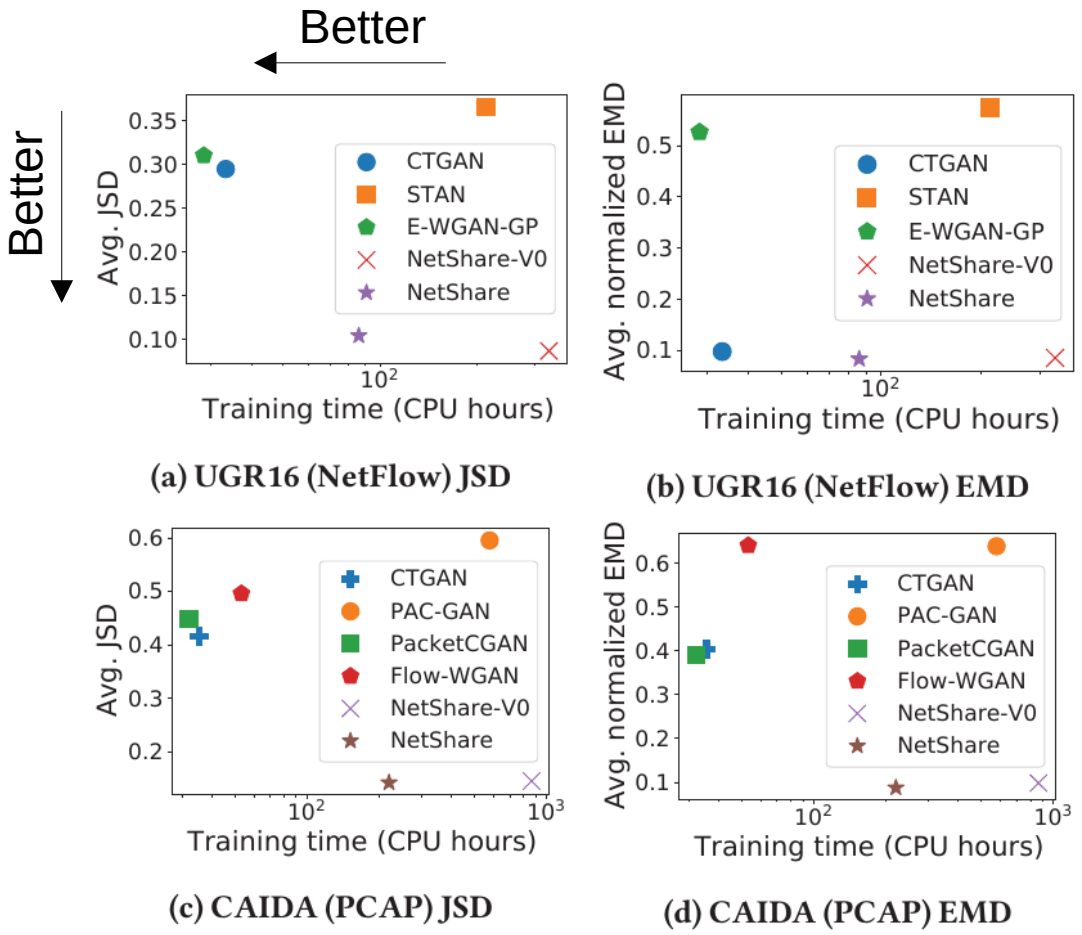


(a) # of packets per flow **(b) # of bytes per flow**

Figure 2: Distribution of NetFlow's (unbounded) fields on UGR16 dataset: *left*: flow size; *right*: flow volume.

Similarly, baselines fail to reproduce prevalence of common service L4 ports (eg., 53, 80, 443, etc.) seen in real data, but of course NetShare does.

Challenge 3 (C3): *Existing GAN-based frameworks exhibit poor scalability-fidelity tradeoffs on network traces.*



(a) UGR16 (NetFlow) JSD

(b) UGR16 (NetFlow) EMD

(c) CAIDA (PCAP) JSD

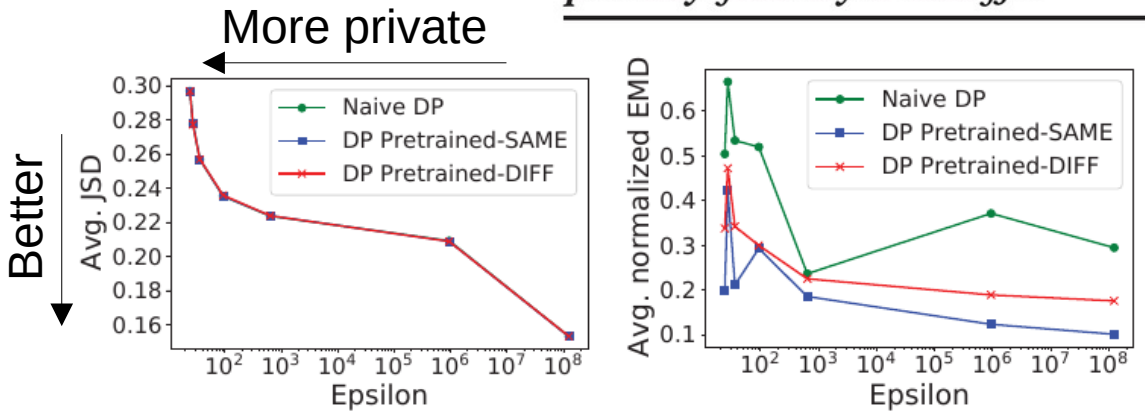
(d) CAIDA (PCAP) EMD

Figure 4: Scalability-fidelity trade-offs: Scalability is measured with total CPU hours (↓) and fidelity is measured with the average JSD across categorical fields and the average normalized EMD across continuous fields (↓).

JSD: *Jensen-Shannon Divergence*, a metric for comparing similarity of (categorical) probability distributions (based on relative entropy).

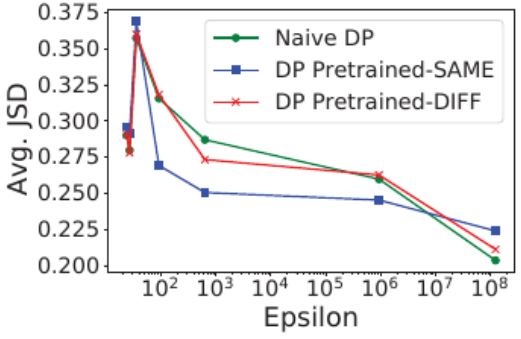
EMD: *Earth Mover's Distance*, a metric for comparing similarity of continuous probability distributions (based on difference of CDFs).

Challenge 4 (C4): *Existing frameworks exhibit poor privacy-fidelity tradeoffs.*

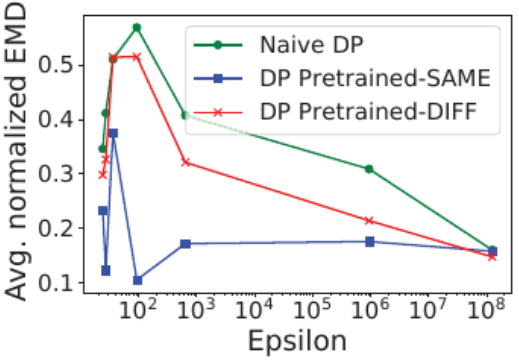


(a) NetFlow (UGR16) JSD

(b) NetFlow (UGR16) EMD



(c) PCAP (CAIDA) JSD



(d) PCAP (CAIDA) EMD

Figure 5: Privacy-fidelity trade-offs: Privacy is measured with (ϵ,δ) in DP (↓) and fidelity is measured as average JSD across categorical fields and the average normalized EMD[1] across continuous fields (↓).

Naive DP means using DP-SGD for all training.

(Other lines are NetShare's modifications explained later.)

DP-SGD: *differentially-private stochastic gradient descent*, a popular variant of SGD that messes with the gradients a bit to limit the amount of privacy loss in a controlled manner.

How does NetShare address these challenges?

Insight 1 (I1): *We reformulate header trace generation as a time series generation problem of generating flow records for the entire trace rather than a per-epoch tabular approach (Figure 6).*

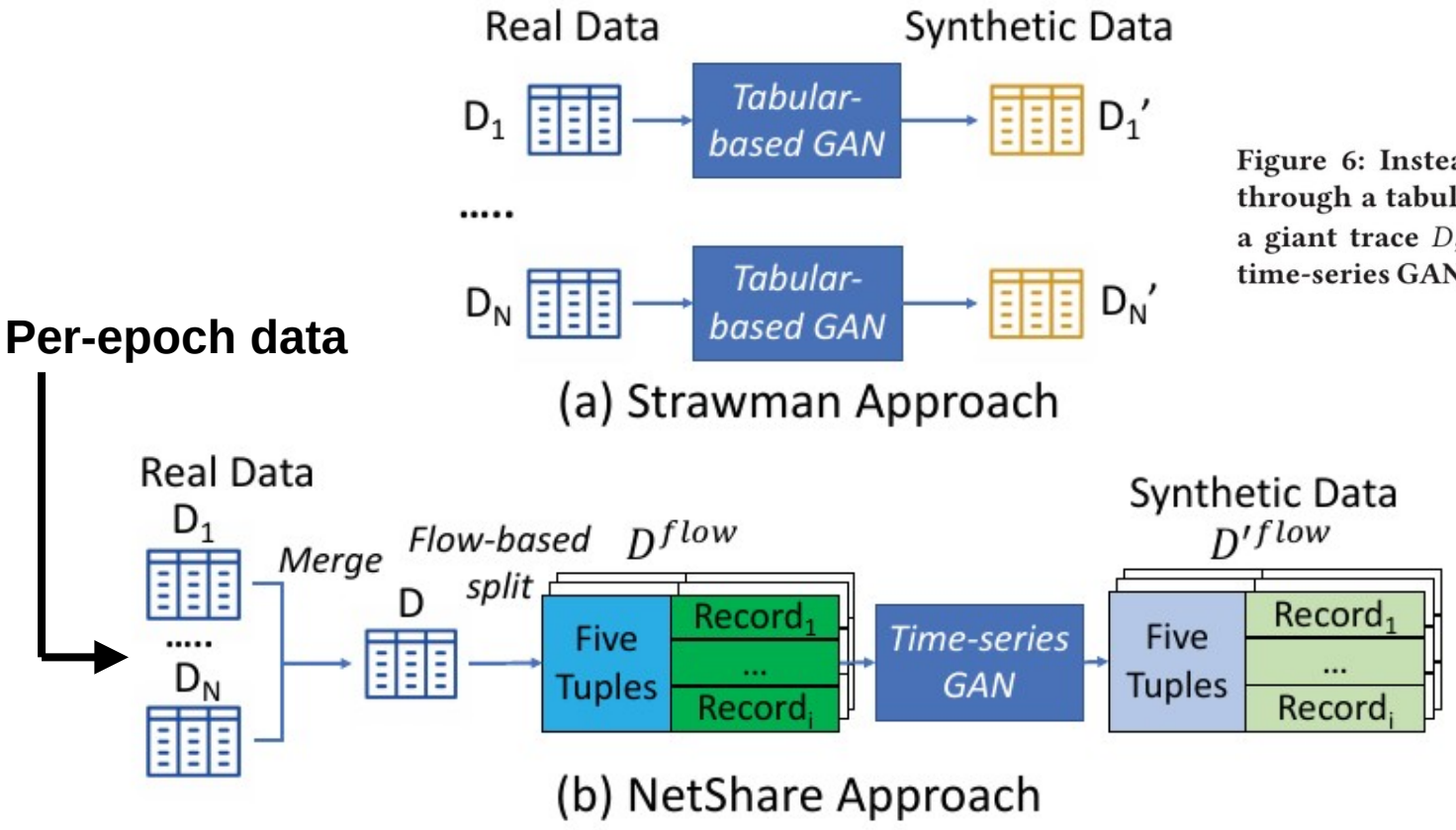


Figure 6: Instead of generating measurement epochs D_i through a tabular GAN, we *merge* multiple epochs D_i into a giant trace D, split the trace into *flows* D^{flow}, and use time-series GAN.

Insight 2 (I2): ***We use a careful combination of domain knowledge and machine learning to inform the representation of header fields to balance fidelity-privacy-scalability tradeoffs (Table 2).***

* *Log-transform* for packets/bytes per flow.

* *Word2Vec* type thing (from prior work called *IP2Vec*) for ports.
    → Solves the problem of the model not “understanding” semantics of well-known service ports.
    → Forms a dictionary of ports from large CAIDA trace assuming all relevant ports are seen.
    → (Implicit point) Non-service ports are typically chosen at random and not considered a leak of private information.

* *Bitwise encoding* of IP addresses.
    → Can’t use Word2Vec type thing because it’s dictionary based to the IPs would leak.

Insight 3 (I3): ***We can improve the scalability-fidelity tradeoff via fine tuning and parallel training (Fig. 7).***

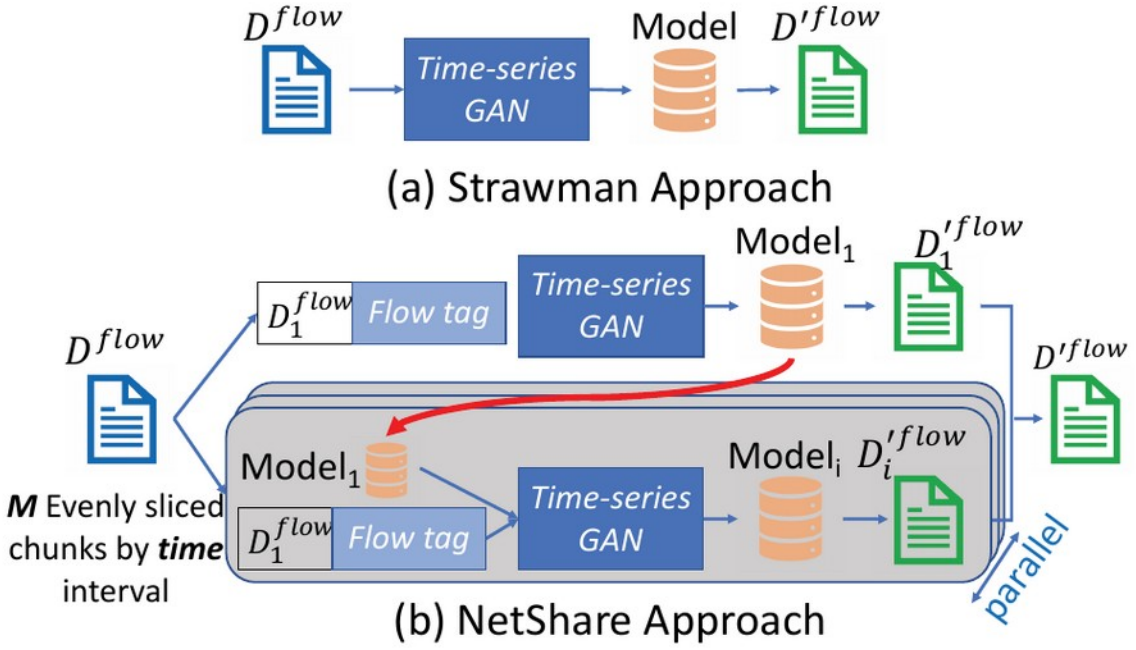


Figure 7: We split D^{flow} into M evenly *time-spaced chunks* with explicit "flow tags" to capture cross-chunk correlations. We use the first chunk as a pre-trained model for parallel training of later chunks.

Flow tag: a binary vector with info about which chunk/epoch the flow is active in (i.e., one bit per epoch).

(Does anyone understand why this is expected to "capture the inter-chunk correlation"?)

Insight 4 (I4): ***We can improve privacy-fidelity tradeoffs by carefully using public datasets (Fig.8).***

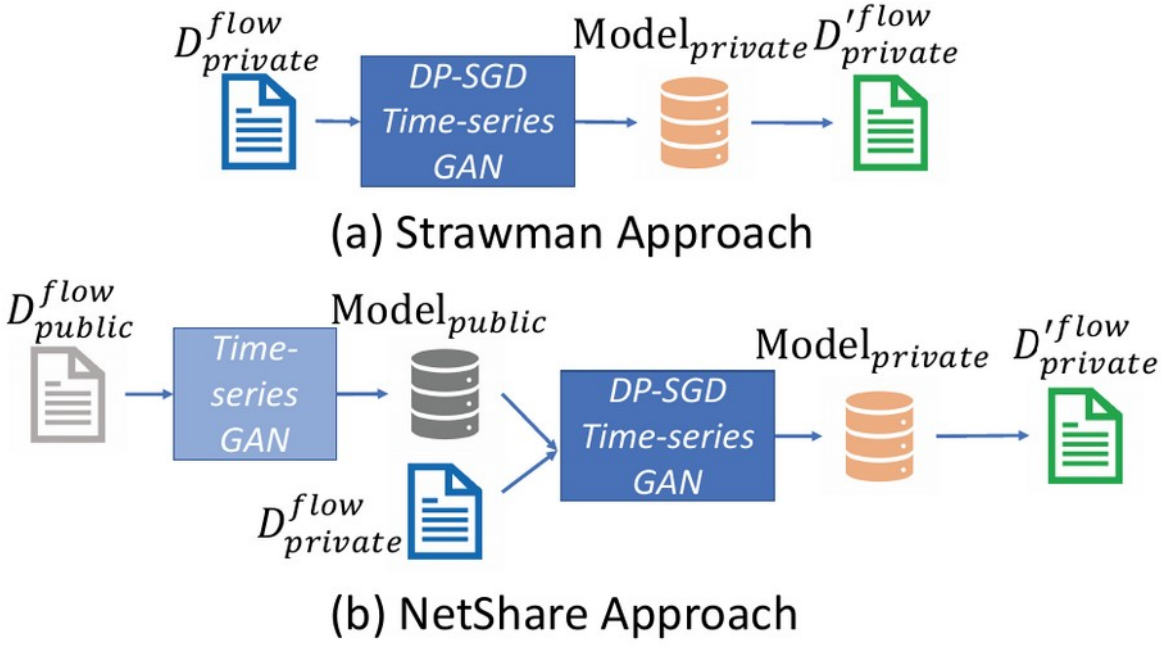


Figure 8: We use public traces to pre-train a public model Model_{public}, then fine-tune on private data.

How do we know NetShare actually works?

Finding 1: ***NetShare achieves 46% better fidelity than baselines on feature distribution metrics across traces.***

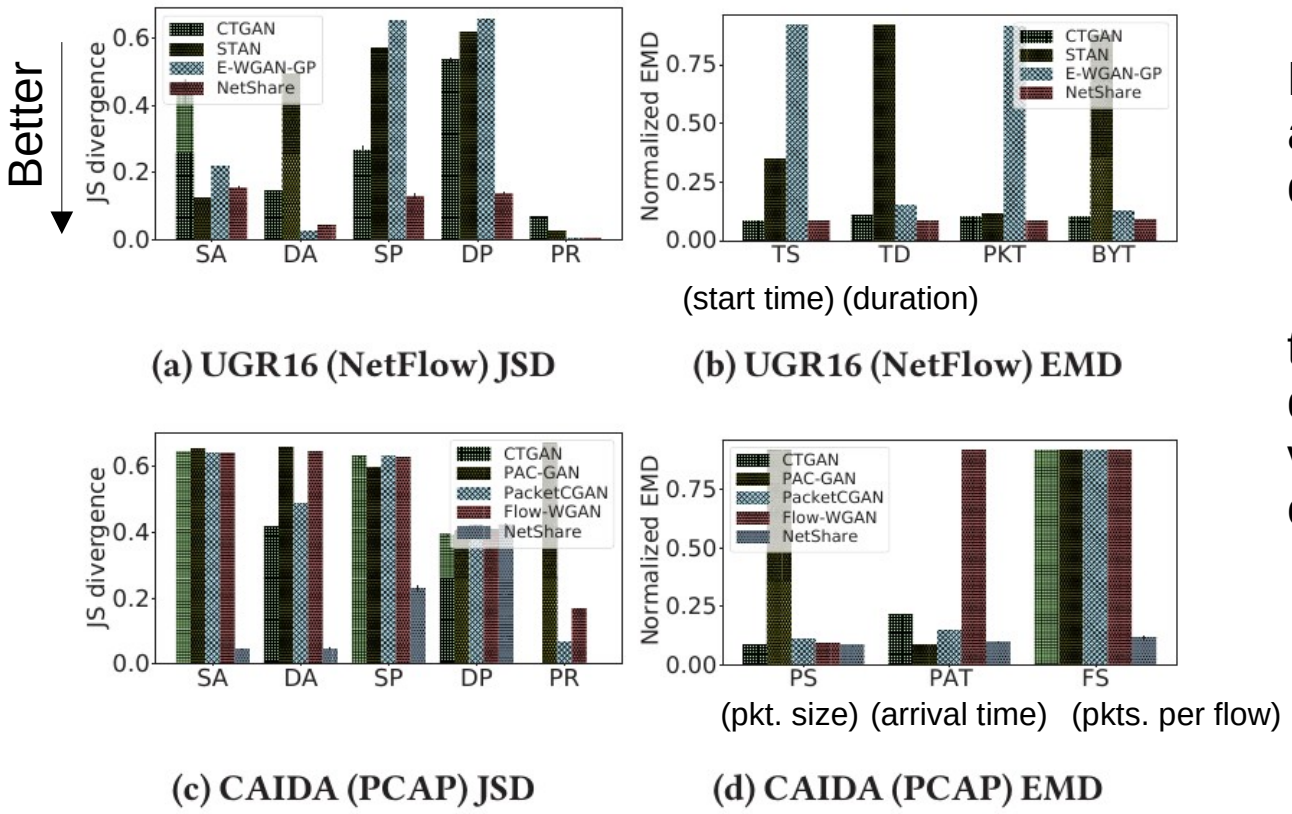


Figure 10: Jensen-Shannon divergence (↓) and *normalized* Earth Mover's Distance (EMD) (↓) between real and synthetic distributions on UGR16 (NetFlow) and CAIDA (PCAP).

Note these are all still based on distributions…

…how many times each distinct field value is observed.

Finding 2: ***NetShare provides better fidelity for downstream network management tasks across different traces.***

Example application: traffic type prediction from labeled netflow data

Comparison between:
(i) train on real, test on real and
(ii) train on synthetic, test on real.

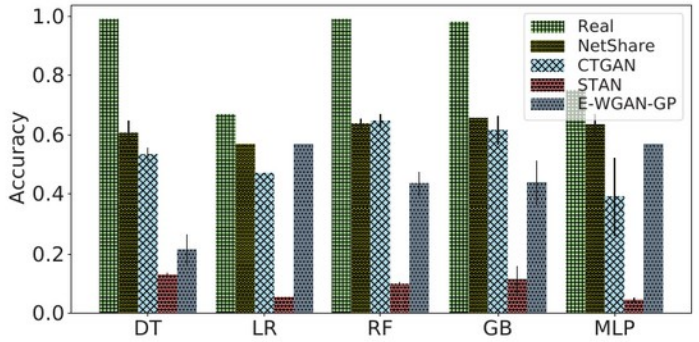


Figure 12: NetFlow traffic type prediction accuracy (↑) on TON: *all* classifiers achieve the highest accuracy with synthetic data generated by NetShare.

Comparison between:
(i) train on real, test on real and
(ii) train on synthetic, test on synthetic.

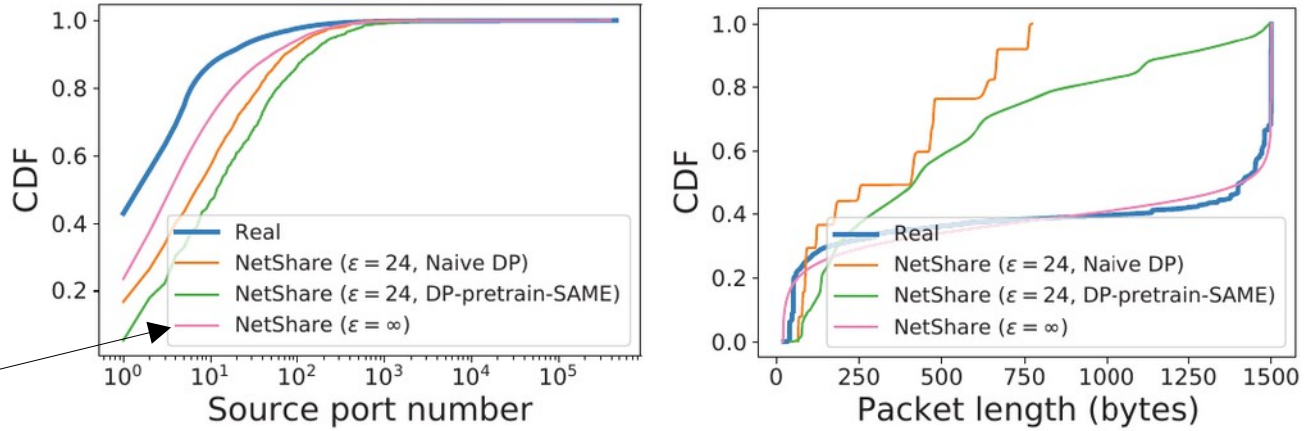
Table 3: Rank correlation (↑) of prediction algorithms on CIDDS and TON. Higher is better.

	NetShare	CTGAN	STAN	E-WGAN-GP
CIDDS	**0.90**	0.60	0.60	0.70
TON	**0.70**	0.10	0.60	-0.60

Other examples: ranking sketch algorithms, header-based anomaly detection with NetML.

Finding 3: *Pre-training NetShare on public data can improve the fidelity of differentially-private traces.*

No added noise,
i.e., no privacy.



(a) Source port **(b) Packet length (bytes)**

Figure 15: Packet length and port CDFs computed without noise and under the same (ϵ,δ) with or without pre-training.

...still sort of an open problem.

Finding 4: ***NetShare achieves a better scalability-fidelity trade-off than baselines.***

(Already shown!)

… that’s it!

Discussion Questions