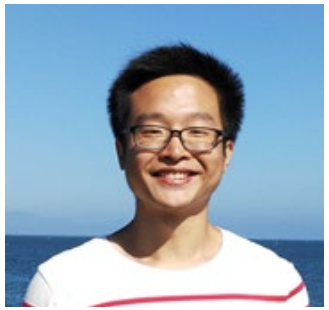
Practical GAN-based Synthetic IP Header Trace Generation using NetShare

Yucheng Yin

CMU



Zinan Lin

CMU, MSR

Minhao Jin

CMU

Giulia Fanti

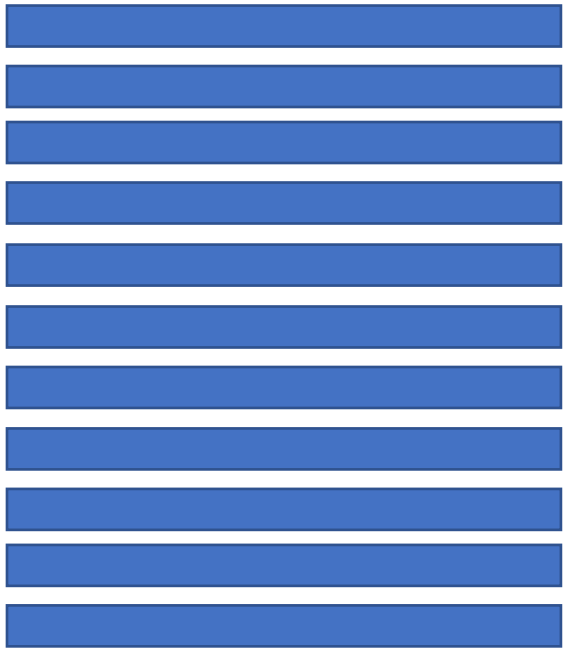
CMU

Vyas Sekar

CMU

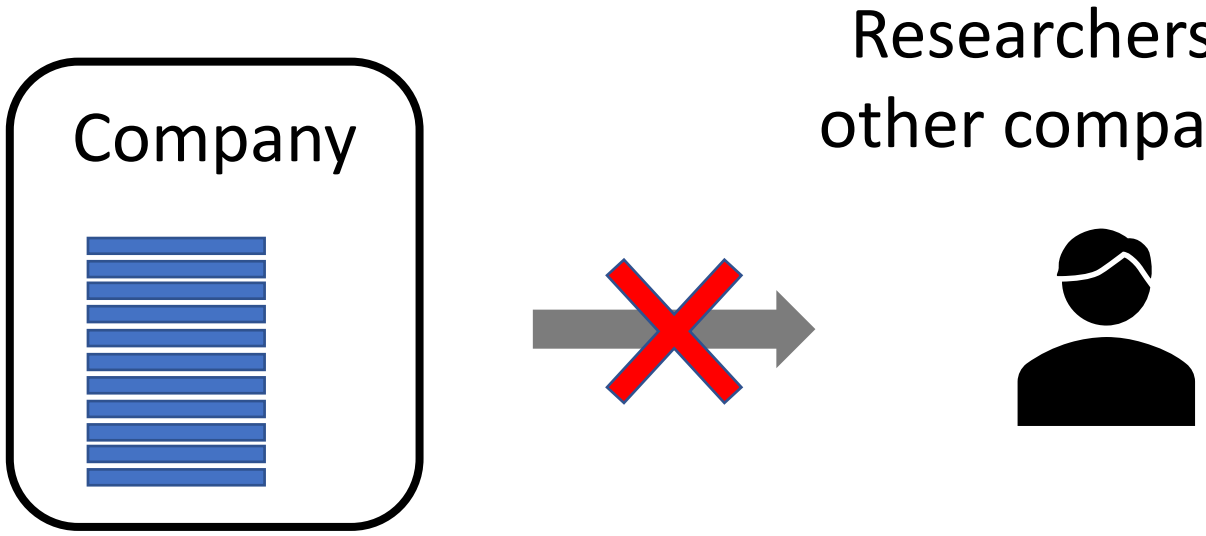
Many Applications of Network Trace Analysis

Network Trace:
Timestamp, Header, Size



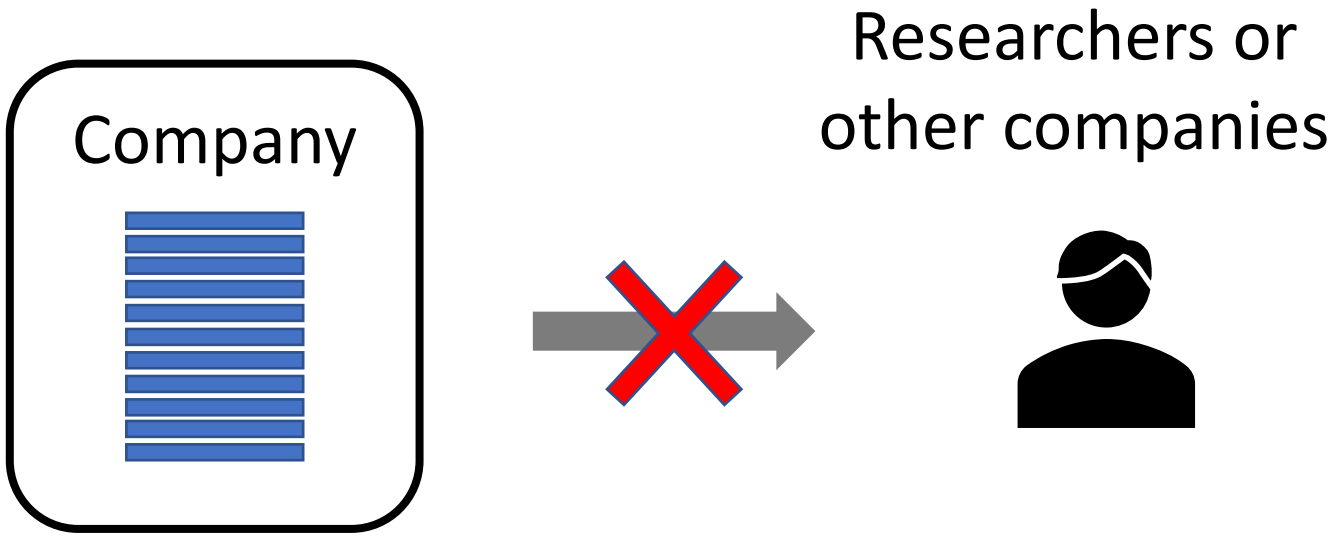
- Structural characterization
- Predictive modeling
- Algorithm evaluation
- Security and resilience
- ….

Pain point for academia: Reproducibility crisis



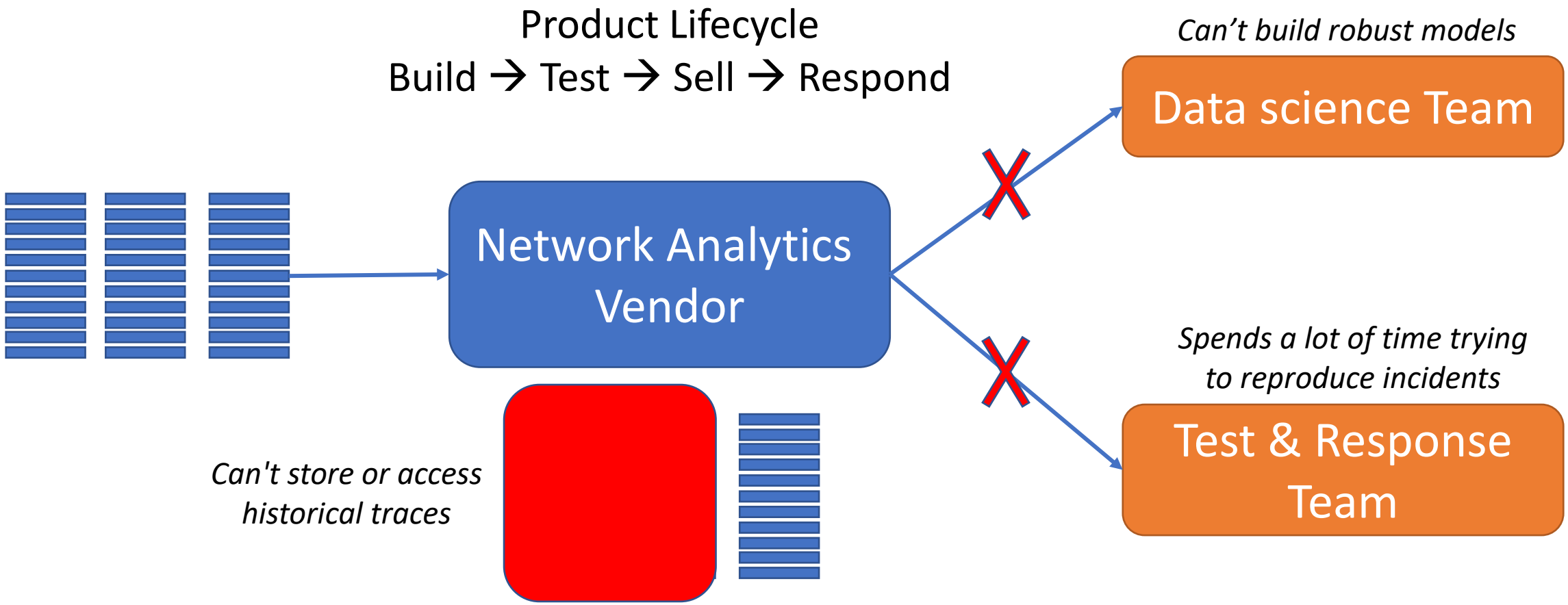
Bad outcomes: Research is not reproducible

Pain point for academia: Collaboration

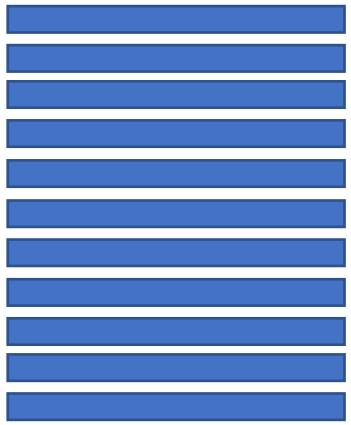


Bad outcomes: Collaborative opportunities go untapped

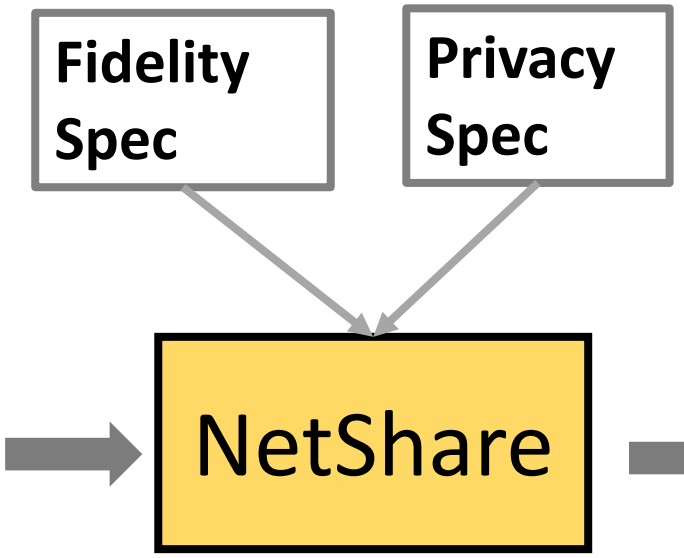
Pain point for product vendors



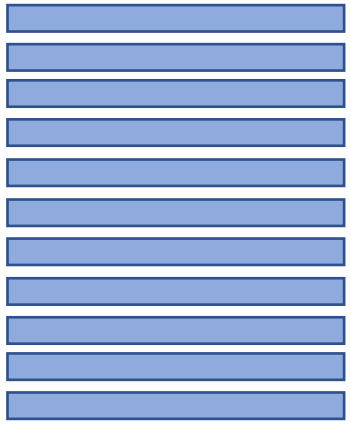
Our Vision: NetShare



IP Header Traces
(e.g., PCAP, NetFlow)



High-fidelity
Privacy-Preserving
Synthetic Header Traces



Low-friction
Backwards-compatible

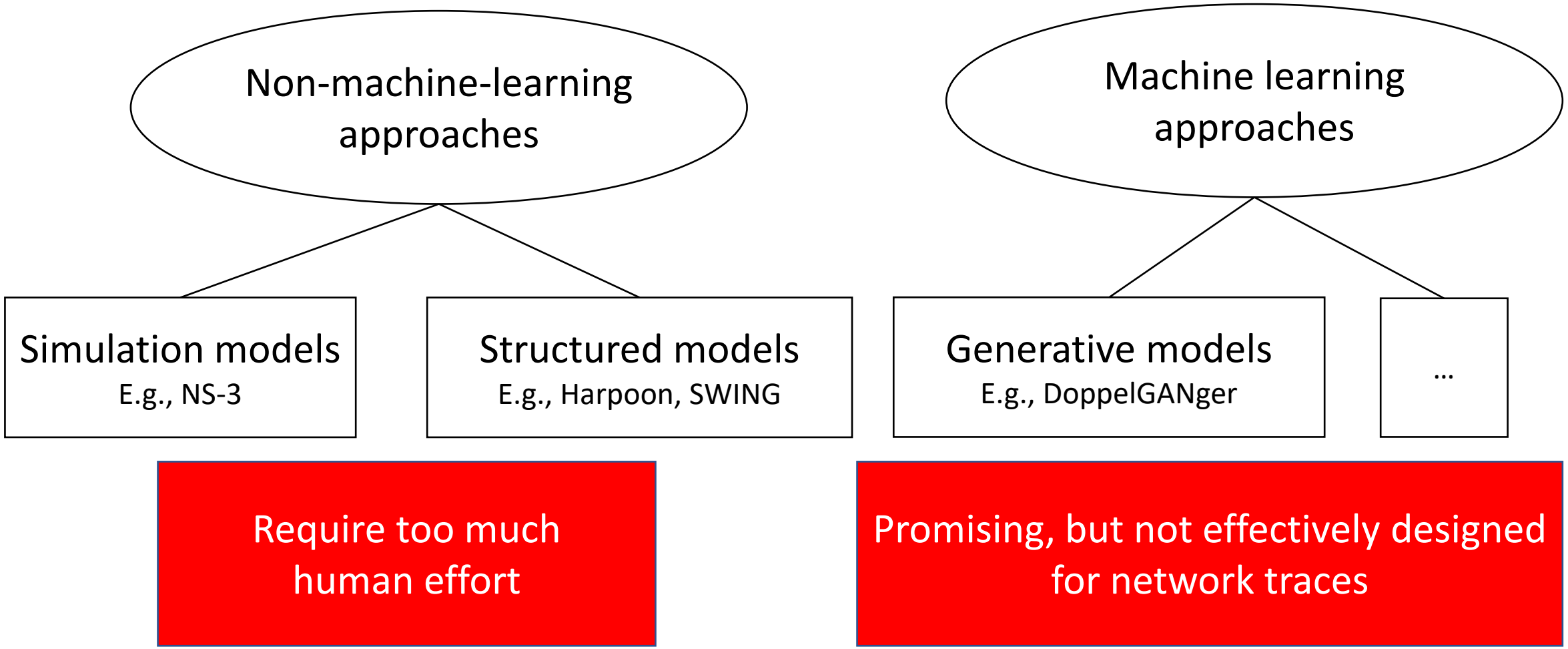
Network
Analytics
Workflows

Robust models
Reduced cost
Faster dev/test time

Outline

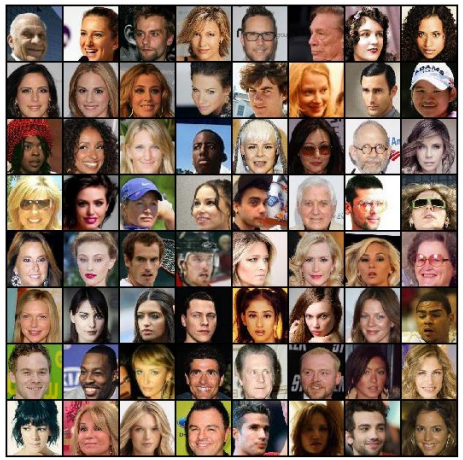
- Motivation
- Background
- Challenges & Ideas
- Implementation & Evaluation
- Limitations and Future Work

Taxonomy of Synthetic Trace Generation



Generative Adversarial Networks (GANs)

Real training images



GANs

Generated synthetic images

Learn high-dimensional correlations with potentially less human expertise

Motivating question

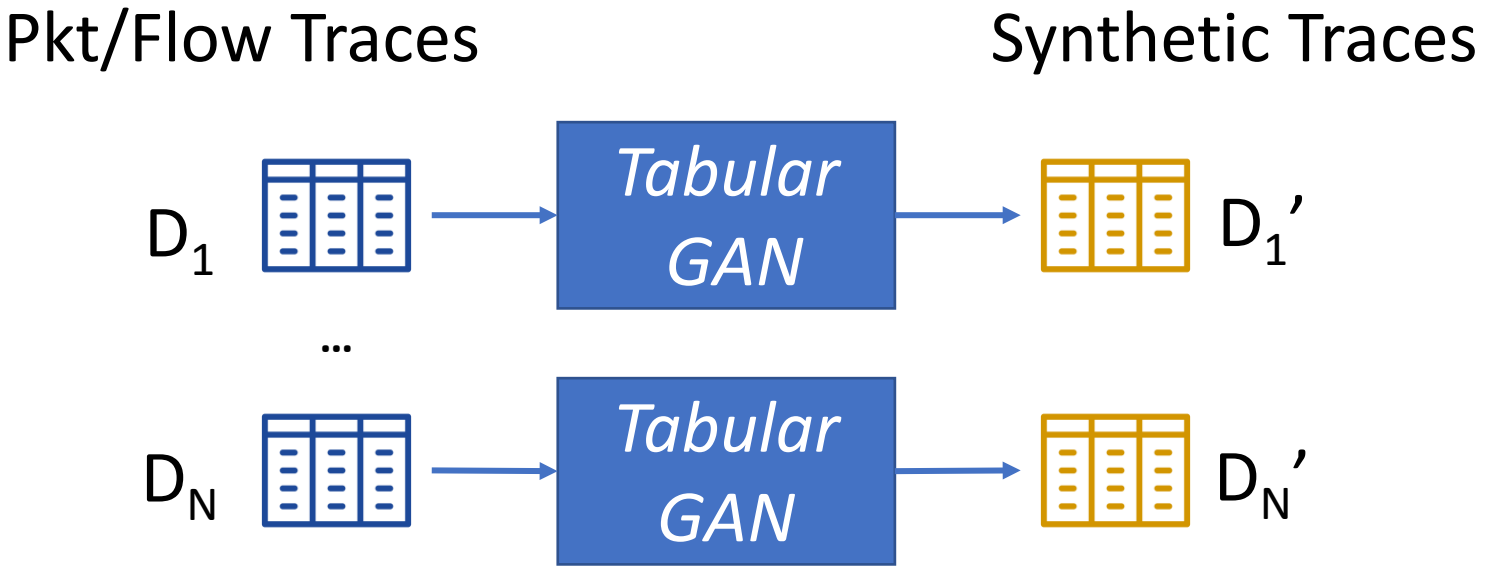
Can we use GANs to generate **synthetic header traces** with good **fidelity-scalability-privacy** tradeoffs?

Outline

- Motivation
- Background
- Challenges & Ideas
- Implementation & Evaluation
- Limitations and Future Work

Seemingly natural stramwan

Idea: Model the trace as a “table” or “CSV” with simple headers of ints, bits etc



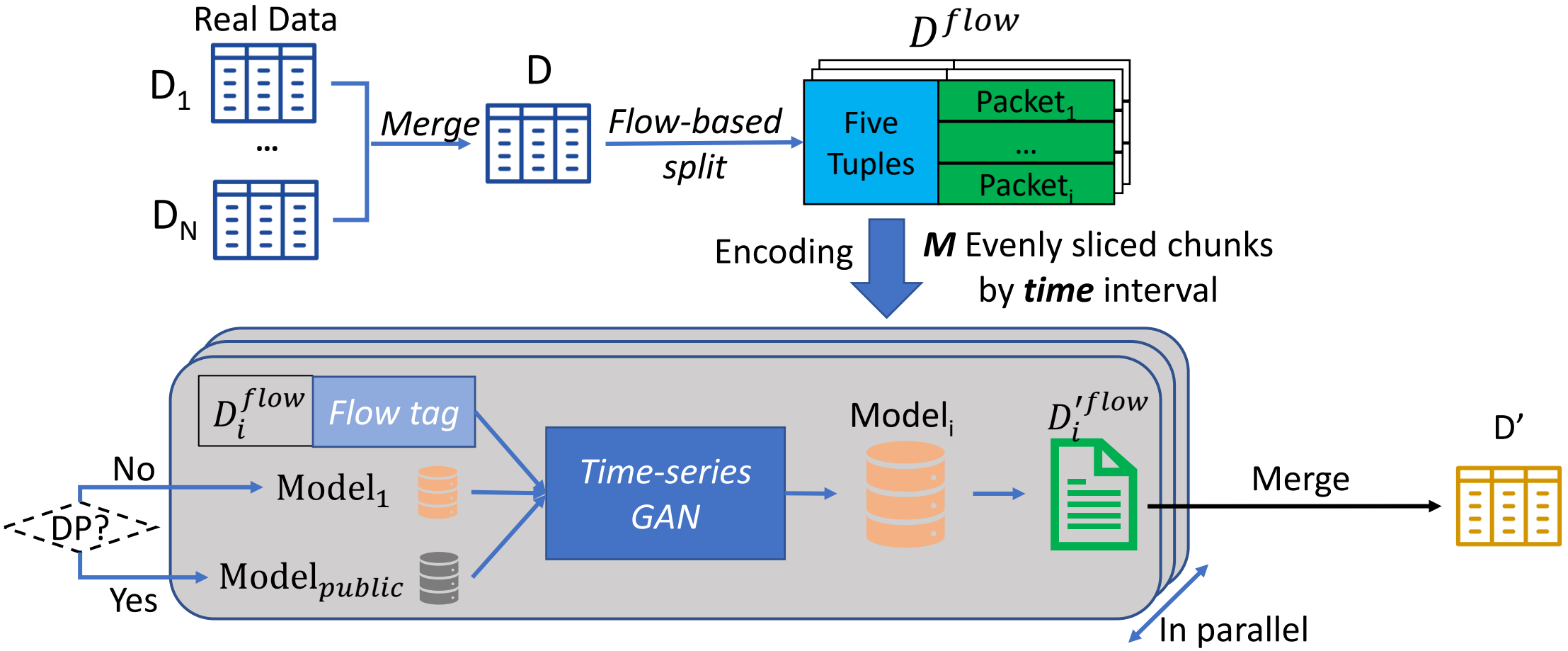
Challenges

- C1: Header field correlations
- C2: Modeling fields with large support
- C3: Scalability
- C4: Privacy-fidelity tradeoffs

NetShare Summary of Key Ideas

- Reformulate as a timeseries problem to capture correlations
- Domain knowledge to improve data encodings
- Use “incremental” fine tuning by chunking the trace
- Using public data to improve Differential Privacy vs. Fidelity tradeoff

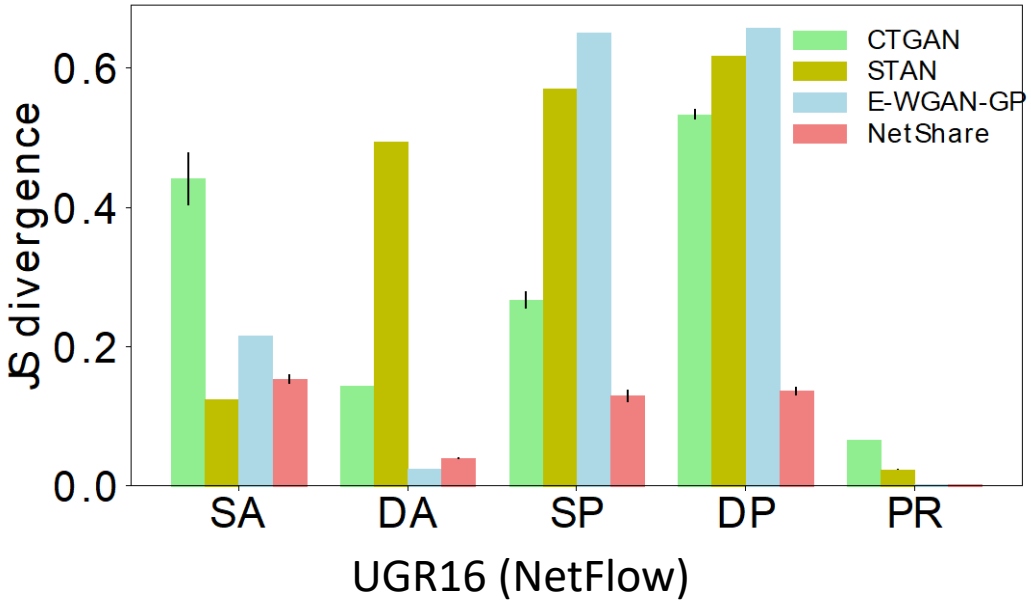
NetShare: End-to-end view

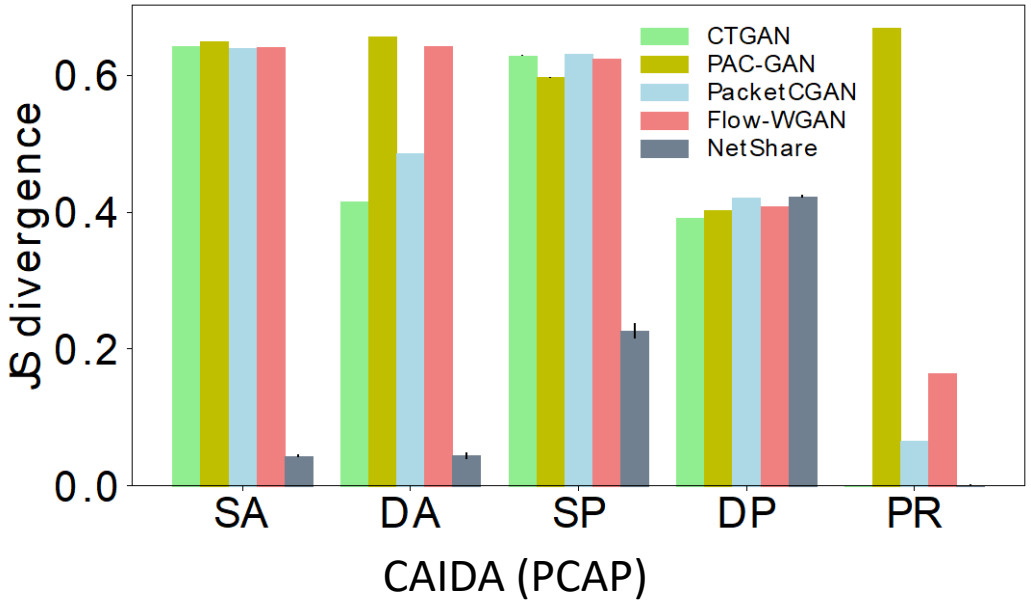


Outline

- Motivation
- Background
- Challenges & Ideas
- Implementation & Evaluation
- Limitations and Future Work

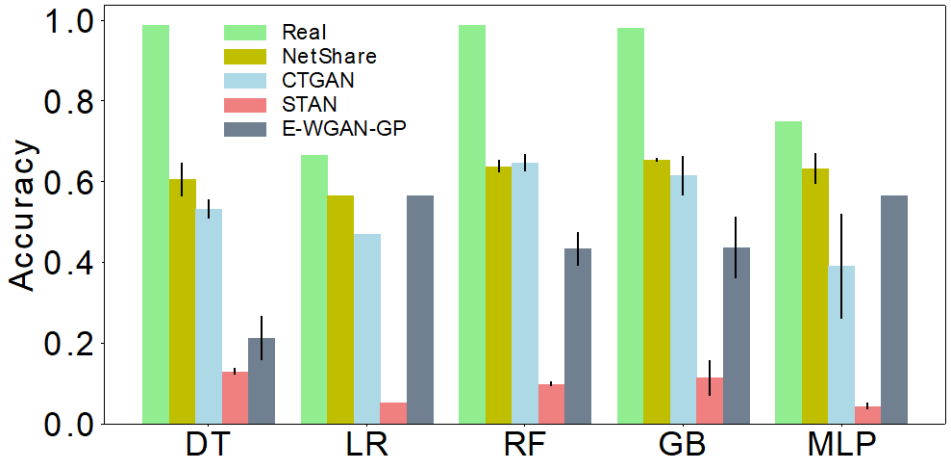
Evaluation – field distribution





NetShare achieves 46% better fidelity than baselines on feature distribution metrics across traces.

App #1: flow-based traffic type prediction



NetShare achieves the best accuracy compared with real data across different classifiers

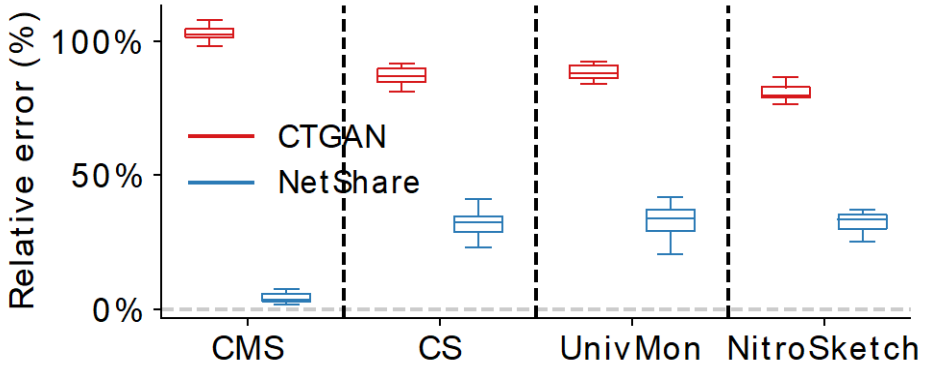
Table 4: Rank correlation of prediction algorithms on CIDDS and TON. Higher is better.

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

NetShare preserves rankings of different classifiers best among all baselines

App #2: sketch-based network telemetry

- Heavy-hitter detection using different sketching algorithms



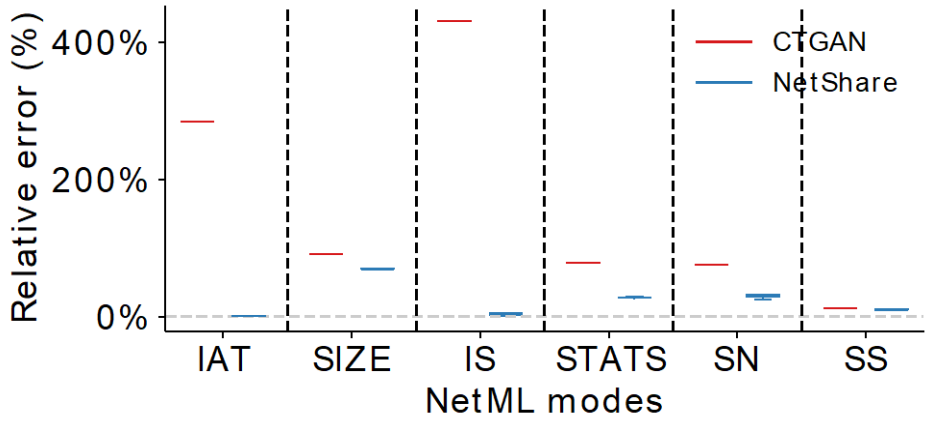
NetShare achieves 48% smaller relative error rate compared with baselines on average

Table 3: Rank correlation of sketching algorithms on heavy hitter estimation of PCAP datasets. Higher is better.

	CAIDA (Dst IP)	DC (Src IP)	CA (Five tuple)
NetShare	**1.00**	**1.00**	**1.00**
CTGAN	0.80	0.40	N/A

NetShare preserves rankings of different sketching algorithms across different datasets/heavy hitters of interest

App #3: Header-based anomaly detection (NetML)



NetShare achieves a lower relative error on most datasets and modes of NetML

Table 4: Rank correlation (↑) of modes of NetML for PCAP anomaly detection.

	NetShare	CTGAN	PAC-GAN	PacketCGAN	Flow-WGAN
CAIDA	**1.00**	N/A	N/A	N/A	N/A
DC	**0.94**	0.43	N/A	N/A	N/A
CA	**0.88**	-0.26	0.37	-0.26	N/A

NetShare preserves rankings of different modes of NetML best among all baselines

NetShare as a Python package

- Installation is easy (one-line): pip3 install –e .
- Example usage

```python
import netshare.ray as ray
from netshare import Generator

if __name__ == '__main__':
    ray.config.enabled = True
    ray.init(address="auto")

    generator = Generator(config="config.json")
    generator.train_and_generate(work_folder='results')

    ray.shutdown()
```

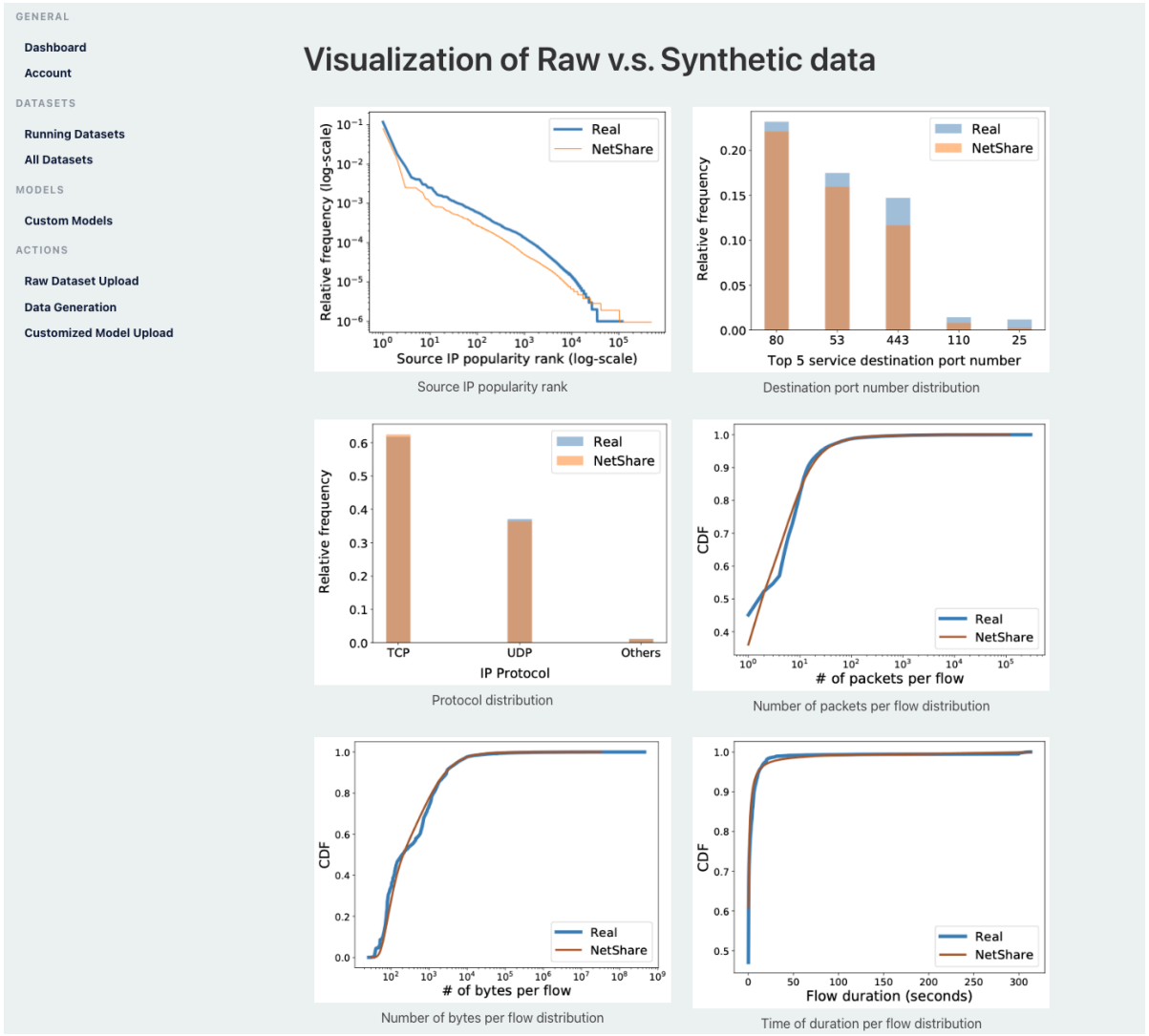
Driver code (~10 lines)

```json
{
    "global_config": {
        "original_data_file": "raw.pcap",
        "dataset_type": "pcap",
        "n_chunks": 10,
        "dp": false
    },
    "default": "pcap.json"
}
```

Configuration files (~10 lines)

https://github.com/netsharecmu/NetShare

NetShare as a (prototype) web service



https://www.pcapshare.com

Limitations and Future Work

- Extending NetShare to support more features
  - E.g., stateful, session-oriented protocols, payload data
- More downstream tasks
  - E.g., QoE inference, device/application fingerprinting
- Improving scalability further?
- What are the relevant privacy properties for our domain?

Takeaways

- Academic and industry pain points in trace-driven analysis in networking:
  Data-driven workflows are stymied by data access, retention policies, data silos
- New Opportunity: Synthetic Network Trace Data Generation via GANs
- NetShare: Feasibility and promise of GANs
  - Robust across datasets, domains, and downstream tasks
- Join us and contribute ☺

https://users.ece.cmu.edu/~vsekar/projects/datafuel/
https://github.com/projectdatafuel/
https://github.com/netsharecmu/NetShare