Using *Graph Neural Networks* for network traffic categorisation

Maarten Meijer

Graduate intern at GÉANT

10 October 2025, Karlsruhe

Public

Who we are?

UNIVERSITY OF TWENTE.

GÉANT

Maarten Meijer
Graduate Intern @ GÉANT
Masters Computer Science student @ UTwente (The Netherlands)

maarten.meijer@geant.org

Guy Roberts
Senior Transport Network Architect @ GÉANT

Doina Bucur
Network data scientist,
Assistant Professor in Computer Science @ UTwente (The Netherlands)

Daniela Brauner
Research Engagement Manager @ GÉANT

Science flows

LHC experiments
data transfers

Satellite data “ground”
distribution

Astronomical
data transfers

Why identify science flows?

- (Inter)national R&E networks carry a large volume of science traffic
- It is useful to understand the nature of the traffic
  - The science **experiment** and network **activity** involved
  - Efficient network use, traffic steering, future provisioning and capacity planning
  - Performance measurement (flows, data rates, …)
  - Site traffic profiling, or traffic accounting for shared inter-continental links
  - Quantify global behaviour and analyse trade-offs at scale
- And not only for NRENs, scientists are also interested in having more info

- HOW?
  - Normal network flows monitoring. But can current tooling achieve this?
    - Yes, in overlay networks VRFs, e.g., LHCONE,
    - No, only for specific communities/experiments; and the others?

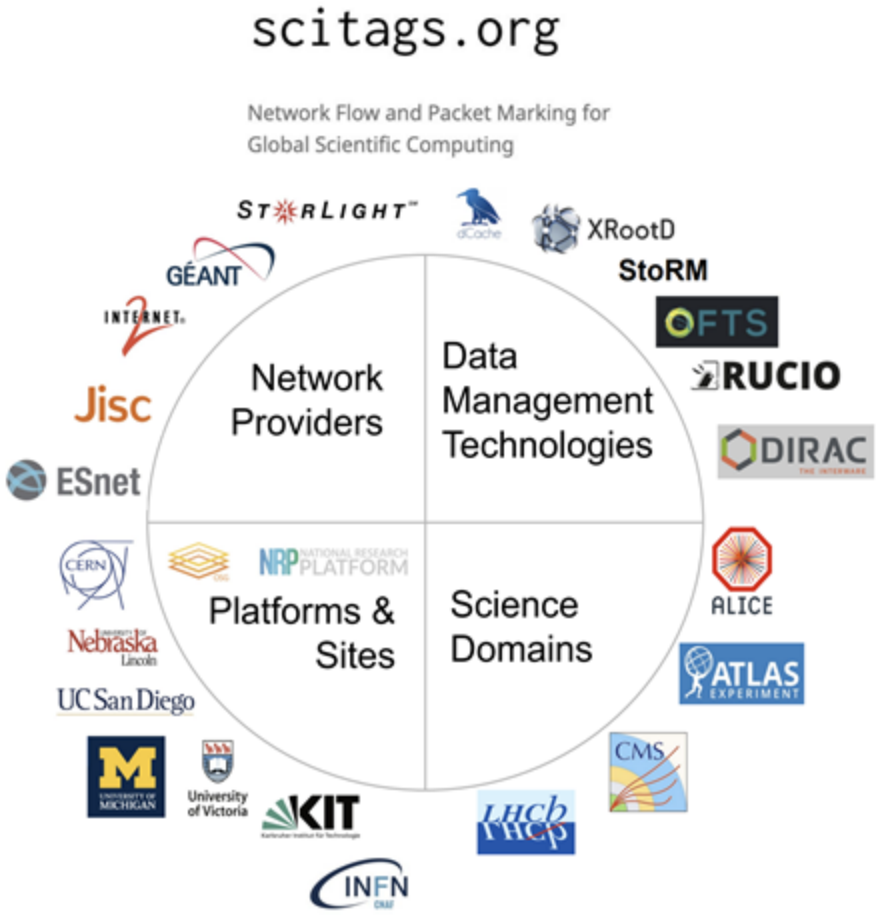
How scientists are doing now?

Marking data packets and flows with experiment and applications IDs for better accounting.

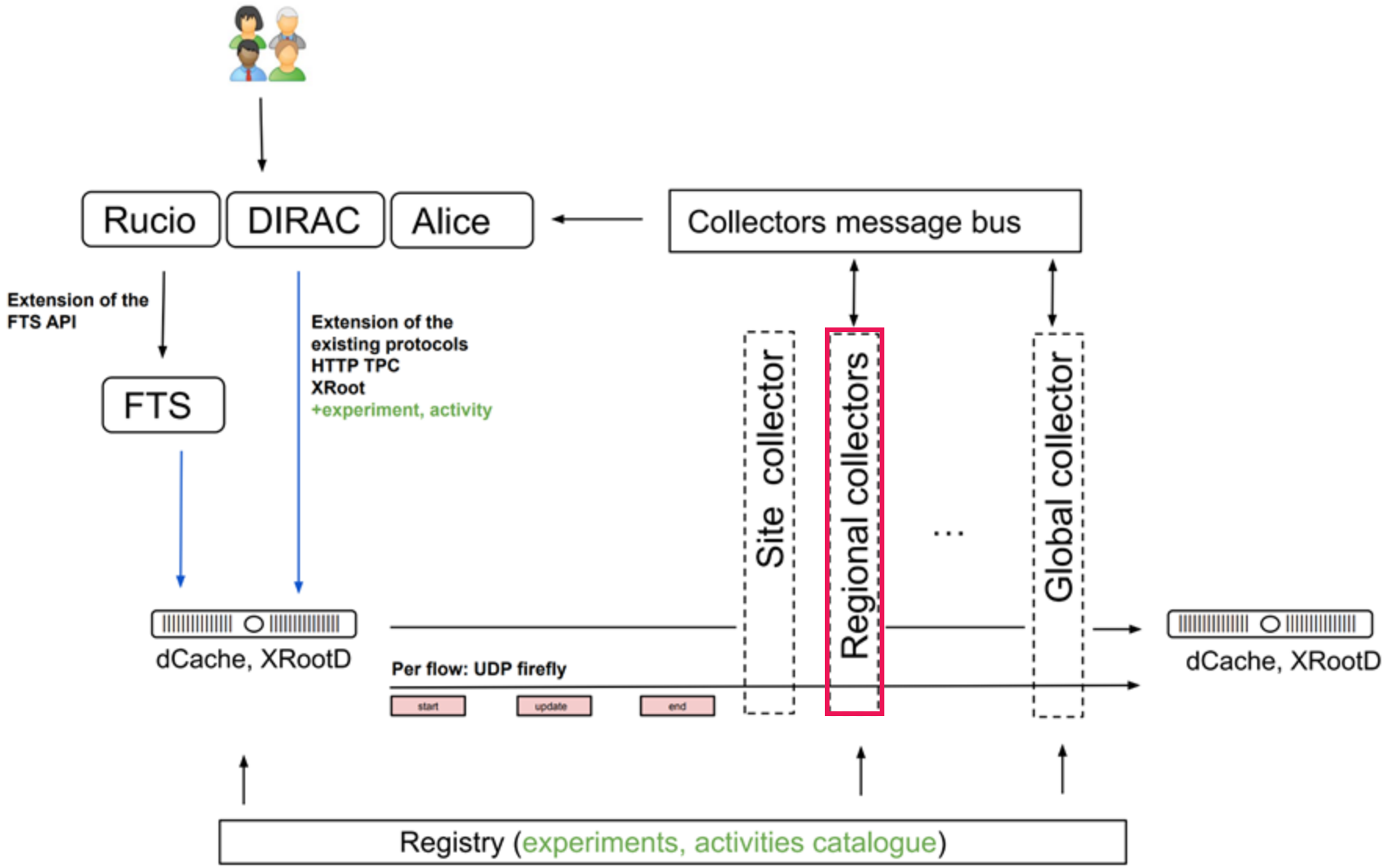
Defining a standard(s) for exchange of information between scientific communities, sites and network operators.

Two options are being pursued:

- Packet marking: encoding experiment/activity directly in packets (tag in the IPv6 flowlabel field)
- Flow marking: ending a separate UDP packet (**firefly**) with metadata (Identifying the experiment and activity of traffic)

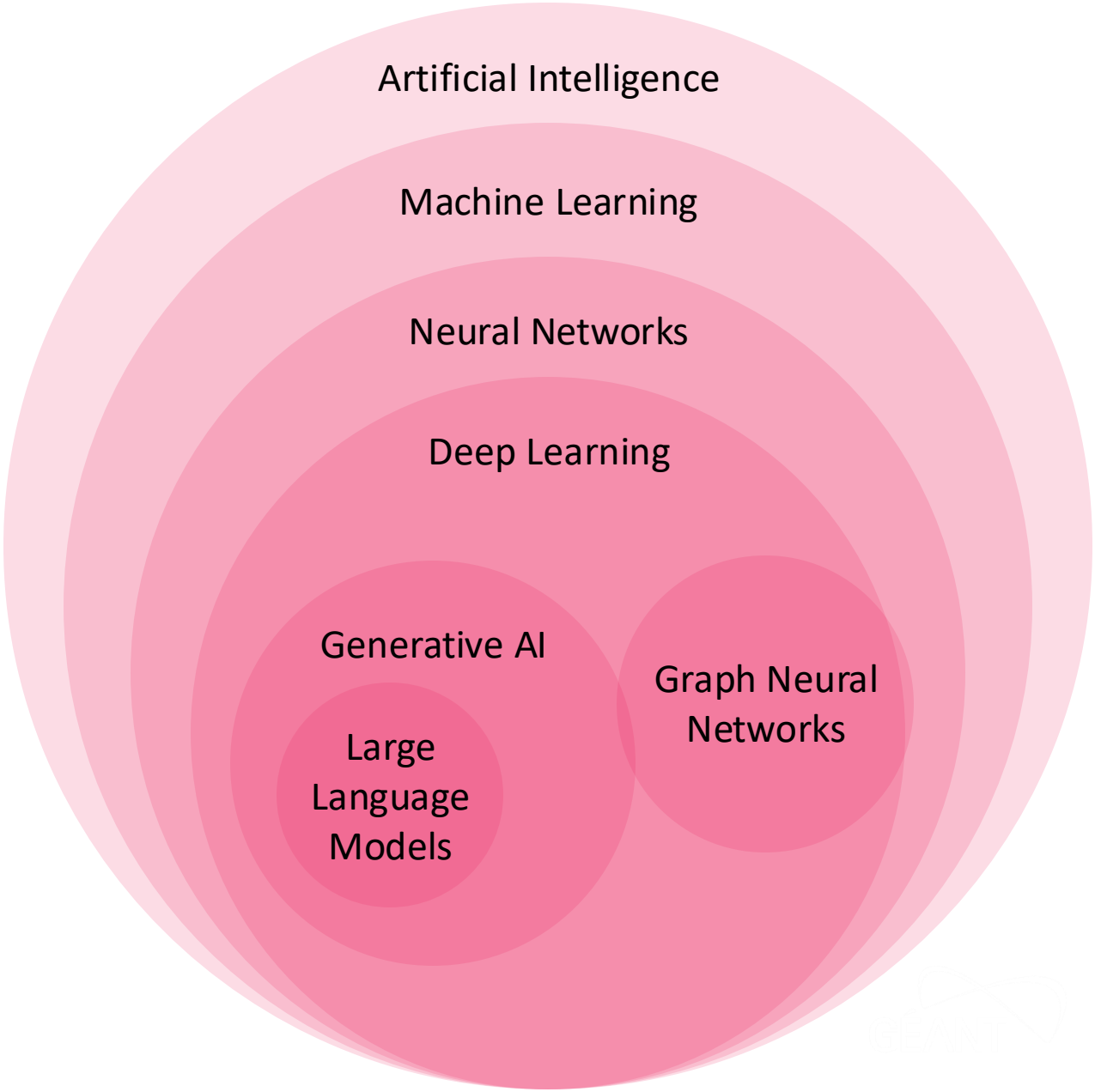


How scitags work - fireflies today

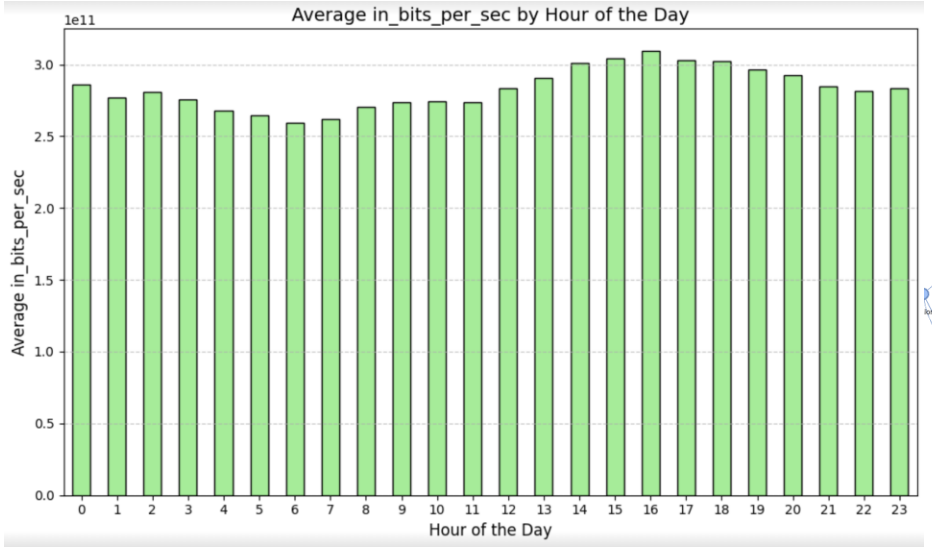


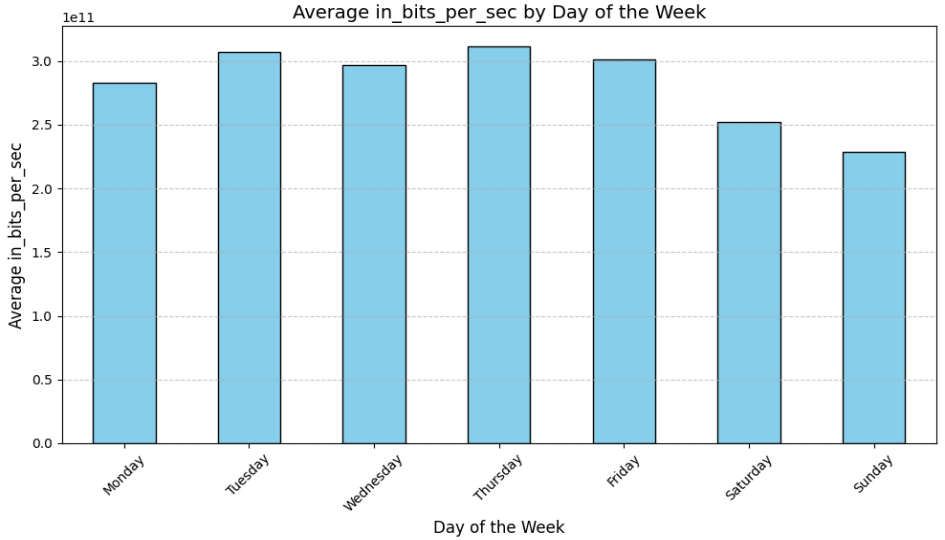
Why is machine learning a candidate for this?

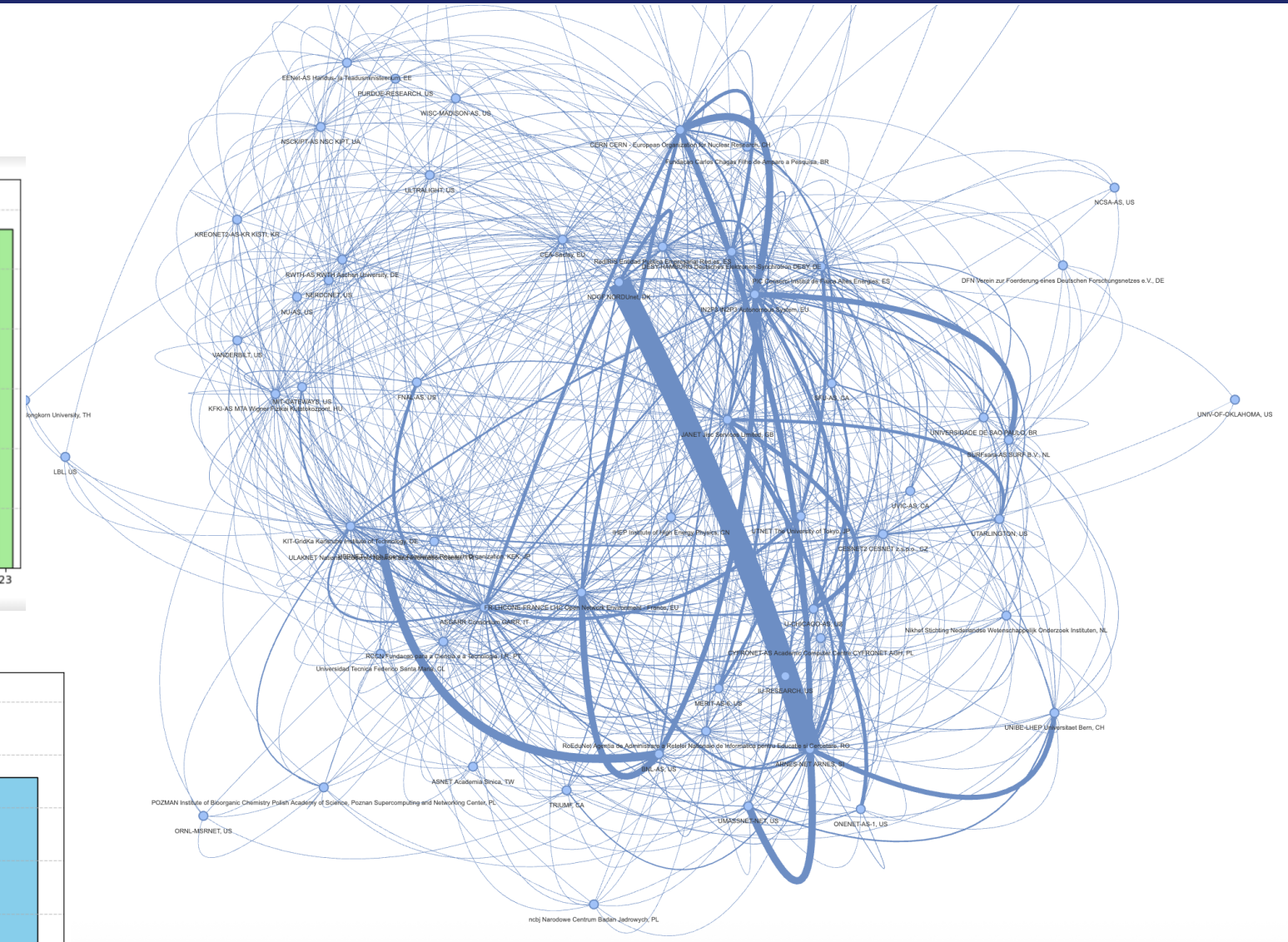
- It scales to large, high-dimensional datasets (network flows) with many attributes
- It can adapt to the dynamics of science, scientist activities changes over time (new tools, new paths, new experiments) – not rule based
- Classification models are very useful: (cyber)anomaly detection and other applications



Why we choose GNN?



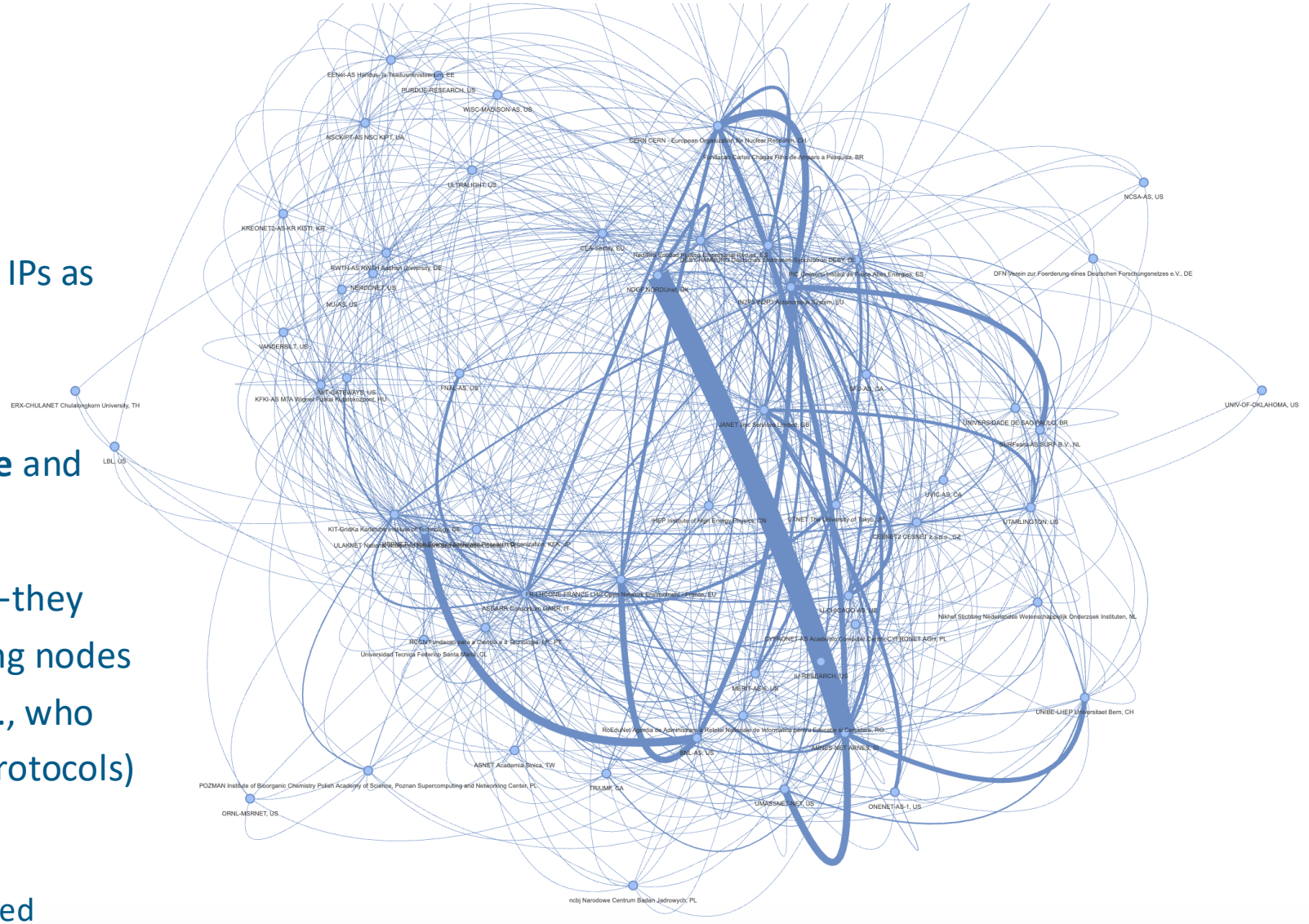




Constructed graph of 15 minutes of flow data

Our experiment

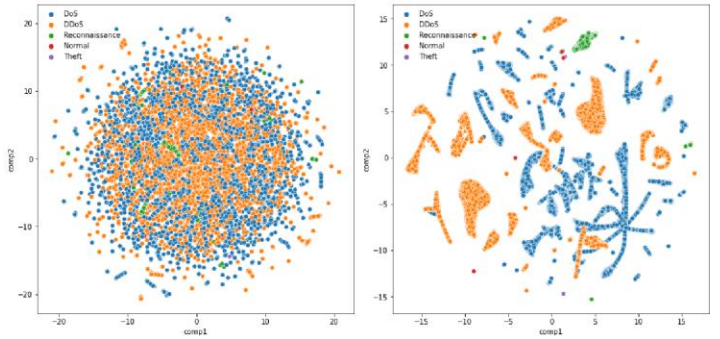
- Network traffic forms a graph structure; IPs as nodes and flows as edges
- Graph Neural Networks (GNN) naturally models the **relational structure** and **topology** of network traffic
- **GNNs are designed to learn on graphs**—they aggregate information from neighbouring nodes to capture **contextual relationships** (e.g., who talks to whom, how often, over which protocols)
- Learning type:
  - Supervised, unsupervised, self-supervised



Constructed graph of 15 minutes of flow data

Our reference

- Experiments on GNNs – self-supervised for intrusion detection
  - Anomal-E[1] is a **GNN** that uses E-GraphSAGE[2] to **aggregate information** in a graph context, and uses DGI to **generate edge (flow) embeddings** with high information density
  - Outlier detection and clustering (unsupervised)

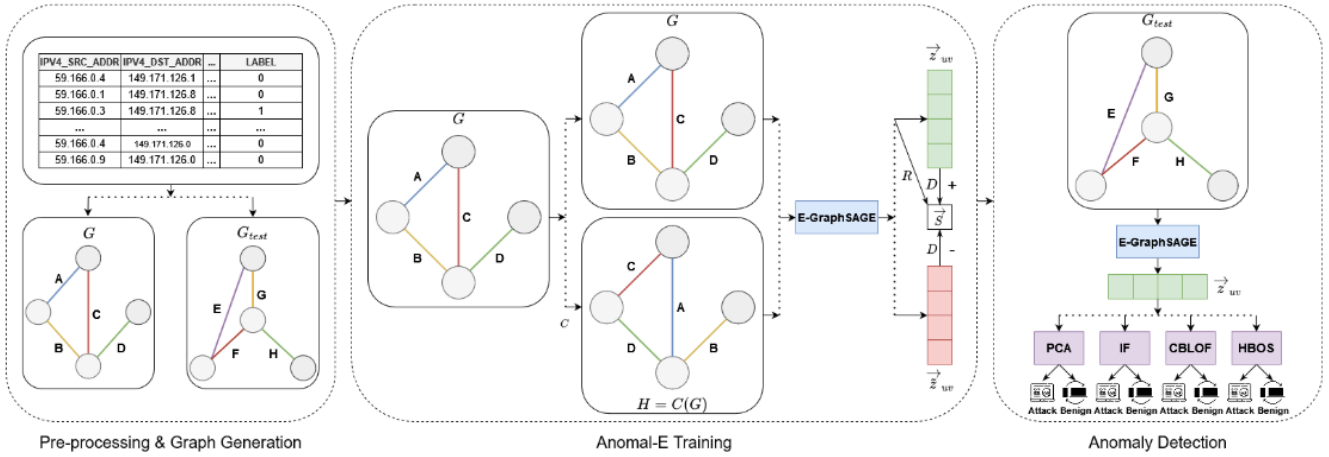


(a) raw data (b) edge embedded data

Fig. 4: Visualisation of dimensionality reduction a) Sample of BoT-IoT raw validation data, b)Sample of edge embeddings generated by E-GraphSAGE (Multiclass).

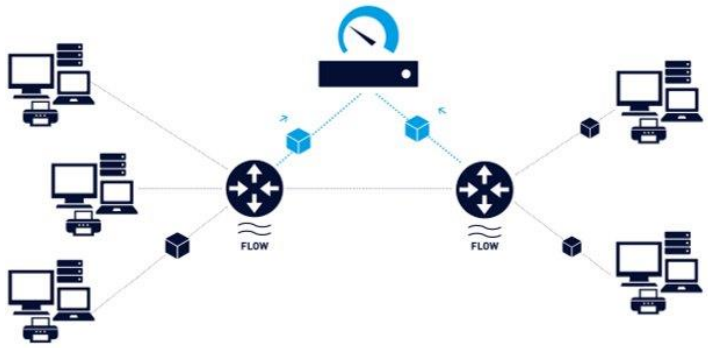
Table 6: NF-CSE-CIC-IDS2018-v2 results (4% contamination).

	Raw Features			Embeddings		
	Acc	Macro F1	DR	Acc	Macro F1	DR
PCA	85.91%	73.76%	74.71%	97.11%	92.57%	79.16%
IF	86.1%	74.09%	75.39%	89.79%	81.11%	91.84%
CBLOF	94.61%	86.18%	69.16%	97.80%	94.38%	82.67%
HBOS	88.81%	78.82%	84.22%	96.86%	91.89%	77.79%



[1] *Caville, E., Lo, W. W., Layeghy, S., & Portmann, M. (2022). Anomal-E: A self-supervised network intrusion detection system based on graph neural networks.*
[2] *Lo, W. W., Layeghy, S., Sarhan, M., Gallagher, M., & Portmann, M. (2022, April 25). E-GraphSAGE: A graph neural network based intrusion detection system for IoT.*

Our experiment



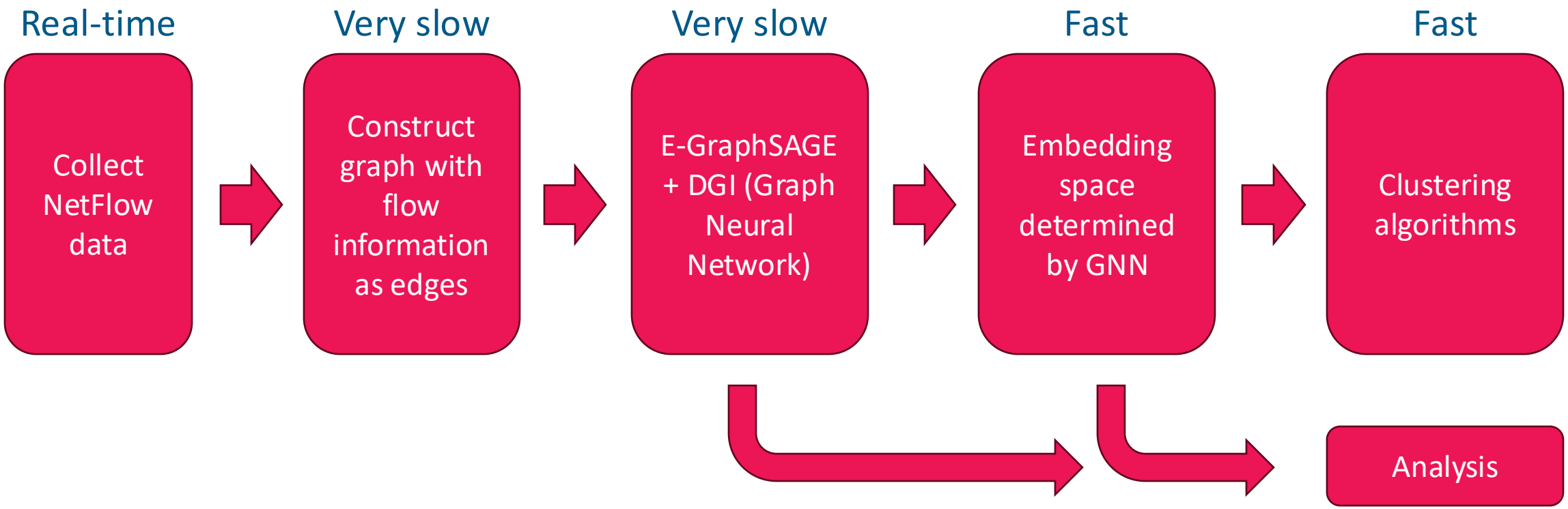
Setup:

- Computing
  - NVIDIA A16 GPU (8x16GB VRAM), 72 CPU cores, 1024GB memory
- Data
  - Collect NetFlow and store, ~70 million flows per hour, ~100GB of JSON data per hour
  - Use Apache Spark analytics engine to process millions of flows
    - For some experiments, "tag" flows from the LHCONE VRF and balance dataset to have 50% LHCONE flows

Juniper IPFIX Fields
IPv4 Source Address
IPv4 Destination Address
IPv4 ToS
IPv4 Protocol
L4 Source Port
L4 Destination Port
ICMP Type and Code
Input Interface
⋮
TCP Flags
Minimum TTL
Maximum TTL
Number of Flow Bytes
Number of Flow Packets
Time the Flow Started
Time the Flow Ended

Our experiment

- Model
  - Based on Anomal-E, graph edge classification model
  - Use cluster-based outlier detection algorithm on the high-dimensional embedding (output) space of the neural network



Experiments 1 and 2

Experiment 1	Experiment 2
Collect 1 hour of NetFlow data	**Collect** 1 hour of NetFlow data
Enrich data with whether or not a flow is from the LHCONE VRF and other boring stuff	**Enrich** data with whether or not a flow is from the LHCONE VRF and other boring stuff
Take all LHCONE flows, and equally as many random flows: total of **2,285,932** flows	Take all flows: total of **64,238,693** (of which 1,142,966 in LHCONE)
Build graph with **IPs** as nodes and **flow information** as edges	Build graph with **AS numbers** as nodes and **flow information** as edges
Train GNN model	Train GNN model
Analyse output and perform clustering algorithms	Analyse output and perform clustering algorithms

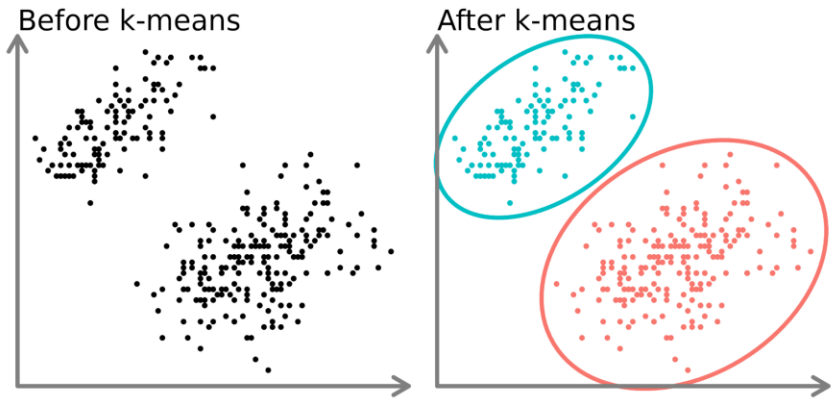
Graph Neural Network (GNN) output

The GNN creates "fingerprints" of the flows

	0	1	2	3	4	5	6	7	8	9	...	246	247	248	249	250	251	252	253	254	255
0	0.221395	-0.185073	0.373323	-0.211745	0.091503	0.040689	0.013104	0.276900	-0.219805	-0.136094	...	-0.206647	0.190476	0.371733	-0.208708	0.153977	0.061814	0.292769	0.081404	0.379638	0.160174
1	0.221395	-0.185073	0.373323	-0.211745	0.091503	0.040689	0.013104	0.276900	-0.219805	-0.136094	...	-0.206647	0.190476	0.371733	-0.208708	0.153977	0.061814	0.292769	0.081404	0.379638	0.160174
2	0.221395	-0.185073	0.373323	-0.211745	0.091503	0.040689	0.013104	0.276900	-0.219805	-0.136094	...	-0.206647	0.190476	0.371733	-0.208708	0.153977	0.061814	0.292769	0.081404	0.379638	0.160174
3	0.221395	-0.185073	0.373323	-0.211745	0.091503	0.040689	0.013104	0.276900	-0.219805	-0.136094	...	-0.206647	0.190476	0.371733	-0.208708	0.153977	0.061814	0.292769	0.081404	0.379638	0.160174
4	0.221395	-0.185073	0.373323	-0.211745	0.091503	0.040689	0.013104	0.276900	-0.219805	-0.136094	...	-0.206647	0.190476	0.371733	-0.208708	0.153977	0.061814	0.292769	0.081404	0.379638	0.160174
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4691833	0.480258	-0.082203	0.880358	0.051322	0.268252	0.258964	-0.233386	0.403043	0.074915	-0.113126	...	0.025958	-0.108880	0.199563	-0.503819	-0.174624	-0.004047	0.096968	0.714666	0.610057	0.002281
4691834	0.606819	-0.253353	0.564960	0.180517	0.366050	0.416110	0.256840	0.679993	-0.071344	-0.003415	...	0.105906	0.228686	0.618178	0.157628	0.390622	0.101710	0.638273	0.469125	0.495056	0.448570
4691835	1.099752	-0.397158	1.126398	0.084618	0.504665	0.390928	-0.782568	0.417989	0.386479	0.134922	...	-0.428235	0.908410	0.412572	0.350417	0.416121	-0.061199	1.833144	1.080173	0.703303	0.988090
4691836	0.093823	-0.235978	0.287673	-0.284216	-0.034197	-0.049141	-0.405947	0.289327	-0.048443	0.011677	...	-0.051013	0.320742	0.214195	0.067191	0.143425	0.087214	0.336799	0.215782	0.183972	0.153269
4691837	-0.167690	-0.500528	0.562167	-0.165826	0.079433	0.183501	-0.419025	0.238307	-0.089871	-0.124399	...	0.022309	0.616948	-0.101039	0.350284	-0.048753	0.259371	0.948760	0.258383	-0.116613	0.343237

Clustering in experiment 1

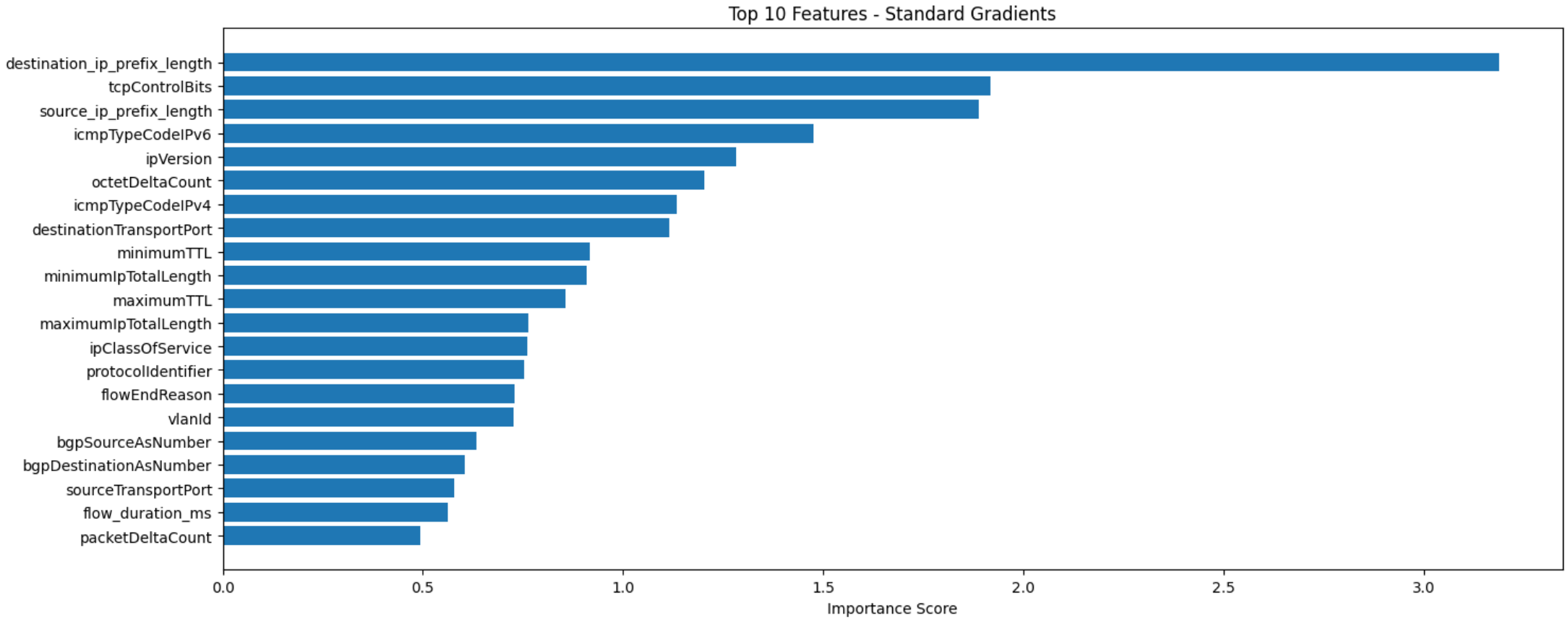
Clustering (K-Means): group data points into clusters based on their inherent similarity



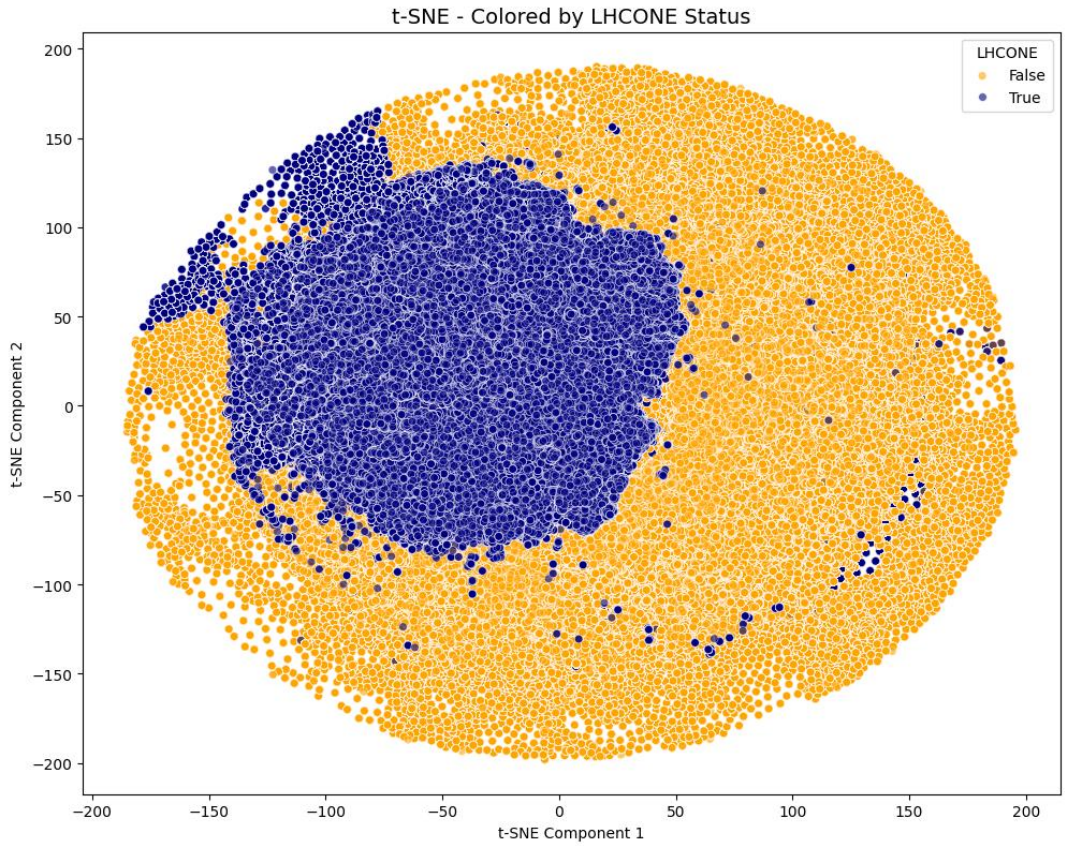

```
N_CLUSTERS=2: ARI=0.421, NMI=0.381, Silhouette=0.372
  Best cluster LHCONE F1-score=0.799
  All cluster F1-scores: 0.799, 0.269
N_CLUSTERS=3: ARI=0.344, NMI=0.311, Silhouette=0.387
  Best cluster LHCONE F1-score=0.783
  All cluster F1-scores: 0.278, 0.059, 0.783
N_CLUSTERS=4: ARI=0.321, NMI=0.302, Silhouette=0.391
  Best cluster LHCONE F1-score=0.783
  All cluster F1-scores: 0.285, 0.057, 0.783, 0.000
N_CLUSTERS=5: ARI=0.565, NMI=0.561, Silhouette=0.270
  Best cluster LHCONE F1-score=0.783
  All cluster F1-scores: 0.451, 0.001, 0.783, 0.000, 0.059

BEST CONFIGURATION:
 ├─ N_CLUSTERS: 5
 ├─ ARI: 0.565
 ├─ NMI: 0.561
 ├─ Silhouette: 0.270
 ├─ LHCONE F1-score in best cluster: 0.783
 ├─ Best correlating cluster ID: 2
 ├─ Cluster size: 1,630,155 flows
 ├─ Cluster LHCONE precision: 94.1%
 ├─ Cluster LHCONE recall: 67.1%
 └─ Cluster LHCONE F1-score: 78.3%
```

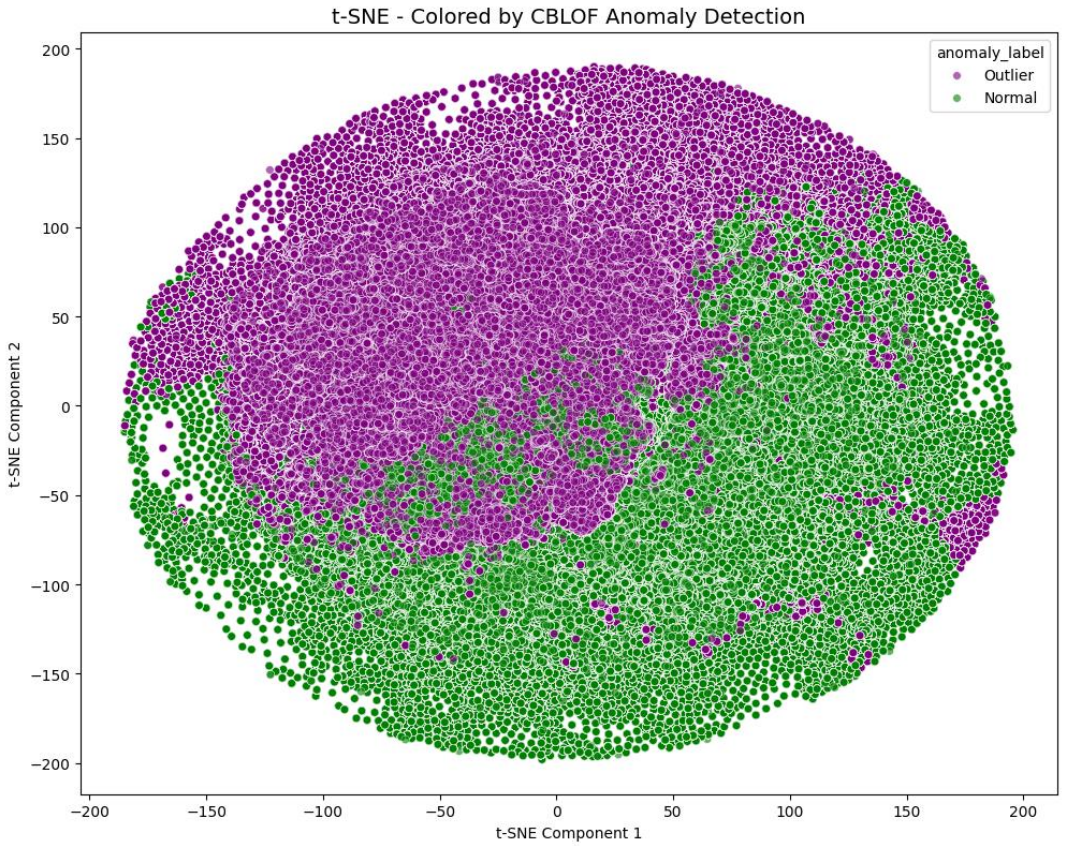
Feature importances experiment 1

Feature importances taken from gradients of GNN model: *"which input features most affect the output embedding?"*



Flow "fingerprints" of experiment 1





94.1% of LHCONE flows were grouped in one group

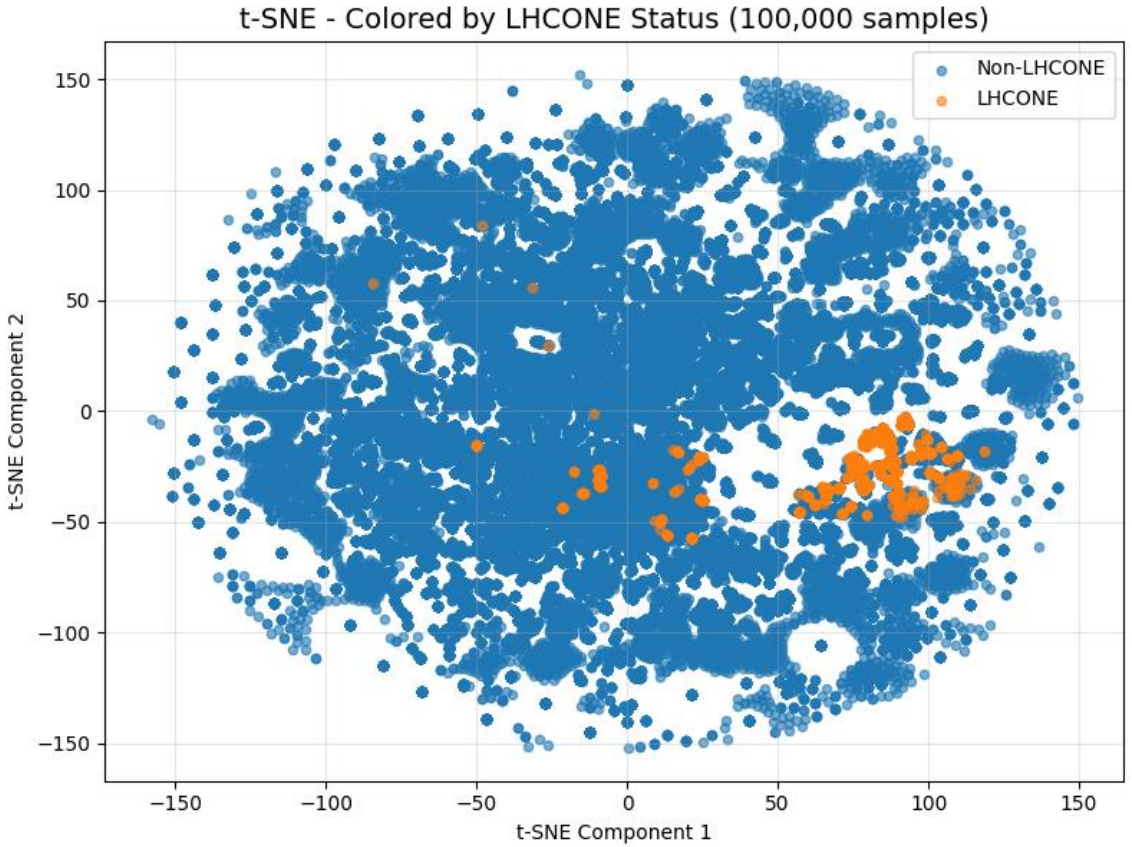
Flow "fingerprints" of experiment 1

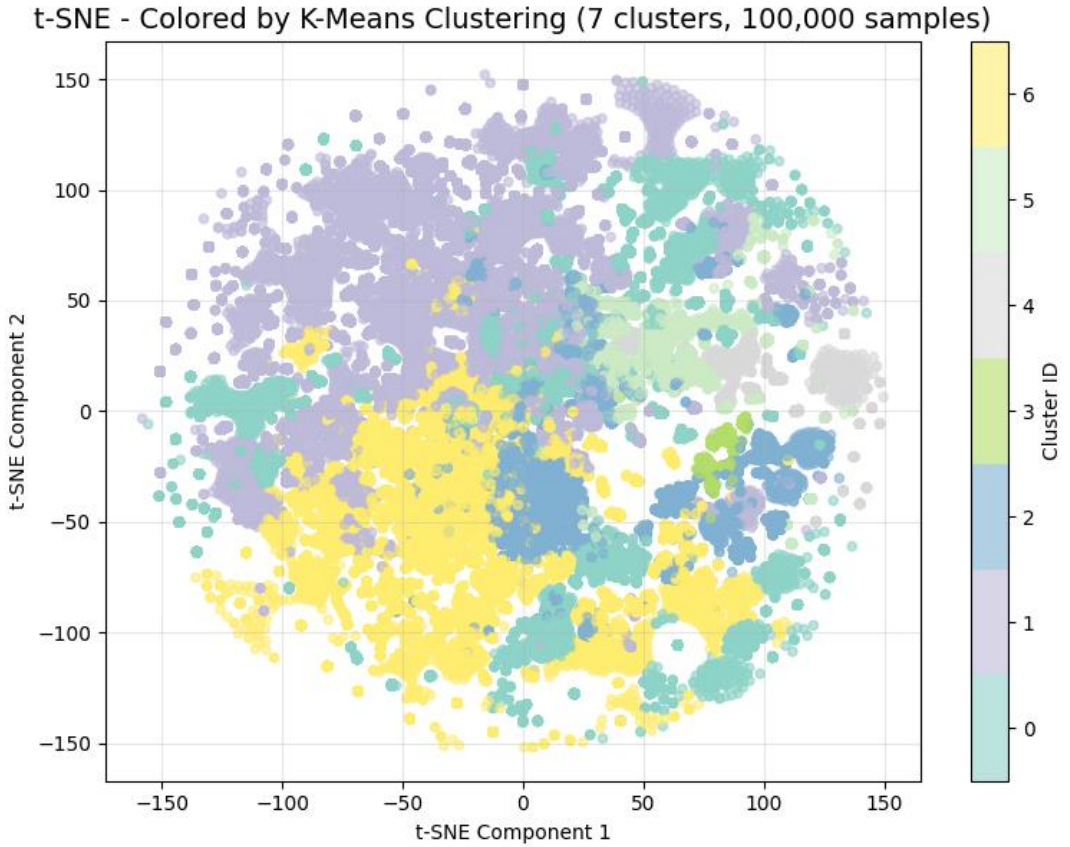
Juniper IPFIX Fields	Values	Compared to median
IPv4 Source Prefix Length	48	+24
IPv4 Destination Prefix Length	48	+24
Flow duration (ms)	0	Same
Source AS	137 (GARR)	-2477
Destination AS	513 (CERN)	-2101
L4 Source Port	55072	+39686
L4 Destination Port	1103	-19473
L4 Protocol	TCP	Same
⋮		
Number of Flow Packets	7	+6
Number of Flow Bytes	10500	+9999
Minimum TTL	61	Same
Maximum TTL	61	Same
TCP Flags	.A....	Not same
IP Version	6	+6
IP Class of Service	0	Same

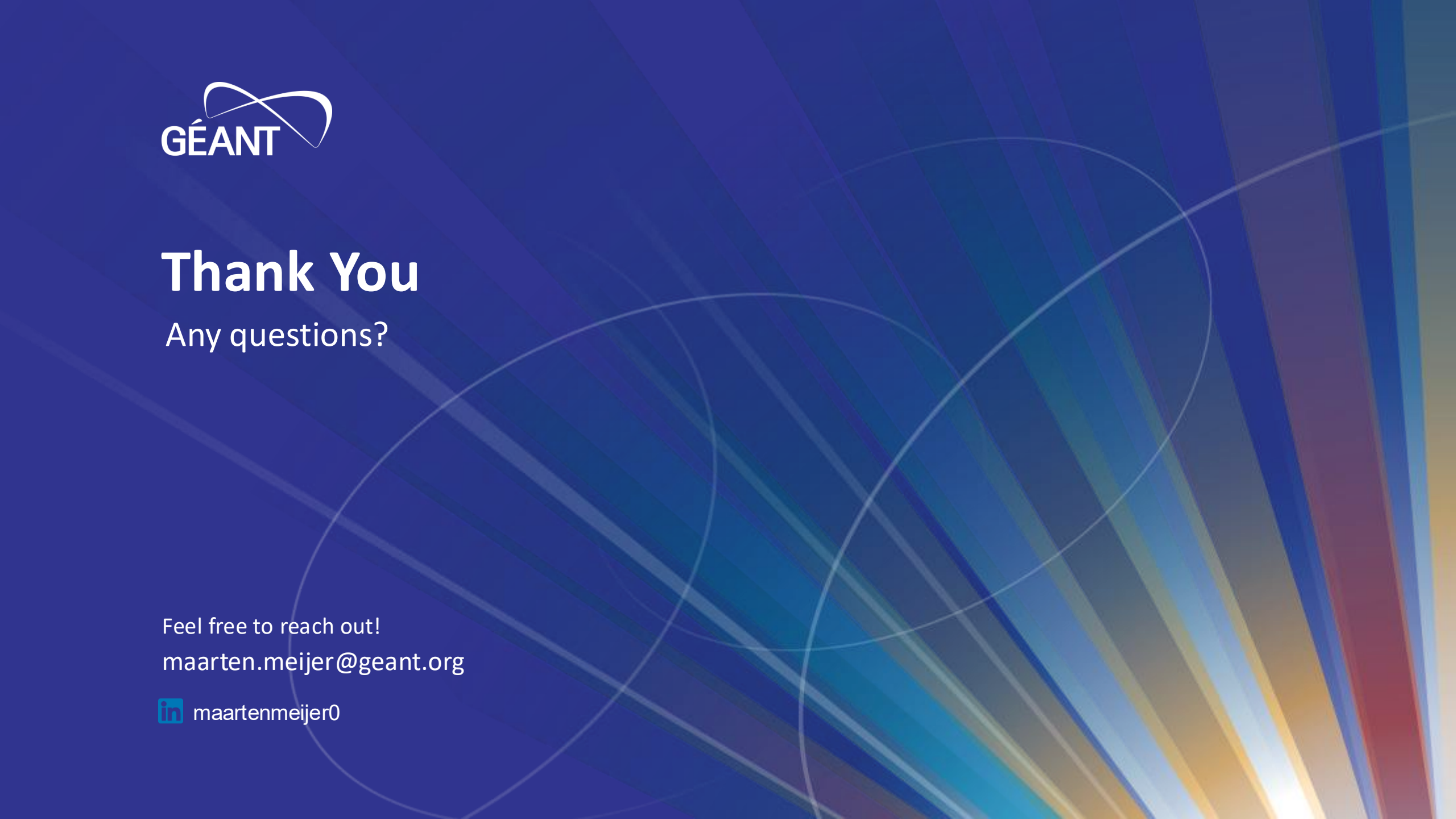
Flow "fingerprints" of experiment 2

Graph is only 100,000 flows, but calculations are with all flows from an hour (64 million)

Cluster	% of total flows in cluster	% of cluster in LHCONE VRF
0	17.2%	0.0%
1	33.9%	0.1%
2	10.8%	7.5%
3	1.1%	62.7%
4	2.2%	0.2%
5	5.5%	0.0%
6	29.4%	0.4%







Thank You

Any questions?

Feel free to reach out!

maarten.meijer@geant.org

maartenmeijer0