Minsky Lab

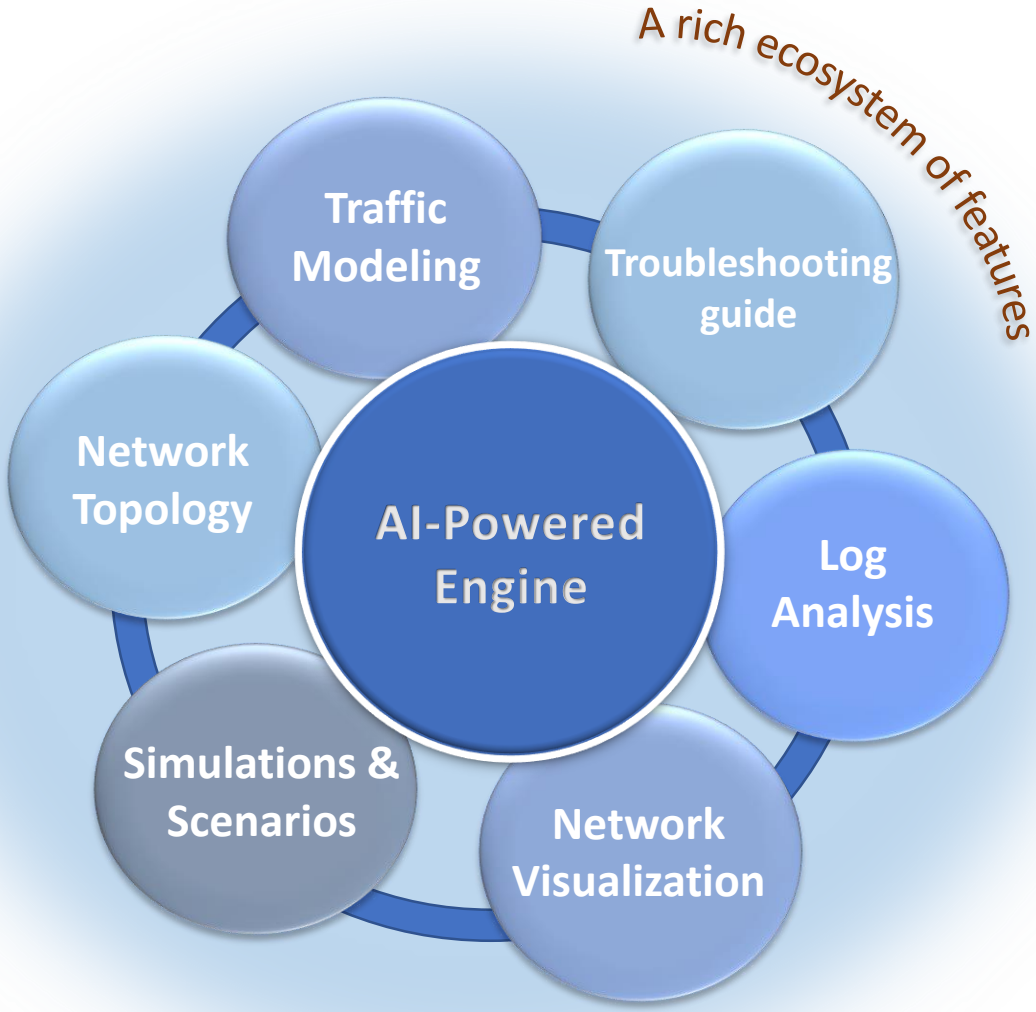
Digital-Twin Junior

TOGA NETWORKS LTD., A HUAWEI COMPANY

Digital Twin Junior:

a Comprehensive Portfolio

of AI-driven Network Intelligence Tools



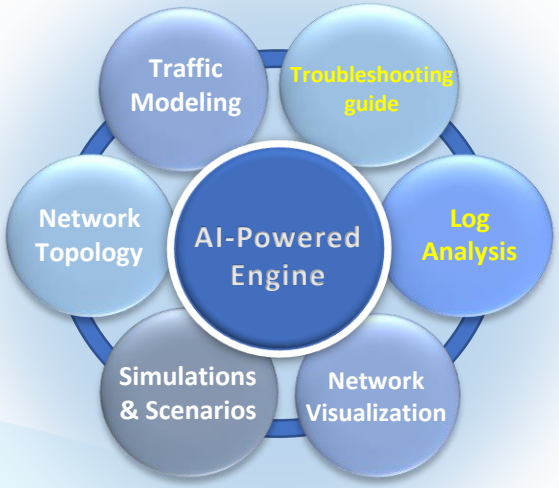
Included capabilities:

- ☑ Network simulation and scenario modeling
- ☑ Traffic Modeling: analysis and forecasting
- ☑ Network Topology mapping and diagnostics
- ☑ Automated log interpretation

\+ **Interactive tools:** Network visualization, Troubleshooting guide

Digital Twin Junior:

Log Analysis and Interactive Troubleshooting



Network status extraction: logs, alarms

2019-06-12T10:20:30.000Z WARNING PKI/4/HTTP_GET_CRL_ERR: file_name:[test.crl] Manually obtaining CRL [test.crl] through HTTP failed (Reason=[URL resolution failed]). category=PKI subCategory=CRL component=HTTP, eventID=HTTP_GET_CRL_ERR,vendor=Huawei,product=Fat AP,version=V200R022C00, timestamp: [2019-06-12 10:20:30] [rid:T6zHJ38AAAEAAAo2BCwAAAMk ip:[192.158.1.38]:1813/UDP

2019-06-13T11:21:31.000Z WARNING PKI/4/HTTP_GET_CRL_ERR: file_name:[test1.crl] Manually obtaining CRL [test1.crl] through HTTP failed (Reason: Failed to save the response message). category=PKI subCategory=CRL eventID=HTTP_GET_CRL_ERR,vendor=Huawei,product=Fat AP,version=V200R022C00, timestamp: [2019-06-13 11:21:31] [rid:T6zHJ38AAAEAAAo2BCwAAAMk ip:[192.158.1.39]:1813/UDP

Log entry explanation

Alarm Description:
After receiving a Hello packet from a neighbor, the device records information about the neighbor and starts a timer for the neighbor. If the device does not receive any Hello packet from the PIM neighbor before the timer expires, the device reports a trap.

Log Name	Describes the log name)	hwPimNeighborLos
NbrIntIndex	Index of the interface where the neighbor resides	2
NbrUpTime	Time when a neighbor relationship is established	2020-12-12 10:20:30

Recommendation for a solution

1. Check whether the reset msdp peer command is run in the user view on the peer device.
   - If is performed, wait until the peer is re-established.
   - If not, go to Step 2.
2. Run the display this command in the MSDP view of the peer device to check whether an MSDP peer is configured.
   - If yes, go to step 3.
   - If no, run the peer connect-interface <interface-type> <interface-number> command in the MSDP view to configure an MSDP peer.
3. Run the display interface brief command to check whether the shutdown command is configured on the peer interface.
   - If so, run the undo shutdown command in the interface view to clear the alarm.
   - If no, go to step 4.
4. Collect alarm and configuration information, and contact technical support personnel.

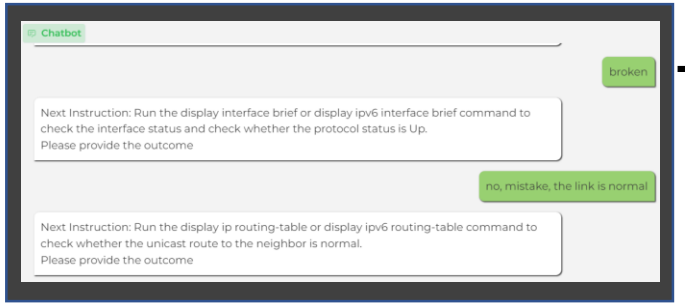
Key abilities of the network assistant

- ☑ Analyze and explain network events
- ☑ Identify causes
- ☑ Suggest solutions
- ☑ Resolve alarms
- ☑ Summarize incidents

Summary of the incident

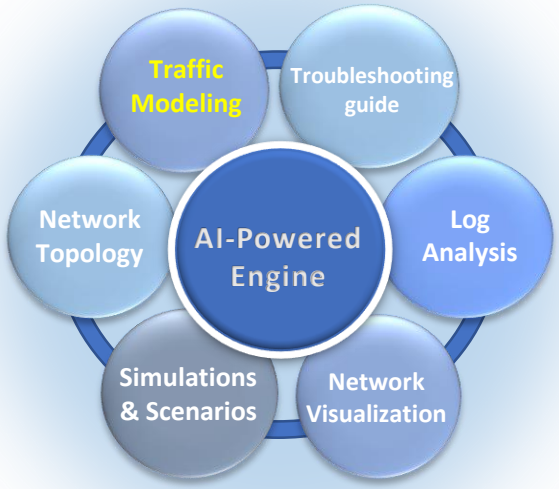
The root cause of hwPimNeighborLoss was investigated. Here are the main findings:
Step 1: Ping/Ping IPv6 command outcome: broken (link between neighbors is not normal)
Step 2: Display interface brief/Display IPv6 interface brief command outcome: no, mistake (link is normal)
Step 3: Display IP routing-table/Display IPv6 routing-table command outcome: seems ok
Conclusion: The problem was not solved.

Interactive step-by-step procedure following

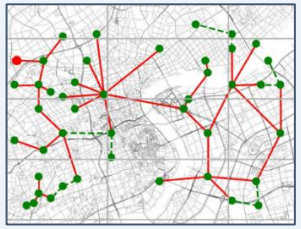


- ➢ Identify API call
- ➢ Extract arguments
- ➢ Execute
- ➢ Get output

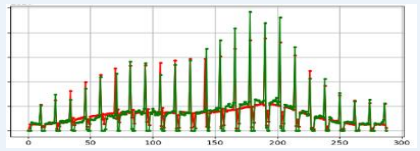
Digital Twin Junior: Traffic Modeling



Forecasting



- Horizon – from very short (1 min) to very long (1 day)
- Predict peaks & 'quite periods' – be prepared for overloading and energy saving time
- Unstable signal (e.g. hardly responding cell towers) – no problem, prediction is based on neighborhood data
- Holidays and other key events – we learn and predict even unusual traffic patterns
- Different topologies – known or unknown



prediction restores peaks when they are signal

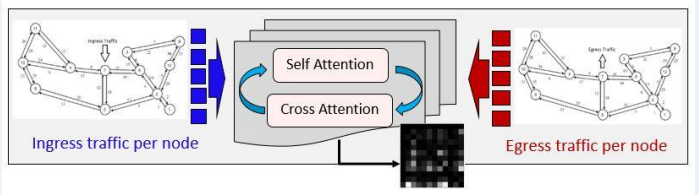
prediction ignores peaks when they are noise

ground truth

prediction

Estimation

- Traffic Matrix from link loads: Route-Aware GNN
- Traffic Matrix from Ingress/Egress Traffic: Attention-based GNN (Deep Gravity)



Completion

Completion in Time

- 'Estimate the present' – fill in the missing values (traffic imputation)
- 'Predict the future' almost without knowing the past (with up to 80% of the historical data missing)

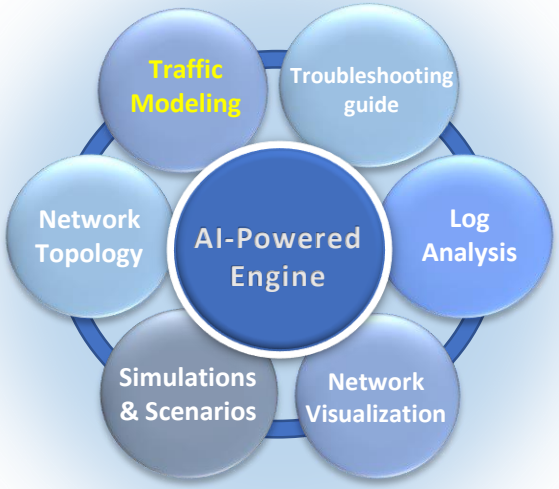
Completion in Space

- Traffic Inversion for sampled traffic in network monitoring tools

Key techniques: Attention-based Graph Convolutional Neural Networks

Can be applied to: Backbone networks, Campus network, Datacenter networks

Digital Twin Junior: Traffic Inversion



Network monitoring

As networks evolve and become more complex, the need for network monitoring tools has remained constant.

It is highly important to ensure the network sustainability and guarantee all the processes to run smoothly and error-free for users.

Network traffic monitoring contains tools for data collection, analysis, diagnosis and solutions to resolve problematic issues.

Network Traffic Collection

The collection of **full detailed traffic** data does not scale well with increasing network attributes, like size, link bandwidth, speed etc.

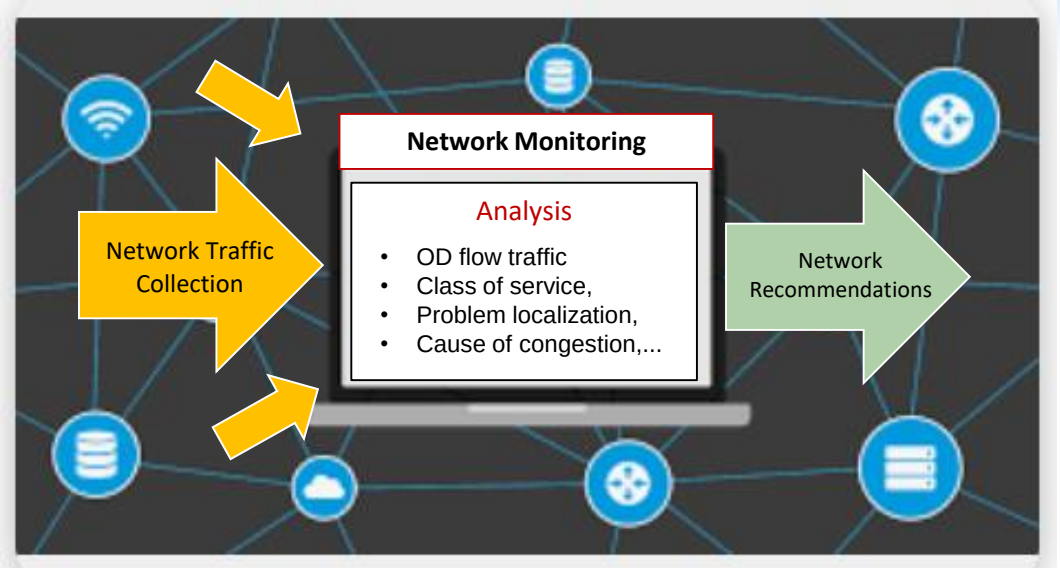
Network Traffic Sampling

- The load on the measurement infrastructure is reduced.
- Problems: Only a portion of the traffic is exported. Some flows can be missed or represented by a very small portion. Sampled flows may not provide a sufficient amount of information about full network traffic

Network Traffic Inversion

Flow inversion is a restoration of the full traffic data or its statistics.

Existing approaches cannot restore the actual fine-grained traffic with good accuracy.



Some standards:

External products:
- NetFlow
- sFlow
- IPFIX

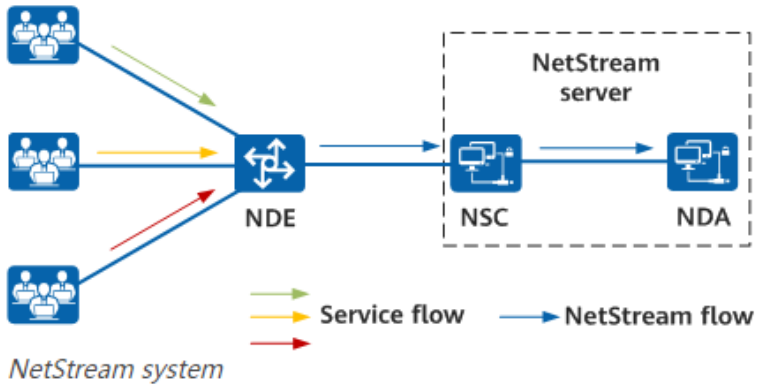
Huawei product:
- NetStream (our main use case):

Full fine-grained traffic inversion

- In most cases sampling is inevitable…
- How to obtain a full information from sampled data?

NetStream

An internal Huawei network monitoring tool that collects and analyzes a network traffic per flow.



Full flows are not directly available from NetStream sampled data.

Let's consider 2 possibilities:

Statistics:
- ✓ Easy to calculate
- ✗ Loose a temporal structure of the flow data

DNN/AI tools:
- ✓ May be very accurate
- ✗ Require a lot of labeled data for the training

Input data

Device (router)

equipped with a network monitoring tool

Network Monitoring Functionality

Standard sampled data tuple:

timestamp, {source IP, destination IP, input port, output port, PPS, BPS}

NetStream data example

	A	B	C	D	E	F	G	H
1	bps	deviceIp	dstIp	inputDeviceInterface	outputDeviceInterface	pps	srcIp	timestamp
2	499600	10.79.253.46	10.58.128.98	14	15	50	10.243.33.3	1658688600
3	98600	10.79.253.46	7.186.131.9	79	14	110	10.58.83.203	1658688060

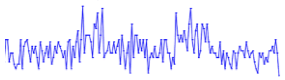
- ✓ **Timestamp**
- ✓ **Source IP**
- ✓ **Destination IP**
- ✓ **Input port**
- ✓ **Output port**

Information for traffic data pre-processing

- Subnet
- Application
- Group of servers
- Traffic clustering/detection per user

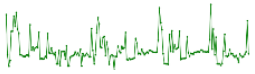
- ✓ **PPS**
- ✓ **BPS**

Sampled traffic



restore

Full traffic



- For specific end-user, subnet
- Total Ingress/Egress traffic inversion
- For application/group of applications

Full fine-grained traffic inversion

Data discussion:

- Lots of observations are available from monitoring devices → natural solution → train DNN
- **Big problem**: our sampled data is not labeled (corresponding full traffic is not usually available)
- **Literature suggests**: incorporating knowledge of different noise and sampling patterns can significantly improve signal reconstruction
- So, to achieve accurate recovery, **it is crucial to replicate the sampling procedure** used by the monitoring device
- **Why not to just multiply by sampling ratio?** Even if total traffic is sampled deterministically (our use case) by inspecting every Rth packet, it would be generally inaccurate to assume that also every Rth application packet is considered

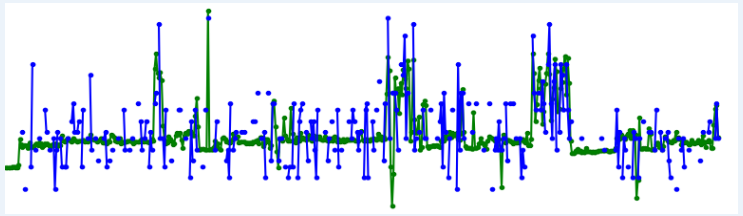
Our algorithm highlights

- ✓ Estimates an **actual time-dependent fine-grained traffic** with low error
- ✓ Combines statistical approach with recent AI techniques:
  - Is data-driven and AI-based: it reflects the actual data features
  - **Doesn't require labeled data**
  - Can be automatically adapted to a new data
- ✓ Is implemented as a lightweight and efficient deep neural network

Simple statistical manipulations

Assumption: if sampling ratio is know, we can scale back the sampled signal and obtain a good solution

Compare true signal to scaled sampled signal

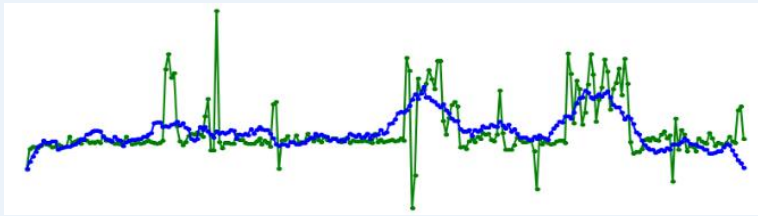


MAPE = 31%

Sampling Ratio * sampled application traffic

full application traffic

Compare true signal to **smoothed** scaled sampled signal



MAPE = 18%

Sampling Ratio * smoothed sampled application traffic

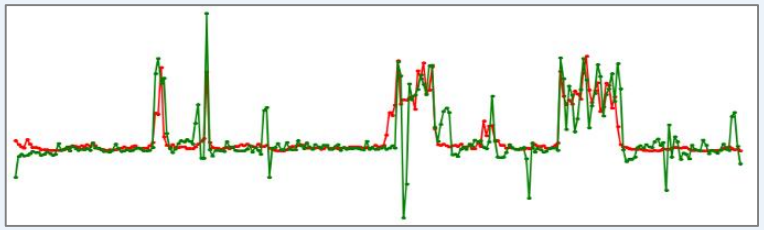
full application traffic

Simple statistics are not enough:

✓ **Good average** fit for a large time period

✗ **Poor local** fit per timestamp (either too noisy or oversmoothed)

Our model: Statistics + Attention-based DNN



MAPE = 12%

Our result

full application traffic

Train data modeling

Definitions:

R - sampling ratio

n - sampled number of packets

N - total number of packets

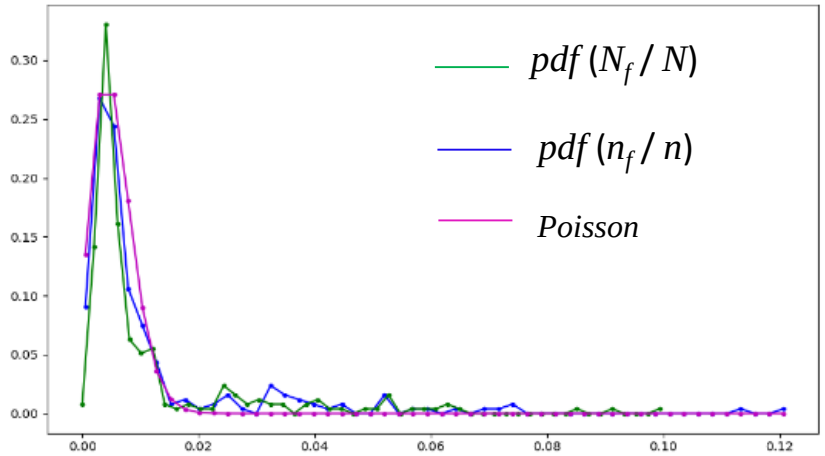
n_f - sampled number of flow packets (per application per user)

N_f - total number of flow packets

(traffic is aggregated)

Empirical result as a base for the sampling principle: $pdf\,(n_f / n) = pdf\,(N_f / N)$

$N_f << N$, hence the 'the law of small numbers' (Poisson) is straightforward to apply

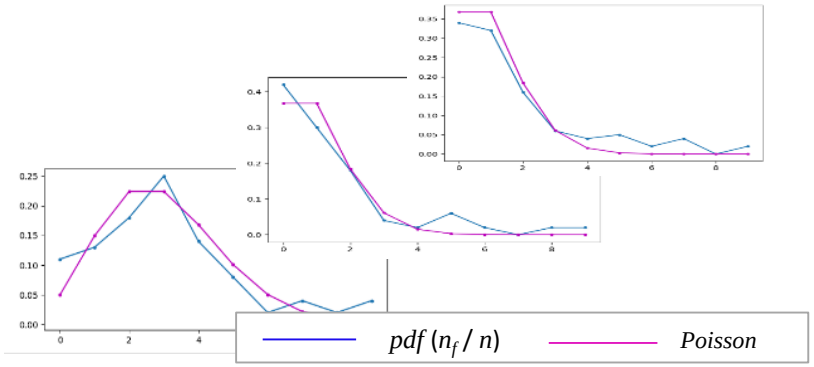


$pdf\,(n_f / n) \sim pdf\,(N_f / N) \sim Poisson\ distribution$

Approach:

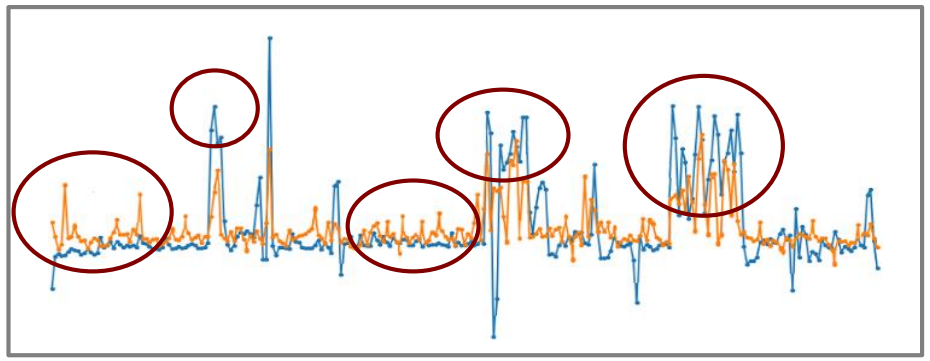
- Extract and apply a noise scheme/sampling to NetStream data to create a 'noisier' signal.
- Train a network to restore a noise signal from 'noisier' signal

Generated train data: inputs = $n_f^{sampled}$, labels = $n_f^{original}$

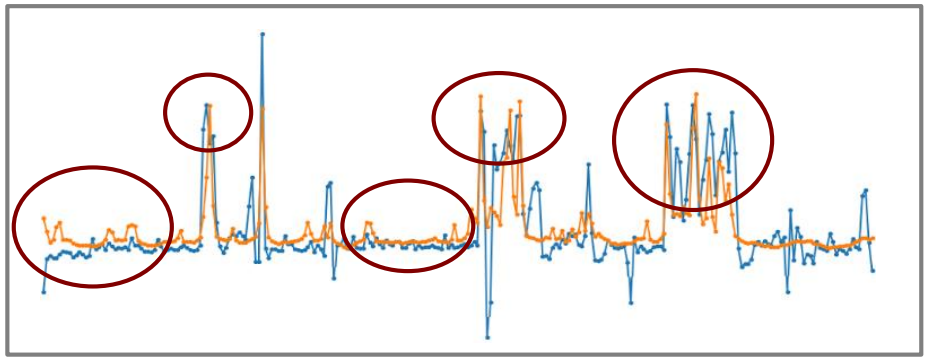


Spoiler: Why Poisson data modeling is better

Restored traffic (orange) vs ground truth traffic (blue)



(a) with random sampling



(b) with Poisson sampling

Conclusion: Although the network trained on double-sampled data, that underwent random sampling (instead of Poisson one) succeeded to restore the overall pattern, its result is much noisier and some important peaks are missed or underestimated

Model structure

Network MONITORING TOOL

(e.g. NETSTREAM)

Training phase

Original monitored traffic → Train data modeling → Modified monitored traffic → Attention based DNN *- Training mode -* → Restored monitored traffic

Statistics based module

Functionality: Apply sampling principle to create double-sampled data

Details: Extract traffic segments (pps, bps) and apply sampling principle

If sampling principle is known

- Calculate statistics of the extracted application traffic
- Generate train data based on calculated statistics

If sampling principle is unknown:

- Apply general sampling procedure (based on random noise) to generate train data

DNN based module

Functionality: Train a NN to restore full data from sampled data

Details:

- Sequence-to-sequence Encoder-Decoder scheme
- Convolution + Attention mechanism
- Double-loss scheme:
  - estimation loss makes the output close to the provided label
  - self-loss with high drop out serves as a regularization term

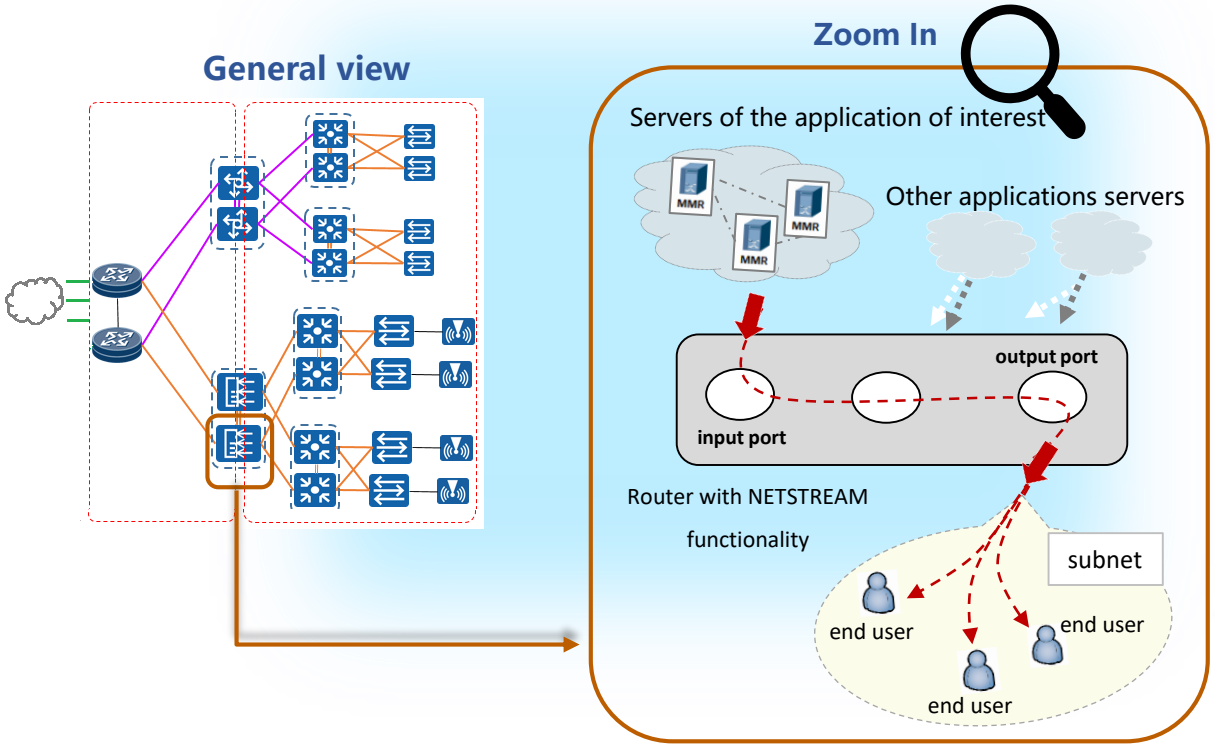
Lightweight: small number of parameters, can be trained on CPU

Inference phase

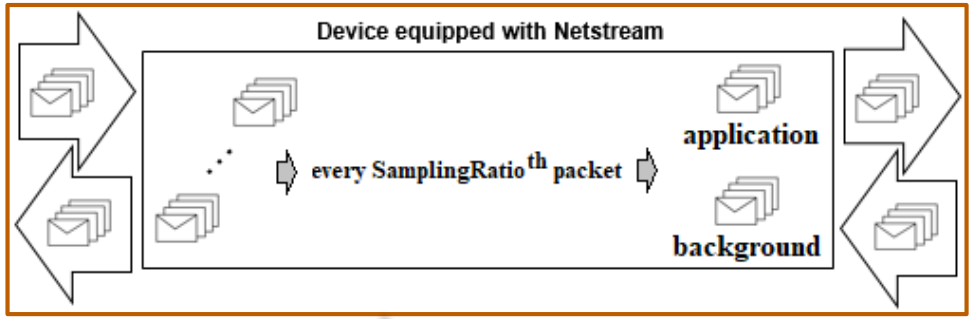
Original monitored traffic → Attention based DNN *- Inference mode -* → Restored full traffic

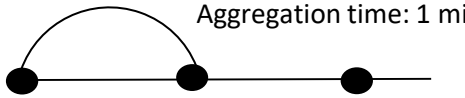
Use case: Campus Network

Real campus network with 2 routers equipped with NetStream feature are mounted on top of the campus network



NetStream feature allows to collect sampled traffic that enters and exits the campus

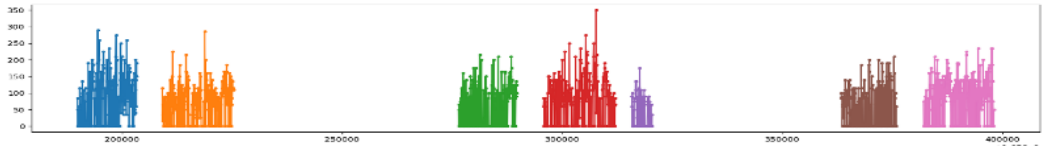




Requirement:

To reconstruct the complete traffic generated by a Conference Call (CC) application on a per-user basis

A few examples of the CC traffic.



Each color denotes a separate conference call PPS along the time axis

Conference Call traffic inversion

Training

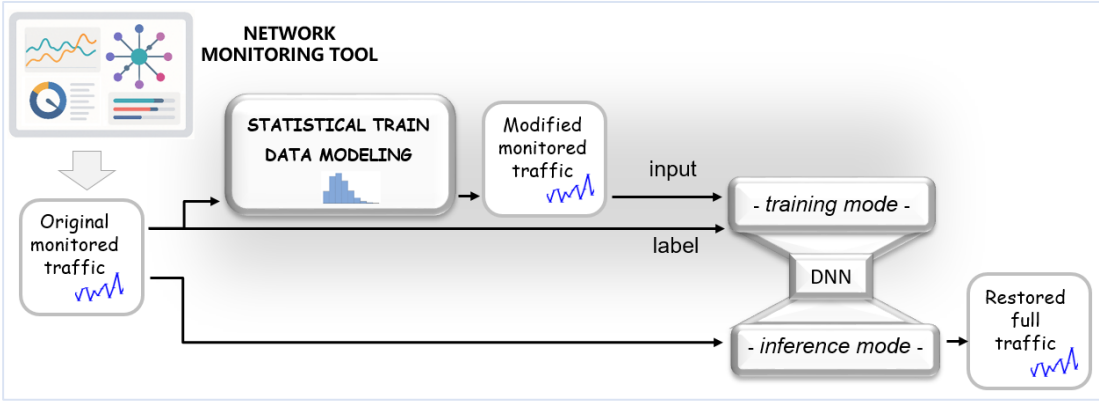
Sampled data: NetStream measurements contained 19 large batches, each batch including both CC and background traffic samples from about 5-7 days of records

Data preparation

1) Define input CC traffic by identifying CC server IP (‘source IP’ from data tuple)
2) Extract CC PPS and BPS segments corresponding to various users (users = different 'destination IP’ from data tuple)

Model training

1) Use extracted CC segments as inputs to the model
2) Train model



Verification

Sampled data: NetStream measurements.

Ground truth: a few samples with total duration of 1,584 min of full fine grained CC traffic records (obtained with Wireshark).

Mean Absolute Percentage Error

$$MAPE = \frac{100\%}{n} \sum \frac{|True(t) - Estimation(t)|}{|True(t)|}$$

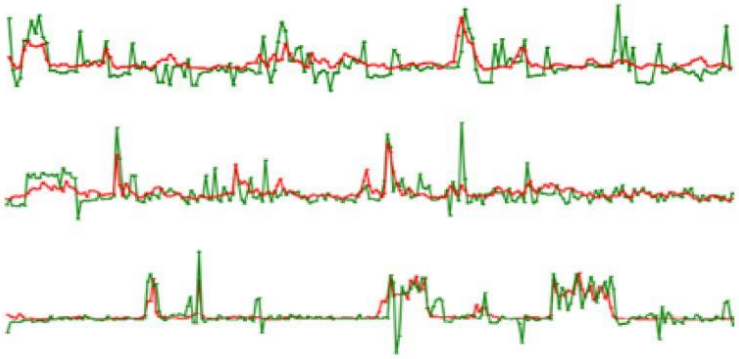
Symmetric Mean Absolute Percentage Error

$$SMAPE = \frac{100\%}{n} \sum \frac{|True(t) - Estimation(t)|}{0.5 * |True(t) + Estimation(t)|}$$

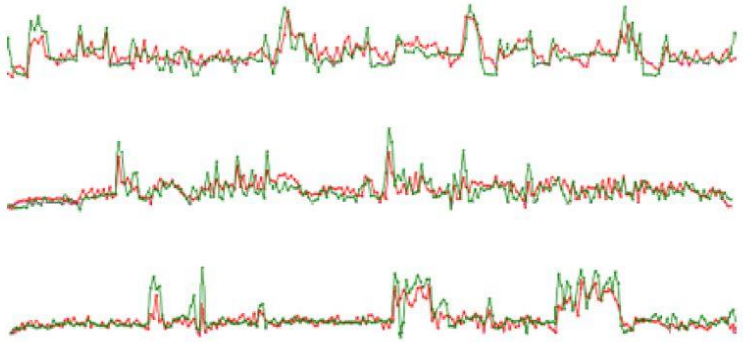
Average inference time for 50 timestamp-long chunk: 17 ms for max number of parameters required. Can be further finetuned

Evaluation and results

3 conference call samples pps



3 conference call samples bps

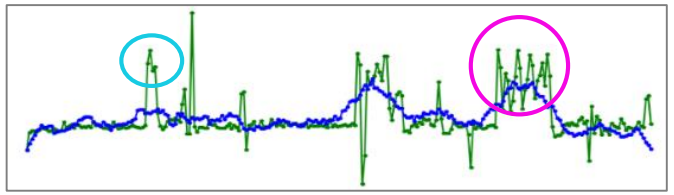


Restored signal is shown in red, true signal is shown in green

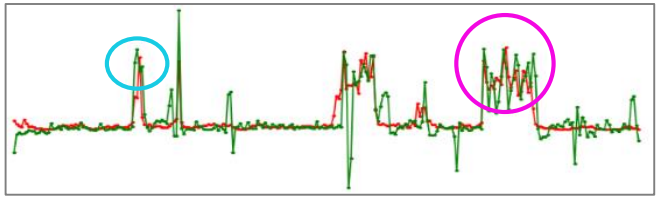
Method	MAPE (pps)	SMAPE (pps)	SMAPE (bps)
Upscaling (unsmoothed)	31.4%	14.0%	33.8%
Upscaling (smoothed)	18.0%	11.5%	27.3%
Our method (DNN + stats)	11.7%	5.6%	13.1%

Visual comparison of the statistics-based method (blue) and our method (red) to the ground truth (green).

Statistical results hardly restore peaks (cyan circles) and over smooth the data (magenta circles)



SMAPE = 13%

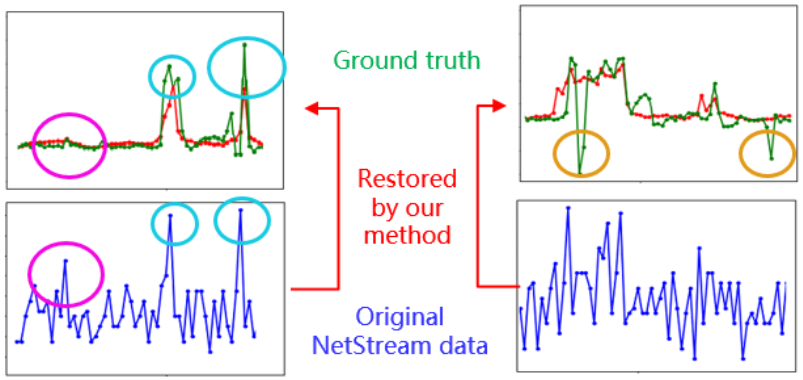


SMAPE = 5.1%

Summary and Discussion

Developed a **self-supervised algorithm** to restore full traffic data from highly sampled traffic measurements

- No labeled training data required
- Low cost (adjustable number of parameters, can be trained on CPU, short inference time
- Efficient & Accurate



(a) Method restores true peaks (cyan circles) and ignores irrelevant sampled peaks (magenta circle)

(b) Limitations: Low values are not presented in NetStream data and cannot be restored (yellow circles)

General applicability: Due to the standard input the product can be applied to the data collected by other vendors tools (like NetFlow)

Additional benefits and suggestions

- **Cost reduction**: A trade-off between sampling ratio and inversion accuracy can be found.
- **Optimization**: Internal parameters (aging cache etc.) can be fine-tuned with respect to the desired accuracy.


Thank you