Building a Network Digital Twin: Emulation vs. Simulation vs. Analytical vs. Neural Networks

Prof. Albert Cabellos

alberto.cabellos@upc.edu

SANS Workshop 2024

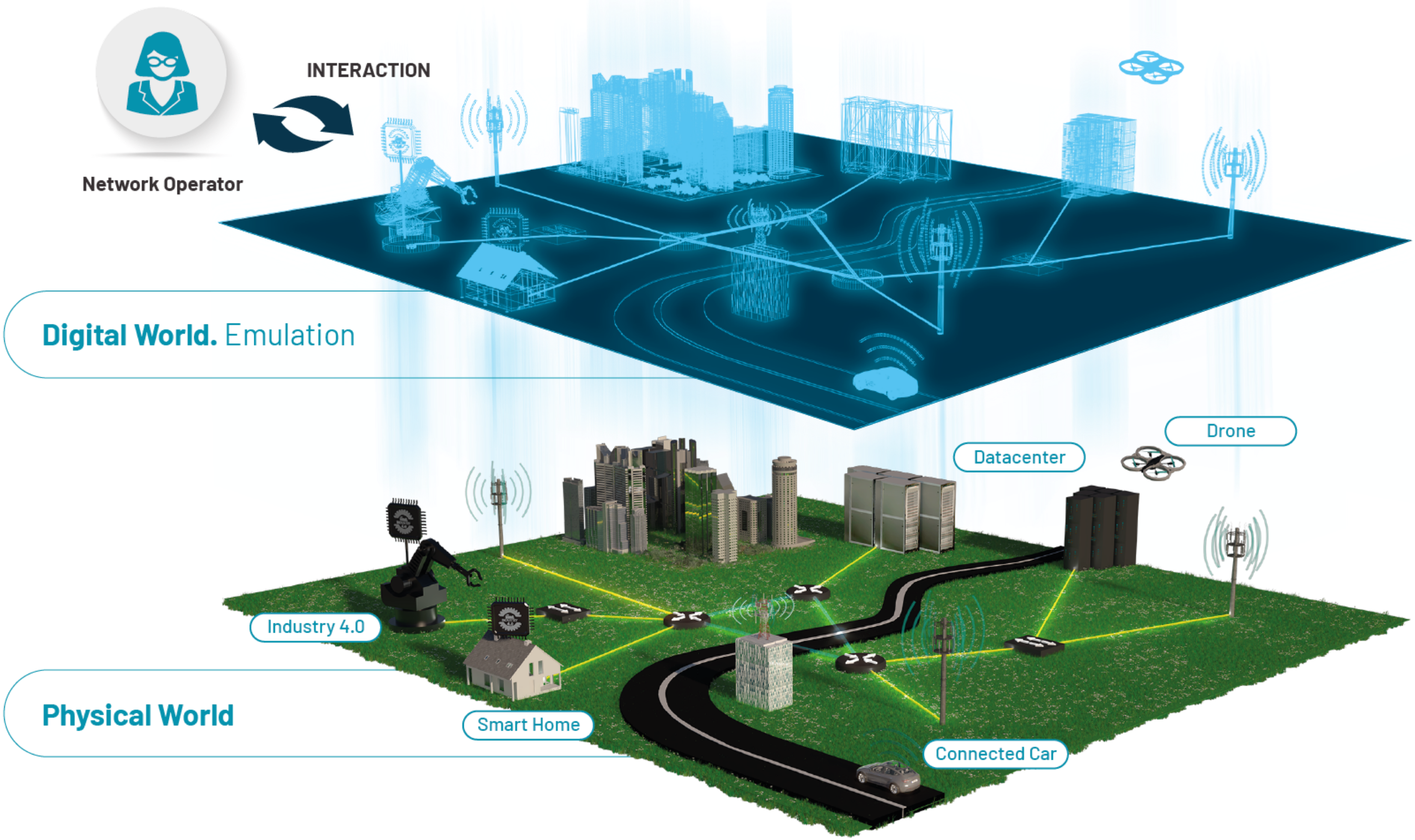
Online, May 2024

Barcelona Neural Networking Center

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

v5

What is a
Network Digital Twin?

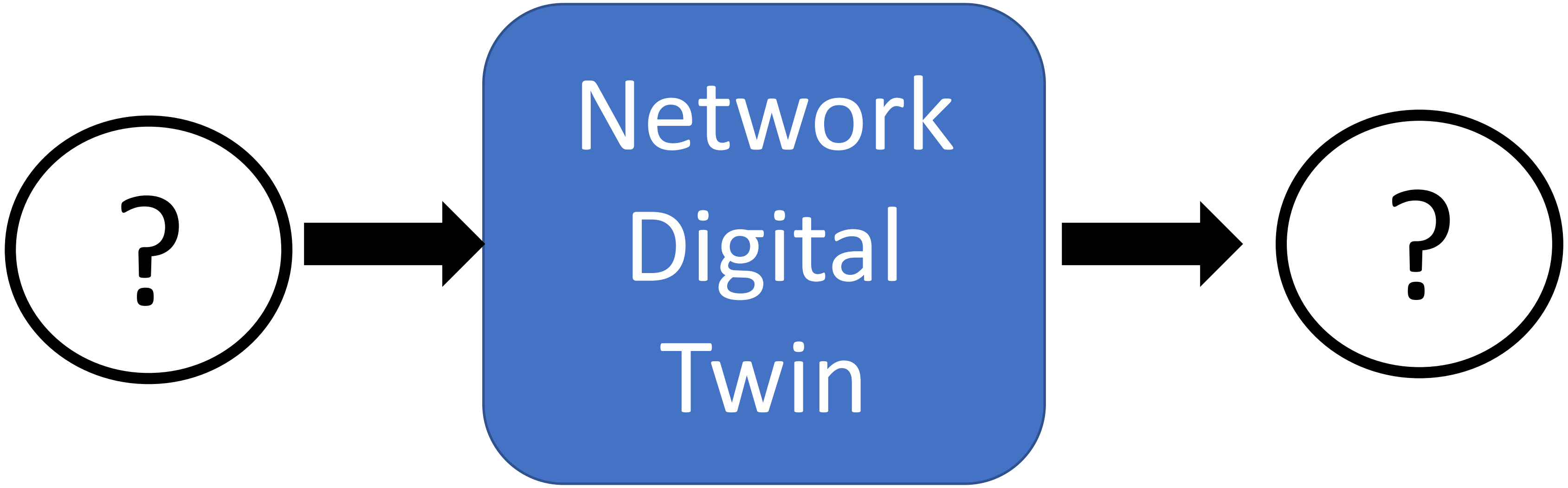


Almasan P, Ferriol-Galmés M, Paillisse J, Suárez-Varela J, Perino D, López D, Perales AA, Harvey P, Ciavaglia L, Wong L, Ram V. Digital Twin Network: Opportunities and Challenges. arXiv preprint arXiv:2201.01144. 2022 Jan 4.

How to build a Network Digital Twin?

What are the inputs and outputs?

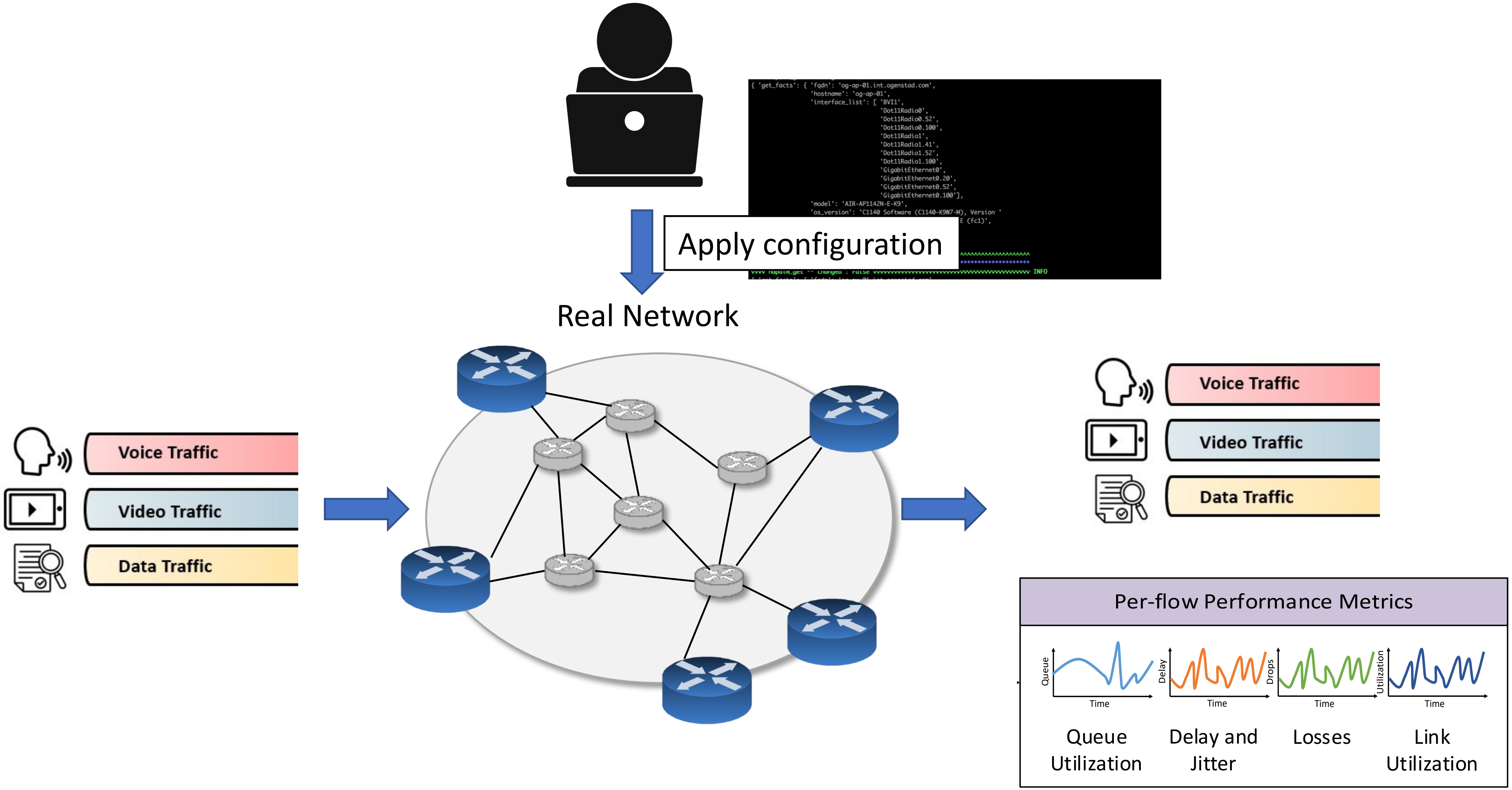
?



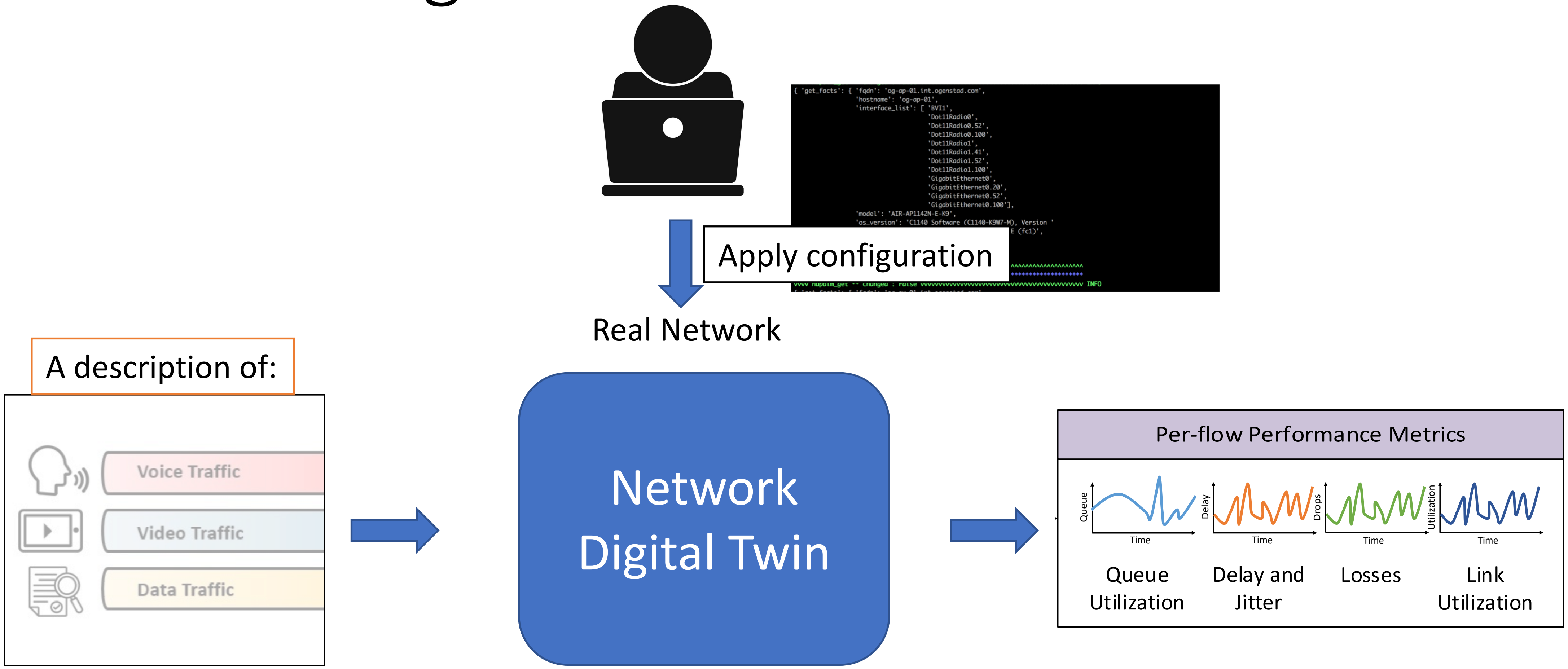
Before discussing how to build the Digital Twin, we need to clearly define the inputs and outputs

Network Digital Twin that focuses on **peformance**

Network as a black-box



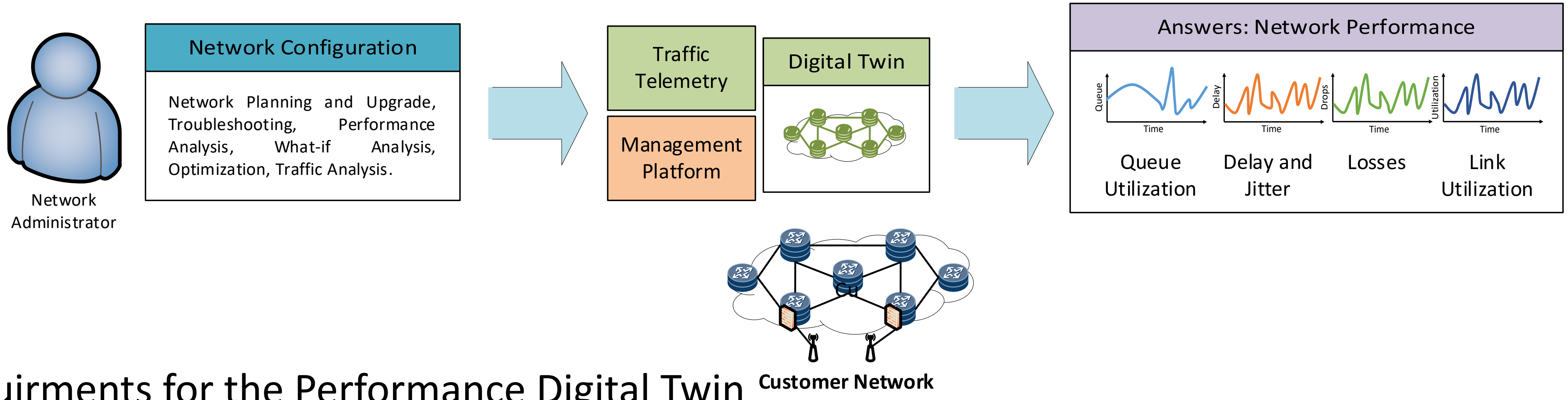
Network Digital Twin



Note that configuration and traffic con be obtained via a standard Telemetry and Management platforms

Use-Cases for Network Digital Twin

Use-cases of the Performance Network Digital Twin (II)



- Requirments for the Performance Digital Twin
  - **Fast and Accurate**
- What-if
  - **What** will be the impact on the network load **if** we acquire Company X?
  - **What** is the optimal network upgrade to support a new set of users?
- Optimization
  - How can I support new user SLAs with the same resources?

More use-cases at: Almasan P, Ferriol-Galmés M, Paillisse J, Suárez-Varela J, Perino D, López D, Perales AA, Harvey P, Ciavaglia L, Wong L, Ram V. Digital Twin Network: Opportunities and Challenges. arXiv preprint arXiv:2201.01144. 2022 Jan 4. https://arxiv.org/abs/2201.01144

How to build a Network Digital Twin?

Network Digital Twin

Configuration

Topology, Link Capacity
Routing
- Overlay: SRv6, MPLS…
- Underlay: OSPF, BGP…

Scheduling Policy (arbitrary)
- Queue Length
- Policy
- Hierarchy of policies

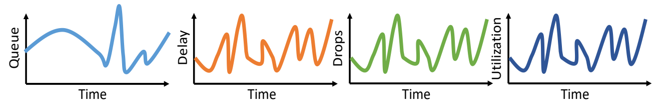
ECMP, LAG, etc

Traffic Load

Traffic Matrix
Start/End Flow
Flow Model (VoIP, VoD, Web, etc)
- Inter-arrival time
- Size distribution

Network Digital Twin

Per-flow Performance Metrics



Queue Utilization | Delay and Jitter | Losses | Link Utilization

Building a Network Digital Twin using: Simulation

Building a Digital Twin with a Simulator

Configuration

Topology, Link Capacity
Routing
- Overlay: SRv6, MPLS…
- Underlay: OSPF, BGP…

Scheduling Policy (arbitrary)
- Queue Length
- Policy
- Hierarchy of policies

ECMP, LAG, etc

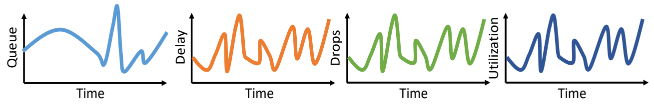
Traffic Load

Traffic Matrix
Start/End Flow
Flow Model (VoIP, VoD, Web, etc)
- Inter-arrival time
- Size distribution

Network Digital Twin

```
#include <stdio.h>
#include <unistd.h>

int main ()
{
        printf("Running 'net join' with the following parameters: \n");
        char *domain="mydomain";
        char *user="domainjoinuser";
        char *pass="mypassword";
        char *vastool="/opt/quest/bin/vastool";
        char *ou="OU=test,DC=mtdomian,DC=local";
        char unjoin[512];

        sprintf(unjoin,"/opt/quest/bin/vastool -u %s -w '%s' unjoin -f",user,pass);

        printf("Domain: %s\n",domain);
        printf("User: %s\n",user);
        printf("----------------\n");

        printf("\nUnjoin...........\n");
        system(unjoin);

        printf("\nJoin.............\n");
        execl("/opt/quest/bin/vastool", "vastool", "-u", user, "-w", pass, "join", "-c", ou,
"-f", domain, (char*)0);

}
```

Per-flow Performance Metrics



Queue Utilization | Delay and Jitter | Losses | Link Utilization

Network Digital Twin is built using a network simulator

Building a Digital Twin with a Simulator

- We have built a Digital Twin using the OMNET++ simulator
- This is a discrete-event simulator
  - It simulates the propagation, transmission and forwarding of each and every packet
  - Other well-known discrete-event simulator are NS2/3, GN3, Cisco packet tracer
- Is it **fast** and **accurate**?
  - Accuracy = Delay measured at the real network vs delay measured at the simulator
  - Accuracy is expressed as **Error in %**
  - **What about the simulation time?**

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Building a Digital Twin with a Simulator

- We compare the delay obtained with the **simulator** to that of a **real network**

Dataset	MAPE	MAE (μs)	R^2
TREX Synthetic	54.167%	80.682	0.716
TREX Multi-Burst	46.911%	62.075	0.997

Table 1: Differences in average packet delay per-flow per windows between scenarios run with OMNeT++ [13] simulator and testbed.

- Accuracy is **very low**
- With enough coding effort, you can get the error close to 0
- **But…**

Building a Digital Twin with a Simulator

- Simulation time scales linearly with the number of packets (discrete-events)
- 1 billion packets takes **11h** (Xeon E, 64GB RAM) of CPU time
- Roughly equivalent to **1min** of a single 10Gbps link

Simulation time (Y) vs. Number of packets (X)

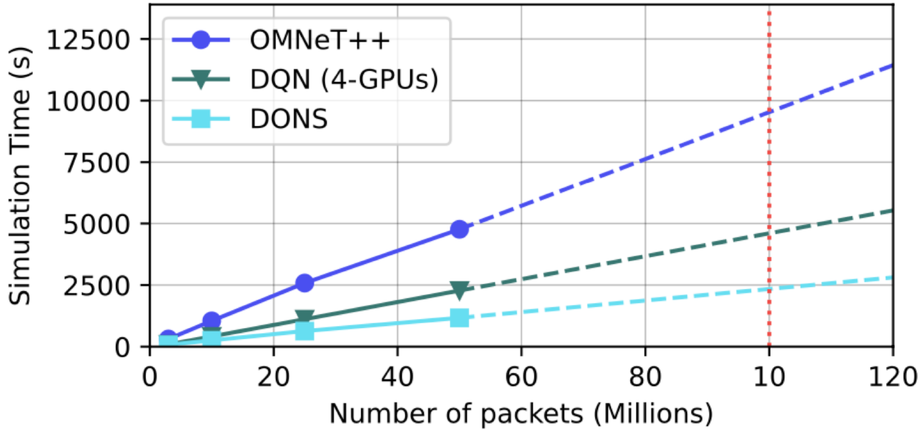
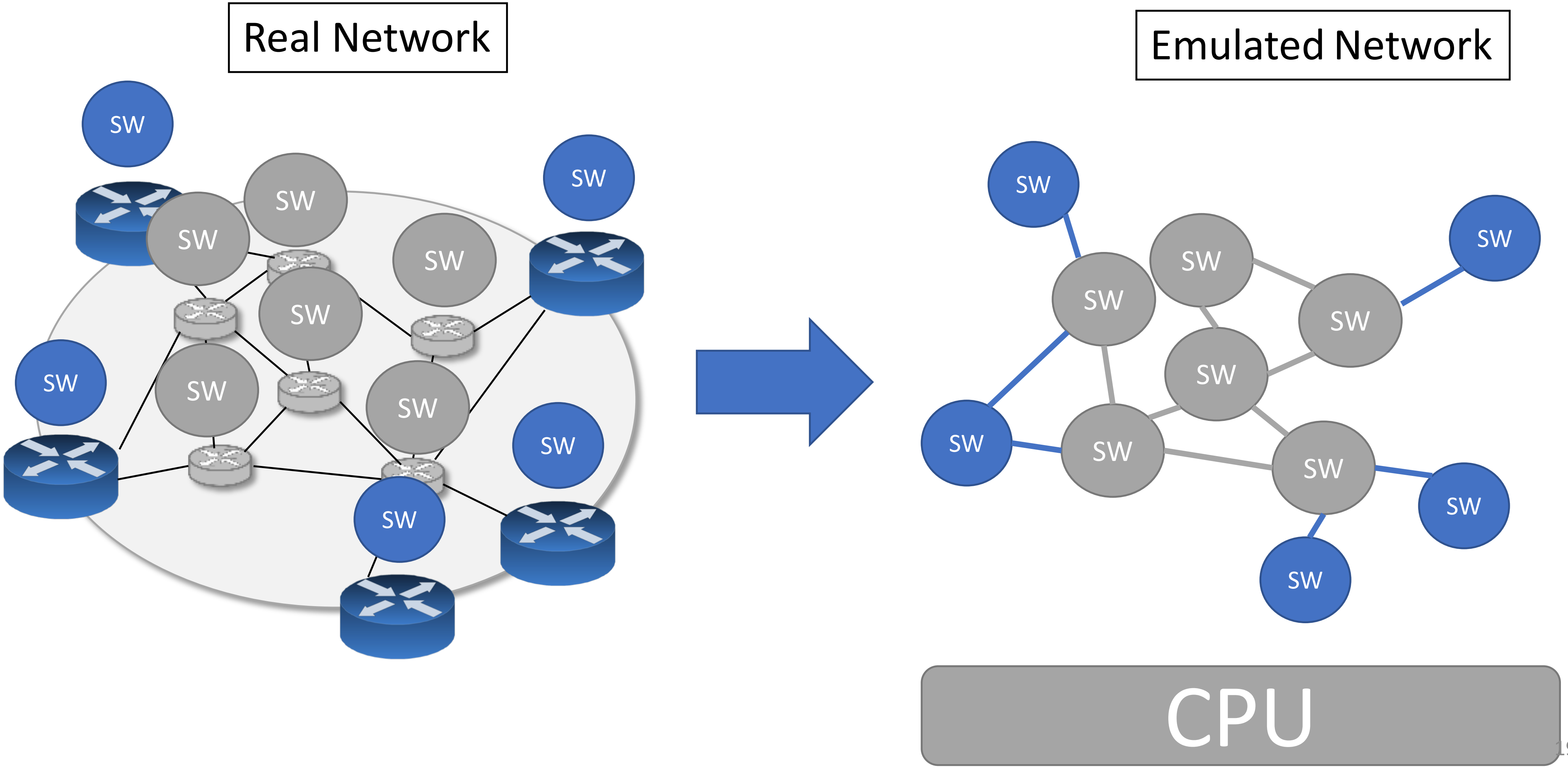


Figure 2: Relationship between the simulation cost and number of packets in the network scenario.

It is impractical to build a Network Digital Twin using a Discrete-Event Simulator because of its high computational cost

Building a Network Digital Twin using: Emulation

Building a Network Digital Twin using **Emulation**



Building a Network Digital Twin using <u>Emulation</u>

- Poor accuracy of network emulation
  - Because emulation does not use specific hardware built for networking
  - If your network infrastructure is already fully virtualized, then emulating it requires as many resources as running the real one
    - Otherwise peformance will be lower
- Emulation has many relevant use-cases
  - Training
  - Debugging (why my SYN packets are being dropped)
  - Testing new features (what happens if I actívate this feature?)

It is impractical to build a Network Digital Twin using emulation
<u>because of it offers very low speed</u>

Lochin, Emmanuel, Tanguy Perennou, and Laurent Dairaine. "When should I use network emulation?." *annals of telecommunications-annales des télécommunications* 67, no. 5 (2012): 247-255.

Building a Network Digital Twin using: Queuing Theory

Building a Network Digital Twin using Queuing Theory

- Queing Theory represents our **<u>best available</u>** analytical tool for computer networks modelling.
- It models the network as a series of queues serviced by routers

Leonard Kleinrock pioneered the application of QT to packet-switched network in the 70s.

Sundarapandian, V. (2009). "7. Queueing Theory". *Probability, Statistics and Queueing Theory*. PHI Learning. ISBN 978-8120338449.

Building a Network Digital Twin using Queuing Theory

Configuration

Topology, Link Capacity
Routing
- Overlay: SRv6, MPLS…
- Underlay: OSPF, BGP…

Scheduling Policy (arbitrary)
- Queue Length
- Policy
- Hierarchy of policies

ECMP, LAG, etc

Traffic Load

Traffic Matrix
Start/End Flow
Flow Model (VoIP, VoD, Web, etc)
- Inter-arrival time
- Size distribution

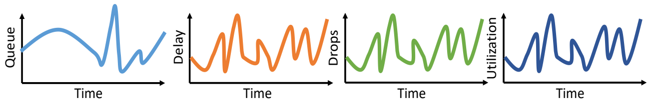
Performance Network Digital Twin

$$\boldsymbol{Q}[(0,q),(i,q_{i+})] = \lambda_i \quad 0 < i \leq p \tag{4}$$
$$\boldsymbol{Q}[(s,q),(s,q_{i+})] = \lambda_i I_{q_i<b_i} \quad 0 < i \leq p \tag{5}$$
$$\boldsymbol{Q}[(s,q^0_{s+}),(0,q^0)] = \mu_s \quad 0 < s \leq p \tag{6}$$
$$\boldsymbol{Q}[(s,q),(\min\{i : q_i > 0\}, q_{s-})] = \mu_s \tag{7}$$

$$\boldsymbol{Q}[q,q_{i+}] = \lambda_i I_{q_i<b_i} \quad 0 < i \leq p \tag{8}$$
$$\boldsymbol{Q}[q,q_{i-}] = \frac{I_{q_i>0} w_i}{\sum_{q_i>0} w_i} \mu_i \quad 0 < i \leq p \tag{9}$$

Network Digital Twin is built using a equations

Per-flow Performance Metrics



Queue Utilization | Delay and Jitter | Losses | Link Utilization

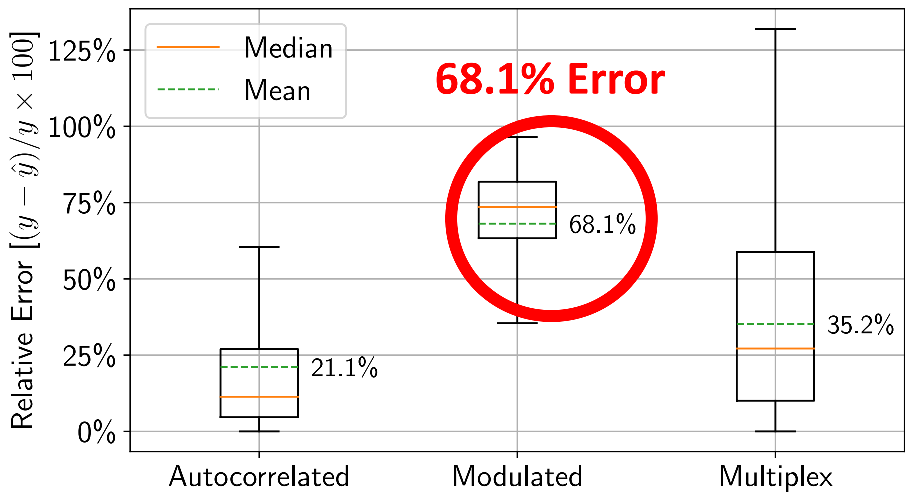
Ferriol-Galmés, M., Rusek, K., Suárez-Varela, J., Xiao, S., Cheng, X., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). RouteNet-Erlang: A Graph Neural Network for Network Performance Evaluation. *In Proc. Of IEEE INFOCOM 2022 https://arxiv.org/abs/2202.13956*

Building a Network Digital Twin using Queuing Theory

- The QT Digital Twin is fast (milliseconds)
- QT Digital Twin scales linearly with the number of queues.
  - It can support real-world networks
- The main limitation with QT is that it has **poor accuracy under realistic traffic models**
- This is a well-known limitation

It is impractical to build a Network Digital Twin using QT **because it is not accurate with realistic traffic**

Error (lower is better) when predicting the performance of flows with different traffic models.



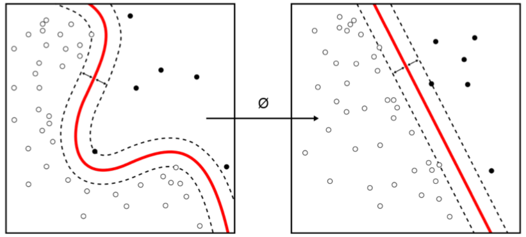
Modulated is roughly equivalent to TCP traffic

Building a Network Digital Twin using: Graph Neural Networks

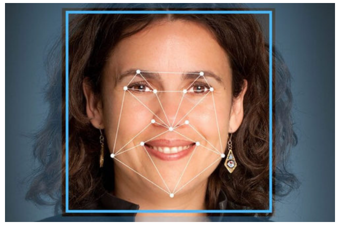
What are
Graph Neural Networks?

Overview of the most common NN architectures

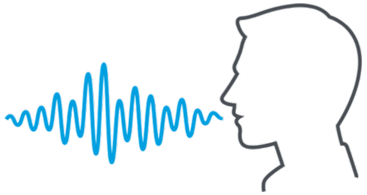
Type of NN	Information Structure
Fully Connected NN	Arbitrary
Convolutional NN	Spatial
Recurrent NN	Sequential
Graph NN	Relational



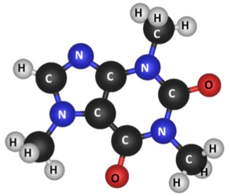
Classification, Unsupervised Learning



Images and video



Text and voice

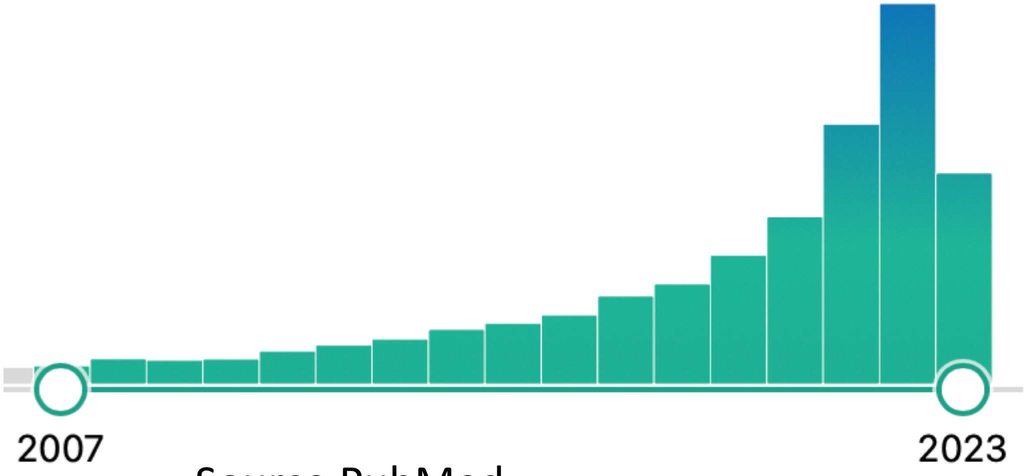


Graphs (molecules, maps, networks)

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261*(2018).

GNN are a hot topic in AI

- Top trends in Graph Machine Learning in 2020 “New cool applications of GNN” [1]
- “Machine Learning on graphs becomes a first-class citizen at AI conferences” [2] (NeurIPS 2019)
- Graph Neural Networking Challenge organized by ITU-T received more than 120 participants from over 27 countries [3]
- GNN helped solved long-standing problems: AlphaFold [4]

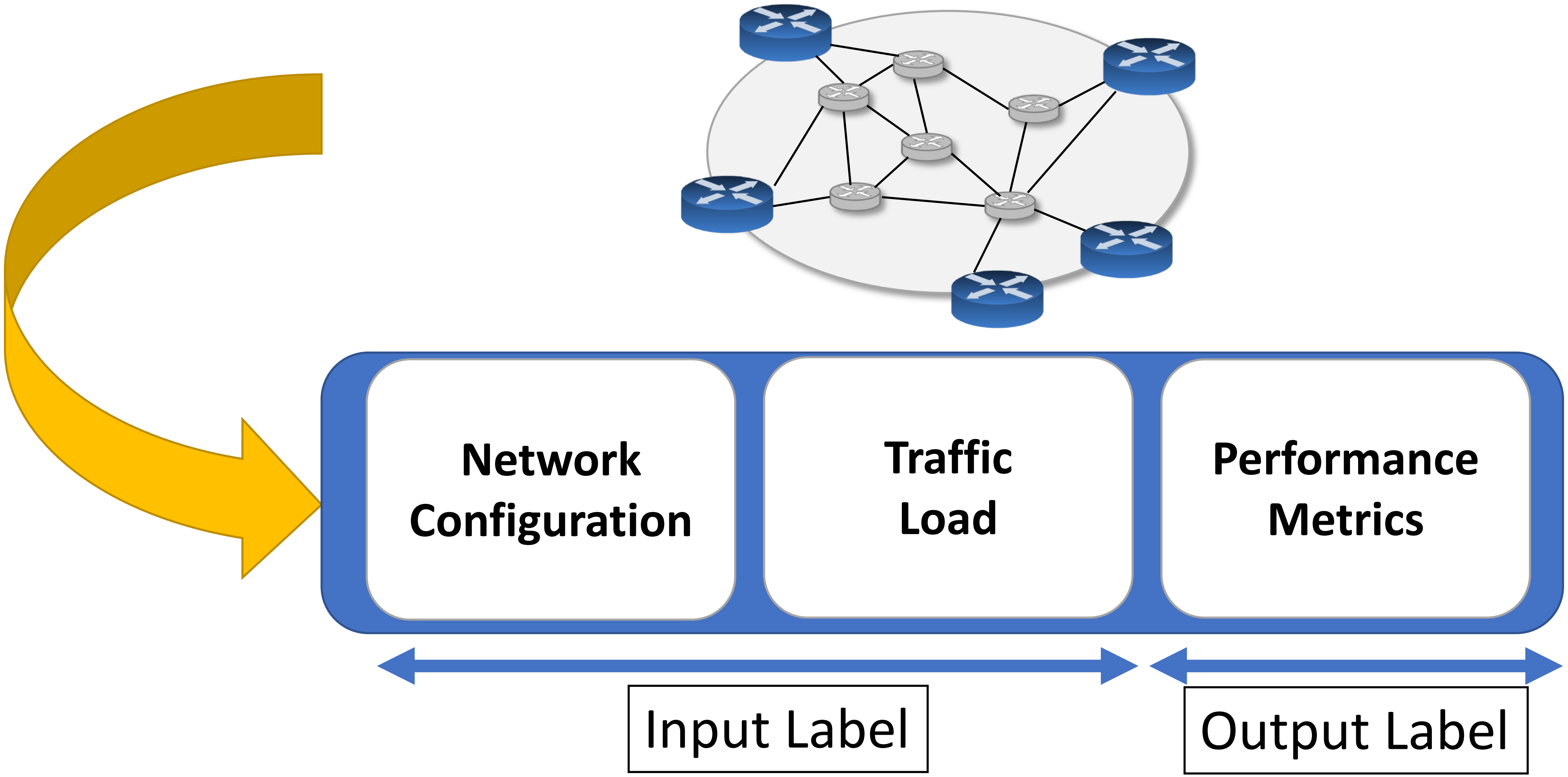


Source PubMed
Keyword: Graph Neural Network

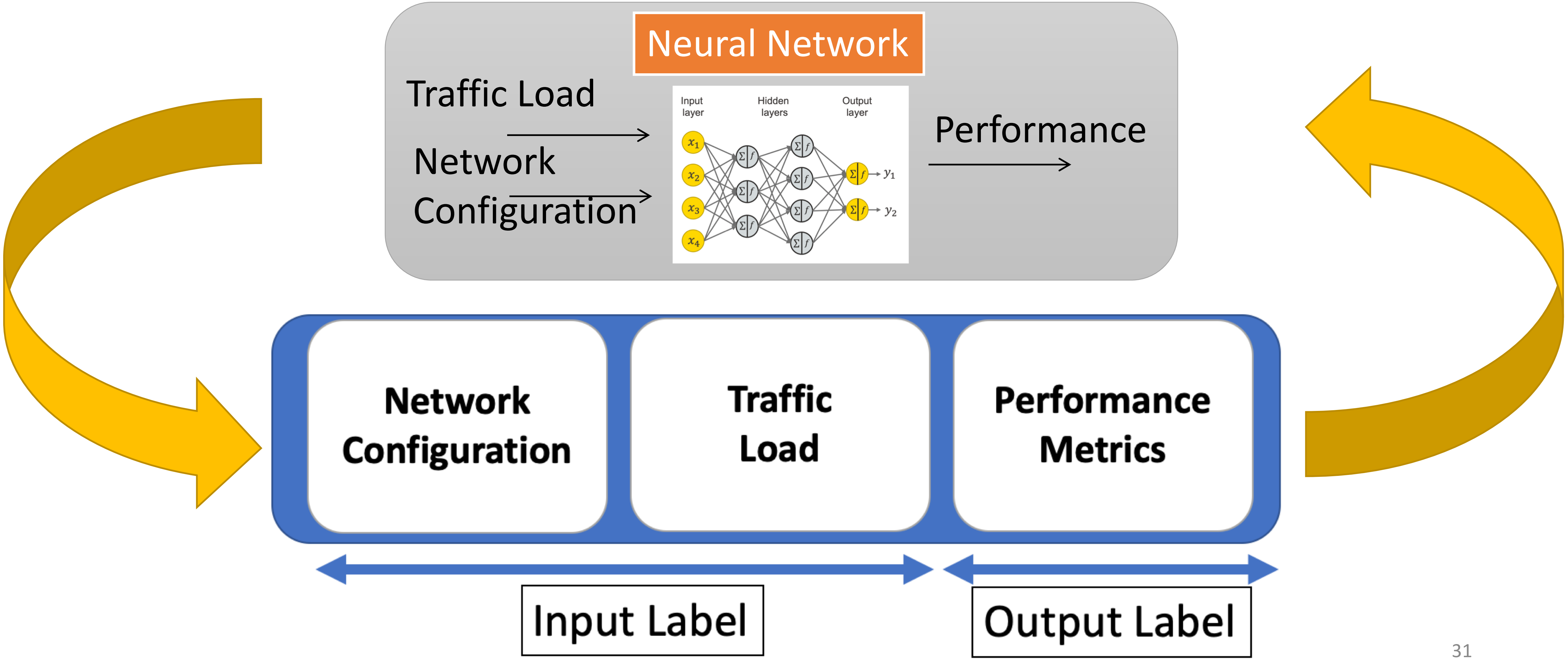
[1] https://towardsdatascience.com/top-trends-of-graph-machine-learning-in-2020-1194175351a3
[2] https://medium.com/mlreview/machine-learning-on-graphs-neurips-2019-875eecd41069
[3] https://www.itu.int/en/ITU-T/AI/challenge/2020/Pages/default.aspx
[4] https://alphafold.ebi.ac.uk

RouteNet: A Network Digital Twin that uses Graph Neural Networks

Building a Network Digital Twin using Neural Nets



Building a Network Digital Twin using Neural Nets



Building a Network Digital Twin using GNNs

Flows (time-series)

Start/End Flow
Flow Model (VoIP, VoD, Web, etc)
- Inter-arrival time
- Flow-packet distribution

Configuration

Topology, Link Capacity
- **100x larger than training**

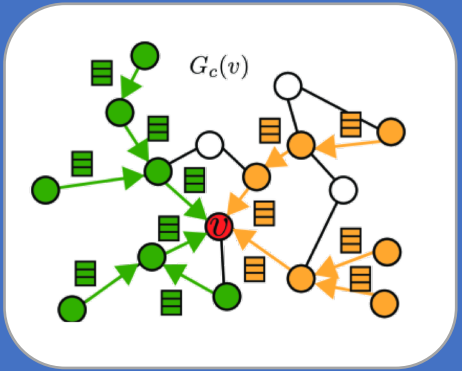
Routing
- Overlay: SRv6, MPLS…
- Underlay: OSPF, BGP…

Scheduling Policy (arbitrary)
- Queue Length
- Policy
- Hierarchy of policies

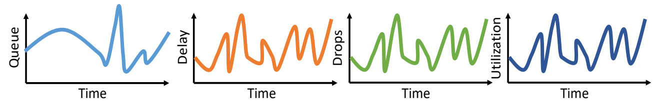
ECMP, LAG, etc

RouteNet: An open-source NDT

Network Digital Twin



Per-flow Performance Metrics

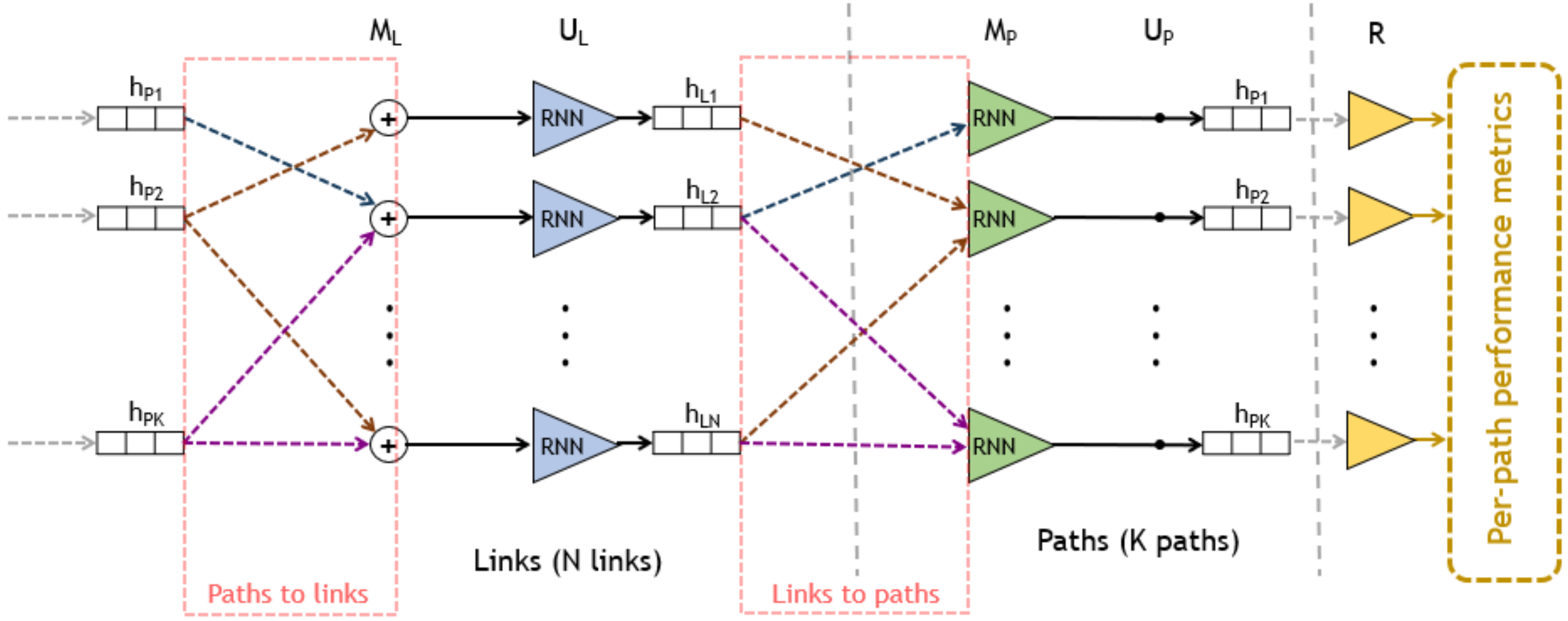


Queue Utilization | Delay and Jitter | Losses | Link Utilization

Network Digital Twin is built using a GNN
RouteNet: A custom built Graph Neural Network architecture tailored for computer networks

Ferriol-Galmés, M., Rusek, K., Suárez-Varela, J., Xiao, S., Cheng, X., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). RouteNet-Erlang: A Graph Neural Network for Network Performance Evaluation. *In Proc. Of IEEE INFOCOM 2022 https://arxiv.org/abs/2202.13956*

RouteNet's Architecture (simplified)



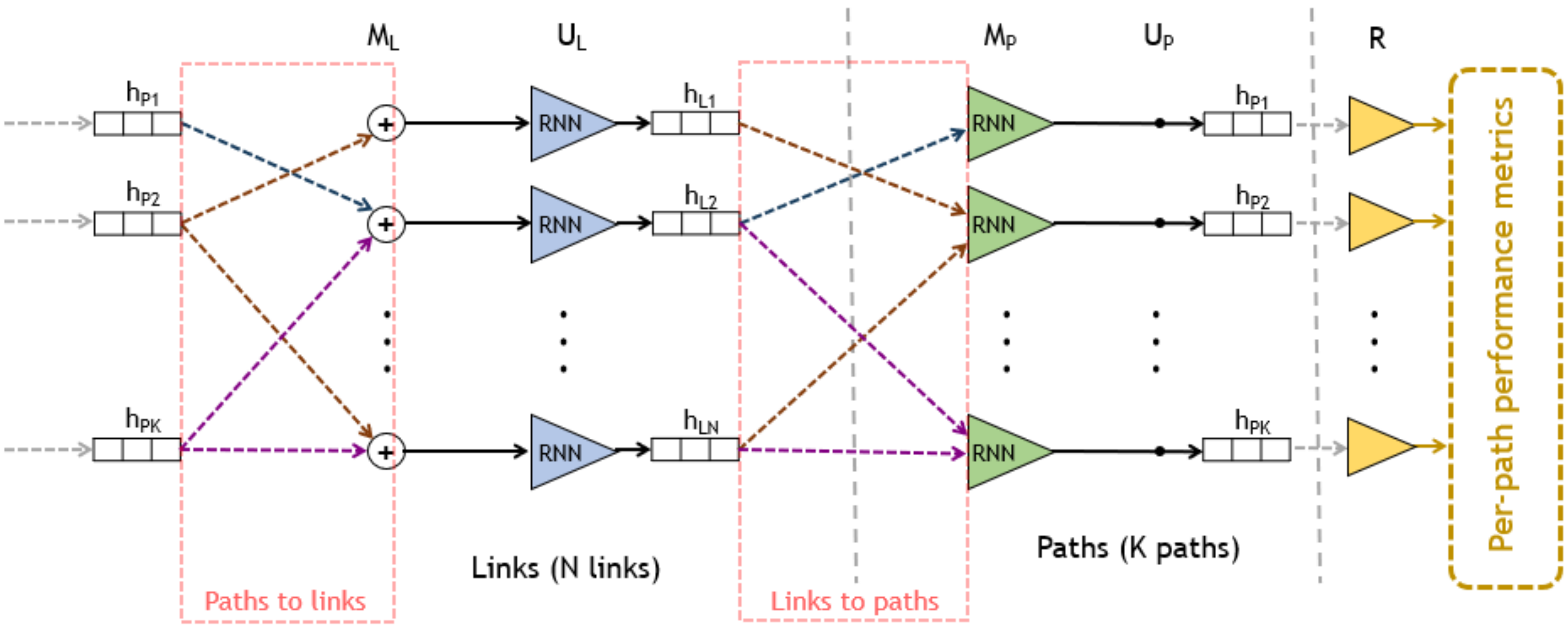
Input: $\mathbf{x}_p$, $\mathbf{x}_l$, $\mathcal{R}$
Output: $\mathbf{h}_p^T$, $\mathbf{h}_l^T$, $\hat{\mathbf{y}}_p$

foreach $p \in \mathcal{R}$ **do**
  $\mathbf{h}_p^0 \leftarrow [\mathbf{x}_p, 0 \ldots, 0]$
end
foreach $l \in \mathcal{N}$ **do**
  $\mathbf{h}_l^0 \leftarrow [\mathbf{x}_l, 0 \ldots, 0]$
end
for $t = 1$ *to* T **do**
  foreach $p \in \mathcal{R}$ **do**
    foreach $l \in p$ **do**
      $\mathbf{h}_p^t \leftarrow RNN_t(\mathbf{h}_p^t, \mathbf{h}_l^t)$
      $\tilde{\mathbf{m}}_{p,l}^{t+1} \leftarrow \mathbf{h}_p^t$
    end
    $\mathbf{h}_p^{t+1} \leftarrow \mathbf{h}_p^t$
  end
  foreach $l \in \mathcal{N}$ **do**
    $\mathbf{m}_l^{t+1} \leftarrow \sum_{p:l \in p} \tilde{\mathbf{m}}_{p,l}^{t+1}$
    $\mathbf{h}_l^{t+1} \leftarrow U_t\left(\mathbf{h}_l^t, \mathbf{m}_l^{t+1}\right)$
  end
end
$\hat{\mathbf{y}}_p \leftarrow F_p(\mathbf{h}_p)$

- RouteNet models the relationship between links and paths
  - State of a links depends on the paths that traverse that link
  - State of a paths depends on the links of that path
- This is a circular dependency

$$\mathbf{h}_{l_i} = f(\mathbf{h}_{p_1}, \ldots, \mathbf{h}_{p_j}), \quad l_i \in p_k, k = 1, \ldots, j$$
$$\mathbf{h}_{p_k} = g(\mathbf{h}_{l_{k(0)}}, \ldots, \mathbf{h}_{l_{k(|p_k|)}})$$

RouteNet's Architecture (simplified)



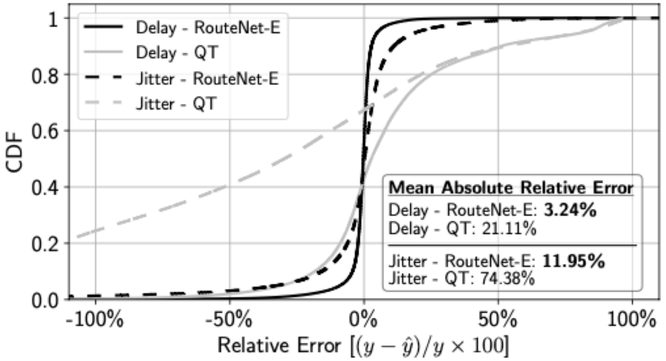
Input: $\mathbf{x}_p, \mathbf{x}_l, \mathcal{R}$
Output: $\mathbf{h}_p^T, \mathbf{h}_l^T, \hat{\mathbf{y}}_p$
foreach $p \in \mathcal{R}$ **do**
  $\mathbf{h}_p^0 \leftarrow [\mathbf{x}_p, 0 \ldots, 0]$
end
foreach $l \in \mathcal{N}$ **do**
  $\mathbf{h}_l^0 \leftarrow [\mathbf{x}_l, 0 \ldots, 0]$
end
for $t = 1$ *to* T **do**
  foreach $p \in \mathcal{R}$ **do**
    foreach $l \in p$ **do**
      $\mathbf{h}_p^t \leftarrow RNN_t(\mathbf{h}_p^t, \mathbf{h}_l^t)$
      $\tilde{\mathbf{m}}_{p,l}^{t+1} \leftarrow \mathbf{h}_p^t$
    end
    $\mathbf{h}_p^{t+1} \leftarrow \mathbf{h}_p^t$
  end
  foreach $l \in \mathcal{N}$ **do**
    $\mathbf{m}_l^{t+1} \leftarrow \sum_{p:l \in p} \tilde{\mathbf{m}}_{p,l}^{t+1}$
    $\mathbf{h}_l^{t+1} \leftarrow U_t\left(\mathbf{h}_l^t, \mathbf{m}_l^{t+1}\right)$
  end
end
$\hat{\mathbf{y}}_p \leftarrow F_p(\mathbf{h}_p)$

Key insight: Graph Neural Networks are not a **black box.**
Custom GNN architectures need to be research to tackle different networking problems.

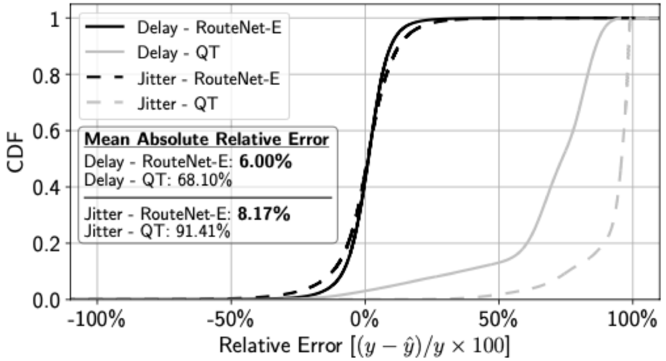
$$\mathbf{h}_{l_i} = f(\mathbf{h}_{p_1}, \ldots, \mathbf{h}_{p_j}), \quad l_i \in p_k, k = 1, \ldots, j$$
$$\mathbf{h}_{p_k} = g(\mathbf{h}_{l_{k(0)}}, \ldots, \mathbf{h}_{l_{k(|p_k|)}})$$

Building a Network Digital Twin using GNNs

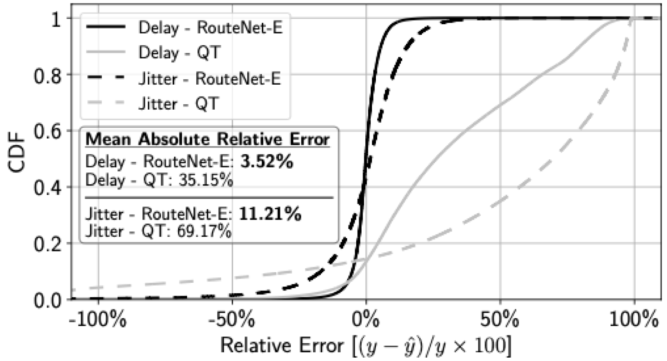
- RouteNet achieves remarkable accuracy under arbitrary traffic models



(d) Autocorrelated exponentials



(e) Modulated exponentials

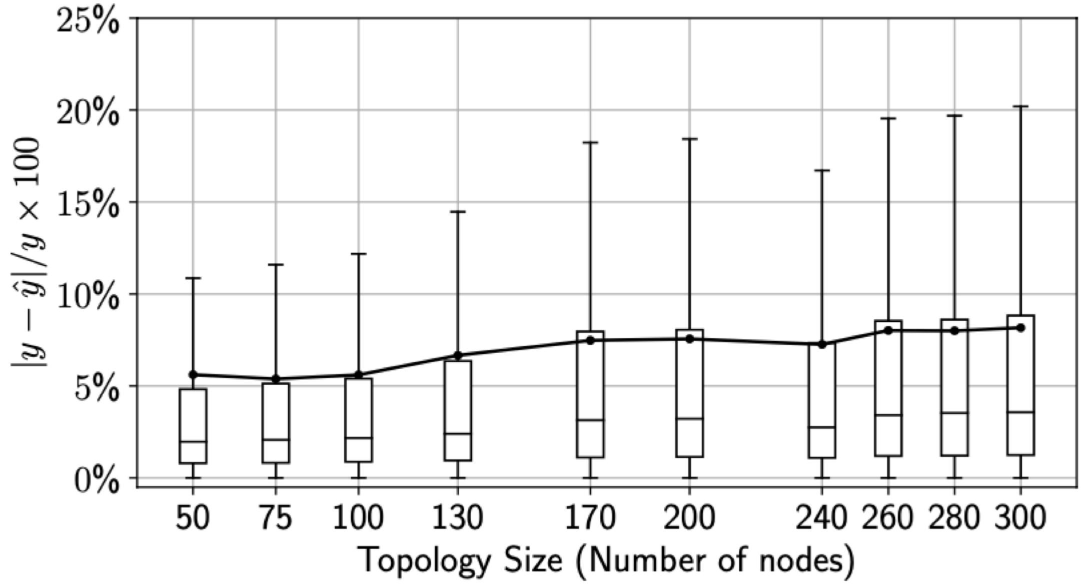


(f) All traffic models multiplexed

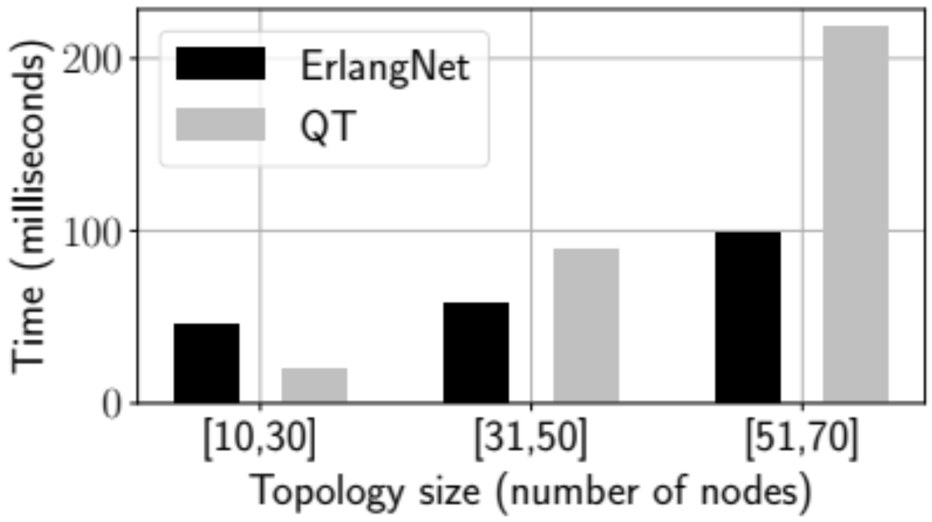
- Error when estimating the delay is <10%

Building a Network Digital Twin using GNNs

- RouteNet can operate in **networks not seen in training**
- RouteNet can scale to networks up to **100x larger** that the ones seen in training



- RouteNet speed: milliseconds



Building a Network Digital Twin using data from Real Networks

https://bnn.upc.edu/challenge/gnnet2023

Cash Prizes:

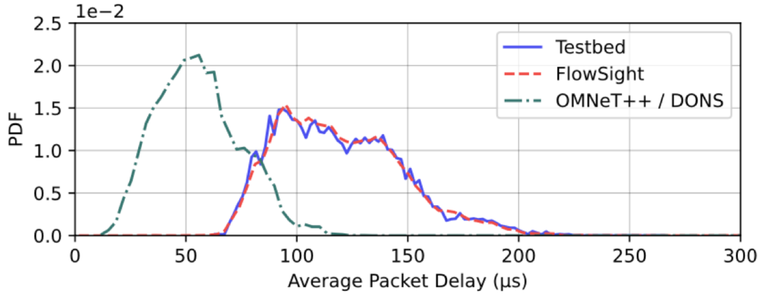
1st Prize: 2000 EUR

2nd Prize: 500 EUR

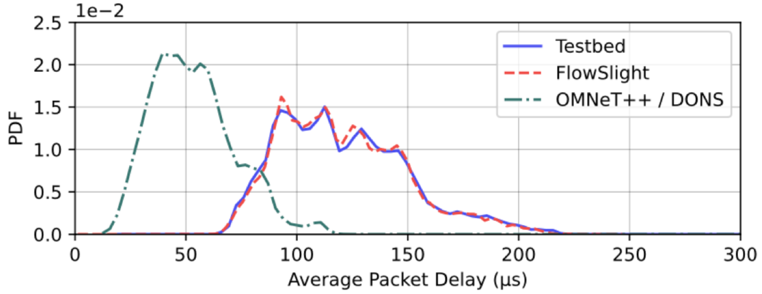
- In 2023 we organized a challenge along with ITU
- Build the **first** Network Digital Twin using data from a **real network**
- Input traffic are realistic packet traces
- Baseline was **RouteNet**

Building a Network Digital Twin using GNNs

- RouteNet was extended to support **packet-traces** as input
- No additional changes to the architecture required
- Provides remarkable performance with error < 5%



(a) TREX Synthetic



(b) TREX Multi-burst

Conclusions

Conclusions

Technology	Accuracy	Speed	Why?
Emulation	Poor	Slow	Emulation is useful to check for configuration errors or test the interaction between different protocols. **It is not accurate in performance estimation.**
Simulation	Good	Slow	Simulation time scales with the amount of packets, 1min of a 10Gbps link takes 11h to simulate. **It is too slow for performance estimation.**
Analytical Models (Queuing Theory)	Poor	Fast	Fast and accurate, **but does not work well under realistic traffic models (e.g., TCP)**
Neural Nets (MLP and Recurrent NN, see Backup slides)	Poor	Fast	Fast and accurate, **but it does not work in scenarios not seen in training (e.g, Link failure)**
Graph Neural Networks	Good	Fast	GNNs are tailored to learn network-structured data. **They offer oustanding accuracy in scenarios not seen in training.**

Learn: All papers are free online

https://github.com/knowledgedefinednetworking/Papers/wiki

Play: Code and Datasets open-source

https://knowledgedefinednetworking.org

Code: IGNNITION framework

https://ignnition.net/

Building a Network Digital Twin using: <u>**Neural Networks** (MLP and RNN)</u>

Building a Network Digital Twin using Neural Nets

Configuration

Topology, Link Capacity
Routing
- Overlay: SRv6, MPLS…
- Underlay: OSPF, BGP…

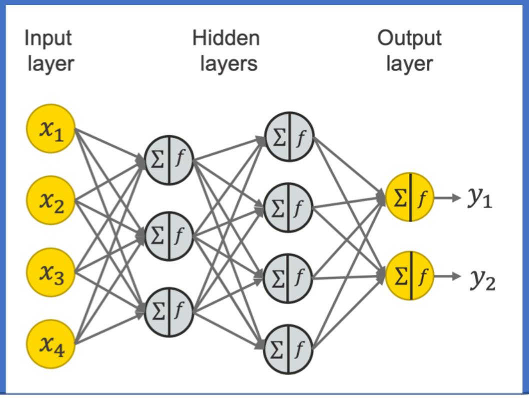
Scheduling Policy (arbitrary)
- Queue Length
- Policy
- Hierarchy of policies

ECMP, LAG, etc

Traffic Load

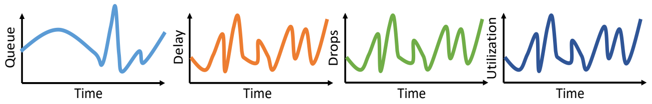
Traffic Matrix
Start/End Flow
Flow Model (VoIP, VoD, Web, etc)
- Inter-arrival time
- Size distribution

Performance Network Digital Twin



Network Digital Twin is built using a neural network

Per-flow Performance Metrics



Queue Utilization | Delay and Jitter | Losses | Link Utilization

Building a Network Digital Twin using Neural Nets

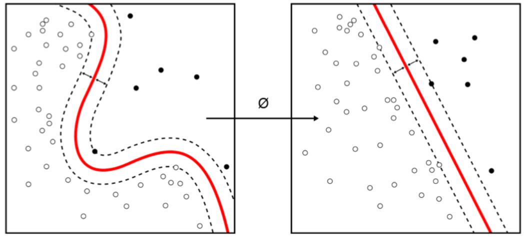
- Both RNN and MLP are fast (milliseconds)
- They scale –roughly- constantly (O(1)) with all network parameters
- They offer poor accuracy when operating in configurations (routing, link failures) not seen in training

	Accuracy Error (MAPE) when estimating the delay. Percentage error of the real vs. predicted value	
	MLP (Fully-connected)	Recurrent NN
Same routing as in training	12.3%	10.0%
Different routing as in training	1150%	30.5%
Link Failure	125%	63.8%

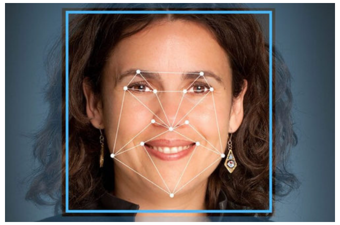
It is impractical to build a Network Digital Twin using MLPs and RNNs **because they do not support different network topologies, routing or link-failures**

Overview of the most common NN architectures

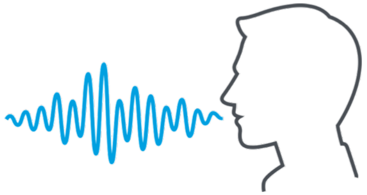
Type of NN	Information Structure
Fully Connected NN (e.g., MLP)	Arbitrary
Convolutional NN	Spatial
Recurrent NN	Sequential
Graph NN	Relational



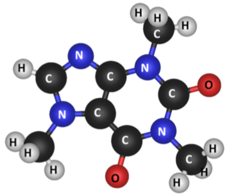
Classification, Unsupervised Learning



Images and video



Text and voice



Graphs (molecules, maps, networks)

Overview of the most common NN architectures

Type of NN	Information Structure
Fully Connected NN (e.g., MLP)	Arbitrary
Con	
Recurrent NN	Sequential
Graph NN	Relational

Classification, Unsupervised Learning

eo

Text and voice

Graphs (molecules, maps, networks)

RNNs, MLPs and CNNs are unable to understand information structured as a network