Forging Reliable Edge Services: Harnessing Deep Learning Models for Fault Tolerance

Giuliano Casale

g.casale@imperial.ac.uk

Quality of Service Research Lab

Department of Computing - Imperial College London

IEEE ISSRE 2023, Florence

Acknowledgements: **Shreshth Tuli**, Nicholas J. Jennings, Alim Ul Gias, Lucy Cherkasova

The rise of Edge computing

Emerging edge applications:

- Autonomous driving
- AR/VR, edge gaming
- Smart cities, ports, farming, …
- Industrial IoT

Strict QoS/QoE requirements:

- Ultra-low latency
- High availability
- Zero perceived downtime
- Reliability often quoted as a pressing challenge!

NETWORKWORLD

OPINION

Industrial IoT faces big challenges

Industrial IoT needs ultra-high reliability, always-on availability, and extremely low latency – as well as standardization – all of which makes it the most challenging IoT genre to implement.

HELP NET SECURITY

Multi-cloud and edge deployments threatened by security and connectivity problems

The Growing Importance of AI and Automation in Building and Maintaining 5G Networks

OPP.TODAY

IoT and Edge Devices

- Millions of tiny and embedded devices

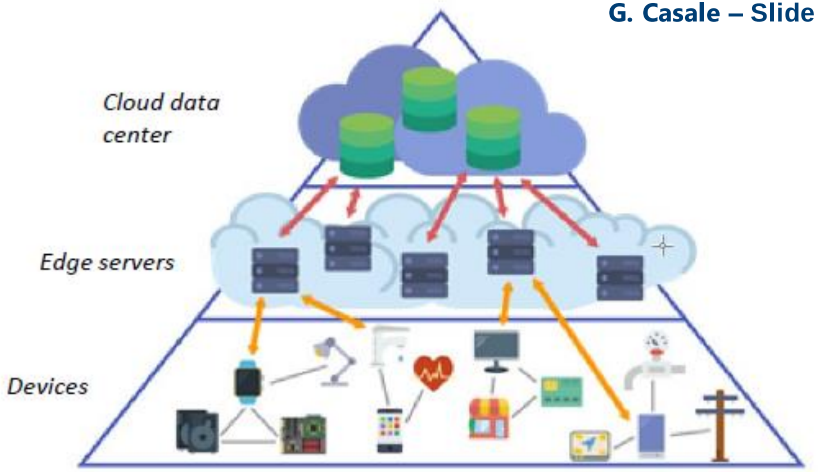
- Rapid growth in edge AI hardware
  - Edge accelerators (VPUs, TPUs)
  - Efficiency: when normalized for power and cost, comparable to server GPUs
  - On-device learning can benefit edge QoS management

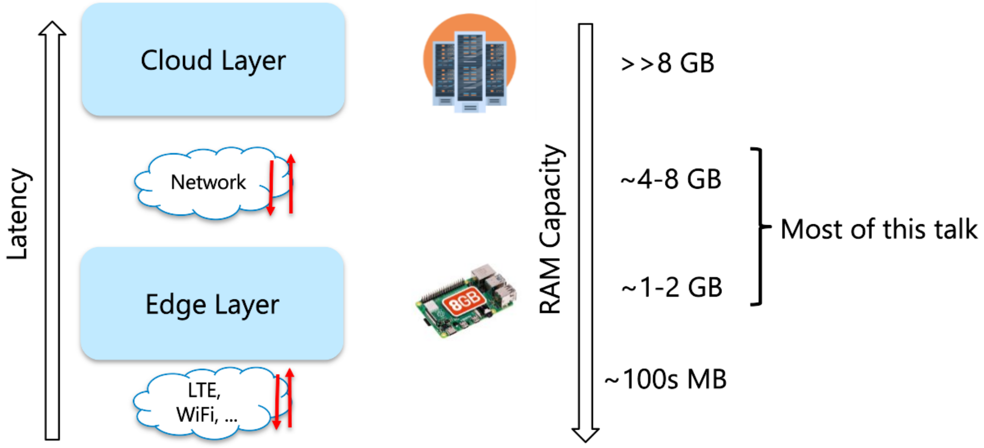
- It's an exciting time to investigate what AI can do for edge reliability!

Ref: Liang, Shenoy, Irwin, AI on the Edge: Characterizing AI-based IoT Applications Using Specialized Edge Architectures, IISWC, 2020.

Basic Concepts

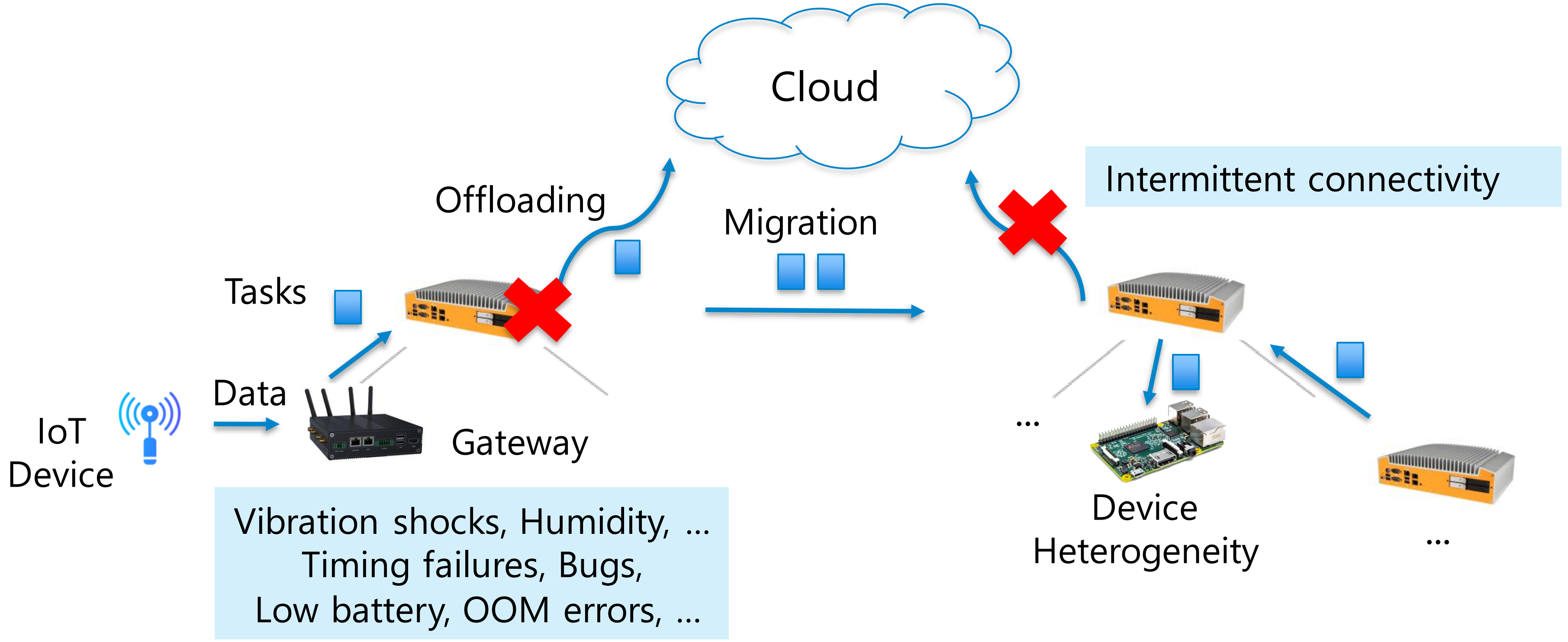
- Many architectures and platforms
  - Fog, MEC, "path computing", cloudlets, private vs public edge, …
- Common challenges and themes:
  - Processing data closer to where it is generated
  - Latency vs. resource constraints
  - Need for a high degree of automation





Edge Reliability: Why this is not a solved problem?

- The edge is a dynamic and failure-rich environment



Taxonomy of Fault Tolerance (FT) Techniques

Reactive

- Checkpoint restart
- Replication
- Resubmission
- …

Proactive

- Preemptive migration
- Self-healing
- Rejuvenation
- …

Offline history DB

Edge-layer Deep NNs?

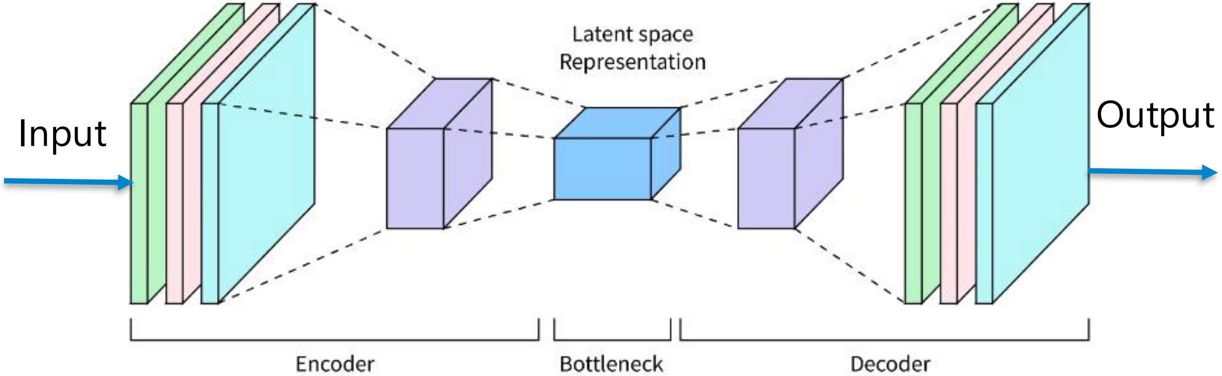
Levels of FT intelligence

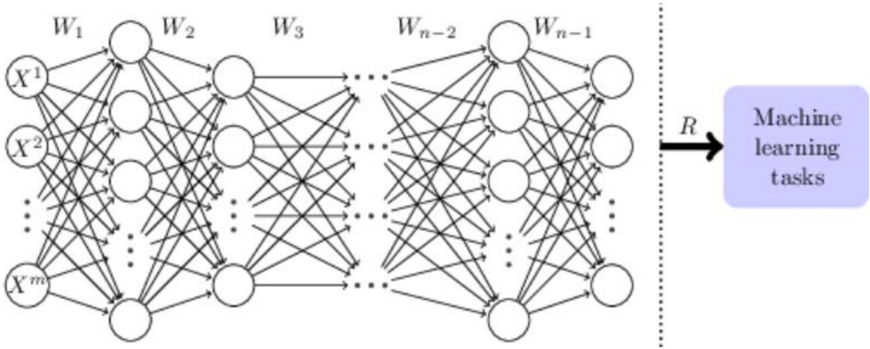
Monitoring → Fault-detection → Local analysis → Global analysis → Pattern Learning

Ref. Engelmann *et al.*, Proactive Fault Tolerance Using Preemptive Migration, PDP 2009.

Embedding data in latent space

- A low-dimensional smooth representation
- Many good properties:
  - Abstract representations
  - No feature engineering
  - Expose natural clusters
  - Represent time series context
  - Compress high-dimensional data
  - ...
- Often resource hungry
  - e.g., Transformers: $O(n^2)$ complexity w.r.t. the input length





Talk Outline

1. How are Deep models being used for edge reliability?

⇨ Methods from the literature for fault detection, diagnosis and prediction

2. How can Deep models handle edge-layer spatio-temporal correlations?

⇨ Results on preemptive task migration

3. Resource constraints: how can we reduce memory footprint?

⇨ Results on FT in edge federations

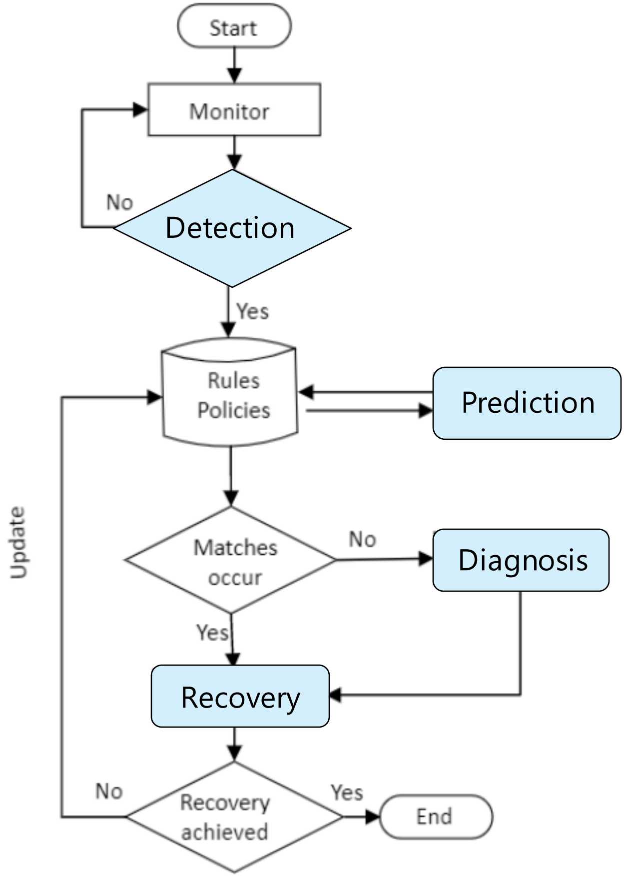
Imperial College London

Fault Detection, Diagnosis and Prediction

FT workflows

- Often an integrated workflow of methods rules, and heuristics
  - Unsupervised clustering
  - Time series forecasting
  - Bayesian networks
  - Ensemble learning

- Recovery is often optimization-based
  - Mathematical programming
  - Stochastic models
  - Metaheuristics

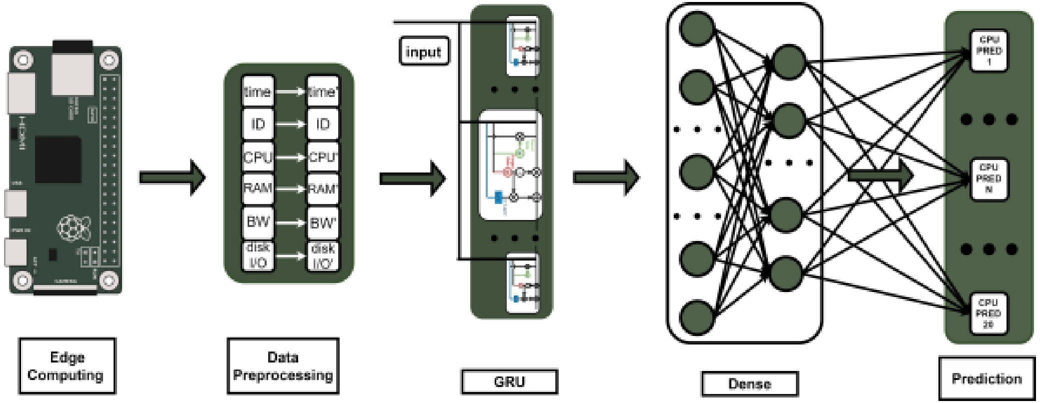


Ref: Adeni et al., Proactive Self-Healing Approaches in Mobile Edge Computing: A Systematic Literature Review, 2023.

On-Device Deep Models for Failure Prediction

- On-device learning (rPI 3)
  - Predict timing failures, OOM errors, congestion
  - LSTM/GRU based
- About 100MB memory
  - Knowledge distillation
  - Quantization
  - Pruning
- ~4-10ms for inference

- Lower RMSE than classic ML methods
  - e.g., SVMs, Boosting

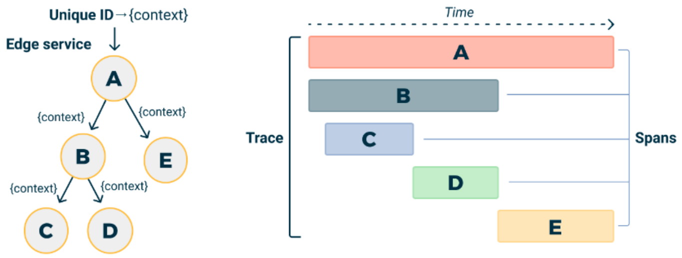


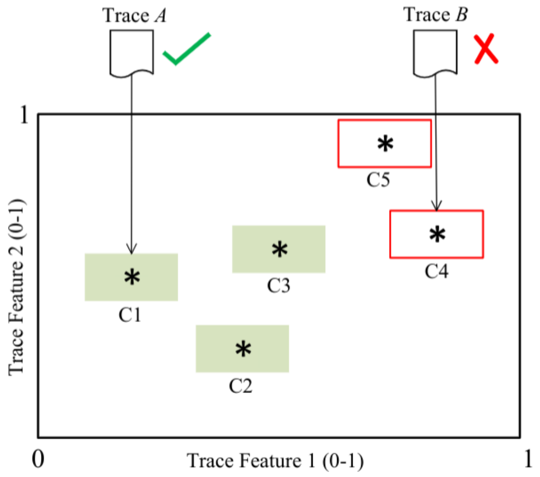
Method	RMSE
HBES-GRU	0.0641
GA-LSTM	0.0674
Keras-Tuner	0.0785
AUCROP	0.0814
XGBoost	0.1139
Auto-sklearn	0.1055

Ref: Violos et al., Hypertuning GRU Neural Networks for Edge Resource Usage Prediction, 2021.

Integrating Deep Models with Tracing

- Edge-layer tracing picking up
  - E.g, Azure IoT Edge, OpenTelemetry Collector, …
- Tail-based sampling can reduce footprint
  - Only sample traces that are 'interesting'
- Embedding methods:
  - Word2vec, graph embedding, DBN, …
- Sample using ML classifier or online clustering
  - High accuracy (often >0.90 F1 score)

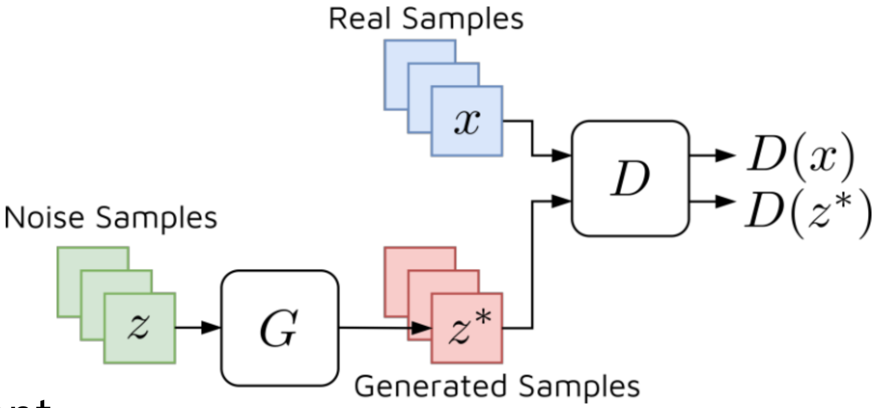




Ref: Gias et al.; SampleHST: Efficient On-the-Fly Selection of Distributed Traces. NOMS 2023.

Integrating Deep Models in Fault Diagnosis

- ML fault classifiers affected by class imbalance problem
  - Overfitting, bias and loss of information issues with random sampling
  - Generative Adversarial Networks (GANs) increasingly adopted as a solution
- GANs for data augmentation
  - Independent GANs for minority and majority class
  - Often boosts F1 (+5%-50%)
  - Discriminator may also replace the ML classifier
- GAN compression for edge deployment



Ref: Zhang *et al.*, PWG-IDS: An Intrusion Detection Model for Solving Class Imbalance in IIoT Networks Using Generative Adversarial Networks, arxiv:2110.03445.

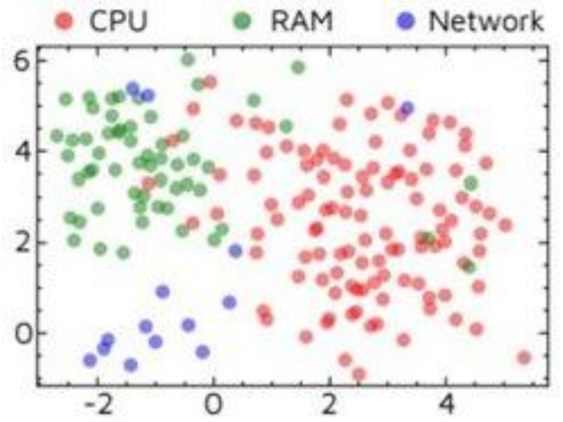
Few-Shot Learning with Prototype Networks

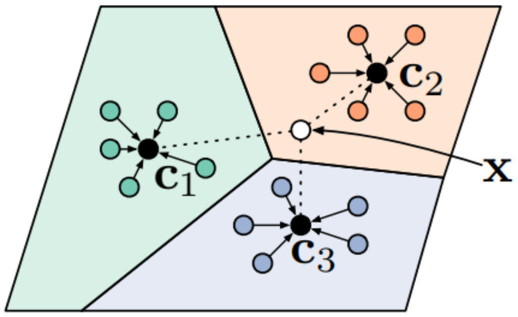
- Supervised (or self-supervised) few shot classifier
- Compute prototype $\mathbf{c}_k$ of each class in latent space
- S_k : labelled examples for class *k* (few for each class)

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$$

- f_ϕ : learnable embedding function
- Loss function: $J(\phi) = -\log p_\phi(y = k \,|\, \mathbf{x})$
- Query point mapped to a class using distance *d*

$$p_\phi(y = k \,|\, \mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))}$$



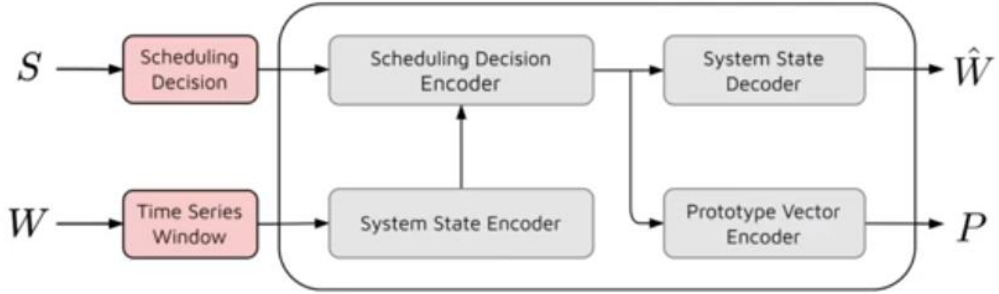


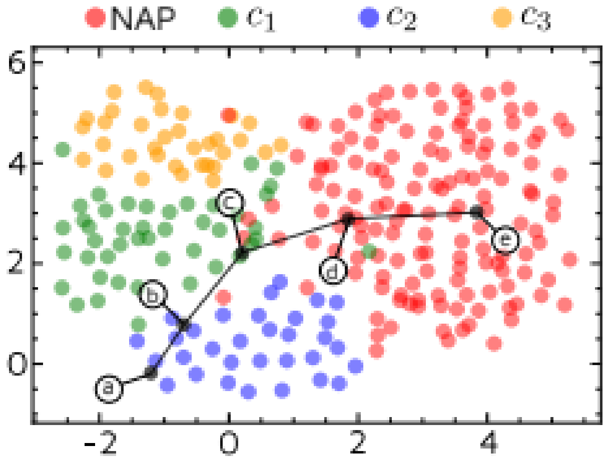
Prototype-based classification

Refs: Snell *et al.*; Prototypical Networks for Few-shot Learning; NIPS 2017.
Medina *et al.*; Self-supervised prototypical transfer learning for few-shot classification, AutoML 2020.

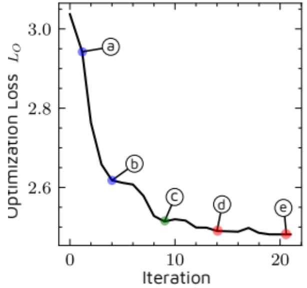
Deeper Models: Fault Prototypes with DeepFT

- Class prototypes updated at runtime





- Embedding NN also a surrogate model of edge layer spatio-temporal correlations
- Scheduler decision evolved online towards the no-fault class (NAP)

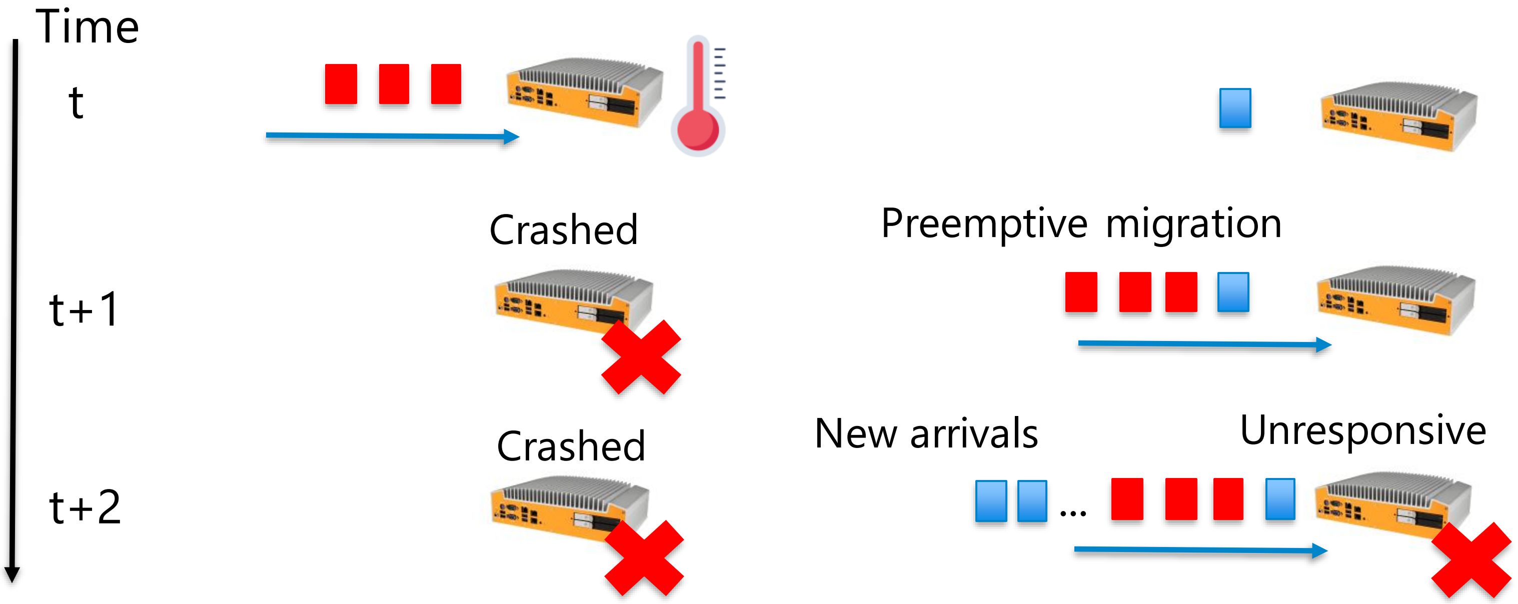


Ref: Tuli et al.; DeepFT: Fault-Tolerant Edge Computing using a Self-Supervised Deep Surrogate Model; INFOCOM 2023.

Modeling Spatio-Temporal Correlations Across Edge Devices

Deep models of spatio-temporal correlations

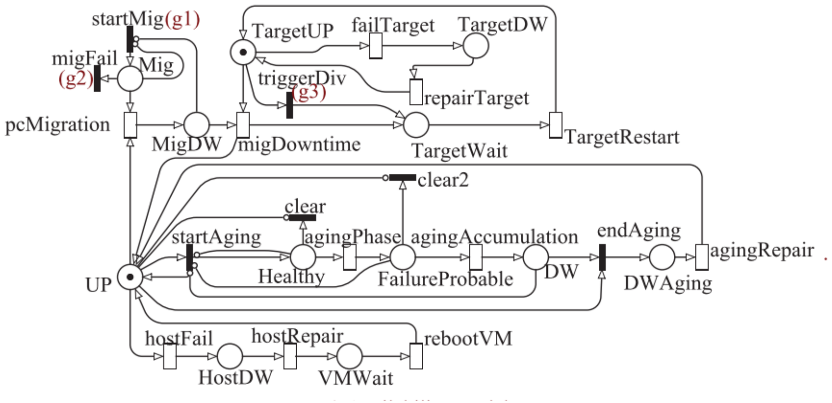
Example: pre-emptive task migration (or "bag of tasks" migration)



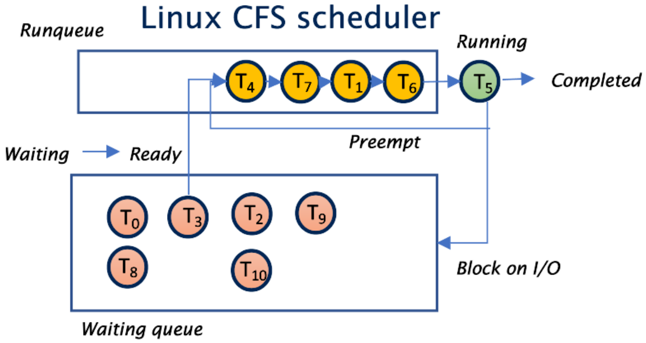
Coupling is difficult to characterize far from steady-state conditions!

System modelling

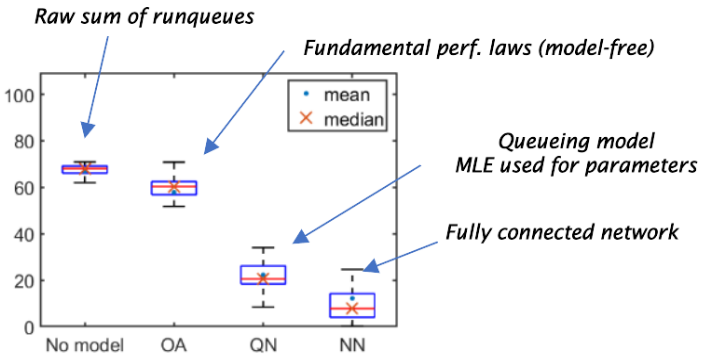
- Many stochastic models for FT and performability
  - SPNs, CTMCs, QNs, …
- Focus on "time" and "counts"
- Yet if data is abundant, NN often more accurate in dynamic settings:



Task consolidation on Linux host



Comparison of QoS prediction errors (MAPE)



Imperial College London

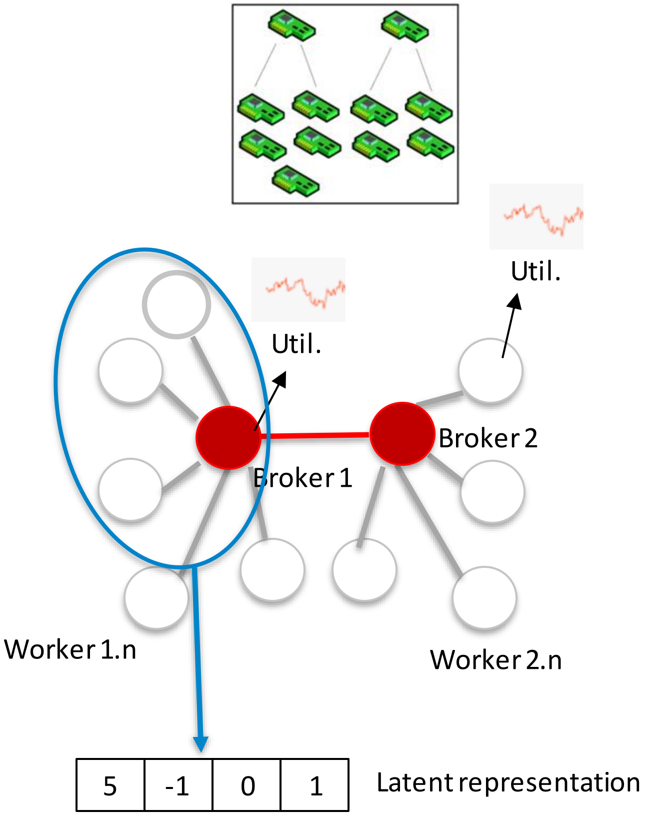
Modeling Correlations with GNNs

Graph attention networks (GATs):

- Node features are resource utilizations, periodically acquired from edge devices.
- Output is a latent representation at each host

Latent space representations based on:

- Graph convolution
  - message-passing to represent workload and utilization cross-correlations.
- Graph attention
  - assign different importance to each edge neighbor's contribution.

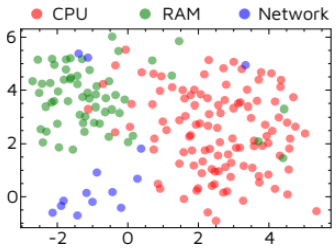


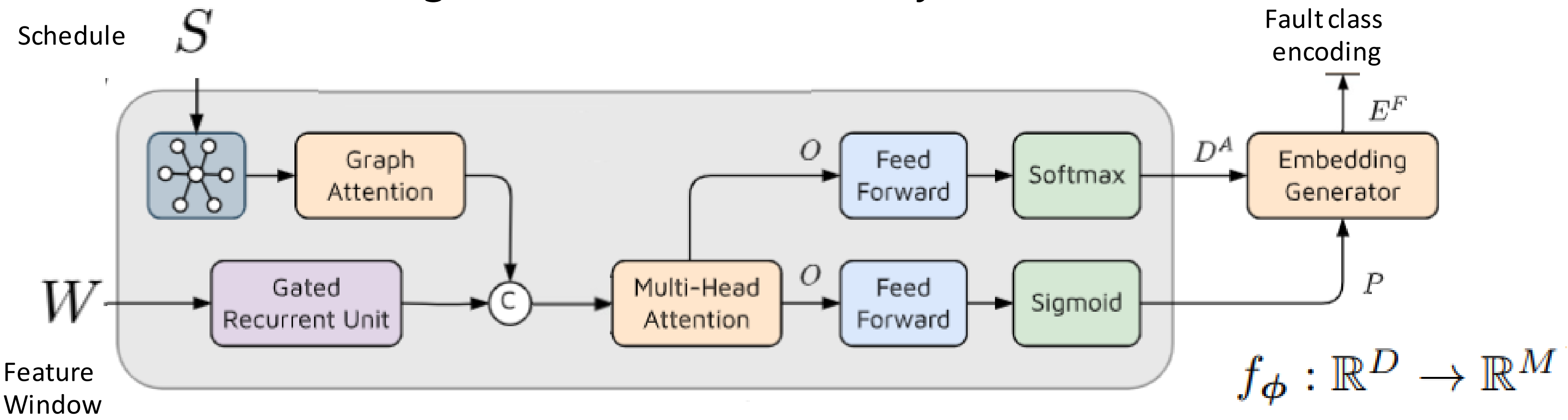
Imperial College London

Fault Prototype Encoder

A new component for runtime edge FT systems:

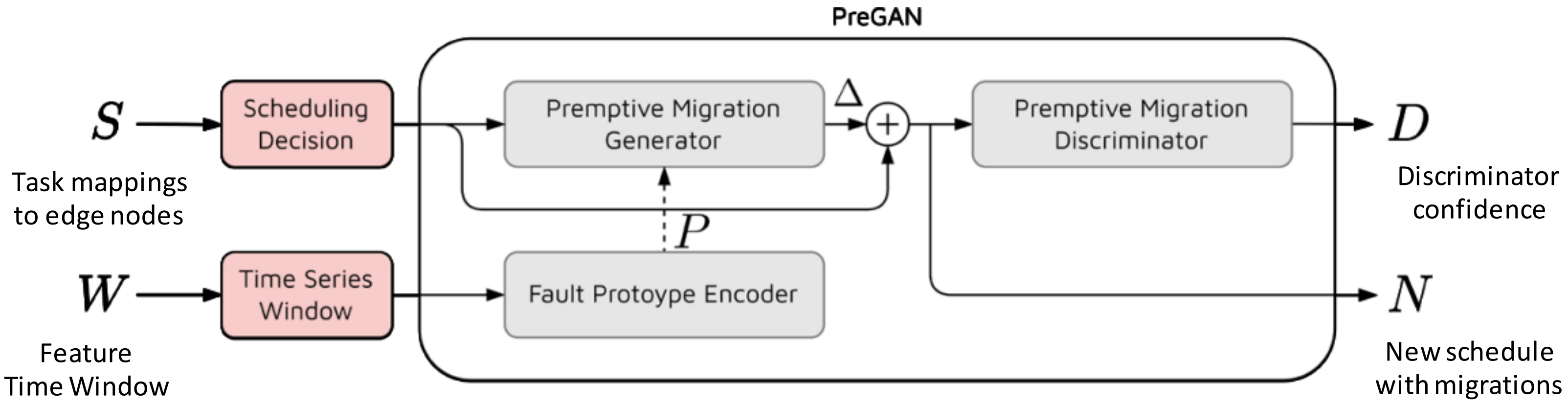
- Discriminator to identify fault class (D^A)
- Also outputs a fault class encoding (E^F)
- Host-level embeddings zeroed for non-faulty nodes





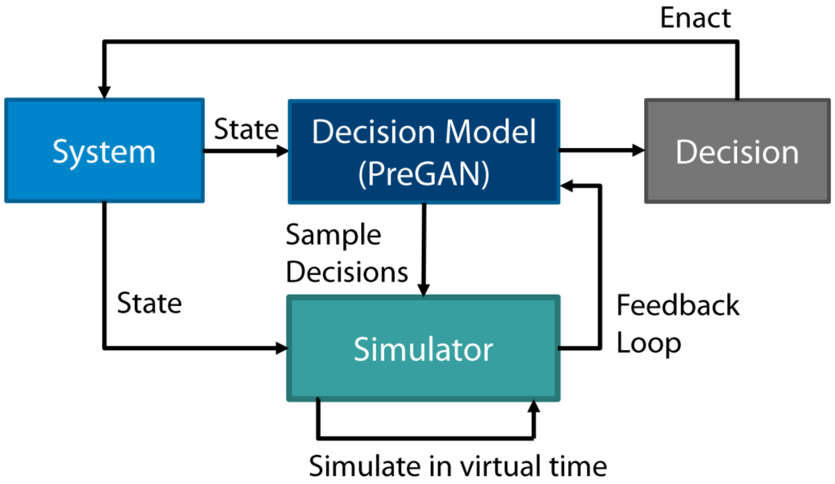
Ref: Tuli et al.; PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing. INFOCOM 2022.

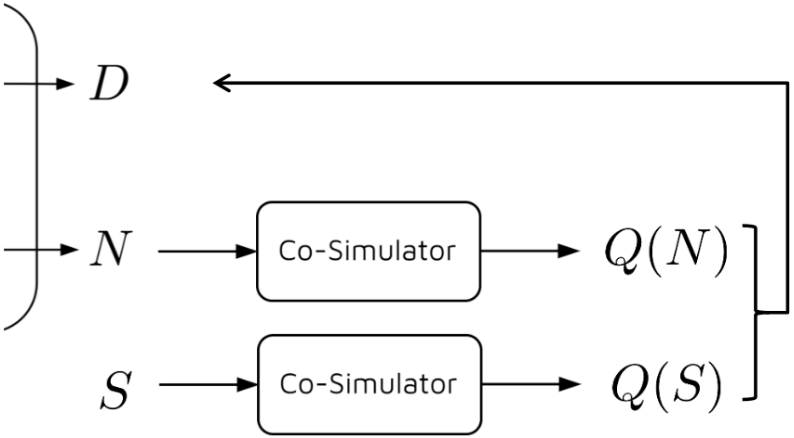
A Preemptive Migration Application: the PreGAN model



- Generator introduces a candidate pre-emptive migration Δ
  - Conceptually similar to a conditional GAN
  - GAN acts as a simulation surrogate
- Discriminator scores confidence on the proposed migration

PreGAN: Runtime Fine-Tuning



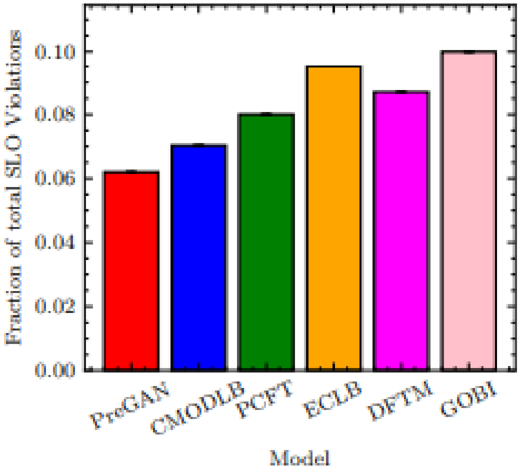


- GAN discriminator trained to discard new schedules with worse QoS

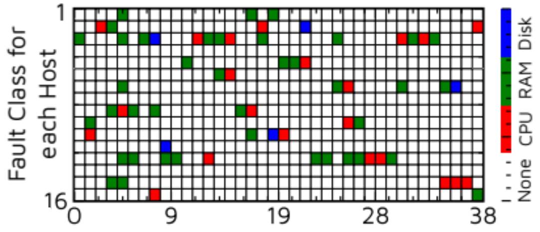
$$L_D = \begin{cases} \log(D) + \log(1-D) & \text{if } Q(N) \geq Q(S) \\ \log(1-D) + \log(D) & \text{otherwise} \end{cases}$$

PreGAN: Results

- Testbed: 8x4GB + 8x8GB raspberry PIs 4
- DeFog IoT applications
- Baselines:
  - CMODLB: K-Means clustering and Swarm Optim.
  - PCFT: Particle Swarm Optimization
  - ECLB: Bayesian Optimization and Neural Nets
  - DFTM: Integer Linear Programming
  - GOBI: Scheduling only (with a co-simulator)



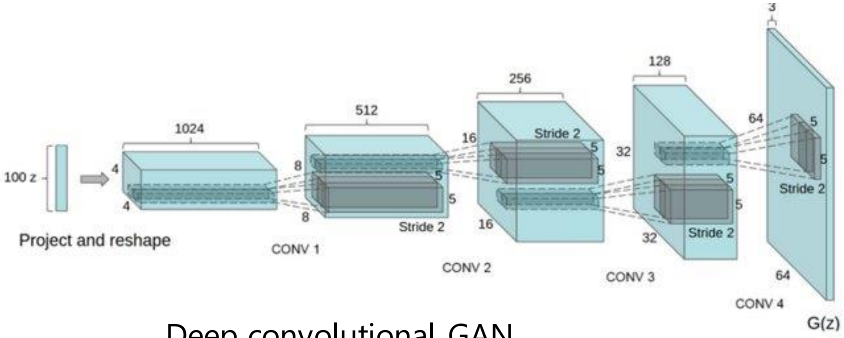
Method	Detection				Diagnosis	
	Accuracy	Precision	Recall	F1 Score	HR@100	NDCG@100
DFTM	0.8731 ±0.0234	0.7713 ±0.0823	0.8427 ±0.0199	0.8054 ±0.0872	0.5129 ±0.0212	0.4673 ±0.0019
ECLB	0.9413 ±0.0172	0.7812 ±0.0711	0.8918 ±0.0203	0.8329 ±0.0901	0.4913 ±0.0010	0.5239 ±0.0024
PCFT	0.8913 ±0.0108	0.8029 ±0.0692	**0.9018 ±0.0165**	0.8495 ±0.0312	0.5982 ±0.0094	0.5671 ±0.0020
CMODLB	0.9128 ±0.0112	0.8158 ±0.0343	0.9013 ±0.0091	0.8605 ±0.0284	**0.6309 ±0.0025**	0.5432 ±0.0031
PreGAN	**0.9635 ±0.00921**	**0.8723 ±0.0221**	**0.9018 ±0.0121**	**0.8868 ±0.0629**	0.6232 ±0.0069	**0.5898 ±0.0080**



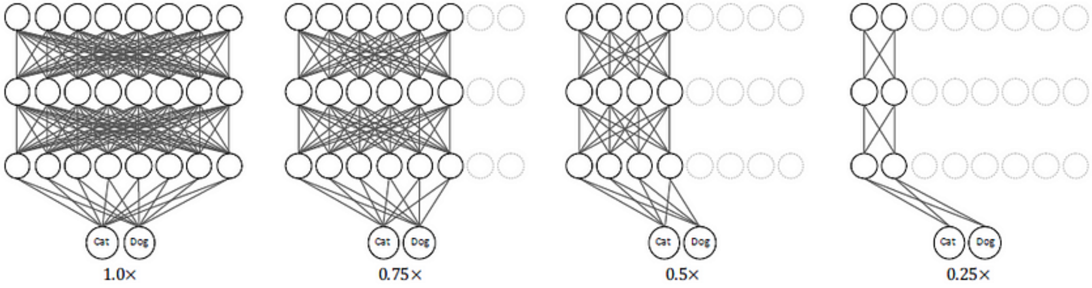
Dealing with Memory Constraints

DL Model Compression

- GANs can easily saturate device memory
- Several compression techniques exist
  - Pruning
  - Knowledge Distillation
  - Quantization
  - Splitting
  - Low-rank factorization
  - Memory Compression
  - Slimming
  - ...



Deep convolutional GAN

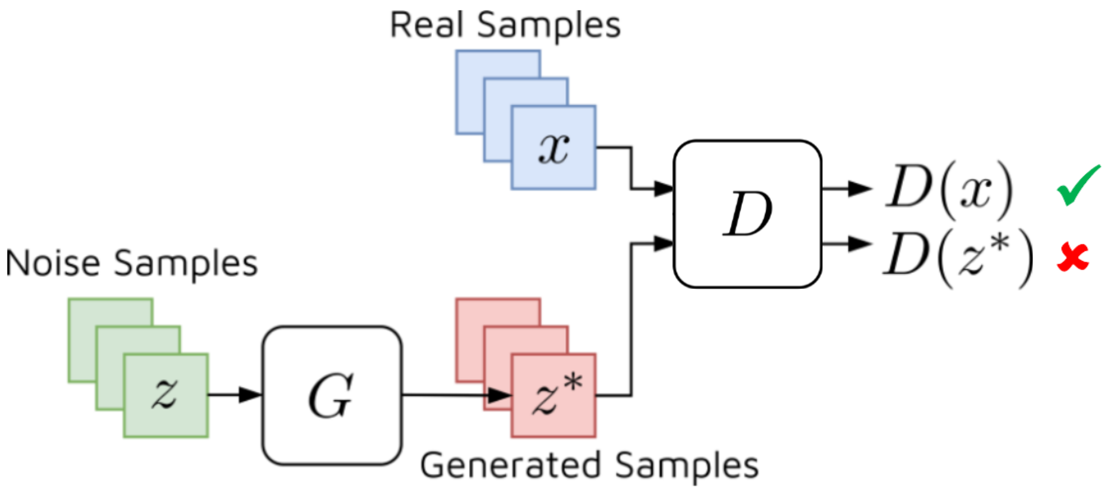


Slimmable neural networks

Refs: D. Tantawy *et al.*; A survey on GAN acceleration using memory compression techniques; JEAS, 2021.
J. Yu *et al.;* Slimmable neural networks; ICLE 2019.

Generative Adversarial Network (GAN)

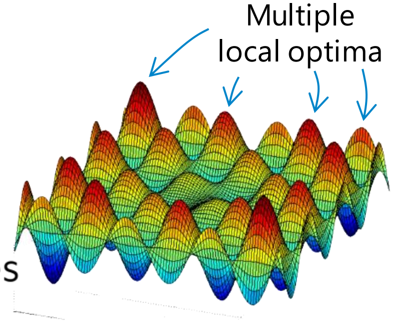
- Generator create *fake* samples from random noise
- Discriminator classifies inputs as *real* or *fake*
- Adversarial training in a zero-sum game

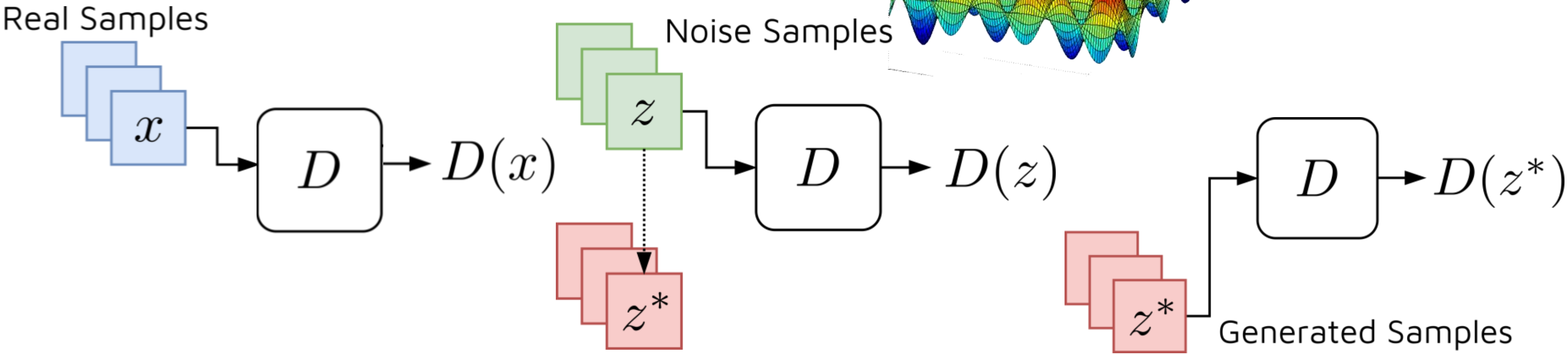


Generative Optimization Network (GON)

\+ No generator required (Up to +250% higher F1/GB than slimmable GANs)

\- Longer training times than GANs (+5% to +40%)





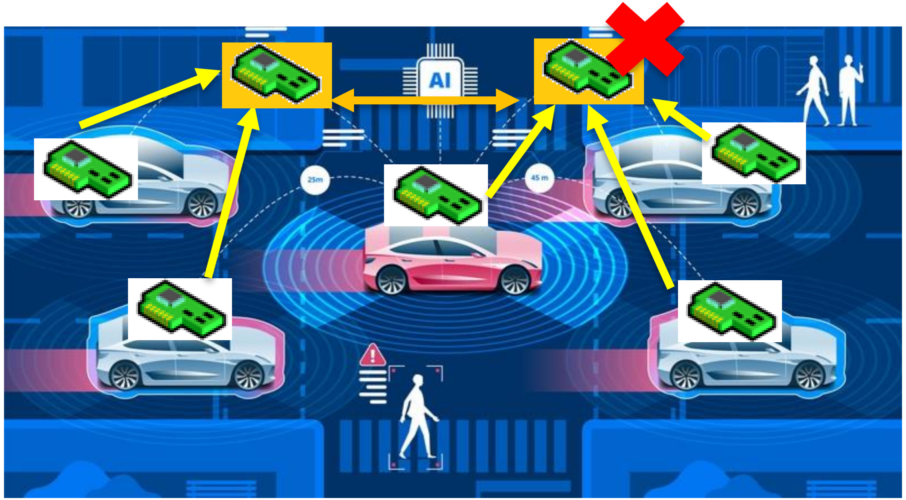
$$z \leftarrow z + \gamma \nabla_z \log\left(D(z)\right)$$

Training (SGD+mini-batches):

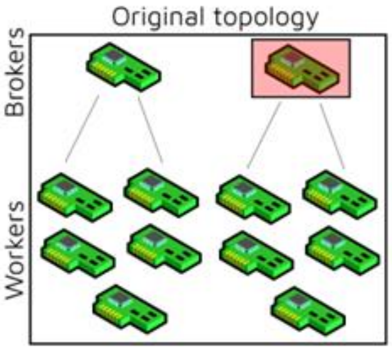
$$\log(D(G, M, S)) + \log(1 - D(G, z^*, S))$$

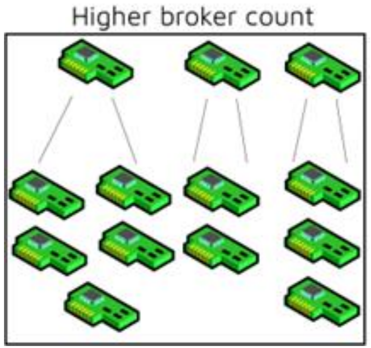
Example: Reliability in Edge Federations

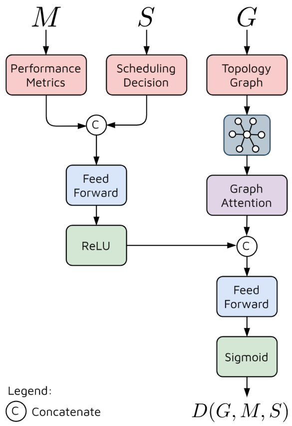
- Edge architectures:
  - Local edge infrastructures (LEIs)
  - Configurable broker-worker roles
  - QoS vs. limited resources
- Federation can share resources and rebalance task load across LEIs
- Problem: Broker resilience
  - QoS/SLOs calls for fast remediation to broker failures
  - Method must be lightweight

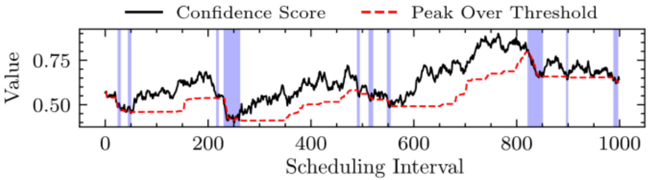


CAROL: a GON-based FT Technique









1. Broker failure detection

2. Recovery Candidates

3. CAROL GON forecasts QoS within Tabu Search

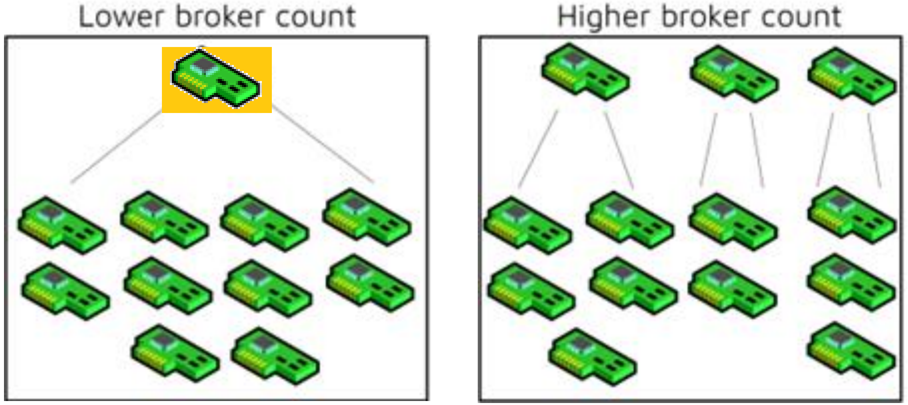
4. Confidence triggered fine-tuning of CAROL GON

Iteration time: ~100ms
Node-shift time: ~2s
GON memory size: ~1 GB

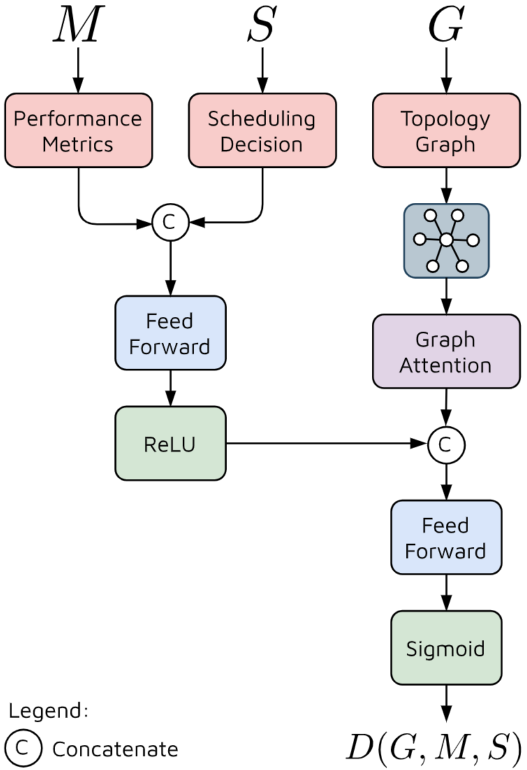
A Surrogate Model based on GONs

Deep model captures "normal" relationship between QoS, topology and task allocations.

- Input: metrics, schedule, candidate recovery



- Output:
  - Confidence Score *D(G, M, S)*: measure that the input is from the real data distribution



Ref: Tuli *et al.*; CAROL: Confidence-Aware Resilience Model for Edge Federations. DSN 2022.

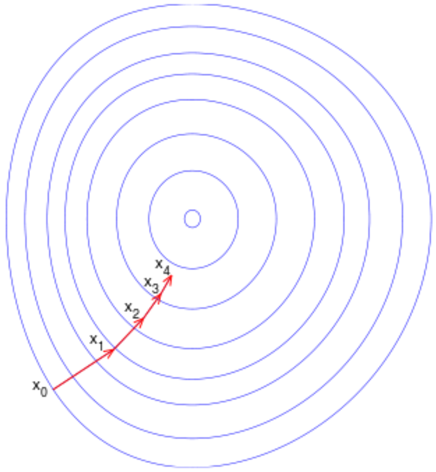
Using the GON

- We train this using the GON approach to generate realistic performance measures

$$L = \log\left(D(M, S, G; \theta)\right) + \log\left(1 - D(Z^*, S, G)\right)$$

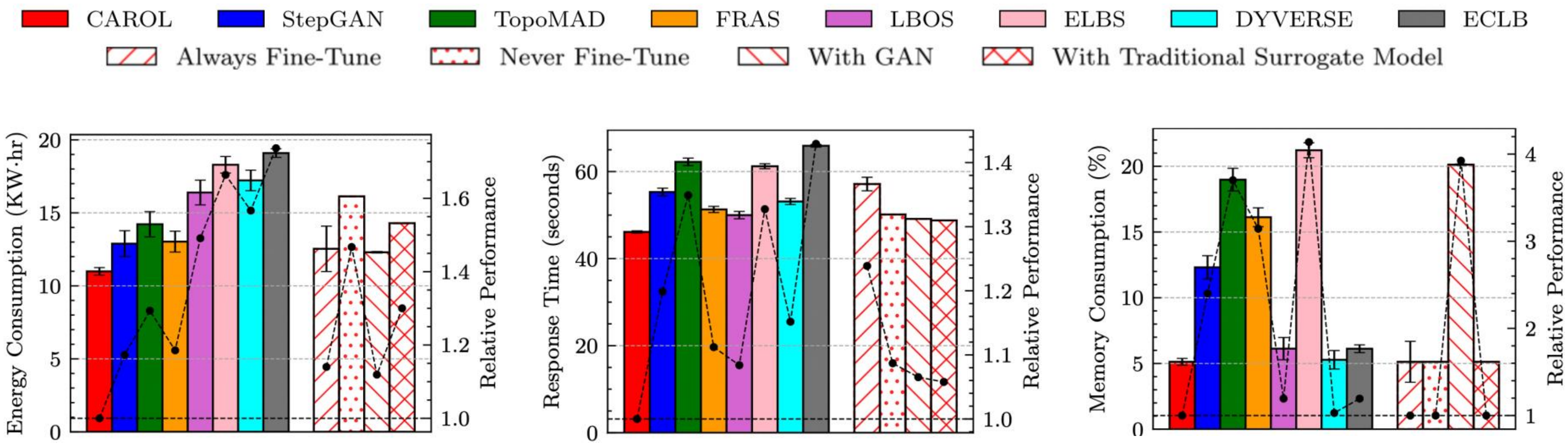
- Forecast topology performance using GON as surrogate
  - Find expected performance metrics for given schedule *S* and topology *G*

$$M \leftarrow M + \gamma \nabla_M \log\left(D(M, S, G)\right)$$



Imperial College London

Results



CAROL gives >16.5% lower energy consumption

CAROL gives >8% lower response time

CAROL consumes only 40% memory compared to StepGAN

Conclusion

Closing takeaways

- Many new challenges in edge computing stress existing FT techniques
- Resource constraints should not stop us from looking at edge-layer NNs
- Deep architectures can embed whole FT workflows
- Deep models offer new tools to represent correlations at the edge
- Generative, co-simulation and few learning help cope with data constraints

Some Directions for the Future

- Broader coverage of fault types is needed
- Concept drift / adaptive AI
- Software engineering methodologies
- Explainability of Deep Models
- What synthesis shall we reach with classic FT approaches and models?