Aalto University
School of Science

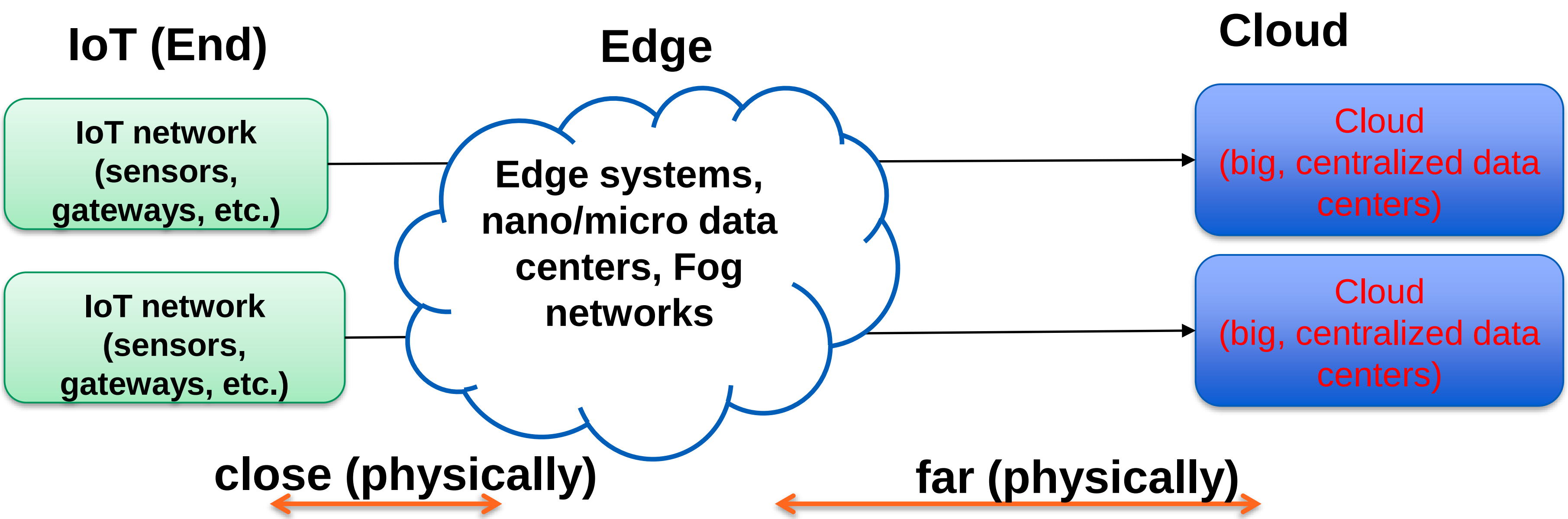
Machine Learning with Edge Systems

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, *https://rdsea.github.io*

Learning objectives

- **Understand and analyze the relationship between edge computing and ML**
- **Explore and study basic concepts and issues when engineering ML in edge systems**
- **Identify and work on ML optimization problems across levels of abstraction in edge systems**

IoT-Edge-Cloud



"Edge" is just an abstraction view

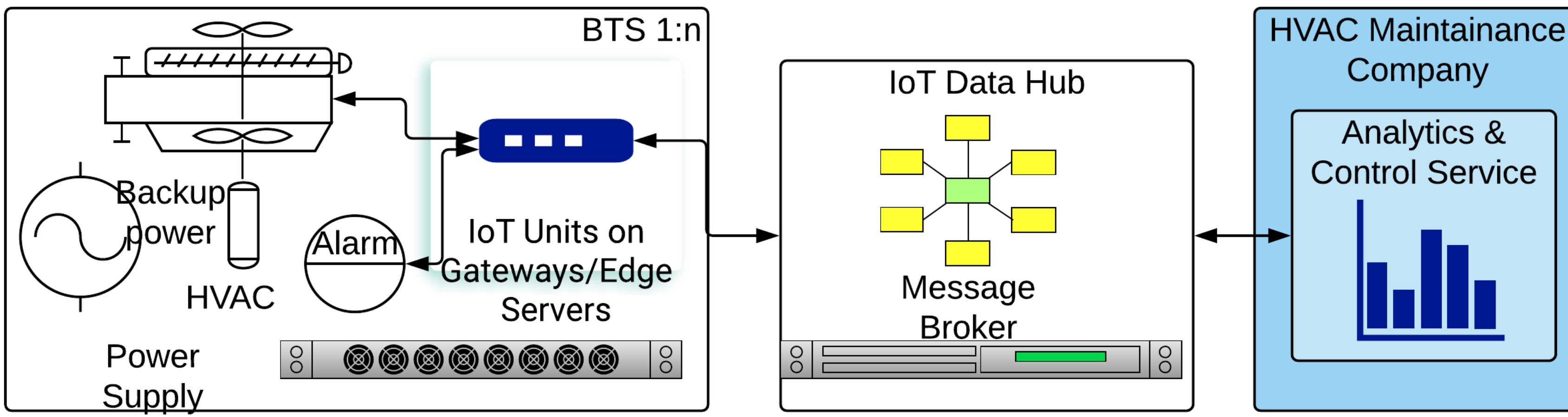
Edge computing

- **Edge computing paradigm focuses on distributed computing at the edge and end devices**
  - *many distributed low-end as well as a limited number of high-end devices/machines for different purposes*
- **Leveraging common technologies like in the cloud and specific ones**
  - *e.g., virtualization, messaging systems, storage/database, Web services*
- **But with different constraints**

Edge computing

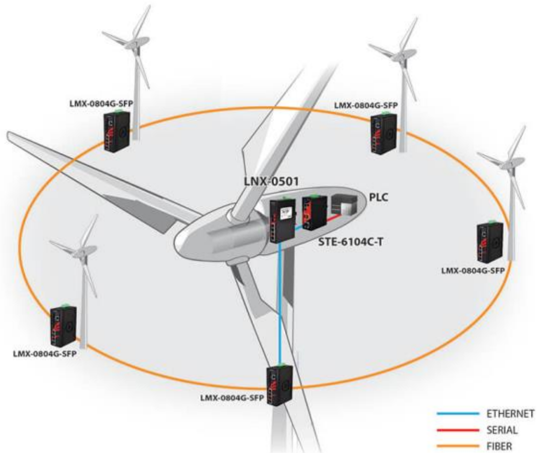
- **Computation/analytics can be done at the edge**
  - where data is generated, close to the data sources
    - *next to IoT devices and sensing equipment,*
  - many distributed (moving) locations, e.g., in the shopping center, in the car
- **Near real-time processing is needed in most situations**
- **Very heterogeneity w.r.t system models, hardware architectures, network connectivity, protocols**

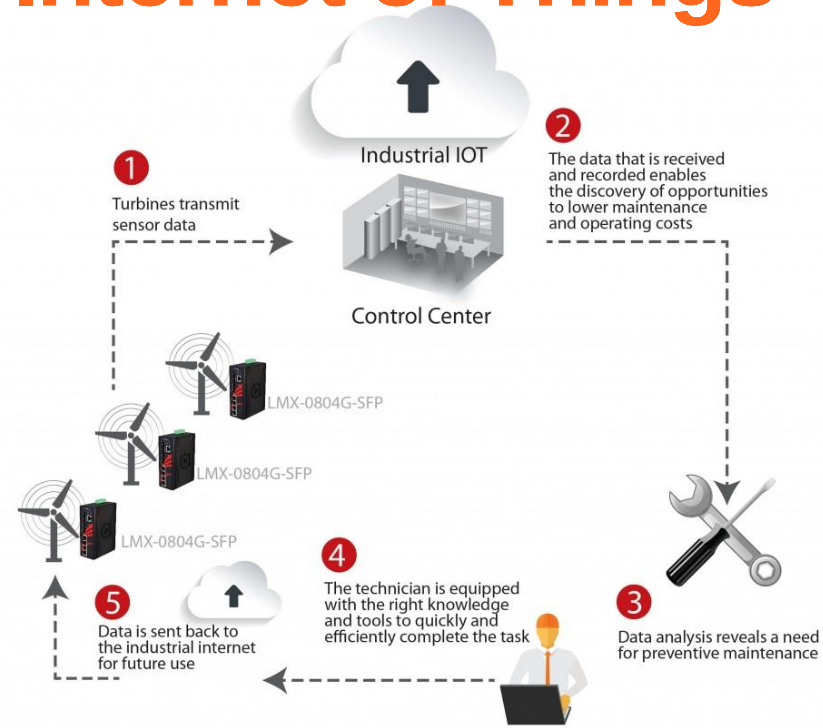
Example: Predictive maintenance



Move to the edge

Example: Industrial Internet of Things





Figures source: http://www.windpowerengineering.com/design/electrical/controls/wind-farm-networks/talking-turbines-internet-things/

Example: video analytics at the edge

Use Case 3: Video Analytics

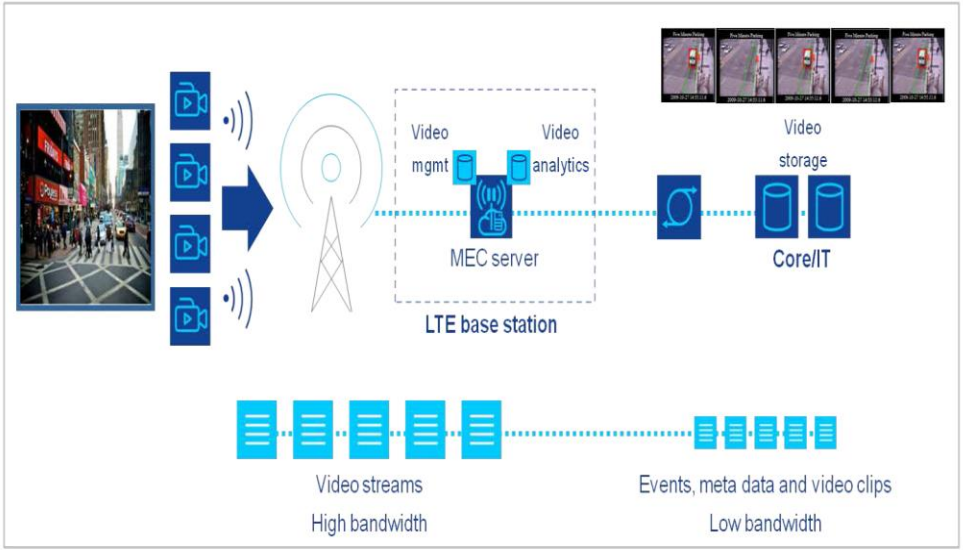


Figure 4: Example of video analytics

Figure source:
https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

Why do we have to support ML/data analytics at the edge?

What kind of benefits?

Machine learning/big data analytics in the edge

- **Many applications can benefit from ML/data analytics capabilities**
  - Inferencing/classification in mobile devices
  - Realtime ML-based steering (autonomous cars, speech control, traffic controls)
  - Realtime detection: fraud detection, anomaly detection, accidents
  - Manufacturing (Industrial Internet of Things)

Machine learning/big data analytics in the edge

- **Close to data sources → "data locality" benefits**
  - Security & privacy
  - Performance
  - Customization/Personalization
  - Cost saving

Aalto University
School of Science

Basic concepts/issues when engineering ML in edge systems

Very new area! a lot of ongoing research and development!

What do we need to consider when supporting ML in the edge?

- **Network problems**
  - High latency, low-bandwidth, unreliable connectivity
- **Computation capabilities**
  - Constrained processing power, a lot of specific chips and accelerators, and limited memory
- **Storage is not enough for big data**
- **V* issues in data**
  - Out of distribution data, unlabeled data, time series data, streaming data
- **Energy/power usage of devices/machines**

What do we need to consider when supporting ML in the edge?

- **Edge with hardware heterogeneity**
  - common hardware (e.g., AMD, Intel, ARM), SoC and microcomputers, microcontrollers
  - with/without common and AI-based accelerators like FPGA, GPU, and TPU

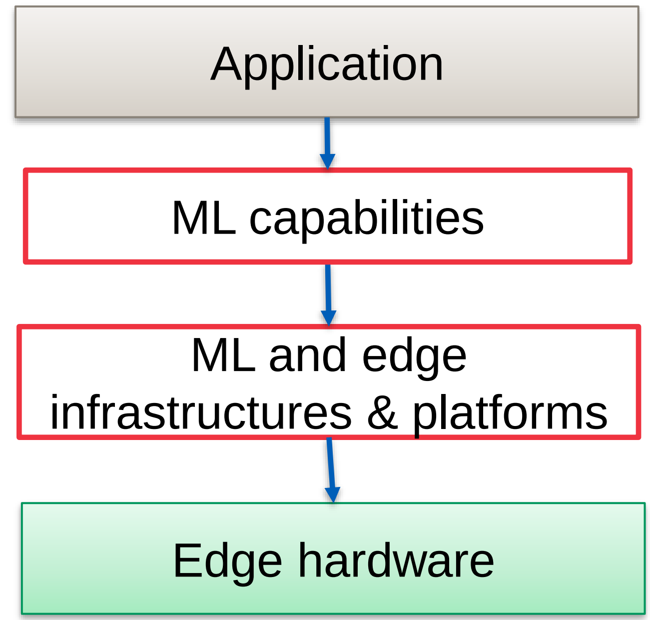
→ Requirements for certain types of ML might not be fulfilled: computation-intensive ML (e.g., video analytics)

Pervasive embedded edge devices

- Raspberry PI4
- Google Coral
- Jetson Nano
- Xilinx
- A huge number of MCUs (Microcontroller Units)

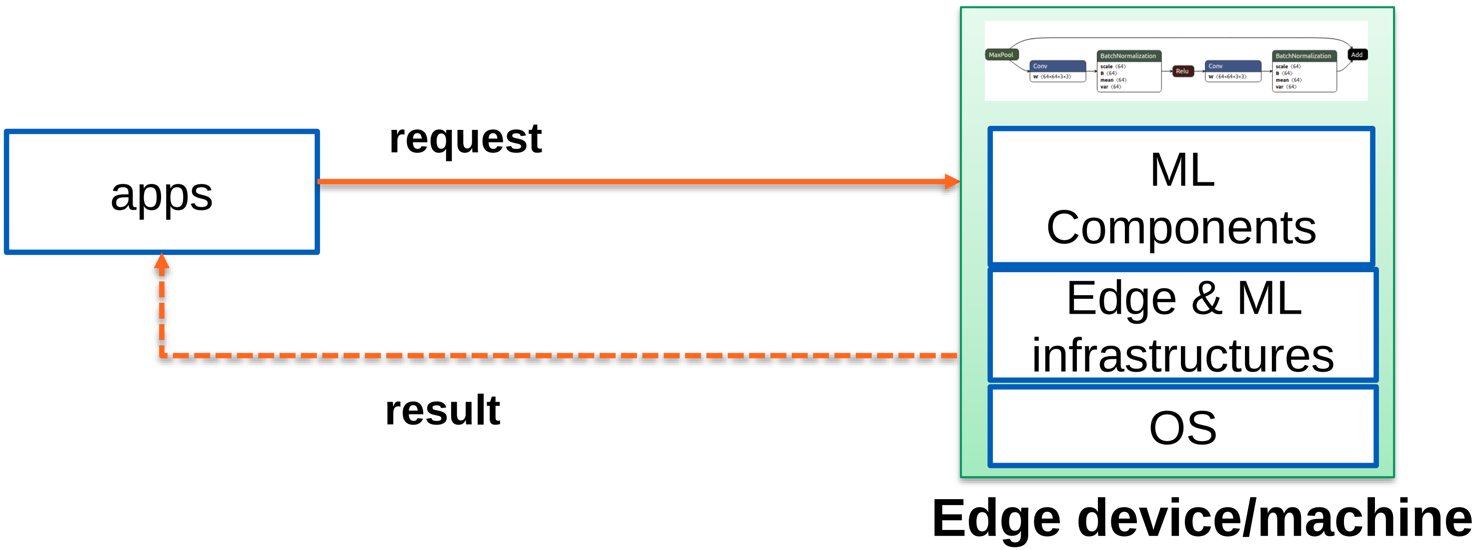
Interaction models in edge ML systems

Which components do what and where are they?



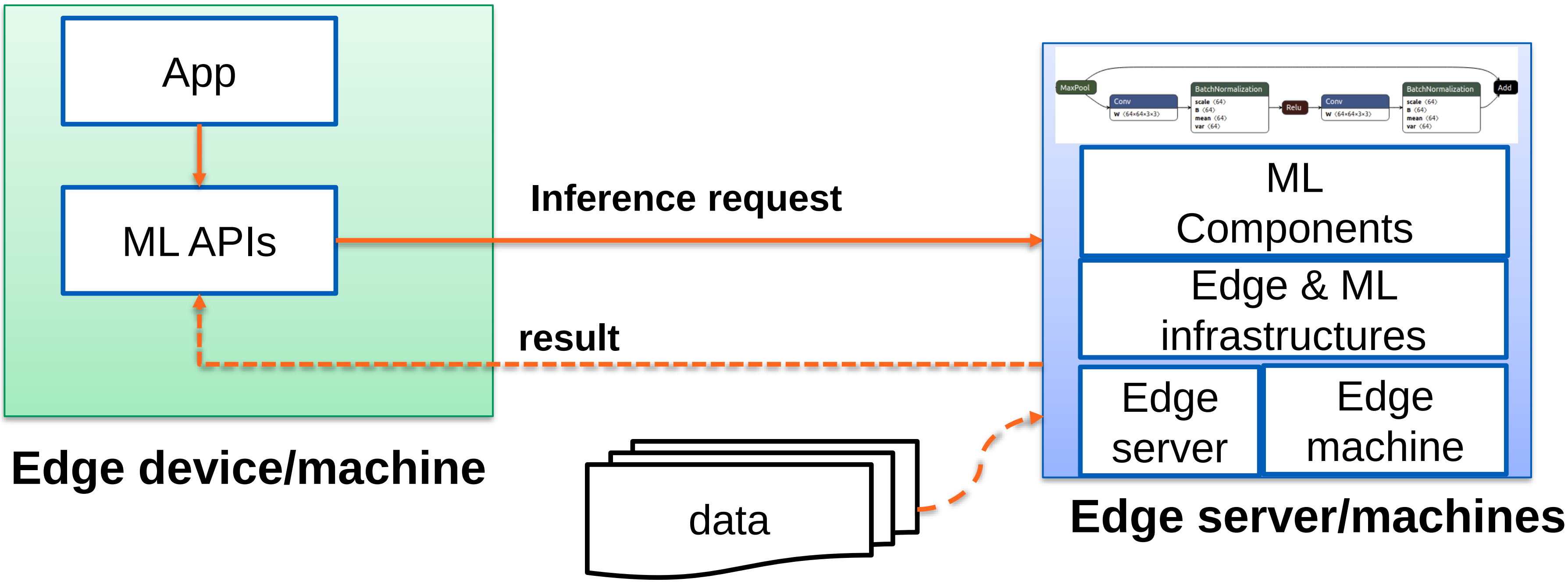
Interaction models

Standalone/in-device ML capabilities within independent devices



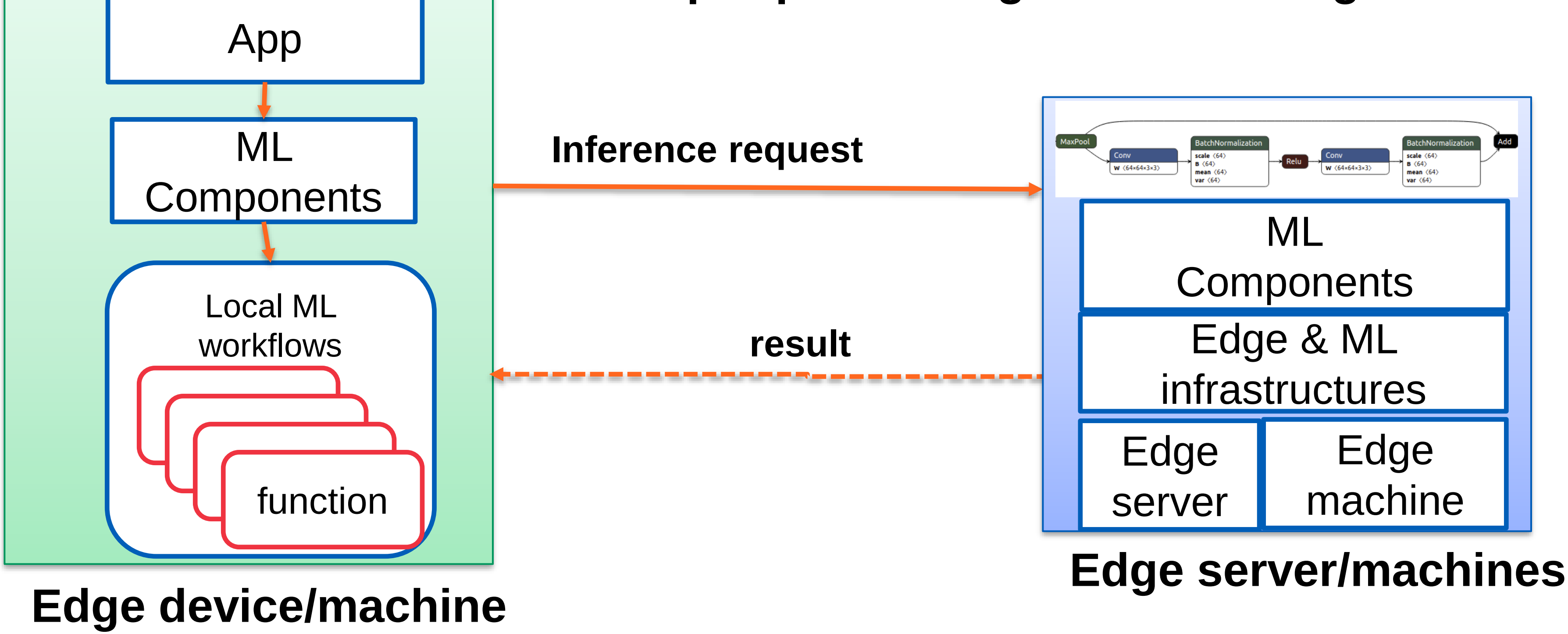
Interaction models

Common client-server model without local processing

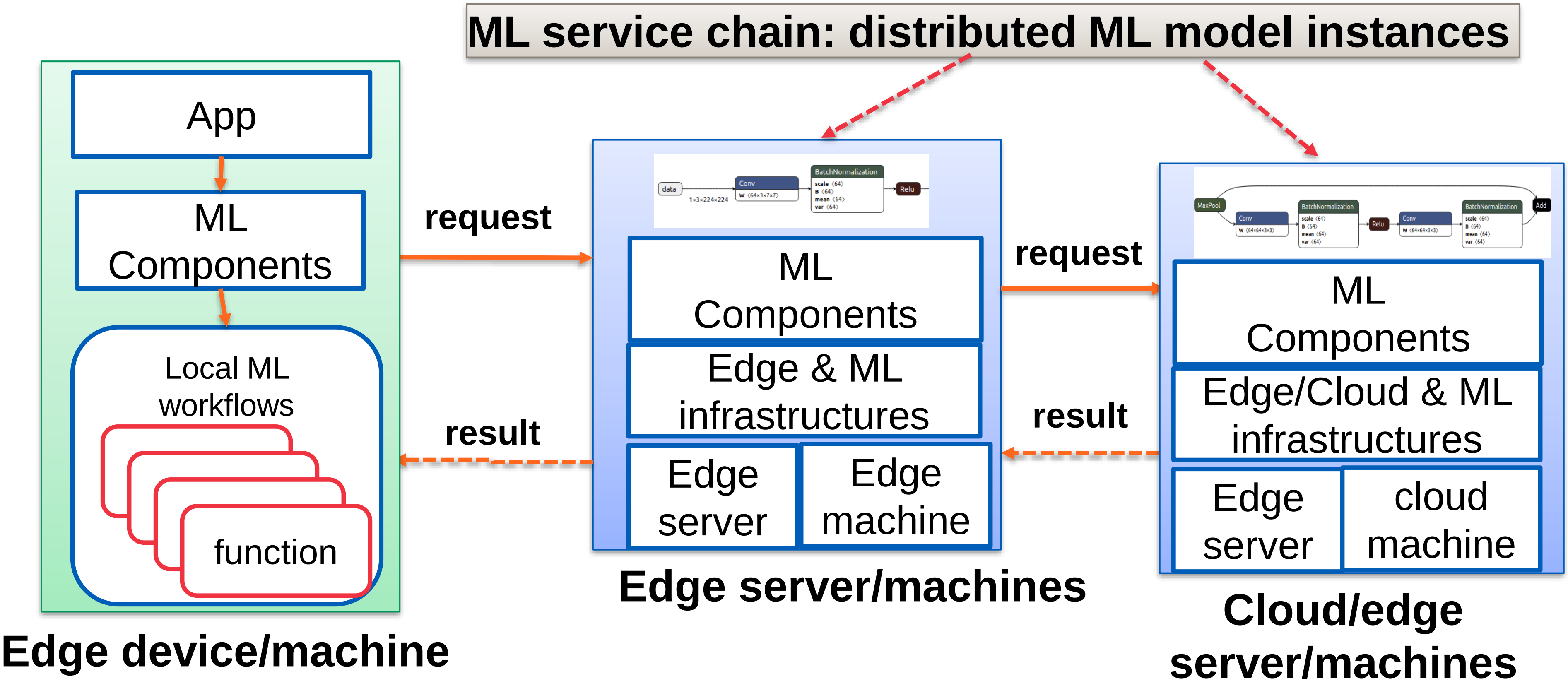


Interaction models

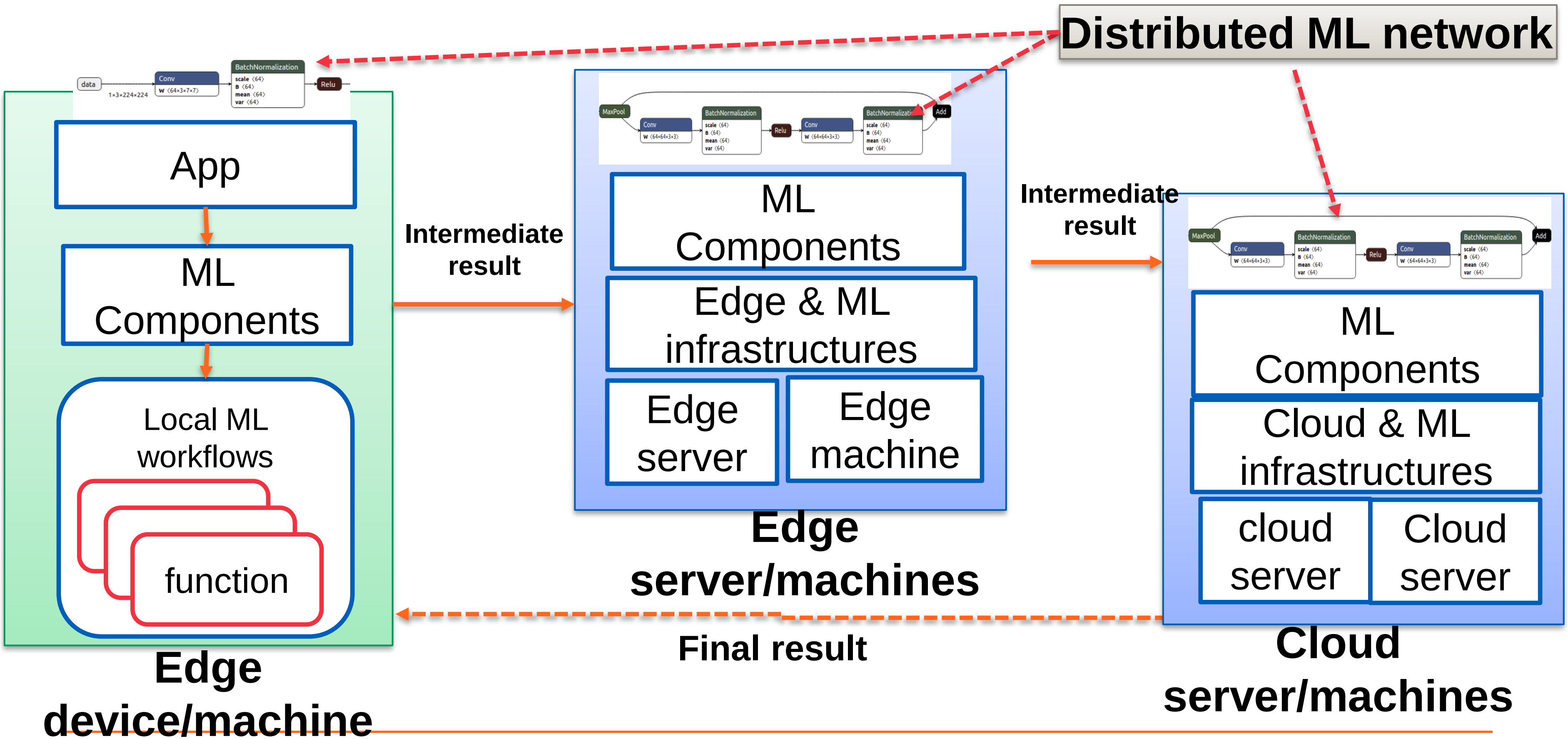
Local pre-processing and offloading



Interaction models

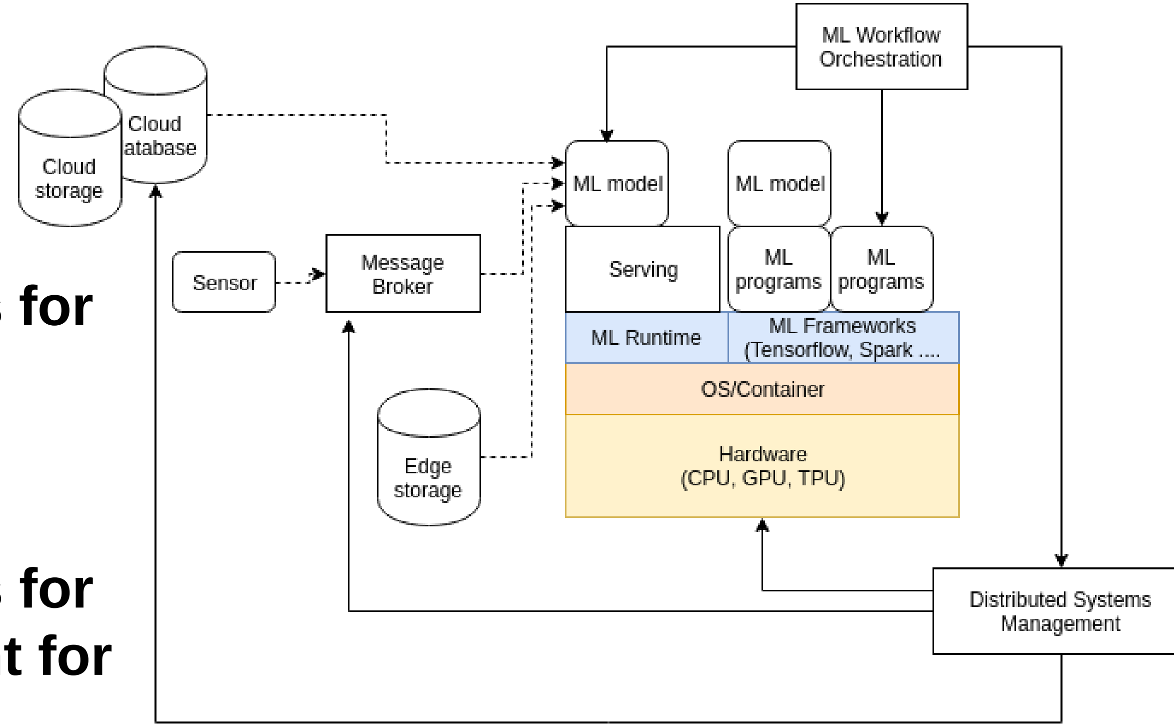


Interaction models



Software systems for ML in the edge

- **What are key features for ML runtime and programming frameworks?**
- **What are key features for resource management for running ML?**



Suitable ML and runtime for the edge: key requirements

- **Energy consumption**
- **Resource constraints**
  - less computation capabilities → precision and accuracy?
- **Latency and uncertainty**
- **Interfaces with different networks capabilities**
- **Support accelerators**
  - E.g., FPGA, AI Accelerators (e.g. Intel® Movidius Myriad X VPU)
- **Trade-offs between generic versus specific features**

Examples of ML frameworks and Runtime for the edge

- **TF-lite (https://www.tensorflow.org/lite)**
- **https://github.com/Microsoft/EdgeML**
- **uTensor: https://github.com/uTensor/uTensor**
- **Androi NN (https://developer.android.com/ndk/guides/neuralnetworks)**
- **CoreML (https://developer.apple.com/machine-learning/core-ml/)**
- **PyTorch mobile (https://pytorch.org/mobile/home/)**
- **Snapdragon Neural Processing Engine SDK**
  - https://developer.qualcomm.com/docs/snpe/overview.html

Changes in MLOps

- **MLOps (ML DevOps)**
  - DevOps principles for ML
  - In ML engineering processes: key artefacts are ML models, data and runtime libs
  - New areas, still a lot of ongoing research work
- **Changes in ML with edge systems**
  - DevOps and DataOps activities in the edge
  - Optimization and training activities
  - Tests and benchmarks
  - Monitoring

Example of MLOps

https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Is it the same in the edge?

What would be MLOps for ML in the edge?

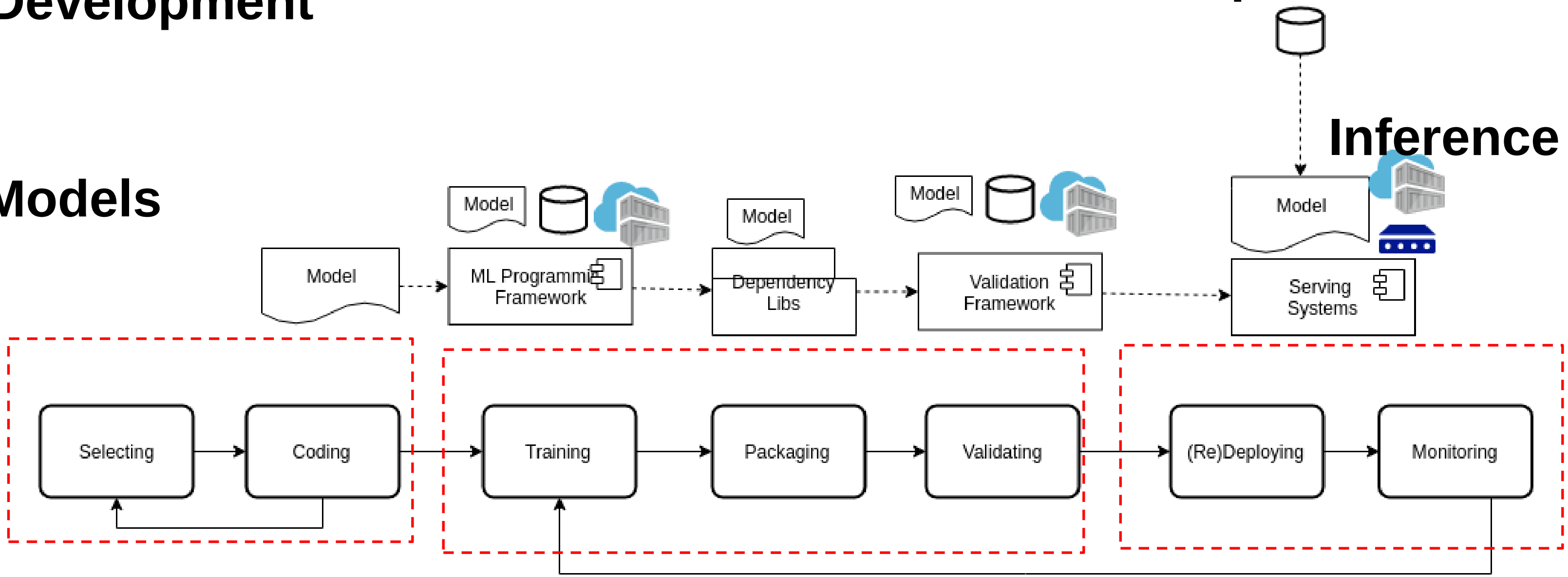
MLOps in edge systems

Development

Operations

Artefacts: Models framework

Phases Activities



Where?

done in common dev environment

?

Edge

Train in clouds/on-premise but edge deployment

- **Training in cloud and/or on-premise, and inferences in the edge**
  - Issues of optimization, loss in transferring/conversion
  - Accuracy loss due to the conversion
- **Training and inferences in the edge**
  - Difficult with tools
  - Accuracy loss due to the training (limited)

Training in cloud and inference in the edge

https://blogs.gartner.com/paul-debeasi/files/2019/01/Train-versus-Inference.png

Can you guess some issues that we need to deal in this case?

Examples

https://developer.qualcomm.com/docs/snpe/overview.html

Some optimization problems

Multiple levels of optimization

Scope/level of abstraction	Research issues
Edge ML Service	**ML serving, ML elasticity**
Cross edge machines ML platforms/infrastructures	**ML function partitioning, orchestration, deployment, observability ..**
In-device/-machine ML platform	**Device-machine specific optimization**

Aalto University
School of Science

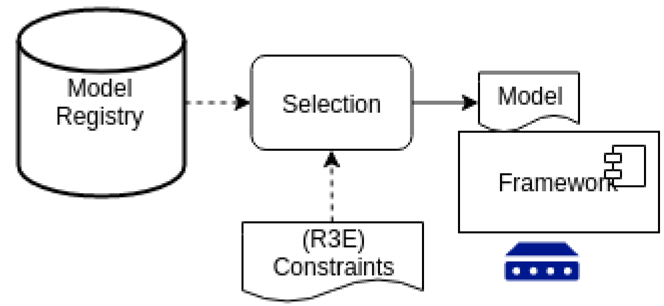
Our focus: to understand and practice engineering analytics during ML development

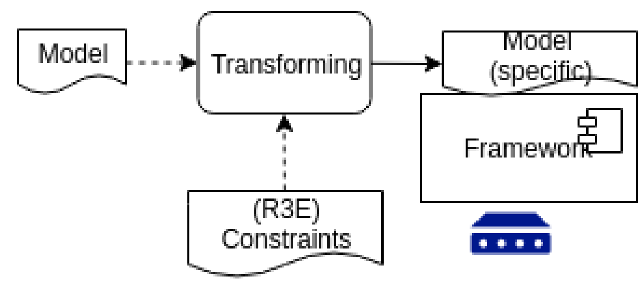
Selected problems: transfer learning

- **Transfer learning**
  - Repurpose a model trained for a task for another task
  - Basically it is an optimization of an existing model for a new task
  - Need model selection, reuse and model retraining
- **Transfer learning for the edge**
  - Conversion/Translation: transforming typical models in common environments to edge models
  - Symbiotic engineering: learning with simulations and inference with real data
  - Application domains adaptation: adapt models among application domains

Selected problems: model selection and conversion

- **Model management and selection**
  - Precision and time tradeoffs with computational requirements
  - Work with microcontrollers and accelerators
- **Transforming**
  - A model can be supported by different frameworks
- **How will these issues affect Robustness and Reliability?**





Example: model conversion

- **Conversion**
  - just a simple form of “transforming”
- **A model fits into a single device/machine or into a set of machines?**
- **Single device/machine: no distributed computing**
  - focus on ML service and in-device optimization levels
- **A set of machines:**
  - which are distributed computing models for ML across machines

Selected problems: model optimization

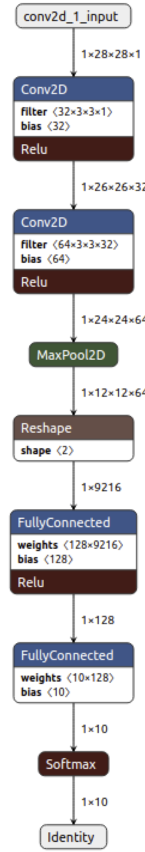
- **Pruning**
  - Prune graphs for training, remove features in ML models which are not significant
- **Quantization**
  - Reduce precision representation, storage, bandwidth
- **Conditional computation/Regularization**
  - Activate certain units of the model
- **How will these issues affect Robustness, Reliability and Elasticity?**

Tools/frameworks

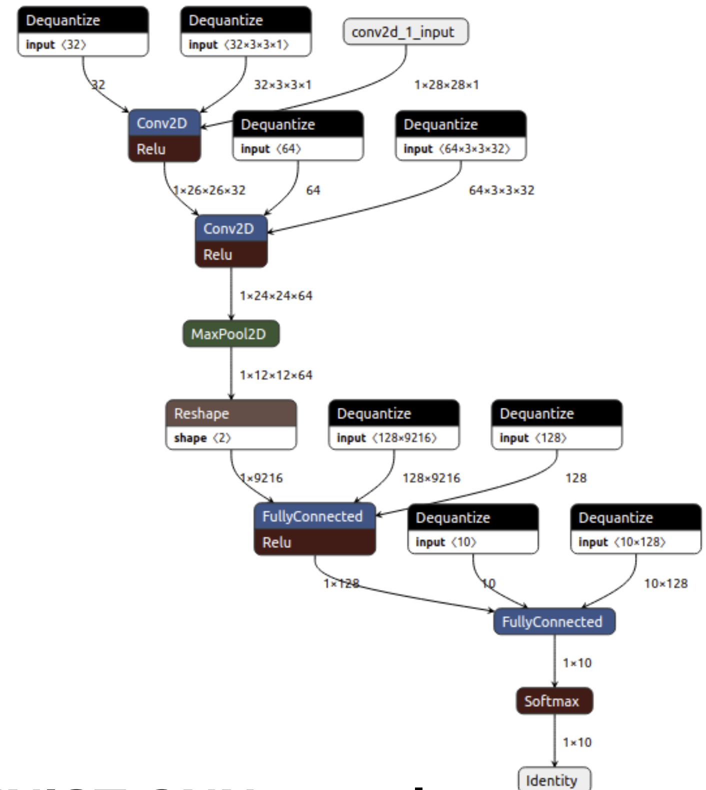
- **ONNX (Open Neural Network Exchange) format**
  - Can be used as an intermediate representation compiled by tools to specific targets
- **Nvidia TensorRT**
  - JetPack SDK
- **OpenVINO (https://docs.openvinotoolkit.org/latest/index.html)**
- **Apache TVM (https://tvm.apache.org/)**
  - VTA (Versatile Tensor Accelerator)

Example of Quantification by reducing floating point

32 bit floating point



16 bit floating point



MNIST CNN sample

Conversion: the case of distributed models

- **Goal:**
  - if you have a model, now how to split it into edge/cloud?
- **Possible approaches**
  - partitioned model: split a model into different sub models
  - distributed ML networks: distribute the model graph across edge/cloud systems
  - federated learning: distributed training parts
  - chain of distributed ML models
- **Not a simple task – need to combine many techniques**

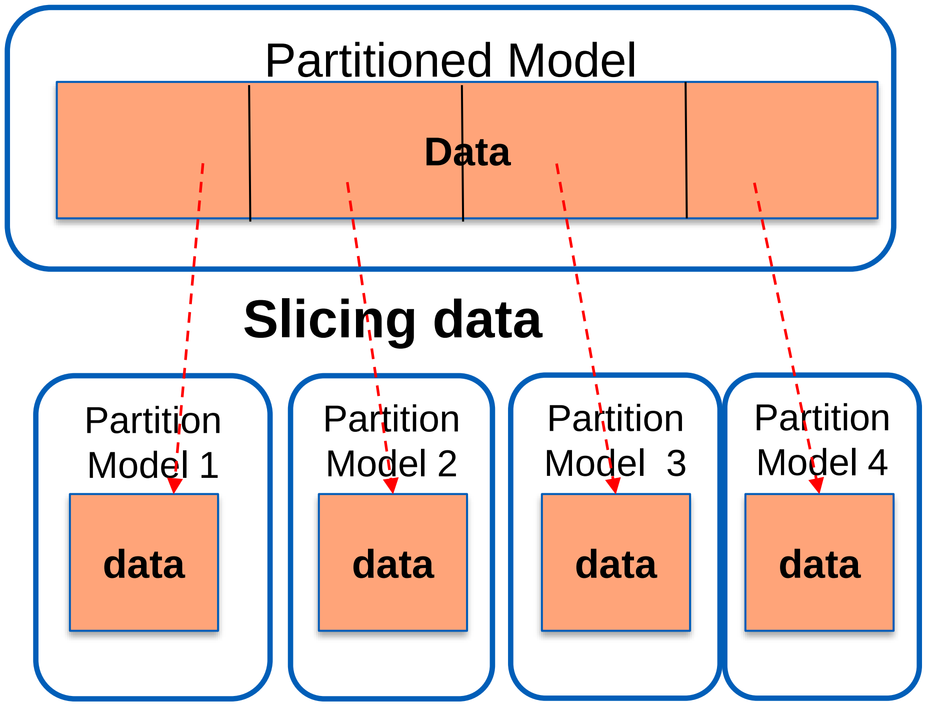
Partitioned models

- **A kind of “function partitioning” problems**
- **Training many partition/sub models, each for a partition data**
  - e.g., network operations in a city versus in country sides
  - a partitioned model consists of multiple sub models
    - *Work as a single model*
- **Slice input data into partitions, data in a suitable partition will be mapped into partition models (e.g., data partition)**

- **We can have a partitioned model running in multiple edges (each edge, e.g., host a partition model)**

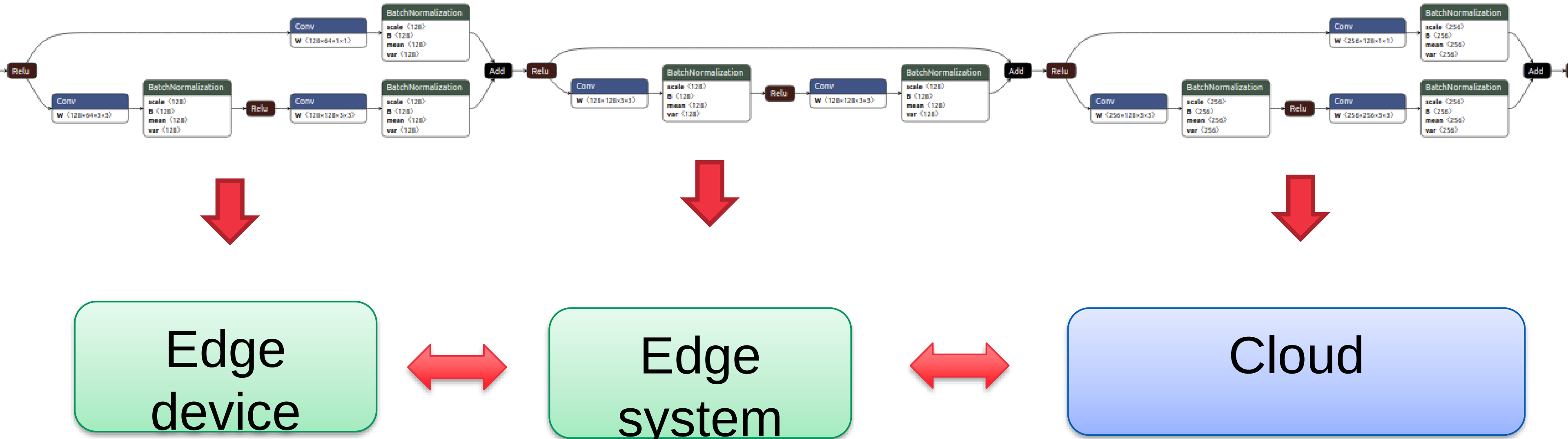
Partitioned models

- How to manage sub models for a partitioned model
- How to slice data for training and for inferences
- How to encapsulate complex runtime aspects to enable “virtualized” partitioned model serving



Distributed ML graph

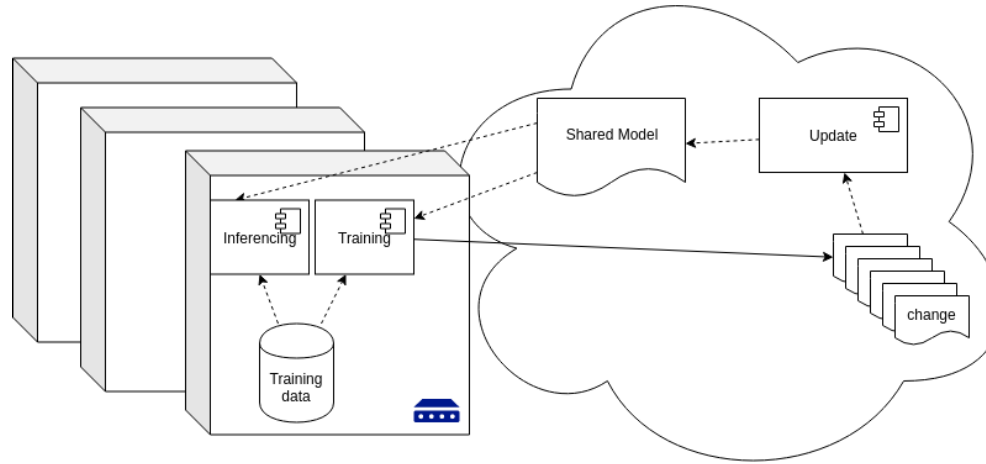
Assume that you can partition a complex ML graph, what could be possible issues?



How to partition? What would be the exchanges among subsystems

Selected problems: federated training with edges

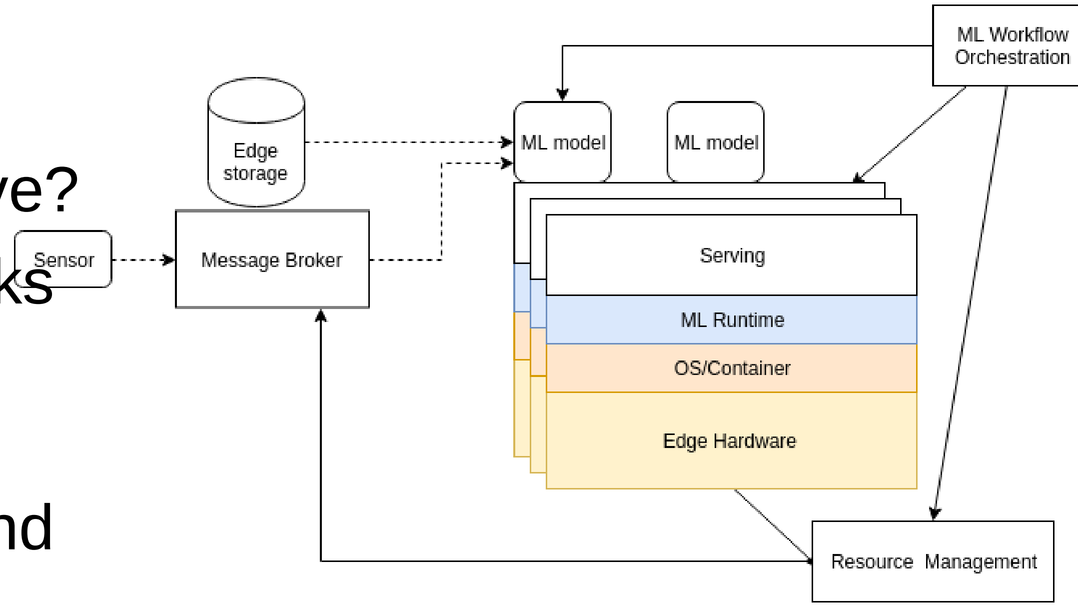
Machine learning is decentralized with a distributed set of devices holding data and carrying out (sub) training/inferencing



- **What about Reliability and Resilience?**
  - Consensus in updates, secured aggregation protocols, dynamicity and elasticity

Selected problems: ML Serving

- **ML Serving (and R3E)**
  - Which types of dynamic service models we could have?
  - How to distribute tasks in model serving?
  - How to partition ML tasks in both edge and cloud?



Study log

- **<u>No study log but</u> read papers and do the hands-on tutorial**
- **You can pickup some points mentioned as the topic for your individual project**
  - Or incorporate some ideas into your individual project

- **We expect ML with edge systems will increasingly been developed for many advanced software systems!**
  - Good areas for master theses/research projects.

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io