VLAIO TETRA

Machine Learning Operations for Edge Condition Monitoring

(MLOps4ECM)

Part I – MLOps and Edge Intelligence in Practice: Industry Tech-Talks

With the support of

VLAIO

HBC.2023.0062

MLOps and Edge Intelligence in Practice

TECH TALKS

MLOps for video fire detection on surveillance cameras

Araani – Maarten Callens

Real-time processing on PLC's with AI

CTRL Engineering – Glenn De Loose

Managing an AI server: Kubeflow

Howest – Thomas Huyghebaert

Smarter data pipelines with Dagster

Leap Technologies – Wannes De Groote

How does Siemens enable industrial grade AI

Siemens – Niels Debusschere

AI Ops Capability Maturity Model

Superlinear – Pierre Gerardi

No-nonsense approach to MLOps

Vintecc – Gilles Ballegeer

With the support of VLAIO

VLAIO TETRA

Machine Learning Operations for Edge Condition Monitoring

(MLOps4ECM)

Lunchbreak

With the support of

VLAIO

HBC.2023.0062

VLAIO TETRA

Machine Learning Operations for Edge Condition Monitoring

(MLOps4ECM)

Part II – Project results

With the support of VLAIO

HBC.2023.0062

KU Leuven Bruges – M-Group

- 8 professors
- 1 research manager
- 1 project office manager
- 6 post-docs
- > 40 junior researchers
- 2 ATP test engineers

Mechanical Engineering

Electrical Engineering

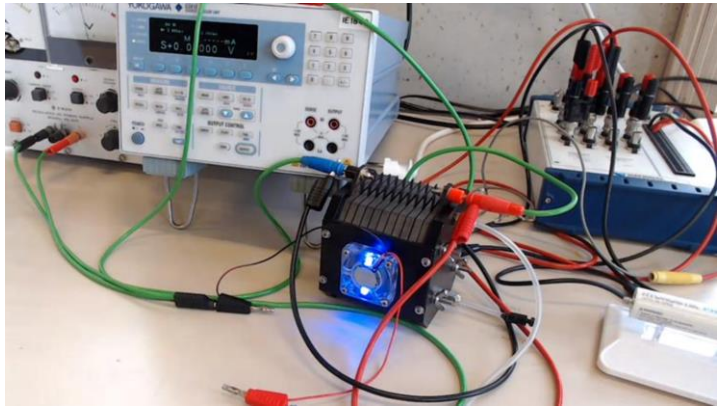
Computer Science

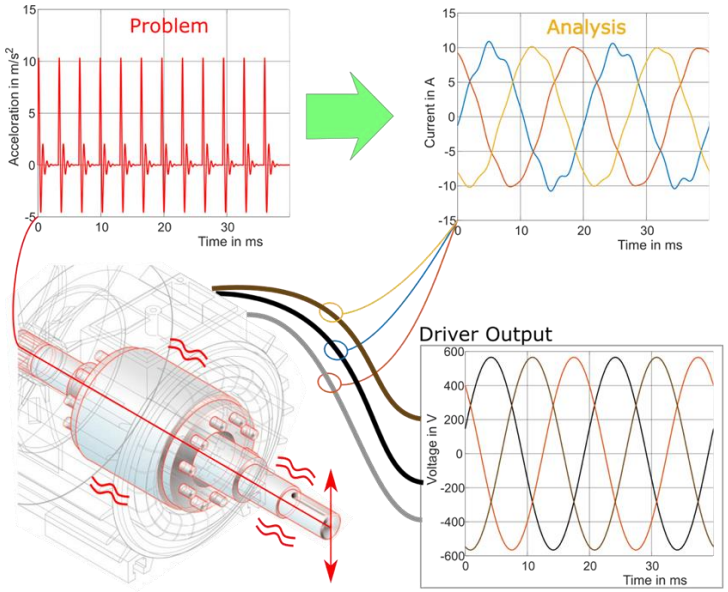
With the support of VLAIO

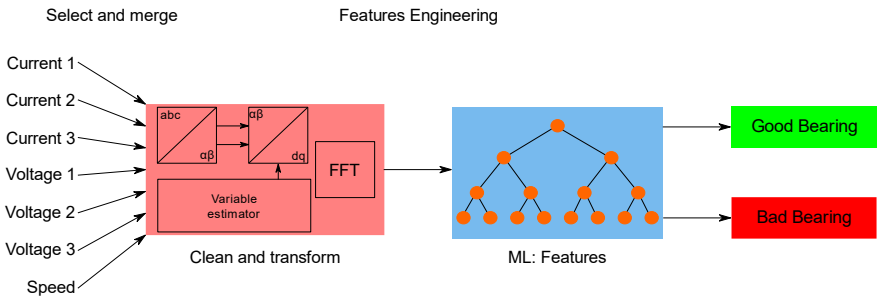
KU Leuven Bruges – M-Group

The Ultimate Machine

- EMC test chambers
- Climate chambers
- Fuel cells
- Smart conditioning monitoring
- Edge computing







With the support of VLAIO

KU Leuven Bruges – M-Group

The Ultimate Factory

- Data-driven process optimisation
- Autonomous systems
- Quality control

With the support of VLAIO

KU Leuven Bruges – M-Group

AI for complex mechatronic systems and processes

OPTIMISATION

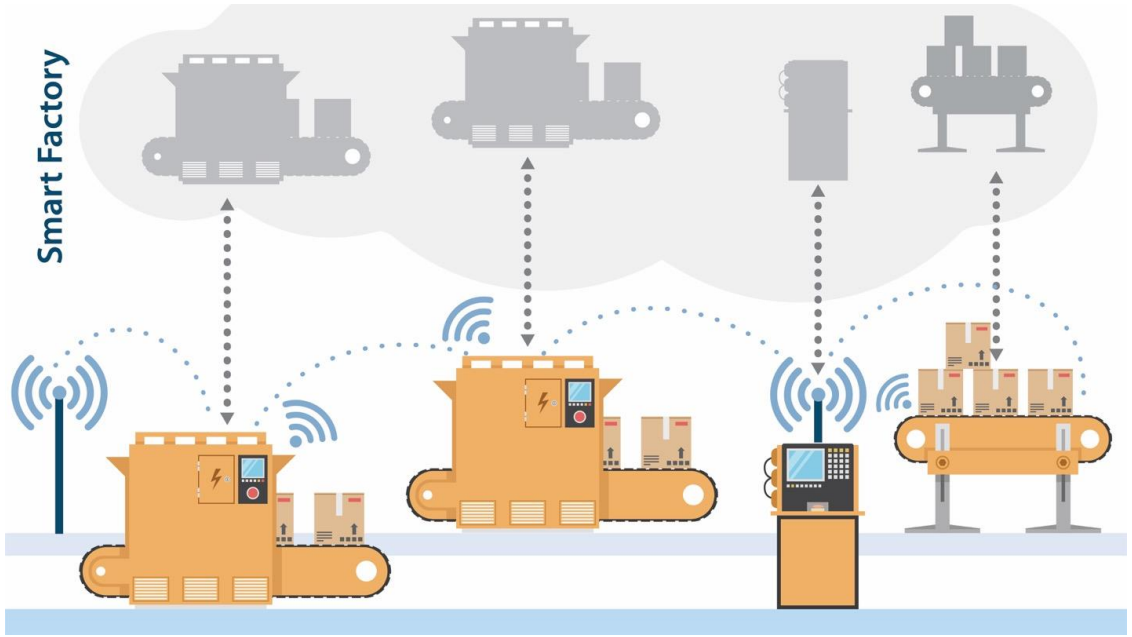
How can unwanted effects be mitigated?

- Link between process parameters & conditions and product quality

MONITORING

What is going wrong and what is the underlying reason?

- Abnormal machine behavior
- Product quality defects



CONTROL

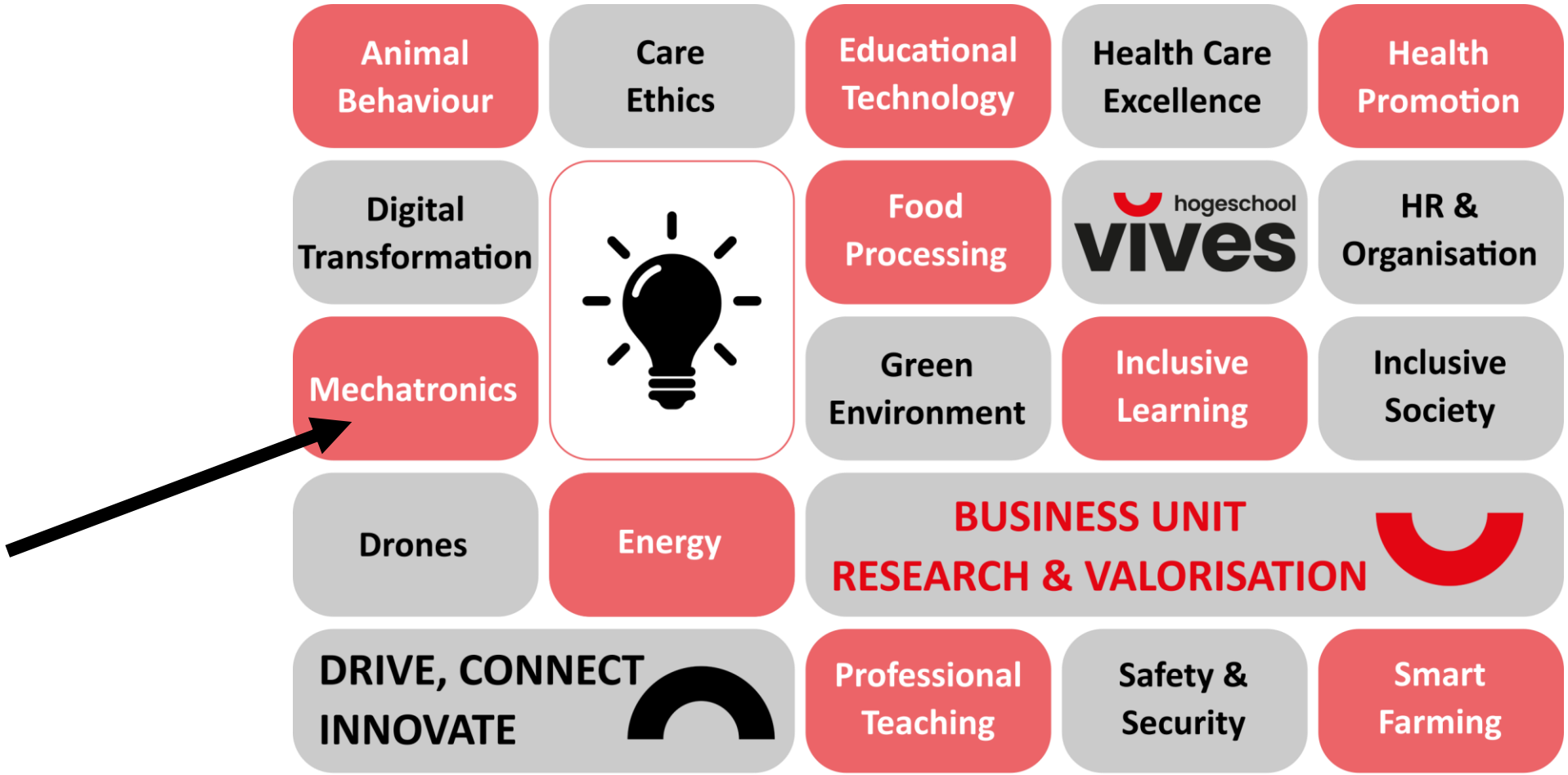
How can the parameter tuning be automated?

- Realise a stable system behavior

TRANSVERSAL ASPECTS:
SAFETY, ROBUSTNESS, EFFICIENCY

With the support of VLAIO

Research at VIVES University of Applied Sciences



With the support of VLAIO

Research Group Mechatronics – Team

Hanne Van den Berghe
Research Manager

Geert Furniere
Researcher
Teacher

Emiel Wallays
PhD Researcher

Noah Debaere
Researcher

Jonas Lannoo
Innovation Manager
Senior Researcher

Jeremy Lebon
Researcher
Teacher

Tom Van Gaever
Researcher
Teacher

William Van Nieuwenhove
Researcher

Alexander D'hoore
Researcher
Teacher

Philippe Van Overbeke
Researcher

Sille Van Landschoot
Researcher
Teacher

Carl Grimmelprez
Researcher
Teacher

Sam Lefebvre
Researcher

Peter Vanbiervliet
Researcher

Philip Vanloofsvelt
Researcher
Teacher

Karolien Vandersickel
Researcher
Teacher

Catherine Middag
Researcher

Tineke Verkest
Researcher

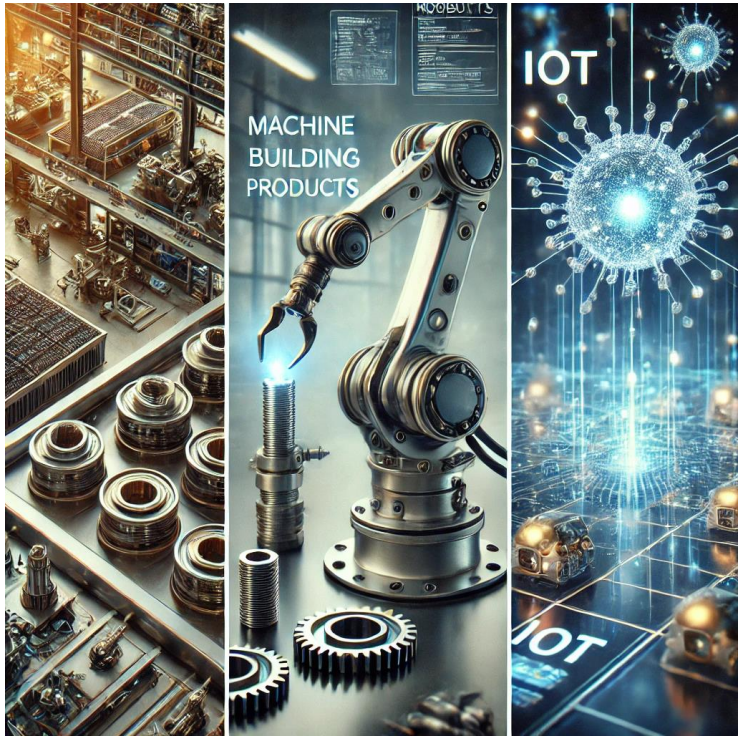
With the support of VLAIO

Research Group Mechatronics

Two complementary research tracks

1. **Machinebuilding and product**
   - Sustainable product design
   - Flexible and efficient production lines
   - AI-driven autonomy

2. **Internet of Things and data**
   - End-to-end prototyping
   - Digitalisation for industry
   - (I)IoT and Edge AI

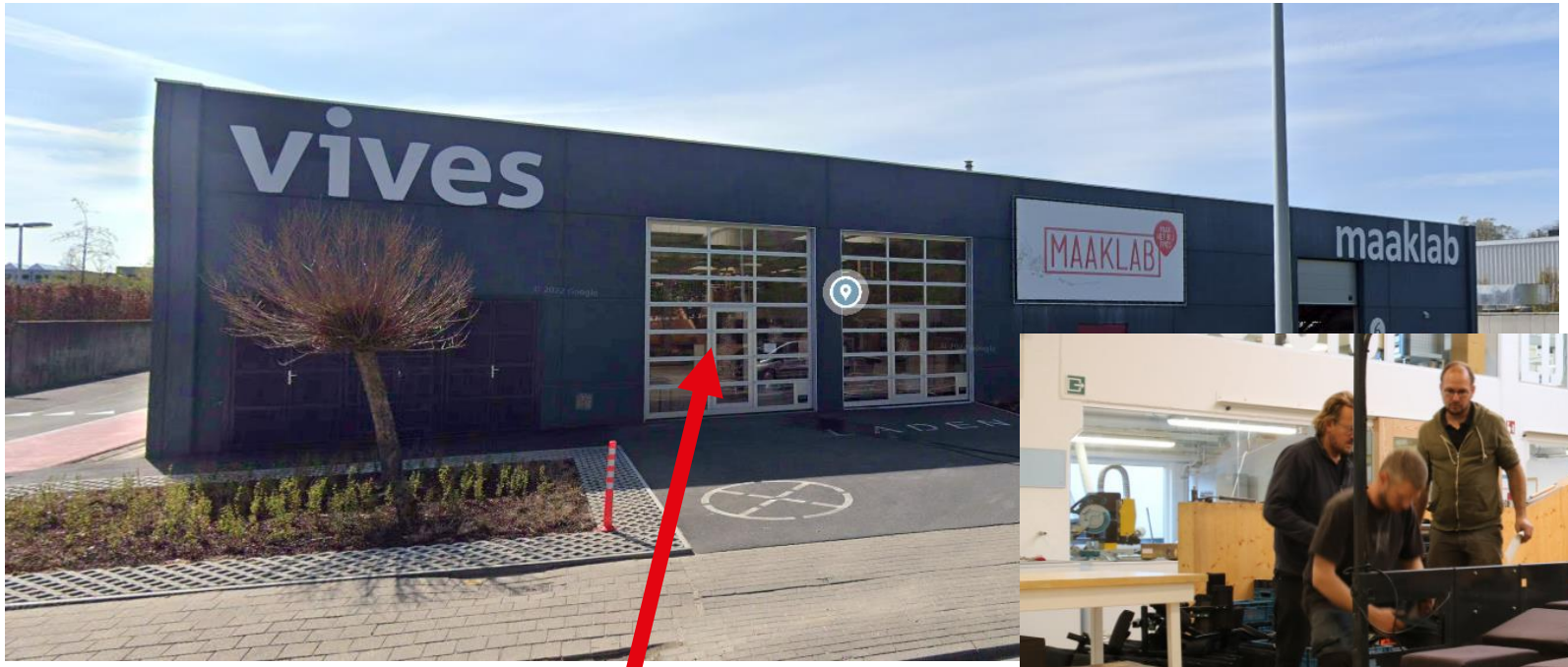


→ Three labs in three locations

With the support of VLAIO

Research Group Mechatronics

Maaklab @ VIVES Campus Kortrijk



With the support of VLAIO

Maaklab @ VIVES Campus Kortrijk

- **Research & services**
  - Product design & materials
  - Prototyping & proof-of-concept
  - Custom parts & machine building
  - Automation & 3D scanning

- **Infrastructure**
  - Extensive machine park
    - CNC, Waterjet, Injection Molding
    - WAAM, EDM (spark erosion), 3D printing
  - Scanning infrastructure
  - Machine vision

With the support of VLAIO

Research Group Mechatronics

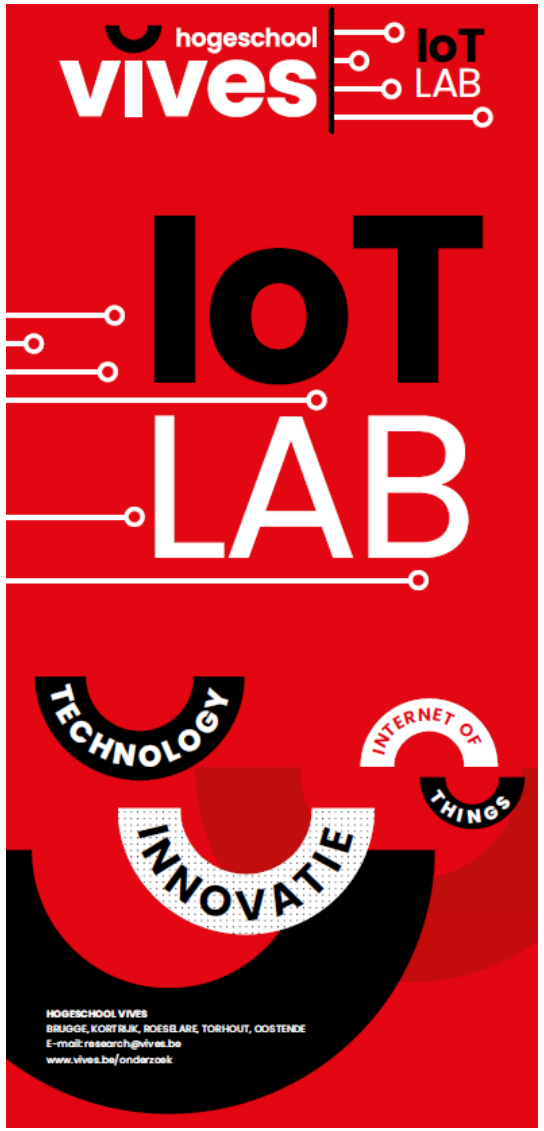
IoT-lab @ VIVES Campus Brugge

With the support of VLAIO

IoT-lab @ VIVES Campus Brugge

- **Development of IoT systems and architectures**
  - Custom hardware, firmware, software
  - Connectivity & low-power
  - Implementation of AI on IoT (edge/tinyML)

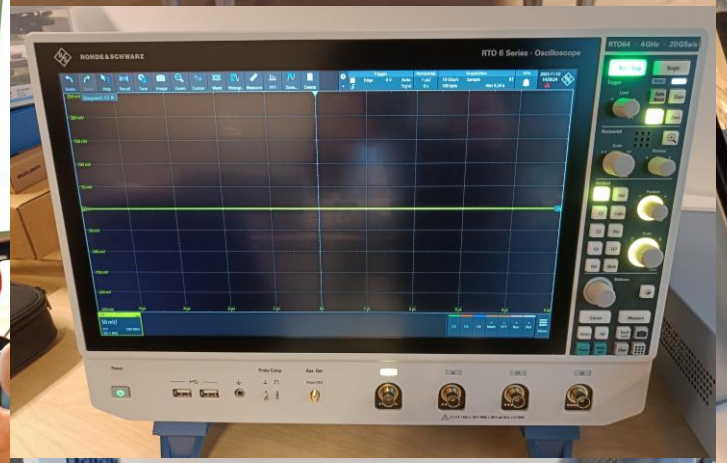
- **Infrastructure**
  - Creating prototypes
  - Test equipment
  - Calculation & hosting servers



With the support of VLAIO


2303-002-000


ROHDE&SCHWARZ
SMCV100B
Vector Signal Generator
1.000 000 000 000 GHz
-30.00 dBm
-30.00 dBm


ROHDE&SCHWARZ
RTO 6 Series - Oscilloscope

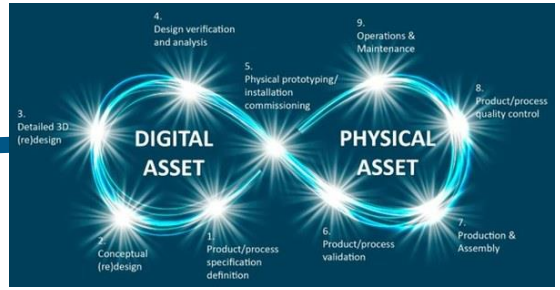

ROHDE&SCHWARZ
NGU201
Source Measure Unit
1.000 0 V
-1.000 00 A
1.000 00 A


ROHDE&SCHWARZ


DELL EMC
DELL EMC

New joint research lab

Infinity lab @ Flanders Make Kortrijk 4th Floor



INFINITY LAB

BY KU LEUVEN hogeschool vives

With the support of VLAIO

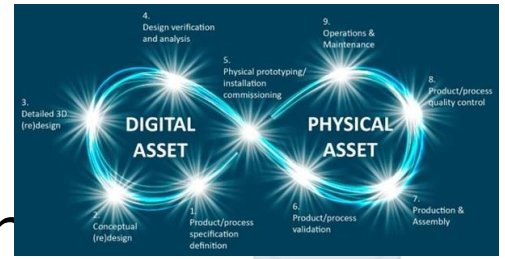
Infinity lab

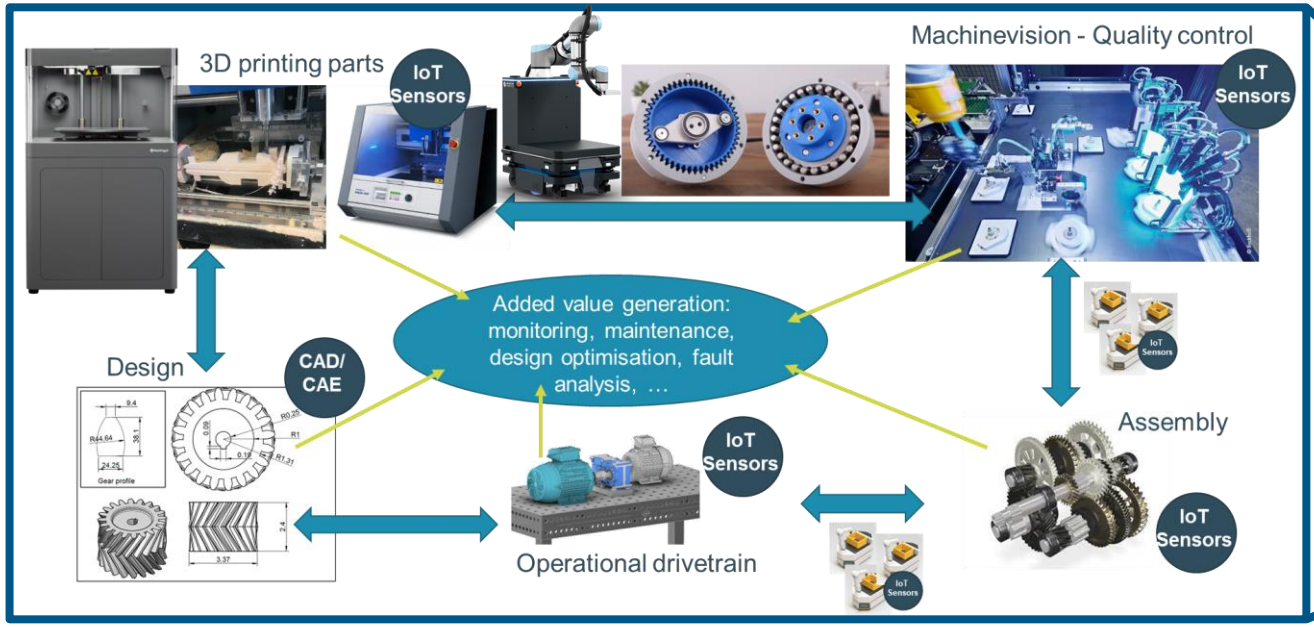
With the support of VLAIO

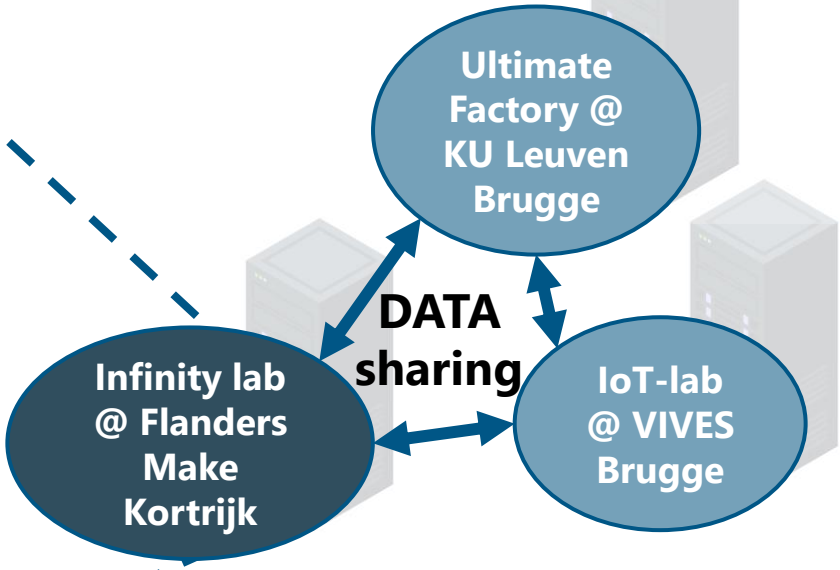
Infinity Lab

- Infinity + Product Lifecycle Management + AI Robotics

→ Straight-through digitalisation as motor for innovation







With the support of VLAIO

VLAIO TETRA

Machine Learning Operations for Edge Condition Monitoring

(MLOps4ECM)

Project Introduction

With the support of

VLAIO

HBC.2023.0062

MLOps4ECM - Motivation

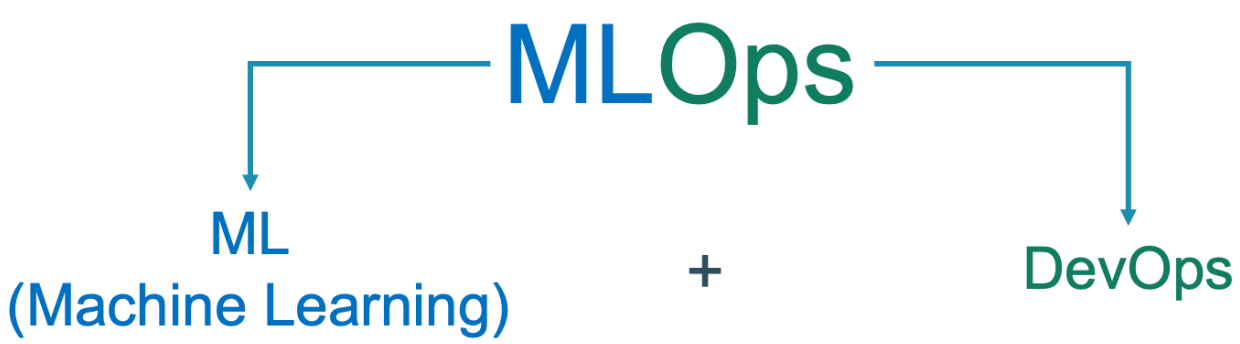
- Complex industrial processes require **data-driven models** to monitor and control operations
- Today, most AI developments remain limited to **R&D or lab-scale** test setups
- **Deployment in the field** demands continuous **monitoring** and feedback to ensure reliability
- **Automation and model updates** are essential to handle **environmental changes, drift, and maintenance cycles**
- Real-time decision-making close to the system = **edge condition monitoring**

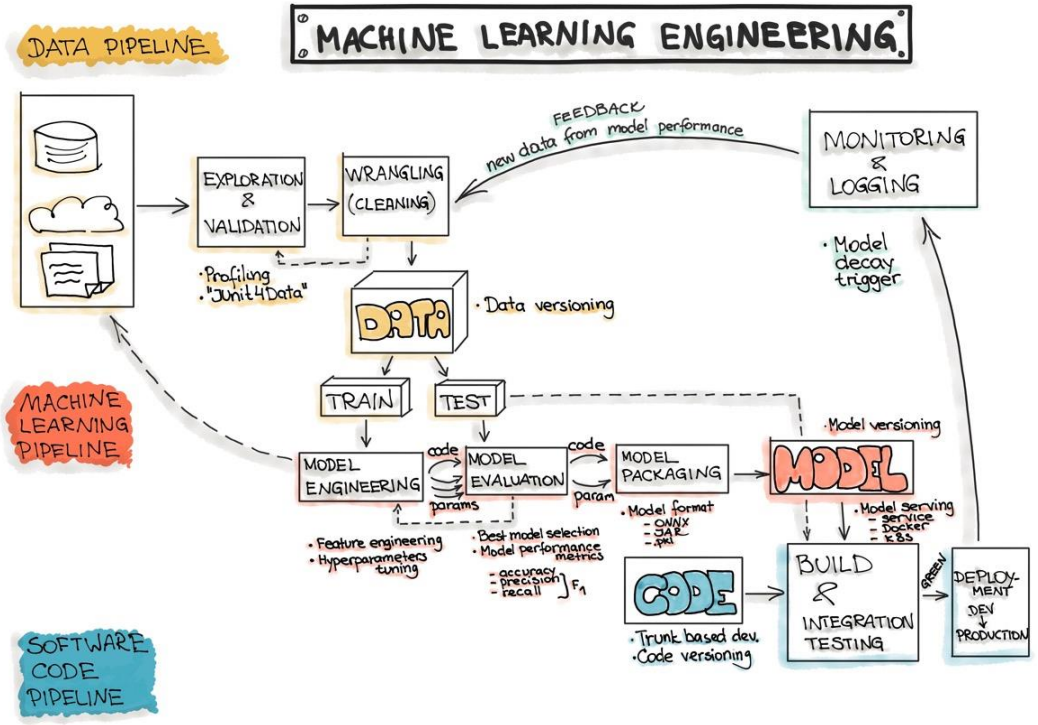
→ **MLOps** = cost reduction, consistent quality, increasing operational efficiency

With the support of VLAIO

MLOps for Edge Condition Monitoring

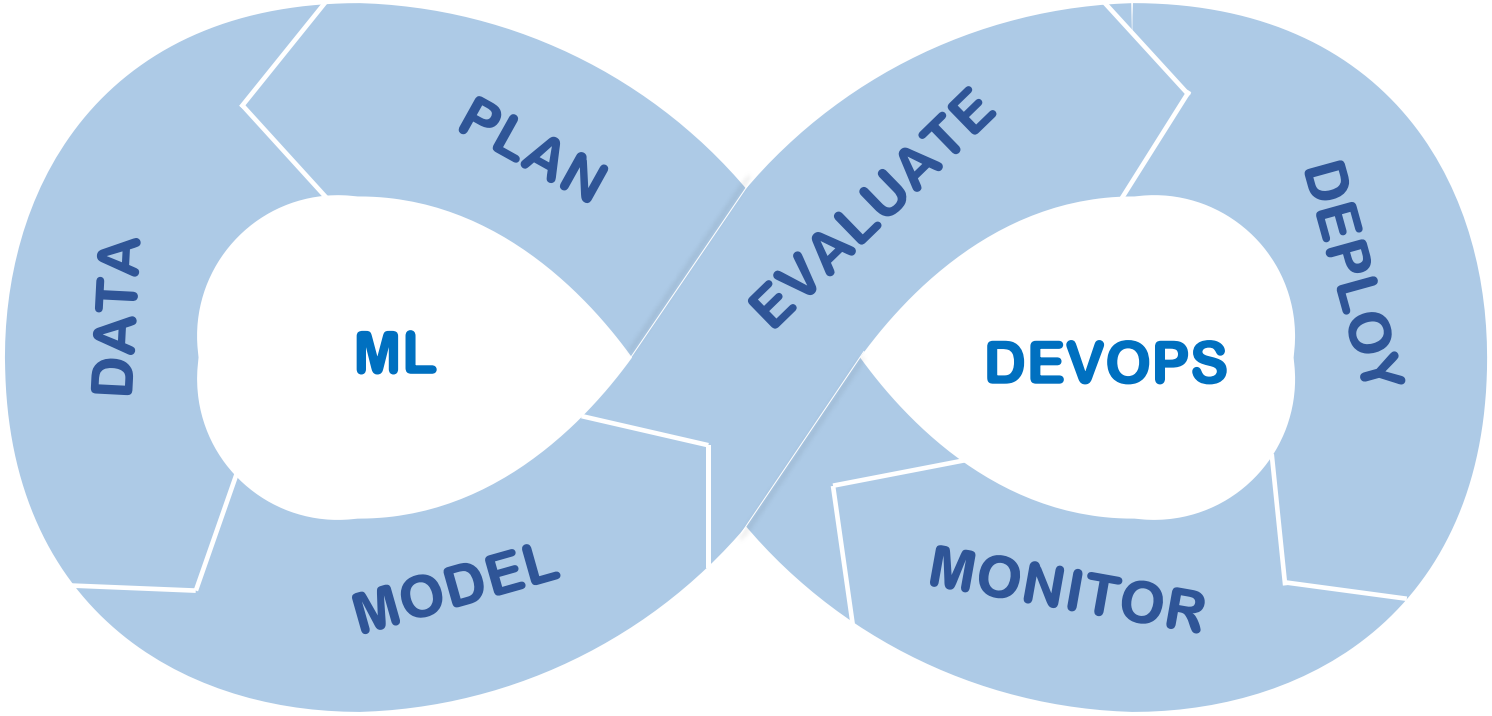
- MLOps = Paradigm used to **create** and **maintain** a machine learning model that will be deployed in **production**





With the support of VLAIO

MLOps lifecycle



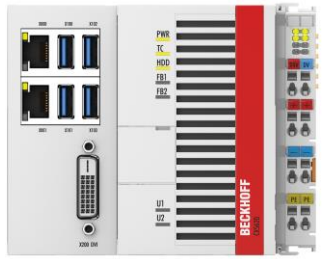
With the support of VLAIO

MLOps4Edge repertoire

Tools

ONNX
CLEARML
DVC
ONNX RUNTIME
EVIDENTLY AI
mlflow
GitLab
PREFECT
APACHE ARROW
APACHE Spark
RAY
OPTUNA
MINIO
dagster

Devices





MLOps4ECM – Goals

Enabling companies to adopt MLOps practices for edge condition monitoring through the structured transfer of knowledge, tools, use cases and demonstrators in a pragmatic, industry-oriented manner.

- Goal 1: Overview of available knowledge, needs, problems + definition of use-cases
- Goal 2: Overview of DevOps and MLOps tools
- Goal 3: Development and implementation of use-cases
- Goal 4: Development and implementation of two demonstrators
- Goal 5: Show the economic impact of using MLOps
- Goal 6: Dissemination of the knowledge through seminars, workshops, education

With the support of VLAIO

Website

- General project info
- **Workshops**
- Guidelines
- Use-cases
- Demonstrators

Visit:

Outline

Now

Drift detection on edge devices

Lara Luys

Training, testing and deploying AI models on the edge

Sille Van Landschoot

Machine vision pipeline for efficient edge deployment

Alexander D'hoore

Follow-up projects

16h00

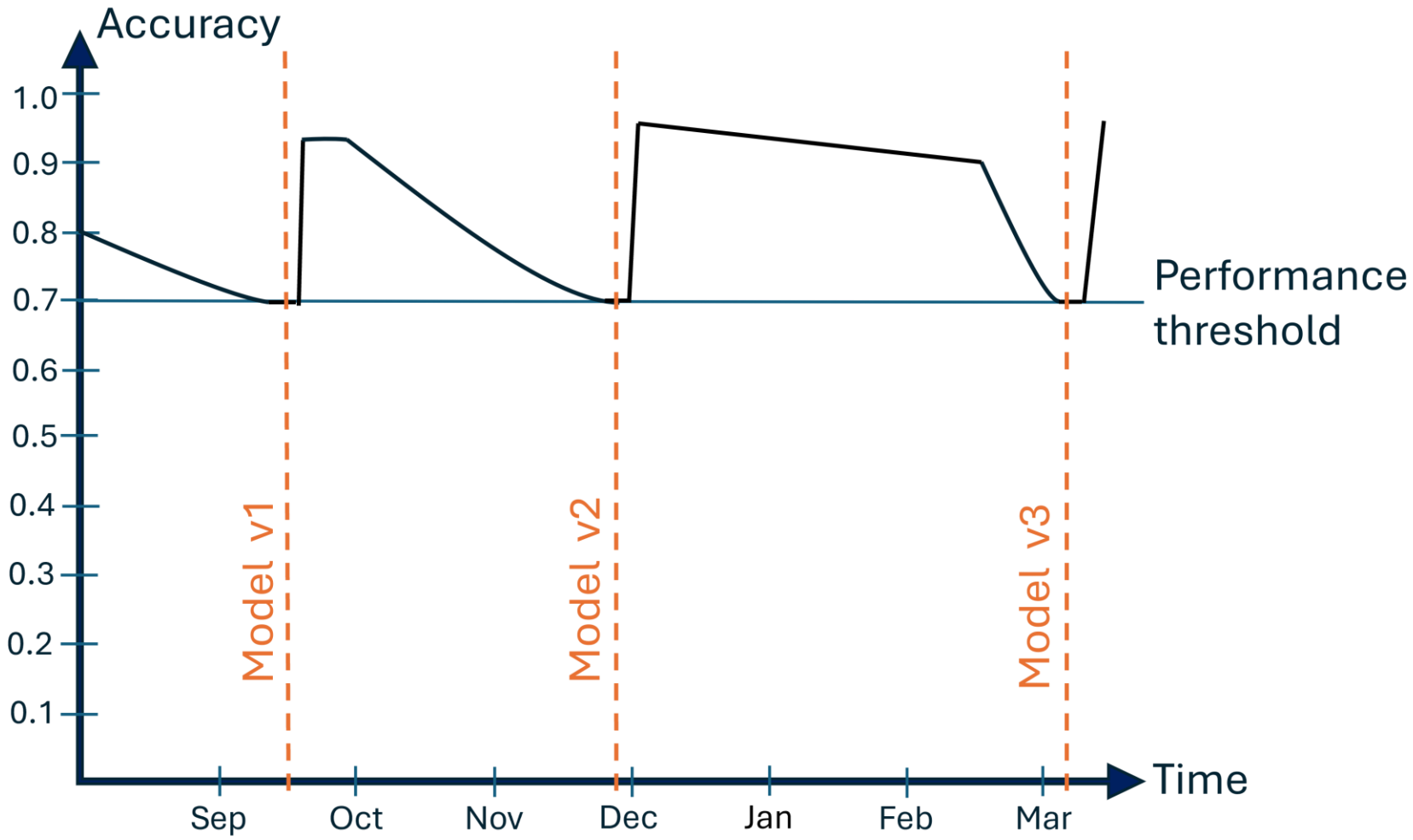
Reception with demos

With the support of VLAIO

Drift detection on edge devices

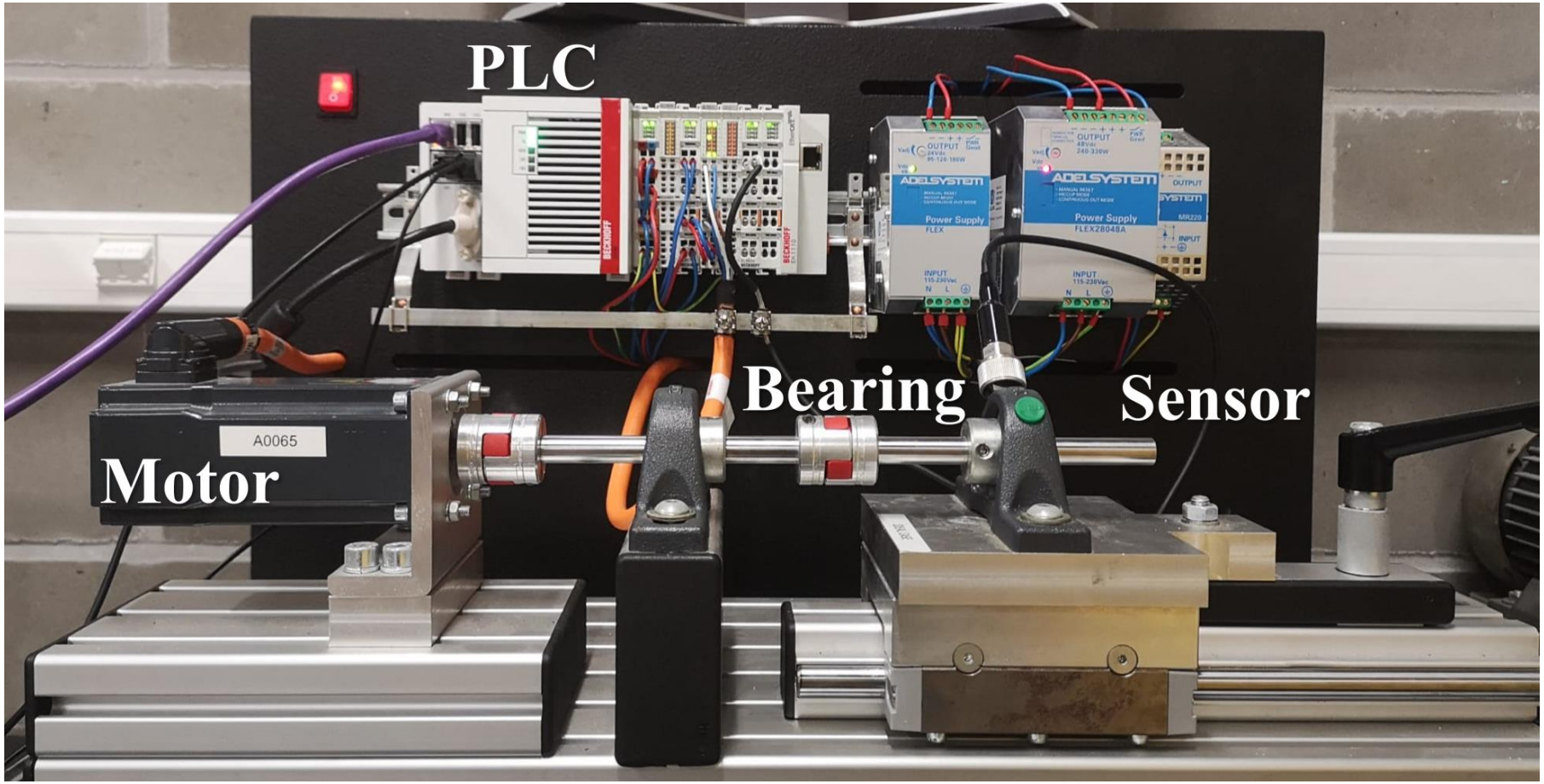
Lara Luys

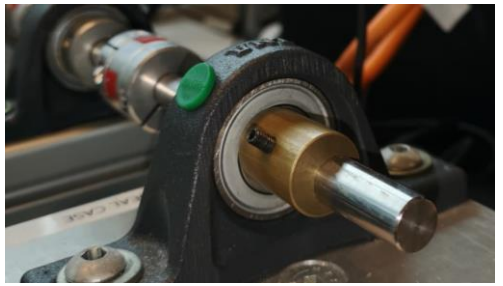
What is drift?



With the support of VLAIO

Time series drift detection demo (icw Beckhoff)

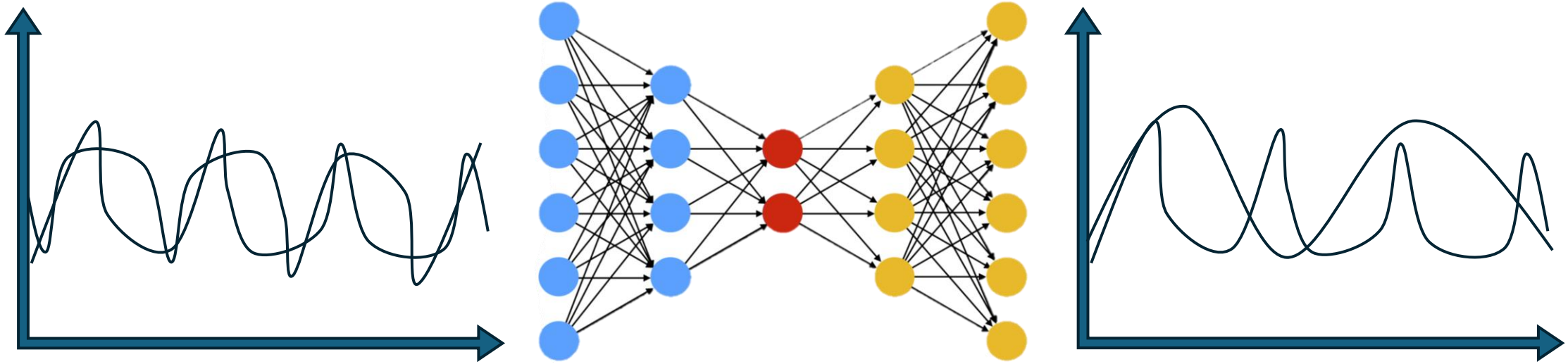




With the support of VLAIO

Drift detection demo (icw Beckhoff)

- Goal: Drift detection of vibration data with autoencoder



With the support of VLAIO

Magic Marble Machine demo (icw CTRL Engineering)

- Student projects
  1. Creation of the magic marble machine
  2. Creation of machine vision models to compare timing for real-time applications.
- MLOps on PLC with vision

With the support of VLAIO

Vandewiele use-case: The setup

VAN DE WIELE

- ML for fault detection in weaving machines
  - Post process
  - Realtime
- 2 datasets: Only faulty data
  - R&D dataset = training data
  - Production dataset → change of sensors + brakes
  - 3 possible drift moments
- Given labels fault detection: results of rule-based algorithms

With the support of VLAIO

Vandewiele use-case: Research questions

1. Does the change of setting, sensors or brakes cause data drift?

2. Which type of edge device is needed for drift on the edge?

With the support of VLAIO

Vandewiele use-case: Is there drift?

- Try different types of drift detection with retraining
- Supervised (use labels)
  - DDM
  - ADWIN
  - EDDM
  - STEPD
- Unsupervised (No labels)
  - KS
  - D3
  - OCDD
  - SPLL
  - BNDM

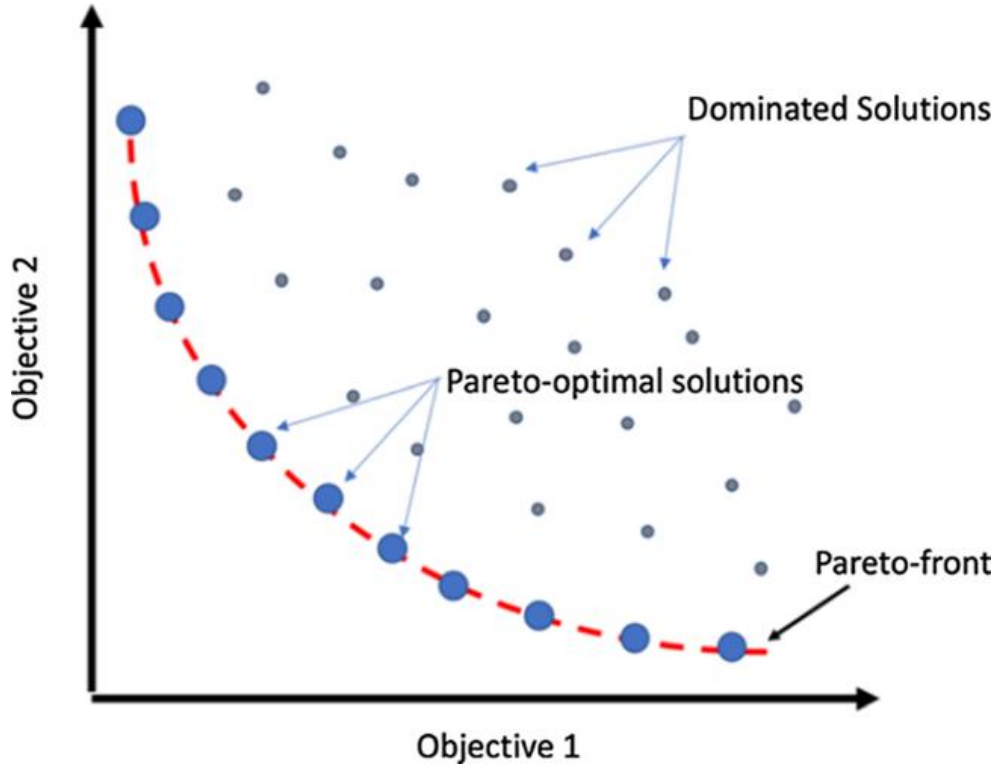
Have hyperparameters to tune sensitivity of drift detector

Use multi-objective hyperparameter search with Optuna

With the support of VLAIO

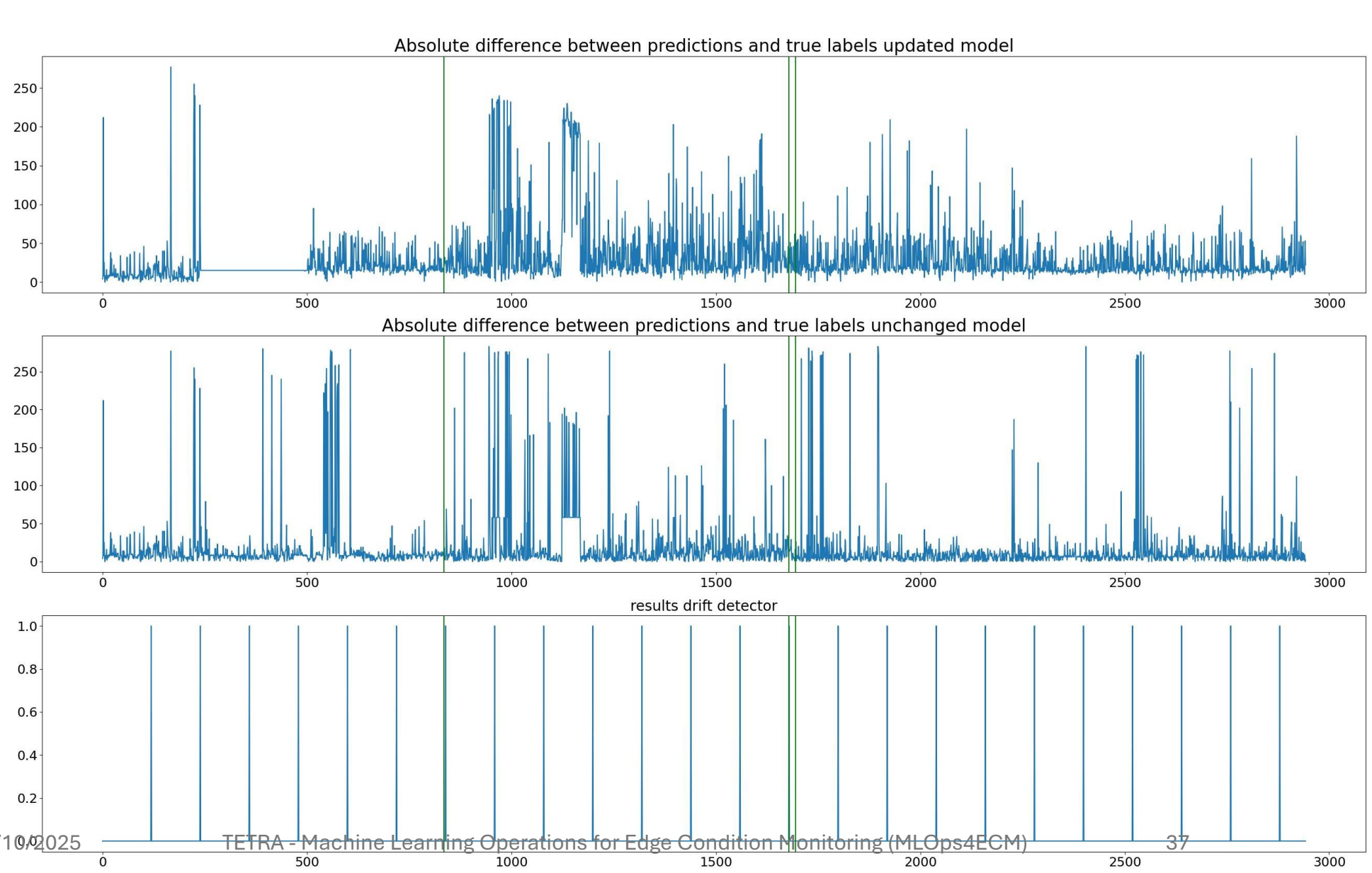
Vandewiele use-case: Multi-objective search

- 3 Metrics which compare results detectors to possible drift moments
  - Mean time to detection = MTD
  - Mean detection ratio = MDR
  - Number of false detections = FD
- Result = Pareto front

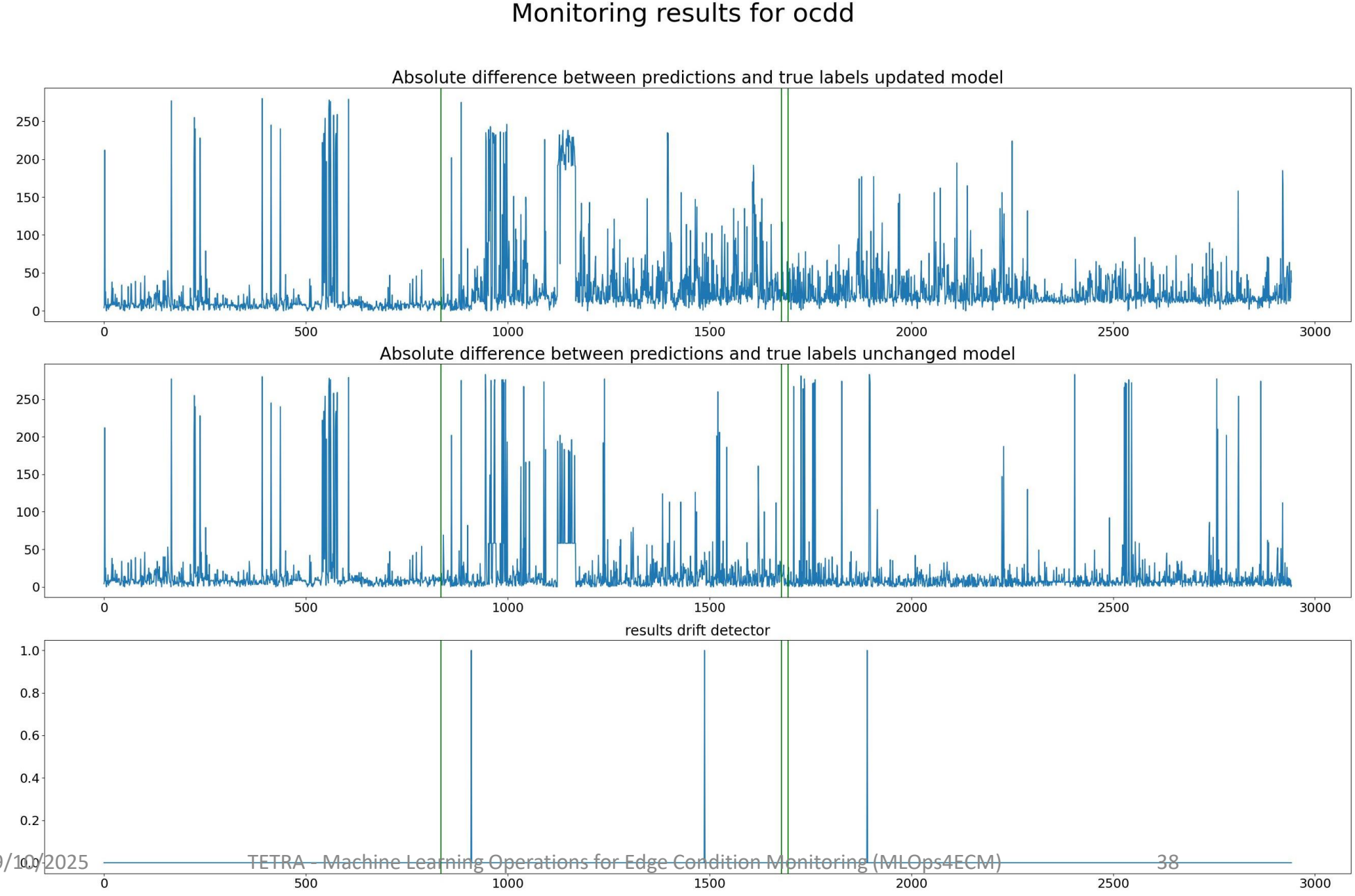


With the support of VLAIO

Vandewiele use-case: Results



Vandewiele use-case: Results



Vandewiele use-case: Conclusion question 1

- Given data is very noisy

  → Faulty data is very inconsistent: Each failure is different

- How to improve?

  → Retry with both faulty and working data
  this data will be more stable.

  → Drift detectors = made for “streaming data”

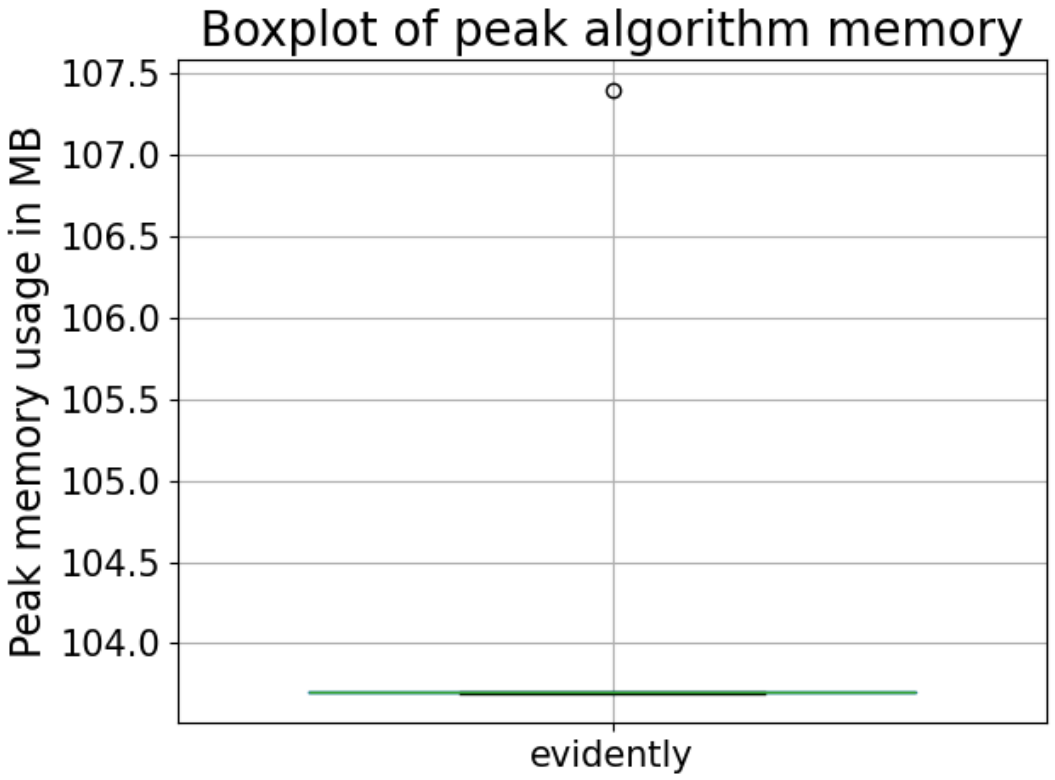
With the support of VLAIO

Vandewiele use-case: which edge device is needed for implementation?

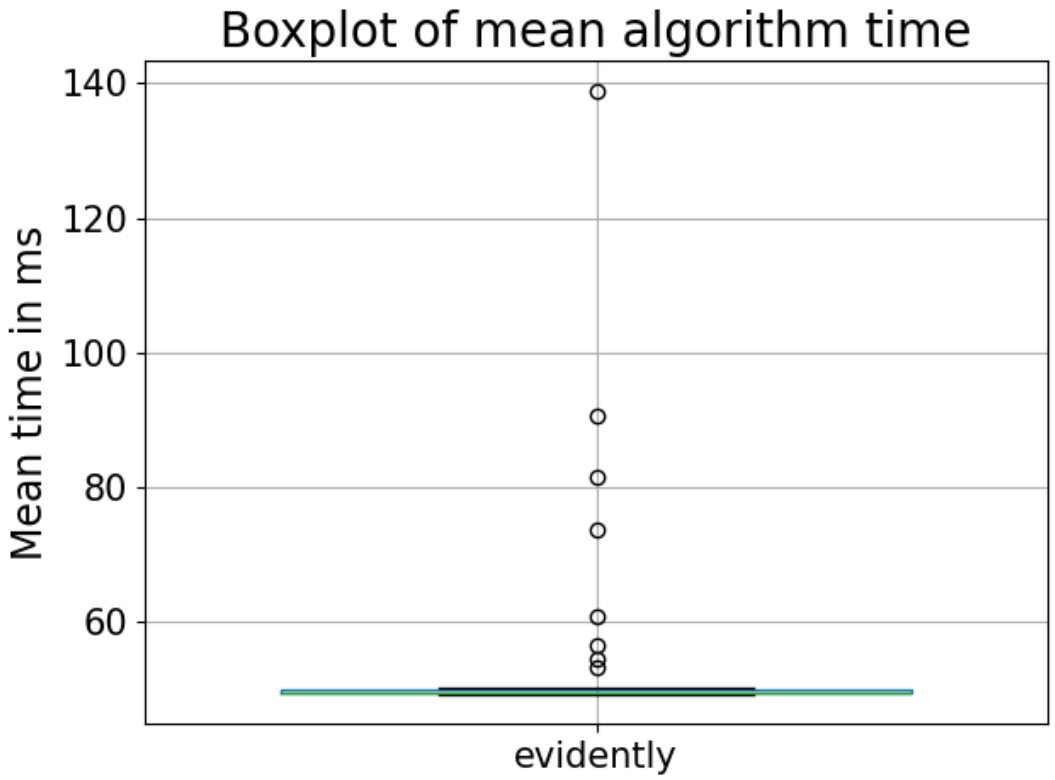
- Use of tool: Evidently
  - Advantage: Easy to implement + nice looking reports
  - Disadvantage: uses a lot of recourses, uses simple threshold techniques
- Drift detectors
  - Advantage: smaller resource usage (depends on detector)
  - Disadvantage: More difficult to implement

With the support of VLAIO

Vandewiele use-case: Evidently resources



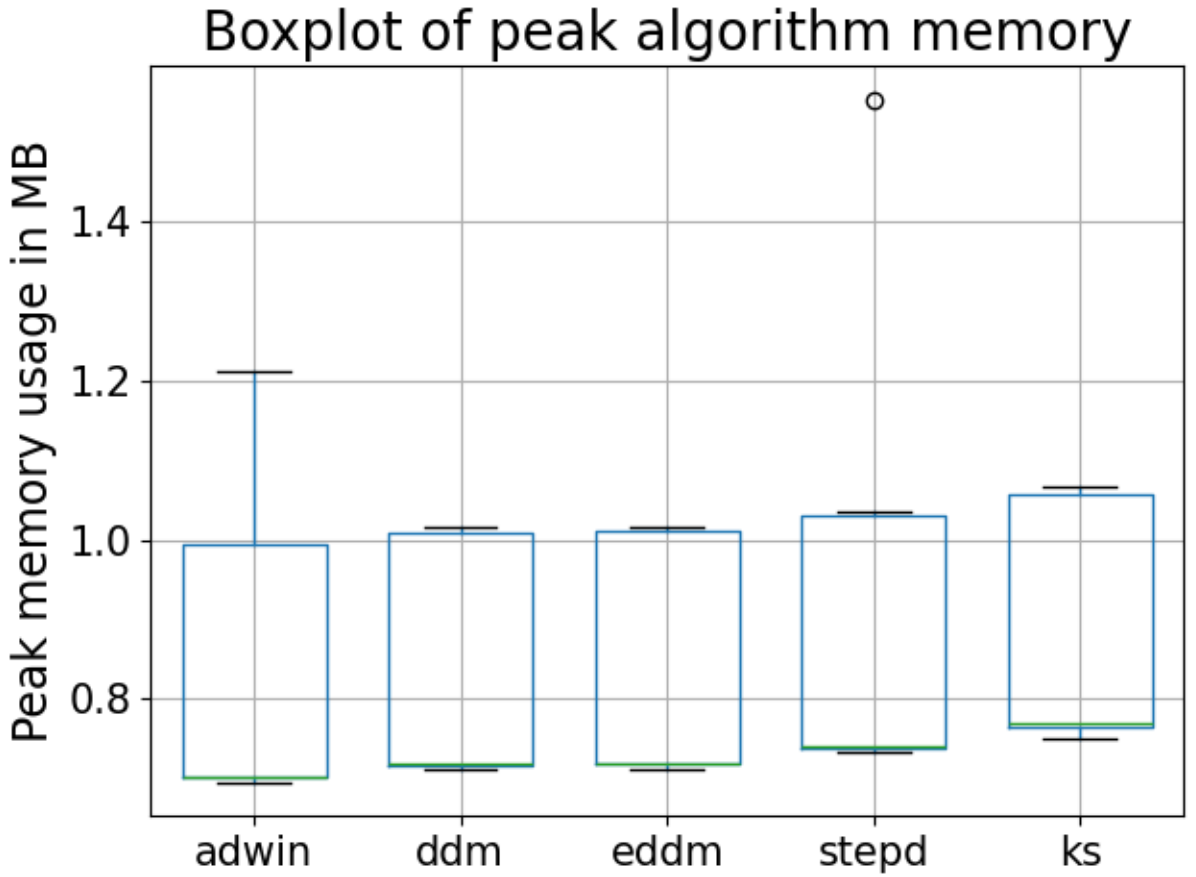
Includes model inference + drift detection



Includes only drift detection

With the support of VLAIO

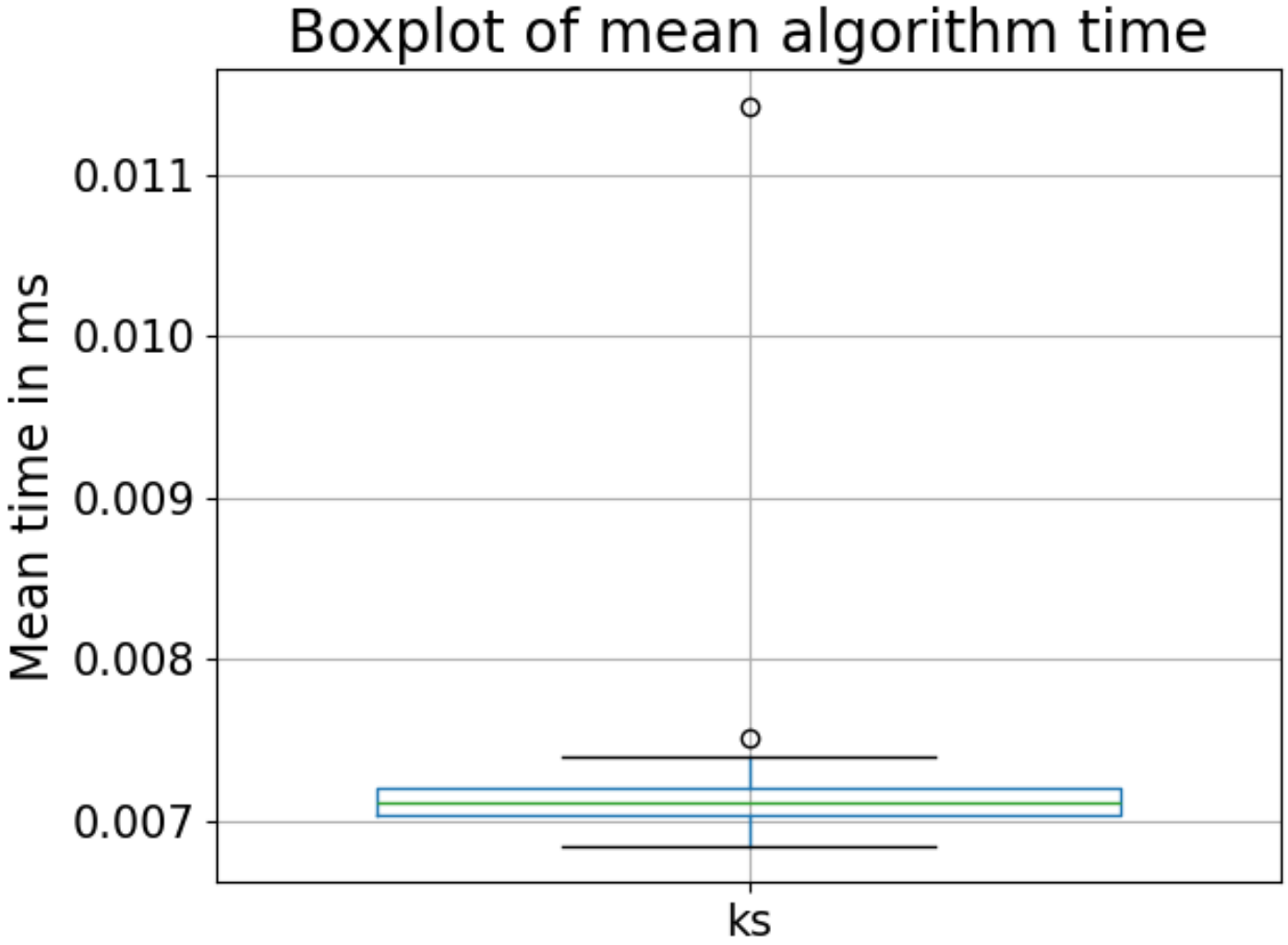
Vandewiele use-case: Drift detectors memory



Includes model inference + drift detection

With the support of VLAIO

Vandewiele use-case: Drift detectors timing



Includes only drift detection

With the support of VLAIO

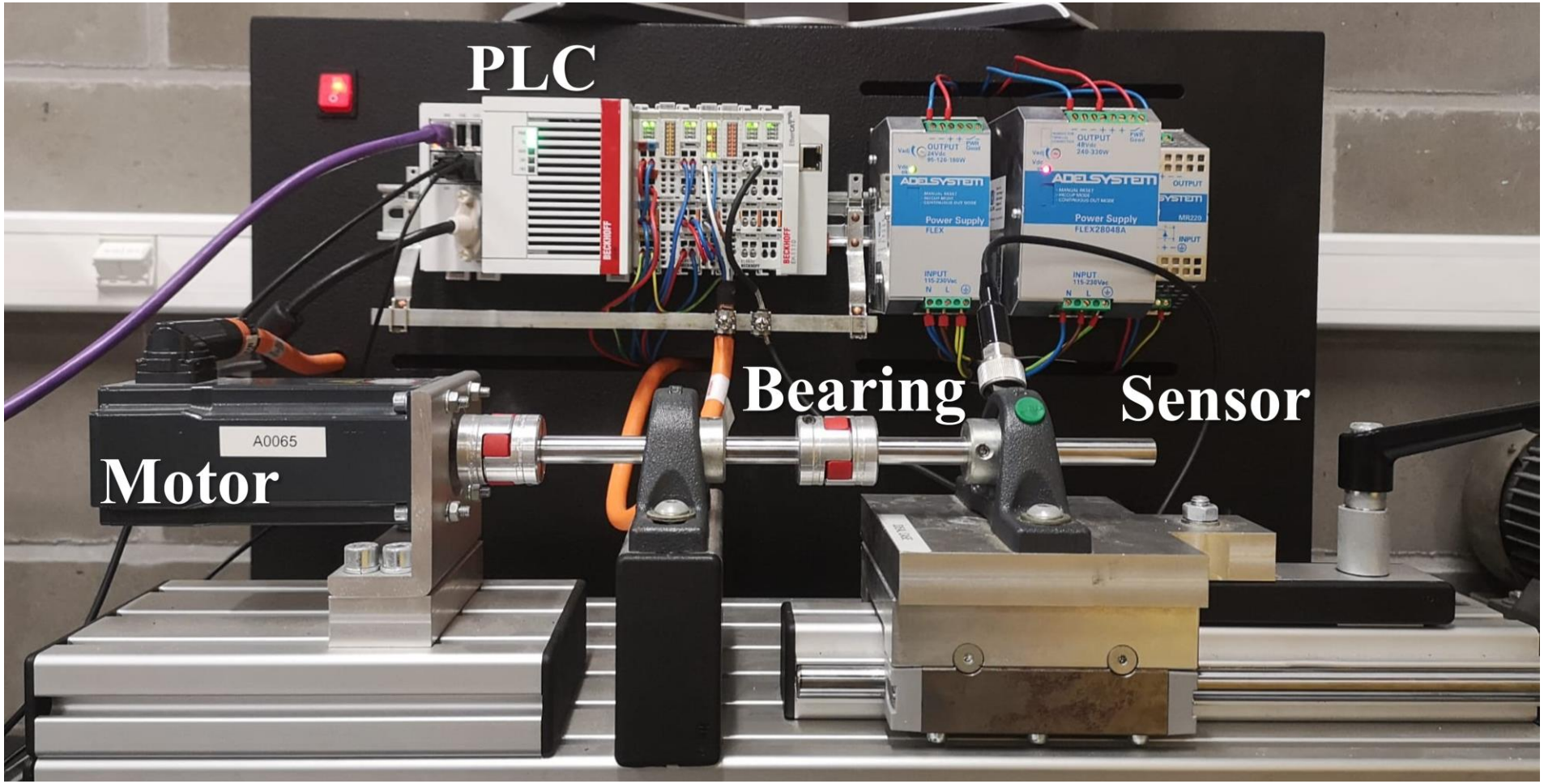
Vandewiele use-case: Type of edge device

- If Evidently: > 100MB
  - Microprocessors: Needs Linux or Windows to run Python
  - Linux boards, industrial PC's, PC-based PLC, etc.
  - E.g. Jetson, ARM Cortex A

- If drift detectors: > 2-6 MB
  - Microcontrollers: Translation to C/C++ needed
  - E.g. ARM cortex M, ESP32

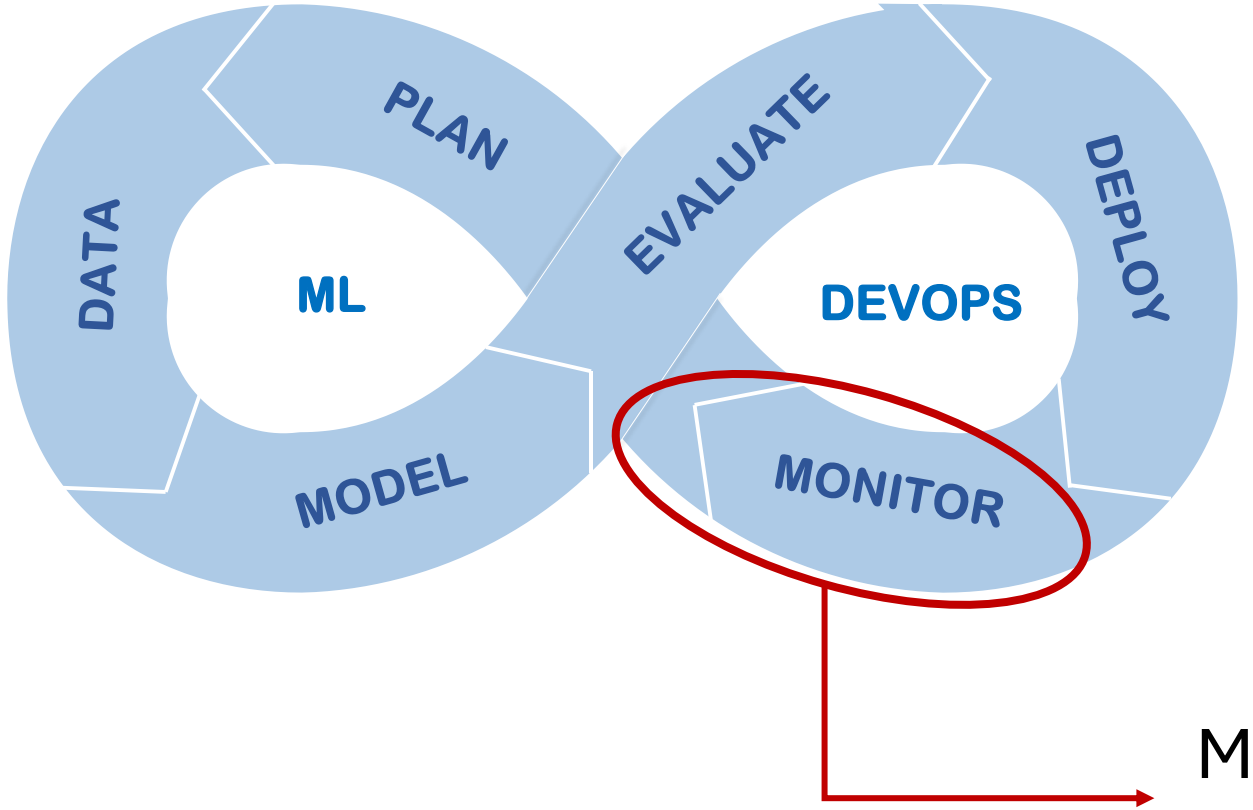
With the support of VLAIO

Beckhoff use-case: The setup



With the support of VLAIO

Beckhoff use-case: Monitoring in production

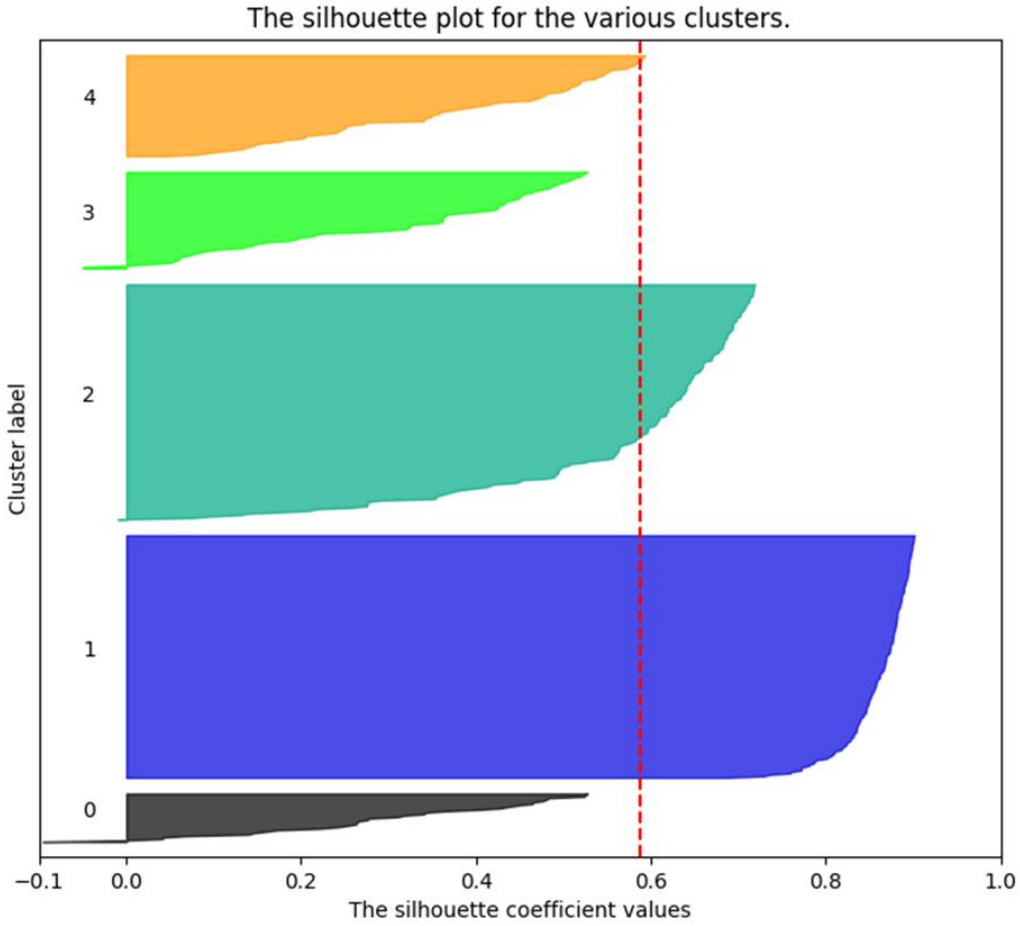


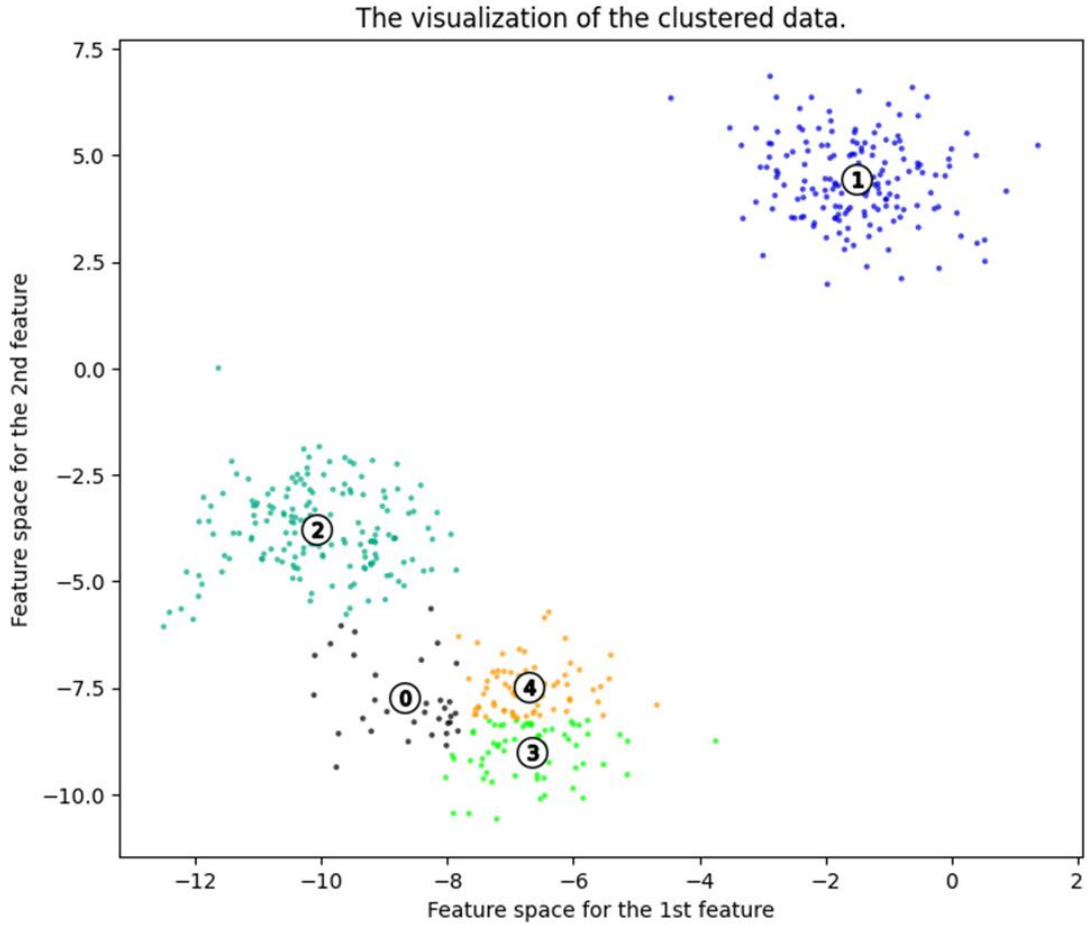
Use latent space of autoencoder!

Memory efficient!
No true labels!

With the support of VLAIO

Beckhoff use-case: The latent space



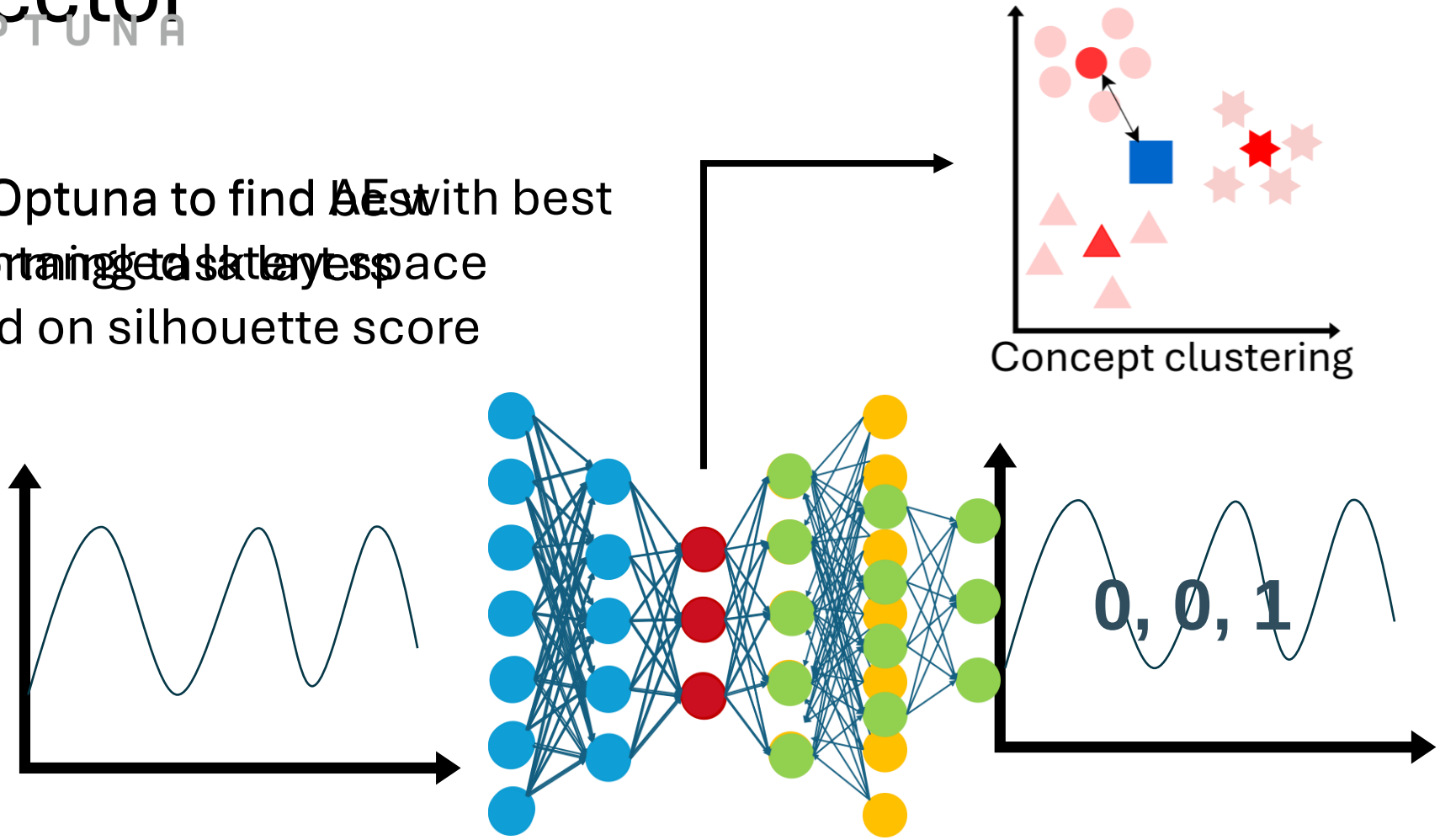


0

With the support of VLAIO

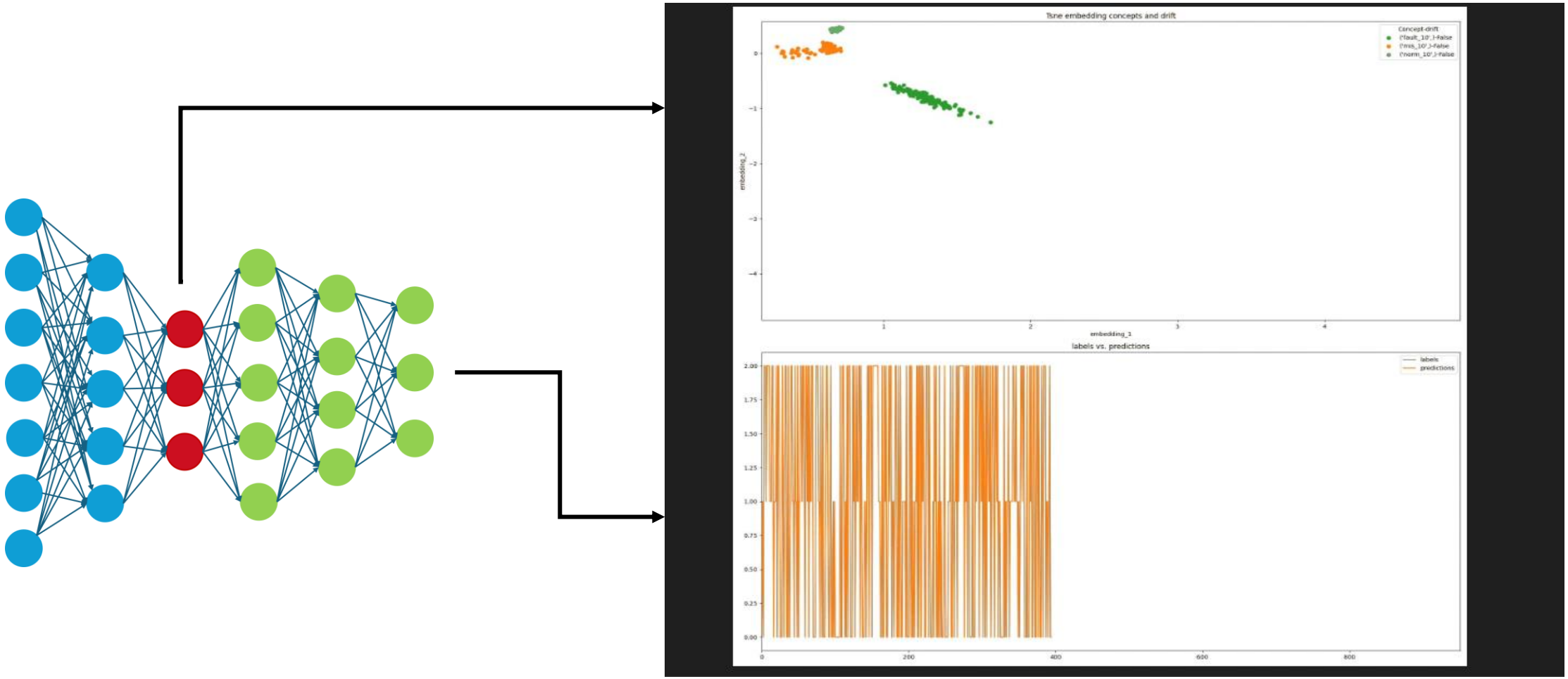
Beckhoff use-case: Latent-Space Drift Detector

Use Optuna to find AE with best performing latent space based on silhouette score



With the support of VLAIO

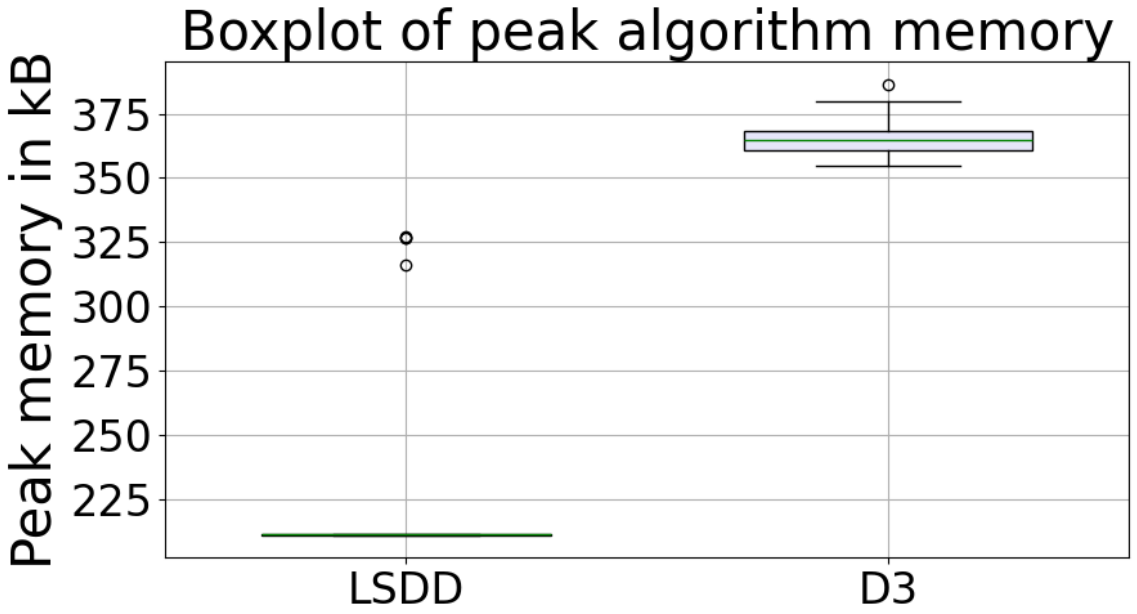
Beckhoff use-case: The results

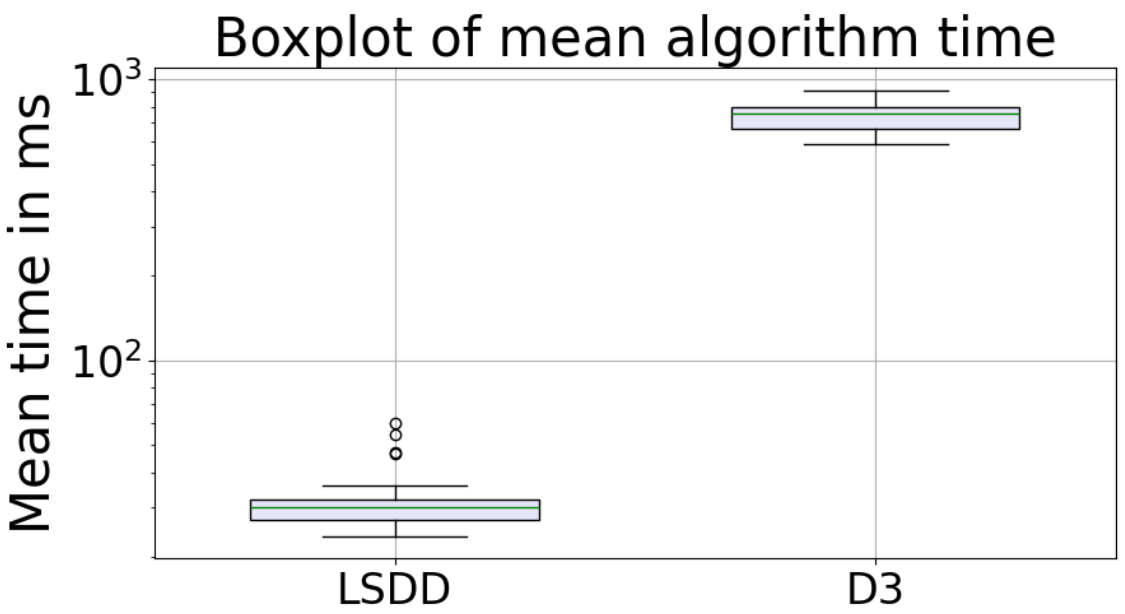


With the support of VLAIO

Beckhoff use-case: The results

Algorithm	Parameters	MTD	MDR	FD
LSDD	N=13, γ=0.077, k=73	2.666	0	0
LSDD	N=10, γ=0.05, k=74	2.666	0	0
D3	w=82, ρ=0.544, threshold=0.6525, k=47	8.5	0.333	0

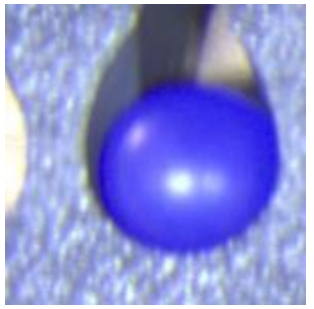


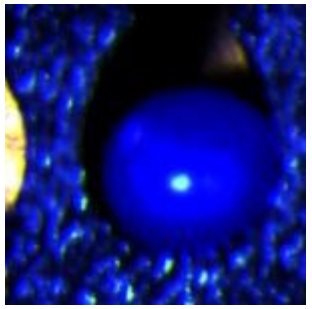


With the support of VLAIO

CTRL-Engineering use-case: Does LSDD work on images?

- Artificially Brighten and darken images = 2 concepts:
  - Dark images
  - Bright images
  - Switch 9 times between 2 concepts





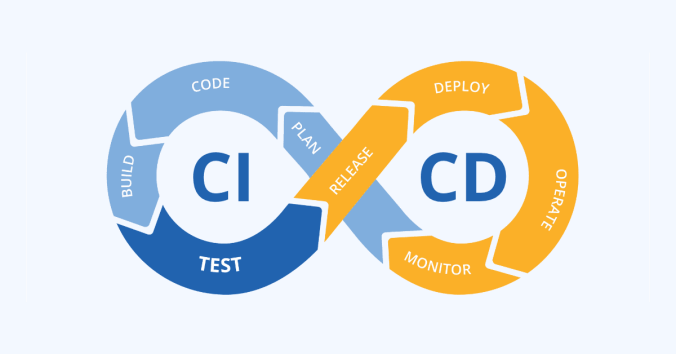
- MTD = 313.555, MDR = 0, FD = 0

With the support of VLAIO

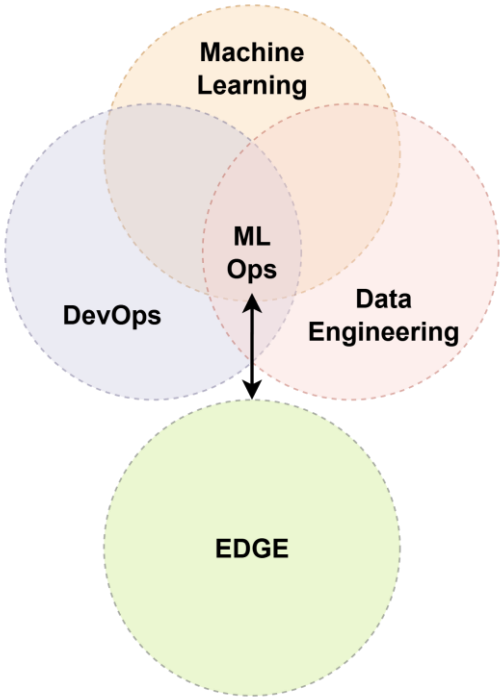
Training, testing and deploying AI models on the edge

Sille Van Landschoot

Edge gap



- MLOPS principles: creation of machine learning products
  - CI/CD automation, workflow orchestration and reproducibility;
  - versioning of data, model, and code;
  - collaboration;
  - continuous ML training and evaluation;
- Edge ML
  - Specialized hardware
  - Unique and custom approaches
  - Constrained resources



With the support of VLAIO

Edge friction

- Developing for the edge is hard
  - We want to use the edge device for **deployment** and inference
  - But we want to use our machine for **development** and training
  - Even better is to use my machine and offload compute to dedicated **servers**
- AI projects
  - **Large datasets** might not fit the edge device or local develoment machine
  - **Compute intensive** workloads for training
  - **Specialized hardware**: Training requirements != inference requirements

- One size doesn't fit all:
  Different stages of the ML lifecycle have different requirements.

With the support of VLAIO

Development vs. Production Mismatch

- **Resource Requirements:** demands powerful *computational resources* that are typically available in server environments.
- **Data Management Challenges:** Datasets, often exceed what can be *stored* on individual developer machines. This creates *bottlenecks*.
- **Training Infrastructure Needs:** AI training processes are computationally intensive and time-consuming, requiring dedicated infrastructure that can be scheduled and managed independently of development workstations.

With the support of VLAIO

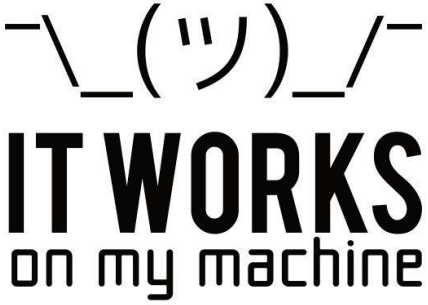
Operational Constraints

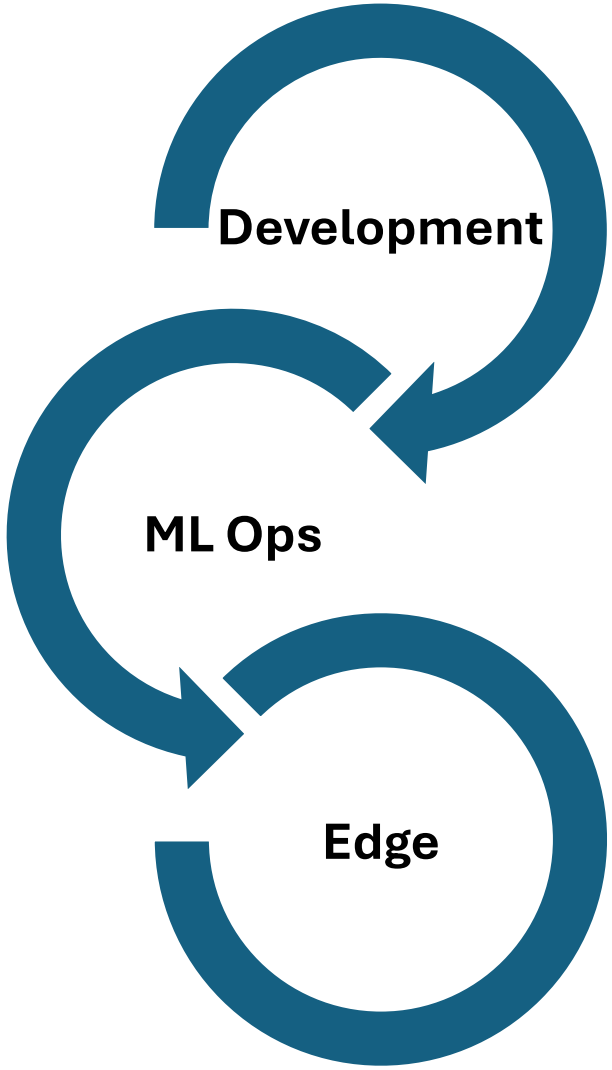
- **Production System Isolation:** Production edge devices must remain *dedicated to inference* operations and cannot be used for training or testing activities.
- **Development-Production Validation Gap:** Development teams need access to *edge-equivalent testing environments* to validate model performance and inference characteristics.
- **Model Engineering Flexibility:** The system must accommodate *arbitrary model architecture choices* and implementation decisions without constraining the development process.
- **Reproducibility and Tracking:** Dataset evolution and model versioning must be systematically tracked to ensure experiment *reproducibility*.

With the support of VLAIO

Bridging the gap(s)

- The core challenge is bridging the gap between
- **AI development:** requiring powerful servers, large datasets, and iterative experimentation
- **Production deployment:** requiring lightweight, real-time inference on resource-constrained edge devices

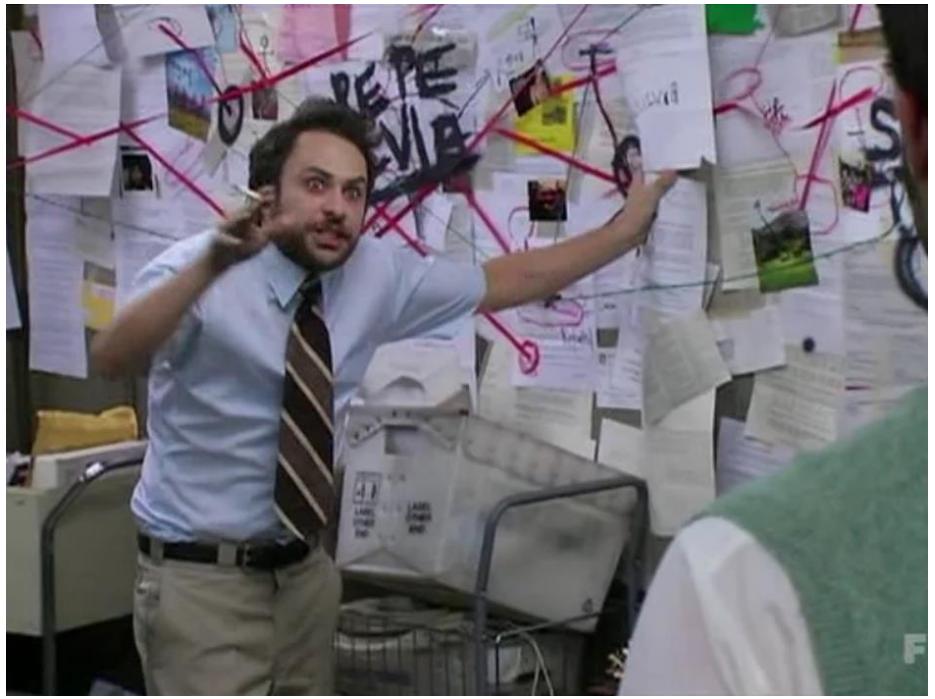




With the support of VLAIO

MLOps is bridging the gap

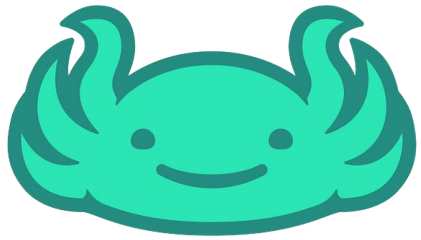
- **The MLOps solution**
  - **Capture & Curate:** Data and model versioning, reproducibility
  - **Automate**: Orchestrate workflows for training, testing and development
  - **Package & Ship**: Build reusable, self containing containers
  - **Observe & Learn**: Traceability by tracking progress and results
  - **Decide**: Maintain control over rollout and rollback



With the support of VLAIO

Bridging the gap

- Capture & Curate:
  - Git
  - LakeFS
  - Minio
- Automate
  - Prefect
- Package & Ship
  - Docker containers
  - Docker Registry



lakeFS

PREFECT

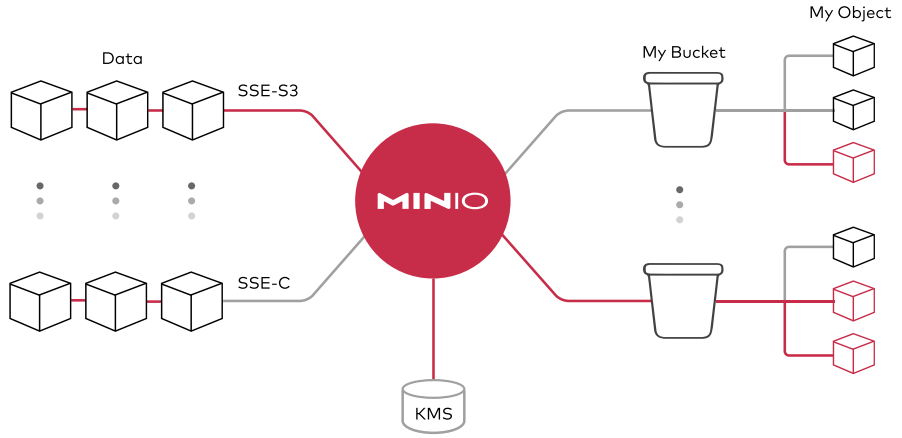
Docker Registry

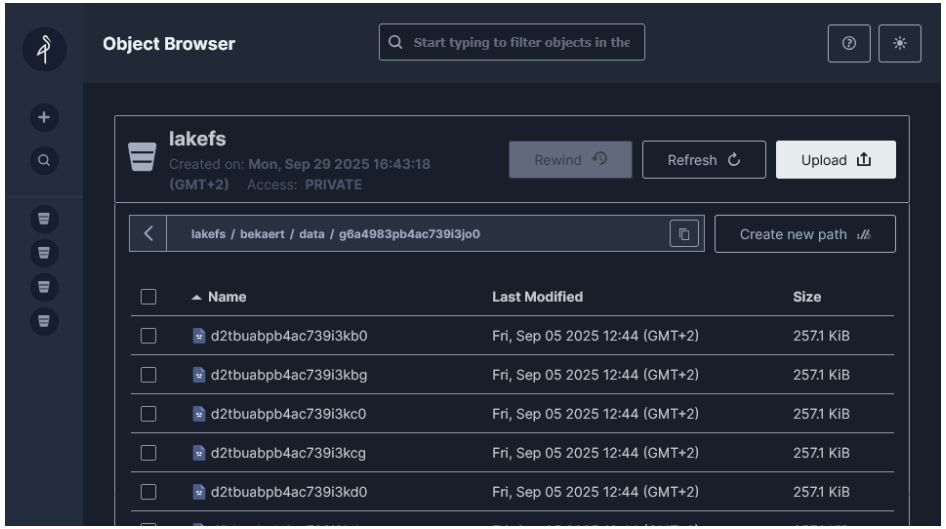
- **Open-source, self-hosted** solutions

With the support of VLAIO

Minio

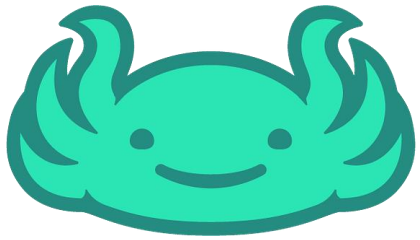
- S3-compatible **object storage** you run yourself—works with standard **S3** SDKs/CLI and pre-signed URLs.
- **Keep data/models on-prem/edge** (latency data sovereignty).
- One **bucketed store** for datasets, models, artifacts, telemetry.
- **Versioning** + immutability → reproducibility and audits.
- **Scales out**; cheap per-TB versus managed cloud.



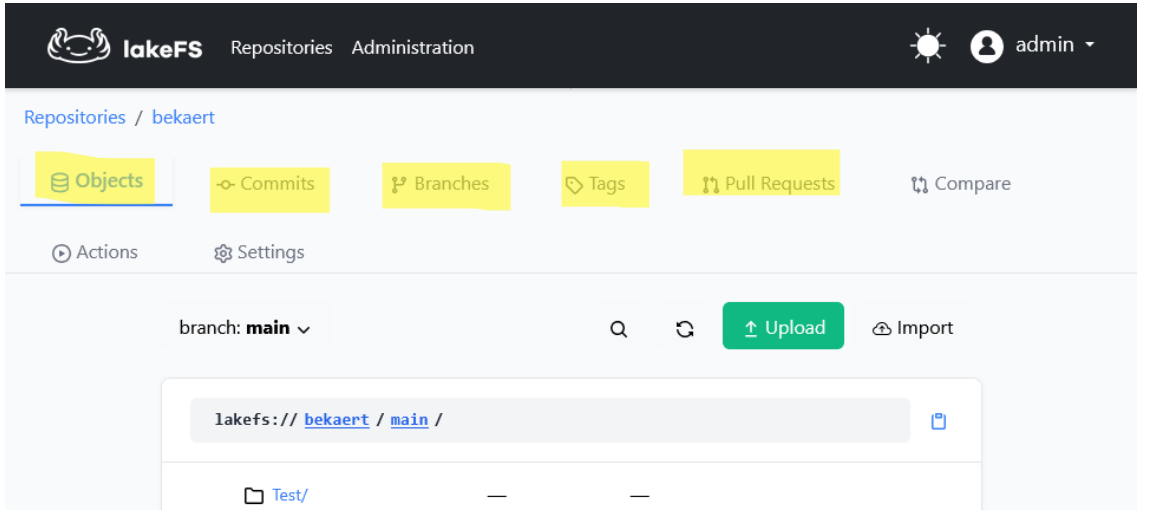


With the support of VLAIO

LakeFS



- Git-like **version control** for datasets
  - **zero-copy** branches/commits/merges
- **Remote-first** data management
- Features
  - **Reproducible training**: pin datasets/models by commit ID
  - **Isolated experiments**: branch fast, run pipelines, merge/promote when green
  - **Safe rollbacks**: revert a bad dataset/model instantly
  - **Auditability**: lineage from data → model → device release

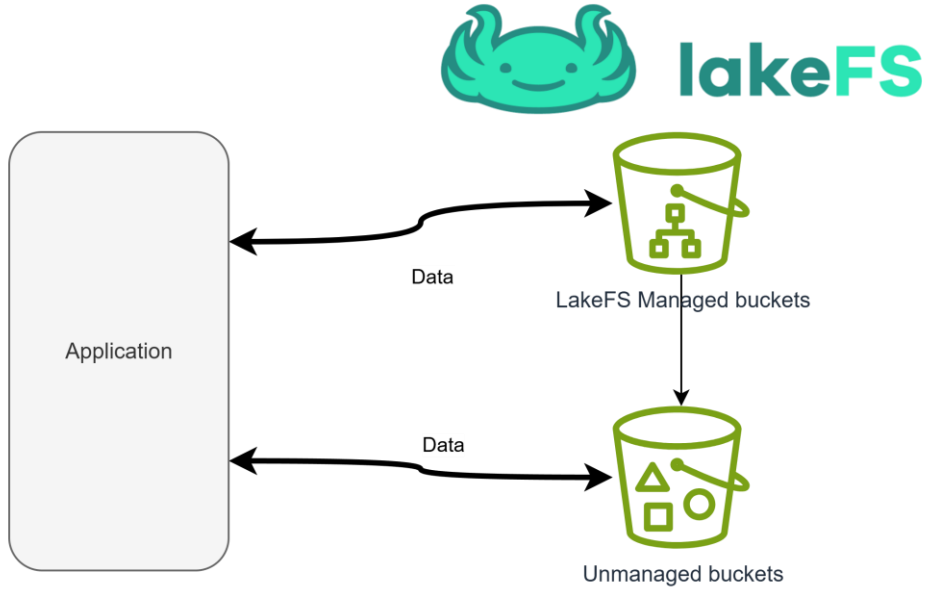



```bash
# Create a branch from main
lakefs branch create my-feature-branch --source main

# Checkout the branch (sets it as active context)
lakefs checkout my-feature-branch

# Upload new images to the LakeFS server
lakefs fs upload \
    --source ./new-images/ \
    --destination lakefs://my-dataset-repo/my-feature-branch/training/

# Commit changes
lakefs commit my-feature-branch --message "Added new training files"
```


With the support of VLAIO

Minio + LakeFS

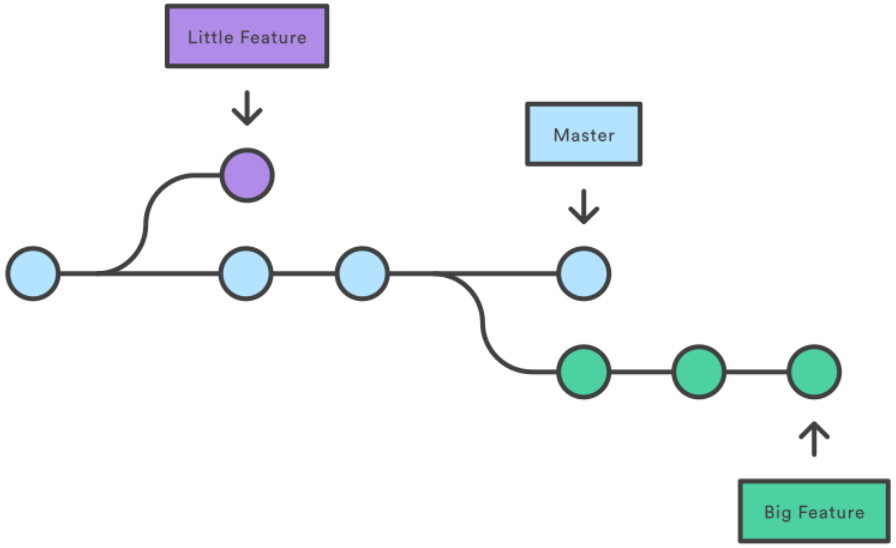
- Uses S3/**Minio object storage** as storage backend
- **Open Source,** Easy to **self-host**
- **Containerized** single command deployment using Docker, Docker Compose, Kubernetes,…
- **Lightweight**, perfect fit for development
- Access data over LakeFS or S3 API (open implementation)

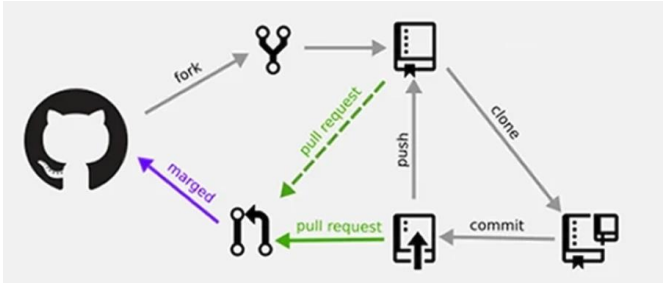


With the support of VLAIO

Git Version Control

- **Single source of truth for code**, configs, pipelines, infrastructure as code
- **GitOps triggers**: commit/PR → CI runs tests/build/train → gated deploy
- **Traceability**: tag releases; link commits to data/model versions & device fleets
- **Reproducibility**: pin everything by commit/tag (env, params, artifacts)
- **Rollbacks**: revert commit/tag → auto-rollback pipeline, fast recovery

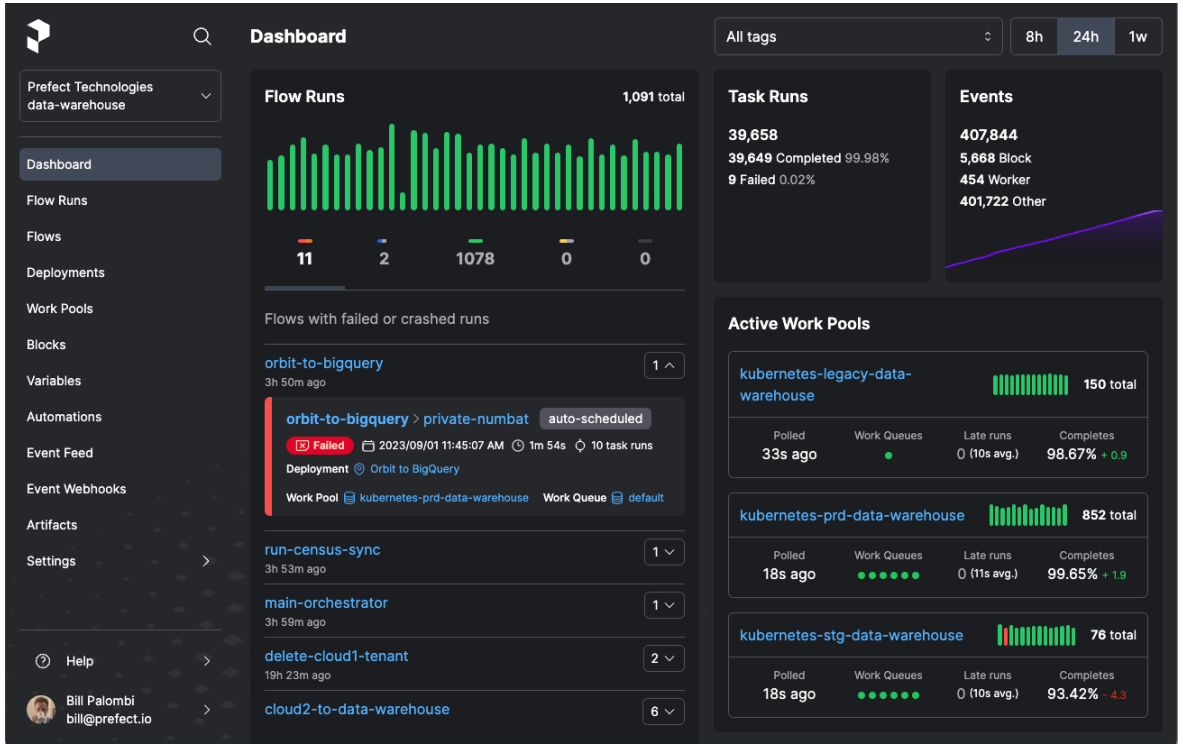




With the support of VLAIO

Prefect

- “Modern **workflow orchestration** for data and ML engineers”
  - Open source
  - Manage, schedule, and monitor data workflows
  - Flexibility and scalability for various automation tasks
  - Useful in data engineering, MLOps, and machine learning pipelines
- **Python-First** and Code-Driven
- **Integrations** with ML libraries, data sources and platforms



With the support of VLAIO

Prefect

- **Flow & Task Abstraction**
  - Organize tasks into flows to structure workflows easily
  - Task dependencies defined intuitively in Python
- **Resilience & Fault Tolerance**
  - Automatic retries, timeouts, and error handling
  - Options to resume, skip, or rollback tasks upon failure
- **Observability & Real-Time Monitoring**
  - Track task status, view logs, and monitor performance
  - Real-time alerts for task failures or anomalies

With the support of VLAIO

Prefect Tasks and Flows

- **Tasks**: Small, reusable functions representing individual steps in a workflow (e.g., data extraction, processing).
- **Flows**: Organized sequences of tasks forming a complete workflow or pipeline.
- **Modular Design**: Build, reuse, and manage tasks within flows for efficient development.
- **Dependency Management**: Control task order and relationships for accurate, predictable execution.

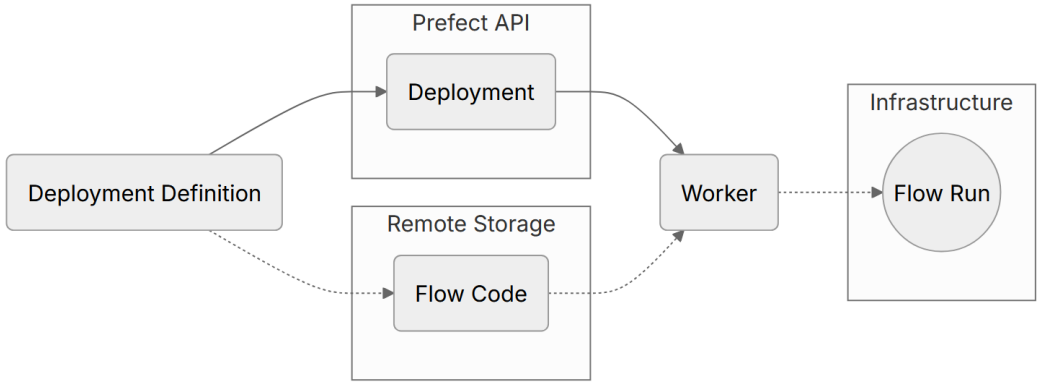
```python
@task
def mbed_compile(target, compiler) -> None:
    run_mbed_container(f"mbed compile -m {target} -t {compiler}")

@task
def prepare_model(tflite_file: str) -> None:
    model_file_name = "model.h"
    shell_run_command(command=f"xxd -i {tflite_file_name} > {model_file_name}")
    replace_text = tflite_file_name.replace('/', '_').replace('.', '_')
    shell_run_command(command=f"sed -i 's/'{replace_text}'/g_model/g' {model_file_name}")
```

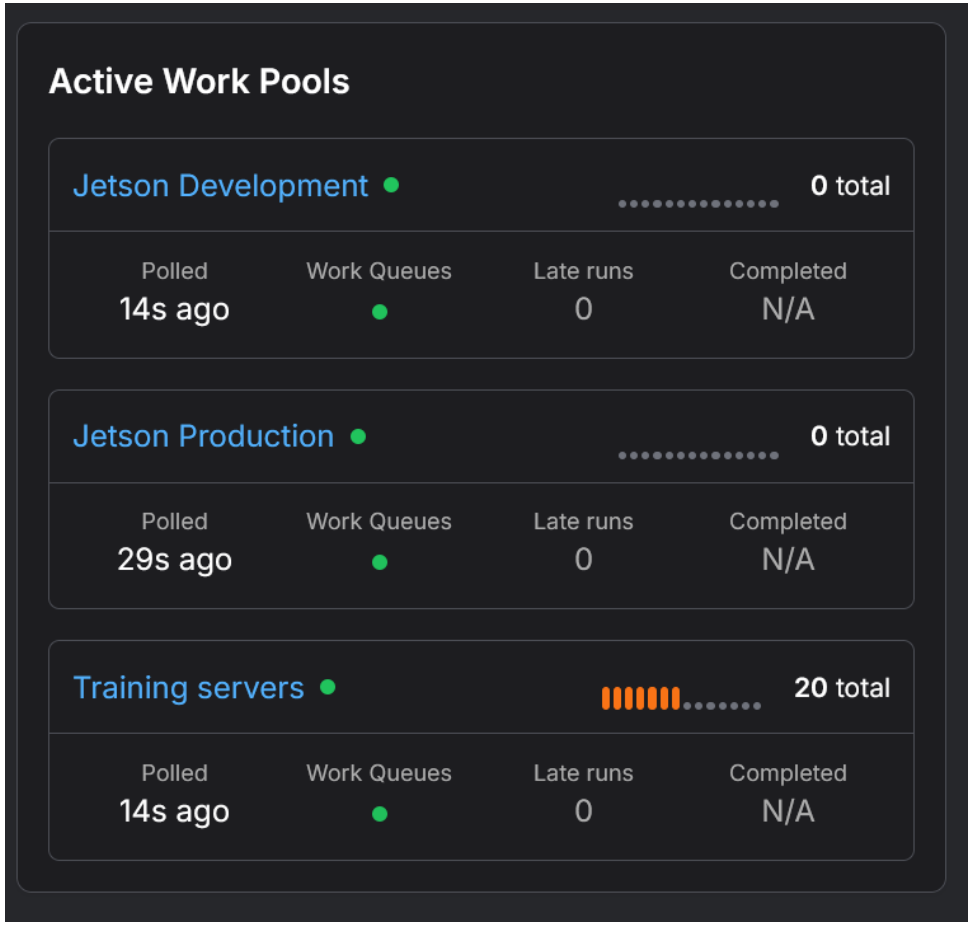
```python
@flow
def firmware_build(mbed_target: str, tflite_file = None) -> None:
    clone_firmware_repository(repo=firmware_repository)
    mbed_config_root()
    mbed_deploy()
    get_model_file(tflite_file)
    prepare_model(tflite_file)
    mbed_compile(target=mbed_target, compiler=mbed_compiler)
    save_firmware(s3_bucket_name=bucket_name, s3_folder=s3_folder, file_path=full_bin_path,
                  mbed_target=mbed_target)
    cleanup()
```


With the support of VLAIO

Prefect Work Pools

- **Dynamic infrastructure provisioning**
- Specialized pools for **specific workloads** with **priorities** and **concurrency**
- Different execution environments:
  - **Docker**
  - Process
  - Kubernetes
  - Azure, Google Cloud, AWS,…

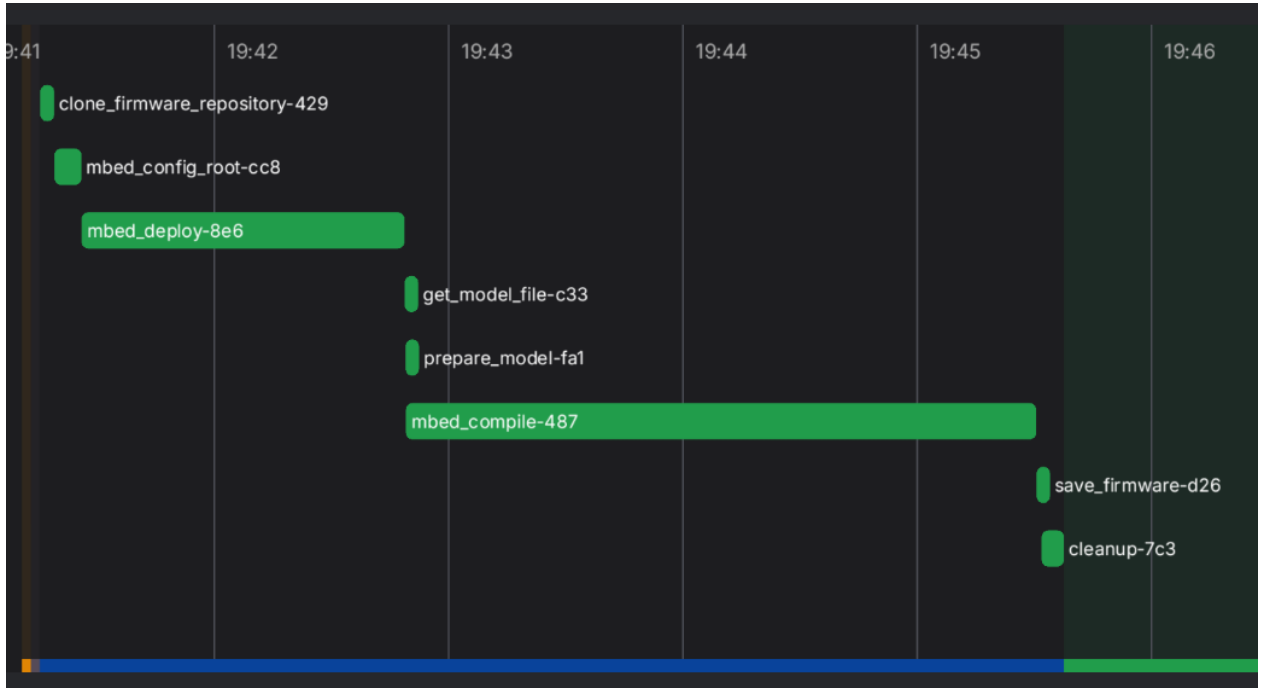


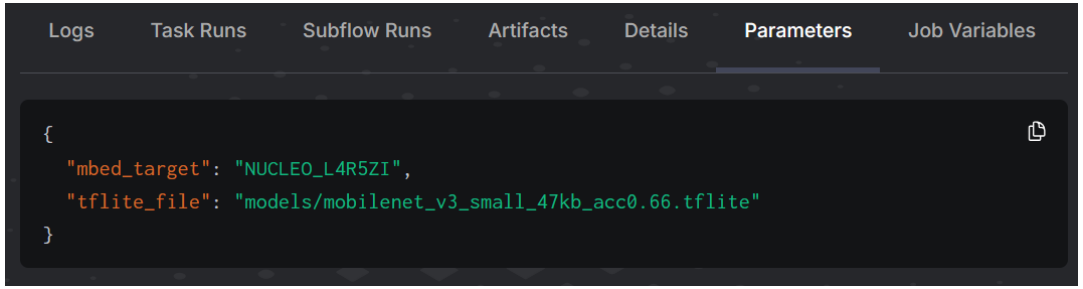


With the support of VLAIO

Prefect Runs

- Represents each **instance of a workflow execution**.
- Tracks **status**, **logs**, and **outputs** for real-time monitoring.
- Enables **issue identification** with success, failure, and retry info.





With the support of VLAIO

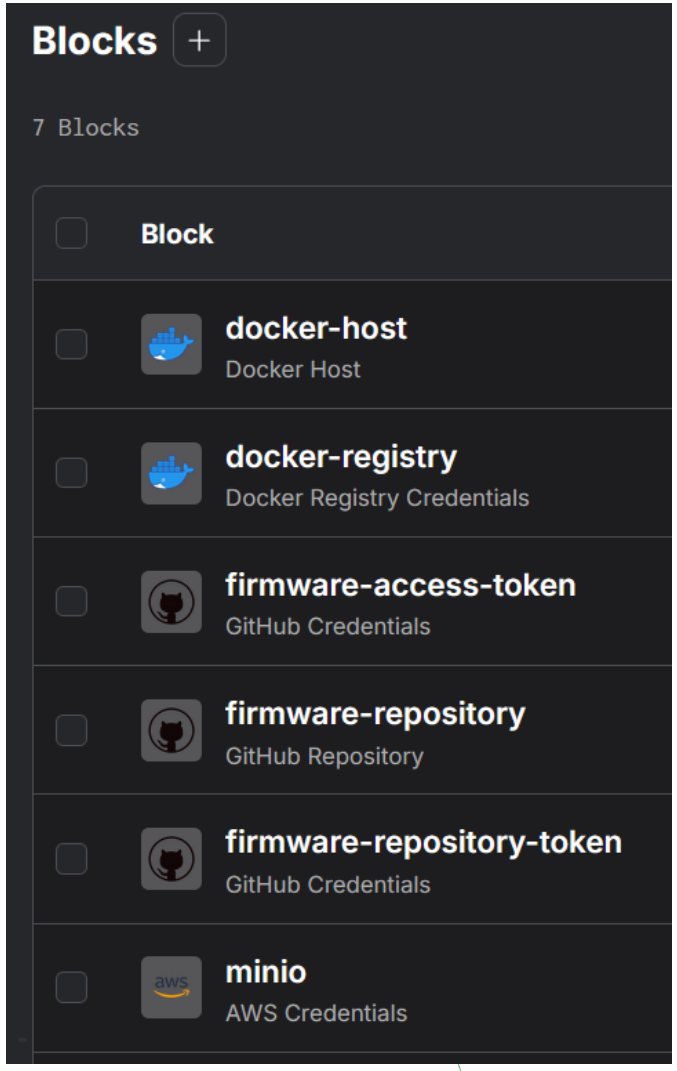
Prefect Variables

- ***Dynamic configuration elements*** used to streamline and adapt workflows.
- **Centralized Configuration**: Manage settings and constants across flows from a single place.
- **Environment-Aware**: Adjust workflow behavior based on environments (dev, staging, production).
- **Reusable & Secure**: Define once, reuse everywhere; securely stored to protect sensitive information.

With the support of VLAIO

Prefect Blocks

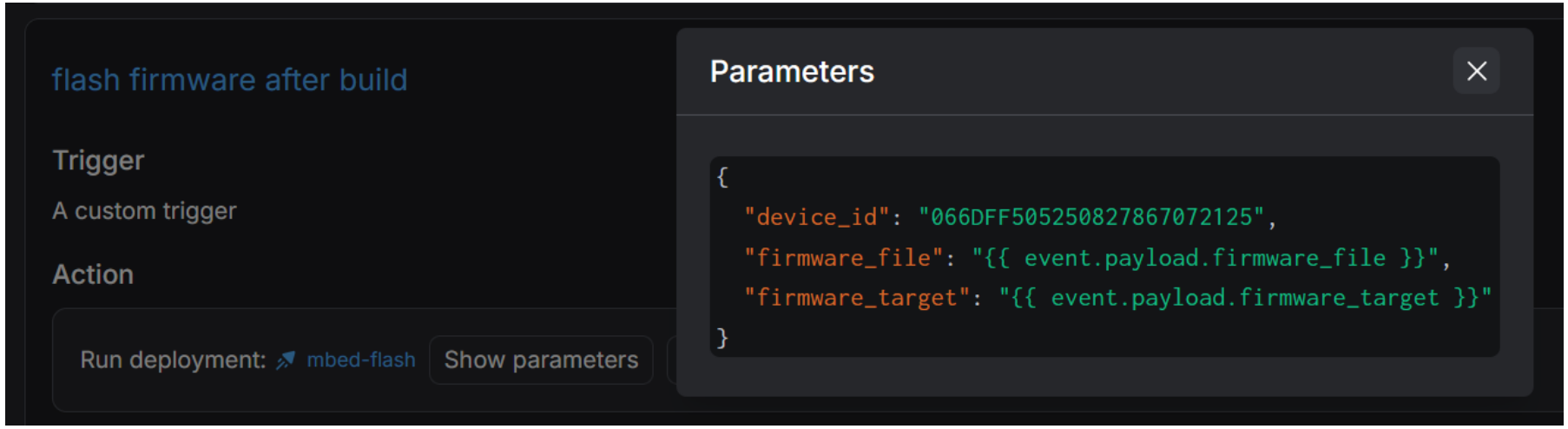
- ***Reusable, modular building blocks*** that simplify the development, deployment, and management of workflows.
- **Pre-built integrations**: Connect to popular tools (databases, cloud providers, messaging services) without custom code.
- **Customizable**: Create and share custom blocks tailored to unique needs.
- **Easy configuration**: Define and reuse configurations across tasks to reduce setup time.



With the support of VLAIO

Prefect Automations

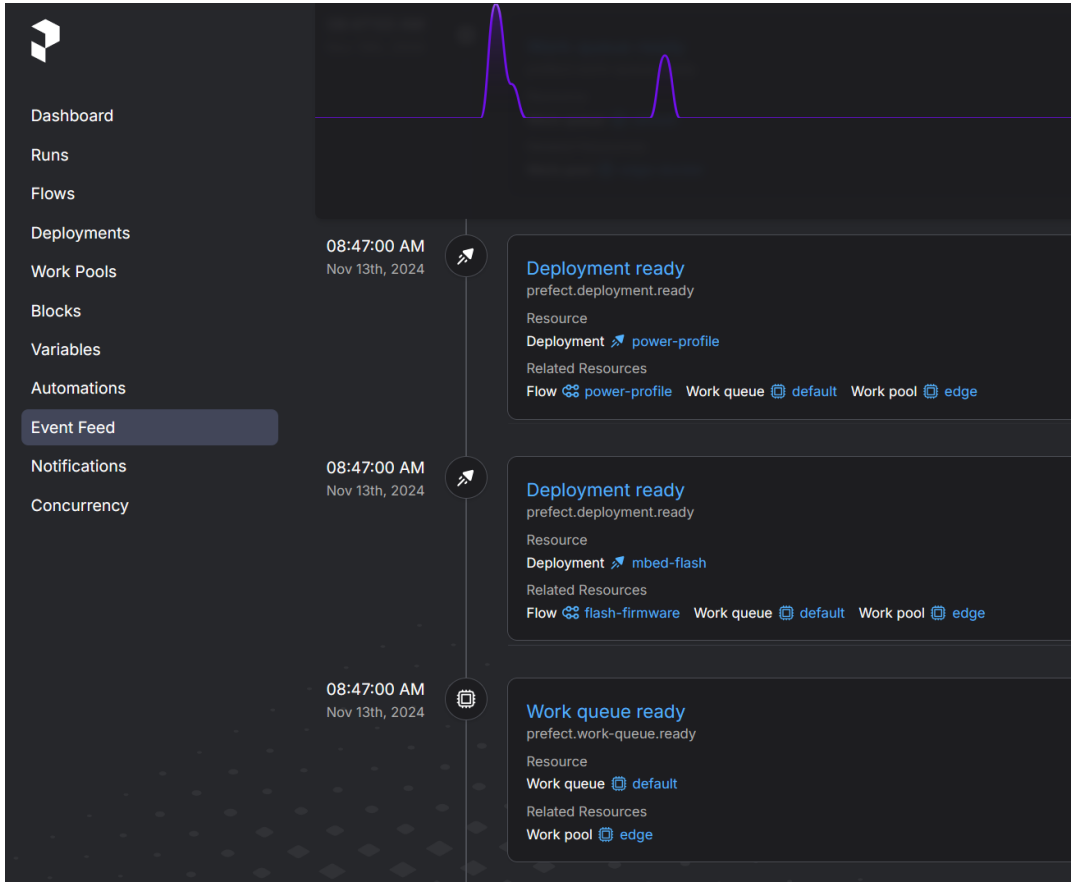
- **Automated Actions**: Trigger responses based on events (e.g., failures, successes).
- No-Code Setup: Easily configure alerts, retries, or escalations without custom code.
- **Improves Reliability**: Reduces manual intervention and ensures timely responses.
- **Customizable**: Tailor actions to specific workflows and business needs.



With the support of VLAIO

Prefect Event feed & Notifications

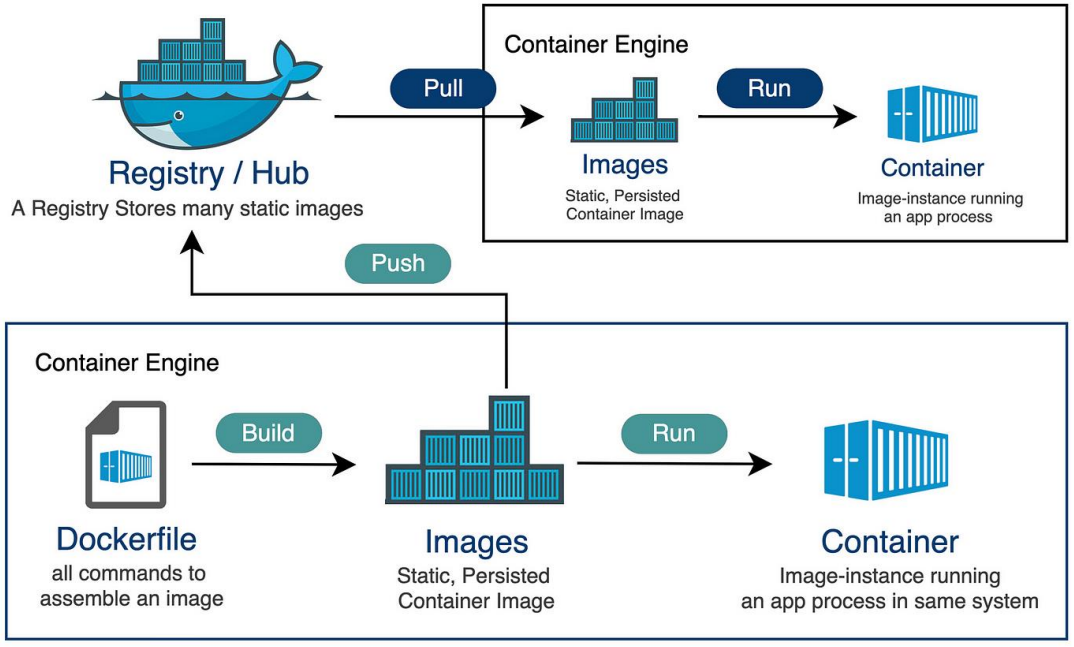
- **Real-time updates** on workflow events (starts, completions, failures).
- **Centralized view** for monitoring all activity.
- Quick **identification of issues**.
- **Alerts** for key events (failures, retries) via email, Slack, etc.
- Customizable **triggers** to focus on relevant updates.
- Keeps teams **informed** for proactive responses.



With the support of VLAIO

Docker Image management

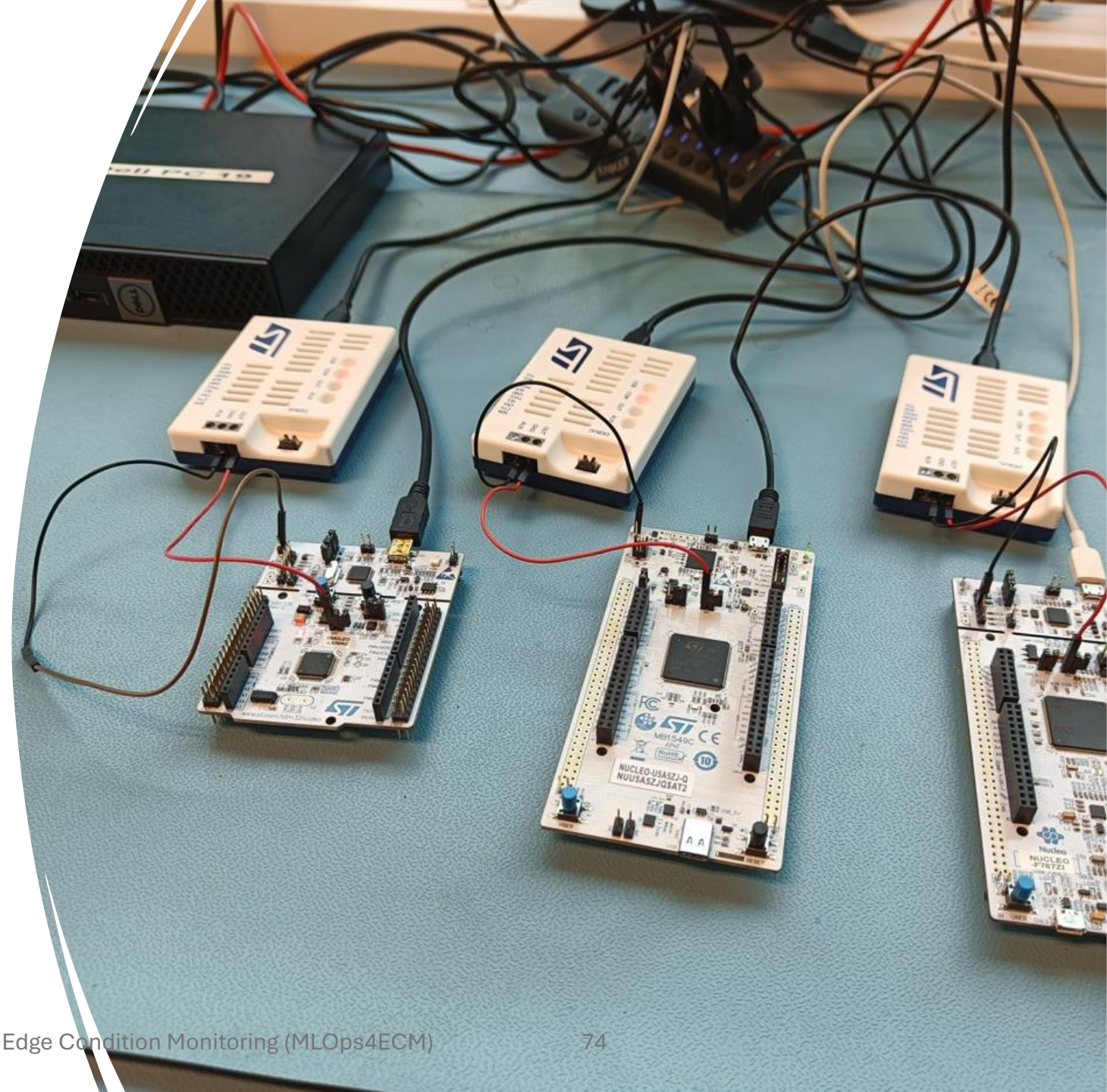
- **Docker Registry** for the management of Docker Images
- Efficient **(re)use of Docker images**, regardless of whether they need to be executed on the server or edge
- **Centralized** registry for easy management
- Easy **local** deployment for on-prem **data sovereignty** and **low-latency** pulls



With the support of VLAIO

Two use-cases

- Testing and validating 900+ AI models on a microcontroller system with energy consumption monitoring
- Pipeline for training, testing and deploying AI models on the Edge

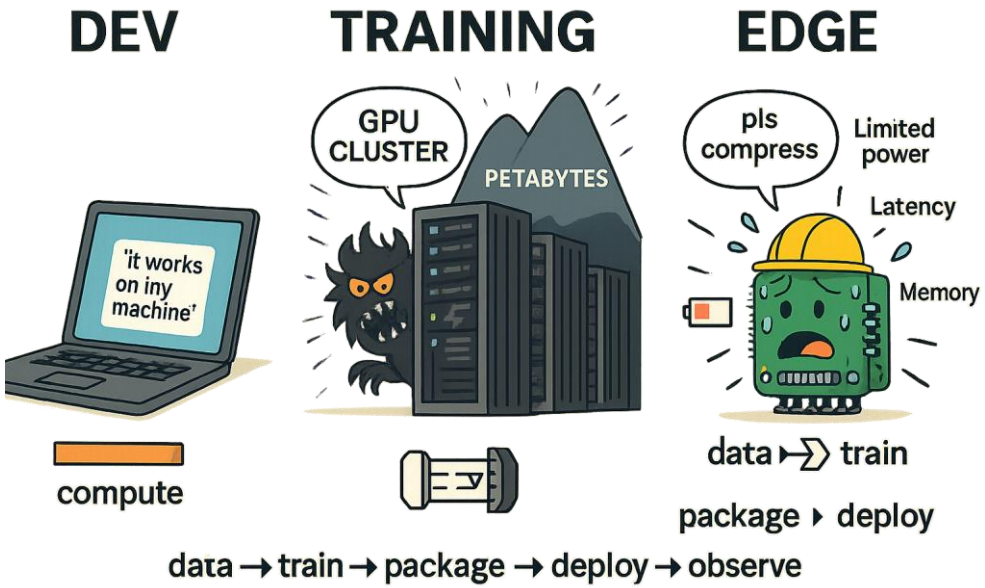


Pipeline for training, testing and deploying AI models on the Edge

With the support of VLAIO

Machine vision on Jetson Orin Nano

- **Large image dataset**
  - Hard to manage and maintain
- Training on edge device is **impractical**
  - Best representation of **production environment**
  - Does not compare to a **development environment**
  - **Not optimized for training**, only inference
  - Constrained resources
    - Storage
    - Memory
    - CPU
    - Bandwidth



With the support of VLAIO

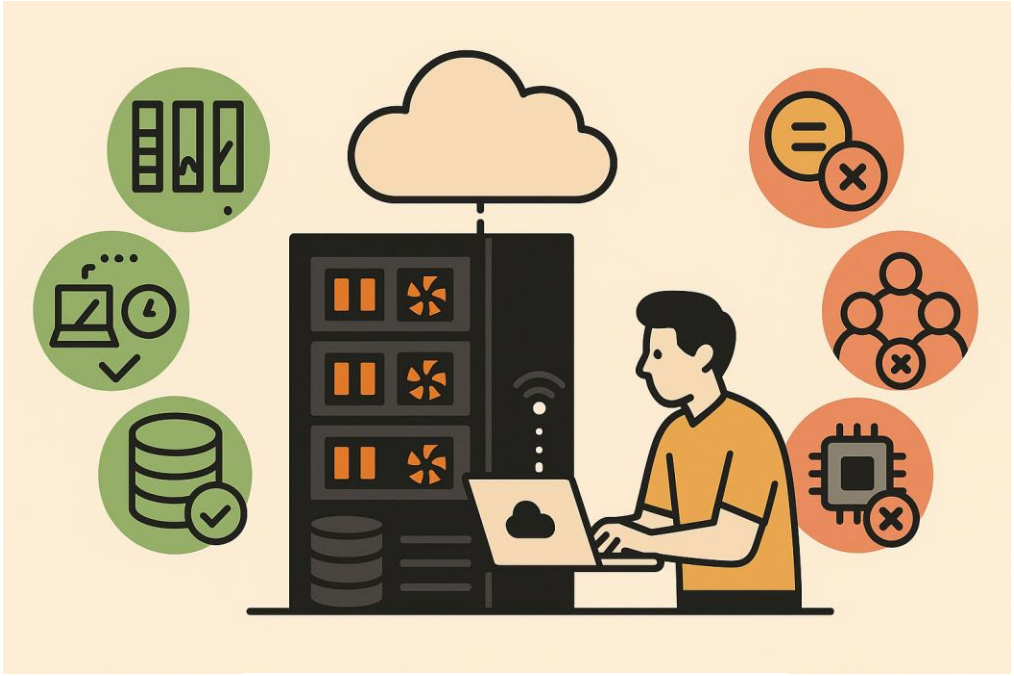
Development on local machine

- ✅ Pros
  - Allows for fast development cycles and instant feedback
  - Use subsamples of the dataset
  - Offline work
  - Full control of OS, tools and drivers
- ❌ Cons
  - Laptops with limited resources: CPU, GPU, RAM, Storage,…
  - Does not represent the production/edge device
  - Does not scale

With the support of VLAIO

Development on a server

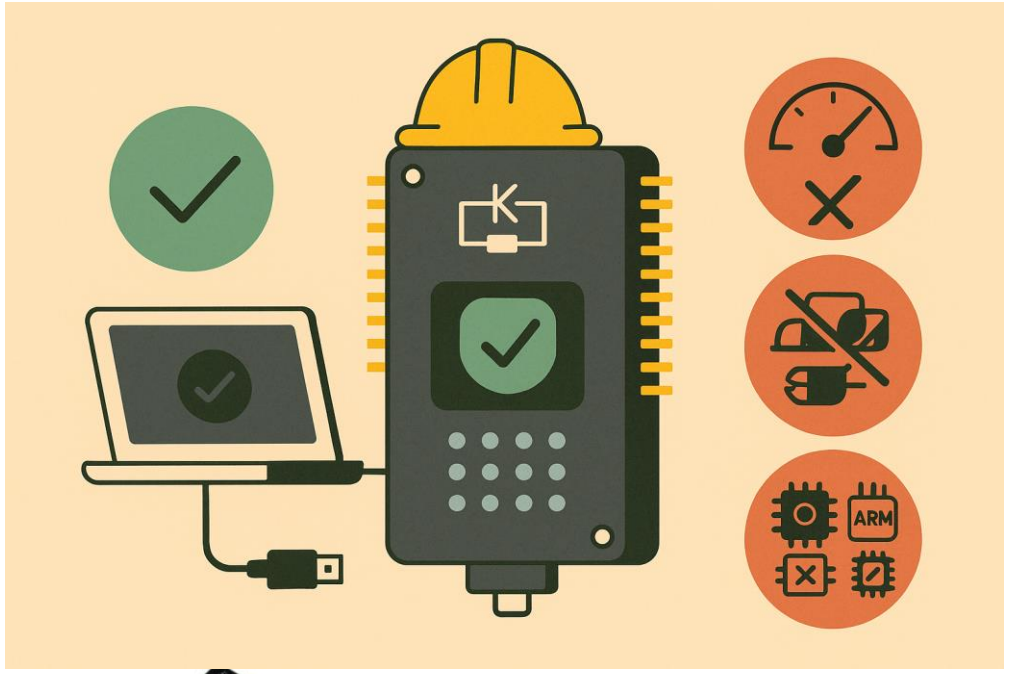
- ✅ Pros
  - Virtually limitless in resources: CPU, GPU, RAM, Storage
  - Remote development, anywhere and anytime
  - Long running detached jobs
  - Data proximity for large dataset
- ❌ Cons
  - Cost
  - Shared resources among multiple developers/projects
  - Still does not represent the edge system



With the support of VLAIO

Development on the edge

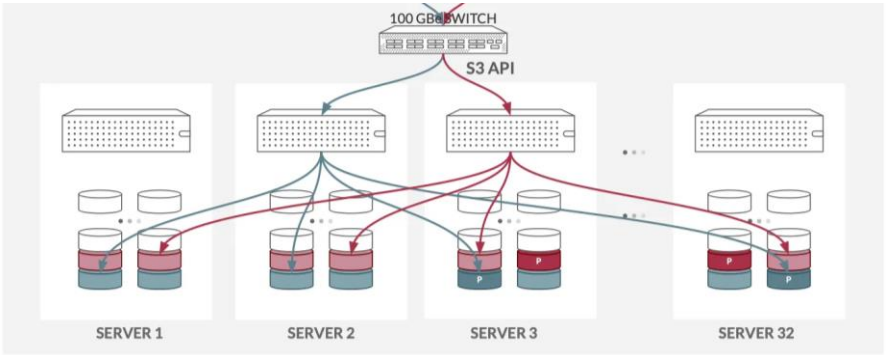
- ✅ Pros
  - Represents the deployment environment
- ❌ Cons
  - Very limited resources, No GPU,
  - Ineffective for training
  - No (full) support for training frameworks, tools and libraries
  - Often a different architecture

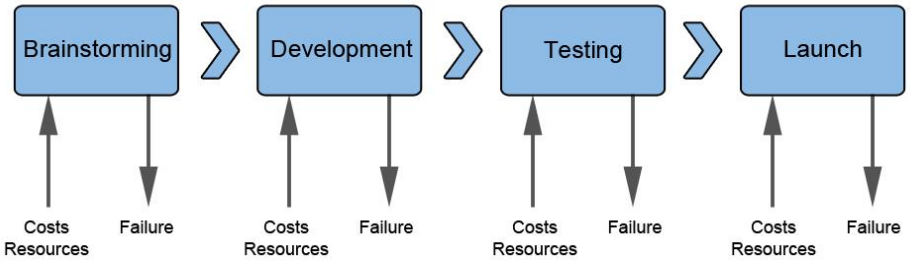


With the support of VLAIO

Development requirements

- **Scalable data storage** and **versioning** for large image datasets
- **GPU-accelerated training** infrastructure separate from developer machines
- **Automated pipeline orchestration** for training workflows
- **Model validation and testing** before deployment
- **Reproducible experiments** with dataset change tracking

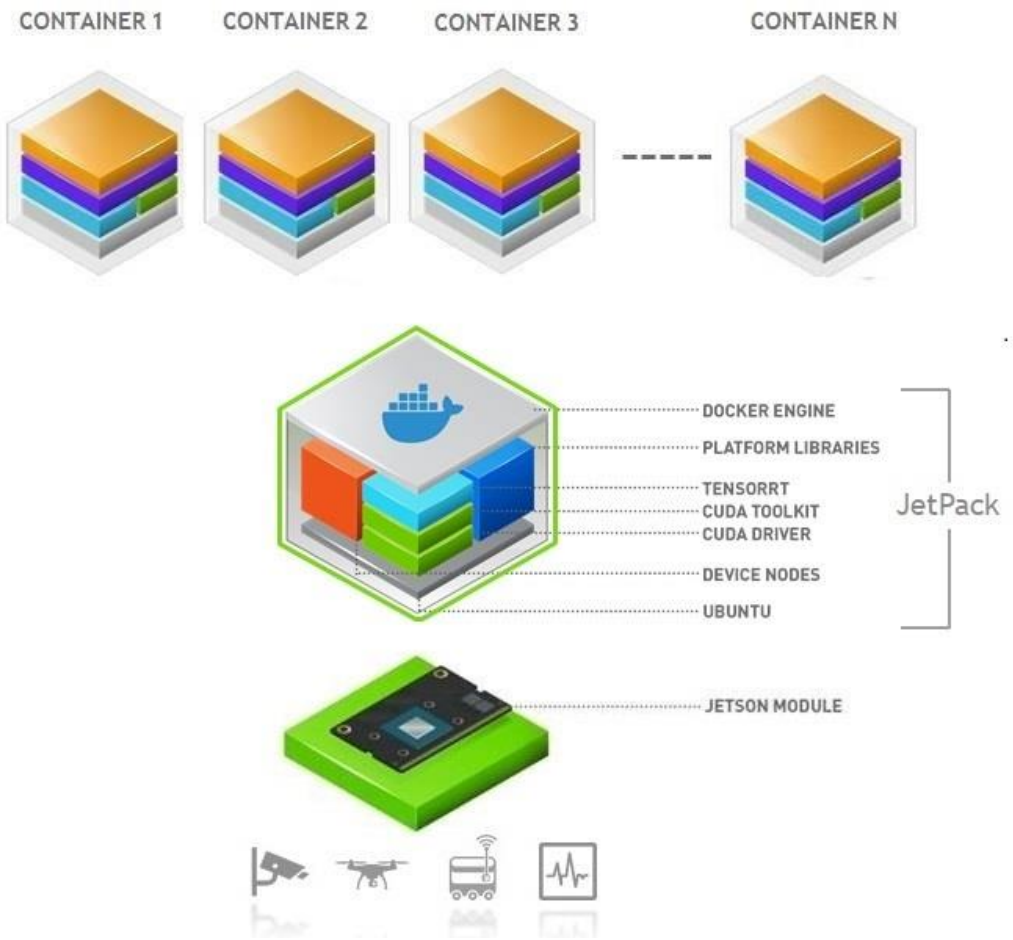




With the support of VLAIO

Production requirements

- **Real-time inference** on NVIDIA Jetson edge devices
- **Containerized deployment** for consistent environments
- **Automated model updates** from development to edge
- **Monitoring and feedback** loops for continuous improvement
- **Camera abstraction** support via standardized protocols (RTSP or gstreamer)



With the support of VLAIO

Integration Challenges

- **Resource optimization** (server GPUs vs. edge efficiency)
- Seamless pipeline from **development** to **production**
- Version control for both **code** and **data**
- Automated **testing** and **validation** gates
- DevOps **tooling** for ML lifecycle management

With the support of VLAIO

Infrastructure

- **MinIO**: Object storage for models and artifacts
- **Prefect**: Workflow orchestration and scheduling
- **Docker Registry**: Container image distribution
- **MediaMTX**: RTSP streaming server for testing
- **Grafana**: Real-time monitoring and alerting

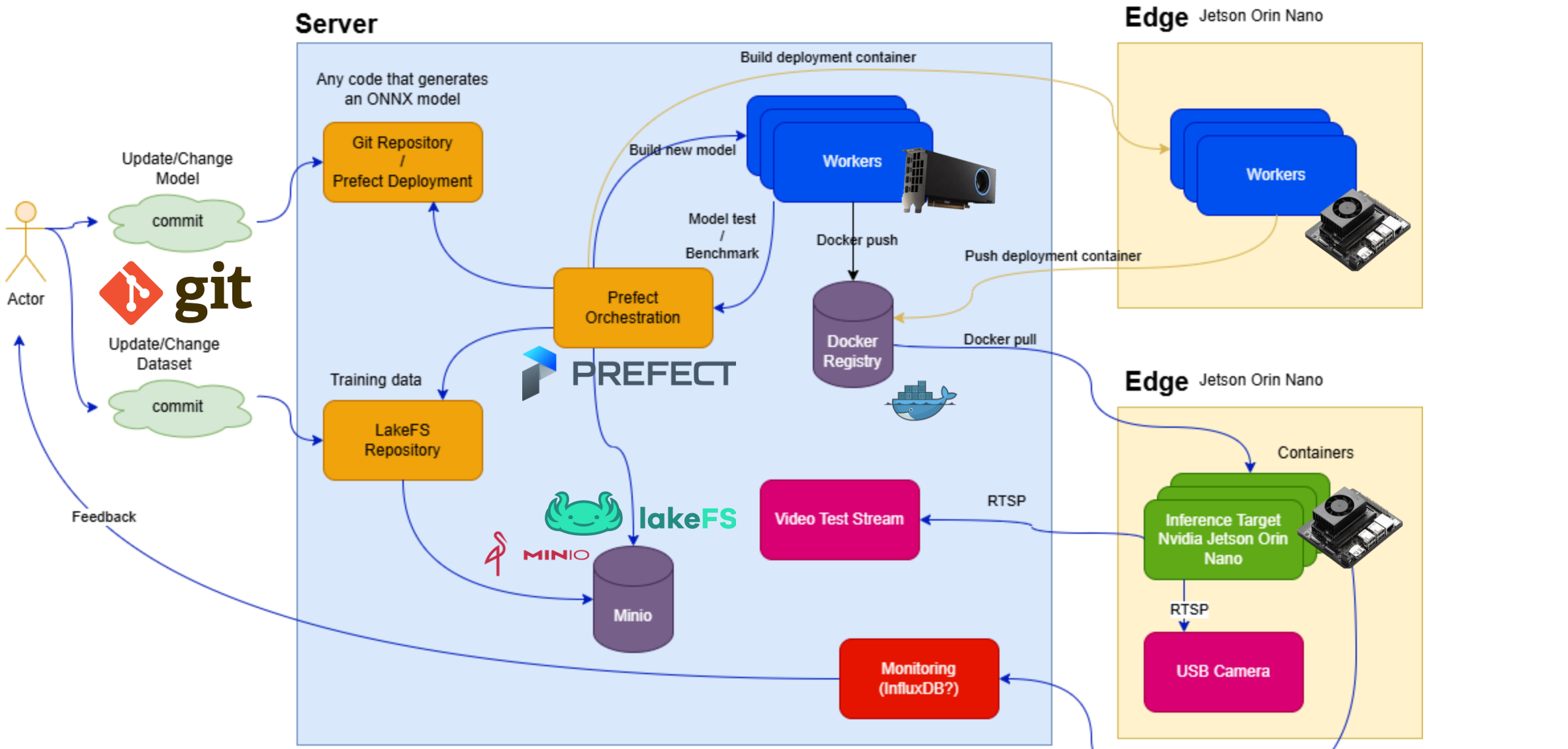
- Infrastructure should support the process of development **and** deployment

With the support of VLAIO

MLOps Workflow

1. **Code Commit** → Git repository triggers pipeline
2. **Data Versioning** → LakeFS manages dataset branches
3. **Training Execution** → Prefect orchestrates containerized training
4. **Model Validation** → Automated testing before deployment
5. **Container Building** → Docker images with trained models
6. **Edge Deployment** → Jetson devices pull and run new containers
7. **Monitoring Loop** → Performance feedback for continuous improvement

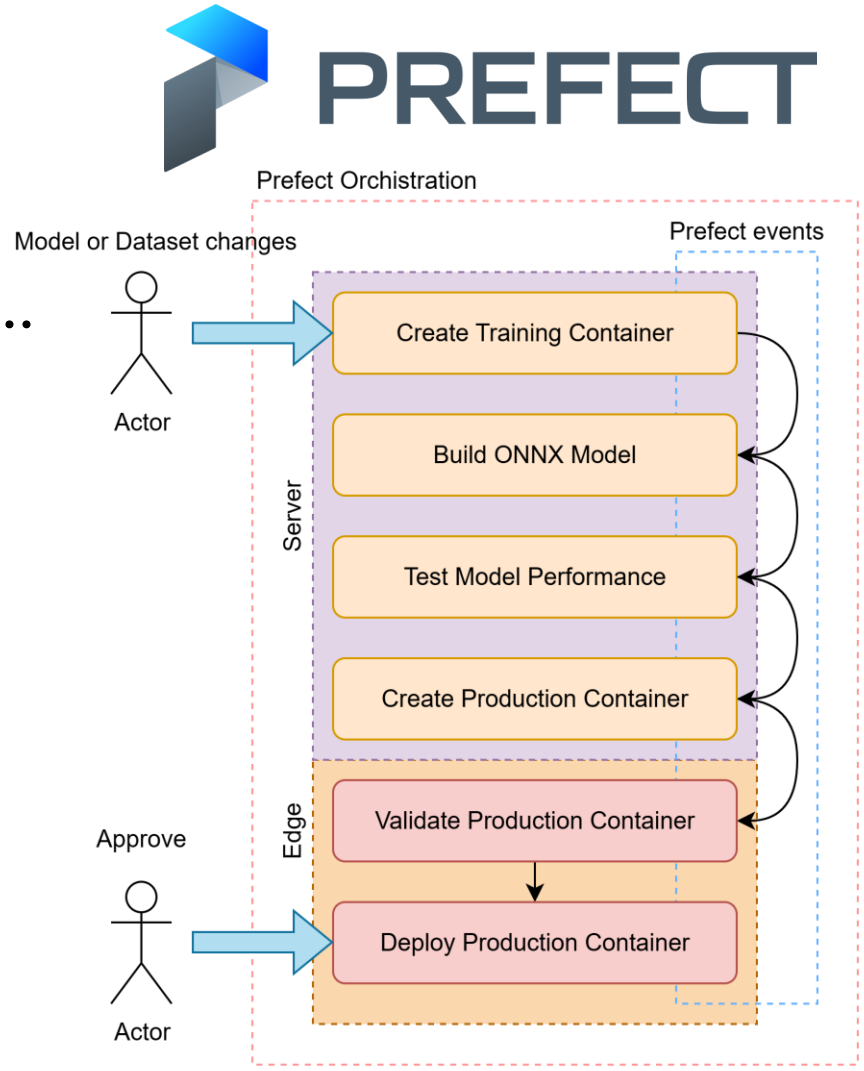
With the support of VLAIO



With the support of VLAIO

MLOps Workflow

- Developer produces events
  - Commit code, Update Model, Change dataset, …
- Prefect **Deployments** run **flows** and **task**
- Tasks generate **events** to trigger other deployments
- Flows are assigned to different **work-pools**
- Wen everything is fine, developer triggers deployment



With the support of VLAIO

What Worked Well

- **Architectural Decisions:**
  - - **Clean Separation**: Keeping ML core logic independent from orchestration proved invaluable for development speed and testing
  - - **Container Strategy**: Docker containers provide consistency across development, testing, and production environments
  - - **Standard protocols**: RTSP for cameras and ONNX for models ensure compatibility and interoperability

- **Development Experience:**
  - - **Dry Run Mode**: Game-changer for rapid iteration without waiting for full training cycles
  - - **Data Versioning**: LakeFS integration provides crucial experiment reproducibility
  - - **Automated Testing**: Built-in performance evaluation catches issues before deployment

Key Lessons Learned

- **MLOps Pipeline Design:**
  - - **Infrastructure Independence**: Core ML functions should work standalone for faster development
  - - **Testing Strategy**: Multiple levels of testing (structural, performance, integration) catch different types of issues
  - - **Monitoring is Critical**: Real-time feedback from edge devices informs model improvements
  - - **Version Everything**: Not just *code*, but *data*, *models*, and *container images*

- **Edge AI Deployment:**
  - - **Resource Optimization**: ONNX format and proper model optimization crucial for Jetson performance
  - - **Update Mechanisms**: Container-based deployment enables reliable model updates in production
  - - **Camera Integration**: RTSP provides standardized interface across different industrial camera systems

With the support of VLAIO

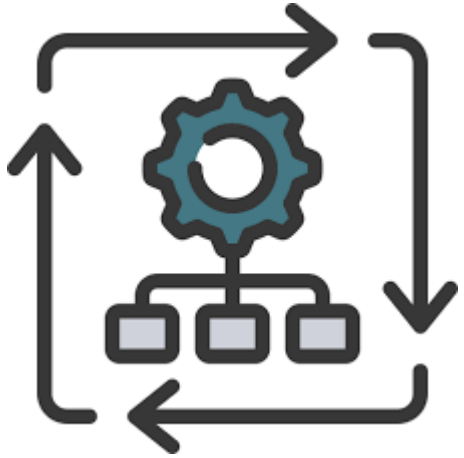
Conclusion

- This project successfully demonstrates a **complete MLOps pipeline** for industrial AI deployment, bridging the gap between research/development and production edge inference.

- This MLOps pipeline provides a **reusable template** for deploying AI models in industrial edge environments, combining the flexibility needed for AI development with the reliability required for production systems.

With the support of VLAIO

Auto deployment & energy monitoring



Sille Van Landschoot

With the support of VLAIO

Auto deployment & energy monitoring

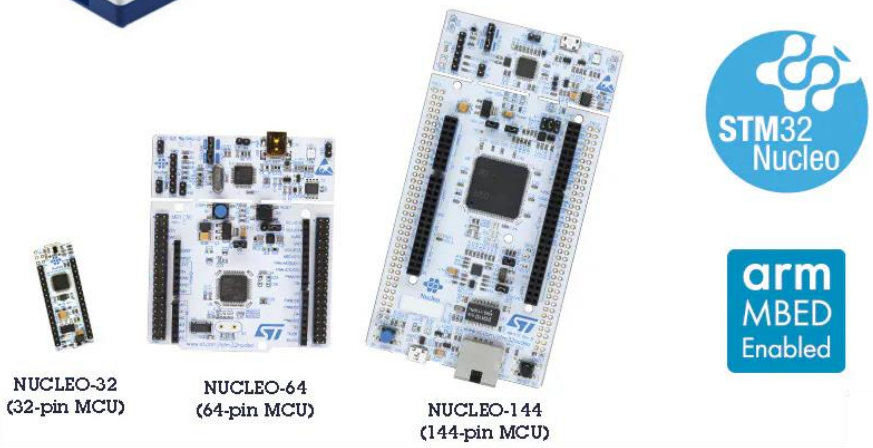
- **Energy monitoring of AI models on embedded devices**
  - 150+ models
  - 6+ targets
- Multiple firmware implementations and revisions
- New targets and new models at any time

  #tests = #models x #targets x #version
- Solution: Automate!
  - How? **Prefect**?

With the support of VLAIO

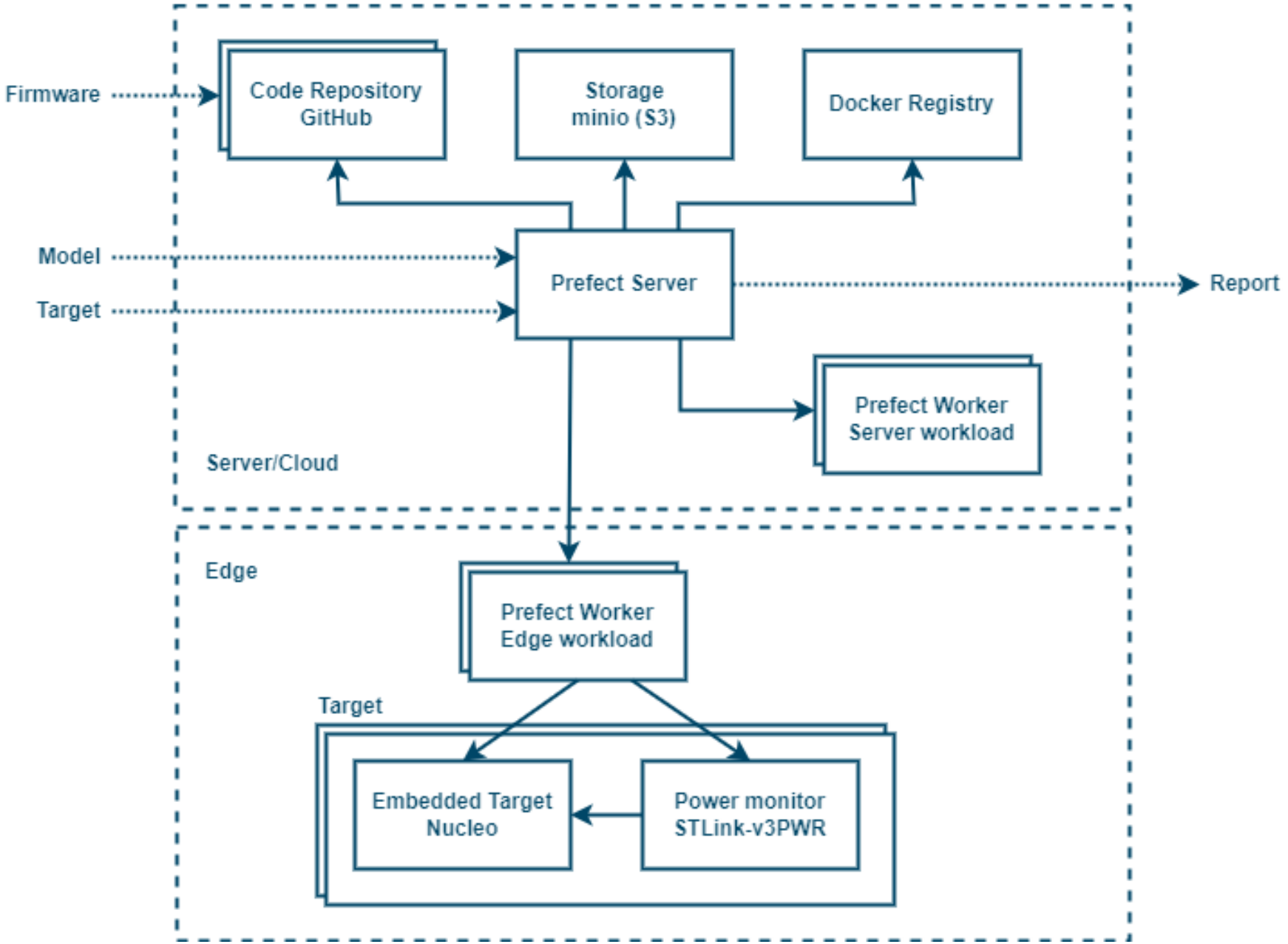
Auto deployment & energy monitoring

- **Version control**: Git(Hub)
- **Model storage**: Minio, S3 compatible
- **Docker image repository**: Docker Registry
- **Task orchestration**: Prefect server
- **Edge nodes**: Prefect workers
- **Targets**: STM32 Nucleo development boards
- **Energy monitoring**: STLink-v3PWR



Get target → Build firmware → Get model → Deploy to target → Monitor energy → Make report

With the support of VLAIO



End-user

- **Blackbox**

Inputs

- Firmware
- Model
- Target

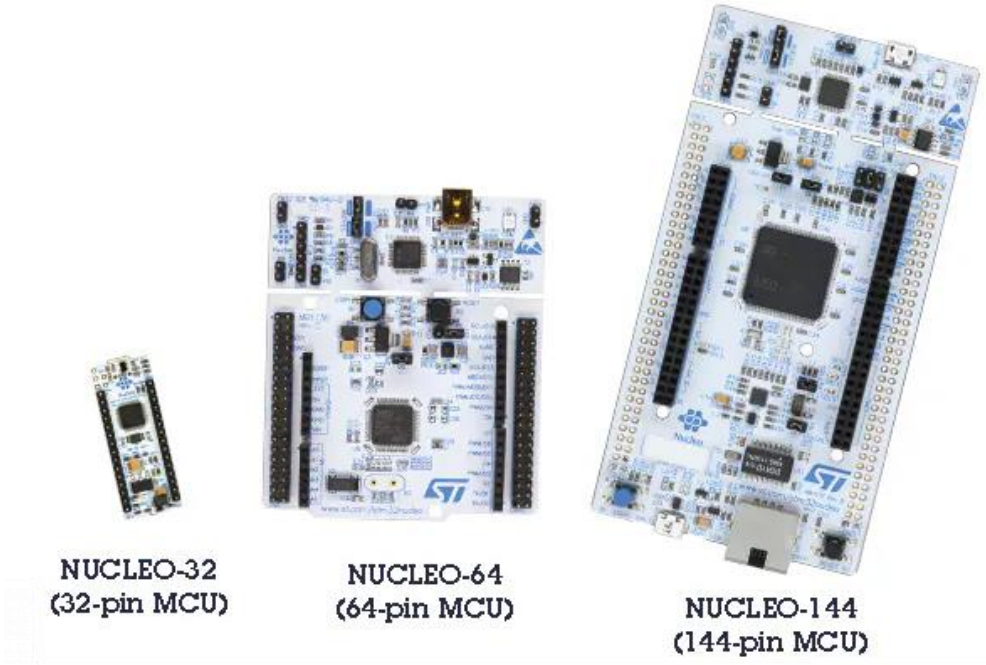
Output

- Power report

With the support of VLAIO

Targets

- Nucleo target development boards
  - **Microcontrollers**
  - Microcontroller architectures Cortex-M
  - Memory **constraints**
    - RAM
    - Flash
  - **Peripherals**
- USB Connections to edge computer
  - UART comunication



With the support of VLAIO

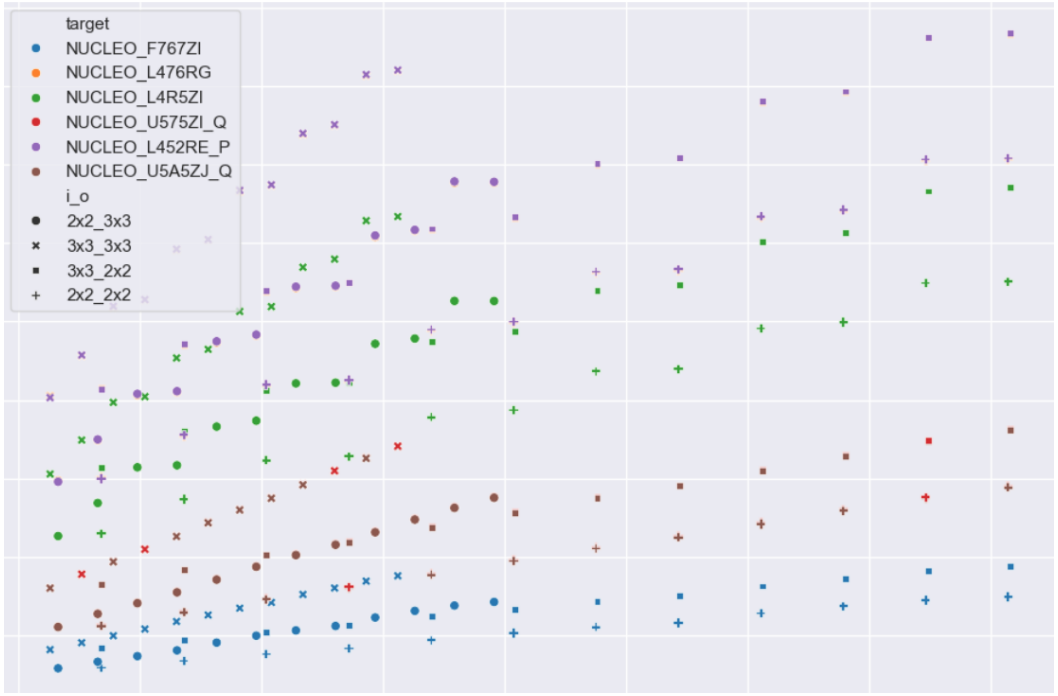
Power monitor

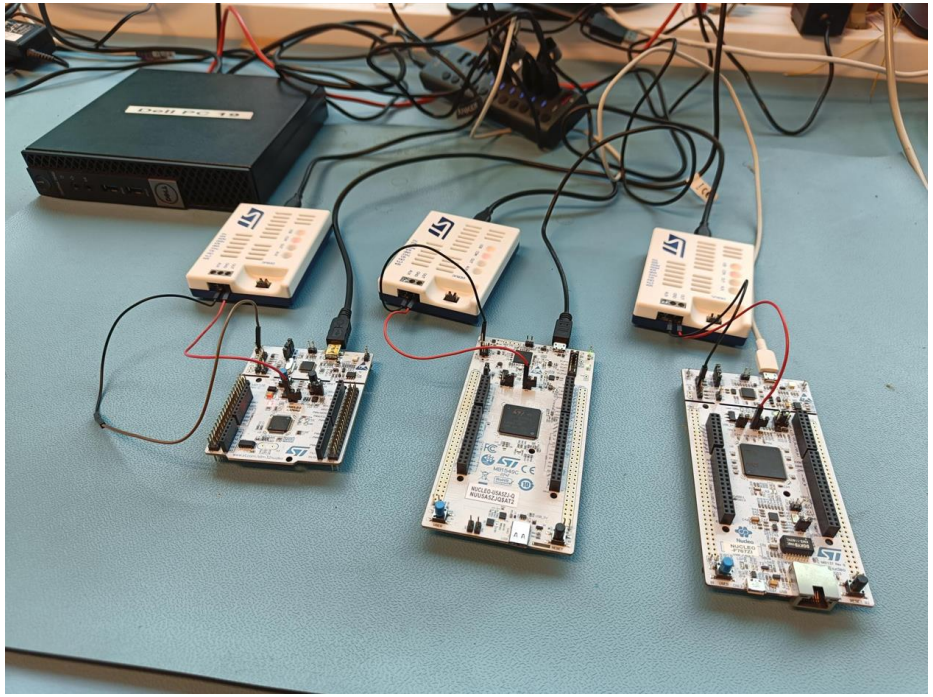
- STLink v3PWR
- Managed over UART
  - **Python script**
  - Start/stop measurements
  - Set power and voltages
  - Reset targets
  - UART for target commands

With the support of VLAIO

Reports and logs

- Plain old **Python scripts or Jupyter Notebook** with matplotlib, pandas,... decorated with Prefect Tasks and Flows





With the support of VLAIO

Lessons learned

- MLOps infrastructure **requires some upfront effort** but pays of in the long run
- Prefect provides **workflow orchestration** with **minimal overhead**
  - Simply add some **annotations to your Python code**
- **Decoupling video streams** from physical camera's allow for flexibility during development
  - Share camera streams across different systems and stages in the development cycle
- Implement 'dry-run' options to speed up MLOps development
- **Open-source solutions** exist to support the ML Ops workflows
- Combining **lightweight tools** can **support development AND production processes**

With the support of VLAIO

Conclusion

- Automation using Prefect is very **flexible and easy for AI workflows**
- Minio and LakeFS allow for **large and remote datasets** with **version control**
- Containers allow for **flexible execution** across **different machines and targets**
- Version control of code and models provides **traceability and reproducibility** and **rollbacks**

Machine vision pipeline for efficient edge deployment

Alexander D’hoore

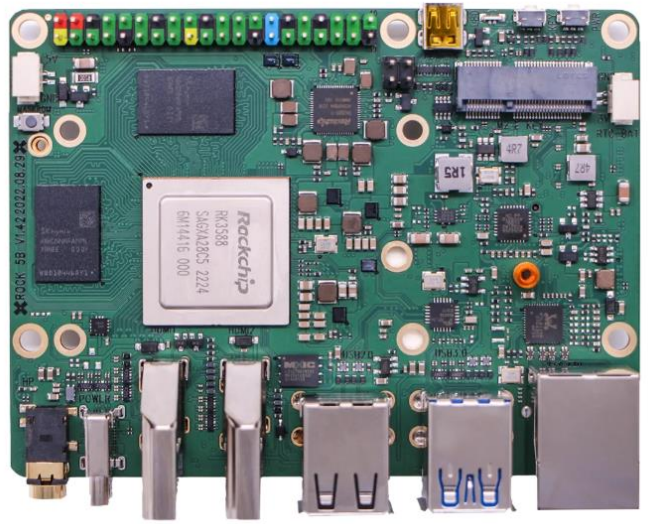
Use-case: Leap Technologies

- **Goal**: Deploy an occlusion-aware **instance segmentation** model on an edge device. **Low-cost hardware** is preferred. What are the minimum hardware resources required?

- **Tools used**: PyTorch, Albumentations, **Ray Data**, Ray Tune, MinIO (S3), TensorBoard, Proxmox, **ONNX Runtime**, OpenVINO, ExecuTorch, **RKNN Toolkit 2**

- **Edge device(s)**: Raxda Rock 5B - Rockchip **RK3588** SoC

- **Result**: Data and training pipeline, with model deployment to an NPU



Use-case: Leap Technologies

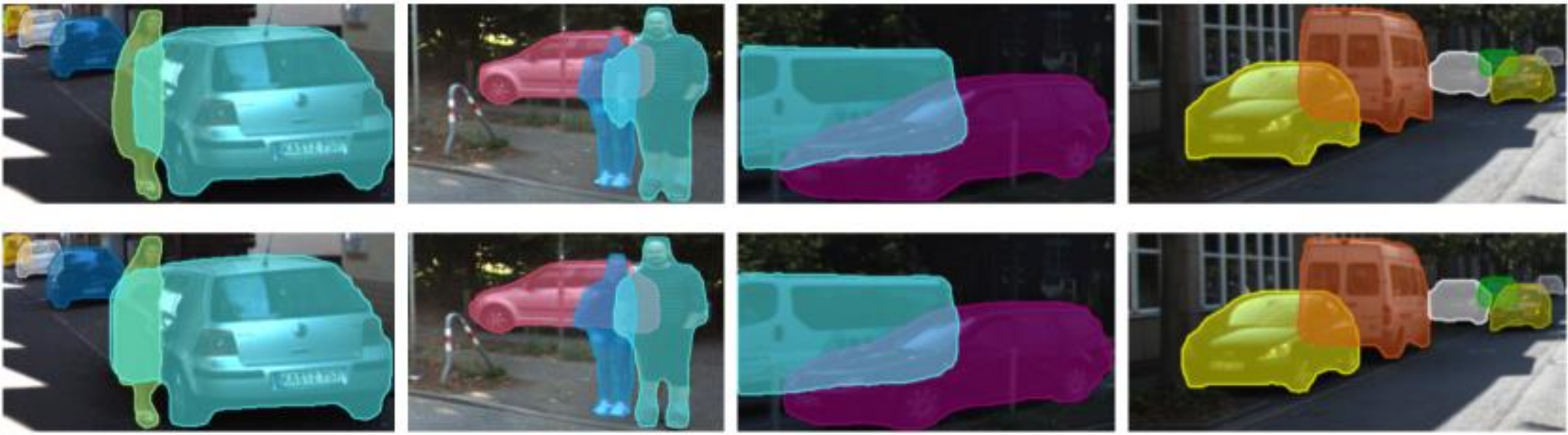
- Where did this project start?
- Leap Technologies provided us with a model:

ORCNN: Learning to See the Invisible: End-to-End Trainable Amodal Instance Segmentation

- And a question: how can we run this efficiently on an edge device?

With the support of VLAIO

Occlusion-aware instance segmentation



With the support of VLAIO

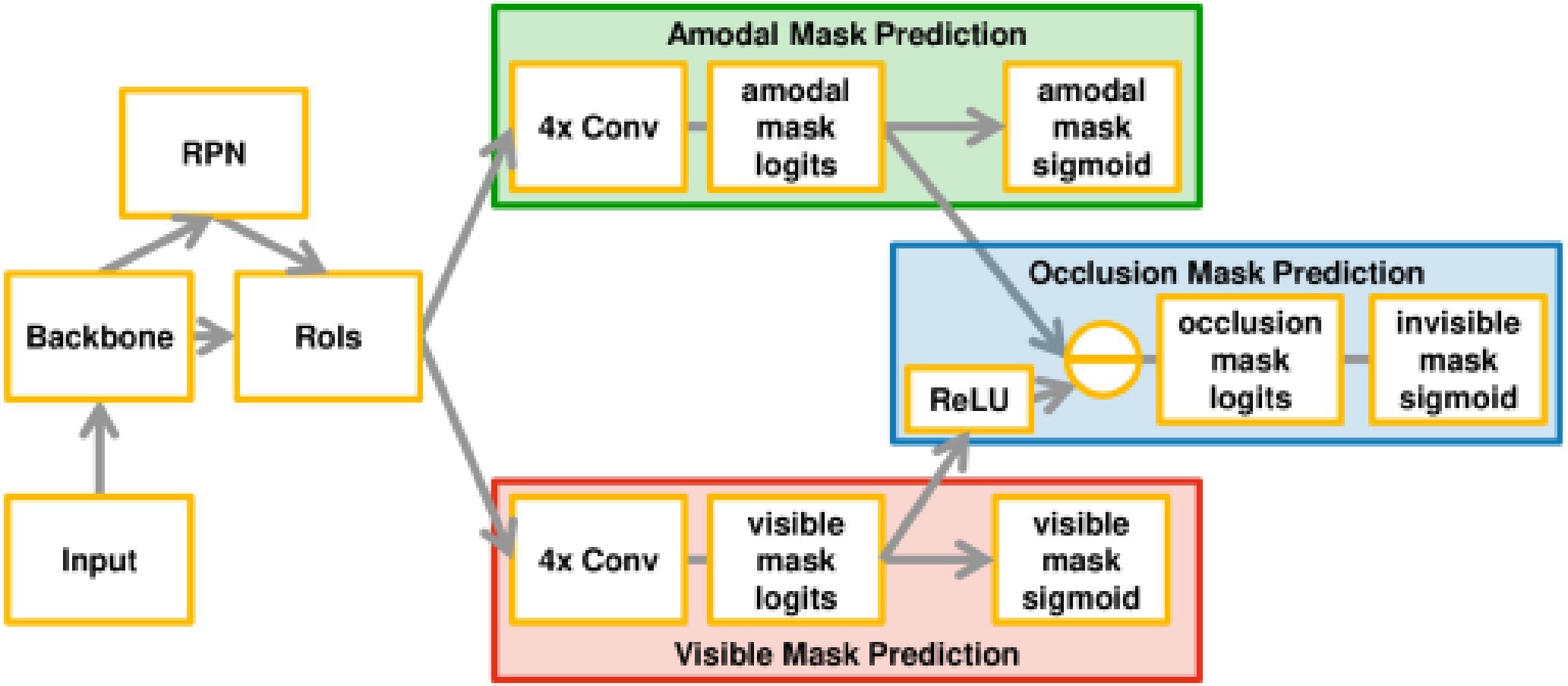
Occlusion-aware instance segmentation

- **Image classification:** Assigns a single label to the entire image
- **Object detection**: Identifies objects with bounding boxes and labels for each
- **Instance segmentation:** Generates pixel-level segmentation masks for each detected object
- **Occlusion-aware segmentation:** Extends instance segmentation by predicting masks for both visible and occluded (invisible) object parts

- These techniques build upon one another, with each successive method being more **computationally demanding,** especially on edge devices.

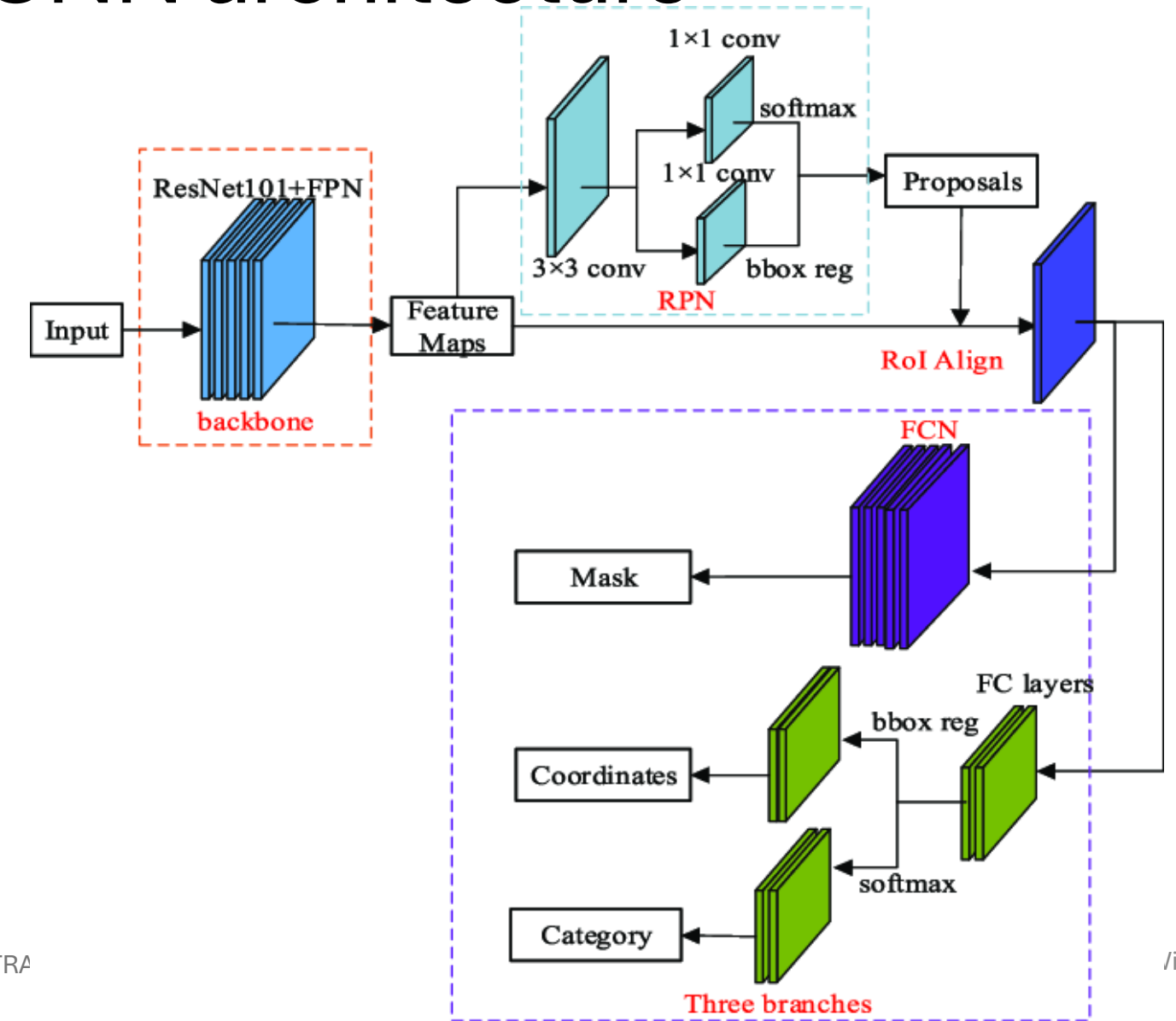
With the support of VLAIO

ORCNN = Mask-RCNN + extra head



With the support of VLAIO

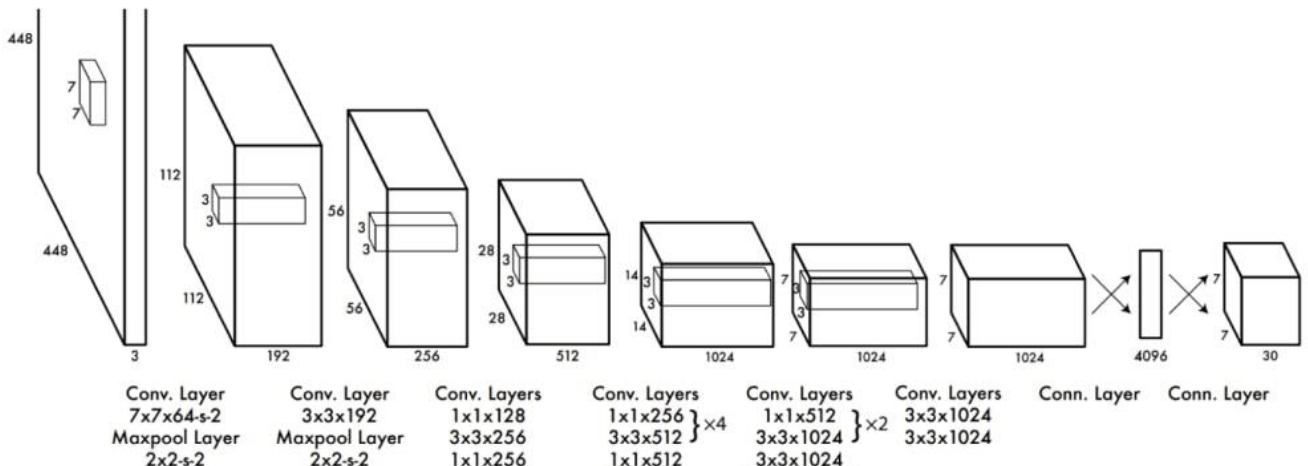
Mask-RCNN architecture



With the support of VLAIO

Two-stage vs one-stage models

- Mask-RCNN: A classic **two-stage** segmentation model
  - Quite good accuracy, but a bit slow
  - Typical two-stage model: first backbone -> than boxes/masks
- Better for edge deployment: **one-stage models**
  - Popular: YOLO (You Only Look Once) family of models
  - Faster inference, but worse performance (in general)



Note about licensing

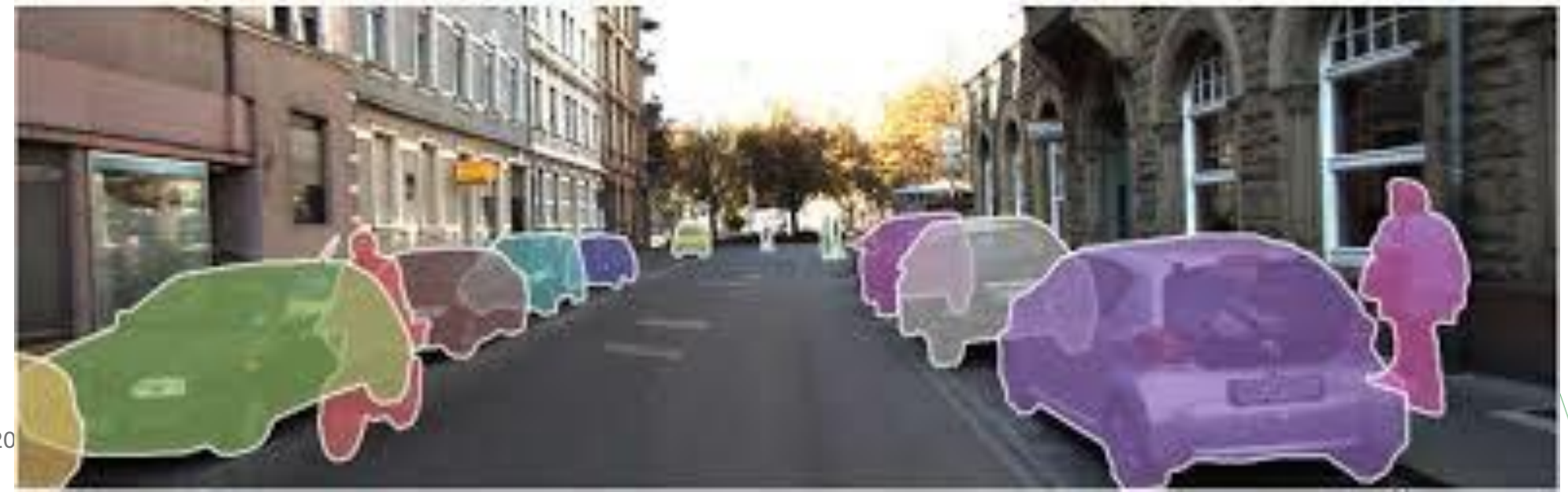
- Open source comes in two flavours:
- Copy-left licenses (GPL, AGPL)
  - All code must remain open
  - Beware if you use in commercial
  - E.g. Linux, Ultralytics YOLO

- Permissive licenses (MIT, Apache)
  - Code free to use how you want
  - Often just need to mention authors
  - E.g. Pytorch, Torchvision

With the support of VLAIO

Occlusion-aware dataset?

- Some datasets exist: COCOA, D2SA, KINS
- Problem: Ground trust often **hand-annotated**
  - Humans must guess what's behind occluded areas
- => Resulting in **low-quality ground truth labels**



Occlusion-aware dataset?

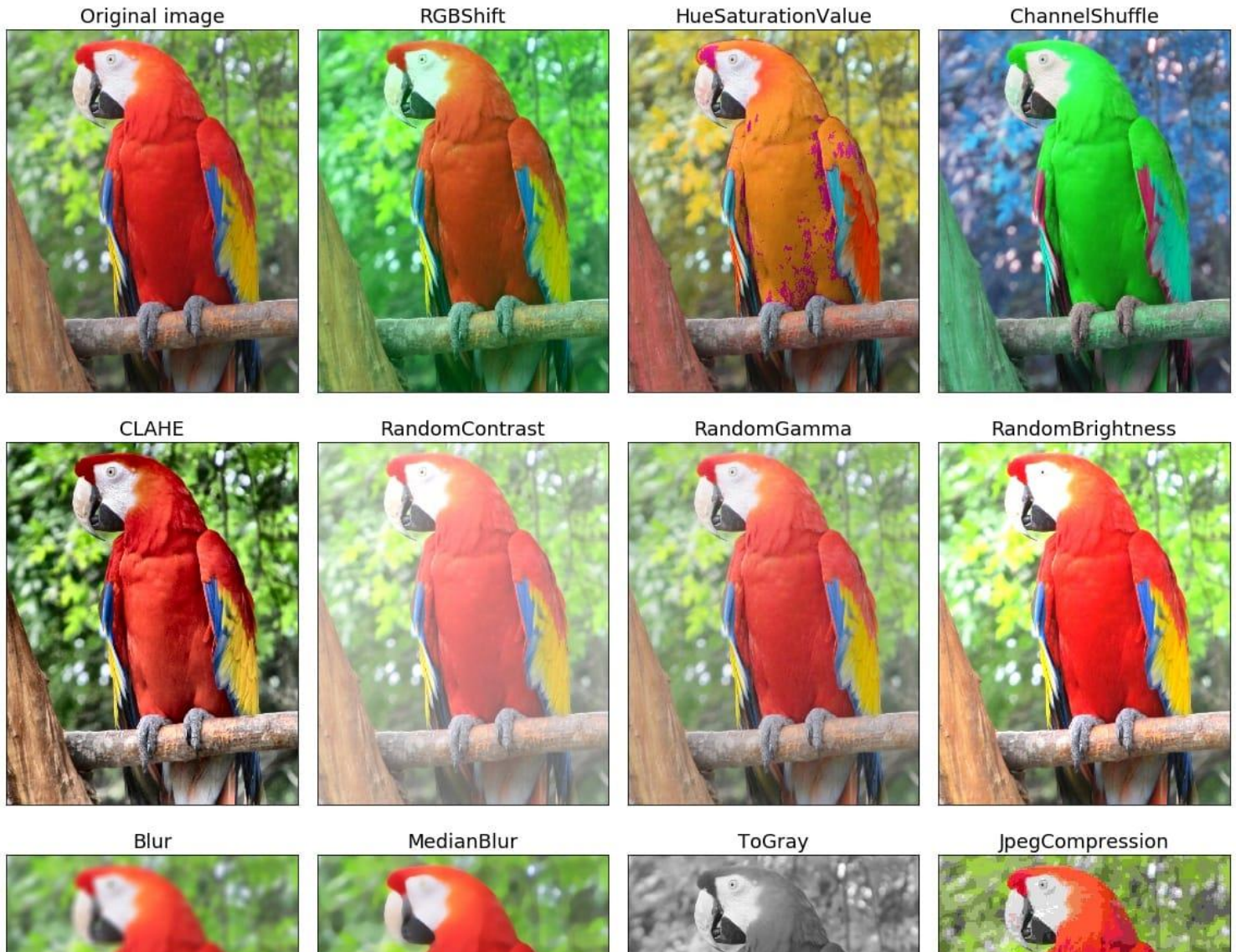
- Better idea:
- Generate a dataset by **copy-pasting objects on top of each other**
- This heavy data augmentation is a form of **synthetic data**

- Synthetic data => we know the ground trust labels

- Make sure to combine this with **classic data augmentation**
- => we use the Python library **Albumentations** for that

With the support of VLAIO

Data augmentation: Albumentations



e support of VLAIO

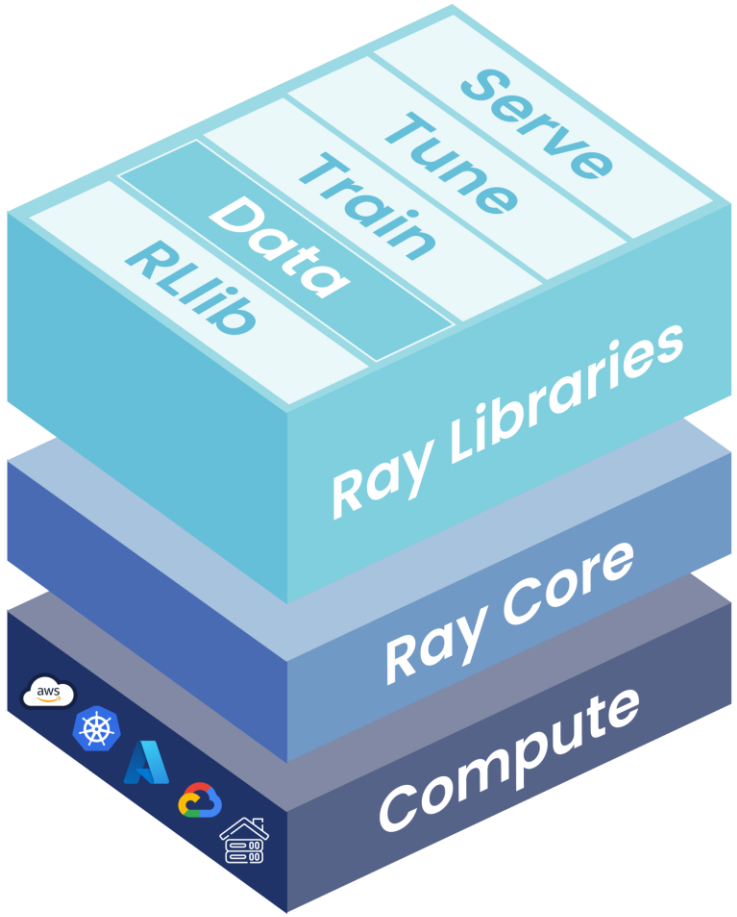
Synthetic data generation

- To generate data we must:
  - Decode PNG/JPGs and segmentation masks (RLE)
  - Cut out objects, augment individual objects
  - Combine objects into scene, on top of a background
  - Encode to PNG/JPG again and store it
- => All of this is very **CPU-heavy**

- How to go faster? Threading? Cluster?

Ray Data: Scalable Datasets for ML

- Solution: **Ray Data** cluster
- Handles the distribution of jobs
  across multiple worker nodes
- An alternative to **Spark**, **SQL warehouses**…
- But **focused on Machine Learning**
- Ray Data => really likes **Parquet** files (in S3)

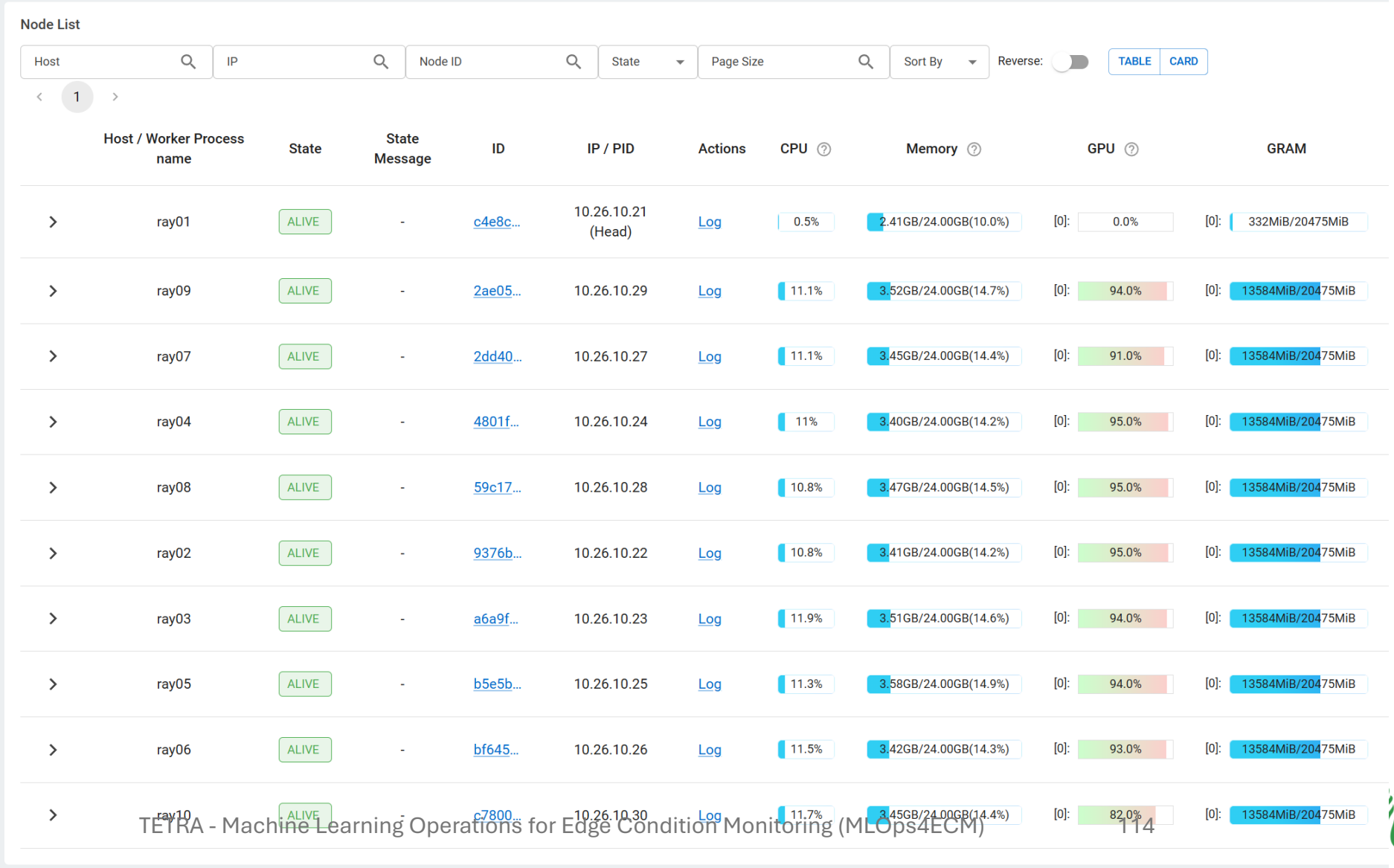


With the support of VLAIO

Ray Data: Scalable Datasets for ML

```python
from typing import Dict
import numpy as np
import torch
import ray

class TorchPredictor:

    def __init__(self):
        self.model = torch.nn.Identity()
        self.model.eval()

    def __call__(self, batch: Dict[str, np.ndarray]) -> Dict[str, np.ndarray]:
        inputs = torch.as_tensor(batch["data"], dtype=torch.float32)
        with torch.inference_mode():
            batch["output"] = self.model(inputs).detach().numpy()
        return batch

ds = (
    ray.data.from_numpy(np.ones((32, 100)))
    .map_batches(TorchPredictor, concurrency=2)
)
```

Ray Data: Scalable Datasets for ML



Node List

Host | IP | Node ID | State | Page Size | Sort By | Reverse: | TABLE | CARD

1

	Host / Worker Process name	State	State Message	ID	IP / PID	Actions	CPU	Memory	GPU	GRAM
>	ray01	ALIVE	-	c4e8c...	10.26.10.21 (Head)	Log	0.5%	2.41GB/24.00GB(10.0%)	[0]: 0.0%	[0]: 332MiB/20475MiB
>	ray09	ALIVE	-	2ae05...	10.26.10.29	Log	11.1%	3.52GB/24.00GB(14.7%)	[0]: 94.0%	[0]: 13584MiB/20475MiB
>	ray07	ALIVE	-	2dd40...	10.26.10.27	Log	11.1%	3.45GB/24.00GB(14.4%)	[0]: 91.0%	[0]: 13584MiB/20475MiB
>	ray04	ALIVE	-	4801f...	10.26.10.24	Log	11%	3.40GB/24.00GB(14.2%)	[0]: 95.0%	[0]: 13584MiB/20475MiB
>	ray08	ALIVE	-	59c17...	10.26.10.28	Log	10.8%	3.47GB/24.00GB(14.5%)	[0]: 95.0%	[0]: 13584MiB/20475MiB
>	ray02	ALIVE	-	9376b...	10.26.10.22	Log	10.8%	3.41GB/24.00GB(14.2%)	[0]: 95.0%	[0]: 13584MiB/20475MiB
>	ray03	ALIVE	-	a6a9f...	10.26.10.23	Log	11.9%	3.51GB/24.00GB(14.6%)	[0]: 94.0%	[0]: 13584MiB/20475MiB
>	ray05	ALIVE	-	b5e5b...	10.26.10.25	Log	11.3%	3.58GB/24.00GB(14.9%)	[0]: 94.0%	[0]: 13584MiB/20475MiB
>	ray06	ALIVE	-	bf645...	10.26.10.26	Log	11.5%	3.42GB/24.00GB(14.3%)	[0]: 93.0%	[0]: 13584MiB/20475MiB
>	ray10	ALIVE	-	c7800...	10.26.10.30	Log	11.7%	3.45GB/24.00GB(14.4%)	[0]: 82.0%	[0]: 13584MiB/20475MiB

Where to store data? S3!

- Local disk => quite limited
- Network file system => possible, but not ideal

- **Object storage** in cloud => max scale, lowest cost
  - **Amazon S3** => modern standard for object storage
- Object storage in cluster => local data, faster access
  - Sync to AWS, Azure, GCP … for long term storage

- We use **Ray Data + Minio S3** for data handling

With the support of VLAIO

Fish Segmentation Dataset

We decided to apply these ideas to instance segmentation of **fish**, to detect **overlap** (singulation).

- Steps to take:
  - Use an existing fish dataset with masks (but without occlusions).
  - Generate a synthetic and augmented dataset using **Ray Data**
  - Train an edge AI model on that data

With the support of VLAIO

Original dataset => little variation

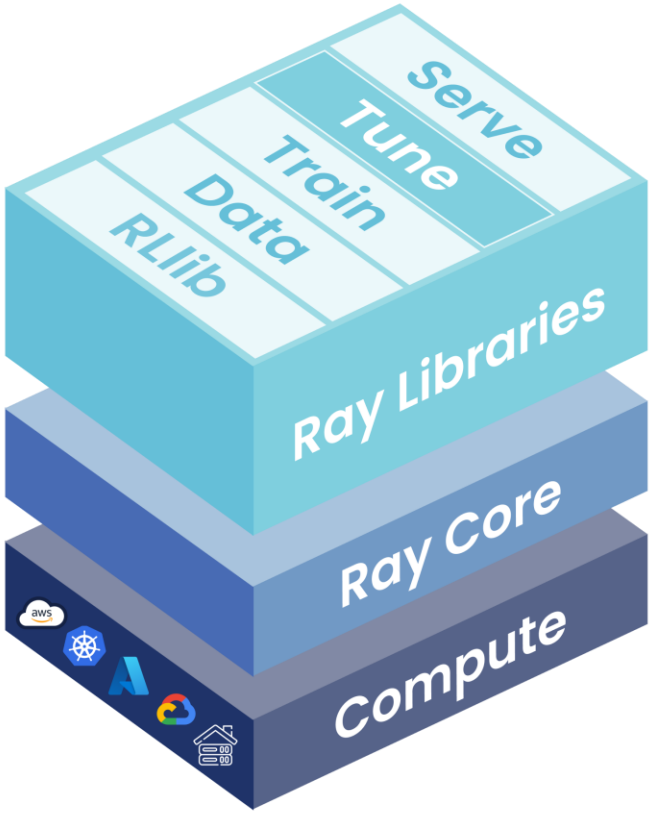
Cut out single fish: from masks

With the support of VLAIO

Compose into complex images

With the support of VLAIO

Model training: Ray Train/Tune



- How can we best **train ML models**?

- On a single laptop, workstation, server?
- What if we want to try many configurations?

- Solution:
- **Ray Train**: distributed model training (1 model, many servers)
- **Ray Tune**: hyper parameter search (many models, many servers)

With the support of VLAIO

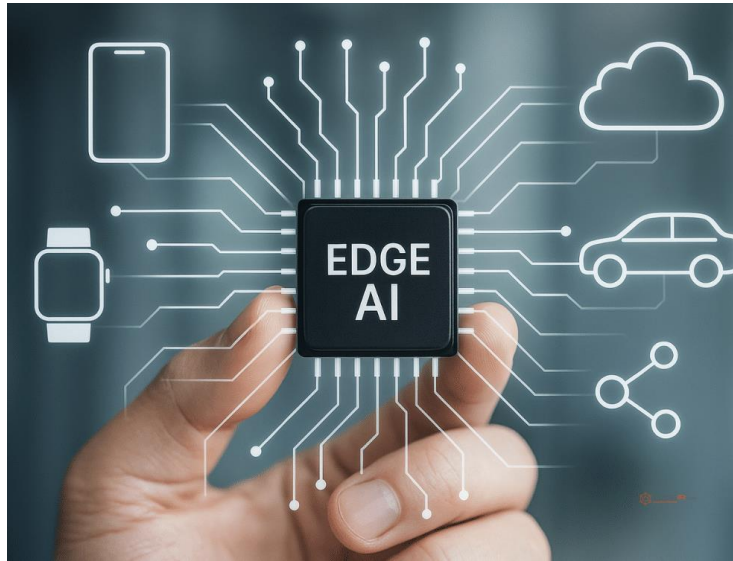
Model training: for the Edge

- We are going to train models for **edge deployment**
- These will not be huge models => fit on single GPU

- We choose: Ray Tune **hyperparameter optimization**

- Ray Tune will **train many models** for us,
  and finds the best combination of parameters

- Hyperparameters: LR, Regularization, layers…



Ubuntu | Ubuntu

```
(venv) root@ray01:~/leap-use-case/03-hyper-tune# python tune_rfdetr.py
2025-10-08 05:14:57,386 INFO worker.py:1771 -- Connecting to existing Ray cluster at address: 10.26.10.21:6379...
2025-10-08 05:14:57,394 INFO worker.py:1942 -- Connected to Ray cluster. View the dashboard at http://10.26.10.21:8265
```

Configuration for experiment	rfdetr_lr_wd_sweep
Search algorithm	BasicVariantGenerator
Scheduler	AsyncHyperBandScheduler
Number of trials	9

```
View detailed results here: ray/rfdetr_lr_wd_sweep
To visualize your results with TensorBoard, run: `tensorboard --logdir /tmp/ray/session_2025-10-08_05-02-33_808306_966/artifacts/2025
-10-08_05-14-57/rfdetr_lr_wd_sweep/driver_artifacts`

Trial status: 9 PENDING
Current time: 2025-10-08 05:14:57. Total running time: 0s
Logical resource usage: 0/144 CPUs, 0/9 GPUs (0.0/9.0 accelerator_type:RTX)
```

Trial name	status	lr	weight_decay
train_rfdetr_tune_bb951_00000	PENDING	3e-05	3e-05
train_rfdetr_tune_bb951_00001	PENDING	0.0001	3e-05
train_rfdetr_tune_bb951_00002	PENDING	0.0003	3e-05
train_rfdetr_tune_bb951_00003	PENDING	3e-05	0.0001
train_rfdetr_tune_bb951_00004	PENDING	0.0001	0.0001
train_rfdetr_tune_bb951_00005	PENDING	0.0003	0.0001
train_rfdetr_tune_bb951_00006	PENDING	3e-05	0.0003
train_rfdetr_tune_bb951_00007	PENDING	0.0001	0.0003
train_rfdetr_tune_bb951_00008	PENDING	0.0003	0.0003

```
Trial train_rfdetr_tune_bb951_00000 started with configuration:
```

Trial train_rfdetr_tune_bb951_00000 config	
batch_size	2
dataset_dir	...ynthetic/dataset2
epochs	100
grad_accum_steps	8
lr	0.00003
resolution	384
weight_decay	0.00003

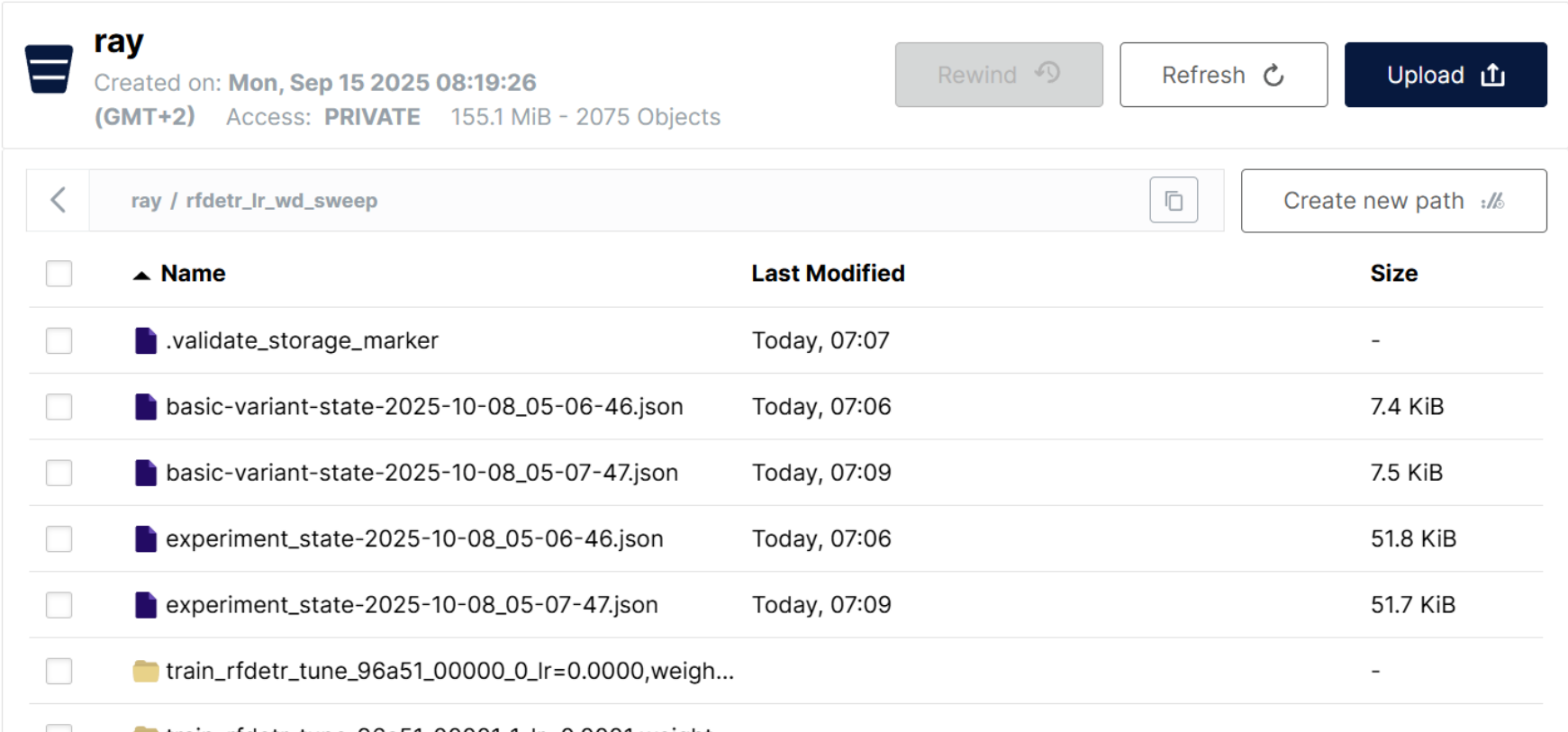
```
Trial train_rfdetr_tune_bb951_00002 started with configuration:
```

Trial name	status	lr	weight_decay
train_rfdetr_tune_abe18_00000	RUNNING	3e-05	3e-05
train_rfdetr_tune_abe18_00001	RUNNING	0.0001	3e-05
train_rfdetr_tune_abe18_00002	RUNNING	0.0003	3e-05
train_rfdetr_tune_abe18_00003	RUNNING	3e-05	0.0001
train_rfdetr_tune_abe18_00004	RUNNING	0.0001	0.0001
train_rfdetr_tune_abe18_00005	RUNNING	0.0003	0.0001
train_rfdetr_tune_abe18_00006	RUNNING	3e-05	0.0003
train_rfdetr_tune_abe18_00007	RUNNING	0.0001	0.0003
train_rfdetr_tune_abe18_00008	RUNNING	0.0003	0.0003

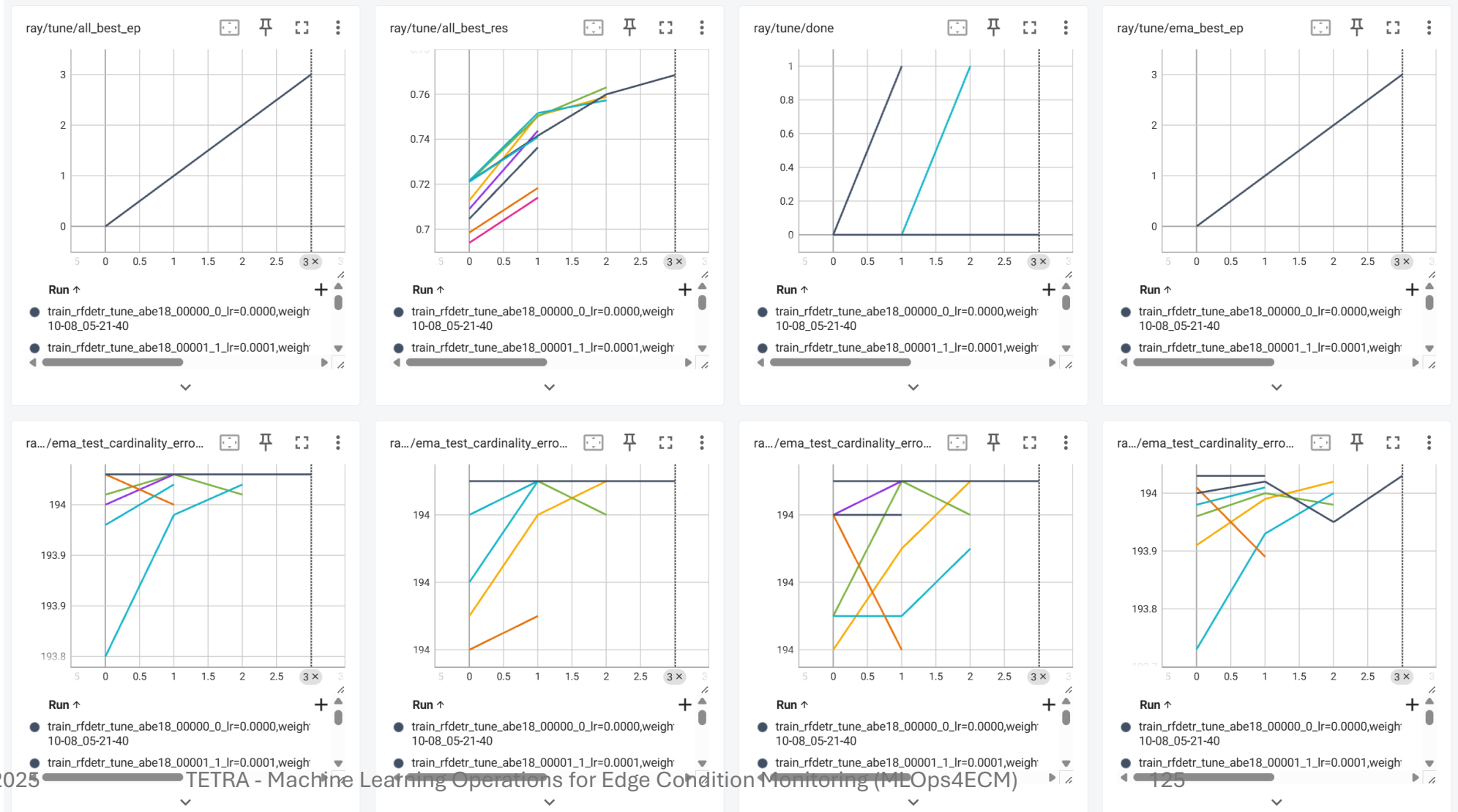
With the support of VLAIO

Ray Tune: store artifacts in S3

- Training creates **artifacts**: models, metrics, predictions…
- We configure Ray Tune to store these in Minio S3



Tensorboard: live training graphs



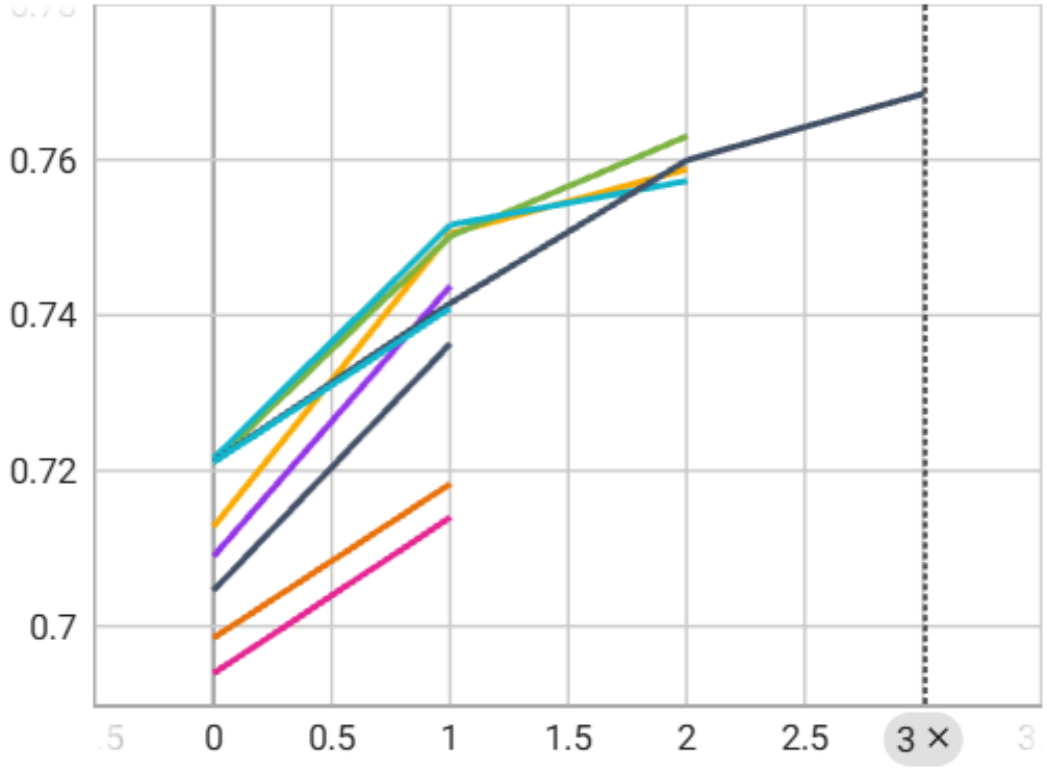
TensorBoard TIME SERIES SCALARS

Filter runs (regex)

Run ↑

- train_rfdetr_tune_abe18_00000_0_lr=0.0000,weight_decay=0.0000_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00001_1_lr=0.0001,weight_decay=0.0000_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00002_2_lr=0.0003,weight_decay=0.0000_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00003_3_lr=0.0000,weight_decay=0.0001_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00004_4_lr=0.0001,weight_decay=0.0001_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00005_5_lr=0.0003,weight_decay=0.0001_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00006_6_lr=0.0000,weight_decay=0.0003_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00007_7_lr=0.0001,weight_decay=0.0003_2025-10-08_05-21-40
- train_rfdetr_tune_abe18_00008_8_lr=0.0003,weight_decay=0.0003_2025-10-08_05-21-40

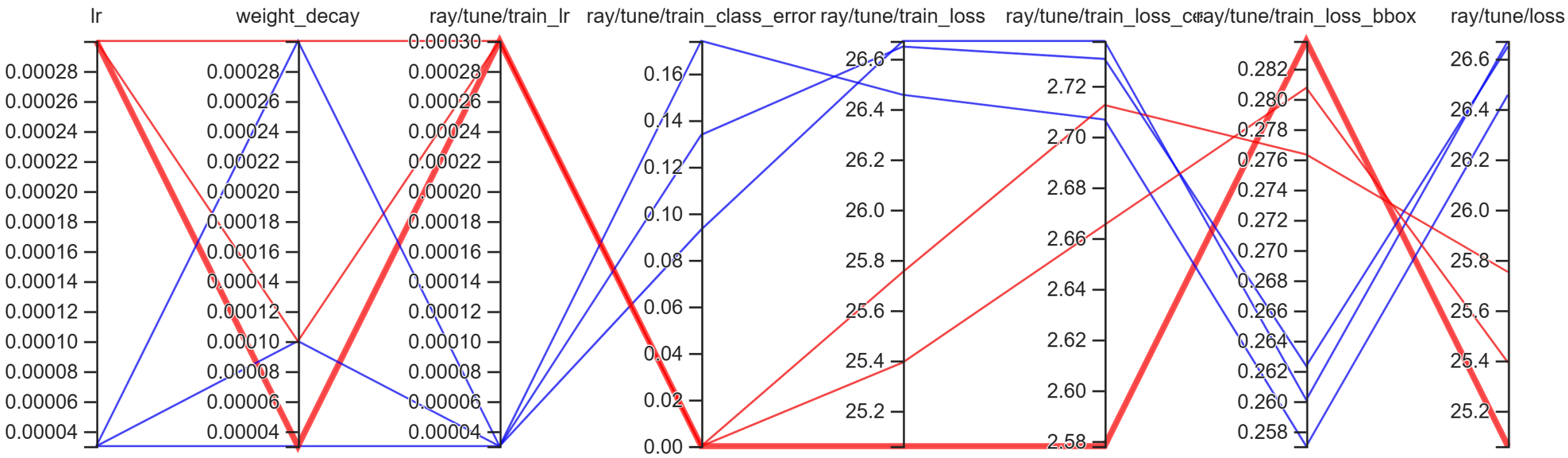
ray/tune/all_best_res



Run ↑

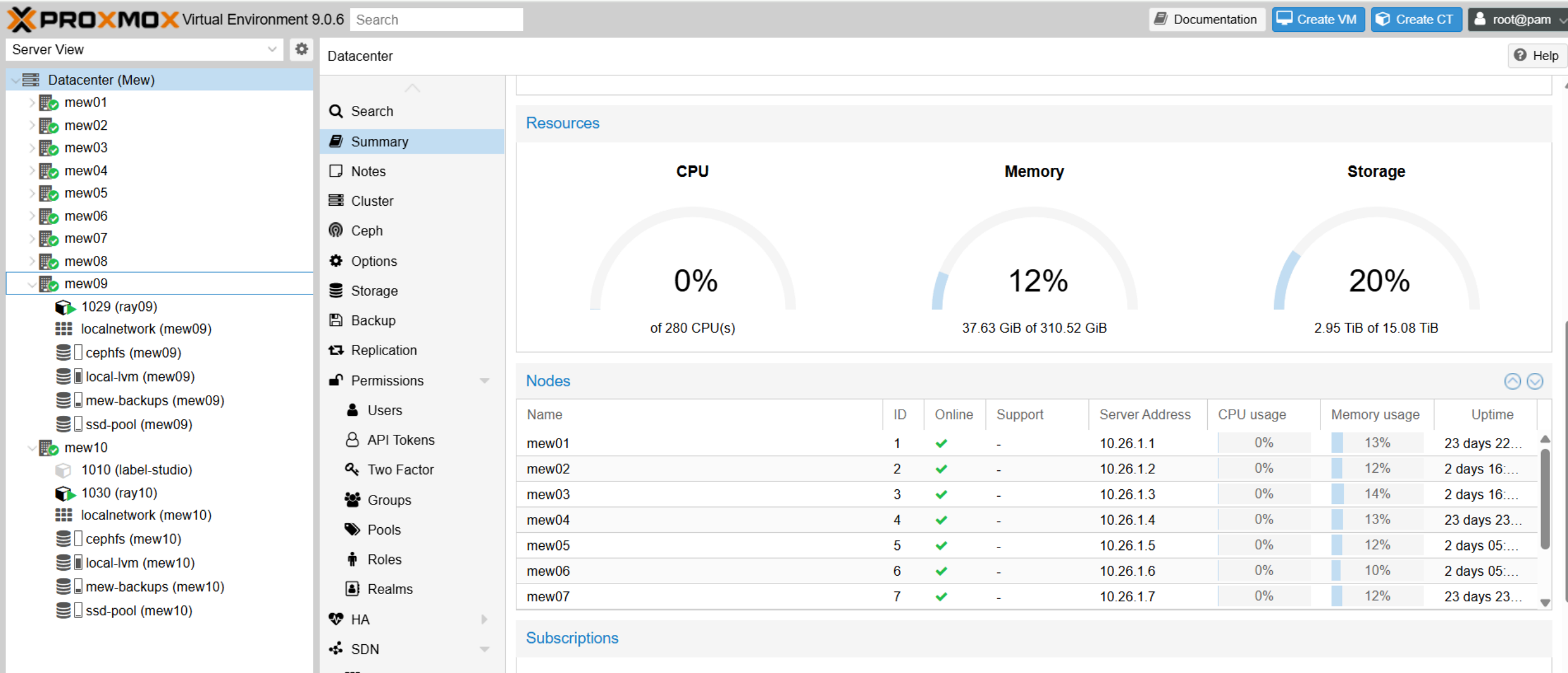
- train_rfdetr_tune_abe18_00000_0_lr=0.0000,weigh
  10-08_05-21-40
- train_rfdetr_tune_abe18_00001_1_lr=0.0001,weigh

Tensorboard + hyperparameter search



With the support of VLAIO

How we manage servers? Proxmox



Deployment: Which edge device?

- Most be quite **powerful** => instance segmentation is heavy
- But we want **cheap** hardware => so not a GPU server

- Jetson? Linux ARM board?

- => Radxa Rock 5B, with Rockchip SoC
- Has an **NPU** for neural acceleration

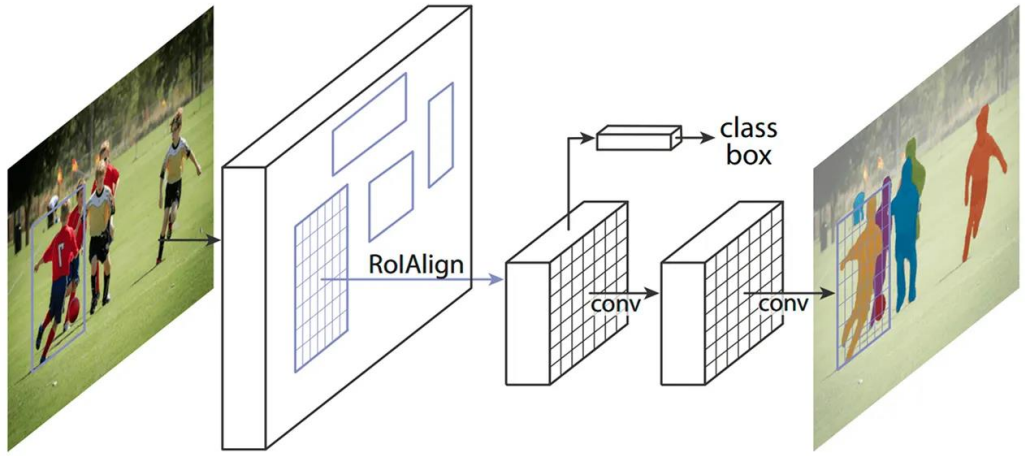
First model: Mask-RCNN

- Torchvision Mask-RCNN (Apache)
- Pytorch version: 8 seconds per frame
- ONNX Runtime (CPU): Just as bad

- OpenVINO for ARM: much faster! 5s per frame
- OpenVINO pinned to big CPU core: 3s per frame

- Backbone (ResNet-50) on **NPU**:
  2.5 seconds per frame

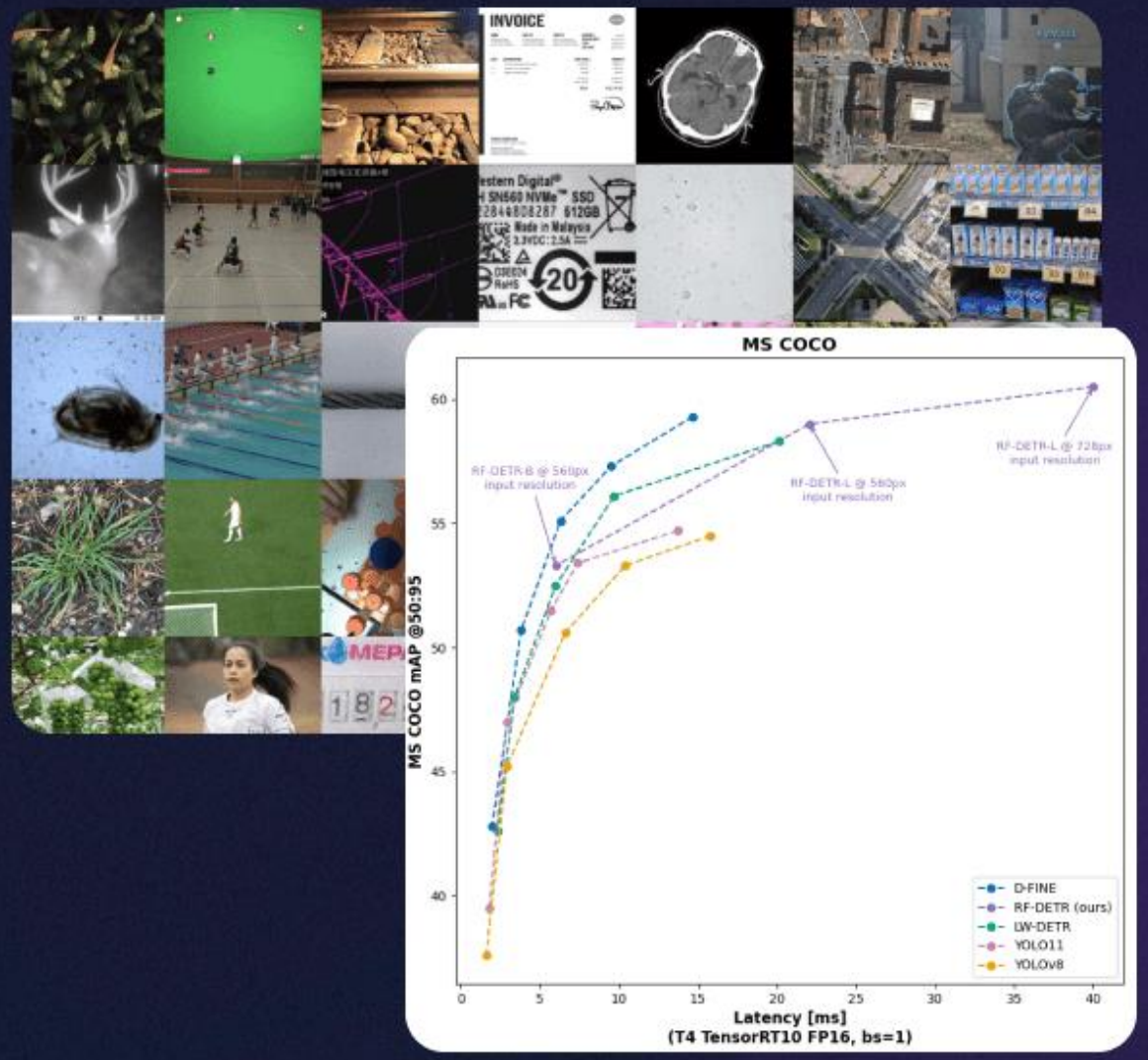
- Results: CPU+NPU **still quite slow**



Second model: RF-DETR

- Vision transformer (ViT) model
  - Two-stage model
  - DINOv2 backbone
- Faster and more accurate than Ultralytics YOLO11
- Permissive license: Apache

RF-DETR: A New SOTA, Real-Time Object Detection Model



roboflow

RF-DETR: Vision Transformer

- Pure Pytorch model: 1300 ms per frame
- ONNX Runtime (CPU): 1250 ms per frame

- Does not run:
  - OpenVINO, ORT ACL, ORC XNNPACK, TVM, Executorch

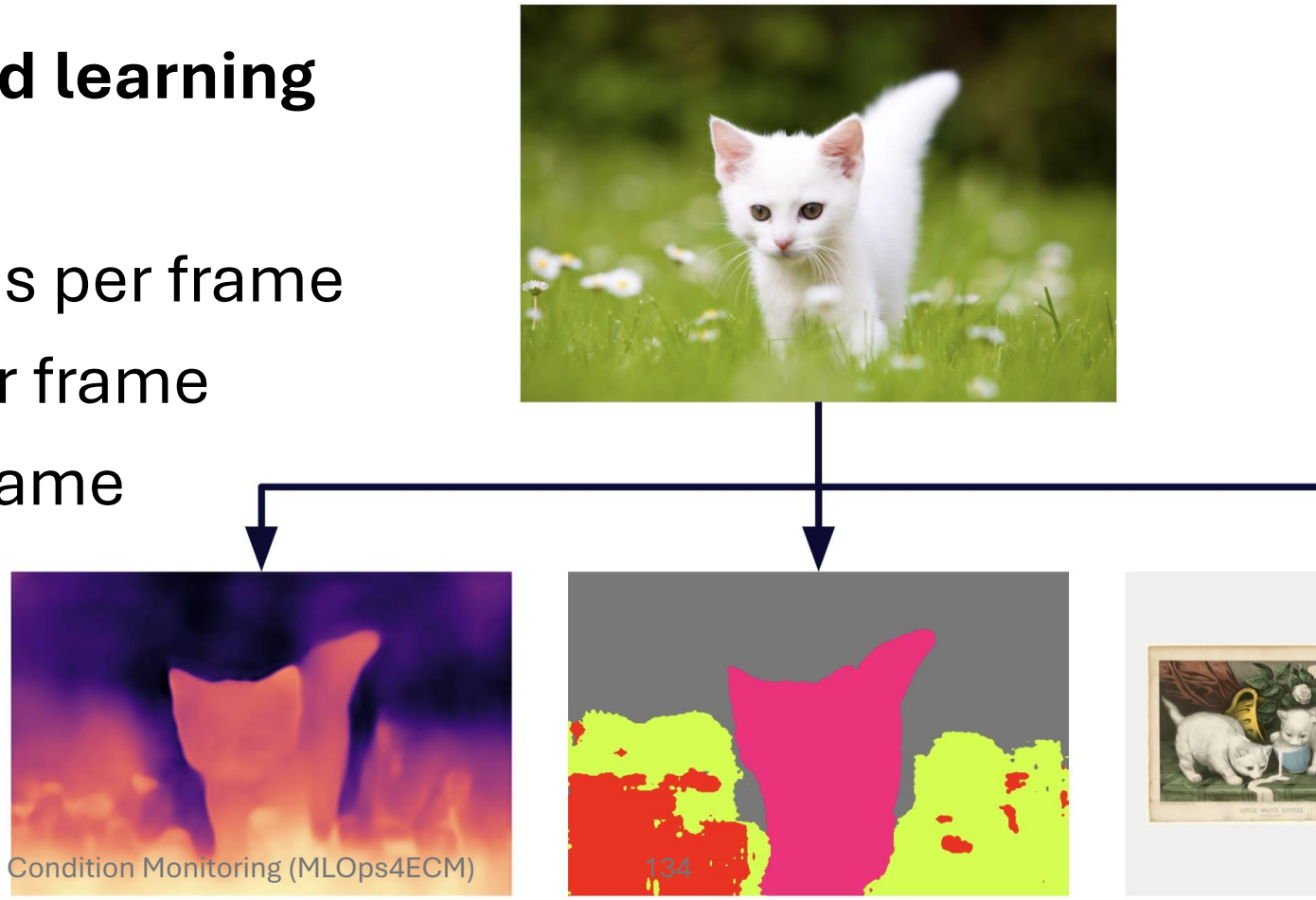
- Transformers **don’t have great support** on edge

RF-DETR backbone: DINOv2

- State-of-the-art vision encoder
- Trained using **self-supervised learning**

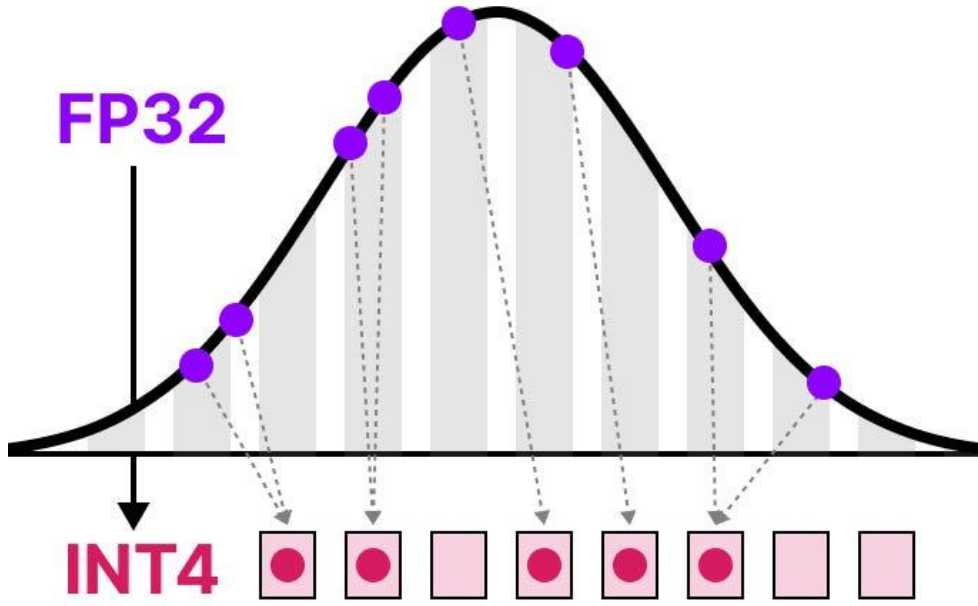
- ONNX Runtime (CPU): 950 ms per frame
- OpenVINO (CPU): 472 ms per frame
- Rockchip NPU: 367 ms per frame

- **Surprise**: runs on the NPU!



RF-DETR: Pushing the limit

- Backbone NPU + Detection ONNX Runtime
  - => 360 ms + 350 ms = 710 ms per frame
- Lower resolution: 432x432 -> 384x384
  - => 260 ms + 260 ms = 520 ms per frame
- Dynamic quantization of detection model on ONNX Runtime
  - => 260 ms + 210 ms = 470 ms per frame



Advantages of using the NPU

- Rockchip NPU gives us a **nice speed-up** (higher FPS)

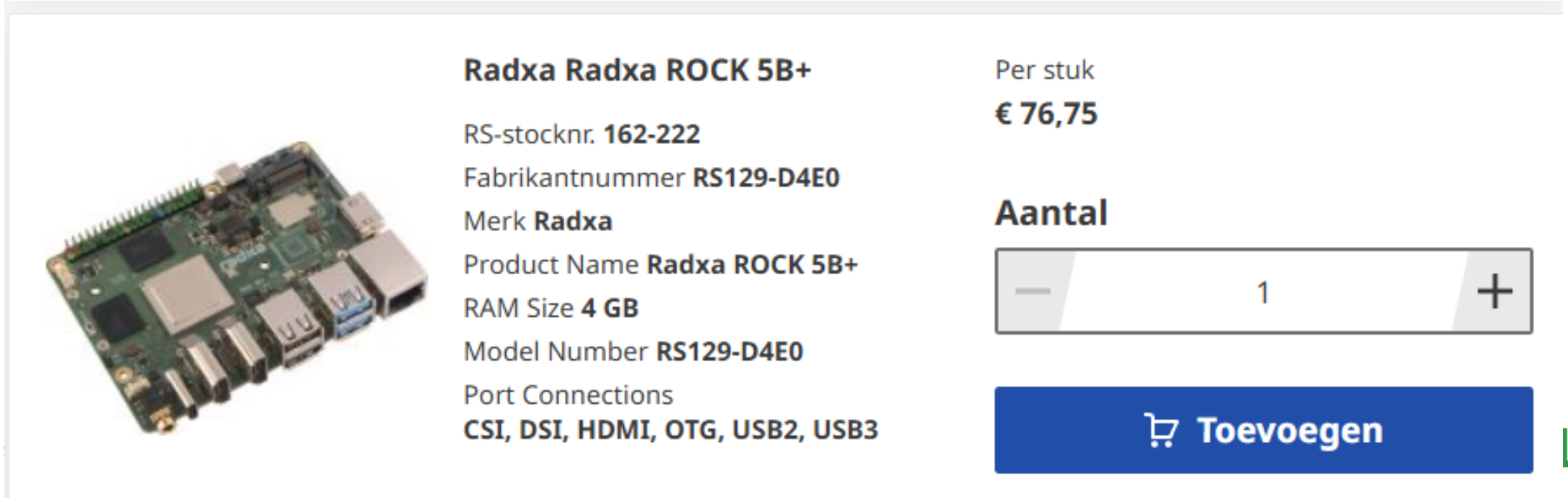
- NPU also has **lower power usage**
  - Rock 5B goes from 14W to 10W for higher FPS
  - NPU is more power efficient than the CPU when running the same neural network

- Now **CPU is free** to do other tasks

With the support of VLAIO

Conclusion: SOTA on ARM board

- We have deployed a state-of-the-art model
- This runs at 2 FPS on a < 100 euro board
- RF-DETR detection model runs at 5 FPS



Demo: two models

- Top: detection @ 5 FPS
- Bottom: segmentation @ 2 FPS

- Bottom trained on **synthetic data**
  - Generated with Ray Data
  - Transformers are very data-hungry

- Bottom trained using **Ray Tune**
  - Transformers have many parameters

Questions?

- Top: detection @ 5 FPS
- Bottom: segmentation @ 2 FPS

- Bottom trained on **synthetic data**
  - Generated with Ray Data
  - Transformers are very data-hungry

- Bottom trained using **Ray Tune**
  - Transformers have many parameters

Follow-up projects

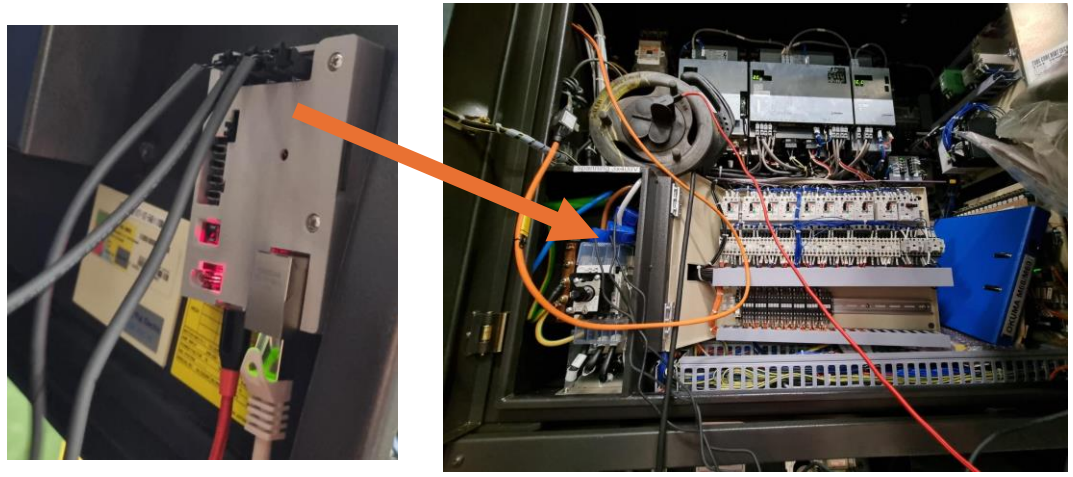
With the support of VLAIO

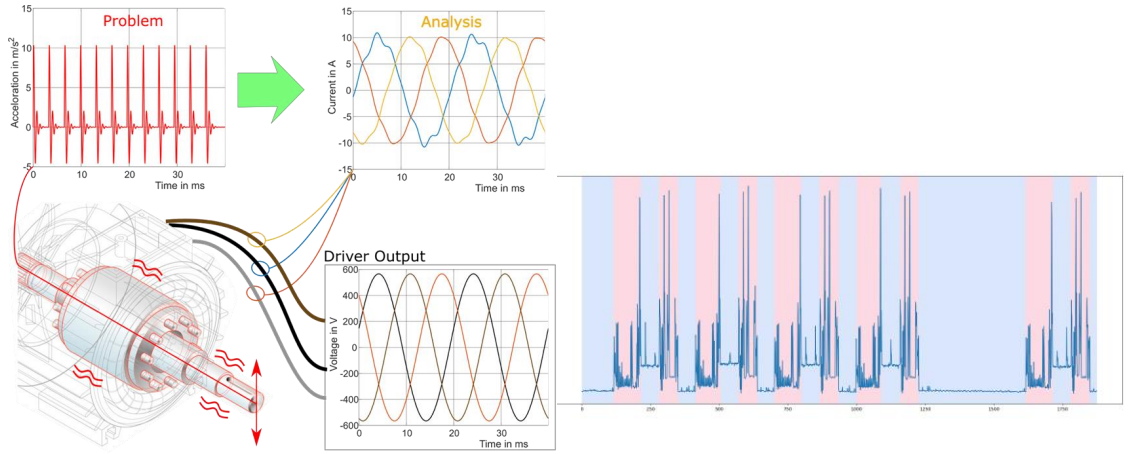
C3 Retrokit

Optimising retrofits with RetroKit

- Edge-based condition monitoring
- Modular machine learning approach
- Component- and system-level analysis

→ Call to action: Machine available?





With the support of VLAIO

TETRA AI4Automation

Unlocking Intelligence for Smart Factory Automation

- Goal: Supporting manufacturing companies in implementing AI within production environments through hands-on use cases, demonstrations, and guidance.
- Exploring the latest industrial AI tools (Siemens, Beckhoff, Advantech, Microsoft, NVIDIA) to identify practical, high-impact solutions.
- Call to action: Specific challenge? Let's talk!

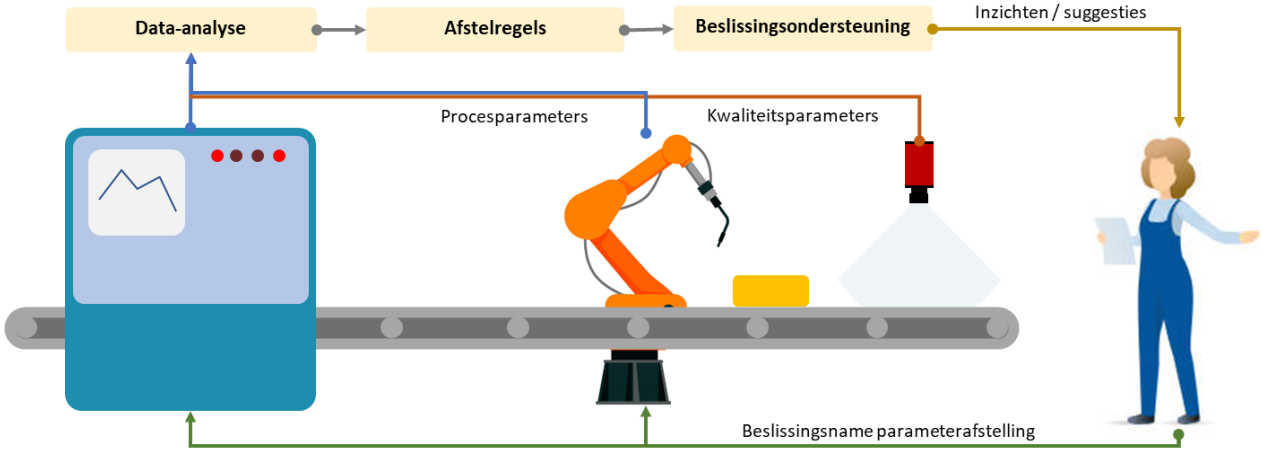
Sectors:
Food Processing – Polymer Processing – Machine building

With the support of VLAIO

CORNET-COOCK+ DeciQual

Data-driven decision support for quality control and process optimization

- Determining the influence of process parameters on quality parameters
- Communicating insights to the operator
- Supporting the operator towards (optimal) process control and parameter tuning



Sectors:
Food Processing – Polymer Processing – Machine building

With the support of VLAIO

Do you recognize similar challenges in your own company context?

Get in touch via
mathias.verbeke@kuleuven.be
jonas.lannoo@vives.be

With the support of VLAIO

Thanks to

With the support of VLAIO

Thanks to

Onderzoekers:

- Lara Luys (KU Leuven Brugge)
- Alexander D'hoore (Hogeschool VIVES Brugge)
- Sille Van Landschoot (Hogeschool VIVES Brugge)
- Noah Debaere (Hogeschool VIVES Brugge)

Promotoren:

- Mathias Verbeke (KU Leuven Brugge)
- Jonas Lannoo (Hogeschool VIVES Brugge)

With the support of VLAIO

VLAIO TETRA

Machine Learning Operations for Edge Condition Monitoring

(MLOps4ECM)

Reception with demos

Met steun van VLAIO

HBC.2023.0062