



# AI Timeseries Model Benchmarking as a 6G Service



**Prof. Dr. Andreas Kassler** (thanks to Khalid, Phil,..)  
Deggendorf Institute of Technology, Germany  
Karlstads Universitet, Sweden

# Evolution of Telco Networks

## 2000s

Closed Hardware Platforms

- Proprietary
- Difficult to Innovate

## 2010s

Virtualized Networks

- HW/SW decoupling
- Separate Control/Dataplane
- Network Programmability
- Rule-based, reactive Management

## 2020s

Cloud Nativeness

- Containerization
- Micro-Services
- Orchestration
- Rule-based Control-loops, traditional algorithms

## 2030s

AI nativeness

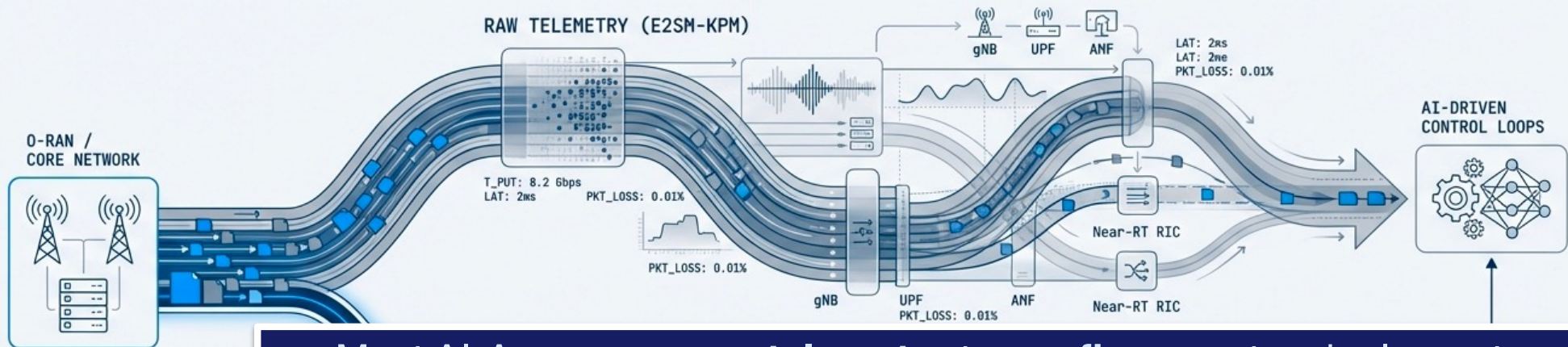
- AI-embedded capabilities
- AI native infrastructure
- Intelligent and Autonomous Control
- Proactive, Intelligent Management
- 1000s AI Models



# Key Enablers for AI Nativeness



# Example: AI-Driven Network Management



- ❑ Most AI-Apps consume **telemetry** to **configure** network elements
- ❑ Configurations need to be **updated through AI Models**
- ❑ **Proactive Network Management requires Forecasting**
- ❑ Example of AI-based Timeseries Forecasting Apps
  - ❑ Traffic Demand,
  - ❑ Ressource Availability,
  - ❑ Anomaly Detection,
  - ❑ User Mobility ..

# AI-based App examples (some)

## rApps (Non-RT RIC)



- **Host Location:**  
Central Cloud
- **Control Loop:**  
> 1 second
- **Optimization Focus:**  
Policy Management &  
Long-term optimization
- **Update Tolerance:**  
High tolerance for  
deployment delays.

## xApps (Near-RT RIC)



- **Host Location:**  
Regional Cloud
- **Control Loop:**  
10ms - 1 second
- **Optimization Focus:**  
Resource Control &  
Traffic Steering
- **Update Tolerance:**  
Moderate; sensitive to  
extreme queuing.

## dApps (Embedded CU/DU)



- **Host Location:**  
Edge Cloud
- **Control Loop:**  
< 10 milliseconds
- **Optimization Focus:**  
Real-time Radio Access &  
Beamforming
- **Update Tolerance:**  
Extremely fragile. Spawning  
delays instantly violate SLAs.

How to implement/update AI Models efficiently?

# AI-native stack in CNCF

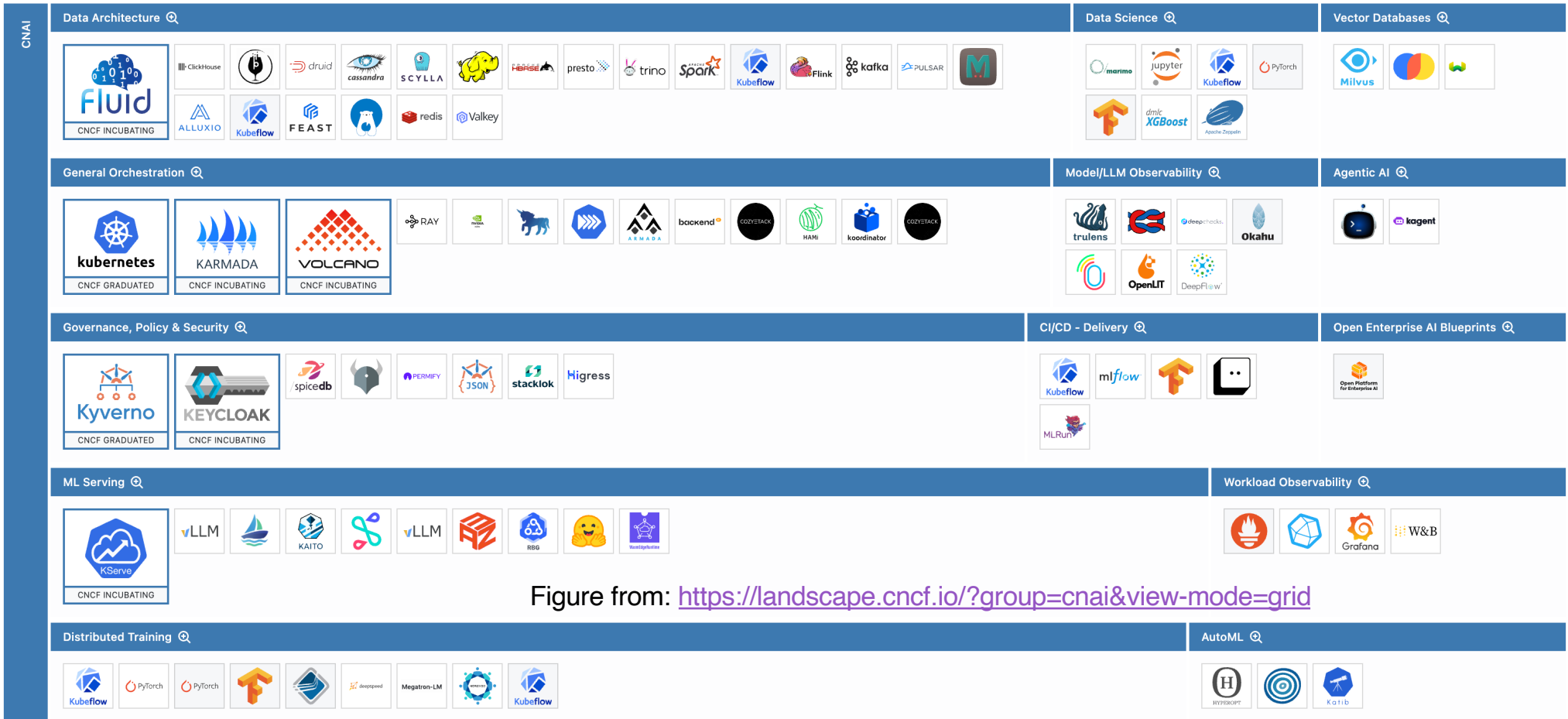
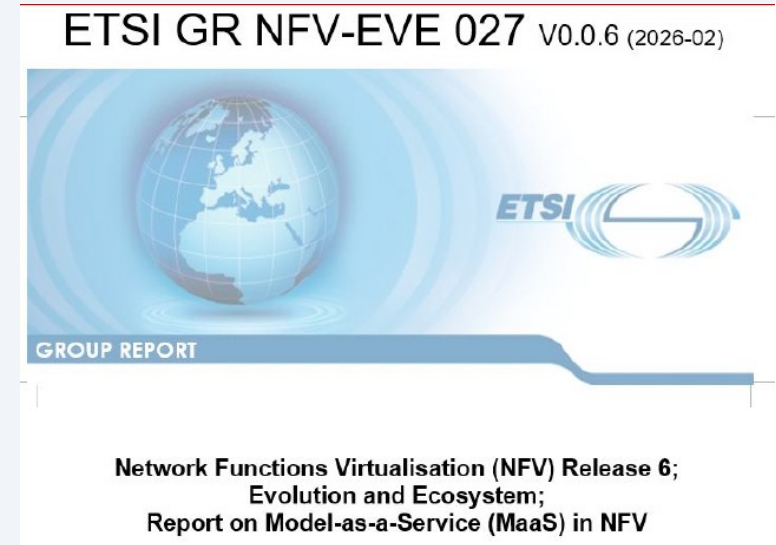


Figure from: <https://landscape.cncf.io/?group=cnai&view-mode=grid>

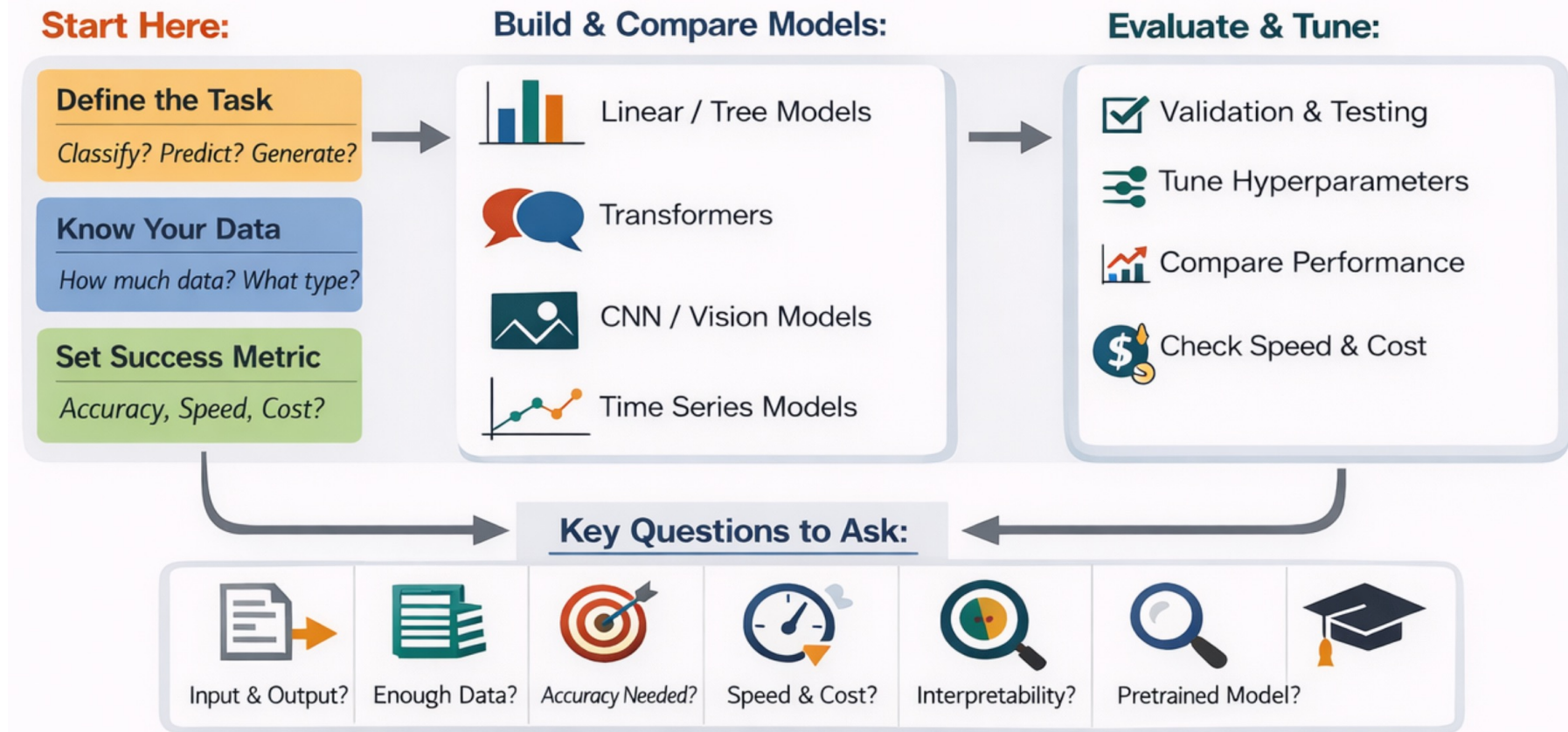
Many generic tools → How to use them in Telco AI-native Networks?

# (AI) Model as a Service

- **Model as a Service (MaaS):**
  - make AI models repeatable, deployable, and consumable at scale
  - models are treated as managed assets
- **Automation of the “model pipeline”**
  - **AI Model Benchmarking**
    - **which model fits my data and use-case best?**
      - Data preparation, Feature enrichment, Hyperparameters
      - Model Training, Model evaluation, Model versioning
  - **AI-Lifecycle Management**
    - **when to re-train and redeploy AI model where?**
      - Deployment, monitoring
      - Dynamic updating → hard
- **Typical capabilities**
  - High-performance inference service (quantization, pruning, runtime optimization, ...)
  - Privacy-preserving learning → Federated and Split Learning support
  - Integration with AI Agents







# The Pain of finding a good AI-Model






Good Model? Need to benchmark several ones on your dataset → Need to implement basic blocks for each AI-Model → Labour intensive → Automate!

# Key Factors to Consider when Choosing an AI Model








## Model Performance

-  **Accuracy / F1-Score**  
Predictive quality
-  **Calibration / Fairness**  
Bias, fairness, reliability
-  **Robustness / Stability**  
Generalizability, noise
-  **Inference Speed**  
Time per prediction

## Infrastructure & Deployment

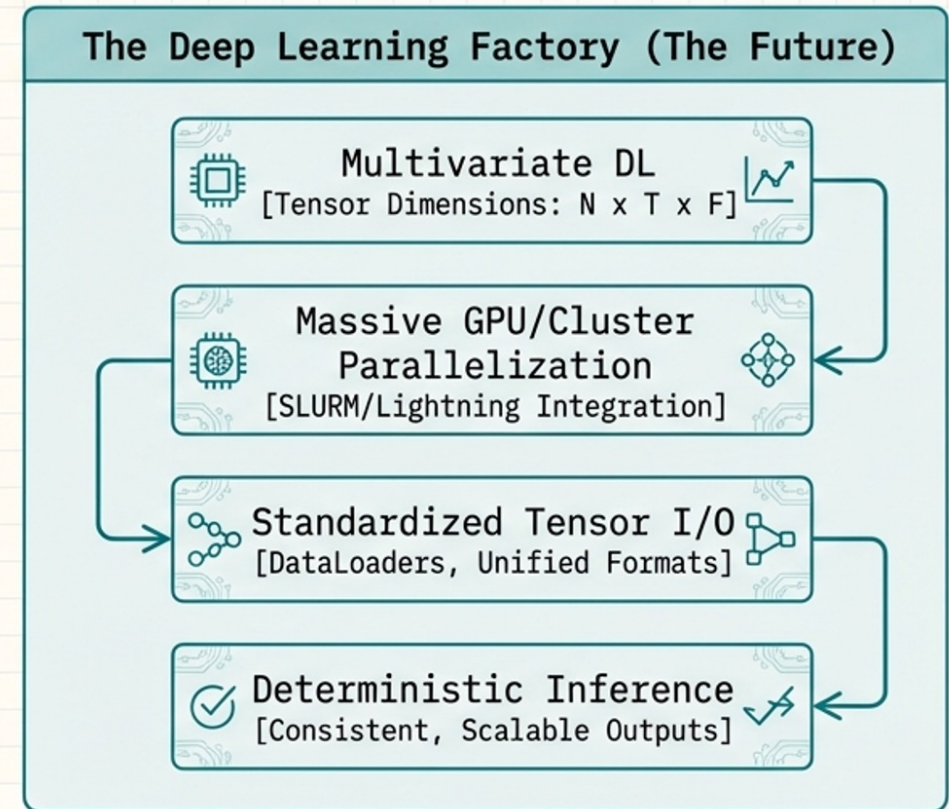
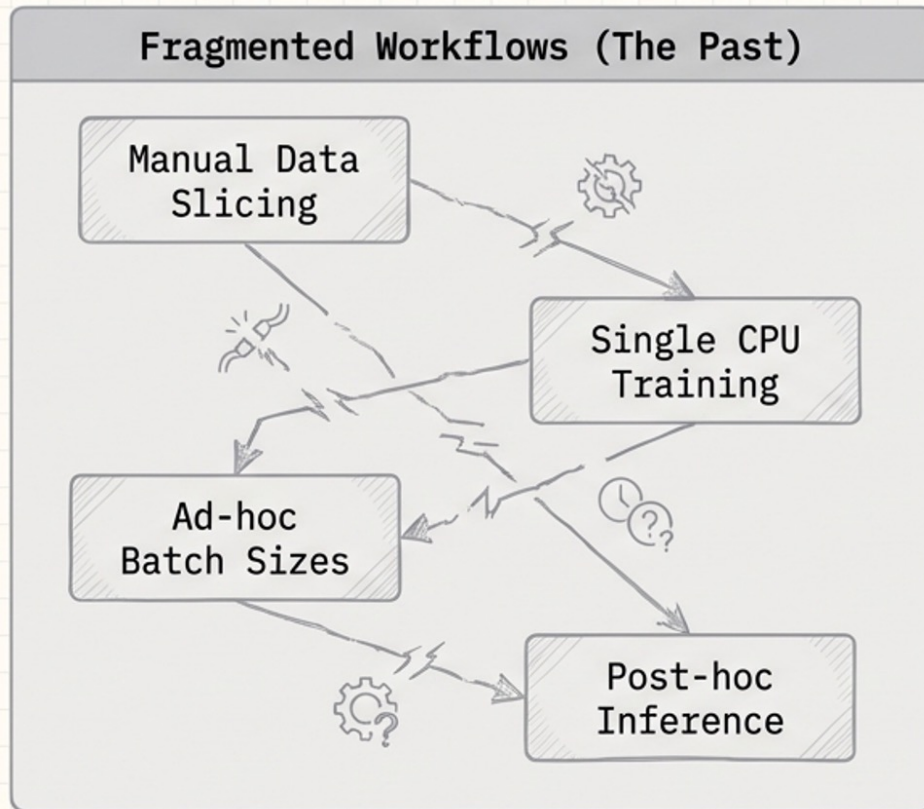
-  **Training Cost / Time**  
Time and resources needed
-  **Memory Usage**  
RAM and storage
-  **Cost / ROI**  
Cost per request
-  **Energy Consumption**  
Power and efficiency

## Evaluation KPIs:

-  Accuracy / F1
-  Latency / Throughput
-  Memory Usage
-  Cost / ROI
-  Interpretability
-  Complexity
-  Energy Efficiency

Need to consider infrastructure KPIs for optimal deployment decisions!  
**Example:** If two model have similar accuracy, pick the one with lower energy consumption

# The Reality of ML Model Development and Benchmarking

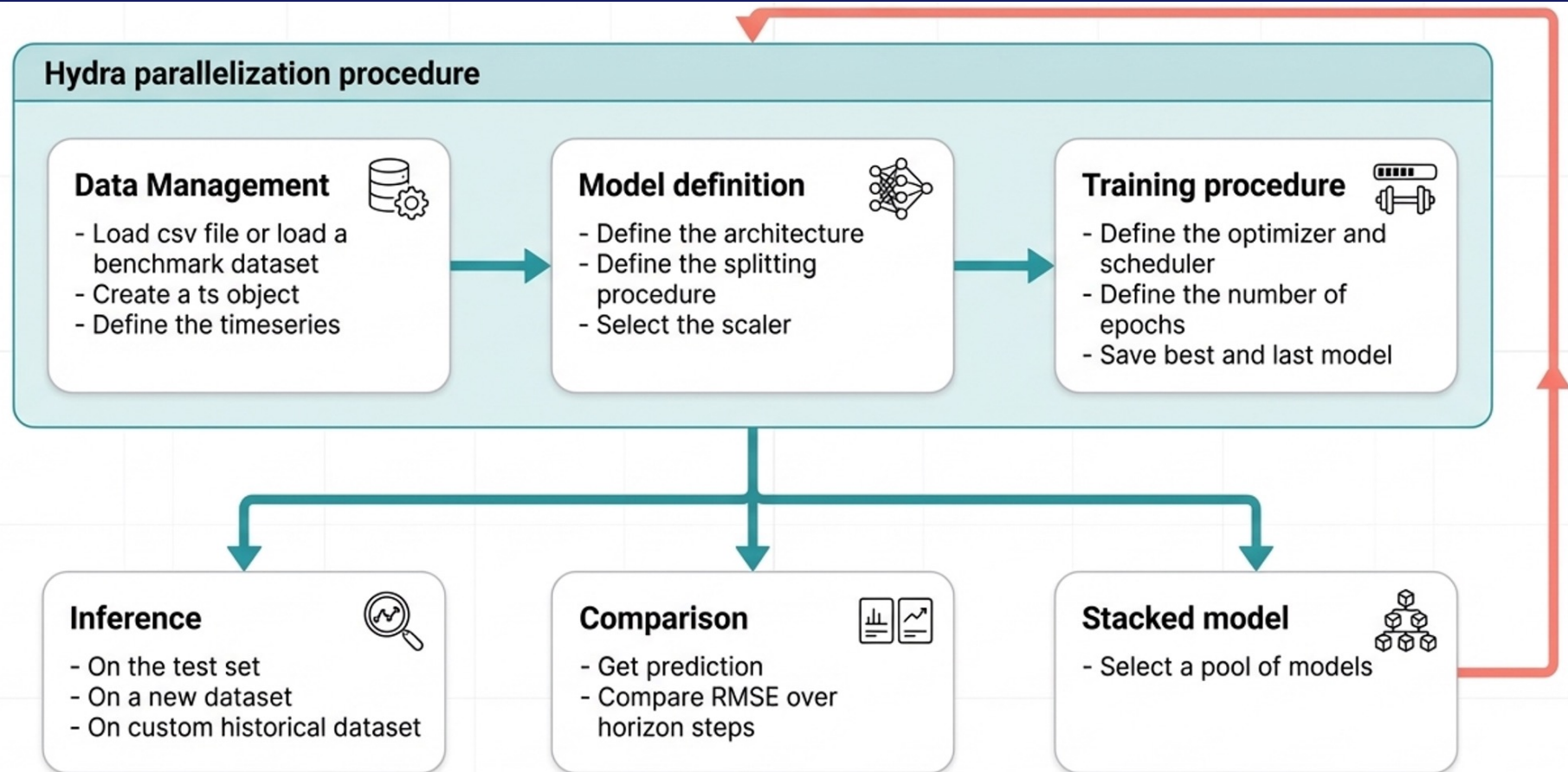


AI-based Models demand infrastructure built natively for Deep Learning, PyTorch Lightning, SLURM/K8S to automate ML Model benchmarking at Scale

# Key Requirements for AI Model Benchmarking as a Service

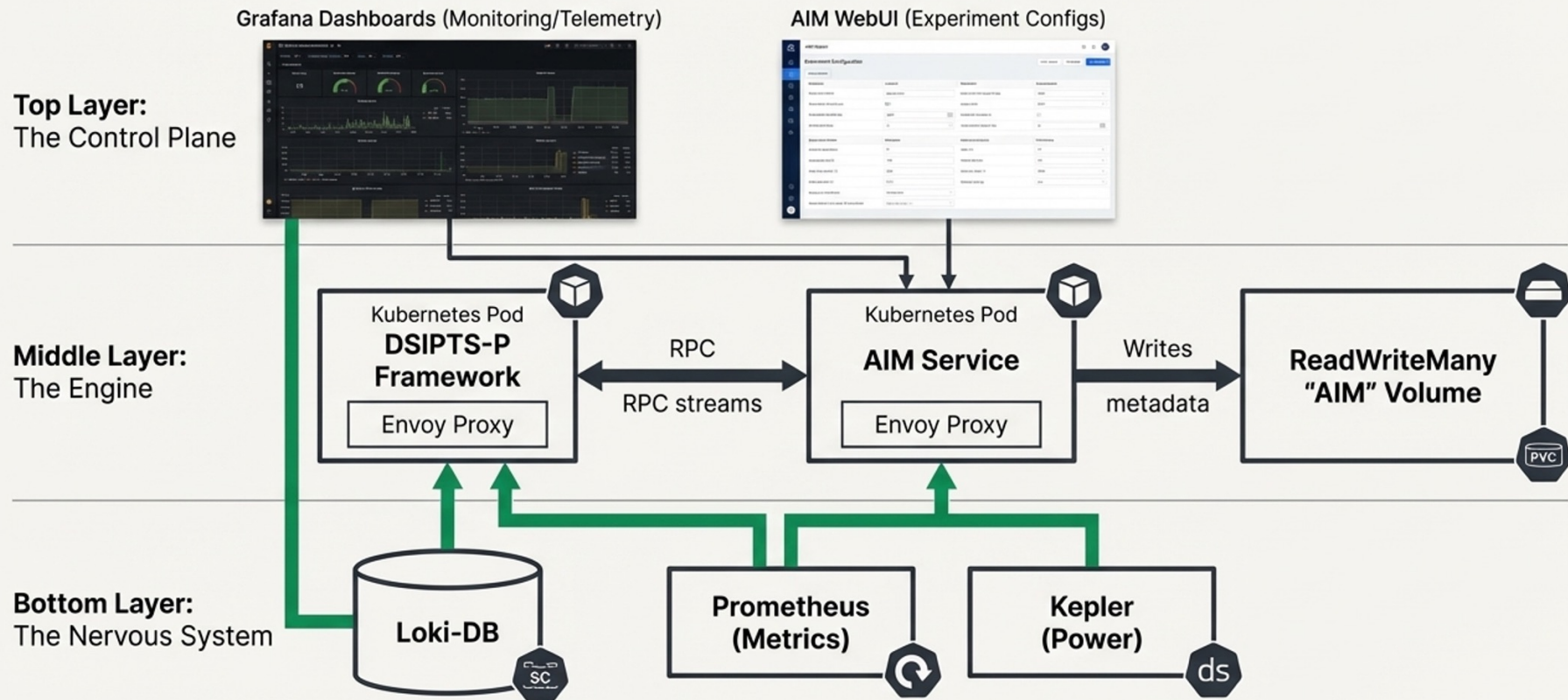


# ML Model Development Automation



Whole Pipeline automated → Only YAML files/GUI for configuration  
Automated training, versioning, inference and KPI scoring of diverse ML-Models

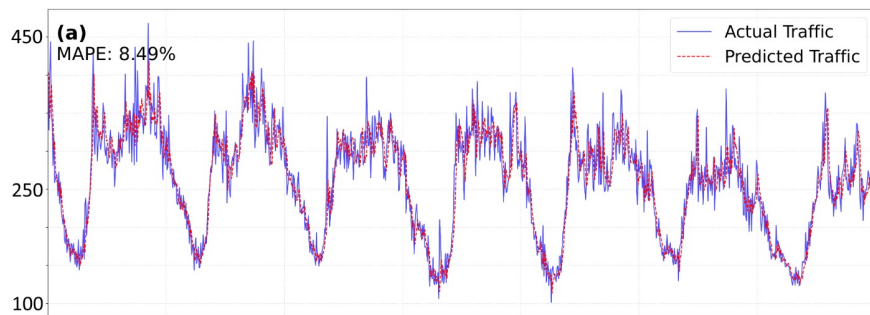
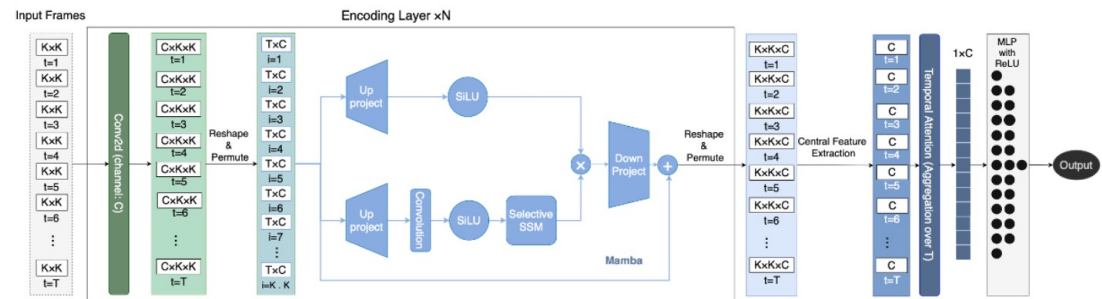
# AI Model Benchmarking Framework Architecture



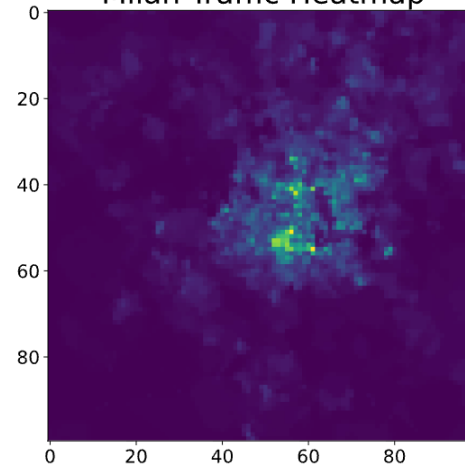
K8S Integration can launch training and inference on different hardware to compare performance vs. energy consumption (through Kepler)

# Example: HiSTM - Complexity vs. Accuracy

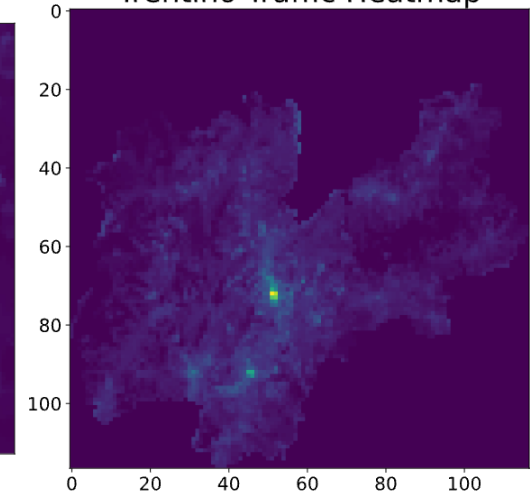
Model	MAE ↓	RMSE ↓	R <sup>2</sup> Score ↑	SSIM ↑
PatchTST	20.231	45.235	0.5635	0.9187
STN	7.3908	16.8824	0.9546	0.9853
VMRNN-B	7.1659	19.0876	0.9420	0.9843
PredRNN++	7.0000	19.1650	0.9425	0.9893
xLSTM	6.4672	15.0901	0.9637	0.9870
VMRNN-D	6.4151	16.3284	0.9575	0.9873
STTRE	5.5558	11.4426	0.9791	0.9917
Autoformer	5.5850	12.2750	0.9762	0.9920
Informer	5.4100	11.4600	0.9792	0.9922
TimesNet	5.3540	11.3270	0.9797	0.9922
HiSTM	<b>5.2196</b>	<b>11.2476</b>	<b>0.9799</b>	<b>0.9925</b>



Milan Traffic Heatmap

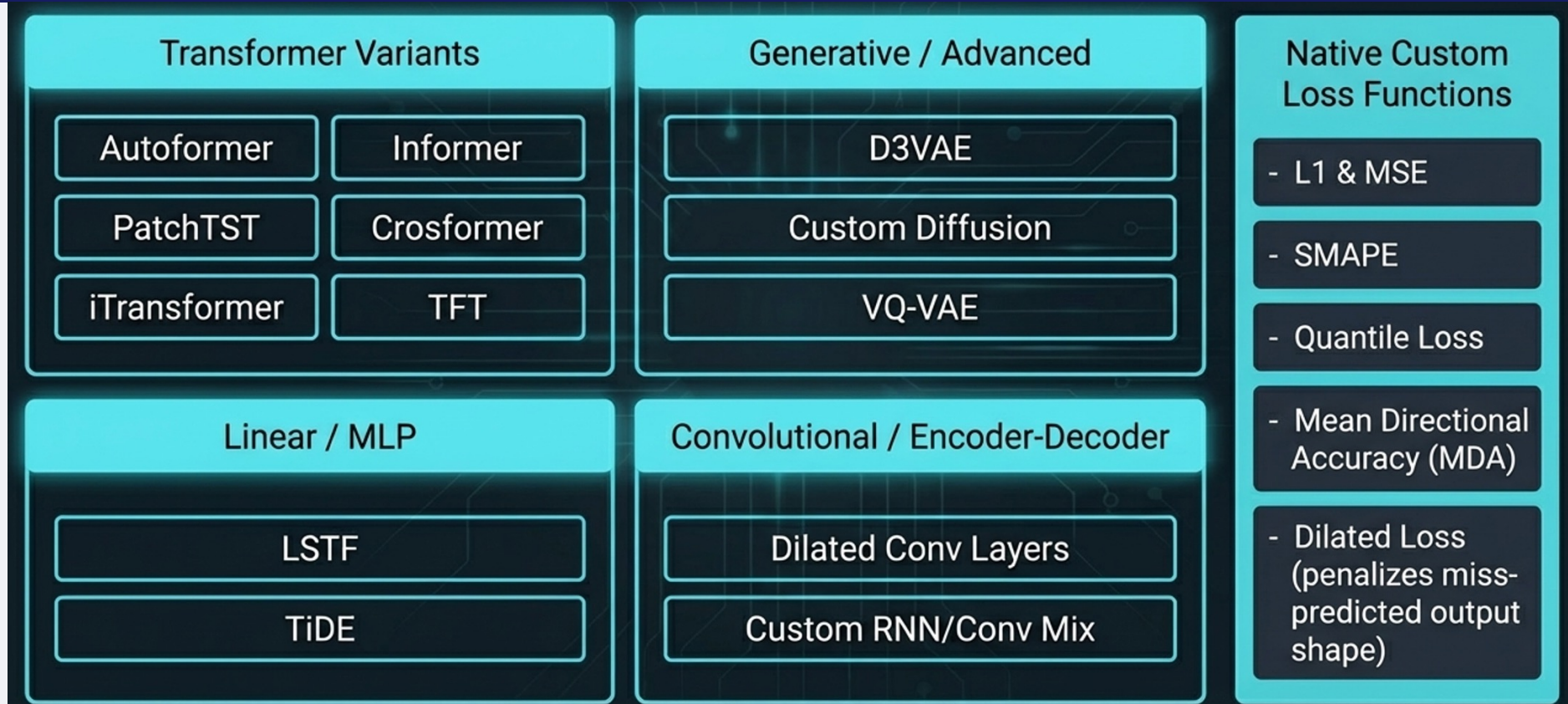


Trentino Traffic Heatmap



## Example Evaluation: Parameter Size vs Accuracy

# Supported Model Architectures (many more...)



Additional support for SplitLearning, Federated Learning, LSTM, xLSTM, Mamba, STNs, Bring your Own Model

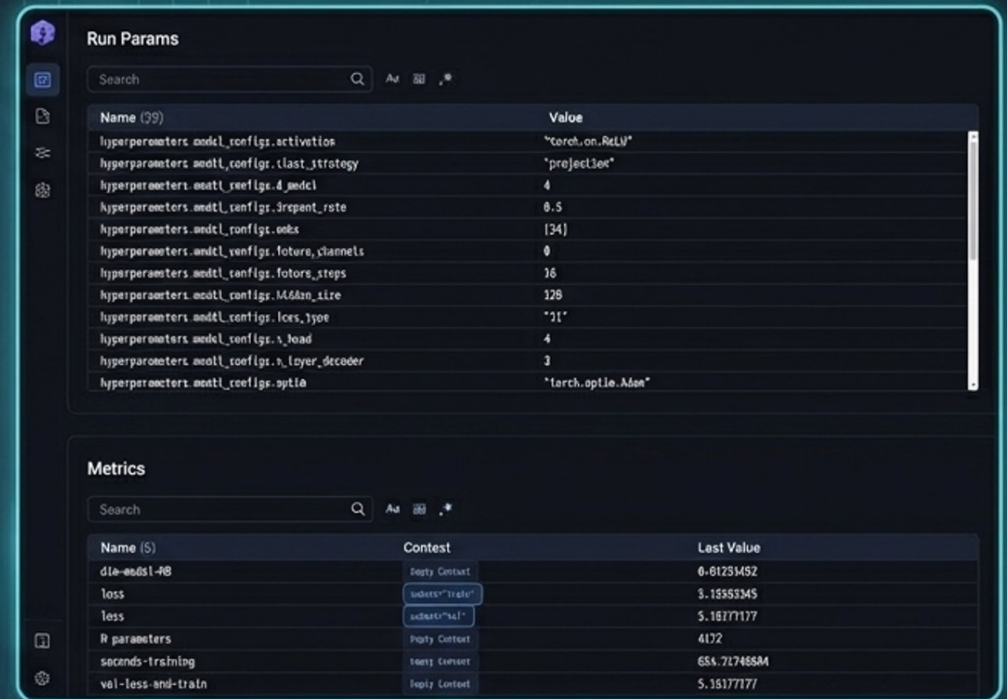
# Massive Parallel Hardware Accel. Training/Inference Jobs



**Single Node:** Standard execution.

**Joblib Back-end:** Local parallel execution across multiple CPU/GPU cores.

**SLURM Cluster:** Custom launcher allocating resources seamlessly without breaking cluster policy rules.



The screenshot shows a dashboard with two main sections: 'Run Params' and 'Metrics'. The 'Run Params' section contains a table with 19 rows of hyperparameters and their values. The 'Metrics' section contains a table with 6 rows of performance metrics and their values.











Name (99)	Value
hyperparameters.amcl_configs.activation	"torch.nn.ReLU"
hyperparameters.amcl_configs.class_strategy	"project3oe"
hyperparameters.amcl_configs.d_amcl	4
hyperparameters.amcl_configs.dropout_rate	0.5
hyperparameters.amcl_configs.epochs	[34]
hyperparameters.amcl_configs.fotore_channels	0
hyperparameters.amcl_configs.fotore_steps	16
hyperparameters.amcl_configs.MdAn_size	128
hyperparameters.amcl_configs.loss_type	"l1"
hyperparameters.amcl_configs.n_load	4
hyperparameters.amcl_configs.n_layer_decoder	3
hyperparameters.amcl_configs.optimizer	"torch.optim.Adam"

Name (5)	Context	Last Value
dlr-eadsl-48	Empty Context	0-0123M52
loss	indices="train"	3.15553M5
loss	indices="val"	5.16177117
R parameters	Empty Context	4172
seconds-training	Empty Context	655.717465M
val-loss-and-train	Empty Context	5.16177117

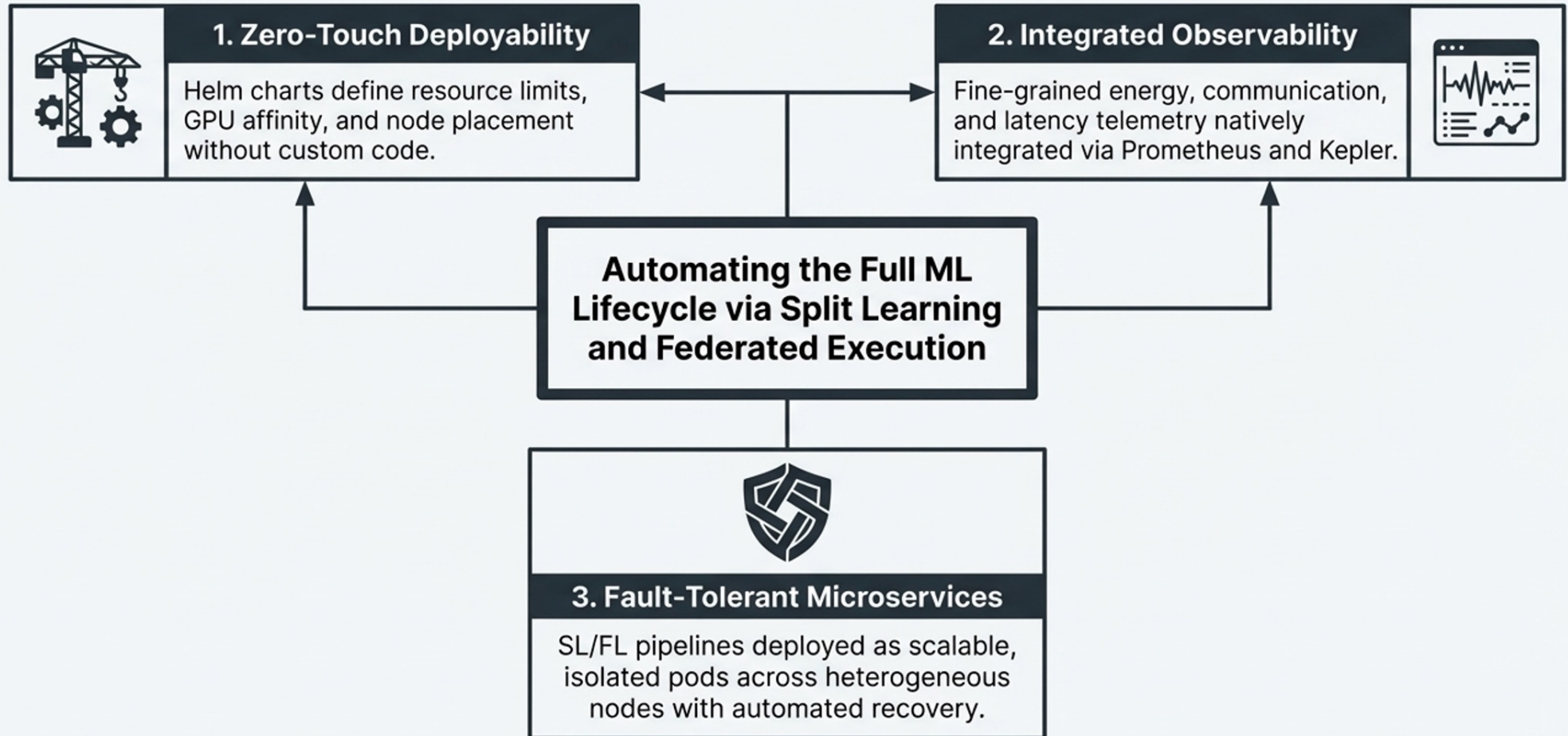
K8S/SLURM Integration: launch AI-model benchmarking jobs(training/inference) for 100s models on different hardware accelerators at scale

# DSIPTS-P Federated and Split-Learning

	Traditional Tools (Kubeflow, MLflow, PyTorch)	DSIPTS-P
Network-aware distributed learning		
GPU/Edge-aware pod placement		
Real-time per-container energy telemetry		
Automated SL round recovery		
Declarative YAML edge-cloud configuration		

Recent extensions natively support Federated Learning and Split Learning (clients and servers are different training pods, connected through K8S network interfaces)

# DSIPTS-P Federated and Split-Learning



Khalid Ali, Phil Aupke, Andreas Kassler: DEMO: Distributed Edge & Split Learning for Energy-Aware Spatiotemporal Forecasting, to appear in IEEE Infocom 2026, Tokyo, May 18, 2026.

# Infocom 2026 Demo

Scenario A

## Split Learning Optimization

Balancing energy draw and end-to-end latency via dynamic model partition points.

Scenario B

## Resilience Under Chaos

Surviving controlled network impairments, packet delays, and catastrophic pod failures.

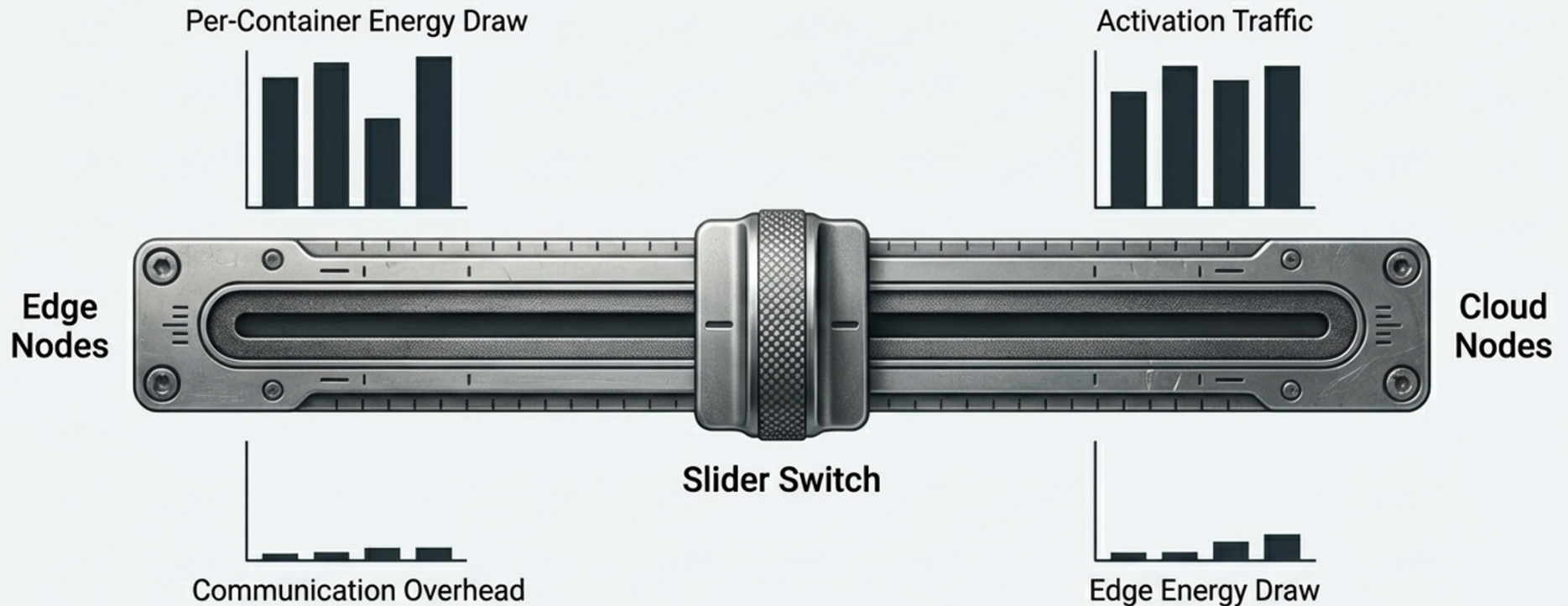
Scenario C

## Hardware Agility

Seamless hot-swapping between CPU edge nodes and GPU cloud clusters with multi-model deployment.

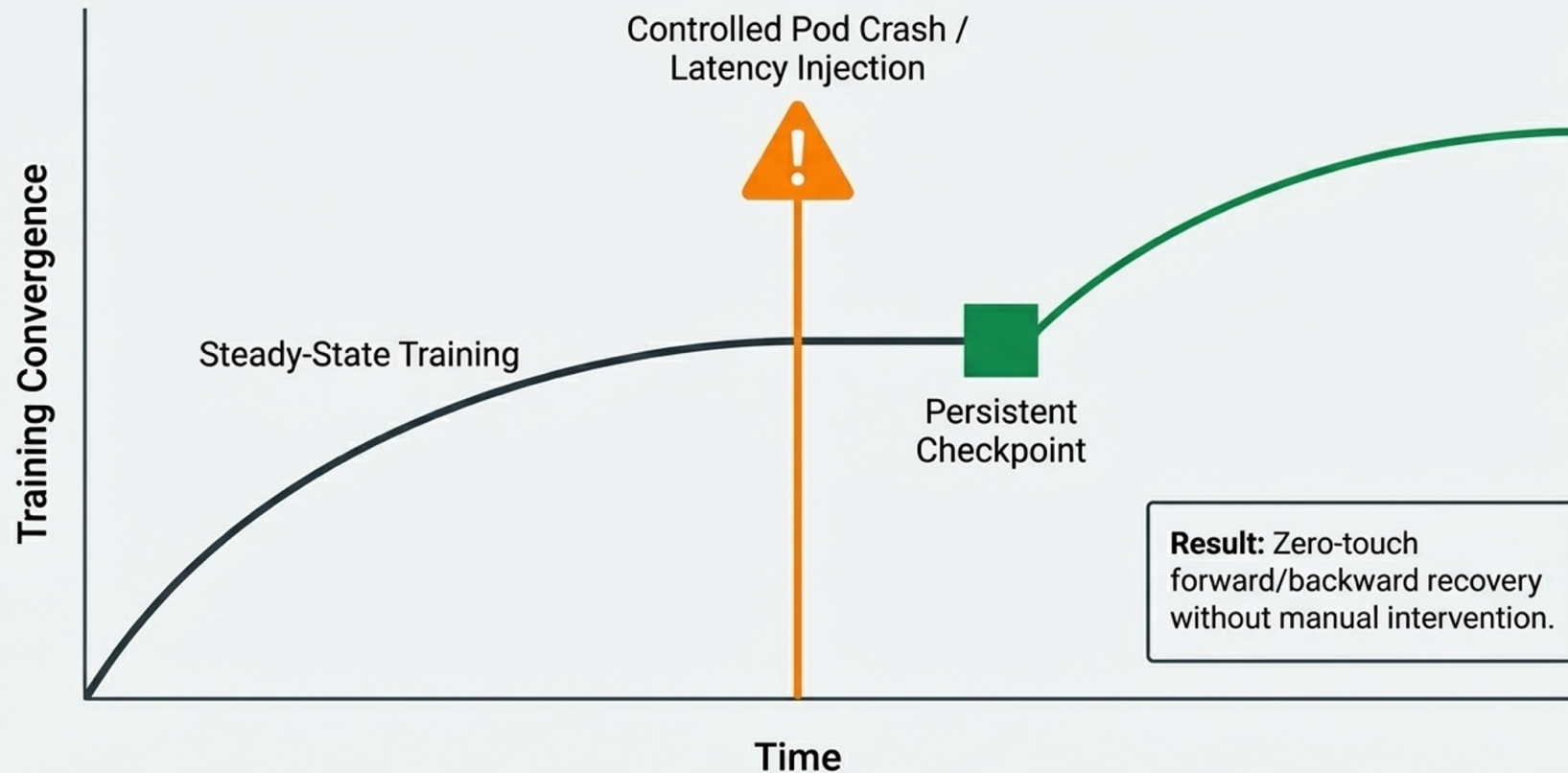
Target Environment: Real-world 5G base station traffic and gNB RB datasets

# Scenario A



Through GUI, change SL partition points to explore tradeoffs between energy draw, communication overhead and accuracy

# Scenario B



Simple YAML file configuration inject faults in inter-pod communication to evaluate impact on convergence and inference time, crash recovery, etc.

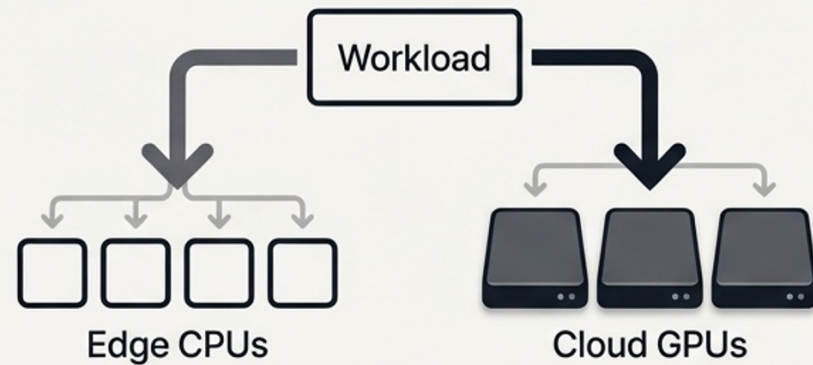
# Scenario C

## Model Swap Interface



Configuration: Identical SL Partitioning

## Heterogeneous Compute Allocation



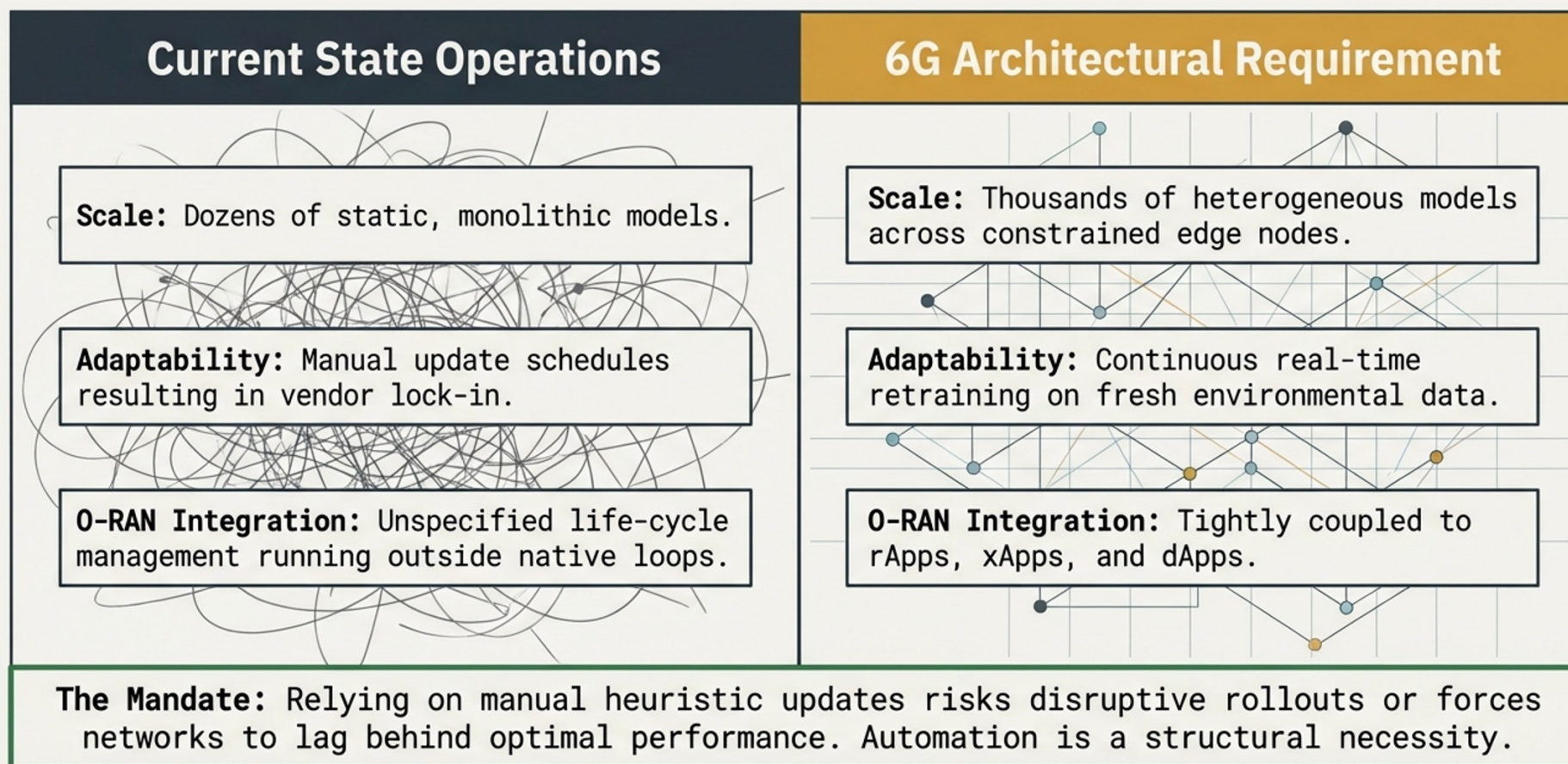
## Real-time Telemetry Impact



Through GUI → compare performance of different ML-models in terms of convergence speed, resource utilization, GPU power, ....

# 6G MLOps Pitfall

Now you found the best model using our Framework. Are we done?



# Why AI-model Update is hard in Telecom?

## Deployment examples

### Example 1

#### Traffic steering xApp

- Learns cell-load imbalance and hands over users **proactively**
- Needs **recalibration** after traffic mix, mobility, or topology changes

### Example 2

#### Energy-saving dApp / PRB slicing policy

- Turns carriers/features on-off or **reslices PRBs** under demand variation
- Needs **versioning** across sites and different gNB hardware



## Dynamically updating AI-Models is difficult

### 1. ML lifecycle complexity

- drift detection and retraining trigger
- hyperparameter search / model selection
- accuracy vs. latency vs. robustness after each new version

### 2. Deployability complexity

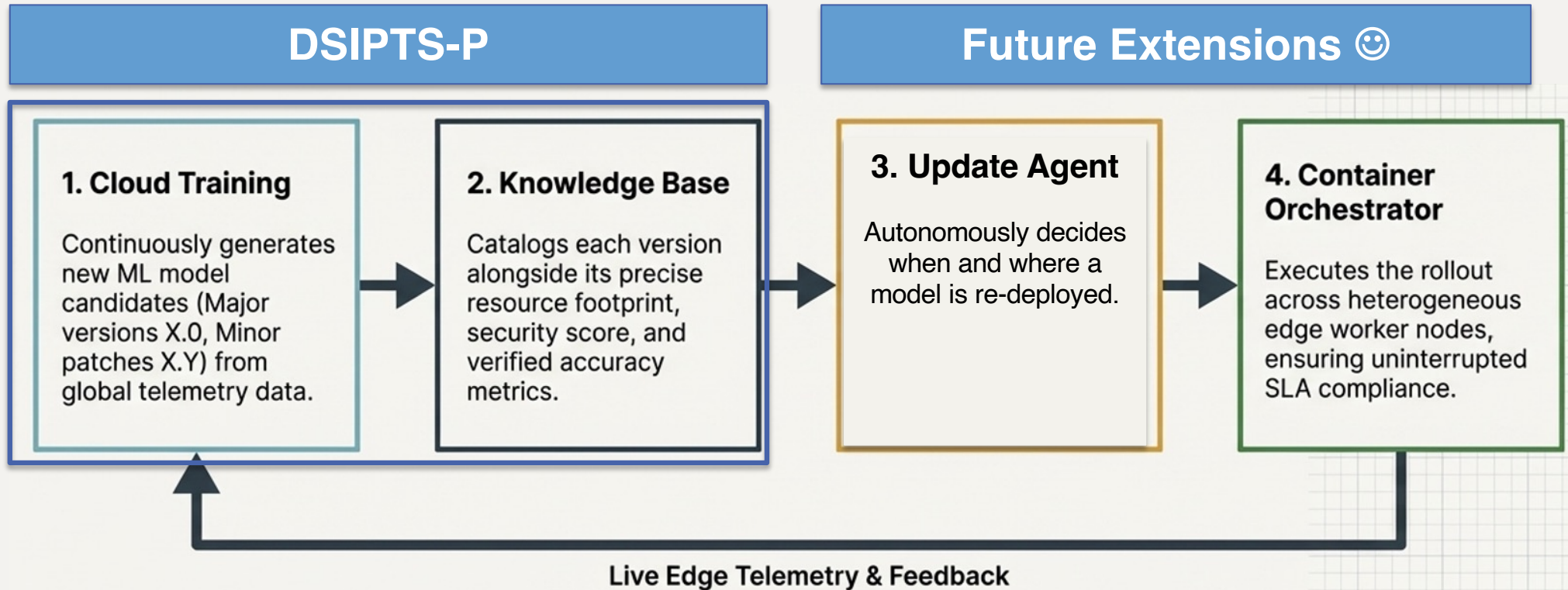
- low-downtime rollout, canary, rollback
- hardware-aware runtime choice (CPU/GPU/NPU, cloud/edge)
- energy-aware placement and inference optimization

### 3. Telecom integration complexity

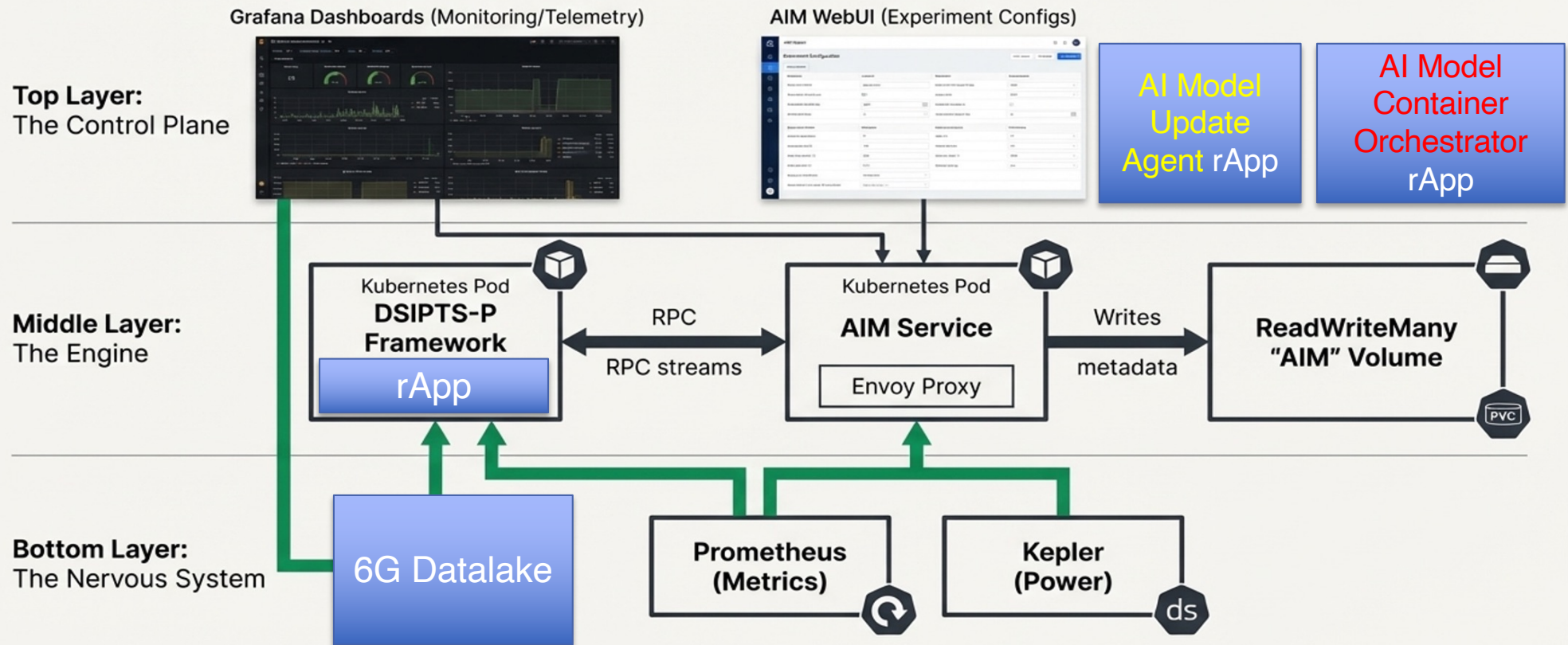
- where should it run: rApp, xApp, or dApp? On which node?
- which telemetry to subscribe to, from which data source / query IDs?
- which API calls close the loop, and what latency / compute budget is tolerated?

**The challenge is not only finding a good or better model - it is updating the right model, in the right place, at the right time with the right interfaces, without breaking the network operation.**

# 6G AI Model Benchmarking as a Service Framework



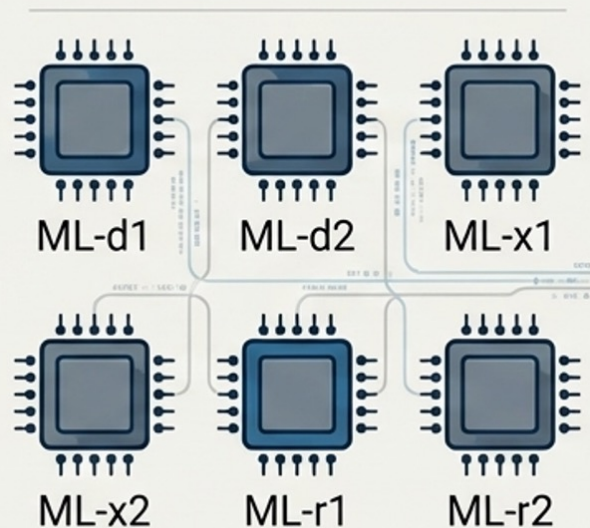
# AI Model Benchmarking Framework Architecture



**AI Model Update agent** decides when which models need updates  
**Container Orchestrator** optimizes placement of AI model, wires it with 6G Telemetry  
Those can be implemented through AI Agents for continuous reasoning

# Example: Model Repository

## The Models



## The Upgrades

- Major Versions:
  - +2% Accuracy | -2% Stability | -7% Service Time
- Minor Versions:
  - +0.1% Accuracy | -0.1% Stability

**Update Agent:** Decide when and where to update which AI-Apps: (1) Always Update (2) Never Update (3) Random (4) Server Load-based (5) Drift Detection Triggered

# Future Research Directions

## ML / MLOps research

### Model quality optimization over time

- drift detection, active retraining
- HPO / NAS / model selection
- compression and inference optimization
- benchmarking frameworks and experiment tracking

## Systems / cloud research

### Reliable and efficient deployment

- K8s orchestration, canaries, rollback
- hardware-aware scheduling and compilation
- Multi-KPI aware placement / autoscaling (energy, accuracy, latency)
- observability, fault tolerance, policy control

## Telecom / O-RAN research

### Network-specific control integration

- rApp/xApp/dApp placement and functional split
- telemetry subscriptions and data pipelines
- E2/Near-RT/Non-RT control APIs and loop timing
- per-use-case latency, compute, and SLA budgets

Existing solutions usually optimize one space or a pair of spaces  
Telecom-grade continuous AI-model updating needs all three at once

# Summary

- Telco AI cloud needs AI Infra, AI Agents, MaaS, Digital Twin
- Finding a good model requires benchmarking at scale → HPC, K8s, AI Infra
- We provide a framework for automating AI model benchmarking and versioning at scale (incl. Split- and Federated Learning)
  - It can also be exposed for customers/users who want to train their own AI Algorithms, not necessarily related to Telco → Stock prices, Solar Power, ...
- MLOps principles need to be integrated into 6G architecture for continuously updating to the best AI models
  - MLOps and Telco need to be integrated into coherent E2E framework
  - Can be done by AI Agent connected to dApps for model benchmarking, versioning, automatic deployment orchestration HPC required for massive loads of 1000 models retraining continuously

# Thank You!

▪ Have Questions? → [kassler@ieee.org](mailto:kassler@ieee.org)