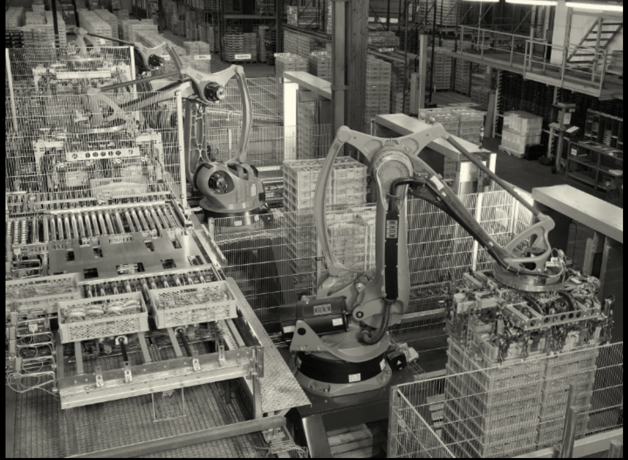
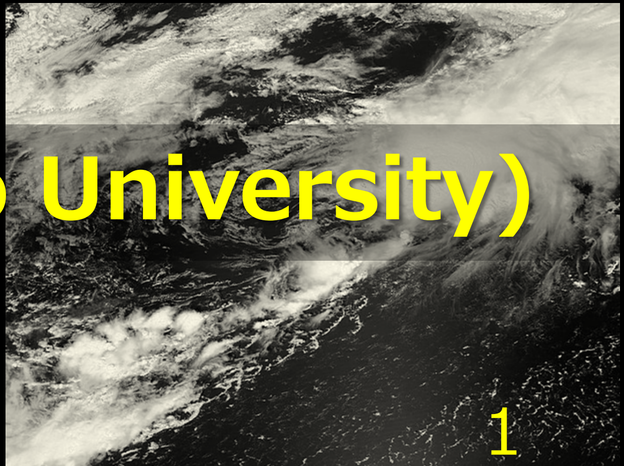



A Lightweight Concept Drift Detection Method for On-Device Learning on Resource-Limited Edge Devices



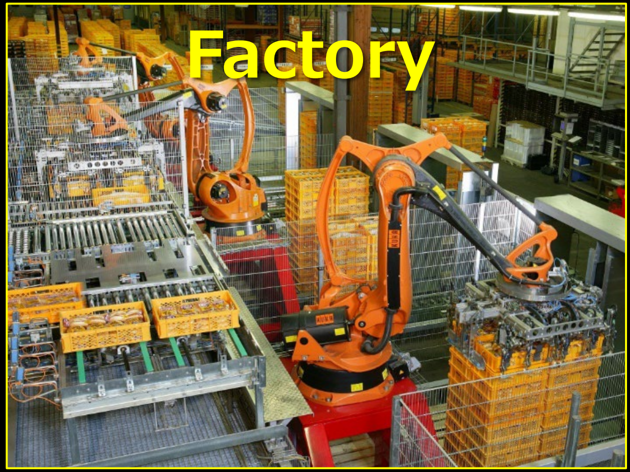


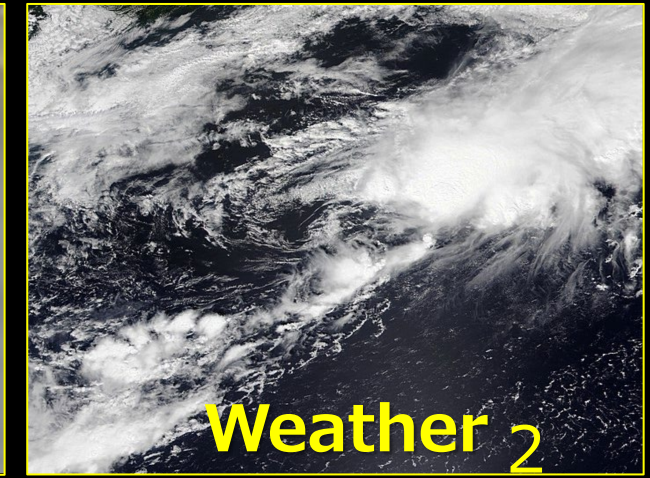
Takeya Yamada & Hiroki Matsutani (Keio University)

IoT: Applications

- ML application in real fields (e.g., anomaly detection)

  Factory, monitoring, robot, safety, security, surveillance, …

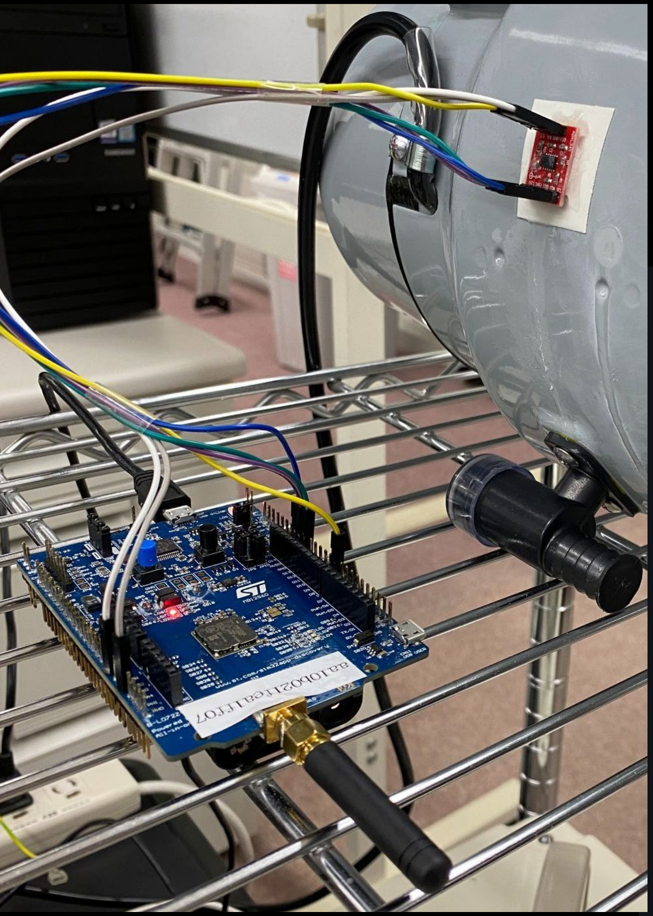


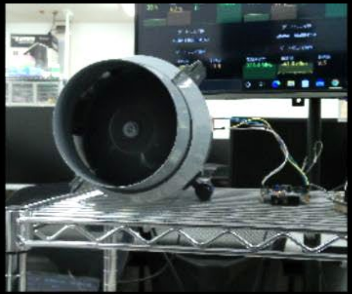


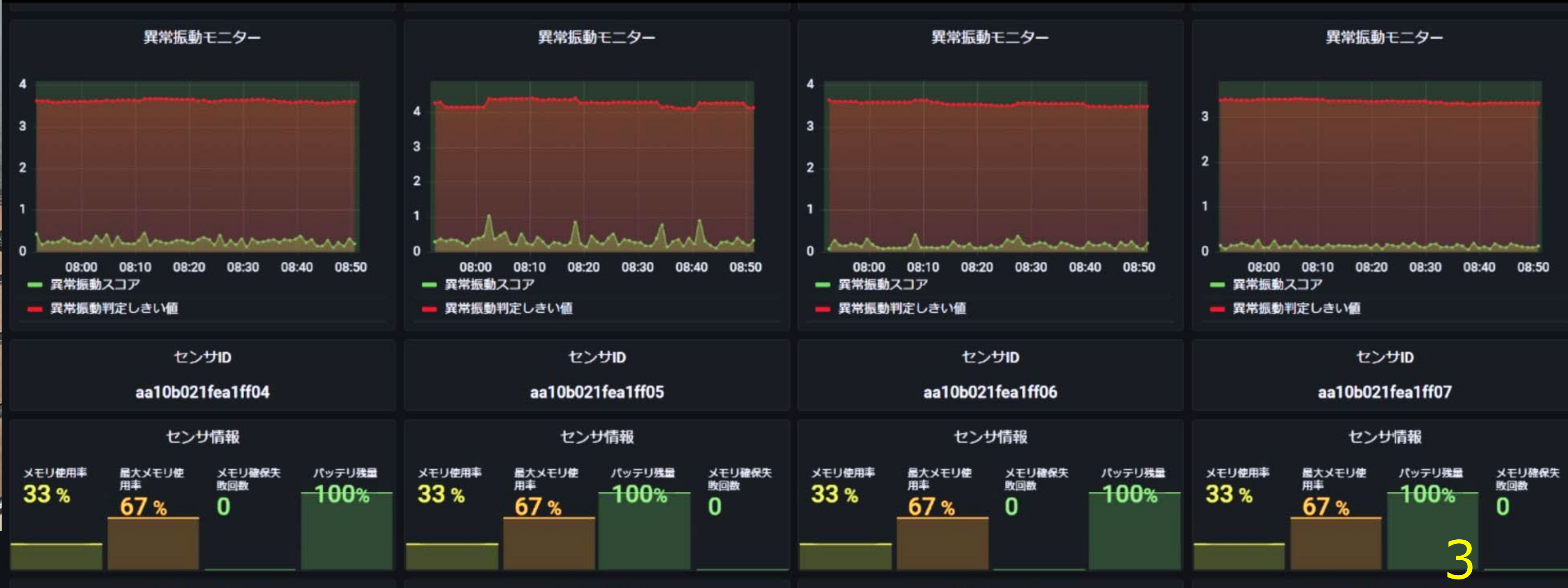
Edge AI: Equipment monitoring

- **Monitoring of air-conditioning systems (e.g., fans)**

  Using wireless sensor nodes that can train and predict

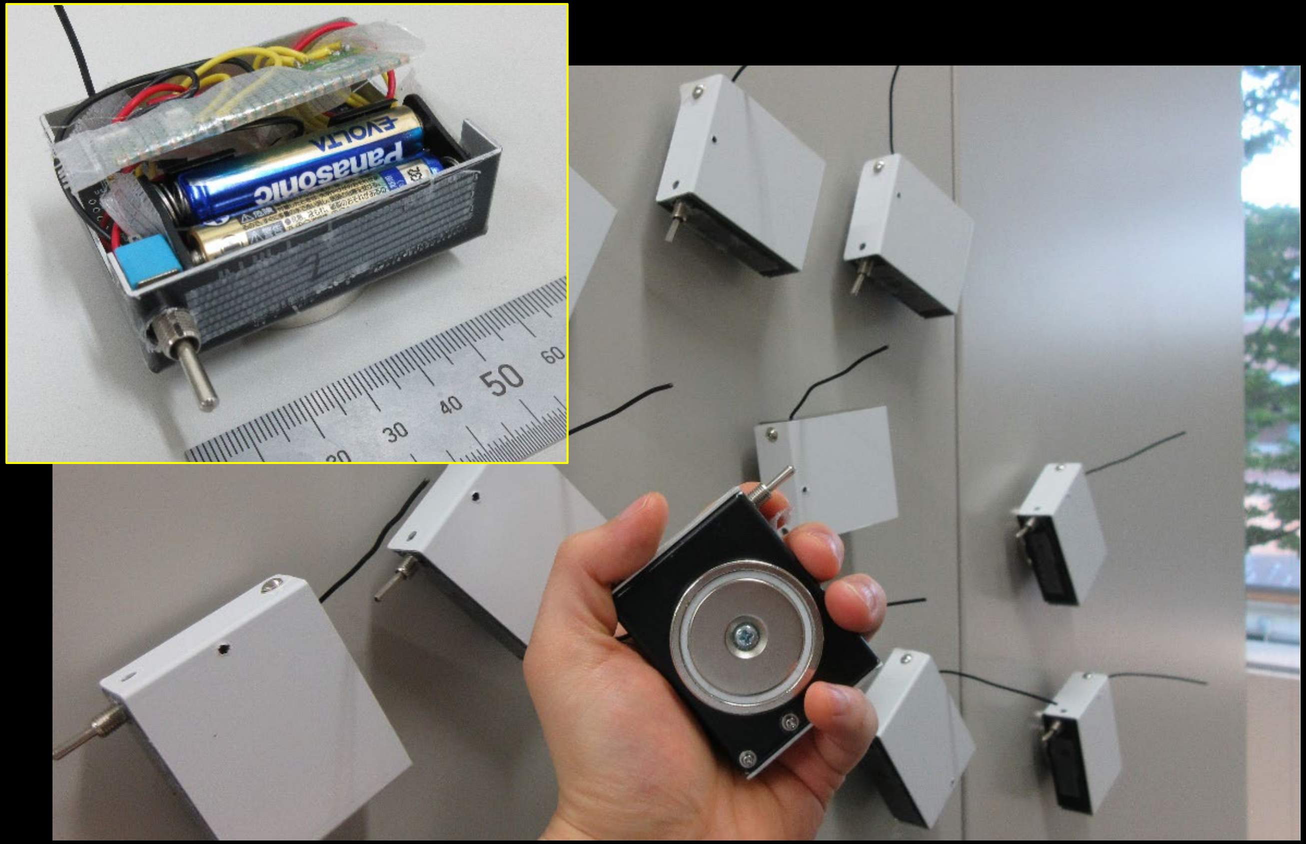






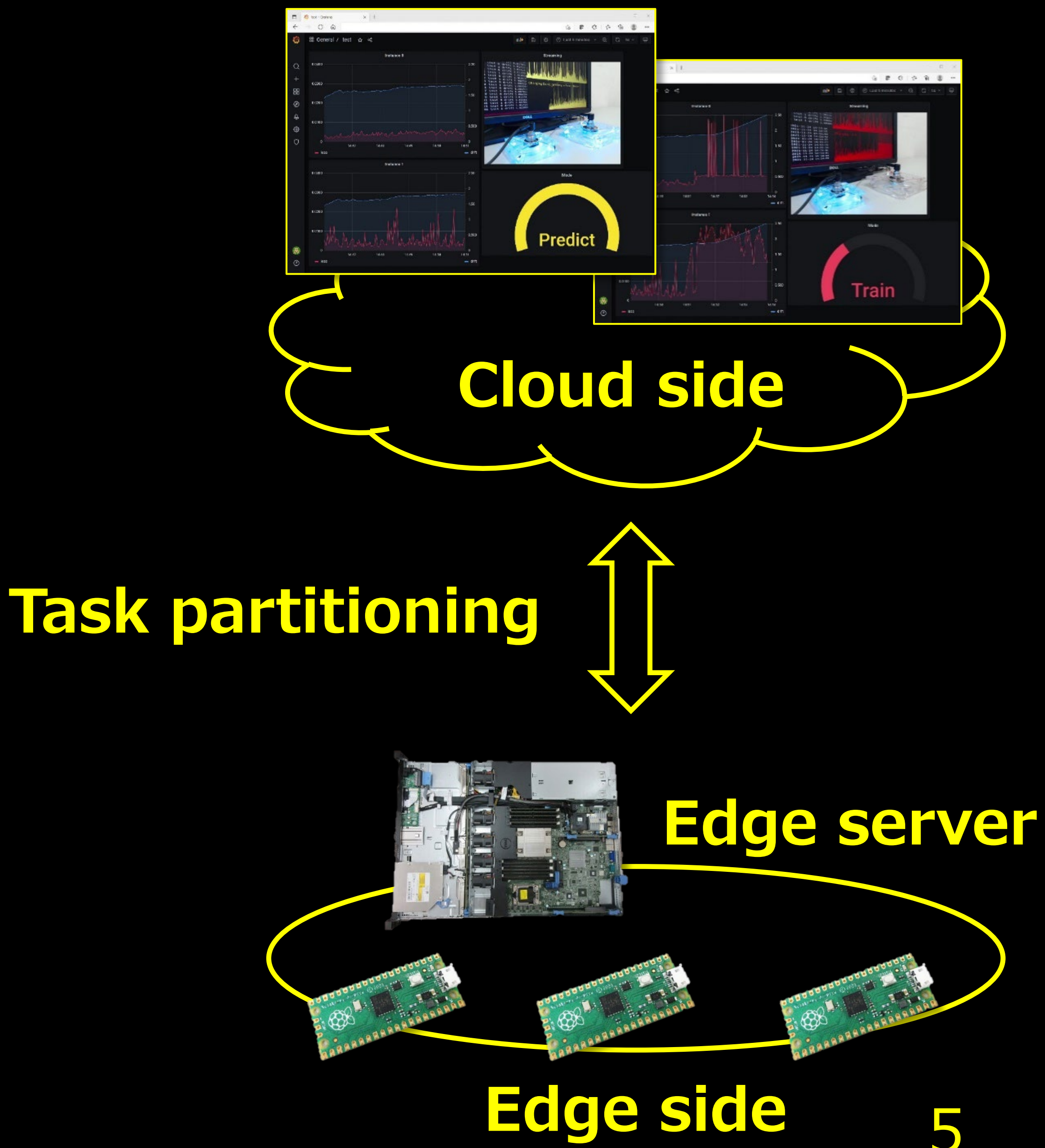
Edge AI: Equipment monitoring

- **Wireless sensor nodes that can train and predict [1]**
  - Raspberry Pi Pico, sensors, magnet, battery, LoRa module
  - On-device learning of neural networks

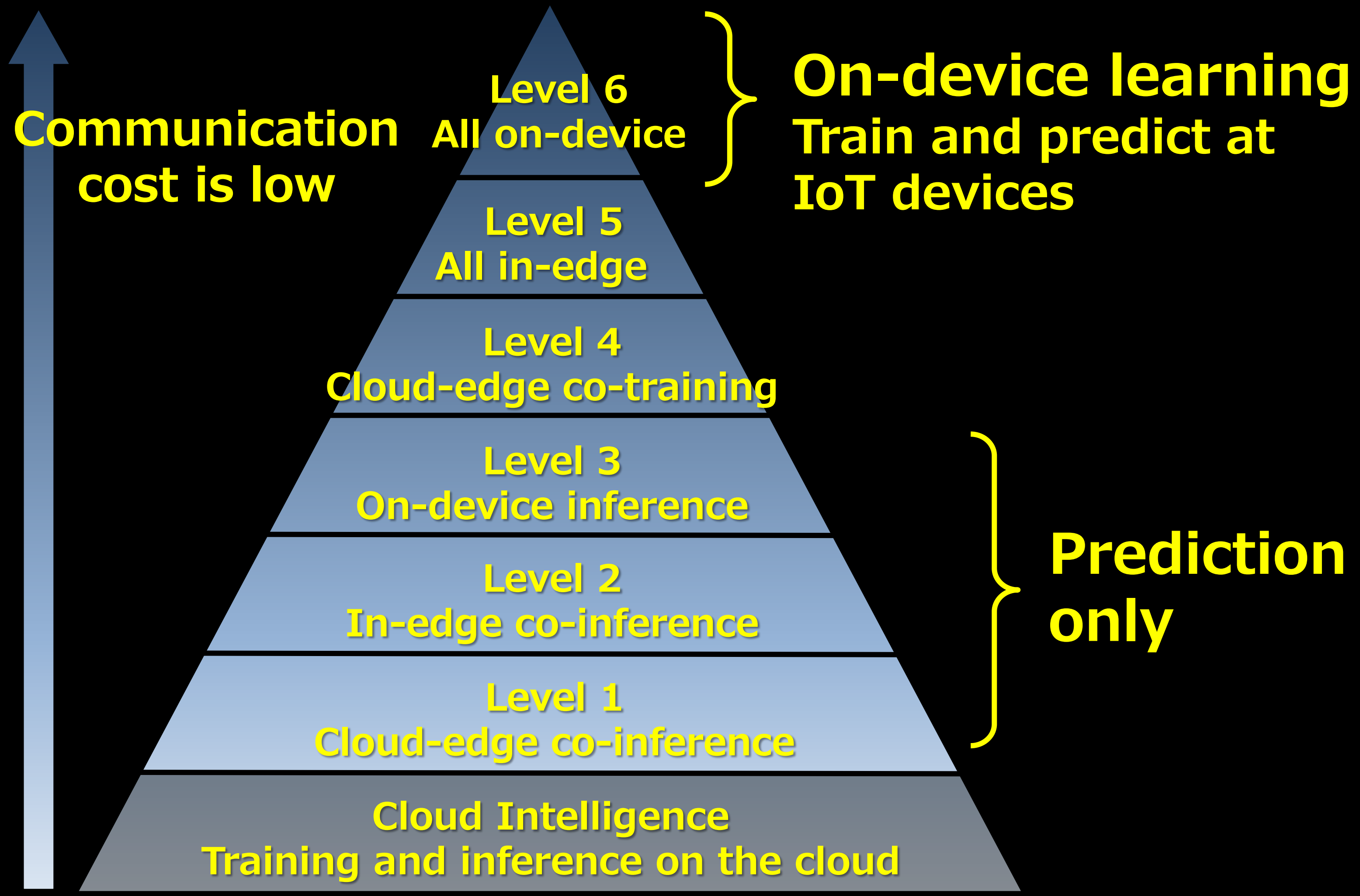


[1] Hiroki Matsutani et al., "On-Device Learning: A Neural Network Based Field-Trainable Edge AI", arXiv:2203.01077 (2022).

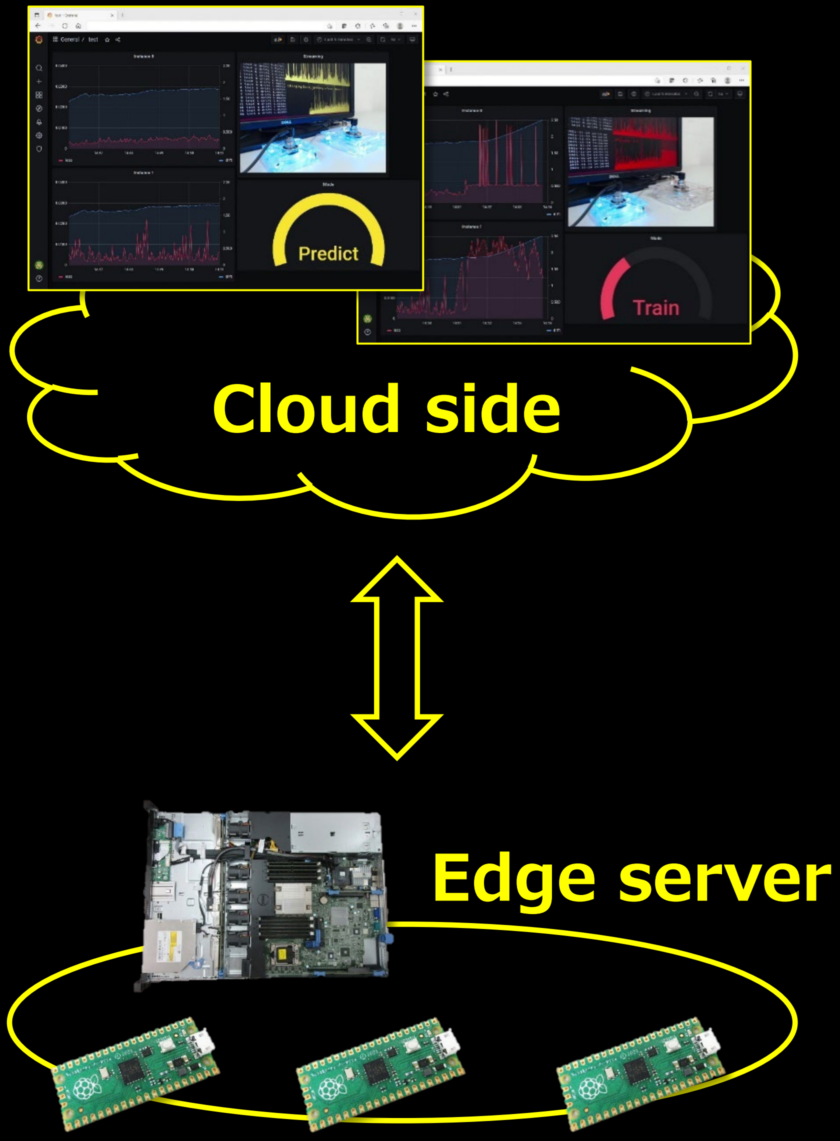
Edge AI: Classification



Edge AI: Classification



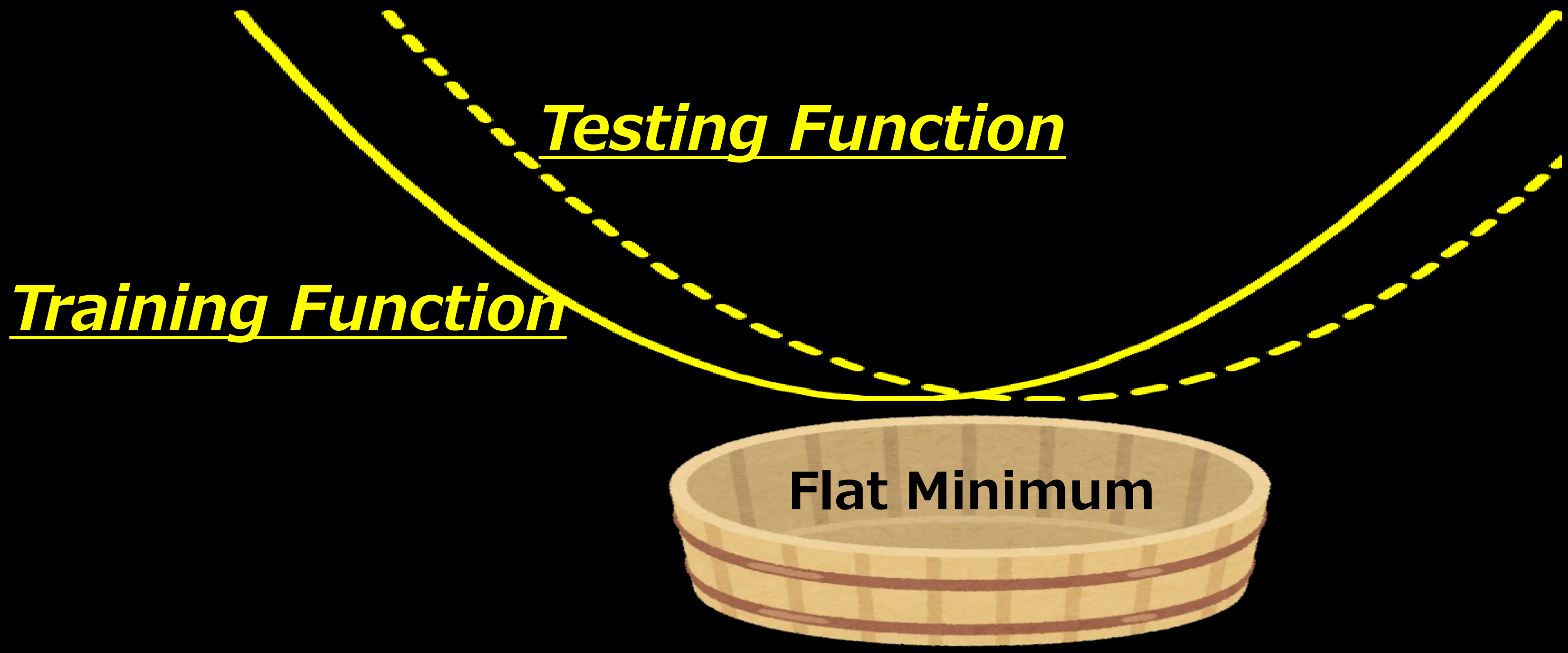
Six-level rating for edge intelligence [1]



[1] Z. Zhou et al., "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing", Proceedings of the IEEE (2019).

On-device learning: Motivation

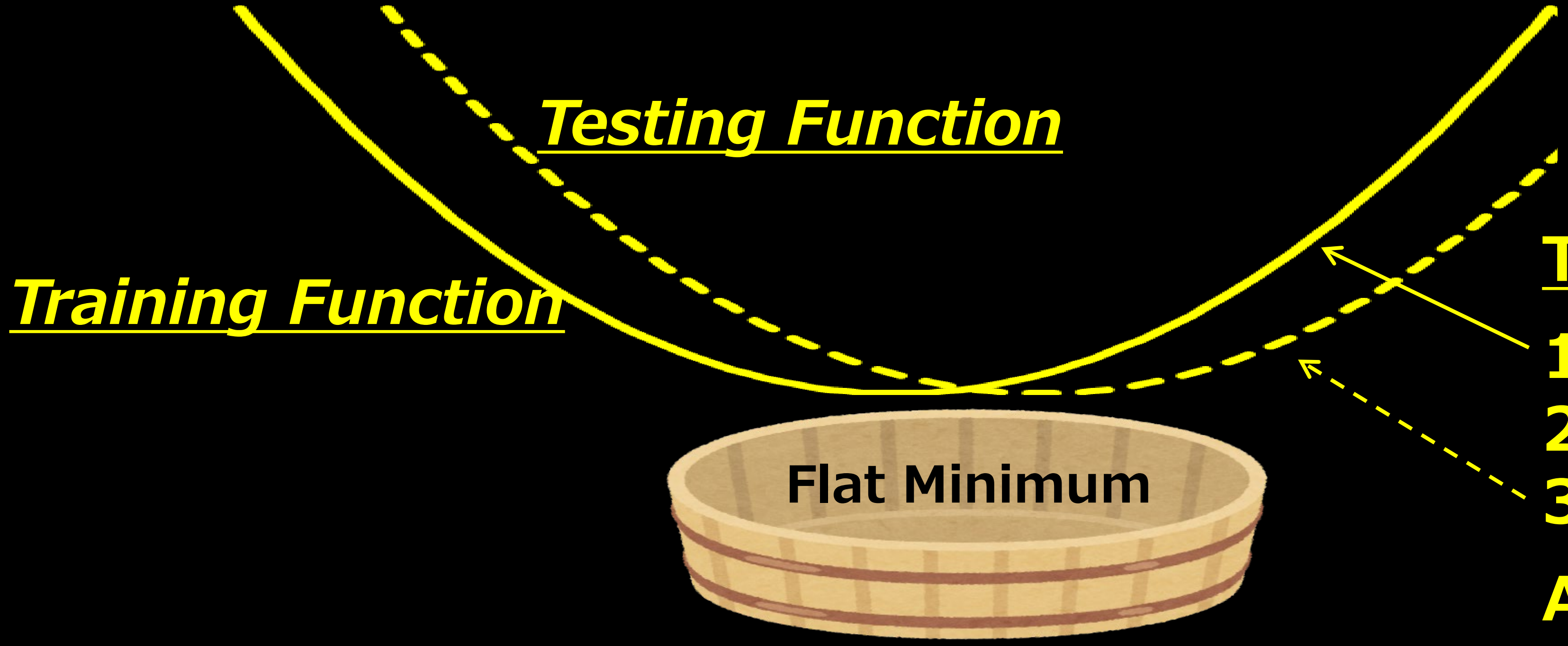
- Challenges of edge AI: Addressing the gap between training data and deployed environment



Typical solution
✓ Generalization capability to absorb the gap

On-device learning: Motivation

- **Challenges of edge AI: Addressing the gap between training data and deployed environment**



Typical edge AI use case

1. **Collect train data**
2. **Train at server**
3. **Predict at edge**

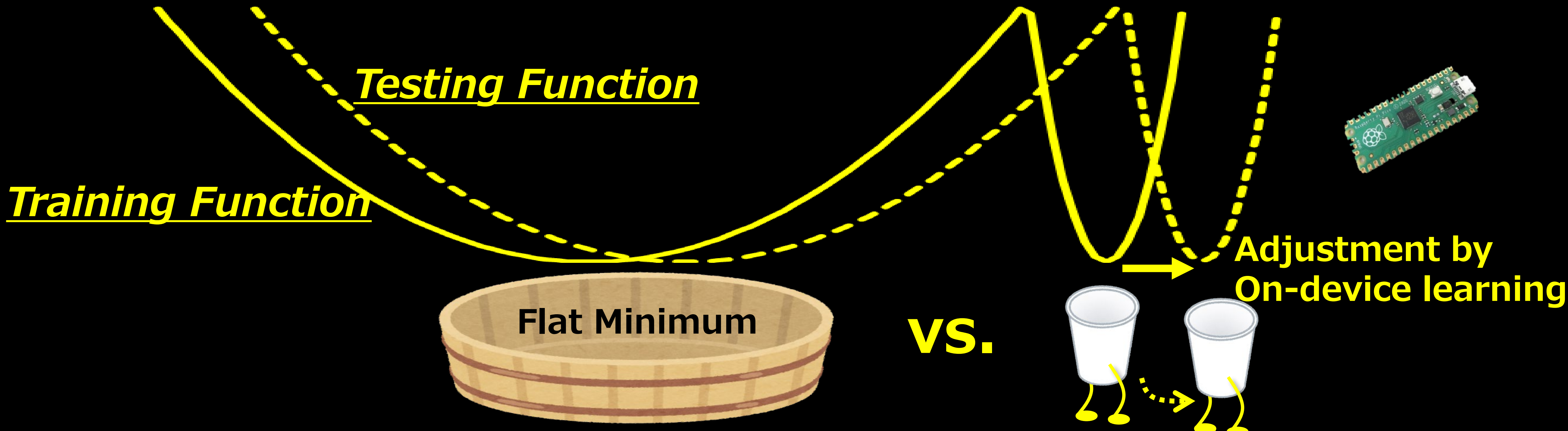
At different environments

Typical solution

✓ **Generalization capability to absorb the gap**

On-device learning: Motivation

- Challenges of edge AI: Addressing the gap between training data and deployed environment at low-cost



Typical solution
- ✓ Generalization capability to absorb the gap

Our approach [1]
- ✓ Small neural networks
- ✓ Train at deployed environment

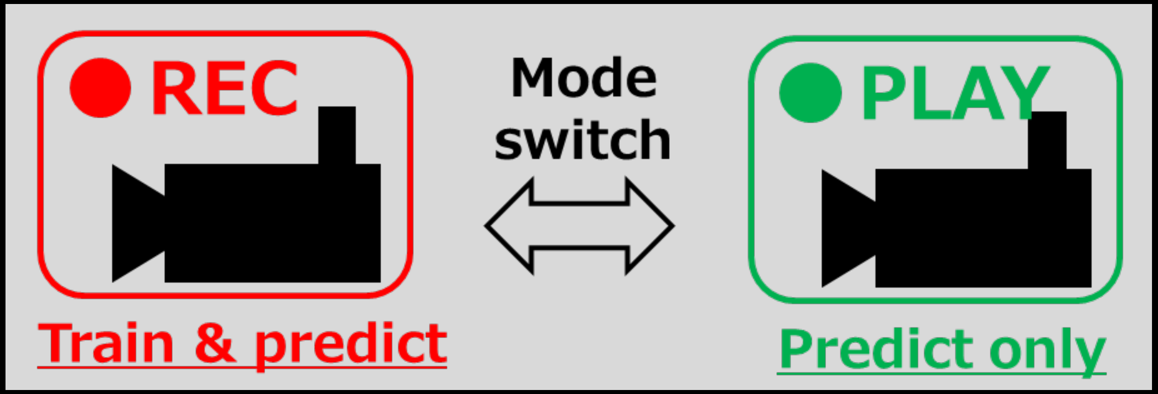
[1] Mineto Tsukada et al., "A Neural Network-Based On-device Learning Anomaly Detector for Edge Devices", IEEE Trans. on Computers (2020).

On-device learning: Two modes

1. Train mode

2. Predict-only mode

Question: ***How and when is the mode changed?***

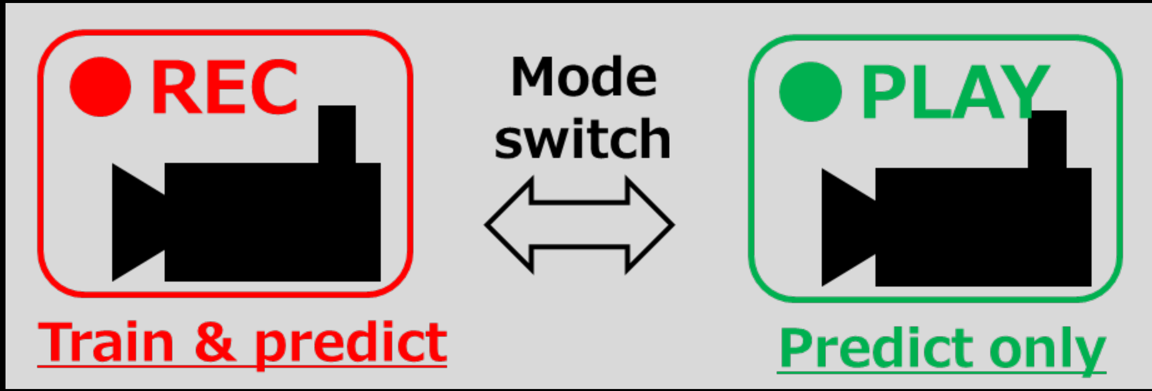


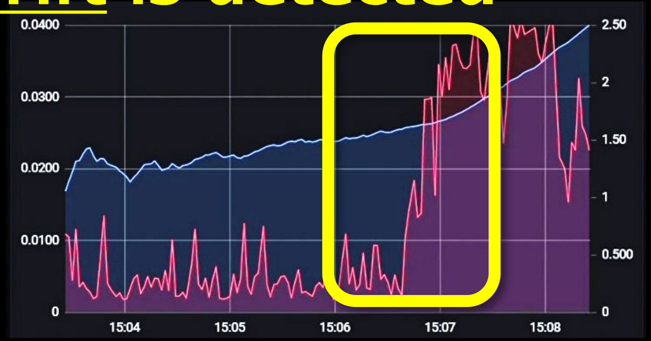
[1] Hiroki Matsutani et al., "On-Device Learning: A Neural Network Based Field-Trainable Edge AI", arXiv:2203.01077 (2022).

On-device learning: Trigger to retrain

1. Manual retraining	2. Automatic retraining
Field-engineers can train edge AI whenever they want	Automatically trained when concept drift is detected

Triggered by train button





Triggered by concept drift

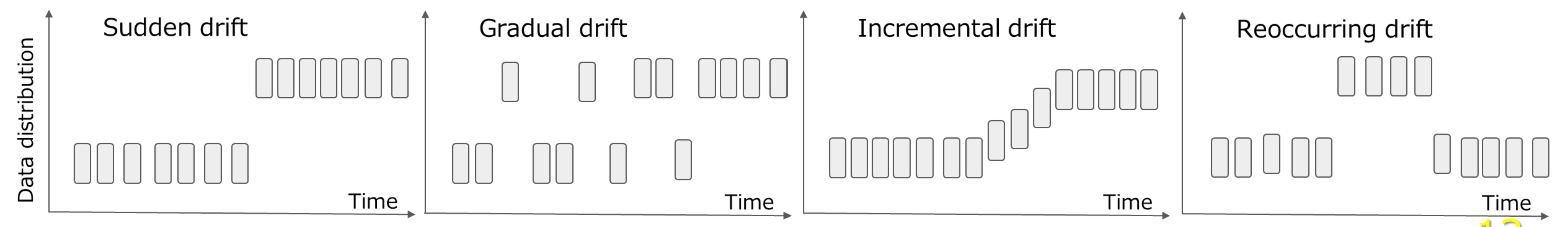
A lightweight concept drift detection for automatic retraining

[1] Hiroki Matsutani et al., "On-Device Learning: A Neural Network Based Field-Trainable Edge AI", arXiv:2203.01077 (2022).

On-device learning: Trigger to retrain

- **Concept drift**

  Phenomenon where statistical properties of target data change over time



[1] Jie Lu et al., "Learning under Concept Drift: A Review", IEEE Trans. on Knowledge and Data Engineering (2019).

On-device learning: Trigger to retrain

- **Concept drift**

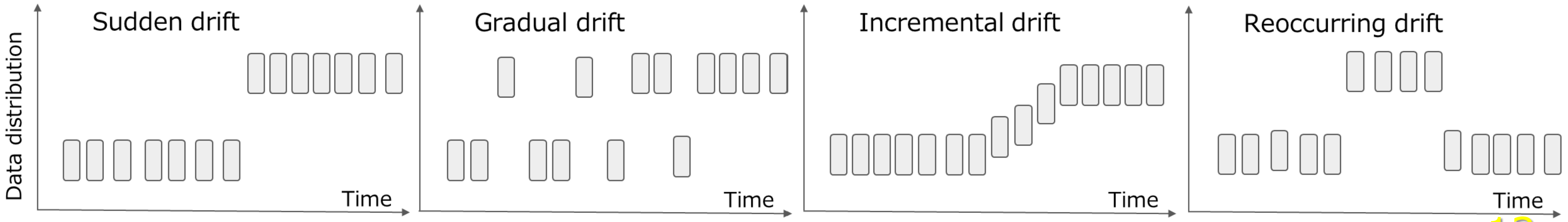
  Phenomenon where statistical properties of target data change over time

Old data distribution before drift does not appear

Old data distribution is gradually replaced with new one

Data distribution is incrementally shifted from old one to new one

Old data distribution reoccurs after the data distribution has been changed

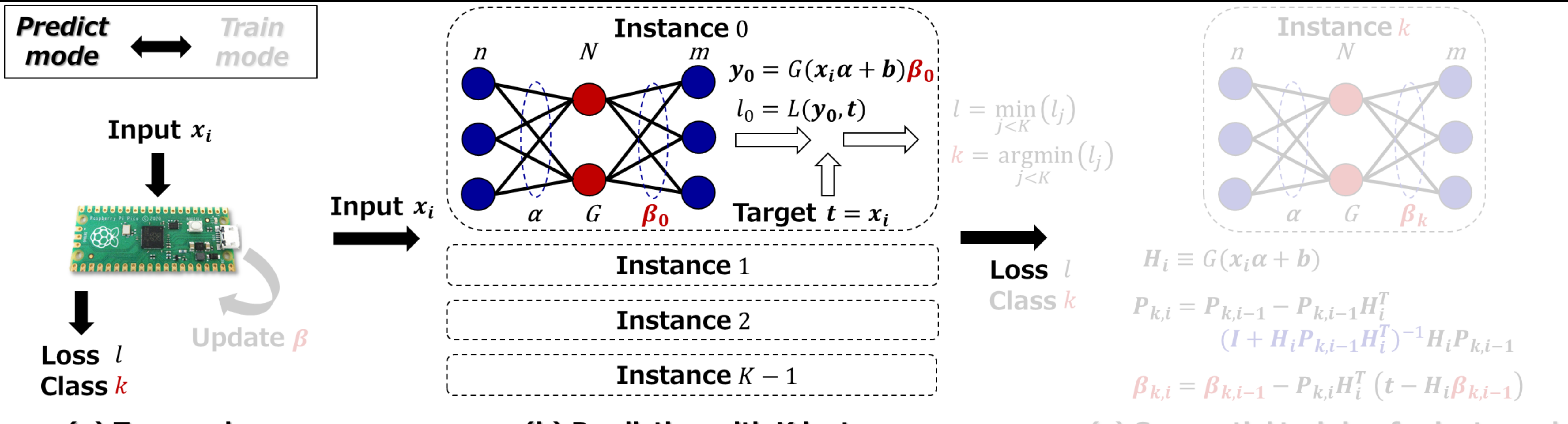


[1] Jie Lu et al., "Learning under Concept Drift: A Review", IEEE Trans. on Knowledge and Data Engineering (2019).

On-device learning: Prediction

- **Prediction is done by *K* autoencoder instances, each of which is specialized to each class**

  Input: *n*-dimensional data, Output: Loss *l* and class *k*

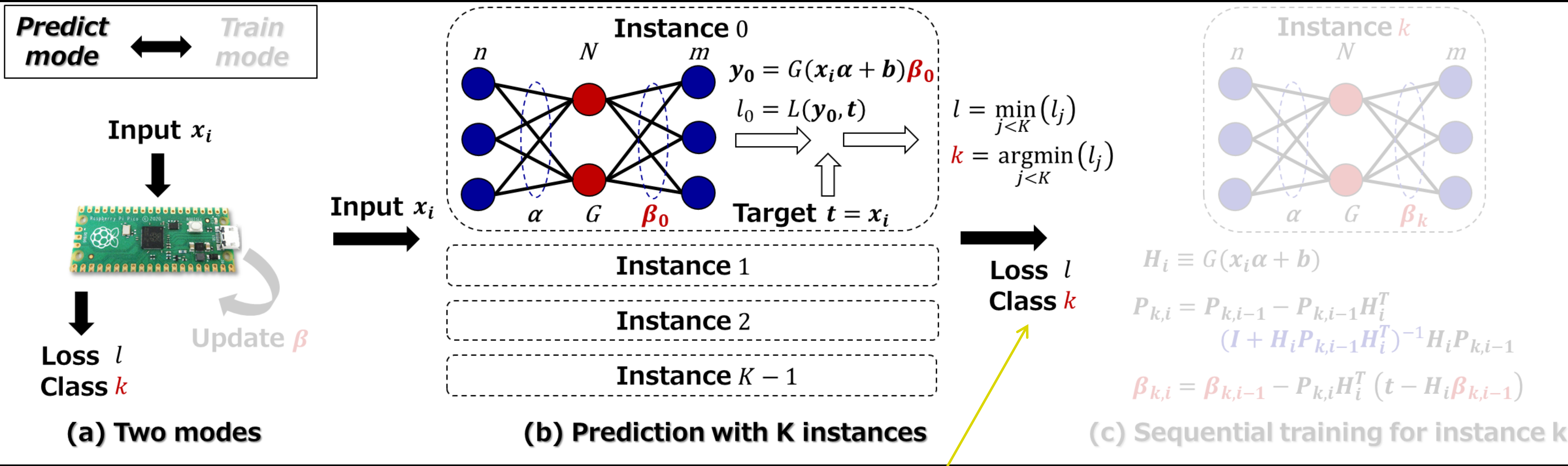


(a) Two modes (b) Prediction with K instances (c) Sequential training for instance k

On-device learning: Prediction

- **Prediction is done by K autoencoder instances, each of which is specialized to each class**

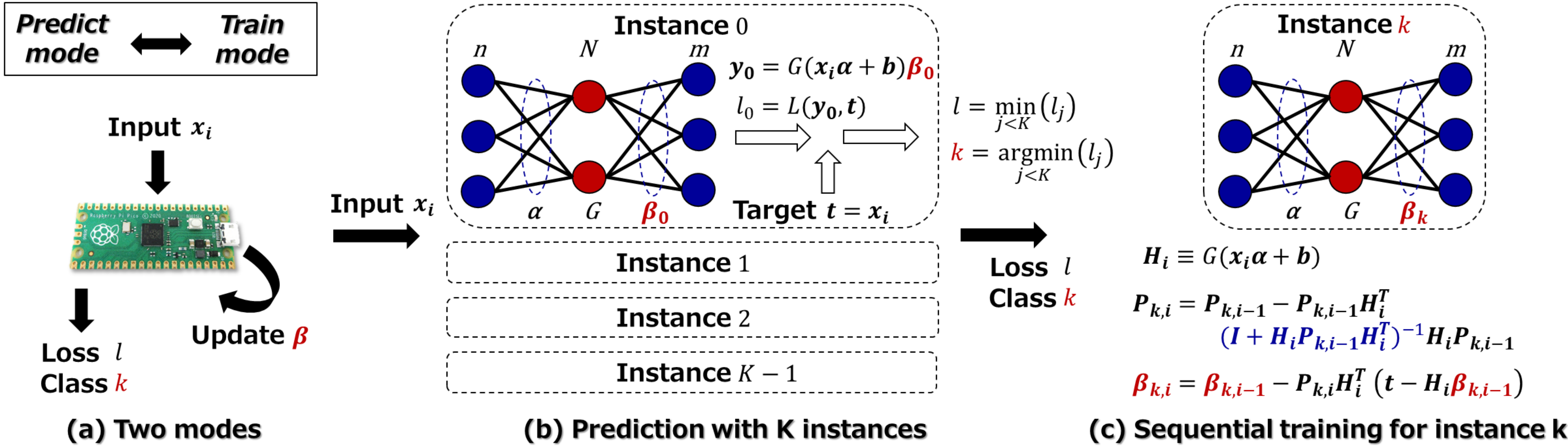
Input: n-dimensional data, Output: Loss l and class k



Instance with the smallest loss value is “the closest” instance or class

On-device learning: Sequential training

- "The closest instance" is updated with the input data
  - OS-ELM [1] is used as sequential training algorithm
  - Weight parameter β is sequentially updated w/ input data x



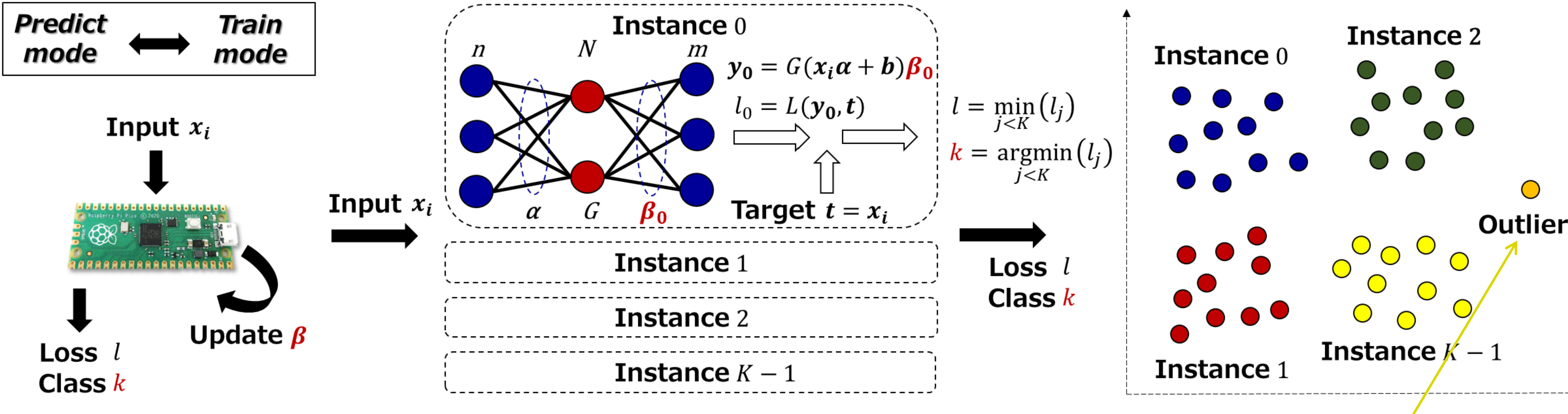
(a) Two modes (b) Prediction with K instances (c) Sequential training for instance k

[1] N. Y. Liang, G. B. Huang, P. Saratchandran, N. Sundararajan, "A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks", IEEE Trans. on Neural Networks, vol. 17, no. 6, pp. 1411-1423, Nov. 2006.

On-device learning: Sequential training

- "The closest instance" is updated with the input data

  By repeating the sequential training of incoming data, each autoencoder is trained to be specialized to each class

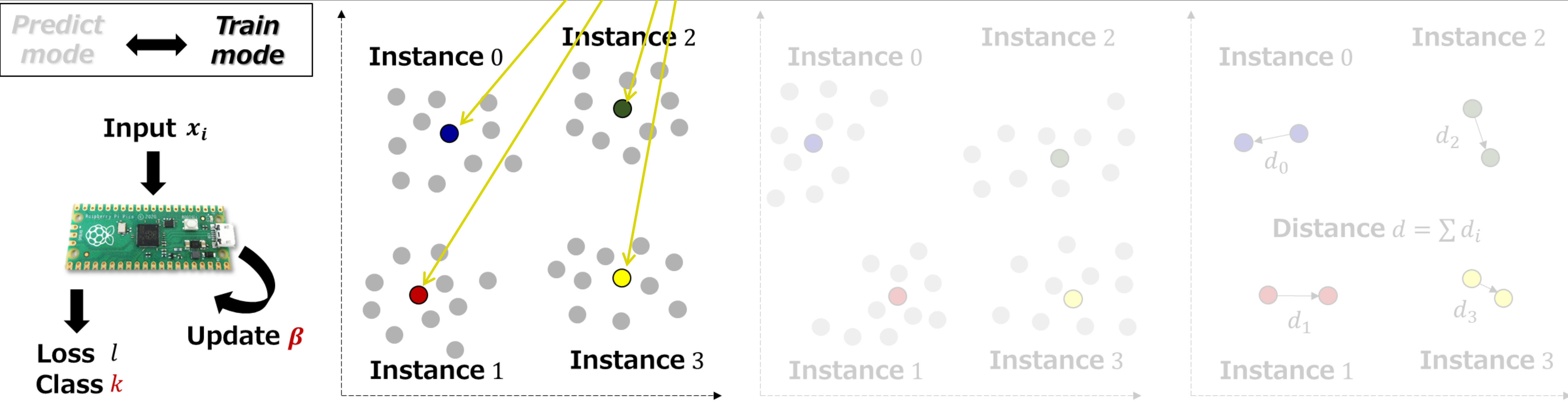


(a) Two modes (b) Prediction with K instances (c) K instances specialized for K patterns

An input data is detected as anomaly if all the instances detect it as anomaly

Concept drift detection algorithm

- Train time: Trained centroids sequentially updated

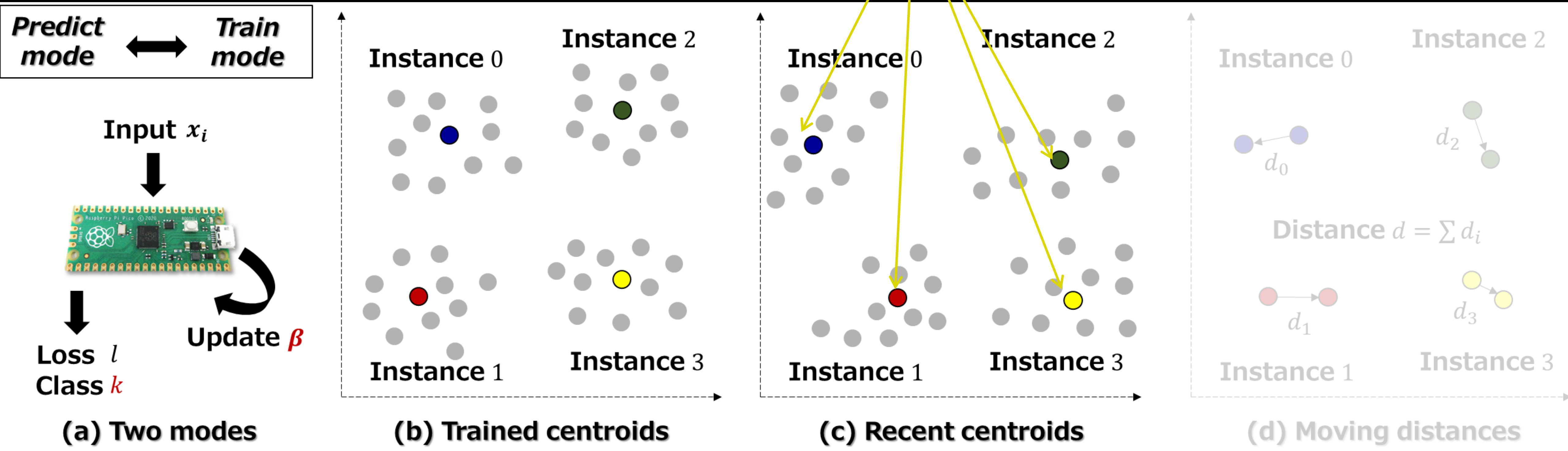


(a) Two modes (b) Trained centroids (c) Recent centroids (d) Moving distances

Centroids are sequentially updated every time incoming data is sequentially trained

Concept drift detection algorithm

- Train time: Trained centroids sequentially updated
- Predict time: Recent centroids sequentially updated after an anomaly is detected

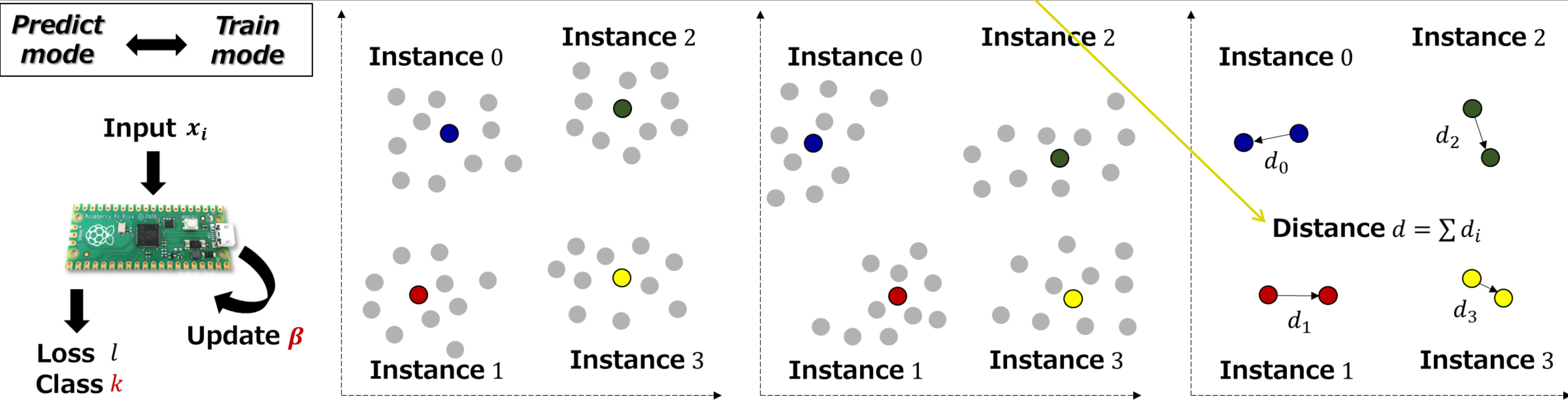


Centroids are sequentially updated every time prediction is done for incoming data

Concept drift detection algorithm

- **Train time: Trained centroids sequentially updated**
- **Predict time: Recent centroids sequentially updated**

Drift is detected when moving distances exceed a threshold



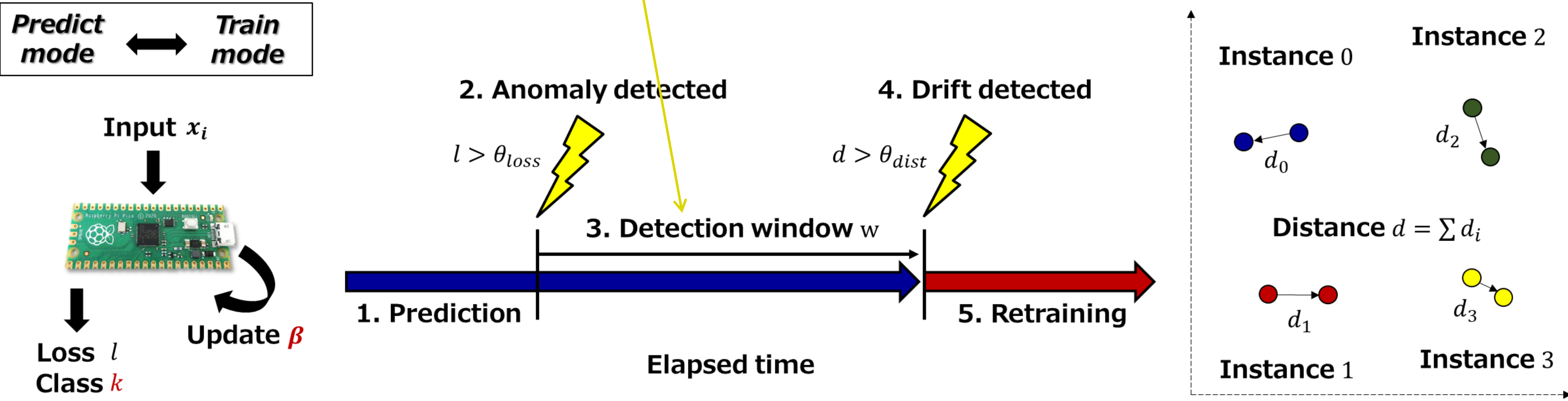
(a) Two modes (b) Trained centroids (c) Recent centroids (d) Moving distances

After a certain time-window is passed, the moving distances are evaluated

Concept drift detection algorithm

- Train time: Trained centroids sequentially updated
- Predict time: Recent centroids sequentially updated

Drift is detected when moving distances exceed a threshold



(a) Two modes (b) Concept drift detection flow (c) Moving distances

Timing chart of concept drift detection and retraining (Steps 1, 2, 3, 4, and 5)

Evaluations: Comparisons

- Proposed detector is compared w/ other approaches

Detect the drifts and trigger retraining of the discriminative model

	Detector	Discriminative model
Proposed method	Proposed method	OS-ELM
Baseline	None	OS-ELM
Quant Tree [1]	Quant Tree	OS-ELM
SPLL [2]	SPLL	OS-ELM
ONLAD [3]	None	OS-ELM w/ forgetting method

Trainable neural network that has a single hidden layer is used as the discriminative model for anomaly detection

[1] Giacomo Boracchi et al., "Quant Tree: Histograms for Change Detection in Multivariate Data Streams", ICML'18.
[2] Ludmila Kuncheva, "Change Detection in Streaming Multivariate Data Using Likelihood Detectors", IEEE Trans. on Knowledge and Data Engineering (2013).
[3] Mineto Tsukada et al., "A Neural Network-Based On-device Learning Anomaly Detector for Edge Devices", IEEE Trans. on Computers (2020).

Evaluations: Comparisons

No detection
(No retraining)

Sequential algorithm

	Detector	Discriminative model
Proposed method	Proposed method	OS-ELM
Baseline	None	OS-ELM
Quant Tree [1]	Quant Tree	OS-ELM
SPLL [2]	SPLL	OS-ELM
ONLAD [3]	None	OS-ELM w/ forgetting method

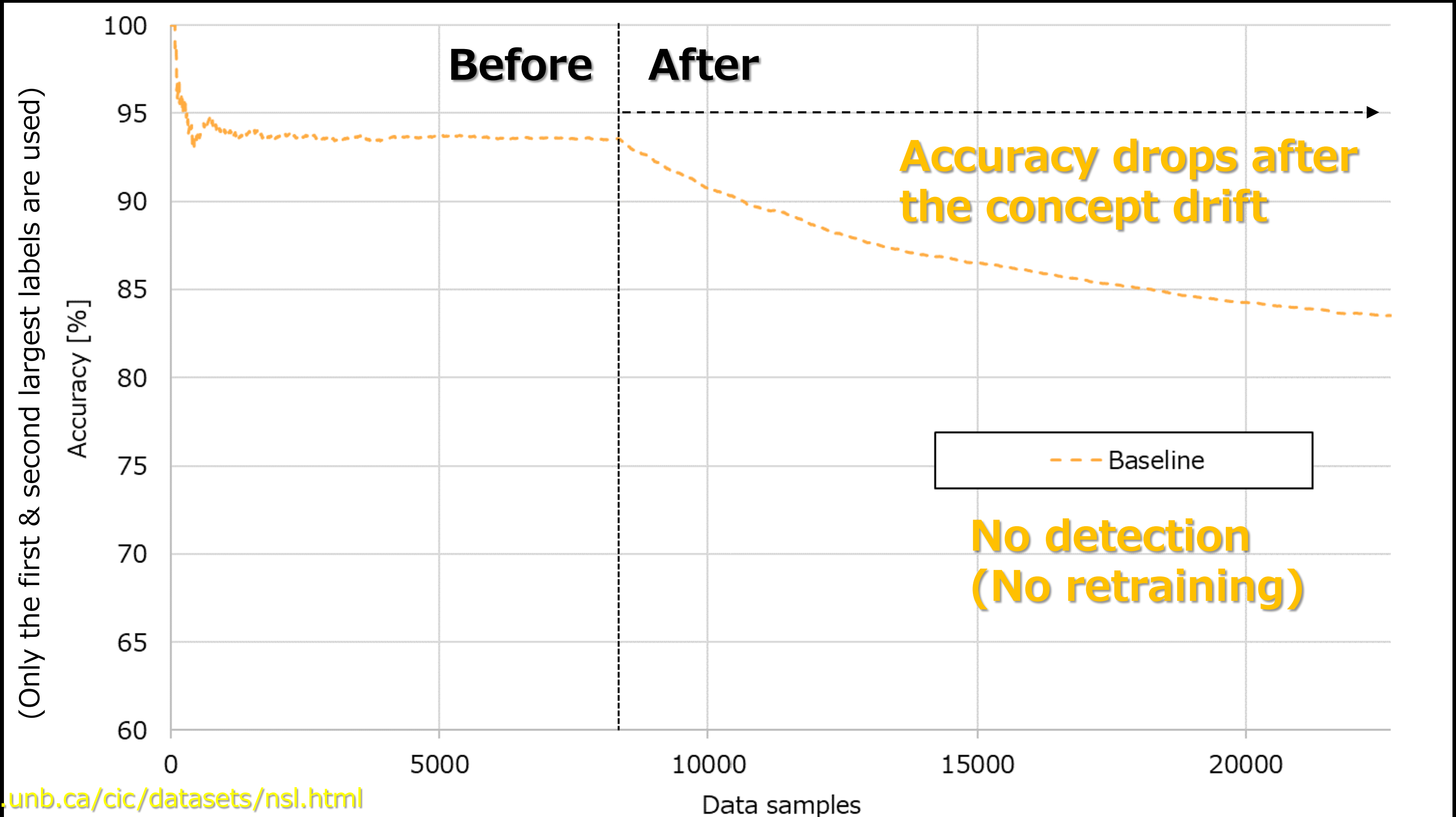
Actively retraining
while forgetting old data

Batch algorithms

[1] Giacomo Boracchi et al., "Quant Tree: Histograms for Change Detection in Multivariate Data Streams", ICML’18.
[2] Ludmila Kuncheva, "Change Detection in Streaming Multivariate Data Using Likelihood Detectors", IEEE Trans. on Knowledge and Data Engineering (2013).
[3] Mineto Tsukada et al., "A Neural Network-Based On-device Learning Anomaly Detector for Edge Devices", IEEE Trans. on Computers (2020).

Evaluations: Dataset

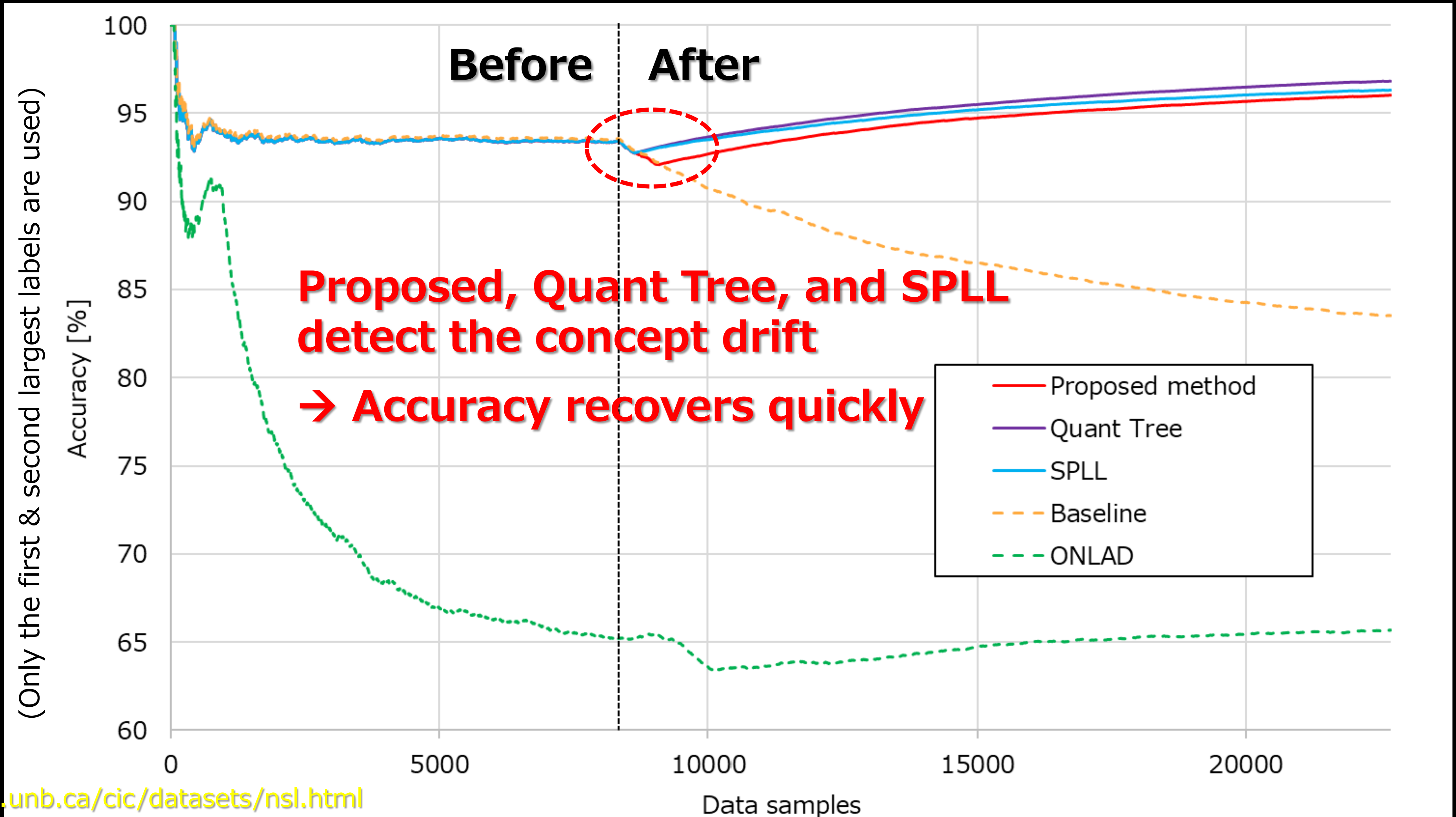
- Train & test samples of NSL-KDD dataset [1] are concatenated at 8333rd sample as a concept drift



[1] https://www.unb.ca/cic/datasets/nsl.html

Evaluations: Accuracy

- Train & test samples of NSL-KDD dataset [1] are concatenated at 8333rd sample as a concept drift



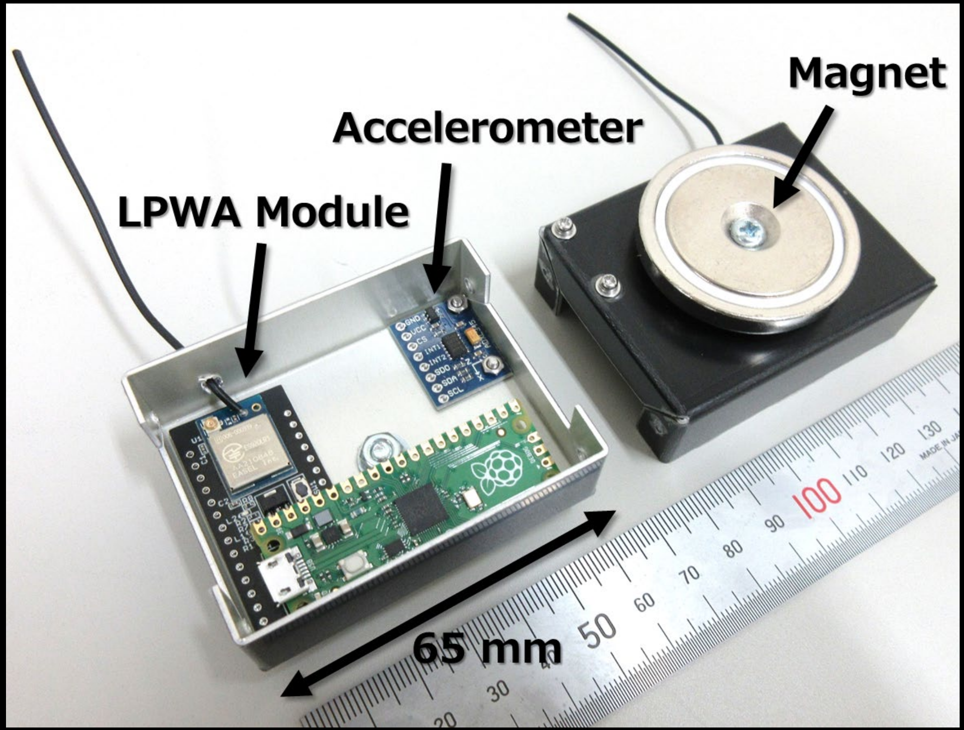
[1] https://www.unb.ca/cic/datasets/nsl.html

Evaluations: Memory utilization

[1] https://github.com/matutani/cooling-fan

- Memory utilization for Cooling fan dataset [1]
  Frequency spectrum (1 - 512Hz)

- Our target platform
  Raspberry Pi Pico (264 kB SRAM)
  Accelerometer

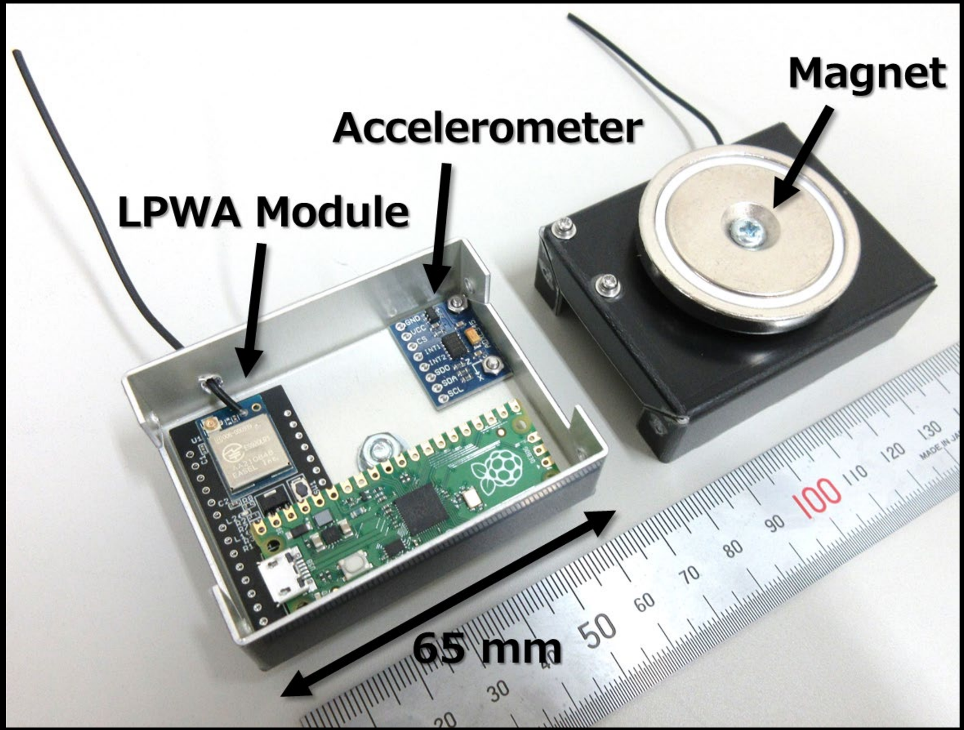


Wireless sensor nodes for anomaly detection on vibration patterns

Evaluations: Memory utilization

[1] https://github.com/matutani/cooling-fan

- Memory utilization for Cooling fan dataset [1]
  Frequency spectrum (1 - 512Hz)



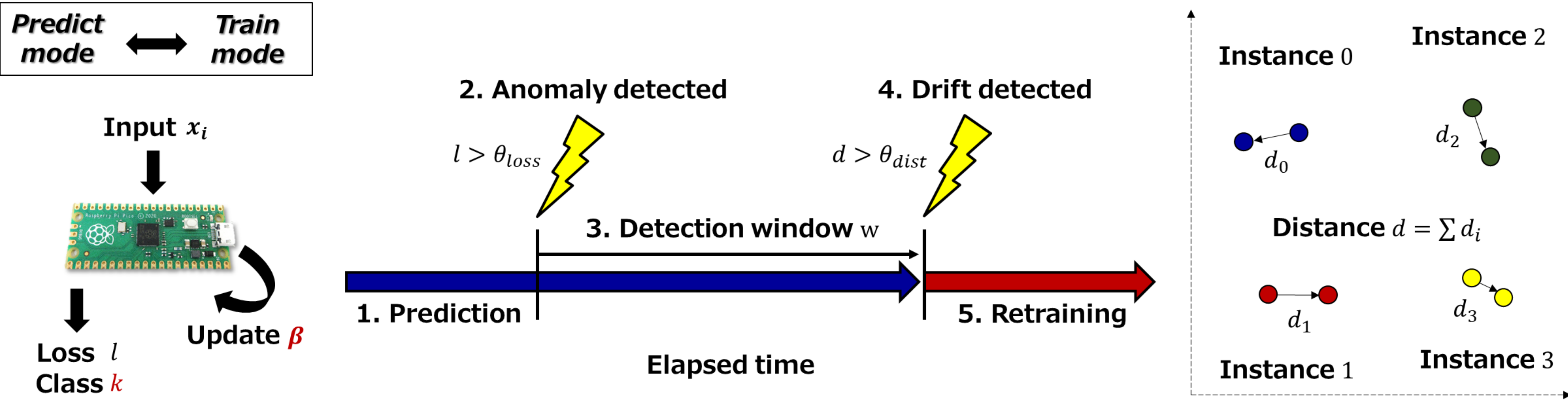
- Our target platform
  Raspberry Pi Pico (264 kB SRAM)

Sequential algorithm can significantly save memory utilization

	Batch size	Memory utilization
Proposed method	1 (Sequential)	69 kB
Quant Tree	235	619 kB
SPLL	235	1933 kB

Summary

- **A lightweight concept drift detection for on-device learning at tiny devices (e.g., Raspberry Pi Pico)**



(a) Two modes (b) Concept drift detection flow (c) Moving distances

Concept drifts can be detected as well as existing batch-based methods while reducing memory utilization by the sequential algorithm