# Final Report

Shanmu Wang, CR/RIX1, June 2023

BOSCH

# Content

- **1. Washing Machine Experiments**
  - Data Collection
  - Model Training and Testing

- **2. Time-Series Anomaly Detection**
  - Literature Scouting:
    - Federated Leaning in Industrial IoT
    - Transformer in Time-Series
  - Code Implementation:
    - Forecasting to Classification
    - Modify to an Easy-to-use Class
    - Integration to Time-Series Benchmark Code
    - Experiments of Cutting down Dataset

- **3. Other Support**
  - Classifier for SIG dataset from *DC/PAN*

# 1 Washing Machine Experiments

## a. Problem Definition

- When operating a washing machine, clothing might get stuck in between **gasket** and **the door**
  - which is referred to as **"dogbite"**
- This can cause **wear** and **tear** on the seal and lead to water leakage

- Use a **camera** to detect gasket's status and warn users if stuck
- Three main tasks:
  - building datasets
  - eliminating imaging flare
  - training a CV-based classification model





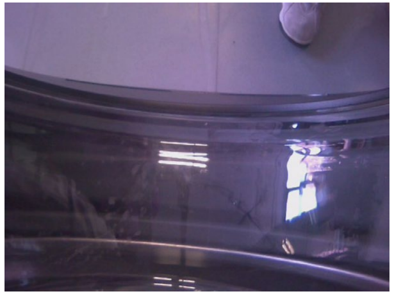


BOSCH

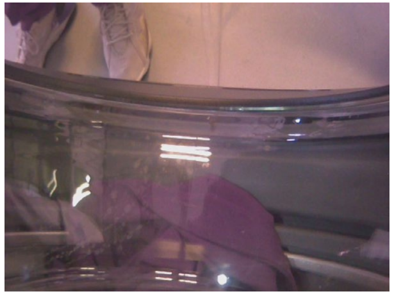
# 1 Washing Machine Experiments
## b. Data Collection

- **Hardware:**
  - An **IMX219-160 camera** with 800 megapixels and a 160-degree field of view (FOV)
  - An **NVIDIA Jetson Nano** with 4G 64-bit memory and a computing power of 472 GFLOPs

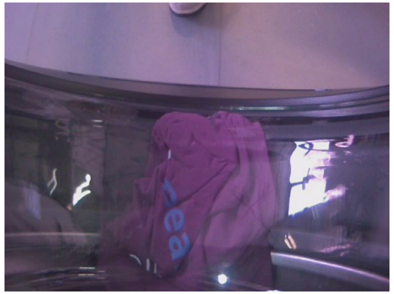
- **Dataset Collection:**



Class 0: w/o cloth



Class 1: not dogbite

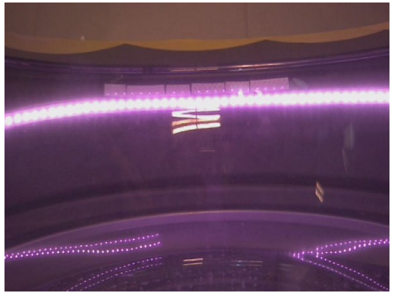
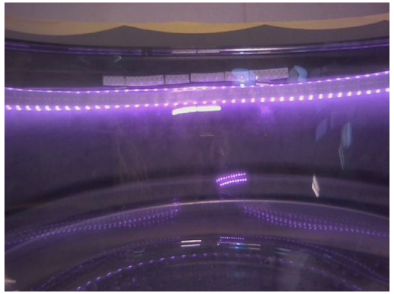


Class 2: dogbite

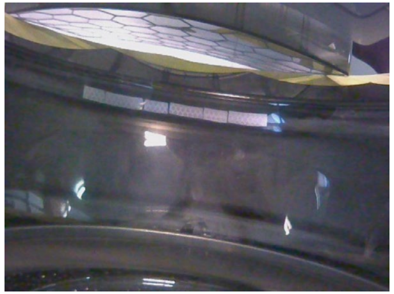
| | |
|---|---|
| **Amount** | Class 0: 37, Class 1: 84, Class 2: 83 |
| **Figure Size** | 640 * 480 pixels |
| **Clothing type** | scarves, T-shirts, etc. |
| **Scenarios** | not being stuck → partially stuck → mostly stuck |
| **Operator’s posture** | 2~3 different postures |

# 1 Washing Machine Experiments
## b. Data Collection

- **Mitigating the image flare:**

Effect of the infrared light strip



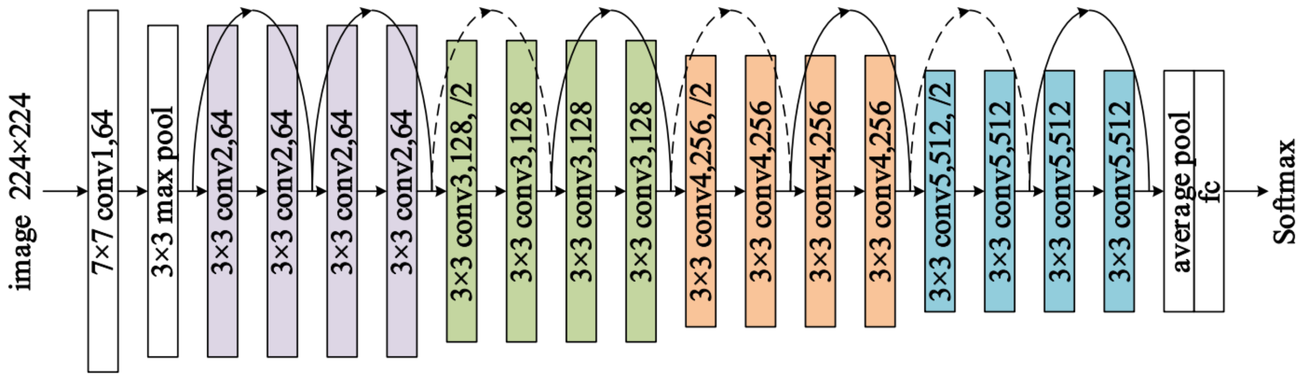

Effect of the grey acrylic & black foam board

- Infrared light strip can eliminate flare to some extend, but hard to deal with flare caused by strong light
- Black foam board has the best effect but will also block users' from seeing the running status of the washing machine

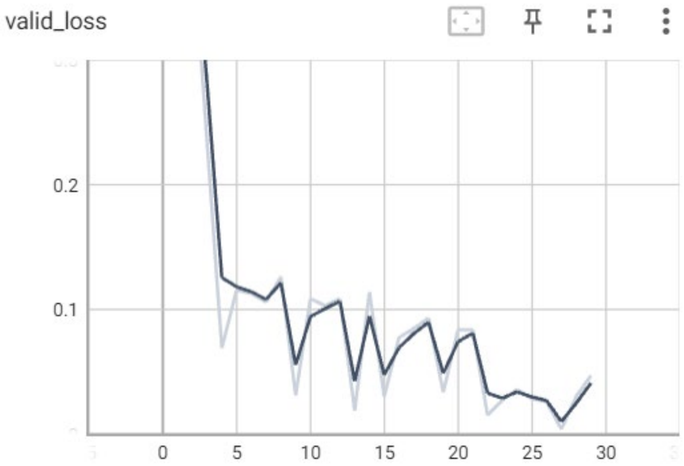
# 1 Washing Machine Experiments

## c. Model Training

- **Dataset:** 7:1:2 for training, validation, and testing




The structure of ResNet-18




Loss on the validation dataset

- The trained model achieved an accuracy of over **97%** on the test dataset
- The performance of ResNet-34 and ResNet-18 is quite similar

# 1 Washing Machine Experiments
## c. Model Training

- **Misclassification Example**:
  - True label: Class 2 - dogbite
  - Predict label: Class 1 - w/ cloth but not dogbite
- Even for human observers, this image could be **misleading**
- The model assigned a probability of only **54%** for class 1, only slightly higher than the probability for class 2

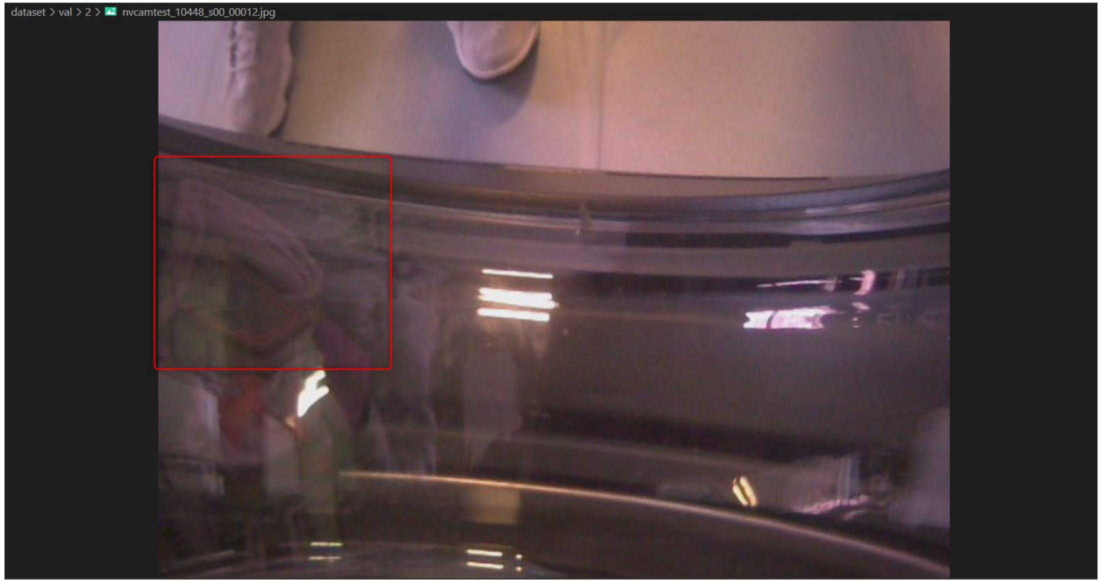



Figure that has been misclassified by our model

# 2 Time Series Anomaly Detection
## a. Problem Definition

- Time-series data from different industries or machines
  - **Anomaly Detection** is essential and challenge, because of:
    - Need to protect **Data privacy**
    - **High cost of customizing models** for **different sensors** deployed in **different scenarios**
- **Key idea:**
  - **Federated Learning** for protecting privacy
  - **Pretrained General Model** that can be finetuned to apply to various tasks
    - with the ability to suitable for input of different length, transfer learning, etc.
  - In a framework of FL, data from edge devices can be used to improve the general model in a private way



BOSCH

# 2 Time Series Anomaly Detection

## b. Literature Scouting: SemiPFL

- **SemiPFL: Personalized Semi-Supervised Federated Learning Framework for Edge Intelligence**
  - IEEE Internet of Things Journal (2023)
- Semi-supervised training method:

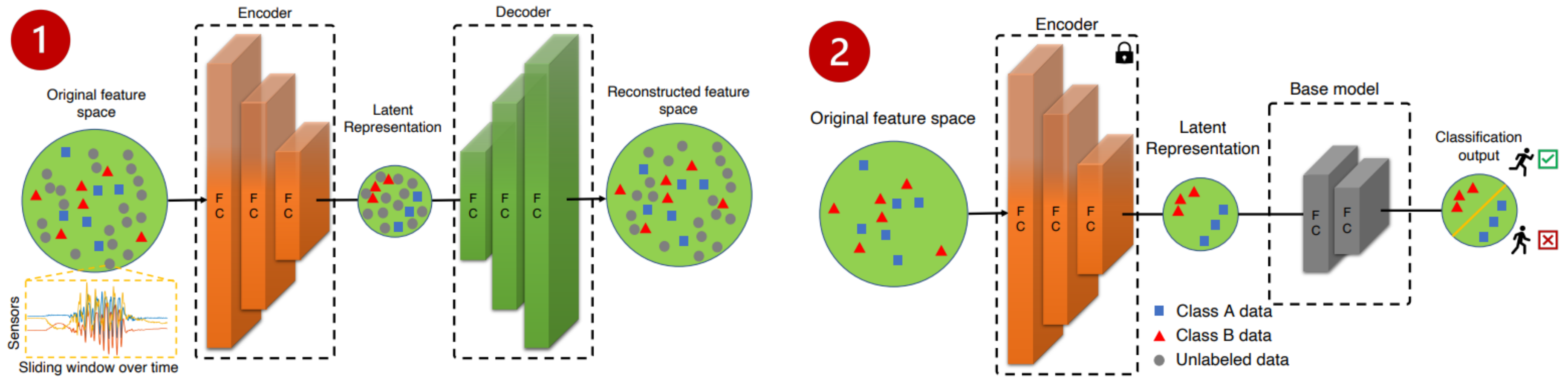



Fig. 2: Overview of the semi-supervised learning method used in this work. First, we train an autoencoder on the whole labeled and unlabeled data. Then, we use the encoder to transform the data to its latent representation, and we use the labeled data to train a base classifier. The final model is the trained encoder followed by the base classifier.




BOSCH

# 2 Time Series Anomaly Detection

## b. Literature Scouting: SemiPFL

- **SemiPFL architecture:**
  - A semi-supervised federated learning architecture

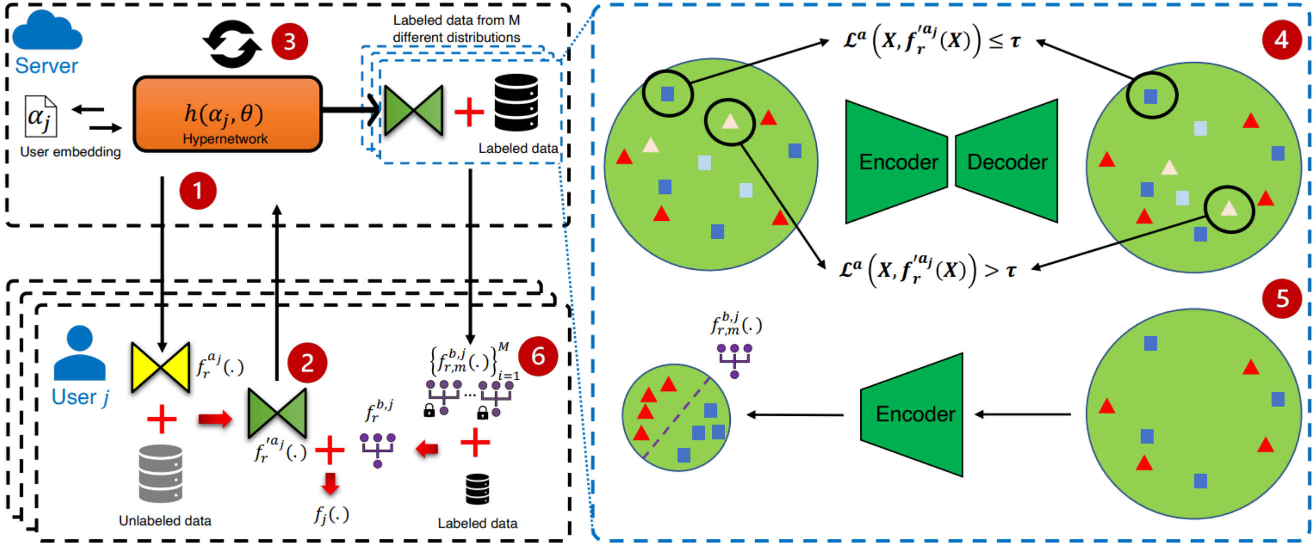



Fig. 3: Overall architecture of the proposed SemiPFL. 1) The server selects a user, generates a personalized autoencoder using the user embedding $\alpha_j$ via a Hyper-network, and sends it to the user. 2) The user fine-tunes the model using its unlabeled data and sends it back to the server. 3) The server updates the user embedding document and Hyper-network using the fine-tuned model. 4) The server encodes its labeled data using the encoding part of the user autoencoder, and 5) trains a set of base models using supervised learning and sends it to the user. 6) The user generates its base model by aggregating them using labeled data.

**Algorithm 1:** SemiPFL

**Output:** Personalized models: $\{f_j(.)\}_{j=1}^K$

**for** $r = 0, 1, \cdots, R-1$ **do**
- 1.1- Server randomly select user $j \in [K]$;
- 1.2- $f_r^{a_j}(.) = h(\alpha_j, \theta)$;
- 1.3- $f'^{a_j}_r(.) \leftarrow f_r^{a_j}(.)$;
- 2- **for** $t \in [T]$ **do**
  - 2.1- User j sample mini-batch $\gamma \subset U_j \bigcup \{X|(X,Y) \in D_j\}$;
  - 2.2- $f'^{a_j}_r(.) \leftarrow f'^{a_j}_r(.) - \eta \nabla_{f'^{a_j}_r(.)} \mathcal{L}^a_j(\gamma)$;
- **end**
- 3- Eqn. (12);
- 4- Eqn. (13);
- 5.1- $D'^j_{S_m} \leftarrow \{(f'^{a_j}_{r,enc}(X), Y)|(X,Y) \in D_{S_m}\} \forall m \in [M]$;
- 5.2- $\{f^{b,j}_{r,m}(.)\}_{m=1}^M \leftarrow \{f^b(.)\}_{m=1}^M$;
- 5.3- **for** $\{t, m\} \in [T \times M]$ **do**
  - 5.3.1- Server sample mini-batch $\gamma \subset D'^j_{S_m}$;
  - 5.3.2- $f^{b,j}_{r,m}(.) \leftarrow f^{b,j}_{r,m}(.) - \eta \nabla_{f^{b,j}_{r,m}(.)} \mathcal{L}(\gamma)$;
- **end**
- 6.1 Freeze $\{f^{b,j}_{r,m}(.)\}_{m=1}^M$;
- 6.2 Initialize $\{\chi_m\} m = 1^M = \{\frac{1}{M}\}_{m=1}^M$;
- 6.2- $f'^{b,j}_r(.) \leftarrow \sum_{m=1}^M \chi_m \cdot f^{b,j}_{r,m}(.)$;
- 6.3- $D'_j \leftarrow \left\{(f'^{a_j}_{r,enc}(X), Y)|(X,Y) \in D_j\right\}$;
- 6.4- **for** $t \in [T']$ **do**
  - 6.4.1- User j sample mini-batch $\gamma \subset D'_j$;
  - 6.4.2- $f'^{b,j}_r(.) \leftarrow f'^{b,j}_r(.) - \eta \nabla_{f'^{b,j}_r(.)} \mathcal{L}_j(\gamma)$;
  - Subject to: $\sum_{m=1}^M \chi_m = 1$;
- **end**
- 6.5- $f_j(.) \leftarrow f'^{b,j}_r(f'^{a_j}_{r,enc}(.))$ ;

**end**

# 2 Time Series Anomaly Detection
## b. Literature Scouting: Pyraformer

- **Pyraformer: ICLR 2022 Oral**

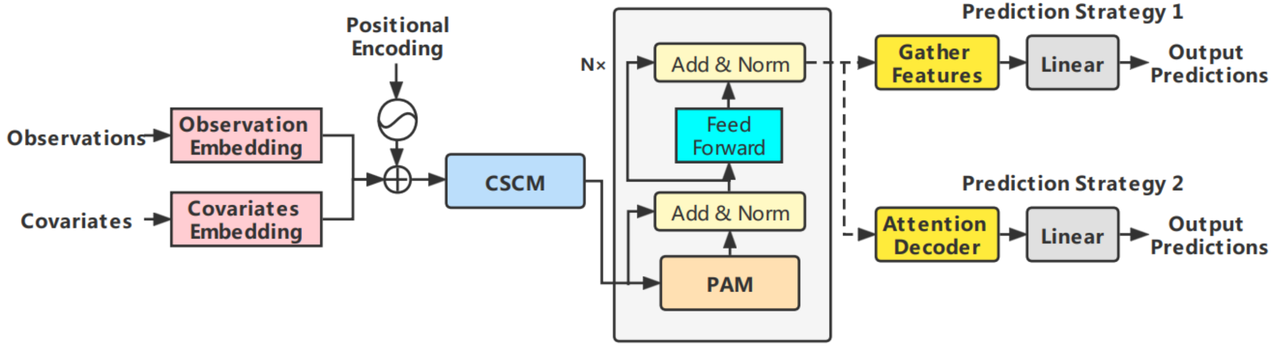



Figure 2: The architecture of Pyraformer: The CSCM summarizes the embedded sequence at different scales and builds a multi-resolution tree structure. Then the PAM is used to exchange information between nodes efficiently.

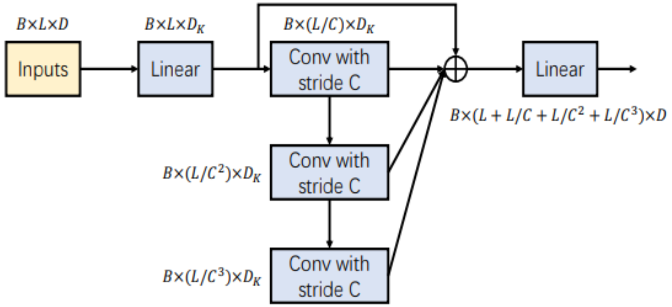
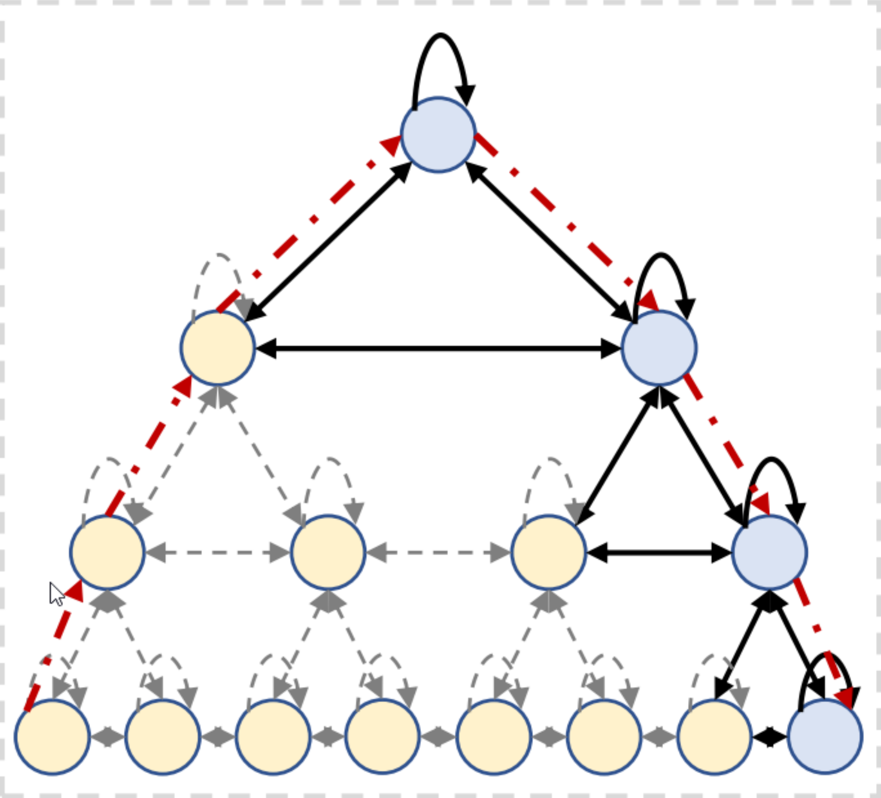



Figure 3: Coarser-scale construction module: $B$ is the batch size and $D$ is the dimension of a node.

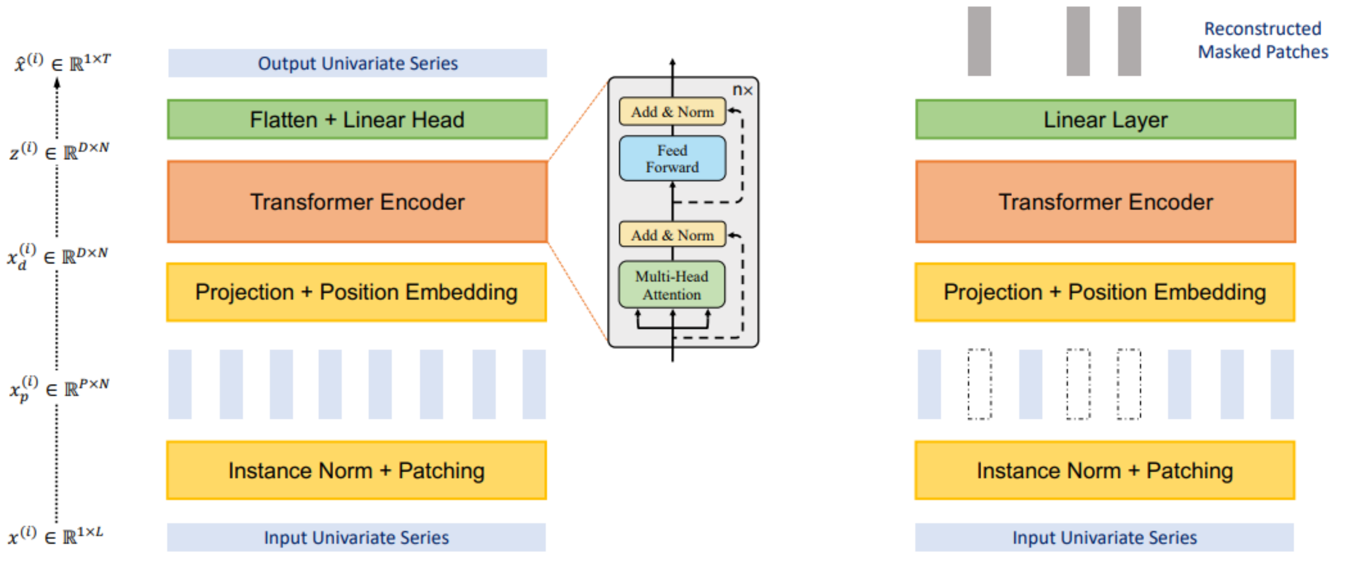


- Proposed **pyramidal attention module (PAM)**
- More **efficient**, support **multi-resolution representation**
- such a sparse attention mechanism is not supported in the existing deep learning libraries, such as Pytorch and TensorFlow

# 2 Time Series Anomaly Detection

## b. Literature Scouting: PatchTST

- **PatchTST: ICLR 2023**



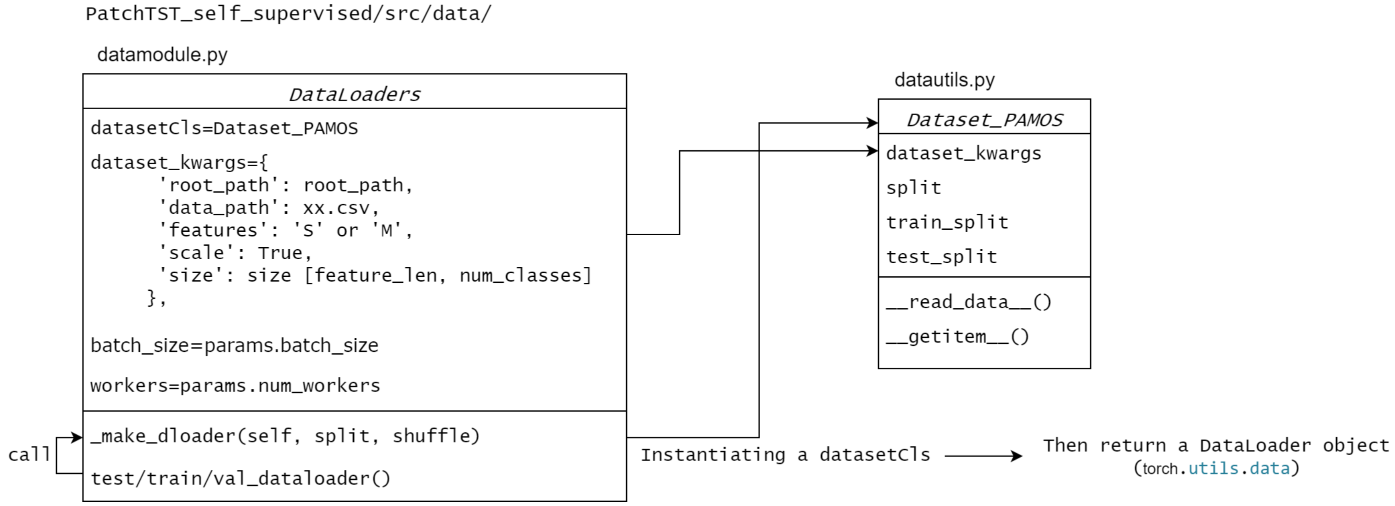

(b) Transformer Backbone (Supervised) (c) Transformer Backbone (Self-supervised)

- Achieve SOTA performance in **Long-term time series forecasting** tasks
- Proposed the **patching** scheme, improve the efficiency and make it possible for suiting different tasks
- Proposed **self-supervised learning** and **finetune** method

# 2 Time Series Anomaly Detection

## c. Code Implementation

- Forecasting to Classification
  - $z \in R^{D \times N} \rightarrow flatten \rightarrow dropout \rightarrow Conv1D \rightarrow Max\ pooling \rightarrow Conv1D \rightarrow Batch\ Normalization \rightarrow FC \rightarrow Softmax$
- Suit for **PAMOS dataset** & Modify to an **easy-to-use** class
- Re-org and integrate the model into the existing time-series benchmark repo




```
if __name__ == '__main__':
    PatchTST_model = Build_Model()
    PatchTST_model.compile_and_fit(mode='pretrain', batch_size=32, epochs=200, validation_split=0.7, dset_name='FordA', pretrained_model_id=2)
    PatchTST_model.compile_and_fit(mode='finetune', batch_size=128, epochs=100, validation_split=0.7, dset_name='FordA',\
                                   pretrained_model_id=2, ft_pretrained_epoch=200, ft_model_id=1, finetune_mode='two-stage')
    PatchTST_model.compile_and_fit(mode='full_supervised', batch_size=32, epochs=100, validation_split=0.7, dset_name='FordB', fs_model_id=1)
```

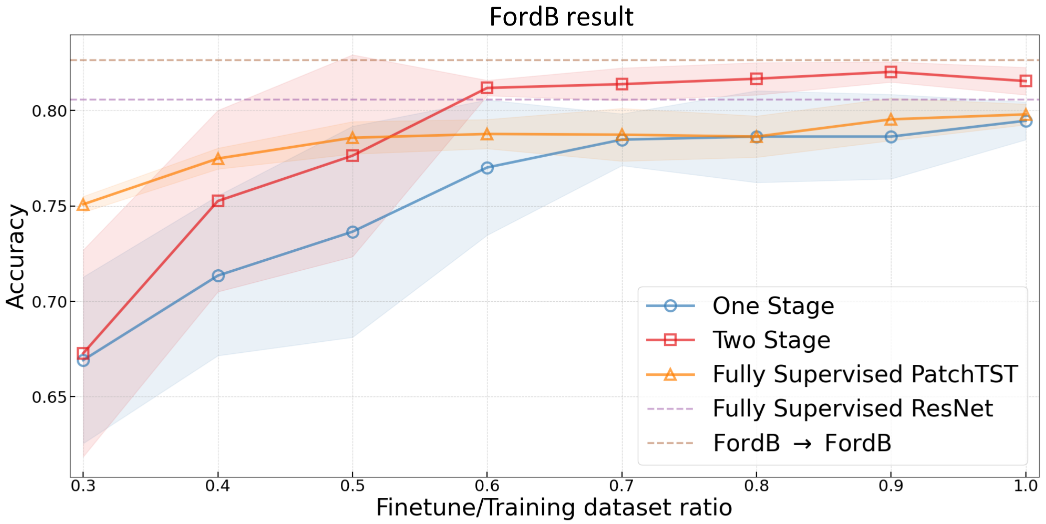
# 2 Time Series Anomaly Detection
## d. Experiments Results

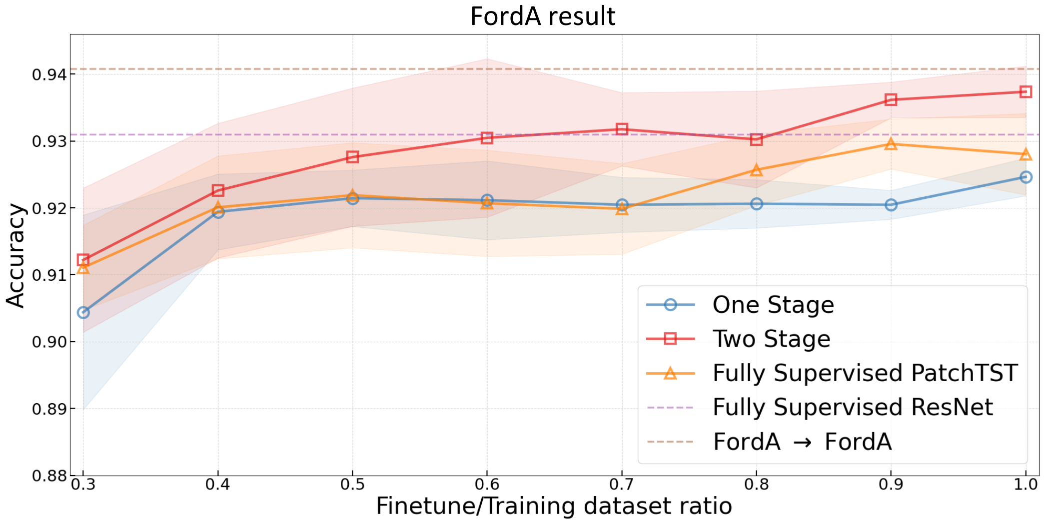
- **Datasets**:
  - Choose **PAMOS FordA & FordB** as the experimental datasets
  - Unit-variable time-series
- **Experiment Settings:**
  - **Baselines:** Fully Supervised ResNet, Fully Supervised PatchTST
  - **Focus on transfer leaning:** Pretrained with FordA (Self-supervised) then finetune with FordB, or vice versa
    - **One-stage** (only finetune the classification head)
    - **Two-stage** (first finetune the head, then the entire network, namely the head + backbone)
  - Cutting down the size of data used for fully supervised training and finetuning
  - Repeat for **10** times under each setting

| | FordA | FordB |
|---|---|---|
| Train | 3600 | 3635 |
| Test | 1319 | 809 |
| No. of classes | 2 | |
| Length of feature | 500 | |
| ResNet (T:V=7:3) | 91.3% | 80.6% |
| FCN (T:V=7:3) | 90.5% | 71.6% |

# 2 Time Series Anomaly Detection
## d. Experiments Results

- **Observations:**
  - Pretrain and finetune on the same dataset can achieve best performance
  - Two-stage finetune is better than One-stage
  - Two-stage finetune can surpass ResNet baseline with less data
  - One-stage finetune can also achieve satisfying results

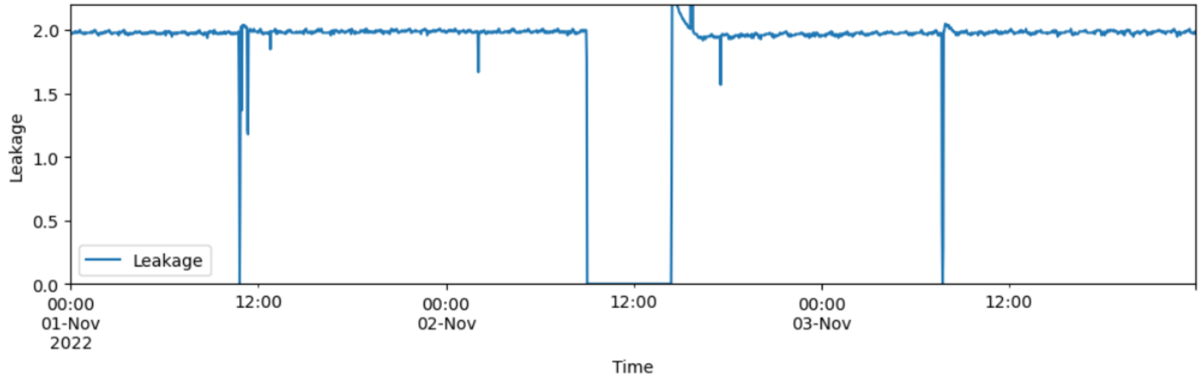


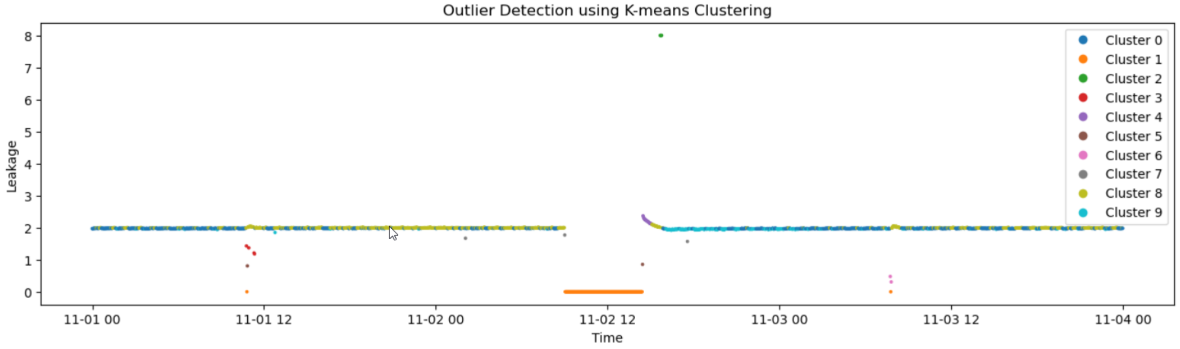



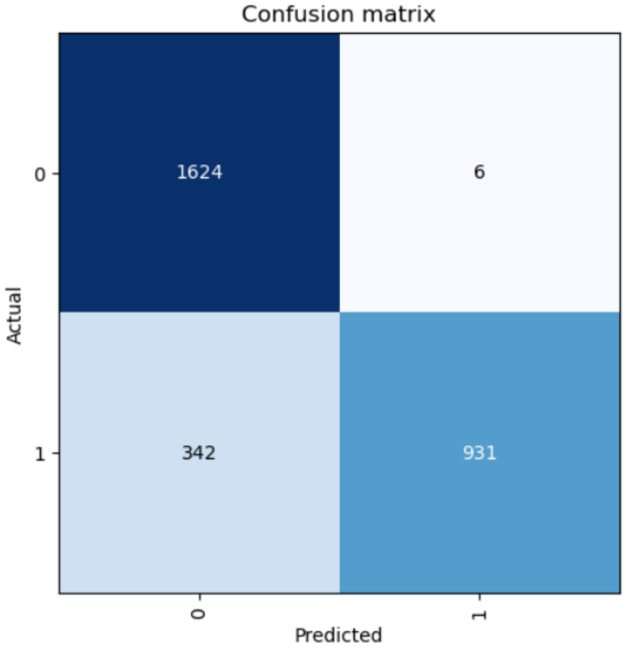
# 3 Other Support

## Classifier for SIG dataset from DC/PAN

- **Preprocessing for SIG dataset:**
  - Concatenate all single csv files to obtain daily data
  - Use $K - means$ to automatically assign label for SIG dataset




- Use a simple ResNet to classify, the overall accuracy on the validation set is **93.5%**

# 祝所有RIX人前程似锦！

Best wishes to all RIXers' future endeavors :-)

BOSCH