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

2. Time-Series Anomaly Detection
   - Literature Scouting:
     - Federated Learning 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.
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:

- 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 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
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:**

![Diagram of SemiPFL](image)

**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.
2 Time Series Anomaly Detection

b. Literature Scouting: SemiPFL

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

![Diagram](image)

**Algorithm 1: SemiPFL**

Output: Personalized models: \( \{ f_j() \}_{\ell=1}^{K} \)

for \( r = 0, 1, \ldots, R - 1 \)

1. Server randomly select user \( j \in [K] \);
2. \( f_{\alpha_j}^{\text{init}}(\cdot) = h(\alpha_j, \theta) \);
3. \( f_{\nu_j}^{\text{init}}(\cdot) \leftarrow f_{\nu_j}^{\text{init}}(\cdot) \);

for \( t \in [T] \)

1. User \( j \) sample mini-batch \( \gamma \subset U_j \cup \{ X | (X, Y) \in D_j \} \);
2. \( f_{r_{\gamma j}}^{\text{init}}(\cdot) \leftarrow f_{r_{\gamma j}}^{\text{init}}(\cdot) - \eta \nabla f_{r_{\gamma j}}^{\text{init}}(\cdot) \mathcal{L}(\gamma) \);

end

3. Eqn. (12);
4. Eqn. (13);

1. \( D_{\gamma_{\gamma_{\gamma \gamma}}} \leftarrow \{ (f_{r_{\gamma j}}^{\text{init}}(X), Y) | (X, Y) \in D_{\gamma_{\gamma_{\gamma \gamma}}} \} \forall m \in [M] \);
2. \( \{ f_{r_{\gamma j}}^{\text{init}}(\cdot) \}_{m=1}^{M} \leftarrow \{ f_{r_{\gamma j}}^{\text{init}}(\cdot) \}_{m=1}^{M} \);
3. for \( \{ t, m \} \in [T \times M] \)

5.1. Server sample mini-batch \( \gamma \subset D_{\gamma_{\gamma_{\gamma \gamma}}}^j \);
5.2. \( f_{r_{\gamma j}}^{\text{init}}(\cdot) \leftarrow f_{r_{\gamma j}}^{\text{init}}(\cdot) - \eta \nabla f_{r_{\gamma j}}^{\text{init}}(\cdot) \mathcal{L}(\gamma) \);

end

1. Freeze \( \{ f_{r_{\gamma j}}^{\text{init}}(\cdot) \}_{m=1}^{M} \);
2. Initialize \( \{ \chi_m \}_{m=1}^{M} = \{ \frac{1}{M} \}_{m=1}^{M} \);
3. \( f_{r_{\gamma j}}^{\text{init}}(\cdot) \leftarrow \sum_{m=1}^{M} \chi_m \cdot f_{r_{\gamma j}}^{\text{init}}(\cdot) \);
4. for \( t \in [T] \)

6.1. User \( j \) sample mini-batch \( \gamma \subset D_j^\gamma \);
6.2. \( f_{r_{\gamma j}}^{\text{init}}(\cdot) \leftarrow f_{r_{\gamma j}}^{\text{init}}(\cdot) - \eta \nabla f_{r_{\gamma j}}^{\text{init}}(\cdot) \mathcal{L}(\gamma) \);

Subject to: \( \sum_{m=1}^{M} \chi_m = 1 \);

end

6.5. \( f_j(\cdot) \leftarrow f_{r_{\gamma j}}^{\text{init}}(f_{r_{\gamma j}}^{\text{init}}(\cdot)) \);
2 Time Series Anomaly Detection

b. Literature Scouting: Pyraformer

- Pyraformer: ICLR 2022 Oral

- 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

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.
2 Time Series Anomaly Detection

b. Literature Scouting: PatchTST

- PatchTST: ICLR 2023

- Achieve SOTA performance in Long-term time series forecasting tasks
- Proposed the patching scheme, improve the efficiency and make it possible for suit different tasks
- Proposed self-supervised learning and finetune method
2 Time Series Anomaly Detection

c. Code Implementation

- Forecasting to Classification
  \[ z \in \mathbb{R}^{D \times N} \rightarrow \text{flatten} \rightarrow \text{dropout} \rightarrow \text{Conv1D} \rightarrow \text{Max pooling} \rightarrow \text{Conv1D} \rightarrow \text{Batch Normalization} \rightarrow \text{FC} \rightarrow \text{Softmax} \]

- Suit for PAMOS dataset & Modify to an easy-to-use class

- Re-org and integrate the model into the existing time-series benchmark repo

```python
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

<table>
<thead>
<tr>
<th></th>
<th>FordA</th>
<th>FordB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>3600</td>
<td>3635</td>
</tr>
<tr>
<td>Test</td>
<td>1319</td>
<td>809</td>
</tr>
<tr>
<td>No. of classes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Length of feature</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>ResNet (T:V=7:3)</td>
<td>91.3%</td>
<td>80.6%</td>
</tr>
<tr>
<td>FCN (T:V=7:3)</td>
<td>90.5%</td>
<td>71.6%</td>
</tr>
</tbody>
</table>
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 :-)