Laboratory Assignment: Domain Adaptation for Vibration Data

Total Points: 100

Overview

In this laboratory assignment, you will implement and evaluate different domain adaptation techniques for vibration data classification. You will start with a baseline model and progressively implement more sophisticated adaptation methods.

We have preprocessed the data for you, they are FFT preprocessed vibration data. - Source is recorded under load 3 - Target is recorded under load 1 Each contains 10 classes, out of which 9 are different faults

Prerequisites

- Python 3.x
- PyTorch
- NumPy
- Basic understanding of neural networks and domain adaptation concepts (You can use Google colab)

Part 1: Baseline Model Implementation (20 points)

Task

Modify the baseline model in `model.py` by adding BatchNormalization layers to the feature extractor.

Requirements

1. Add BatchNormalization layers after each convolutional layer in the FeatureExtractor class (5 points)
2. Train the model on the source domain and evaluate it on both source and target domains (5 points)
3. Document the following metrics (10 points):
    - Source domain accuracy and training loss (5 points)
    - Target domain accuracy (5 points)

Grading Criteria

- Code should be running, otherwise 0 for point 2

Part 2: Domain Adaptation Methods Implementation (60 points)

2.1 CORAL Implementation (20 points)

Requirements

1. Implement the CORAL loss function (10 points)
2. Integrate CORAL loss into the training procedure (5 points)
3. Document results and compare with baseline (5 points)

Grading Criteria

- Correct implementation of CORAL loss calculation: https://arxiv.org/abs/1607.01719

  We define the CORAL loss as the distance between the second-order statistics (covariances) of the source and target features:

$$\ell_{CORAL} = \frac{1}{4d^2}\|C_S - C_T\|_F^2 \tag{1}$$

  where $\|\cdot\|_F^2$ denotes the squared matrix Frobenius norm. The covariance matrices of the source and target data are given by:

$$C_S = \frac{1}{n_S - 1}(D_S^\top D_S - \frac{1}{n_S}(\mathbf{1}^\top D_S)^\top(\mathbf{1}^\top D_S)) \tag{2}$$

$$C_T = \frac{1}{n_T - 1}(D_T^\top D_T - \frac{1}{n_T}(\mathbf{1}^\top D_T)^\top(\mathbf{1}^\top D_T)) \tag{3}$$

- Proper integration with the training loop and runing code
- Clear results documentation

2.2 AdaBN Implementation (20 points)

Requirements

1. Implement the AdaBN adaptation (15 points): https://arxiv.org/abs/1603.04779

Algorithm 1 Adaptive Batch Normalization (AdaBN)

for neuron j in DNN **do**
  Concatenate neuron responses on all images of target domain t: $\mathbf{x}_j = [\ldots, x_j(m), \ldots]$
  Compute the mean and variance of the target domain: $\mu_j^t = \mathbb{E}(\mathbf{x}_j^t)$, $\sigma_j^t = \sqrt{\text{Var}(\mathbf{x}_j^t)}$.
end for
for neuron j in DNN, testing image m in target domain **do**
  Compute BN output $y_j(m) := \gamma_j \frac{\left(x_j(m) - \mu_j^t\right)}{\sigma_j^t} + \beta_j$
end for

2. Document results and compare with previous methods (5 points)

Grading Criteria

- Correct implementation of AdaBN procedure
- Proper handling of BatchNorm statistics and runing code as intended
- Clear results documentation

2.3 Adversarial Domain Adaptation Implementation (20 points)

Requirements

1. Implement adversarial training procedure (15 points)
2. Document results and compare with other methods (5 points)

Grading Criteria

- Proper adversarial training procedure and runing code
- Clear results documentation

Part 3: Analysis and Report (20 points)

Requirements

Write a comprehensive report including: 1. Methodology (5 points) - Description of implemented methods - Implementation details - Training procedures

2. Results Analysis (10 points)
   - Comparative analysis of all methods
   - Performance metrics

- Training curves
- Discussion of advantages and limitations of each method

3. Conclusions (5 points)
   - Summary of findings
   - Recommendations for method selection
   - Potential improvements

Grading Criteria

- Depth of analysis
- Quality of visualizations
- Clarity of conclusions
- Technical writing quality

Submission Requirements

1. Code Files:
   - Modified `model.py`
   - Modified `train_utils.py`
   - Any additional helper functions
2. Report:
   - PDF format
   - Maximum 10 pages
   - Must include all required plots and tables
3. Results:
   - Trained model checkpoints
   - Training logs
   - Performance metrics

Deadline and Submission Instructions

- Submit all materials in a single ZIP file
- Naming convention: `StudentID_DomainAdaptation.zip`
- Deadline is 20 of november at 23:59
- Late submissions will incur a 10% penalty per day

Notes

- Code must be well-documented with comments
- Use the provided data loading and evaluation functions
- Code should be running
- Maintain the original code structure
- Include requirements.txt file if using additional packages