

FARAD (Fourier Clouds)

Code Implementation for Fast Bias Correction in Imbalanced Semi-Supervised Learning

1 Code

Code can be available in <https://anonymous.4open.science/r/FARAD-929E/README.md>

2 Overview

This repository contains the PyTorch implementation for the paper entitled “*Fourier Clouds: Fast Bias Correction for Imbalanced Semi-Supervised Learning*”, which builds upon and modifies the original CDMAD approach. The implementation focuses on class-imbalanced semi-supervised learning using the FARAD debiasing technique.

2.1 Core Methodology

The fundamental innovation of FARAD lies in its use of **random-phase images** as reference data for bias estimation. These images preserve the amplitude spectrum of real data while randomizing the phase spectrum, providing a more statistically representative reference compared to the uniform or blank images used in the original CDMAD approach.

3 Implementation Details

3.1 Algorithm Components

The FARAD methodology enhances existing semi-supervised learning algorithms through:

- **Fourier-based reference generation:** Creation of random-phase images that maintain amplitude characteristics
- **Enhanced bias correction:** Improved statistical representation for imbalanced datasets
- **Compatibility:** Integration with popular SSL methods like FixMatch and ReMixMatch

3.2 Key Parameters

The implementation uses the following notation and parameters:

- N_1 (`num_max`): Number of labeled data points for the majority class
- M_1 (`num_max_u`): Number of unlabeled data points for the majority class
- ρ_l (`imb_ratio`): Imbalance ratio for the labeled set
- ρ_u (`imb_ratio_u`): Imbalance ratio for the unlabeled set

4 Running the Code

4.1 FixMatch with FARAD

To execute `fixmatch_farad.py` with the FARAD-enhanced bias correction mechanism:

```
python fixmatch_farad.py \
  --num_max 1500 \
  --num_max_u 3000 \
  --imb_ratio 100 \
  --imb_ratio_u 100 \
  --dataset cifar10 \
  --gpu 0 \
  --manualSeed 0 \
  --out result_fixmatch_farad_cifar10
```

Listing 1: FixMatch-FARAD execution example

Parameter specification:

- GPU device: 0
- Majority class labeled samples: $N_1 = 1500$
- Majority class unlabeled samples: $M_1 = 3000$
- Labeled set imbalance ratio: $\rho_l = 100$
- Unlabeled set imbalance ratio: $\rho_u = 100$
- Dataset: CIFAR-10
- Random seed: 0

4.2 ReMixMatch with FARAD

To execute `remixmatch_farad.py` with FARAD reference images:

```
python remixmatch_farad.py \
  --num_max 450 \
  --num_max_u 1 \
  --imb_ratio 20 \
  --imb_ratio_u 1 \
  --dataset stl10 \
  --gpu 0 \
  --manualSeed 0 \
  --out result_remixmatch_farad_stl10
```

Listing 2: ReMixMatch-FARAD execution example

Parameter specification:

- GPU device: 0
- Majority class labeled samples: $N_1 = 450$
- Majority class unlabeled samples: $M_1 = 1$
- Labeled set imbalance ratio: $\rho_l = 20$
- Unlabeled set imbalance ratio: $\rho_u = 1$
- Dataset: STL-10
- Random seed: 0

Note: For STL-10, the parameters `num_max_u` and `imb_ratio_u` may have specific interpretations when the entire unlabeled set is utilized, following the original CDMAD experimental setup.

5 Training and Evaluation Protocol

5.1 Epoch Definition

In the context of these semi-supervised learning implementations, following established practices from FixMatch, ReMixMatch, and CDMAD studies, an “*epoch*” is defined as a fixed number of training iterations rather than a complete pass through the dataset.

5.2 Validation Schedule

The training protocol includes:

- **Validation frequency:** Performance evaluation on test set after each epoch
- **Default iteration count:** `val_iteration = 500` (configurable)
- **Checkpoint strategy:** Model evaluation and potential checkpoint saving every 500 training steps

6 Dependencies and Requirements

6.1 Core Dependencies

The implementation requires the following software components:

- **PyTorch:** Deep learning framework
- **NumPy:** Numerical computing library
- **SciPy:** Scientific computing extensions
- **torchvision:** Computer vision utilities

6.2 Dataset-Specific Requirements

Dataset loading scripts (e.g., `dataset/fix_cifar10.py`) may require additional dependencies:

- Custom augmentation libraries
- RandAugment (if utilized by dataset files)
- Dataset-specific preprocessing utilities