EffecientAD - Anomaly Detection Framework

EfficientAD is a PyTorch-based framework designed for anomaly detection tasks. The framework leverages teacher-student architectures, autoencoders, and pre-trained models to efficiently detect anomalies in datasets. This README provides details on the functionality, setup, and usage of the script.

Features

- **Teacher-Student Architecture**: Implements a pre-trained teacher model to guide the training of a student model.
- **Autoencoder Integration**: Includes an autoencoder to assist with anomaly detection.
- **Data Augmentation**: Employs various transformations to enhance training robustness.
- **Pretraining Penalty Option**: Supports optional pretraining penalty using ImageNet datasets.
- **Anomaly Map Generation**: Generates and visualizes anomaly maps with overlayed heatmaps.
- **Metrics and Evaluation**: Computes AUROC, precision, recall, F1 score, confusion matrix, and classification report.

File Structure

- **`effecientad.py`**: Main script implementing the anomaly detection pipeline.
- **`common`**: contains utility functions and model definitions:
  - `get_autoencoder`
  - `get_pdn_small`
  - `get_pdn_medium`

- `ImageFolderWithoutTarget`
- `ImageFolderWithPath`
- `InfiniteDataloader`

- **Dataset Directories**: The script expects a specific directory structure for datasets:

```
dataset/
    subdataset/
        training/
              good/
        validation/
            defective/
            good/
        testing/
            defective/
            good/
```

Requirements

Dependencies

- Python 3.8+
- PyTorch 1.10+
- torchvision
- NumPy
- OpenCV
- scikit-learn
- tqdm
- Matplotlib

Install dependencies using:

```
bash
Copy code
pip install torch torchvision numpy opencv-python scikit-lear
n tqdm matplotlib
```

Usage

Command-Line Arguments

The script supports the following arguments:

- `d, --dataset`: Name of the dataset (default: `anomaly_detection`).
- `s, --subdataset`: Sub-dataset for training, validation, and testing (default: `Flowers`).
- `o, --output_dir`: Directory to save output (default: `output/experiments`).
- `m, --model_size`: Model size (`small` or `medium`, default: `small`).
- `w, --weights`: Path to pre-trained model weights (default: `models/teacher_small_tmp_state.pth`).

  (Note : the models are pre-trained using WideRenet-101)
- `i, --imagenet_train_path`: Path to ImageNet training data or set to `none` to disable pretraining penalty.
- `a, --anomaly_detection_path`: Path to the anomaly detection dataset.
- `t, --train_steps`: Number of training steps (default: `10000`).

Running the Script

Example command to run the script:

```
python effecientad.py -d anomaly_detection -s Flowers -a /pat
```

```
h/to/dataset -w /path/to/weights.pth
```

Key Functions

1. Data Loading

- `ImageFolderWithoutTarget`: Loads datasets without targets for unsupervised training.
- `ImageFolderWithPath`: Loads test datasets with image paths for evaluation.

2. Teacher-Student Architecture

- The teacher model generates feature representations.
- The student model learns to replicate these representations, with deviations indicating anomalies.

3. Autoencoder

- Used for reconstructing inputs and detecting anomalies by comparing reconstructions.

4. Training

- The training loop optimizes the student model and autoencoder using:
    - Hard loss: Difference between teacher and student representations.
    - Penalty loss: Penalizes the student for deviating from normal patterns.
    - Reconstruction loss: Error in reconstructing input images.

5. Evaluation

- Generates anomaly maps for test images.
- Evaluates performance using:
    - AUROC
    - Precision-Recall Curve

- F1-Score
- Confusion Matrix

6. Anomaly Map Visualization

- Overlayed heatmaps highlight anomalies on original images.
- Output saved as `overlayed.png` files in the `test_output_dir`.

Output

- **Trained Models**: Saved in `output_dir/trainings` as `teacher_final.pth`, `student_final.pth`, and `autoencoder_final.pth`.
- **Anomaly Maps**: Saved in `output_dir/anomaly_maps`.

Advanced Options

Pretraining Penalty

If `imagenet_train_path` is set, ImageNet pretraining data is used to improve performance. This can be disabled by setting it to `none`.

Model Size

Choose between `small` and `medium` model sizes to balance performance and computational requirements.

Dataset Preparation

1. Organize the dataset as:

```
dataset/
    subdataset/
        training/
              good/
        validation/
            defective/
```

```
            good/
        testing/
            defective/
            good/
```

2. Ensure images are properly labeled for evaluation:
    - Place normal images in a `good` directory.
    - Place anomalous images in separate directories for each defect class.

Evaluation Metrics

Optimal Threshold

- Calculated using precision-recall curves to maximize F1-score.

Confusion Matrix

- Displays True Positives, False Positives, True Negatives, and False Negatives.

AUROC

- Measures the area under the ROC curve for classification performance.

Customization

Models

Modify `get_pdn_small`, `get_pdn_medium`, and `get_autoencoder` in `common` to experiment with different architectures.

Transformations

Adjust data augmentation strategies in the `train_transform` and `default_transform` pipelines.
