Kolektor Data

Contents

Kolektor Surface-Defect Dataset (CC BY-NC-SA 4.0).

Description:

This script provides a PyTorch Dataset, DataLoader, and PyTorch Lightning DataModule for the Kolektor Surface-Defect dataset. The dataset can be accessed at Kolektor Surface-Defect Dataset.

License:

The Kolektor Surface-Defect dataset is released under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0). For more details, visit Creative Commons License.

Reference:

Tabernik, Domen, Samo Šela, Jure Skvarč, and Danijel Skočaj. "Segmentation-based deep-learning approach for surface-defect detection." Journal of Intelligent Manufacturing 31, no. 3 (2020): 759-776.

class **anomalib.data.image.kolektor.Kolektor**(***root***=*'./datasets/kolektor'*, ***train_batch_size***=*32*, ***eval_batch_size***=*32*, ***num_workers***=*8*, ***task***=*TaskType.SEGMENTATION*, ***image_size***=*None*, ***transform***=*None*, ***train_transform***=*None*, ***eval_transform***=*None*, ***test_split_mode***=*TestSplitMode.FROM_DIR*, ***test_split_ratio***=*0.2*, ***val_split_mode***=*ValSplitMode.SAME_AS_TEST*, ***val_split_ratio***=*0.5*, ***seed***=*None*)

Bases: `AnomalibDataModule`

Kolektor Datamodule.

Parameters:

- **root** (*Path | str*) – Path to the root of the dataset
- **train_batch_size** (*int, optional*) – Training batch size. Defaults to `32`.
- **eval_batch_size** (*int, optional*) – Test batch size. Defaults to `32`.
- **num_workers** (*int, optional*) – Number of workers. Defaults to `8`.
- **TaskType)** (*task*) – Task type, 'classification', 'detection' or 'segmentation' Defaults to `TaskType.SEGMENTATION`.
- **image_size** (*tuple[int, int], optional*) – Size to which input images should be resized. Defaults to `None`.
- **transform** (*Transform, optional*) – Transforms that should be applied to the input images. Defaults to `None`.
- **train_transform** (*Transform, optional*) – Transforms that should be applied to the input images during training. Defaults to `None`.
- **eval_transform** (*Transform, optional*) – Transforms that should be applied to the input images during evaluation. Defaults to `None`.
- **test_split_mode** (*TestSplitMode*) – Setting that determines how the testing subset is obtained. Defaults to `TestSplitMode.FROM_DIR`
- **test_split_ratio** (*float*) – Fraction of images from the train set that will be reserved for testing. Defaults to `0.2`
- **val_split_mode** (*ValSplitMode*) – Setting that determines how the validation subset is obtained. Defaults to `ValSplitMode.SAME_AS_TEST`
- **val_split_ratio** (*float*) – Fraction of train or test images that will be reserved for validation. Defaults to `0.5`
- **seed** (*int | None, optional*) – Seed which may be set to a fixed value for reproducibility. Defaults to `None`.

`prepare_data()`

Download the dataset if not available.

This method checks if the specified dataset is available in the file system. If not, it downloads and extracts the dataset into the appropriate directory.

Return type:

None

Example

Assume the dataset is not available on the file system. Here's how the directory structure looks before and after calling the *prepare_data* method:

Before:

```
$ tree datasets
datasets
├── dataset1
└── dataset2
```

Calling the method:

```
>> datamodule = Kolektor(root="./datasets/kolektor")
>> datamodule.prepare_data()
```

After:

```
$ tree datasets
datasets
├── dataset1
├── dataset2
└── kolektor
    ├── kolektorsdd
    ├── kos01
    ├── ...
    └── kos50
        ├── Part0.jpg
        ├── Part0_label.bmp
        └── ...
```

class anomalib.data.image.kolektor.**KolektorDataset**(***task***, ***root***='./datasets/kolektor', ***transform***=None, ***split***=None)

Bases: AnomalibDataset

Kolektor dataset class.

Parameters:

- **task** (*TaskType*) – Task type, `classification`, `detection` or `segmentation`
- **root** (*Path* | *str*) – Path to the root of the dataset Defaults to `./datasets/kolektor`.
- **transform** (*Transform, optional*) – Transforms that should be applied to the input images. Defaults to `None`.
- **split** (*str* | *Split* | *None*) – Split of the dataset, usually Split.TRAIN or Split.TEST Defaults to `None`.

anomalib.data.image.kolektor.make_kolektor_dataset(root, train_split_ratio=0.8, split=None)

Create Kolektor samples by parsing the Kolektor data file structure.

The files are expected to follow this structure: - Image files: *path/to/dataset/item/image_filename.jpg*, *path/to/dataset/kos01/Part0.jpg* - Mask files: *path/to/dataset/item/mask_filename.bmp*, *path/to/dataset/kos01/Part0_label.bmp*

This function creates a DataFrame to store the parsed information in the following format:

	path	item	split	label	image_path	mask_path	label_index
0	KolektorSDD	kos01	test	Bad	/path/to/image_file	/path/to/mask_file	1

Parameters:

- **root** (*Path*) – Path to the dataset.
- **train_split_ratio** (*float, optional*) – Ratio for splitting good images into train/test sets. Defaults to `0.8`.
- **split** (*str* | *Split* | *None, optional*) – Dataset split (either ‘train’ or ‘test’). Defaults to `None`.

Returns:

An output DataFrame containing the samples of the dataset.

Return type:

pandas.DataFrame

Example

The following example shows how to get training samples from the Kolektor Dataset:

```
>>> from pathlib import Path
>>> root = Path('./KolektorSDD/')
>>> samples = create_kolektor_samples(root, train_split_ratio=0.8)
>>> samples.head()
       path       item  split label   image_path                    mask_path
   0   KolektorSDD   kos01  train Good  KolektorSDD/kos01/Part0.jpg  KolektorSD
   1   KolektorSDD   kos01  train Good  KolektorSDD/kos01/Part1.jpg  KolektorSD
   2   KolektorSDD   kos01  train Good  KolektorSDD/kos01/Part2.jpg  KolektorSD
   3   KolektorSDD   kos01  test  Good  KolektorSDD/kos01/Part3.jpg  KolektorSD
   4   KolektorSDD   kos01  train Good  KolektorSDD/kos01/Part4.jpg  KolektorSD
```

Previous
Folder Data

Next
MVTec Data