MVTec Data

Contents

MVTec AD Dataset (CC BY-NC-SA 4.0).

Description:

This script contains PyTorch Dataset, Dataloader and PyTorch Lightning DataModule for the MVTec AD dataset. If the dataset is not on the file system, the script downloads and extracts the dataset and create PyTorch data objects.

License:

MVTec AD dataset is released under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0)(https://creativecommons.org/licenses/by-nc-sa/4.0/).

References

- Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, Carsten Steger: The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection; in: International Journal of Computer Vision 129(4): 1038-1059, 2021, DOI: 10.1007/s11263-020-01400-4.
- Paul Bergmann, Michael Fauser, David Sattlegger, Carsten Steger: MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection; in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 9584-9592, 2019, DOI: 10.1109/CVPR.2019.00982.


```
class anomalib.data.image.mvtec.MVTec(root='./datasets/MVTec',
```

***category**='bottle', **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`

MVTec Datamodule.

Parameters:

- **root** (*Path | str*) – Path to the root of the dataset. Defaults to `"./datasets/MVTec"`.
- **category** (*str*) – Category of the MVTec dataset (e.g. “bottle” or “cable”). Defaults to `"bottle"`.
- **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. Defualts to `None`.

Examples

To create an MVTec AD datamodule with default settings:

```
>>> datamodule = MVTec()
>>> datamodule.setup()
```

```
>>> i, data = next(enumerate(datamodule.train_dataloader()))
>>> data.keys()
dict_keys(['image_path', 'label', 'image', 'mask_path', 'mask'])
```

```
>>> data["image"].shape
torch.Size([32, 3, 256, 256])
```

To change the category of the dataset:

```
>>> datamodule = MVTec(category="cable")
```

To change the image and batch size:

```
>>> datamodule = MVTec(image_size=(512, 512), train_batch_size=16, eval_batch_s
```

MVTec AD dataset does not provide a validation set. If you would like to use a separate validation set, you can use the `val_split_mode` and `val_split_ratio` arguments to create a validation set.

```
>>> datamodule = MVTec(val_split_mode=ValSplitMode.FROM_TEST, val_split_ratio=0
```

This will subsample the test set by 10% and use it as the validation set. If you would like to create a validation set synthetically that would not change the test set, you can use the `ValSplitMode.SYNTHETIC` option.

```
>>> datamodule = MVTec(val_split_mode=ValSplitMode.SYNTHETIC, val_split_ratio=0
```

`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 = MVTec(root="./datasets/MVTec", category="bottle")
>> datamodule.prepare_data()
```

After:

```
$ tree datasets
datasets
├── dataset1
├── dataset2
└── MVTec
    ├── bottle
    ├── ...
    └── zipper
```

class **anomalib.data.image.mvtec.MVTecDataset(*task*, *root='./datasets/MVTec'*, *category='bottle'*, *transform=None*, *split=None*)**

Bases: `AnomalibDataset`

MVTec dataset class.

Parameters:

- **task** (*TaskType*) – Task type, `classification`, `detection` or `segmentation`.
- **root** (*Path* | *str*) – Path to the root of the dataset. Defaults to `./datasets/MVTec`.
- **category** (*str*) – Sub-category of the dataset, e.g. 'bottle' Defaults to `bottle`.
- **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`.

Examples

```
from anomalib.data.image.mvtec import MVTecDataset
from anomalib.data.utils.transforms import get_transforms

transform = get_transforms(image_size=256)
dataset = MVTecDataset(
    task="classification",
    transform=transform,
    root='./datasets/MVTec',
    category='zipper',
)
dataset.setup()
print(dataset[0].keys())
# Output: dict_keys(['image_path', 'label', 'image'])
```

When the task is segmentation, the dataset will also contain the mask:

```
dataset.task = "segmentation"
dataset.setup()
print(dataset[0].keys())
# Output: dict_keys(['image_path', 'label', 'image', 'mask_path', 'mask'])
```

The image is a torch tensor of shape (C, H, W) and the mask is a torch tensor of shape (H, W).

```
print(dataset[0]["image"].shape, dataset[0]["mask"].shape)
# Output: (torch.Size([3, 256, 256]), torch.Size([256, 256]))
```

anomalib.data.image.mvtec.make_mvtec_dataset(*root*, *split=None*,

extensions=None)

Create MVTec AD samples by parsing the MVTec AD data file structure.

The files are expected to follow the structure:

path/to/dataset/split/category/image_filename.png path/to/dataset/ground_truth/category/mask_filename.png

This function creates a dataframe to store the parsed information based on the following format:

	path	split	label	image_path	mask_path	label_index
0	datasets/name	test	defect	filename.png	ground_truth/defect/filename_mask.png	1

Parameters:

- **root** (*Path*) – Path to dataset
- **split** (*str* | *Split* | *None, optional*) – Dataset split (ie., either train or test). Defaults to `None`.
- **extensions** (*Sequence[str]* | *None, optional*) – List of file extensions to be included in the dataset. Defaults to `None`.

Examples

The following example shows how to get training samples from MVTec AD bottle category:

```
>>> root = Path('./MVTec')
>>> category = 'bottle'
>>> path = root / category
>>> path
PosixPath('MVTec/bottle')
```

```
>>> samples = make_mvtec_dataset(path, split='train', split_ratio=0.1, seed=0)
>>> samples.head()
   path         split label image_path                           mask_path
0  MVTec/bottle train good MVTec/bottle/train/good/105.png MVTec/bottle/ground_
1  MVTec/bottle train good MVTec/bottle/train/good/017.png MVTec/bottle/ground_
2  MVTec/bottle train good MVTec/bottle/train/good/137.png MVTec/bottle/ground_
```

```
3  MVTec/bottle train good MVTec/bottle/train/good/152.png MVTec/bottle/ground_
4  MVTec/bottle train good MVTec/bottle/train/good/109.png MVTec/bottle/ground_
```

Returns:

an output dataframe containing the samples of the dataset.

Return type:

DataFrame

Previous
Kolektor Data

Next
Visa Data