Visa Data

Contents

Visual Anomaly (VisA) Dataset (CC BY-NC-SA 4.0).

Description:

This script contains PyTorch Dataset, Dataloader and PyTorch
Lightning DataModule for the Visual Anomal (VisA) dataset.

If the dataset is not on the file system, the script downloads and
extracts the dataset and create PyTorch data objects.

License:

The VisA 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/).

Reference:

- Zou, Y., Jeong, J., Pemula, L., Zhang, D., & Dabeer, O. (2022). SPot-the-Difference Self-supervised Pre-training for Anomaly Detection and Segmentation. In European Conference on Computer Vision (pp. 392-408). Springer, Cham.

```
class anomalib.data.image.visa.Visa(root='./datasets/visa', category='capsules', 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`

VisA Datamodule.

Parameters:

- **root** (*Path | str*) – Path to the root of the dataset Defaults to `"./datasets/visa"`.
- **category** (*str*) – Category of the Visa dataset such as `candle`. Defaults to `"candle"`.
- **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`.
- **task** (*TaskType*) – 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. Defatuls to `0.5`.
- **seed** (*int | None, optional*) – Seed which may be set to a fixed value for reproducibility. Defaults to `None`.

`apply_cls1_split()`

Apply the 1-class subset splitting using the fixed split in the csv file.

adapted from amazon-science/spot-diff

Return type:

`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 = Visa()
>> datamodule.prepare_data()
```

After:

```
$ tree datasets
datasets
├── dataset1
├── dataset2
└── visa
    ├── candle
    ├── ...
    ├── pipe_fryum
```

```
│   ├── Data
│   └── image_anno.csv
├── split_csv
│   ├── 1cls.csv
│   ├── 2cls_fewshot.csv
│   └── 2cls_highshot.csv
├── VisA_20220922.tar
└── visa_pytorch
    ├── candle
    ├── ...
    ├── pcb4
    └── pipe_fryum
```

`prepare_data` ensures that the dataset is converted to MVTec format. `visa_pytorch` is the directory that contains the dataset in the MVTec format. `visa` is the directory that contains the original dataset.

class **anomalib.data.image.visa.VisaDataset**(*task*, *root*, *category*, *transform=None*, *split=None*)

Bases: `AnomalibDataset`

VisA dataset class.

Parameters:

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

To create a Visa dataset for classification:

```
from anomalib.data.image.visa import VisaDataset
from anomalib.data.utils.transforms import get_transforms
```

```
transform = get_transforms(image_size=256)
dataset = VisaDataset(
    task="classification",
    transform=transform,
    split="train",
    root="./datasets/visa/visa_pytorch/",
    category="candle",
)
dataset.setup()
dataset[0].keys()

# Output
dict_keys(['image_path', 'label', 'image'])
```

If you want to use the dataset for segmentation, you can use the same code as above, with the task set to `segmentation`. The dataset will then have a `mask` key in the output dictionary.

```
from anomalib.data.image.visa import VisaDataset
from anomalib.data.utils.transforms import get_transforms

transform = get_transforms(image_size=256)
dataset = VisaDataset(
    task="segmentation",
    transform=transform,
    split="train",
    root="./datasets/visa/visa_pytorch/",
    category="candle",
)
dataset.setup()
dataset[0].keys()

# Output
dict_keys(['image_path', 'label', 'image', 'mask_path', 'mask'])
```

Previous
MVTec Data

Next
Video Data