Avenue Data

Contents

CUHK Avenue Dataset.

Description:

This module provides a PyTorch Dataset and PyTorch Lightning DataModule for the CUHK Avenue dataset. If the dataset is not already present on the file system, the DataModule class will download and extract the dataset, converting the .mat mask files to .png format.

Reference:

- Lu, Cewu, Jianping Shi, and Jiaya Jia. "Abnormal event detection at 150 fps in Matlab." In Proceedings of the IEEE International Conference on Computer Vision, 2013.

class **anomalib.data.video.avenue.Avenue**(***root***=*'./datasets/avenue'*, ***gt_dir***=*'./datasets/avenue/ground_truth_demo'*, ***clip_length_in_frames***=*2*, ***frames_between_clips***=*1*, ***target_frame***=*VideoTargetFrame.LAST*, ***task***=*TaskType.SEGMENTATION*, ***image_size***=*None*, ***transform***=*None*, ***train_transform***=*None*, ***eval_transform***=*None*, ***train_batch_size***=*32*, ***eval_batch_size***=*32*, ***num_workers***=*8*, ***val_split_mode***=*ValSplitMode.SAME_AS_TEST*, ***val_split_ratio***=*0.5*, ***seed***=*None*)

Bases: `AnomalibVideoDataModule`

Avenue DataModule class.

Parameters:

- **root** (*Path | str*) – Path to the root of the dataset Defaults to `./datasets/avenue`.
- **gt_dir** (*Path | str*) – Path to the ground truth files Defaults to `./datasets/avenue/ground_truth_demo`.
- **clip_length_in_frames** (*int, optional*) – Number of video frames in each clip. Defaults to `2`.
- **frames_between_clips** (*int, optional*) – Number of frames between each consecutive video clip. Defaults to `1`.
- **target_frame** (*VideoTargetFrame*) – Specifies the target frame in the video clip, used for ground truth retrieval Defaults to `VideoTargetFrame.LAST`.
- **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`.
- **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`.
- **val_split_mode** (*ValSplitMode*) – Setting that determines how the validation subset is obtained. Defaults to `ValSplitMode.FROM_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`.

Examples

To create a DataModule for Avenue dataset with default parameters:

```
datamodule = Avenue()
```

```
datamodule.setup()

i, data = next(enumerate(datamodule.train_dataloader()))
data.keys()
# Output: dict_keys(['image', 'video_path', 'frames', 'last_frame', 'original_i

i, data = next(enumerate(datamodule.test_dataloader()))
data.keys()
# Output: dict_keys(['image', 'mask', 'video_path', 'frames', 'last_frame', 'or

data["image"].shape
# Output: torch.Size([32, 2, 3, 256, 256])
```

Note that the default task type is segmentation and the dataloader returns a mask in addition to the input. Also, it is important to note that the dataloader returns a batch of clips, where each clip is a sequence of frames. The number of frames in each clip is determined by the `clip_length_in_frames` parameter. The `frames_between_clips` parameter determines the number of frames between each consecutive clip. The `target_frame` parameter determines which frame in the clip is used for ground truth retrieval. For example, if `clip_length_in_frames=2`, `frames_between_clips=1` and `target_frame=VideoTargetFrame.LAST`, then the dataloader will return a batch of clips where each clip contains two consecutive frames from the video. The second frame in each clip will be used as the ground truth for the first frame in the clip. The following code shows how to create a dataloader for classification:

```
datamodule = Avenue(
    task="classification",
    clip_length_in_frames=2,
    frames_between_clips=1,
    target_frame=VideoTargetFrame.LAST
)
datamodule.setup()

i, data = next(enumerate(datamodule.train_dataloader()))
data.keys()
# Output: dict_keys(['image', 'video_path', 'frames', 'last_frame', 'original_i

data["image"].shape
# Output: torch.Size([32, 2, 3, 256, 256])
```

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

After:

```
$ tree datasets
datasets
├── dataset1
├── dataset2
└── avenue
    ├── ground_truth_demo
    │   ├── ground_truth_show.m
    │   ├── Readme.txt
    │   ├── testing_label_mask
    │   └── testing_videos
    ├── testing_videos
    │   ├── ...
    │   └── 21.avi
    ├── testing_vol
    │   ├── ...
    │   └── vol21.mat
    ├── training_videos
    │   ├── ...
    │   └── 16.avi
    └── training_vol
```

```
        ├── ...
        └── vol16.mat
```

class **anomalib.data.video.avenue.AvenueDataset(*task*, *split*, *root='./datasets/avenue'*, *gt_dir='./datasets/avenue/ground_truth_demo'*, *clip_length_in_frames=2*, *frames_between_clips=1*, *transform=None*, *target_frame=VideoTargetFrame.LAST*)**

Bases: `AnomalibVideoDataset`

Avenue Dataset class.

Parameters:

- **task** (*TaskType*) – Task type, 'classification', 'detection' or 'segmentation'
- **split** (*Split*) – Split of the dataset, usually Split.TRAIN or Split.TEST
- **root** (*Path* | *str*) – Path to the root of the dataset Defaults to `./datasets/avenue`.
- **gt_dir** (*Path* | *str*) – Path to the ground truth files Defaults to `./datasets/avenue/ground_truth_demo`.
- **clip_length_in_frames** (*int, optional*) – Number of video frames in each clip. Defaults to `2`.
- **frames_between_clips** (*int, optional*) – Number of frames between each consecutive video clip. Defaults to `1`.
- **target_frame** (*VideoTargetFrame*) – Specifies the target frame in the video clip, used for ground truth retrieval. Defaults to `VideoTargetFrame.LAST`.
- **transform** (*Transform, optional*) – Transforms that should be applied to the input images. Defaults to `None`.

Examples

To create an Avenue dataset to train a classification model:

```
transform = A.Compose([A.Resize(256, 256), A.pytorch.ToTensorV2()])
dataset = AvenueDataset(
    task="classification",
    transform=transform,
    split="train",
    root="./datasets/avenue/",
```

```
)

dataset.setup()
dataset[0].keys()

# Output: dict_keys(['image', 'video_path', 'frames', 'last_frame', 'original_i
```

If you would like to test a segmentation model, you can use the following code:

```
dataset = AvenueDataset(
    task="segmentation",
    transform=transform,
    split="test",
    root="./datasets/avenue/",
)

dataset.setup()
dataset[0].keys()

# Output: dict_keys(['image', 'mask', 'video_path', 'frames', 'last_frame', 'or
```

Avenue video dataset can also be used as an image dataset if you set the clip length to 1. This means that each video frame will be treated as a separate sample. This is useful for training a classification model on the Avenue dataset. The following code shows how to create an image dataset for classification:

```
dataset = AvenueDataset(
    task="classification",
    transform=transform,
    split="test",
    root="./datasets/avenue/",
    clip_length_in_frames=1,
)

dataset.setup()
dataset[0].keys()
# Output: dict_keys(['image', 'video_path', 'frames', 'last_frame', 'original_i

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

anomalib.data.video.avenue.make_avenue_dataset(*root*, *gt_dir*, *split=None*)

Create CUHK Avenue dataset by parsing the file structure.

The files are expected to follow the structure:

- path/to/dataset/[training_videos|testing_videos]/video_filename.avi
- path/to/ground_truth/mask_filename.mat

Parameters:

- **root** (*Path*) – Path to dataset
- **gt_dir** (*Path*) – Path to the ground truth
- **split** (*Split* | *str* | *None = None, optional*) – Dataset split (ie., either train or test). Defaults to `None`.

Example

The following example shows how to get testing samples from Avenue dataset:

```
>>> root = Path('./avenue')
>>> gt_dir = Path('./avenue/masks')
>>> samples = make_avenue_dataset(path, gt_dir, split='test')
>>> samples.head()
   root     folder         image_path                      mask_path
0  ./avenue testing_videos ./avenue/training_videos/01.avi ./avenue/masks/01_la
1  ./avenue testing_videos ./avenue/training_videos/02.avi ./avenue/masks/01_la
...
```

Returns:

an output dataframe containing samples for the requested split (ie., train or test)

Return type:

DataFrame

Previous
Video Data

Next
Shanghai Tech Data