MMEditing

MMEditing Authors

2022 年 12 月 08 日

开始你的第一步

您可以在页面左下角切换中英文文档。

备注： 目前英文版有更多的内容，如果您希望帮助我们翻译一部分文档，可以通过 issue 联系我们。

CHAPTER 1

概述（待更新）

CHAPTER 2

入门指引（待更新）

CHAPTER 3

配置文件（待更新）

CHAPTER 4

数据集（待更新）

CHAPTER 5

推理（待更新）

CHAPTER 6

训练 & 测试（待更新）

CHAPTER 7

实用工具（待更新）

CHAPTER 8

可视化（待更新）

CHAPTER 9

自定义模型（待更新）

CHAPTER 10

自定义数据集（待更新）

CHAPTER 11

自定义数据预处理流程（待更新）

CHAPTER 12

自定义损失函数（待更新）

CHAPTER 13

迁移（待更新）

CHAPTER 14

mmedit.apis

CHAPTER 15

mmedit.datasets

15.1 datasets

class mmedit.datasets.**AdobeComp1kDataset**(*ann_file: str = ''*, *metainfo: Optional[dict] = None*, *data_root: str = ''*, *data_prefix: dict = {'img_path': ''}*, *filter_cfg: Optional[dict] = None*, *indices: Optional[Union[int, Sequence[int]]] = None*, *serialize_data: bool = True*, *pipeline: List[Union[dict, Callable]] = []*, *test_mode: bool = False*, *lazy_init: bool = False*, *max_refetch: int = 1000*)

Adobe composition-1k dataset.

The dataset loads (alpha, fg, bg) data and apply specified transforms to the data. You could specify whether composite merged image online or load composited merged image in pipeline.

Example for online comp-1k dataset:

```
[
    {
        "alpha": 'alpha/000.png',
        "fg": 'fg/000.png',
        "bg": 'bg/000.png'
    },
    {
```

(下页继续)

(续上页)

```
        "alpha": 'alpha/001.png',
        "fg": 'fg/001.png',
        "bg": 'bg/001.png'
    },
]
```

Example for offline comp-1k dataset:

```
[
    {
        "alpha": 'alpha/000.png',
        "merged": 'merged/000.png',
        "fg": 'fg/000.png',
        "bg": 'bg/000.png'
    },
    {
        "alpha": 'alpha/001.png',
        "merged": 'merged/001.png',
        "fg": 'fg/001.png',
        "bg": 'bg/001.png'
    },
]
```

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **data_root** (*str, optional*) –The root directory for `data_prefix` and `ann_file`. Defaults to None.
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- ****kwargs** –Other arguments passed to `mmengine.dataset.BaseDataset`.

实际案例

See unit-tests TODO: Move some codes in unittest here

load_data_list() → List[dict]

Load annotations from an annotation file named as `self.ann_file`

In order to be compoatible to both new and old annotation format, we copy implementations from mmengine and do some modificatoins.

返回 A list of annotation.

返回类型 list[dict]

parse_data_info(*raw_data_info: dict*) → Union[dict, List[dict]]

Join data_root to each path in data_info.

class mmedit.datasets.**BasicConditionalDataset**(*ann_file: str = ''*, *metainfo: Optional[dict] = None*, *data_root: str = ''*, *data_prefix: Union[str, dict] = ''*, *extensions: Sequence[str] = ('.jpg', '.jpeg', '.png', '.ppm', '.bmp', '.pgm', '.tif')*, *lazy_init: bool = False*, *classes: Optional[Union[str, Sequence[str]]] = None*, ***kwargs*)

Custom dataset for conditional GAN. This class is the combination of *BaseDataset* (https://github.com/open- mmlab/mmclassification/blob/1.x/mmcls/datasets/base_dataset.py) # noqa and *CustomDataset* (https://github.com/open- mmlab/mmclassification/blob/1.x/mmcls/datasets/custom.py). # noqa.

The dataset supports two kinds of annotation format.

1. An annotation file is provided, and each line indicates a sample:

   The sample files:

   ```
   data_prefix/
   ├── folder_1
   │   ├── xxx.png
   │   ├── xxy.png
   │   └── ...
   └── folder_2
       ├── 123.png
       ├── nsdf3.png
       └── ...
   ```

   The annotation file (the first column is the image path and the second column is the index of category):

   ```
   folder_1/xxx.png 0
   folder_1/xxy.png 1
   folder_2/123.png 5
   folder_2/nsdf3.png 3
   ...
   ```

   Please specify the name of categories by the argument `classes` or `metainfo`.

2. The samples are arranged in the specific way:

```
data_prefix/
├── class_x
│   ├── xxx.png
│   ├── xxy.png
│   └── ...
│       └── xxz.png
└── class_y
    ├── 123.png
    ├── nsdf3.png
    ├── ...
    └── asd932_.png
```

If the `ann_file` is specified, the dataset will be generated by the first way, otherwise, try the second way.

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for `data_prefix` and `ann_file`. Defaults to ''.
- **data_prefix** (*str | dict*) –Prefix for the data. Defaults to ''.
- **extensions** (*Sequence[str]*) –A sequence of allowed extensions. Defaults to ('.jpg', '.jpeg', '.png', '.ppm', '.bmp', '.pgm', '.tif').
- **lazy_init** (*bool*) –Whether to load annotation during instantiation. In some cases, such as visualization, only the meta information of the dataset is needed, which is not necessary to load annotation file. `Basedataset` can skip load annotations to save time by set `lazy_init=False`. Defaults to False.
- ****kwargs** –Other keyword arguments in `BaseDataset`.

property CLASSES

Return all categories names.

property class_to_idx

Map mapping class name to class index.

返回 mapping from class name to class index.

返回类型 dict

extra_repr() → List[str]

The extra repr information of the dataset.

full_init()

Load annotation file and set `BaseDataset._fully_initialized` to True.

get_cat_ids(*idx: int*) → List[int]

Get category id by index.

参数 **idx** (*int*) –Index of data.

返回 Image category of specified index.

返回类型 cat_ids (List[int])

get_gt_labels()

Get all ground-truth labels (categories).

返回 categories for all images.

返回类型 np.ndarray

property img_prefix

The prefix of images.

is_valid_file(*filename: str*) → bool

Check if a file is a valid sample.

load_data_list()

Load image paths and gt_labels.

class mmedit.datasets.**BasicFramesDataset**(*ann_file: str = ''*, *metainfo: Optional[dict] = None*, *data_root: Optional[str] = None*, *data_prefix: dict = {'img': ''}*, *pipeline: List[Union[dict, Callable]] = []*, *test_mode: bool = False*, *filename_tmpl: dict = {}*, *search_key: Optional[str] = None*, *file_client_args: Optional[str] = None*, *depth: int = 1*, *num_input_frames: Optional[int] = None*, *num_output_frames: Optional[int] = None*, *fixed_seq_len: Optional[int] = None*, *load_frames_list: dict = {}*, ***kwargs*)

BasicFramesDataset for open source projects in OpenMMLab/MMEditing.

This dataset is designed for low-level vision tasks with frames, such as video super-resolution and video frame interpolation.

The annotation file is optional.

If use annotation file, the annotation format can be shown as follows.

```
Case 1 (Vid4):

    calendar 41
```

(下页继续)

(续上页)

```
    city 34
    foliage 49
    walk 47

Case 2 (REDS):

    000/00000000.png (720, 1280, 3)
    000/00000001.png (720, 1280, 3)

Case 3 (Vimeo90k):

    00001/0266 (256, 448, 3)
    00001/0268 (256, 448, 3)
```

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ‘’.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str, optional*) –The root directory for `data_prefix` and `ann_file`. Defaults to None.
- **data_prefix** (*dict, optional*) –Prefix for training data. Defaults to dict(img=’’, gt=’’).
- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **filename_tmpl** (*str*) –Template for each filename. Note that the template excludes the file extension. Default: ‘{}’.
- **search_key** (*str*) –The key used for searching the folder to get data_list. Default: ‘gt’.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See `mmengine.fileio.FileClient` for details. Default: None.
- **depth** (*int*) –The depth of path. Default: 1
- **num_input_frames** (*None | int*) –Number of input frames. Default: None.
- **num_output_frames** (*None | int*) –Number of output frames. Default: None.
- **fixed_seq_len** (*None | int*) –The fixed sequence length. If None, BasicFramesDataset will obtain the length of each sequence. Default: None.
- **load_frames_list** (*dict*) –Load frames list for each key. Default: dict().

实际案例

Assume the file structure as the following:

mmediting (root) ├── mmedit ├── tools ├── configs ├── data │ ├── Vid4 │ │ ├── BIx4 │ │ │ ├── city │ │ │ │ ├── img1.png │ │ ├── GT │ │ │ ├── city │ │ │ │ ├── img1.png │ │ ├── meta_info_Vid4_GT.txt │ ├── places │ │ ├── sequences ||| ├── 00001 │ │ │ ├── 0389 │ │ │ │ ├── img1.png │ │ │ │ ├── img2.png │ │ │ │ ├── img3.png │ │ ├── tri_trainlist.txt

Case 1: Loading Vid4 dataset for training a VSR model.

```
dataset = BasicFramesDataset(
    ann_file='meta_info_Vid4_GT.txt',
    metainfo=dict(dataset_type='vid4', task_name='vsr'),
    data_root='data/Vid4',
    data_prefix=dict(img='BIx4', gt='GT'),
    pipeline=[],
    depth=2,
    num_input_frames=5)
```

Case 2: Loading Vimeo90k dataset for training a VFI model.

```
dataset = BasicFramesDataset(
    ann_file='tri_trainlist.txt',
    metainfo=dict(dataset_type='vimeo90k', task_name='vfi'),
    data_root='data/vimeo-triplet',
    data_prefix=dict(img='sequences', gt='sequences'),
    pipeline=[],
    depth=2,
    load_frames_list=dict(
        img=['img1.png', 'img3.png'], gt=['img2.png']))
```

See more details in unittest

tests/test_datasets/test_base_frames_dataset.py TestFramesDatasets().test_version_1_method()

load_data_list() → List[dict]

Load data list from folder or annotation file.

返回 A list of annotation.

返回类型 list[dict]

class mmedit.datasets.**BasicImageDataset**(*ann_file: str = ''*, *metainfo: Optional[dict] = None*, *data_root: Optional[str] = None*, *data_prefix: dict = {'img': ''}*, *pipeline: List[Union[dict, Callable]] = []*, *test_mode: bool = False*, *filename_tmpl: dict = {}*, *search_key: Optional[str] = None*, *file_client_args: Optional[dict] = None*, *img_suffix: Optional[Union[str, Tuple[str]]] = ('.jpg', '.JPG', '.jpeg', '.JPEG', '.png', '.PNG', '.ppm', '.PPM', '.bmp', '.BMP', '.tif', '.TIF', '.tiff', '.TIFF')*, *recursive: bool = False*, ***kwards*)

BasicImageDataset for open source projects in OpenMMLab/MMEditing.

This dataset is designed for low-level vision tasks with image, such as super-resolution and inpainting.

The annotation file is optional.

If use annotation file, the annotation format can be shown as follows.

```
Case 1 (CelebA-HQ):

    000001.png
    000002.png

Case 2 (DIV2K):

    0001_s001.png (480,480,3)
    0001_s002.png (480,480,3)
    0001_s003.png (480,480,3)
    0002_s001.png (480,480,3)
    0002_s002.png (480,480,3)

Case 3 (Vimeo90k):

    00001/0266 (256, 448, 3)
    00001/0268 (256, 448, 3)
```

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str, optional*) –The root directory for `data_prefix` and `ann_file`. Defaults to None.
- **data_prefix** (*dict, optional*) –Prefix for training data. Defaults to dict(img=None, ann=None).

- **pipeline** (*list, optional*) –Processing pipeline. Defaults to [].
- **test_mode** (*bool, optional*) –`test_mode=True` means in test phase. Defaults to False.
- **filename_tmpl** (*dict*) –Template for each filename. Note that the template excludes the file extension. Default: dict().
- **search_key** (*str*) –The key used for searching the folder to get data_list. Default: 'gt'.
- **file_client_args** (*dict, optional*) –Arguments to instantiate a FileClient. See `mmengine.fileio.FileClient` for details. Default: None.
- **suffix** (*str or tuple[str], optional*) –File suffix that we are interested in. Default: None.
- **recursive** (*bool*) –If set to True, recursively scan the directory. Default: False.

备注： Assume the file structure as the following:

```
mmediting (root)
├── mmedit
├── tools
├── configs
├── data
|   ├── DIV2K
|   |   ├── DIV2K_train_HR
|   |   |   ├── image.png
|   |   ├── DIV2K_train_LR_bicubic
|   |   |   ├── X2
|   |   |   ├── X3
|   |   |   ├── X4
|   |   |   |   ├── image_x4.png
|   |   ├── DIV2K_valid_HR
|   |   ├── DIV2K_valid_LR_bicubic
|   |   |   ├── X2
|   |   |   ├── X3
|   |   |   ├── X4
|   ├── places
|   |   ├── test_set
|   |   ├── train_set
|   |   ├── meta
|   |   |    ├── Places365_train.txt
|   |   |    ├── Places365_val.txt
```

实际案例

Case 1: Loading DIV2K dataset for training a SISR model.

```
dataset = BasicImageDataset(
    ann_file='',
    metainfo=dict(
        dataset_type='div2k',
        task_name='sisr'),
    data_root='data/DIV2K',
    data_prefix=dict(
        gt='DIV2K_train_HR', img='DIV2K_train_LR_bicubic/X4'),
    filename_tmpl=dict(img='{}_x4', gt='{}'),
    pipeline=[])
```

Case 2: Loading places dataset for training an inpainting model.

```
dataset = BasicImageDataset(
    ann_file='meta/Places365_train.txt',
    metainfo=dict(
        dataset_type='places365',
        task_name='inpainting'),
    data_root='data/places',
    data_prefix=dict(gt='train_set'),
    pipeline=[])
```

load_data_list() → List[dict]

Load data list from folder or annotation file.

返回 A list of annotation.

返回类型 list[dict]

class mmedit.datasets.**CIFAR10**(*data_prefix: str*, *test_mode: bool*, *metainfo: Optional[dict] = None*, *data_root: str = ''*, *download: bool = True*, ***kwargs*)

CIFAR10 Dataset.

This implementation is modified from https://github.com/pytorch/vision/blob/master/torchvision/datasets/cifar.py

参数

- **data_prefix** (*str*) –Prefix for data.
- **test_mode** (*bool*) –`test_mode=True` means in test phase. It determines to use the training set or test set.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as categories information. Defaults to None.

- **data_root** (*str*) –The root directory for `data_prefix`. Defaults to `''`.
- **download** (*bool*) –Whether to download the dataset if not exists. Defaults to True.
- ****kwargs** –Other keyword arguments in `BaseDataset`.

extra_repr() → List[str]

The extra repr information of the dataset.

load_data_list()

Load images and ground truth labels.

class mmedit.datasets.**GrowScaleImgDataset**(*data_roots: dict*, *pipeline*, *len_per_stage=1000000*, *gpu_samples_per_scale=None*, *gpu_samples_base=32*, *io_backend: Optional[str] = None*, *file_lists: Optional[Union[str, dict]] = None*, *test_mode=False*)

Grow Scale Unconditional Image Dataset.

This dataset is similar with `UnconditionalImageDataset`, but offer more dynamic functionalities for the supporting complex algorithms, like PGGAN.

Highlight functionalities:

1. Support growing scale dataset. The motivation is to decrease data pre-processing load in CPU. In this dataset, you can provide `imgs_roots` like:

```
{'64': 'path_to_64x64_imgs',
 '512': 'path_to_512x512_imgs'}
```

   Then, in training scales lower than 64x64, this dataset will set `self.imgs_root` as 'path_to_64x64_imgs' ;

2. Offer `samples_per_gpu` according to different scales. In this dataset, `self.samples_per_gpu` will help runner to know the updated batch size.

Basically, This dataset contains raw images for training unconditional GANs. Given a root dir, we will recursively find all images in this root. The transformation on data is defined by the pipeline.

参数

- **imgs_root** (*str*) –Root path for unconditional images.
- **pipeline** (*list[dict | callable]*) –A sequence of data transforms.
- **len_per_stage** (*int, optional*) –The length of dataset for each scale. This args change the length dataset by concatenating or extracting subset. If given a value less than 0., the original length will be kept. Defaults to 1e6.
- **gpu_samples_per_scale** (*dict | None, optional*) –Dict contains `samples_per_gpu` for each scale. For example, `{'32': 4}` will set the scale of

32 with `samples_per_gpu=4`, despite other scale with `samples_per_gpu=self.gpu_samples_base`.

- **gpu_samples_base** (*int, optional*) –Set default `samples_per_gpu` for each scale. Defaults to 32.
- **io_backend** (*str, optional*) –The storage backend type. Options are "disk", "ceph", "memcached", "lmdb", "http" and "petrel". Default: None.
- **test_mode** (*bool, optional*) –If True, the dataset will work in test mode. Otherwise, in train mode. Default to False.

concat_imgs_list_to(*num*)

Concat image list to specified length.

参数 **num** (*int*) –The length of the concatenated image list.

load_data_list()

Load annotations.

prepare_test_data(*idx*)

Prepare testing data.

参数 **idx** (*int*) –Index of current batch.

返回 Prepared training data batch.

返回类型 dict

prepare_train_data(*idx*)

Prepare training data.

参数 **idx** (*int*) –Index of current batch.

返回 Prepared training data batch.

返回类型 dict

update_annotations(*curr_scale*)

Update annotations.

参数 **curr_scale** (*int*) –Current image scale.

返回 Whether to update.

返回类型 bool

class mmedit.datasets.**ImageNet**(*ann_file: str = '', metainfo: Optional[dict] = None, data_root: str = '', data_prefix: Union[str, dict] = '', **kwargs*)

ImageNet Dataset.

The dataset supports two kinds of annotation format. More details can be found in `CustomDataset`.

参数

- **ann_file** (*str*) –Annotation file path. Defaults to ''.
- **metainfo** (*dict, optional*) –Meta information for dataset, such as class information. Defaults to None.
- **data_root** (*str*) –The root directory for `data_prefix` and `ann_file`. Defaults to ''.
- **data_prefix** (*str | dict*) –Prefix for training data. Defaults to ''.
- ****kwargs** –Other keyword arguments in `CustomDataset` and `BaseDataset`.

class mmedit.datasets.**PairedImageDataset**(*data_root*, *pipeline*, *io_backend: Optional[str] = None*, *test_mode=False*, *test_dir='test'*)

General paired image folder dataset for image generation.

It assumes that the training directory is '/path/to/data/train'. During test time, the directory is '/path/to/data/test'. '/path/to/data' can be initialized by args 'dataroot'. Each sample contains a pair of images concatenated in the w dimension (A|B).

参数

- **dataroot** (str | `Path`) –Path to the folder root of paired images.
- **pipeline** (*List[dict | callable]*) –A sequence of data transformations.
- **test_mode** (*bool*) –Store *True* when building test dataset. Default: *False*.
- **test_dir** (*str*) –Subfolder of dataroot which contain test images. Default: 'test'.

load_data_list()

Load paired image paths.

返回 List that contains paired image paths.

返回类型 list[dict]

scan_folder(*path*)

Obtain image path list (including sub-folders) from a given folder.

参数 **path** (str | `Path`) –Folder path.

返回 Image list obtained from the given folder.

返回类型 list[str]

class mmedit.datasets.**SinGANDataset**(*data_root*, *min_size*, *max_size*, *scale_factor_init*, *pipeline*, *num_samples=-1*)

SinGAN Dataset.

In this dataset, we create an image pyramid and save it in the cache.

参数

- **img_path** (*str*) –Path to the single image file.
- **min_size** (*int*) –Min size of the image pyramid. Here, the number will be set to the `min(H, W)`.
- **max_size** (*int*) –Max size of the image pyramid. Here, the number will be set to the `max(H, W)`.
- **scale_factor_init** (*float*) –Rescale factor. Note that the actual factor we use may be a little bit different from this value.
- **num_samples** (*int, optional*) –The number of samples (length) in this dataset. Defaults to -1.

full_init()

Skip the full init process for SinGANDataset.

load_data_list(*min_size*, *max_size*, *scale_factor_init*)

Load annatations for SinGAN Dataset.

参数

- **min_size** (*int*) –The minimum size for the image pyramid.
- **max_size** (*int*) –The maximum size for the image pyramid.
- **scale_factor_init** (*float*) –The initial scale factor.

class mmedit.datasets.**UnpairedImageDataset**(*data_root*, *pipeline*, *io_backend: Optional[str] = None*, *test_mode=False*, *domain_a='A'*, *domain_b='B'*)

General unpaired image folder dataset for image generation.

It assumes that the training directory of images from domain A is '/path/to/data/trainA', and that from domain B is '/path/to/data/trainB', respectively. '/path/to/data' can be initialized by args 'dataroot'. During test time, the directory is '/path/to/data/testA' and '/path/to/data/testB', respectively.

参数

- **dataroot** (str | Path) –Path to the folder root of unpaired images.
- **pipeline** (*List[dict | callable]*) –A sequence of data transformations.
- **io_backend** (*str, optional*) –The storage backend type. Options are "disk", "ceph", "memcached", "lmdb", "http" and "petrel". Default: None.
- **test_mode** (*bool*) –Store *True* when building test dataset. Default: *False*.
- **domain_a** (*str, optional*) –Domain of images in trainA / testA. Defaults to 'A'.
- **domain_b** (*str, optional*) –Domain of images in trainB / testB. Defaults to 'B'.

get_data_info(*idx*) → dict

Get annotation by index and automatically call `full_init` if the dataset has not been fully initialized.

参数 **idx** (*int*) –The index of data.

返回 The idx-th annotation of the dataset.

返回类型 dict

load_data_list()

Load the data list.

返回 The data info list of source and target domain.

返回类型 list

scan_folder(*path*)

Obtain image path list (including sub-folders) from a given folder.

参数 **path** (str | `Path`) –Folder path.

返回 Image list obtained from the given folder.

返回类型 list[str]

15.2 transforms

class mmedit.datasets.transforms.**BinarizeImage**(*keys*, *binary_thr*, *a_min=0*, *a_max=1*, *dtype=<class 'numpy.uint8'>*)

Binarize image.

参数

- **keys** (*Sequence[str]*) –The images to be binarized.
- **binary_thr** (*float*) –Threshold for binarization.
- **amin** (*int*) –Lower limits of pixel value.
- **amx** (*int*) –Upper limits of pixel value.
- **dtype** (*np.dtype*) –Set the data type of the output. Default: np.uint8

transform(*results*)

The transform function of BinarizeImage.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**CenterCropLongEdge**(*keys='img'*)

Center crop the given image by the long edge.

参数 keys (*list[str]*) –The images to be cropped.

transform(*results*)

Call function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**Clip**(*keys*, *a_min=0*, *a_max=255*)

Clip the pixels.

Modified keys are the attributes specified in "keys".

参数

- **keys** (*list[str]*) –The keys whose values are clipped.
- **amin** (*int*) –Lower limits of pixel value.
- **amx** (*int*) –Upper limits of pixel value.

transform(*results*)

transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回

A dict with the values of the specified keys are rounded and clipped.

返回类型 dict

class mmedit.datasets.transforms.**ColorJitter**(*keys*, *channel_order='rgb'*, ***kwargs*)

An interface for torch color jitter so that it can be invoked in mmediting pipeline.

Randomly change the brightness, contrast and saturation of an image. Modified keys are the attributes specified in "keys".

Required Keys:

- [KEYS]

Modified Keys:

- [KEYS]

参数

- **keys** (*list[str]*) –The images to be resized.

- **channel_order** (*str*) –Order of channel, candidates are ‘bgr’ and ‘rgb’. Default: ‘rgb’.

提示

`**kwards` follows the args list of `torchvision.transforms.ColorJitter`.

brightness (float or tuple of float (min, max)): How much to jitter brightness. brightness_factor is chosen uniformly from [max(0, 1 - brightness), 1 + brightness] or the given [min, max]. Should be non negative numbers.

contrast (float or tuple of float (min, max)): How much to jitter contrast. contrast_factor is chosen uniformly from [max(0, 1 - contrast), 1 + contrast] or the given [min, max]. Should be non negative numbers.

saturation (float or tuple of float (min, max)): How much to jitter saturation. saturation_factor is chosen uniformly from [max(0, 1 - saturation), 1 + saturation] or the given [min, max]. Should be non negative numbers.

hue (float or tuple of float (min, max)): How much to jitter hue. hue_factor is chosen uniformly from [-hue, hue] or the given [min, max]. Should have 0<= hue <= 0.5 or -0.5 <= min <= max <= 0.5.

transform(*results: Dict*) → Dict

The transform function of ColorJitter.

参数 **results** (*dict*) –The result dict.

返回 The result dict.

返回类型 dict

class mmedit.datasets.transforms.**CompositeFg**(*fg_dirs*, *alpha_dirs*, *interpolation='nearest'*)

Composite foreground with a random foreground.

This class composites the current training sample with additional data randomly (could be from the same dataset). With probability 0.5, the sample will be composited with a random sample from the specified directory. The composition is performed as:

$$fg_{new} = \alpha_1 * fg_1 + (1 - \alpha_1) * fg_2$$
$$\alpha_{new} = 1 - (1 - \alpha_1) * (1 - \alpha_2)$$

where (fg_1, α_1) is from the current sample and (fg_2, α_2) is the randomly loaded sample. With the above composition, α_{new} is still in *[0, 1]*.

Required keys are “alpha” and “fg”. Modified keys are “alpha” and “fg”.

参数

- **fg_dirs** (*str | list[str]*) –Path of directories to load foreground images from.
- **alpha_dirs** (*str | list[str]*) –Path of directories to load alpha mattes from.

- **interpolation** (*str*) –Interpolation method of *mmcv.imresize* to resize the randomly loaded images. Default: ‘nearest’.

transform(*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**CopyValues**(*src_keys*, *dst_keys*)

Copy the value of source keys to destination keys.

TODO Change to dict(dst=src)

It does the following: results[dst_key] = results[src_key] for (src_key, dst_key) in zip(src_keys, dst_keys).

Added keys are the keys in the attribute “dst_keys”.

Required Keys:

- [SRC_KEYS]

Added Keys:

- [DST_KEYS]

参数

- **src_keys** (*list[str]*) –The source keys.
- **dst_keys** (*list[str]*) –The destination keys.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict with a key added/modified.

返回类型 dict

class mmedit.datasets.transforms.**Crop**(*keys*, *crop_size*, *random_crop=True*, *is_pad_zeros=False*)

Crop data to specific size for training.

参数

- **keys** (*Sequence[str]*) –The images to be cropped.
- **crop_size** (*Tuple[int]*) –Target spatial size (h, w).
- **random_crop** (*bool*) –If set to True, it will random crop image. Otherwise, it will work as center crop. Default: True.

- **is_pad_zeros** (*bool, optional*) –Whether to pad the image with 0 if crop_size is greater than image size. Default: False.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**CropAroundCenter**(*crop_size*)

Randomly crop the images around unknown area in the center 1/4 images.

This cropping strategy is adopted in GCA matting. The *unknown area* is the same as *semi-transparent area*. https://arxiv.org/pdf/2001.04069.pdf

It retains the center 1/4 images and resizes the images to 'crop_size'. Required keys are "fg", "bg", "trimap" and "alpha", added or modified keys are "crop_bbox", "fg", "bg", "trimap" and "alpha".

参数 **crop_size** (*int | tuple*) –Desired output size. If int, square crop is applied.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**CropAroundFg**(*keys*, *bd_ratio_range=(0.1, 0.4)*, *test_mode=False*)

Crop around the whole foreground in the segmentation mask.

Required keys are "seg" and the keys in argument *keys*. Meanwhile, "seg" must be in argument *keys*. Added or modified keys are "crop_bbox" and the keys in argument *keys*.

参数

- **keys** (*Sequence[str]*) –The images to be cropped. It must contain 'seg'.
- **bd_ratio_range** (*tuple, optional*) –The range of the boundary (bd) ratio to select from. The boundary ratio is the ratio of the boundary to the minimal bbox that contains the whole foreground given by segmentation. Default to (0.1, 0.4).
- **test_mode** (*bool*) –Whether use test mode. In test mode, the tight crop area of foreground will be extended to the a square. Default to False.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**CropAroundUnknown**(*keys*, *crop_sizes*, *unknown_source='alpha'*, *interpolations='bilinear'*)

Crop around unknown area with a randomly selected scale.

Randomly select the w and h from a list of (w, h). Required keys are the keys in argument *keys*, added or modified keys are "crop_bbox" and the keys in argument *keys*. This class assumes value of "alpha" ranges from 0 to 255.

参数

- **keys** (*Sequence[str]*) –The images to be cropped. It must contain 'alpha'. If unknown_source is set to 'trimap', then it must also contain 'trimap'.
- **crop_sizes** (*list[int | tuple[int]]*) –List of (w, h) to be selected.
- **unknown_source** (*str, optional*) –Unknown area to select from. It must be 'alpha' or 'trimap'. Default to 'alpha'.
- **interpolations** (*str | list[str], optional*) –Interpolation method of mmcv.imresize. The interpolation operation will be applied when image size is smaller than the crop_size. If given as a list of str, it should have the same length as *keys*. Or if given as a str all the keys will be resized with the same method. Default to 'bilinear'.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**CropLike**(*target_key*, *reference_key=None*)

Crop/pad the image in the target_key according to the size of image in the reference_key .

参数

- **target_key** (*str*) –The key needs to be cropped.
- **reference_key** (*str | None*) –The reference key, need its size. Default: None.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation. Require self.target_key and self.reference_key.

返回

A dict containing the processed data and information. Modify self.target_key.

返回类型 dict

class mmedit.datasets.transforms.**DegradationsWithShuffle**(*degradations*, *keys*, *shuffle_idx=None*)

Apply random degradations to input, with degradations being shuffled.

Degradation groups are supported. The order of degradations within the same group is preserved. For example, if we have degradations = [a, b, [c, d]] and shuffle_idx = None, then the possible orders are

```
[a, b, [c, d]]
[a, [c, d], b]
[b, a, [c, d]]
[b, [c, d], a]
[[c, d], a, b]
[[c, d], b, a]
```

Modified keys are the attributed specified in "keys".

参数

- **degradations** (*list[dict]*) –The list of degradations.
- **keys** (*list[str]*) –A list specifying the keys whose values are modified.
- **shuffle_idx** (*list | None, optional*) –The degradations corresponding to these indices are shuffled. If None, all degradations are shuffled. Default: None.

class mmedit.datasets.transforms.**FixedCrop**(*keys*, *crop_size*, *crop_pos=None*)

Crop paired data (at a specific position) to specific size for training.

参数

- **keys** (*Sequence[str]*) –The images to be cropped.
- **crop_size** (*Tuple[int]*) –Target spatial size (h, w).
- **crop_pos** (*Tuple[int]*) –Specific position (x, y). If set to None, random initialize the position to crop paired data batch. Default: None.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**Flip**(*keys*, *flip_ratio=0.5*, *direction='horizontal'*)

Flip the input data with a probability.

Reverse the order of elements in the given data with a specific direction. The shape of the data is preserved, but the elements are reordered. Required keys are the keys in attributes "keys", added or modified keys are "flip", "flip_direction" and the keys in attributes "keys". It also supports flipping a list of images with the same flip.

Required Keys:

- [KEYS]

Modified Keys:

- [KEYS]

参数

- **keys** (*Union[str, List[str]]*) –The images to be flipped.
- **flip_ratio** (*float*) –The probability to flip the images. Default: 0.5.
- **direction** (*str*) –Flip images horizontally or vertically. Options are "horizontal" | "vertical". Default: "horizontal".

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**FormatTrimap**(*to_onehot=False*)

Convert trimap (tensor) to one-hot representation.

It transforms the trimap label from (0, 128, 255) to (0, 1, 2). If `to_onehot` is set to True, the trimap will convert to one-hot tensor of shape (3, H, W). Required key is "trimap", added or modified key are "trimap" and "format_trimap_to_onehot".

参数 **to_onehot** (*bool*) –whether convert trimap to one-hot tensor. Default: `False`.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GenerateCoordinateAndCell**(*sample_quantity=None*, *scale=None*, *target_size=None*, *reshape_gt=True*)

Generate coordinate and cell. Generate coordinate from the desired size of SR image.

Train or val:

1. Generate coordinate from GT.

#. Reshape GT image to (HgWg, 3) and transpose to (3, HgWg). where *Hg* and *Wg* represent the height and width of GT.

Test:

1. Generate coordinate from LQ and scale or target_size.
2. Then generate cell from coordinate.

参数

- **sample_quantity** (*int | None*) –The quantity of samples in coordinates. To ensure that the GT tensors in a batch have the same dimensions. Default: None.
- **scale** (*float*) –Scale of upsampling. Default: None.
- **target_size** (*tuple[int]*) –Size of target image. Default: None.
- **reshape_gt** (*bool*) –Whether reshape gt to (-1, 3). Default: True If sample_quantity is not None, reshape_gt = True.

The priority of getting ‘size of target image’ is:

1. results[‘gt’].shape[-2:]
2. results[‘lq’].shape[-2:] * scale
3. target_size

transform(*results*)

Call function.

参数

- **results** (*Require either in*) –A dict containing the necessary information
- **augmentation.** (*and data for*) –
- **results** –
- **'lq'** (*1.*) –
- **'gt'** (*2.*) –

- **None** (*3.*) –
- **and** (*the premise is self.target_size*) –
- **len** (*self.target_size*) –

返回 A dict containing the processed data and information. Reshape ‘gt’ to (-1, 3) and transpose to (3, -1) if ‘gt’ in results. Add ‘coord’ and ‘cell’.

返回类型 dict

class mmedit.datasets.transforms.**GenerateFacialHeatmap**(*image_key*, *ori_size*, *target_size*, *sigma=1.0*, *use_cache=True*)

Generate heatmap from keypoint.

参数

- **image_key** (*str*) –Key of facial image in dict.
- **ori_size** (*int | Tuple[int]*) –Original image size of keypoint.
- **target_size** (*int | Tuple[int]*) –Target size of heatmap.
- **sigma** (*float*) –Sigma parameter of heatmap. Default: 1.0
- **use_cache** (*bool*) –If True, load all heatmap at once. Default: True.

generate_heatmap_from_img(*image*)

Generate heatmap from img.

参数 image (*np.ndarray*) –Face image.

results: heatmap (np.ndarray): Heatmap the face image.

transform(*results*)

transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation. Require keypoint.

返回

A dict containing the processed data and information. Add ‘heatmap’.

返回类型 dict

class mmedit.datasets.transforms.**GenerateFrameIndices**(*interval_list*, *frames_per_clip=99*)

Generate frame index for REDS datasets. It also performs temporal augmention with random interval.

Required Keys:

- img_path
- gt_path

- key
- num_input_frames

Modified Keys:

- img_path
- gt_path

Added Keys:

- interval
- reverse

参数

- **interval_list** (*list[int]*) –Interval list for temporal augmentation. It will randomly pick an interval from interval_list and sample frame index with the interval.
- **frames_per_clip** (*int*) –Number of frames per clips. Default: 99 for REDS dataset.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GenerateFrameIndiceswithPadding**(*padding*, *filename_tmpl='{:08d}'*)

Generate frame index with padding for REDS dataset and Vid4 dataset during testing.

Required Keys:

- img_path
- gt_path
- key
- num_input_frames
- sequence_length

Modified Keys:

- img_path
- gt_path

参数 **padding** –padding mode, one of ‘replicate’ | ‘reflection’ | ‘reflection_circle’ | ‘circle’.

Examples: current_idx = 0, num_input_frames = 5 The generated frame indices under different padding mode:

replicate: [0, 0, 0, 1, 2] reflection: [2, 1, 0, 1, 2] reflection_circle: [4, 3, 0, 1, 2] circle: [3, 4, 0, 1, 2]

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GenerateSeg**(*kernel_size=5*, *erode_iter_range=(10, 20)*, *dilate_iter_range=(15, 30)*, *num_holes_range=(0, 3)*, *hole_sizes=[(15, 15), (25, 25), (35, 35), (45, 45)]*, *blur_ksizes=[(21, 21), (31, 31), (41, 41)]*)

Generate segmentation mask from alpha matte.

参数

- **kernel_size** (*int, optional*) –Kernel size for both erosion and dilation. The kernel will have the same height and width. Defaults to 5.
- **erode_iter_range** (*tuple, optional*) –Iteration of erosion. Defaults to (10, 20).
- **dilate_iter_range** (*tuple, optional*) –Iteration of dilation. Defaults to (15, 30).
- **num_holes_range** (*tuple, optional*) –Range of number of holes to randomly select from. Defaults to (0, 3).
- **hole_sizes** (*list, optional*) –List of (h, w) to be selected as the size of the rectangle hole. Defaults to [(15, 15), (25, 25), (35, 35), (45, 45)].
- **blur_ksizes** (*list, optional*) –List of (h, w) to be selected as the kernel_size of the gaussian blur. Defaults to [(21, 21), (31, 31), (41, 41)].

transform(*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GenerateSegmentIndices**(*interval_list*, *start_idx=0*, *filename_tmpl='{:08d}.png'*)

Generate frame indices for a segment. It also performs temporal augmention with random interval.

Required Keys:

- img_path
- gt_path
- key
- num_input_frames
- sequence_length

Modified Keys:

- img_path
- gt_path

Added Keys:

- interval
- reverse

参数

- **interval_list** (*list[int]*) –Interval list for temporal augmentation. It will randomly pick an interval from interval_list and sample frame index with the interval.
- **start_idx** (*int*) –The index corresponds to the first frame in the sequence. Default: 0.
- **filename_tmpl** (*str*) –Template for file name. Default: '{:08d}.png'.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GenerateSoftSeg**(*fg_thr=0.2*, *border_width=25*, *erode_ksize=3*, *dilate_ksize=5*, *erode_iter_range=(10, 20)*, *dilate_iter_range=(3, 7)*, *blur_ksizes=[(21, 21), (31, 31), (41, 41)]*)

Generate soft segmentation mask from input segmentation mask.

Required key is "seg", added key is "soft_seg".

参数

- **fg_thr** (*float, optional*) –Threshold of the foreground in the normalized input segmentation mask. Defaults to 0.2.
- **border_width** (*int, optional*) –Width of border to be padded to the bottom of the mask. Defaults to 25.
- **erode_ksize** (*int, optional*) –Fixed kernel size of the erosion. Defaults to 5.
- **dilate_ksize** (*int, optional*) –Fixed kernel size of the dilation. Defaults to 5.
- **erode_iter_range** (*tuple, optional*) –Iteration of erosion. Defaults to (10, 20).
- **dilate_iter_range** (*tuple, optional*) –Iteration of dilation. Defaults to (3, 7).
- **blur_ksizes** (*list, optional*) –List of (h, w) to be selected as the kernel_size of the gaussian blur. Defaults to [(21, 21), (31, 31), (41, 41)].

transform (*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GenerateTrimap** (*kernel_size*, *iterations=1*, *random=True*)

Using random erode/dilate to generate trimap from alpha matte.

Required key is “alpha”, added key is “trimap”.

参数

- **kernel_size** (*int | tuple[int]*) –The range of random kernel_size of erode/dilate; int indicates a fixed kernel_size. If *random* is set to False and kernel_size is a tuple of length 2, then it will be interpreted as (erode kernel_size, dilate kernel_size). It should be noted that the kernel of the erosion and dilation has the same height and width.
- **iterations** (*int | tuple[int], optional*) –The range of random iterations of erode/dilate; int indicates a fixed iterations. If *random* is set to False and iterations is a tuple of length 2, then it will be interpreted as (erode iterations, dilate iterations). Default to 1.
- **random** (*bool, optional*) –Whether use random kernel_size and iterations when generating trimap. See *kernel_size* and *iterations* for more information. Default to True.

transform (*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GenerateTrimapWithDistTransform**(*dist_thr=20*, *random=True*)

Generate trimap with distance transform function.

参数

- **dist_thr** (*int, optional*) –Distance threshold. Area with alpha value between (0, 255) will be considered as initial unknown area. Then area with distance to unknown area smaller than the distance threshold will also be consider as unknown area. Defaults to 20.
- **random** (*bool, optional*) –If True, use random distance threshold from [1, dist_thr). If False, use *dist_thr* as the distance threshold directly. Defaults to True.

transform(*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GetMaskedImage**(*img_key='gt'*, *mask_key='mask'*, *out_key='img'*, *zero_value=127.5*)

Get masked image.

参数

- **img_key** (*str*) –Key for clean image. Default: 'gt'.
- **mask_key** (*str*) –Key for mask image. The mask shape should be (h, w, 1) while '1' indicate holes and '0' indicate valid regions. Default: 'mask'.
- **img_key** –Key for output image. Default: 'img'.
- **zero_value** (*float*) –Pixel value of masked area.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**GetSpatialDiscountMask**(*gamma=0.99*, *beta=1.5*)

Get spatial discounting mask constant.

Spatial discounting mask is first introduced in: Generative Image Inpainting with Contextual Attention.

参数

- **gamma** (*float, optional*) –Gamma for computing spatial discounting. Defaults to 0.99.
- **beta** (*float, optional*) –Beta for computing spatial discounting. Defaults to 1.5.

spatial_discount_mask(*mask_width*, *mask_height*)

Generate spatial discounting mask constant.

参数

- **mask_width** (*int*) –The width of bbox hole.
- **mask_height** (*int*) –The height of bbox height.

返回 Spatial discounting mask.

返回类型 np.ndarray

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**InstanceCrop**(*config_file*, *key='img'*, *box_num_upbound=-1*, *finesize=256*)

Use maskrcnn to detect instances on image.

Mask R-CNN is used to detect the instance on the image pred_bbox is used to segment the instance on the image

参数

- **config_file** (*str*) –config file name relative to detectron2' s "configs/"
- **key** (*str*) –Unused
- **box_num_upbound** (*int*) –The upper limit on the number of instances in the figure

transform(*results: dict*) → dict

The transform function of InstanceCrop.

参数 **results** (*dict*) –A dict containing the necessary information and data for Conversion

返回

A dict containing the processed data and information.

返回类型 results (dict)

class mmedit.datasets.transforms.**LoadImageFromFile**(*key: str*, *color_type: str = 'color'*, *channel_order: str = 'bgr'*, *imdecode_backend: Optional[str] = None*, *use_cache: bool = False*, *to_float32: bool = False*, *to_y_channel: bool = False*, *save_original_img: bool = False*, *file_client_args: Optional[dict] = None*)

Load a single image or image frames from corresponding paths. Required Keys: - [Key]_path

New Keys: - [KEY] - ori_[KEY]_shape - ori_[KEY]

参数

- **key** (*str*) –Keys in results to find corresponding path.
- **color_type** (*str*) –The flag argument for :func:mmcv.imfrombytes. Defaults to 'color'.
- **channel_order** (*str*) –Order of channel, candidates are 'bgr' and 'rgb'. Default: 'bgr'.
- **imdecode_backend** (*str*) –The image decoding backend type. The backend argument for :func:mmcv.imfrombytes. See :func:mmcv.imfrombytes for details. candidates are 'cv2', 'turbojpeg', 'pillow', and 'tifffile'. Defaults to None.
- **use_cache** (*bool*) –If True, load all images at once. Default: False.
- **to_float32** (*bool*) –Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **to_y_channel** (*bool*) –Whether to convert the loaded image to y channel. Only support 'rgb2ycbcr' and 'rgb2ycbcr' Defaults to False.
- **file_client_args** (*dict*) –Arguments to instantiate a FileClient. If not specified, will infer from file uri. See mmengine.fileio.FileClient for details. Defaults to None.

transform(*results: dict*) → dict

Functions to load image or frames.

参数 **results** (*dict*) –Result dict from :obj:mmcv.BaseDataset.

返回 The dict contains loaded image and meta information.

返回类型 dict

class mmedit.datasets.transforms.**LoadMask**(*mask_mode='bbox'*, *mask_config=None*)

Load Mask for multiple types.

For different types of mask, users need to provide the corresponding config dict.

Example config for bbox:

```
config = dict(img_shape=(256, 256), max_bbox_shape=128)
```

Example config for irregular:

```
config = dict(
    img_shape=(256, 256),
    num_vertices=(4, 12),
    max_angle=4.,
    length_range=(10, 100),
    brush_width=(10, 40),
    area_ratio_range=(0.15, 0.5))
```

Example config for ff:

```
config = dict(
    img_shape=(256, 256),
    num_vertices=(4, 12),
    mean_angle=1.2,
    angle_range=0.4,
    brush_width=(12, 40))
```

Example config for set:

```
config = dict(
    mask_list_file='xxx/xxx/ooxx.txt',
    prefix='/xxx/xxx/ooxx/',
    io_backend='disk',
    color_type='unchanged',
    file_client_kwargs=dict()
)

The mask_list_file contains the list of mask file name like this:
    test1.jpeg
    test2.jpeg
    ...
    ...

The prefix gives the data path.
```

参数

- **mask_mode** (*str*) –Mask mode in [‘bbox’, ‘irregular’, ‘ff’, ‘set’, ‘file’]. Default: ‘bbox’. * bbox: square bounding box masks. * irregular: irregular holes. * ff: free-form holes from DeepFillv2. * set: randomly get a mask from a mask set. * file: get mask from ‘mask_path’ in results.

- **mask_config** (*dict*) –Params for creating masks. Each type of mask needs different configs. Default: None.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**LoadPairedImageFromFile**(*key: str*, *domain_a: str = 'A'*, *domain_b: str = 'B'*, *color_type: str = 'color'*, *channel_order: str = 'bgr'*, *imdecode_backend: Optional[str] = None*, *use_cache: bool = False*, *to_float32: bool = False*, *to_y_channel: bool = False*, *save_original_img: bool = False*, *file_client_args: Optional[dict] = None*)

Load a pair of images from file.

Each sample contains a pair of images, which are concatenated in the w dimension (a|b). This is a special loading class for generation paired dataset. It loads a pair of images as the common loader does and crops it into two images with the same shape in different domains.

Required key is "pair_path". Added or modified keys are "pair", "pair_ori_shape", "ori_pair", "img_{domain_a}", "img_{domain_b}", "img_{domain_a}_path", "img_{domain_b}_path", "img_{domain_a}_ori_shape", "img_{domain_b}_ori_shape", "ori_img_{domain_a}" and "ori_img_{domain_b}".

参数

- **key** (*str*) –Keys in results to find corresponding path.
- **domain_a** (*str, Optional*) –One of the paired image domain. Defaults to 'A'.
- **domain_b** (*str, Optional*) –The other of the paired image domain. Defaults to 'B'.
- **color_type** (*str*) –The flag argument for :func:mmcv.imfrombytes. Defaults to 'color'.
- **channel_order** (*str*) –Order of channel, candidates are 'bgr' and 'rgb'. Default: 'bgr'.

- **imdecode_backend** (*str*) –The image decoding backend type. The backend argument for :func:`mmcv.imfrombytes`. See :func:`mmcv.imfrombytes` for details. candidates are 'cv2', 'turbojpeg', 'pillow', and 'tifffile'. Defaults to None.
- **use_cache** (*bool*) –If True, load all images at once. Default: False.
- **to_float32** (*bool*) –Whether to convert the loaded image to a float32 numpy array. If set to False, the loaded image is an uint8 array. Defaults to False.
- **to_y_channel** (*bool*) –Whether to convert the loaded image to y channel. Only support 'rgb2ycbcr' and 'rgb2ycbcr' Defaults to False.
- **file_client_args** (*dict*) –Arguments to instantiate a FileClient. If not specified, will infer from file uri. See `mmengine.fileio.FileClient` for details. Defaults to `None`.
- **io_backend** (*str, optional*) –io backend where images are store. Defaults to None.

transform (*results: dict*) → dict

Functions to load paired images.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**MATLABLikeResize**(*keys*, *scale=None*, *output_shape=None*, *kernel='bicubic'*, *kernel_width=4.0*)

Resize the input image using MATLAB-like downsampling.

Currently support bicubic interpolation only. Note that the output of this function is slightly different from the official MATLAB function.

Required keys are the keys in attribute "keys". Added or modified keys are "scale" and "output_shape", and the keys in attribute "keys".

参数

- **keys** (*list[str]*) –A list of keys whose values are modified.
- **scale** (*float | None, optional*) –The scale factor of the resize operation. If None, it will be determined by output_shape. Default: None.
- **output_shape** (*tuple(int) | None, optional*) –The size of the output image. If None, it will be determined by scale. Note that if scale is provided, output_shape will not be used. Default: None.
- **kernel** (*str, optional*) –The kernel for the resize operation. Currently support 'bicubic' only. Default: 'bicubic'.
- **kernel_width** (*float*) –The kernel width. Currently support 4.0 only. Default: 4.0.

transform(*results*)

transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**MergeFgAndBg**

Composite foreground image and background image with alpha.

Required keys are "alpha", "fg" and "bg", added key is "merged".

transform(*results: dict*) → dict

Transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**MirrorSequence**(*keys*)

Extend short sequences (e.g. Vimeo-90K) by mirroring the sequences.

Given a sequence with N frames (x1, ⋯, xN), extend the sequence to (x1, ⋯, xN, xN, ⋯, x1).

Required Keys:

- [KEYS]

Modified Keys:

- [KEYS]

参数 keys (*list[str]*) –The frame lists to be extended.

transform(*results*)

transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**ModCrop**(*key='gt'*)

Mod crop images, used during testing.

Required keys are "scale" and "KEY", added or modified keys are "KEY".

参数 key (*str*) –The key of image. Default: 'gt'

transform(*results*)

Transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**Normalize**(*keys*, *mean*, *std*, *to_rgb=False*, *save_original=False*)

Normalize images with the given mean and std value.

Required keys are the keys in attribute "keys", added or modified keys are the keys in attribute "keys" and these keys with postfix '_norm_cfg'. It also supports normalizing a list of images.

参数

- **keys** (*Sequence[str]*) –The images to be normalized.
- **mean** (*np.ndarray*) –Mean values of different channels.
- **std** (*np.ndarray*) –Std values of different channels.
- **to_rgb** (*bool*) –Whether to convert channels from BGR to RGB. Default: False.
- **save_original** (*bool*) –Whether to save original images. Default: False.

transform(*results*)

transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**NumpyPad**(*keys*, *padding*, ***kwargs*)

Numpy Padding.

In this augmentation, numpy padding is adopted to customize padding augmentation. Please carefully read the numpy manual in: https://numpy.org/doc/stable/reference/generated/numpy.pad.html

If you just hope a single dimension to be padded, you must set `padding` like this:

```
padding = ((2, 2), (0, 0), (0, 0))
```

In this case, if you adopt an input with three dimension, only the first dimension will be padded.

参数

- **keys** (*Union[str, List[str]]*) –The images to be padded.
- **padding** (*int | tuple(int)*) –Please refer to the args `pad_width` in `numpy.pad`.

transform(*results*)

Call function.

参数 **results**(*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**PackEditInputs**(*keys: Optional[Tuple[List[str], str, None]] = None*, *pack_all: bool = False*)

Pack the inputs data for SR, VFI, matting and inpainting.

Keys for images include img, gt, ref, mask, gt_heatmap, trimap, gt_alpha, gt_fg, gt_bg. All of them will be packed into data field of EditDataSample.

pack_all (bool): Whether pack all variables in *results* to *inputs* dict. This is useful when keys of the input dict is not fixed. Please be careful when using this function, because we do not Defaults to False.

Others will be packed into metainfo field of EditDataSample.

transform(*results: dict*) → dict

Method to pack the input data.

参数 **results**(*dict*) –Result dict from the data pipeline.

返回

- 'inputs' (obj:*torch.Tensor*): The forward data of models.
- **'data_samples' (obj:*EditDataSample*): The annotation info of the** sample.

返回类型 dict

class mmedit.datasets.transforms.**PairedRandomCrop**(*gt_patch_size*, *lq_key='img'*, *gt_key='gt'*)

Paried random crop.

It crops a pair of img and gt images with corresponding locations. It also supports accepting img list and gt list. Required keys are "scale", "lq_key", and "gt_key", added or modified keys are "lq_key" and "gt_key".

参数

- **gt_patch_size**(*int*) –cropped gt patch size.
- **lq_key**(*str*) –Key of LQ img. Default: 'img'.
- **gt_key**(*str*) –Key of GT img. Default: 'gt'.

transform(*results*)

Transform function.

参数 **results**(*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**PerturbBg**(*gamma_ratio=0.6*)

Randomly add gaussian noise or gamma change to background image.

Required key is "bg", added key is "noisy_bg".

参数 **gamma_ratio** (*float, optional*) –The probability to use gamma correction instead of gaussian noise. Defaults to 0.6.

transform(*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomAffine**(*keys*, *degrees*, *translate=None*, *scale=None*, *shear=None*, *flip_ratio=None*)

Apply random affine to input images.

This class is adopted from https://github.com/pytorch/vision/blob/v0.5.0/torchvision/transforms/ transforms.py#L1015 It should be noted that in https://github.com/Yaoyi-Li/GCA-Matting/blob/master/dataloader/ data_generator.py#L70 random flip is added. See explanation of *flip_ratio* below. Required keys are the keys in attribute "keys", modified keys are keys in attribute "keys".

参数

- **keys** (*Sequence[str]*) –The images to be affined.
- **degrees** (*float | tuple[float]*) –Range of degrees to select from. If it is a float instead of a tuple like (min, max), the range of degrees will be (-degrees, +degrees). Set to 0 to deactivate rotations.
- **translate** (*tuple, optional*) –Tuple of maximum absolute fraction for horizontal and vertical translations. For example translate=(a, b), then horizontal shift is randomly sampled in the range -img_width * a < dx < img_width * a and vertical shift is randomly sampled in the range -img_height * b < dy < img_height * b. Default: None.
- **scale** (*tuple, optional*) –Scaling factor interval, e.g (a, b), then scale is randomly sampled from the range a <= scale <= b. Default: None.
- **shear** (*float | tuple[float], optional*) –Range of shear degrees to select from. If shear is a float, a shear parallel to the x axis and a shear parallel to the y axis in the range (-shear, +shear) will be applied. Else if shear is a tuple of 2 values, a x-axis shear and a y-axis shear in (shear[0], shear[1]) will be applied. Default: None.

- **flip_ratio** (*float, optional*) –Probability of the image being flipped. The flips in horizontal direction and vertical direction are independent. The image may be flipped in both directions. Default: None.

transform(*results*)

transform function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomBlur**(*params*, *keys*)

Apply random blur to the input.

Modified keys are the attributed specified in "keys".

参数

- **params** (*dict*) –A dictionary specifying the degradation settings.
- **keys** (*list[str]*) –A list specifying the keys whose values are modified.

get_kernel(*num_kernels: int*)

This is the function to create kernel.

参数 num_kernels (*int*) –the number of kernels

返回 _description_

返回类型 _type_

class mmedit.datasets.transforms.**RandomCropLongEdge**(*keys='img'*)

Random crop the given image by the long edge.

参数 keys (*list[str]*) –The images to be cropped.

transform(*results*)

Call function.

参数 results (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomDownSampling**(*scale_min=1.0*, *scale_max=4.0*, *patch_size=None*, *interpolation='bicubic'*, *backend='pillow'*)

Generate LQ image from GT (and crop), which will randomly pick a scale.

参数

- **scale_min** (*float*) –The minimum of upsampling scale, inclusive. Default: 1.0.
- **scale_max** (*float*) –The maximum of upsampling scale, exclusive. Default: 4.0.
- **patch_size** (*int*) –The cropped lr patch size. Default: None, means no crop.
- **interpolation** (*str*) –Interpolation method, accepted values are "nearest", "bilinear", "bicubic", "area", "lanczos" for 'cv2' backend, "nearest", "bilinear", "bicubic", "box", "lanczos", "hamming" for 'pillow' backend. Default: "bicubic".
- **backend** (*str | None*) –The image resize backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global imread_backend specified by `mmcv.use_backend()` will be used. Default: "pillow".
- **[scale_min** (*Scale will be picked in the range of*) –
- **scale_max).** –

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation. 'gt' is required.

返回

A dict containing the processed data and information. modified 'gt', supplement 'lq' and 'scale' to keys.

返回类型 dict

class mmedit.datasets.transforms.**RandomJPEGCompression**(*params*, *keys*)

Apply random JPEG compression to the input.

Modified keys are the attributed specified in "keys".

参数

- **params** (*dict*) –A dictionary specifying the degradation settings.
- **keys** (*list[str]*) –A list specifying the keys whose values are modified.

class mmedit.datasets.transforms.**RandomJitter**(*hue_range=40*)

Randomly jitter the foreground in hsv space.

The jitter range of hue is adjustable while the jitter ranges of saturation and value are adaptive to the images. Side effect: the "fg" image will be converted to *np.float32*. Required keys are "fg" and "alpha", modified key is "fg".

参数 **hue_range** (*float | tuple[float]*) –Range of hue jittering. If it is a float instead of a tuple like (min, max), the range of hue jittering will be (-hue_range, +hue_range). Default: 40.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomLoadResizeBg**(*bg_dir*, *flag='color'*, *channel_order='bgr'*)

Randomly load a background image and resize it.

Required key is "fg", added key is "bg".

参数

- **bg_dir** (*str*) –Path of directory to load background images from.
- **flag** (*str*) –Loading flag for images. Default: 'color'.
- **channel_order** (*str*) –Order of channel, candidates are 'bgr' and 'rgb'. Default: 'bgr'.
- **kwargs** (*dict*) –Args for file client.

transform(*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomMaskDilation**(*keys*, *binary_thr=0.0*, *kernel_min=9*, *kernel_max=49*)

Randomly dilate binary masks.

参数

- **keys** (*Sequence[str]*) –The images to be resized.
- **binary_thr** (*float*) –Threshold for obtaining binary mask. Default: 0.
- **kernel_min** (*int*) –Min size of dilation kernel. Default: 9.
- **kernel_max** (*int*) –Max size of dilation kernel. Default: 49.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomNoise**(*params*, *keys*)

Apply random noise to the input.

Currently support Gaussian noise and Poisson noise.

Modified keys are the attributed specified in "keys".

参数

- **params** (*dict*) –A dictionary specifying the degradation settings.
- **keys** (*list[str]*) –A list specifying the keys whose values are modified.

class mmedit.datasets.transforms.**RandomResize**(*params*, *keys*)

Randomly resize the input.

Modified keys are the attributed specified in "keys".

参数

- **params** (*dict*) –A dictionary specifying the degradation settings.
- **keys** (*list[str]*) –A list specifying the keys whose values are modified.

class mmedit.datasets.transforms.**RandomResizedCrop**(*keys*, *crop_size*, *scale=(0.08, 1.0)*, *ratio=(0.75, 1.3333333333333333)*, *interpolation='bilinear'*)

Crop data to random size and aspect ratio.

A crop of a random proportion of the original image and a random aspect ratio of the original aspect ratio is made. The cropped image is finally resized to a given size specified by 'crop_size'. Modified keys are the attributes specified in "keys".

This code is partially adopted from torchvision.transforms.RandomResizedCrop: [https://pytorch.org/vision/stable/_modules/torchvision/transforms/ transforms.html#RandomResizedCrop].

参数

- **keys** (*list[str]*) –The images to be resized and random-cropped.
- **crop_size** (*int | tuple[int]*) –Target spatial size (h, w).
- **scale** (*tuple[float], optional*) –Range of the proportion of the original image to be cropped. Default: (0.08, 1.0).
- **ratio** (*tuple[float], optional*) –Range of aspect ratio of the crop. Default: (3. / 4., 4. / 3.).
- **interpolation** (*str, optional*) –Algorithm used for interpolation. It can be only either one of the following: "nearest" | "bilinear" | "bicubic" | "area" | "lanczos". Default: "bilinear".

get_params(*data*)

Get parameters for a random sized crop.

参数 **data** (*np.ndarray*) –Image of type numpy array to be cropped.

返回 A tuple containing the coordinates of the top left corner and the chosen crop size.

transform(*results*)

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomRotation**(*keys*, *degrees*)

Rotate the image by a randomly-chosen angle, measured in degree.

参数

- **keys** (*list[str]*) –The images to be rotated.
- **degrees** (*tuple[float] | tuple[int] | float | int*) –If it is a tuple, it represents a range (min, max). If it is a float or int, the range is constructed as (-degrees, degrees).

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomTransposeHW**(*keys*, *transpose_ratio=0.5*)

Randomly transpose images in H and W dimensions with a probability.

(TransposeHW = horizontal flip + anti-clockwise rotation by 90 degrees) When used with horizontal/vertical flips, it serves as a way of rotation augmentation. It also supports randomly transposing a list of images.

Required keys are the keys in attributes "keys", added or modified keys are "transpose" and the keys in attributes "keys".

参数

- **keys** (*list[str]*) –The images to be transposed.
- **transpose_ratio** (*float*) –The probability to transpose the images. Default: 0.5.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**RandomVideoCompression**(*params*, *keys*)

Apply random video compression to the input.

Modified keys are the attributed specified in "keys".

参数

- **params** (*dict*) –A dictionary specifying the degradation settings.
- **keys** (*list[str]*) –A list specifying the keys whose values are modified.

class mmedit.datasets.transforms.**RescaleToZeroOne**(*keys*)

Transform the images into a range between 0 and 1.

Required keys are the keys in attribute "keys", added or modified keys are the keys in attribute "keys". It also supports rescaling a list of images.

参数 **keys** (*Sequence[str]*) –The images to be transformed.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**Resize**(*keys: Union[str, List[str]] = 'img'*, *scale=None*, *keep_ratio=False*, *size_factor=None*, *max_size=None*, *interpolation='bilinear'*, *backend=None*, *output_keys=None*)

Resize data to a specific size for training or resize the images to fit the network input regulation for testing.

When used for resizing images to fit network input regulation, the case is that a network may have several downsample and then upsample operation, then the input height and width should be divisible by the downsample factor of the network. For example, the network would downsample the input for 5 times with stride 2, then the downsample factor is 2^5 = 32 and the height and width should be divisible by 32.

Required keys are the keys in attribute "keys", added or modified keys are "keep_ratio", "scale_factor", "interpolation" and the keys in attribute "keys".

Required Keys:

- Required keys are the keys in attribute "keys"

Modified Keys:

- Modified the keys in attribute "keys" or save as new key ([OUT_KEY])

Added Keys:

- [OUT_KEY]_shape
- keep_ratio
- scale_factor
- interpolation

All keys in "keys" should have the same shape. "test_trans" is used to record the test transformation to align the input's shape.

参数

- **keys** (*str | list[str]*) –The image(s) to be resized.
- **scale** (*float | tuple[int]*) –If scale is tuple[int], target spatial size (h, w). Otherwise, target spatial size is scaled by input size. Note that when it is used, *size_factor* and *max_size* are useless. Default: None
- **keep_ratio** (*bool*) –If set to True, images will be resized without changing the aspect ratio. Otherwise, it will resize images to a given size. Default: False. Note that it is used together with *scale*.
- **size_factor** (*int*) –Let the output shape be a multiple of size_factor. Default:None. Note that when it is used, *scale* should be set to None and *keep_ratio* should be set to False.
- **max_size** (*int*) –The maximum size of the longest side of the output. Default:None. Note that it is used together with *size_factor*.
- **interpolation** (*str*) –Algorithm used for interpolation: "nearest"| "bilinear"| "bicubic" | "area" | "lanczos". Default: "bilinear".
- **backend** (*str | None*) –The image resize backend type. Options are *cv2*, *pillow*, *None*. If backend is None, the global imread_backend specified by `mmcv.use_backend()` will be used. Default: None.
- **output_keys** (*list[str] | None*) –The resized images. Default: None Note that if it is not *None*, its length should be equal to keys.

transform (*results: Dict*) → Dict

Transform function to resize images.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**SetValues**(*dictionary*)

Set value to destination keys.

It does the following: results[key] = value

Added keys are the keys in the dictionary.

Required Keys:

- None

Added or Modifyed Keys:

- keys in the dictionary

参数 **dictionary** (*dict*) –The dictionary to update.

transform(*results: Dict*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict with a key added/modified.

返回类型 dict

class mmedit.datasets.transforms.**TemporalReverse**(*keys*, *reverse_ratio=0.5*)

Reverse frame lists for temporal augmentation.

Required keys are the keys in attributes "lq" and "gt", added or modified keys are "lq", "gt" and "reverse".

参数

- **keys** (*list[str]*) –The frame lists to be reversed.
- **reverse_ratio** (*float*) –The probability to reverse the frame lists. Default: 0.5.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**ToTensor**(*keys*, *to_float32=True*)

Convert some values in results dict to *torch.Tensor* type in data loader pipeline.

参数

- **keys** (*Sequence[str]*) –Required keys to be converted.
- **to_float32** (*bool*) –Whether convert tensors of images to float32. Default: True.

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**TransformTrimap**

Transform trimap into two-channel and six-channel.

This class will generate a two-channel trimap composed of definite foreground and background masks and encode it into a six-channel trimap using Gaussian blurs of the generated two-channel trimap at three different scales. The transformed trimap has 6 channels.

Required key is "trimap", added key is "transformed_trimap" and "two_channel_trimap".

Adopted from the following repository: https://github.com/MarcoForte/FBA_Matting/blob/master/networks/transforms.py.

transform(*results: dict*) → dict

Transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

class mmedit.datasets.transforms.**UnsharpMasking**(*kernel_size*, *sigma*, *weight*, *threshold*, *keys*)

Apply unsharp masking to an image or a sequence of images.

参数

- **kernel_size** (*int*) –The kernel_size of the Gaussian kernel.
- **sigma** (*float*) –The standard deviation of the Gaussian.
- **weight** (*float*) –The weight of the "details" in the final output.
- **threshold** (*float*) –Pixel differences larger than this value are regarded as "details".
- **keys** (*list[str]*) –The keys whose values are processed.

Added keys are "xxx_unsharp", where "xxx" are the attributes specified in "keys".

transform(*results*)

transform function.

参数 **results** (*dict*) –A dict containing the necessary information and data for augmentation.

返回 A dict containing the processed data and information.

返回类型 dict

CHAPTER 16

mmedit.engine

16.1 hooks

class mmedit.engine.hooks.**BasicVisualizationHook**(*interval: dict = {}*, *on_train=False*, *on_val=True*, *on_test=True*)

Basic hook that invoke visualizers during validation and test.

参数

- **interval** (*int | dict*) –Visualization interval. Default: {}.
- **on_train** (*bool*) –Whether to call hook during train. Default to False.
- **on_val** (*bool*) –Whether to call hook during validation. Default to True.
- **on_test** (*bool*) –Whether to call hook during test. Default to True.

class mmedit.engine.hooks.**ExponentialMovingAverageHook**(*module_keys*, *interp_mode='lerp'*, *interp_cfg=None*, *interval=-1*, *start_iter=0*)

Exponential Moving Average Hook.

Exponential moving average is a trick that widely used in current GAN literature, e.g., PGGAN, StyleGAN, and BigGAN. This general idea of it is maintaining a model with the same architecture, but its parameters are updated as a moving average of the trained weights in the original model. In general, the model with moving averaged weights achieves better performance.

参数

- **module_keys** (*str | tuple[str]*) –The name of the ema model. Note that we require these keys are followed by '_ema' so that we can easily find the original model by discarding the last four characters.
- **interp_mode** (*str, optional*) –Mode of the interpolation method. Defaults to 'lerp'.
- **interp_cfg** (*dict | None, optional*) –Set arguments of the interpolation function. Defaults to None.
- **interval** (*int, optional*) –Evaluation interval (by iterations). Default: -1.
- **start_iter** (*int, optional*) –Start iteration for ema. If the start iteration is not reached, the weights of ema model will maintain the same as the original one. Otherwise, its parameters are updated as a moving average of the trained weights in the original model. Default: 0.

after_train_iter(*runner: mmengine.runner.runner.Runner*, *batch_idx: int*, *data_batch: Optional[Sequence[dict]] = None*, *outputs: Optional[dict] = None*) → None

This is the function to perform after each training iteration.

参数

- **runner** (*Runner*) –runner to drive the pipeline
- **batch_idx** (*int*) –the id of batch
- **data_batch** (*DATA_BATCH, optional*) –data batch. Defaults to None.
- **outputs** (*Optional[dict], optional*) –output. Defaults to None.

before_run(*runner: mmengine.runner.runner.Runner*)

This is the function perform before each run.

参数 **runner** (*Runner*) –runner used to drive the whole pipeline

引发 **RuntimeError** –error message

every_n_iters(*runner: mmengine.runner.runner.Runner*, *n: int*)

This is the function to perform every n iterations.

参数

- **runner** (*Runner*) –runner used to drive the whole pipeline
- **n** (*int*) –the number of iterations

返回 the latest iterations

返回类型 int

static lerp(*a*, *b*, *momentum=0.999*, *momentum_nontrainable=0.0*, *trainable=True*)

This is the function to perform linear interpolation between a and b.

参数

- **a** (*float*) –number a
- **b** (*float*) –bumber b
- **momentum** (*float, optional*) –momentum. Defaults to 0.999.
- **momentum_nontrainable** (*float, optional*) –Defaults to 0.
- **trainable** (*bool, optional*) –trainable flag. Defaults to True.

返回 _description_

返回类型 _type_

class mmedit.engine.hooks.**GenIterTimerHook**

GenIterTimerHooks inherits from *:class:~mmengine.hooks.IterTimerHook* and overwrites *:meth:self._after_iter*.

This hooks should be used along with *:class:mmedit.engine.runners.loops.GenValLoop* and *:class:mmedit.engine.runners.loops.GenTestLoop*.

class mmedit.engine.hooks.**GenVisualizationHook**(*interval: int = 1000*, *vis_kwargs_list: Optional[Tuple[List[dict], dict]] = None*, *fixed_input: bool = True*, *n_samples: Optional[int] = 64*, *n_row: Optional[int] = 8*, *message_hub_vis_kwargs: Optional[Tuple[str, dict, List[str], List[Dict]]] = None*, *save_at_test: bool = True*, *max_save_at_test: int = 100*, *test_vis_keys: Optional[Union[str, List[str]]] = None*, *show: bool = False*, *wait_time: float = 0*)

Generation Visualization Hook. Used to visual output samples in training, validation and testing. In this hook, we use a list called *sample_kwargs_list* to control how to generate samples and how to visualize them. Each element in *sample_kwargs_list*, called *sample_kwargs*, may contains the following keywords:

- **Required key words:**
  - **'type': Value must be string. Denotes what kind of sampler is used to** generate image. Refers to *:meth:~mmgen.core.sampler.get_sampler*.
- **Optional key words (If not passed, will use the default value):**
  - 'n_rows': Value must be int. The number of images in one row.
  - 'num_samples': Value must be int. The number of samples to visualize.
  - **'vis_mode': Value must be string. How to visualize the generated** samples (e.g. image, gif).
  - **'fixed_input': Value must be bool. Whether use the fixed input** during the loop.
  - 'draw_gt': Value must be bool. Whether save the real images.

- **'target_keys' : Value must be string or list of string. The keys of** the target image to visualize.
- **'name' : Value must be string. If not passed, will use** *sample_kwargs['type']* as default.

For convenience, we also define a group of alias of samplers' type for models supported in MMEditing. Refers to *:attr:self.SAMPLER_TYPE_MAPPING*.

示例

```
>>> # for GAN models
>>> custom_hooks = [
>>>     dict(
>>>         type='GenVisualizationHook',
>>>         interval=1000,
>>>         fixed_input=True,
>>>         vis_kwargs_list=dict(type='GAN', name='fake_img'))]
>>> # for Translation models
>>> custom_hooks = [
>>>     dict(
>>>         type='GenVisualizationHook',
>>>         interval=10,
>>>         fixed_input=False,
>>>         vis_kwargs_list=[dict(type='Translation',
>>>                               name='translation_train',
>>>                               n_samples=6, draw_gt=True,
>>>                               n_rows=3),
>>>                          dict(type='TranslationVal',
>>>                               name='translation_val',
>>>                               n_samples=16, draw_gt=True,
>>>                               n_rows=4)])]
```

NOTE: user-defined vis_kwargs > vis_kwargs_mapping > hook init args

参数

- **interval** (*int*) –Visualization interval. Default: 1000.
- **sampler_kwargs_list** (*Tuple[List[dict], dict]*) –The list of sampling behavior to generate images.
- **fixed_input** (*bool*) –The default action of whether use fixed input to generate samples during the loop. Defaults to True.
- **n_samples** (*Optional[int]*) –The default value of number of samples to visualize. Defaults to 64.

- **n_row** (*Optional[int]*) –The default value of number of images in each row in the visualization results. Defaults to 8.
- **(Optional[Tuple[str** (*message_hub_vis_kwargs*) –List[Dict]]]): Key arguments visualize images in message hub. Defaults to None.
- **dict** –List[Dict]]]): Key arguments visualize images in message hub. Defaults to None.
- **List[str]** –List[Dict]]]): Key arguments visualize images in message hub. Defaults to None.

:param [List[Dict]]]): Key arguments visualize images in message hub.] Defaults to None.

参数

- **save_at_test** (*bool*) –Whether save images during test. Defaults to True.
- **max_save_at_test** (*int*) –Maximum number of samples saved at test time. If None is passed, all samples will be saved. Defaults to 100.
- **show** (*bool*) –Whether to display the drawn image. Default to False.
- **wait_time** (*float*) –The interval of show (s). Defaults to 0.

after_test_iter(*runner: mmengine.runner.runner.Runner*, *batch_idx: int*, *data_batch: dict*, *outputs*)

Visualize samples after test iteraiton.

参数

- **runner** (*Runner*) –The runner of the training process.
- **batch_idx** (*int*) –The index of the current batch in the test loop.
- **data_batch** (*dict, optional*) –Data from dataloader. Defaults to None.
- **outputs** –outputs of the generation model Defaults to None.

after_train_iter(*runner: mmengine.runner.runner.Runner*, *batch_idx: int*, *data_batch: dict = None*, *outputs: Optional[dict] = None*) → None

Visualize samples after train iteration.

参数

- **runner** (*Runner*) –The runner of the training process.
- **batch_idx** (*int*) –The index of the current batch in the train loop.
- **data_batch** (*dict*) –Data from dataloader. Defaults to None.
- **outputs** (*dict, optional*) –Outputs from model. Defaults to None.

after_val_iter(*runner: mmengine.runner.runner.Runner*, *batch_idx: int*, *data_batch: dict*, *outputs*) → None

GenVisualizationHook do not support visualize during validation.

参数

- **runner** (*Runner*) –The runner of the training process.
- **batch_idx** (*int*) –The index of the current batch in the test loop.
- **data_batch** (*Sequence[dict], optional*) –Data from dataloader. Defaults to None.
- **outputs** –outputs of the generation model

vis_from_message_hub(*batch_idx: int*, *color_order: str*, *target_mean: Sequence[Union[float, int]]*, *target_std: Sequence[Union[float, int]]*)

Visualize samples from message hub.

参数

- **batch_idx** (*int*) –The index of the current batch in the test loop.
- **color_order** (*str*) –The color order of generated images.
- **target_mean** (*Sequence[Union[float, int]]*) –The original mean of the image tensor before preprocessing. Image will be re-shifted to `target_mean` before visualizing.
- **target_std** (*Sequence[Union[float, int]]*) –The original std of the image tensor before preprocessing. Image will be re-scaled to `target_std` before visualizing.

vis_sample(*runner: mmengine.runner.runner.Runner*, *batch_idx: int*, *data_batch: dict*, *outputs: Optional[dict] = None*) → None

Visualize samples.

参数

- **runner** (*Runner*) –The runner conatians model to visualize.
- **batch_idx** (*int*) –The index of the current batch in loop.
- **data_batch** (*dict*) –Data from dataloader. Defaults to None.
- **outputs** (*dict, optional*) –Outputs from model. Defaults to None.

class mmedit.engine.hooks.**PGGANFetchDataHook**

PGGAN Fetch Data Hook.

参数 **interval** (*int, optional*) –The interval of calling this hook. If set to -1, the visualization hook will not be called. Defaults to 1.

before_train_iter(*runner*, *batch_idx: int*, *data_batch: Optional[Sequence[dict]] = None*) → None

All subclasses should override this method, if they need any operations before each training iteration.

参数

- **runner** (*Runner*) –The runner of the training process.
- **batch_idx** (*int*) –The index of the current batch in the train loop.
- **data_batch** (*dict or tuple or list, optional*) –Data from dataloader.

update_dataloader(*dataloader: torch.utils.data.dataloader.DataLoader*, *curr_scale: int*) → Optional[torch.utils.data.dataloader.DataLoader]

Updata the data loader.

参数

- **dataloader** (*DataLoader*) –The dataloader to be updated.
- **curr_scale** (*int*) –The current scale of the generated image.

返回

The updated dataloader. If the dataloader do not need to update, return None.

返回类型 Optional[DataLoader]

class mmedit.engine.hooks.**PickleDataHook**(*output_dir*, *data_name_list*, *interval=- 1*, *before_run=False*, *after_run=False*, *filename_tmpl='iter_{}.pkl'*)

Pickle Useful Data Hook.

This hook will be used in SinGAN training for saving some important data that will be used in testing or inference.

参数

- **output_dir** (*str*) –The output path for saving pickled data.
- **data_name_list** (*list[str]*) –The list contains the name of results in outputs dict.
- **interval** (*int*) –The interval of calling this hook. If set to -1, the PickleDataHook will not be called during training. Default: -1.
- **before_run** (*bool, optional*) –Whether to save before running. Defaults to False.
- **after_run** (*bool, optional*) –Whether to save after running. Defaults to False.
- **filename_tmpl** (*str, optional*) –Format string used to save images. The output file name will be formatted as this args. Defaults to ‘iter_{}.pkl’.

after_run(*runner*)

The behavior after each train iteration.

参数 **runner** (*object*) –The runner.

after_train_iter(*runner*, *batch_idx: int*, *data_batch: Optional[Sequence[dict]] = None*, *outputs: Optional[dict] = None*)

The behavior after each train iteration.

参数

- **runner** (*Runner*) –The runner of the training process.
- **batch_idx** (*int*) –The index of the current batch in the train loop.
- **data_batch** (*Sequence[dict], optional*) –Data from dataloader. Defaults to None.
- **outputs** (*dict, optional*) –Outputs from model. Defaults to None.

before_run(*runner*)

The behavior after each train iteration.

参数 **runner** (*object*) –The runner.

class mmedit.engine.hooks.**ReduceLRSchedulerHook**(*val_metric: Optional[str] = None*, *by_epoch=True*, *interval=1*)

A hook to update learning rate.

参数

- **val_metric** (*str*) –The metric of validation. If val_metric is not None, we check val_metric to reduce learning. Default: None.
- **by_epoch** (*bool*) –Whether to update by epoch. Default: True.
- **interval** (*int*) –The interval of iterations to update. Default: 1.

after_train_epoch(*runner: mmengine.runner.runner.Runner*)

Call step function for each scheduler after each train epoch.

参数 **runner** (*Runner*) –The runner of the training process.

after_train_iter(*runner: mmengine.runner.runner.Runner*, *batch_idx: int*, *data_batch: Optional[Sequence[dict]] = None*, *outputs: Optional[dict] = None*) → None

Call step function for each scheduler after each iteration.

参数

- **runner** (*Runner*) –The runner of the training process.
- **batch_idx** (*int*) –The index of the current batch in the train loop.
- **data_batch** (*Sequence[dict], optional*) –Data from dataloader. In order to keep this interface consistent with other hooks, we keep data_batch here. Defaults to None.
- **outputs** (*dict, optional*) –Outputs from model. In order to keep this interface consistent with other hooks, we keep data_batch here. Defaults to None.

after_val_epoch(*runner*, *metrics: Optional[Dict[str, float]] = None*)

Call step function for each scheduler after each validation epoch.

参数

- **runner** (*Runner*) –The runner of the training process.
- **metrics** (*dict, optional*) –The metrics of validation. Default: None.

16.2 optimizers

class mmedit.engine.optimizers.**MultiOptimWrapperConstructor**(*optim_wrapper_cfg: dict*, *paramwise_cfg=None*)

OptimizerConstructor for GAN models. This class construct optimizer for the submodules of the model separately, and return a :class:~ *mmengine.optim.OptimWrapperDict*.

示例

```
>>> # build GAN model
>>> model = dict(
>>>     type='GANModel',
>>>     num_classes=10,
>>>     generator=dict(type='Generator'),
>>>     discriminator=dict(type='Discriminator'))
>>> gan_model = MODELS.build(model)
>>> # build constructor
>>> optim_wrapper = dict(
>>>     constructor='MultiOptimWrapperConstructor',
>>>     generator=dict(
>>>         type='OptimWrapper',
>>>         accumulative_counts=1,
>>>         optimizer=dict(type='Adam', lr=0.0002,
>>>                        betas=(0.5, 0.999))),
>>>     discriminator=dict(
>>>         type='OptimWrapper',
>>>         accumulative_counts=1,
>>>         optimizer=dict(type='Adam', lr=0.0002,
>>>                            betas=(0.5, 0.999))))
>>> optim_dict_builder = MultiOptimWrapperConstructor(optim_wrapper)
>>> # build optim wrapper dict
>>> optim_wrapper_dict = optim_dict_builder(gan_model)
```

参数

- **optim_wrapper_cfg_dict** (*dict*) –Config of the optimizer wrapper.
- **paramwise_cfg** (*dict*) –Config of parameter-wise settings. Default: None.

class mmedit.engine.optimizers.**PGGANOptimWrapperConstructor**(*optim_wrapper_cfg: dict*, *paramwise_cfg: Optional[dict] = None*)

OptimizerConstructor for PGGAN models. Set optimizers for each stage of PGGAN. All submodule must be contained in a :class:~'torch.nn.ModuleList' named 'blocks'. And we access each submodule by *MODEL.blocks[SCALE]*, where *MODLE* is generator or discriminator, and the scale is the index of the resolution scale.

More detail about the resolution scale and naming rule please refers to :class:~'mmgen.models.PGGANGenerator' and :class:~'mmgen.models.PGGANDiscriminator'.

示例

```
>>> # build PGGAN model
>>> model = dict(
>>>     type='ProgressiveGrowingGAN',
>>>     data_preprocessor=dict(type='GANDataPreprocessor'),
>>>     noise_size=512,
>>>     generator=dict(type='PGGANGenerator', out_scale=1024,
>>>                    noise_size=512),
>>>     discriminator=dict(type='PGGANDiscriminator', in_scale=1024),
>>>     nkimgs_per_scale={
>>>         '4': 600,
>>>         '8': 1200,
>>>         '16': 1200,
>>>         '32': 1200,
>>>         '64': 1200,
>>>         '128': 1200,
>>>         '256': 1200,
>>>         '512': 1200,
>>>         '1024': 12000,
>>>     },
>>>     transition_kimgs=600,
>>>     ema_config=dict(interval=1))
>>> pggan = MODELS.build(model)
>>> # build constructor
>>> optim_wrapper = dict(
>>>     generator=dict(optimizer=dict(type='Adam', lr=0.001,
>>>                                   betas=(0., 0.99))),
>>>     discriminator=dict(
```

(下页继续)

(续上页)

```
>>>         optimizer=dict(type='Adam', lr=0.001, betas=(0., 0.99))),
>>>     lr_schedule=dict(
>>>         generator={
>>>             '128': 0.0015,
>>>             '256': 0.002,
>>>             '512': 0.003,
>>>             '1024': 0.003
>>>         },
>>>         discriminator={
>>>             '128': 0.0015,
>>>             '256': 0.002,
>>>             '512': 0.003,
>>>             '1024': 0.003
>>>         }))
>>> optim_wrapper_dict_builder = PGGANOptimWrapperConstructor(
>>>     optim_wrapper)
>>> # build optim wrapper dict
>>> optim_wrapper_dict = optim_wrapper_dict_builder(pggan)
```

参数

- **optim_wrapper_cfg** (*dict*) –Config of the optimizer wrapper.
- **paramwise_cfg** (*Optional[dict]*) –Parameter-wise options.

class mmedit.engine.optimizers.**SinGANOptimWrapperConstructor**(*optim_wrapper_cfg: dict*, *paramwise_cfg: Optional[dict] = None*)

OptimizerConstructor for SinGAN models. Set optimizers for each submodule of SinGAN. All submodule must be contained in a :class:~‘torch.nn.ModuleList‘ named ‘blocks’. And we access each submodule by *MODEL.blocks[SCALE]*, where *MODLE* is generator or discriminator, and the scale is the index of the resolution scale.

More detail about the resolution scale and naming rule please refers to :class:~‘mmgen.models.SinGANMultiScaleGenerator‘ and :class:~‘mmgen.models.SinGANMultiScaleDiscriminator‘.

示例

```
>>> # build SinGAN model
>>> model = dict(
>>>     type='SinGAN',
>>>     data_preprocessor=dict(
>>>         type='GANDataPreprocessor',
>>>         non_image_keys=['input_sample']),
>>>     generator=dict(
>>>         type='SinGANMultiScaleGenerator',
>>>         in_channels=3,
>>>         out_channels=3,
>>>         num_scales=2),
>>>     discriminator=dict(
>>>         type='SinGANMultiScaleDiscriminator',
>>>         in_channels=3,
>>>         num_scales=3))
>>> singan = MODELS.build(model)
>>> # build constructor
>>> optim_wrapper = dict(
>>>     generator=dict(optimizer=dict(type='Adam', lr=0.0005,
>>>                                   betas=(0.5, 0.999))),
>>>     discriminator=dict(
>>>         optimizer=dict(type='Adam', lr=0.0005,
>>>                        betas=(0.5, 0.999))))
>>> optim_wrapper_dict_builder = SinGANOptimWrapperConstructor(
>>>     optim_wrapper)
>>> # build optim wrapper dict
>>> optim_wrapper_dict = optim_wrapper_dict_builder(singan)
```

参数

- **optim_wrapper_cfg** (*dict*) –Config of the optimizer wrapper.
- **paramwise_cfg** (*Optional[dict]*) –Parameter-wise options.

16.3 schedulers

class mmedit.engine.schedulers.**LinearLrInterval**(**args*, *interval=1*, ***kwargs*)

Linear learning rate scheduler for image generation.

In the beginning, the learning rate is ‘start_factor’ defined in mmengine. We give a target learning rate ‘end_factor’ and a start point ‘begin’. If :attr:self.by_epoch is True, ‘begin’ is calculated by epoch, otherwise, calculated by

iteration." Before ‘begin’, we fix learning rate as ‘start_factor’; After ‘begin’, we linearly update learning rate to ‘end_factor’.

参数 **interval** (*int*) –The interval to update the learning rate. Default: 1.

class mmedit.engine.schedulers.**ReduceLR**(*optimizer*, *mode: str = 'min'*, *factor: float = 0.1*, *patience: int = 10*, *threshold: float = 0.0001*, *threshold_mode: str = 'rel'*, *cooldown: int = 0*, *min_lr: float = 0.0*, *eps: float = 1e-08*, ***kwargs*)

Decays the learning rate of each parameter group by linearly changing small multiplicative factor until the number of epoch reaches a pre-defined milestone: end.

Notice that such decay can happen simultaneously with other changes to the learning rate from outside this scheduler.

备注:

The learning rate of each parameter group will be update at regular intervals.

参数

- **optimizer** (*Optimizer or OptimWrapper*) –Wrapped optimizer.
- **mode** (*str, optional*) –One of *min*, *max*. In *min* mode, lr will be reduced when the quantity monitored has stopped decreasing; in *max* mode it will be reduced when the quantity monitored has stopped increasing. Default: ‘min’.
- **factor** (*float, optional*) –Factor by which the learning rate will be reduced. new_lr = lr * factor. Default: 0.1.
- **patience** (*int, optional*) –Number of epochs with no improvement after which learning rate will be reduced. For example, if *patience = 2*, then we will ignore the first 2 epochs with no improvement, and will only decrease the LR after the 3rd epoch if the loss still hasn’t improved then. Default: 10.
- **threshold** (*float, optional*) –Threshold for measuring the new optimum, to only focus on significant changes. Default: 1e-4.
- **threshold_mode** (*str, optional*) –One of *rel*, *abs*. In *rel* mode, dynamic_threshold = best * (1 + threshold) in ‘max’ mode or best * (1 - threshold) in *min* mode. In *abs* mode, dynamic_threshold = best + threshold in *max* mode or best - threshold in *min* mode. Default: ‘rel’.
- **cooldown** (*int, optional*) –Number of epochs to wait before resuming normal operation after lr has been reduced. Default: 0.
- **min_lr** (*float, optional*) –Minimum LR value to keep. If LR after decay is lower than *min_lr*, it will be clipped to this value. Default: 0.

- **eps** (*float, optional*) –Minimal decay applied to lr. If the difference between new and old lr is smaller than eps, the update is ignored. Default: 1e-8.
- **begin** (*int*) –Step at which to start updating the learning rate. Defaults to 0.
- **end** (*int*) –Step at which to stop updating the learning rate.
- **last_step** (*int*) –The index of last step. Used for resume without state dict. Defaults to -1.
- **by_epoch** (*bool*) –Whether the scheduled learning rate is updated by epochs. Defaults to True.

CHAPTER 17

mmedit.evaluation

17.1 metrics

17.2 functional

class mmedit.evaluation.functional.**InceptionV3**(*output_blocks=[3]*, *resize_input=True*, *normalize_input=True*, *requires_grad=False*, *use_fid_inception=True*, *load_fid_inception=True*)

Pretrained InceptionV3 network returning feature maps.

forward(*inp*)

Get Inception feature maps.

参数 **inp** (*torch.Tensor*) –Input tensor of shape Bx3xHxW. Values are expected to be in range (0, 1)

返回 Corresponding to the selected output block, sorted ascending by index.

返回类型 list(torch.Tensor)

mmedit.evaluation.functional.**disable_gpu_fuser_on_pt19**()

On PyTorch 1.9 a CUDA fuser bug prevents the Inception JIT model to run.

Refers to: https://github.com/GaParmar/clean-fid/blob/5e1e84cdea9654b9ac7189306dfa4057ea2213d8/cleanfid/inception_torchscript.py#L9 # noqa https://github.com/GaParmar/clean-fid/issues/5

https://github.com/pytorch/pytorch/issues/64062

mmedit.evaluation.functional.**gauss_gradient**(*img*, *sigma*)

Gaussian gradient.

From https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/ submissions/8060/versions/2/previews/gaussgradient/gaussgradient.m/ index.html

参数

- **img** (*np.ndarray*) –Input image.
- **sigma** (*float*) –Standard deviation of the gaussian kernel.

返回 Gaussian gradient of input *img*.

返回类型 np.ndarray

mmedit.evaluation.functional.**load_inception**(*inception_args*, *metric*)

Load Inception Model from given `inception_args` and `metric`.

This function would try to load Inception under the guidance of 'type' given in *inception_args*, if not given, we would try best to load Tero' s ones. In detailly, we would first try to load the model from disk with the given 'inception_path', and then try to download the checkpoint from 'inception_url'. If both method are failed, pytorch version of Inception would be loaded.

参数

- **inception_args** (*dict*) –Keyword args for inception net.
- **metric** (*string*) –Metric to use the Inception. This argument would influence the pytorch' s Inception loading.

返回 Loaded Inception model. style (string): The version of the loaded Inception.

返回类型 model (torch.nn.Module)

mmedit.evaluation.functional.**prepare_inception_feat**(*dataloader: torch.utils.data.dataloader.DataLoader*, *metric: mmengine.evaluator.metric.BaseMetric*, *data_preprocessor: Optional[torch.nn.modules.module.Module] = None*, *capture_mean_cov: bool = False*, *capture_all: bool = False*) → dict

Prepare inception feature for the input metric.

- If *metric.inception_pkl* is an online path, try to download and load it. If cannot download or load, corresponding error will be raised.
- If *metric.inception_pkl* is local path and file exists, try to load the file. If cannot load, corresponding error will be raised.

- If *metric.inception_pkl* is local path and file not exists, we will extract the inception feature manually and save to ‘inception_pkl’.
- If *metric.inception_pkl* is not defined, we will extrace the inception feature and save it to default cache dir with default name.

参数

- **datalaoder** (*Dataloader*) –The dataloader of real images.
- **metric** (*BaseMetric*) –The metric which needs inception features.
- **data_preprocessor** (*Optional[nn.Module]*) –Data preprocessor of the module. Used to preprocess the real images. If not passed, real images will automatically normalized to [-1, 1]. Defaults to None.
- **capture_mean_cov** (*bool*) –Whether save the mean and covariance of inception feature. Defaults to False.
- **capture_all** (*bool*) –Whether save the raw inception feature. If true, it will take a lot of time to save the inception feature. Defaults to False.

返回 Dict contains inception feature.

返回类型 dict

mmedit.evaluation.functional.**prepare_vgg_feat**(*dataloader: torch.utils.data.dataloader.DataLoader*, *metric: mmengine.evaluator.metric.BaseMetric*, *data_preprocessor: Optional[torch.nn.modules.module.Module] = None*, *auto_save=True*) → numpy.ndarray

Prepare vgg feature for the input metric.

- If *metric.vgg_pkl* is an online path, try to download and load it. If cannot download or load, corresponding error will be raised.
- If *metric.vgg_pkl* is local path and file exists, try to load the file. If cannot load, corresponding error will be raised.
- If *metric.vgg_pkl* is local path and file not exists, we will extract the vgg feature manually and save to ‘vgg_pkl’.
- If *metric.vgg_pkl* is not defined, we will extrace the vgg feature and save it to default cache dir with default name.

参数

- **datalaoder** (*Dataloader*) –The dataloader of real images.
- **metric** (*BaseMetric*) –The metric which needs vgg features.

- **data_preprocessor** (*Optional[nn.Module]*) –Data preprocessor of the module. Used to preprocess the real images. If not passed, real images will automatically normalized to [-1, 1]. Defaults to None.
- **Returns** –np.ndarray: Loaded vgg feature.

CHAPTER 18

mmedit.models

18.1 base_models

class mmedit.models.base_models.**BaseConditionalGAN**(*generator: Union[Dict, torch.nn.modules.module.Module]*, *discriminator: Optional[Union[Dict, torch.nn.modules.module.Module]] = None*, *data_preprocessor: Optional[Union[dict, mmengine.config.config.Config]] = None*, *generator_steps: int = 1*, *discriminator_steps: int = 1*, *noise_size: Optional[int] = None*, *num_classes: Optional[int] = None*, *ema_config: Optional[Dict] = None*, *loss_config: Optional[Dict] = None*)

Base class for Conditional GAM models.

参数

- **generator** (*ModelType*) –The config or model of the generator.
- **discriminator** (*Optional[ModelType]*) –The config or model of the discriminator. Defaults to None.

- **data_preprocessor** (*Optional[Union[dict, Config]]*) –The pre-process config or `GenDataPreprocessor`.
- **generator_steps** (*int*) –The number of times the generator is completely updated before the discriminator is updated. Defaults to 1.
- **discriminator_steps** (*int*) –The number of times the discriminator is completely updated before the generator is updated. Defaults to 1.
- **noise_size** (*Optional[int]*) –Size of the input noise vector. Default to None.
- **num_classes** (*Optional[int]*) –The number classes you would like to generate. Defaults to None.
- **ema_config** (*Optional[Dict]*) –The config for generator's exponential moving average setting. Defaults to None.

data_sample_to_label(*data_sample: List[mmedit.structures.edit_data_sample.EditDataSample]*) → Optional[torch.Tensor]

Get labels from input *data_sample* and pack to *torch.Tensor*. If no label is found in the passed *data_sample*, *None* would be returned.

参数 **data_sample** (*List[EditDataSample]*) –Input data samples.

返回 Packed label tensor.

返回类型 Optional[torch.Tensor]

forward(*inputs: Tuple[Dict[str, Union[torch.Tensor, str, int]], torch.Tensor]*, *data_samples: Optional[list] = None*, *mode: Optional[str] = None*) → List[mmedit.structures.edit_data_sample.EditDataSample]

Sample images with the given inputs. If forward mode is 'ema' or 'orig', the image generated by corresponding generator will be returned. If forward mode is 'ema/orig', images generated by original generator and EMA generator will both be returned in a dict.

参数

- **inputs** (*ForwardInputs*) –Dict containing the necessary information (e.g. noise, num_batches, mode) to generate image.
- **data_samples** (*Optional[list]*) –Data samples collated by `data_preprocessor`. Defaults to None.
- **mode** (*Optional[str]*) –*mode* is not used in *BaseConditionalGAN*. Defaults to None.

返回 Generated images or image dict.

返回类型 List[EditDataSample]

label_fn(*label: Optional[Union[torch.Tensor, Callable, List[int]]] = None*, *num_batches: int = 1*) → torch.Tensor

Sampling function for label. There are three scenarios in this function:

- If *label* is a callable function, sample *num_batches* of labels with passed *label*.
- If *label* is *None*, sample *num_batches* of labels in range of *[0, self.num_classes-1]* uniformly.
- If *label* is a *torch.Tensor*, check the range of the tensor is in *[0, self.num_classes-1]*. If all values are in valid range, directly return *label*.

参数

- **label** (*Union[Tensor, Callable, List[int], None]*) –You can directly give a batch of label through a `torch.Tensor` or offer a callable function to sample a batch of label data. Otherwise, the `None` indicates to use the default label sampler. Defaults to *None*.
- **num_batches** (*int, optional*) –The number of batches label want to sample. If *label* is a Tensor, this will be ignored. Defaults to 1.

返回 Sampled label tensor.

返回类型 Tensor

train_discriminator(*inputs: dict*, *data_samples: List[mmedit.structures.edit_data_sample.EditDataSample]*, *optimizer_wrapper: mmengine.optim.optimizer.optimizer_wrapper.OptimWrapper*) → Dict[str, torch.Tensor]

Training function for discriminator. All GANs should implement this function by themselves.

参数

- **inputs** (*dict*) –Inputs from dataloader.
- **data_samples** (*List[EditDataSample]*) –Data samples from dataloader.
- **optim_wrapper** (*OptimWrapper*) –OptimWrapper instance used to update model parameters.

返回 A `dict` of tensor for logging.

返回类型 Dict[str, Tensor]

train_generator(*inputs: dict*, *data_samples: List[mmedit.structures.edit_data_sample.EditDataSample]*, *optimizer_wrapper: mmengine.optim.optimizer.optimizer_wrapper.OptimWrapper*) → Dict[str, torch.Tensor]

Training function for discriminator. All GANs should implement this function by themselves.

参数

- **inputs** (*dict*) –Inputs from dataloader.

- **data_samples** (*List[EditDataSample]*) –Data samples from dataloader.
- **optim_wrapper** (*OptimWrapper*) –OptimWrapper instance used to update model parameters.

返回 A `dict` of tensor for logging.

返回类型 Dict[str, Tensor]

class mmedit.models.base_models.**BaseEditModel**(*generator*, *pixel_loss*, *train_cfg=None*, *test_cfg=None*, *init_cfg=None*, *data_preprocessor=None*)

Base model for image and video editing.

It must contain a generator that takes frames as inputs and outputs an interpolated frame. It also has a pixel-wise loss for training.

参数

- **generator** (*dict*) –Config for the generator structure.
- **pixel_loss** (*dict*) –Config for pixel-wise loss.
- **train_cfg** (*dict*) –Config for training. Default: None.
- **test_cfg** (*dict*) –Config for testing. Default: None.
- **init_cfg** (*dict, optional*) –The weight initialized config for `BaseModule`.
- **data_preprocessor** (*dict, optional*) –The pre-process config of `BaseDataPreprocessor`.

init_cfg

Initialization config dict.

Type dict, optional

data_preprocessor

Used for pre-processing data sampled by dataloader to the format accepted by *forward()*. Default: None.

Type `BaseDataPreprocessor`

forward(*inputs: torch.Tensor*, *data_samples: Optional[List[mmedit.structures.edit_data_sample.EditDataSample]] = None*, *mode: str = 'tensor'*, ***kwargs*)

Returns losses or predictions of training, validation, testing, and simple inference process.

`forward` method of BaseModel is an abstract method, its subclasses must implement this method.

Accepts `inputs` and `data_samples` processed by *data_preprocessor*, and returns results according to mode arguments.

During non-distributed training, validation, and testing process, `forward` will be called by `BaseModel.train_step`, `BaseModel.val_step` and `BaseModel.val_step` directly.

During distributed data parallel training process, `MMSeparateDistributedDataParallel.train_step` will first call `DistributedDataParallel.forward` to enable automatic gradient synchronization, and then call `forward` to get training loss.

参数

- **inputs** (*torch.Tensor*) –batch input tensor collated by *data_preprocessor*.
- **data_samples** (*List[BaseDataElement], optional*) –data samples collated by *data_preprocessor*.
- **mode** (*str*) –mode should be one of `loss`, `predict` and `tensor`. Default: 'tensor'.
  - `loss`: Called by `train_step` and return loss `dict` used for logging
  - `predict`: Called by `val_step` and `test_step` and return list of `BaseDataElement` results used for computing metric.
  - `tensor`: Called by custom use to get `Tensor` type results.

返回

- If `mode == loss`, return a `dict` of loss tensor used for backward and logging.
- If `mode == predict`, return a `list` of `BaseDataElement` for computing metric and getting inference result.
- If `mode == tensor`, return a tensor or `tuple` of tensor or `dict` or tensor for custom use.

返回类型 ForwardResults

forward_inference(*inputs*, *data_samples=None*, ***kwargs*)

Forward inference. Returns predictions of validation, testing, and simple inference.

参数

- **inputs** (*torch.Tensor*) –batch input tensor collated by *data_preprocessor*.
- **data_samples** (*List[BaseDataElement], optional*) –data samples collated by *data_preprocessor*.

返回 predictions.

返回类型 List[EditDataSample]

forward_tensor(*inputs*, *data_samples=None*, ***kwargs*)

Forward tensor. Returns result of simple forward.

参数

- **inputs** (*torch.Tensor*) –batch input tensor collated by *data_preprocessor*.

- **data_samples** (*List[BaseDataElement], optional*) –data samples collated by *data_preprocessor*.

返回 result of simple forward.

返回类型 Tensor

forward_train(*inputs*, *data_samples=None*, ***kwargs*)

Forward training. Returns dict of losses of training.

参数

- **inputs** (*torch.Tensor*) –batch input tensor collated by *data_preprocessor*.
- **data_samples** (*List[BaseDataElement], optional*) –data samples collated by *data_preprocessor*.

返回 Dict of losses.

返回类型 dict

class mmedit.models.base_models.**BaseGAN**(*generator: Union[Dict, torch.nn.modules.module.Module]*, *discriminator: Optional[Union[Dict, torch.nn.modules.module.Module]] = None*, *data_preprocessor: Optional[Union[dict, mmengine.config.config.Config]] = None*, *generator_steps: int = 1*, *discriminator_steps: int = 1*, *noise_size: Optional[int] = None*, *ema_config: Optional[Dict] = None*, *loss_config: Optional[Dict] = None*)

Base class for GAN models.

参数

- **generator** (*ModelType*) –The config or model of the generator.
- **discriminator** (*Optional[ModelType]*) –The config or model of the discriminator. Defaults to None.
- **data_preprocessor** (*Optional[Union[dict, Config]]*) –The pre-process config or GenDataPreprocessor.
- **generator_steps** (*int*) –The number of times the generator is completely updated before the discriminator is updated. Defaults to 1.
- **discriminator_steps** (*int*) –The number of times the discriminator is completely updated before the generator is updated. Defaults to 1.
- **ema_config** (*Optional[Dict]*) –The config for generator's exponential moving average setting. Defaults to None.

property device: torch.device

Get current device of the model.

返回 The current device of the model.

返回类型 torch.device

property discriminator_steps: int

The number of times the discriminator is completely updated before the generator is updated.

Type int

forward(*inputs: Tuple[Dict[str, Union[torch.Tensor, str, int]], torch.Tensor]*, *data_samples: Optional[list] = None*, *mode: Optional[str] = None*) → Sequence[mmengine.structures.base_data_element.BaseDataElement]

Sample images with the given inputs. If forward mode is 'ema' or 'orig', the image generated by corresponding generator will be returned. If forward mode is 'ema/orig', images generated by original generator and EMA generator will both be returned in a dict.

参数

- **batch_inputs** (*ForwardInputs*) –Dict containing the necessary information (e.g. noise, num_batches, mode) to generate image.
- **data_samples** (*Optional[list]*) –Data samples collated by `data_preprocessor`. Defaults to None.
- **mode** (*Optional[str]*) –*mode* is not used in *BaseGAN*. Defaults to None.

返回 A list of `EditDataSample` contain generated results.

返回类型 SampleList

static gather_log_vars(*log_vars_list: List[Dict[str, torch.Tensor]]*) → Dict[str, torch.Tensor]

Gather a list of log_vars. :param log_vars_list: List[Dict[str, Tensor]]

返回 Dict[str, Tensor]

property generator_steps: int

The number of times the generator is completely updated before the discriminator is updated.

Type int

noise_fn(*noise: Optional[Union[torch.Tensor, Callable]] = None*, *num_batches: int = 1*)

Sampling function for noise. There are three scenarios in this function:

- If *noise* is a callable function, sample *num_batches* of noise with passed *noise*.
- If *noise* is *None*, sample *num_batches* of noise from gaussian distribution.
- If *noise* is a *torch.Tensor*, directly return *noise*.

参数

- **noise** (*Union[Tensor, Callable, List[int], None]*) –You can directly give a batch of label through a `torch.Tensor` or offer a callable function to sample a batch of label data. Otherwise, the `None` indicates to use the default noise sampler. Defaults to *None*.
- **num_batches** (*int, optional*) –The number of batches label want to sample. If *label* is a Tensor, this will be ignored. Defaults to 1.

返回 Sampled noise tensor.

返回类型 Tensor

test_step(*data: dict*) → Sequence[mmengine.structures.base_data_element.BaseDataElement]

Gets the generated image of given data. Same as *val_step()*.

参数 **data** (*dict*) –Data sampled from metric specific sampler. More detials in *Metrics* and *Evaluator*.

返回 Generated image or image dict.

返回类型 List[EditDataSample]

train_discriminator(*inputs: dict*, *data_samples: List[mmedit.structures.edit_data_sample.EditDataSample]*, *optimizer_wrapper: mmengine.optim.optimizer.optimizer_wrapper.OptimWrapper*) → Dict[str, torch.Tensor]

Training function for discriminator. All GANs should implement this function by themselves.

参数

- **inputs** (*dict*) –Inputs from dataloader.
- **data_samples** (*List[EditDataSample]*) –Data samples from dataloader.
- **optim_wrapper** (*OptimWrapper*) –OptimWrapper instance used to update model parameters.

返回 A `dict` of tensor for logging.

返回类型 Dict[str, Tensor]

train_generator(*inputs: dict*, *data_samples: List[mmedit.structures.edit_data_sample.EditDataSample]*, *optimizer_wrapper: mmengine.optim.optimizer.optimizer_wrapper.OptimWrapper*) → Dict[str, torch.Tensor]

Training function for discriminator. All GANs should implement this function by themselves.

参数

- **inputs** (*dict*) –Inputs from dataloader.

- **data_samples** (*List[EditDataSample]*) –Data samples from dataloader.
- **optim_wrapper** (*OptimWrapper*) –OptimWrapper instance used to update model parameters.

返回 A `dict` of tensor for logging.

返回类型 Dict[str, Tensor]

train_step(*data: dict*, *optim_wrapper: mmengine.optim.optimizer.optimizer_wrapper_dict.OptimWrapperDict*) → Dict[str, torch.Tensor]

Train GAN model. In the training of GAN models, generator and discriminator are updated alternatively. In MMEditing's design, *self.train_step* is called with data input. Therefore we always update discriminator, whose updating is relay on real data, and then determine if the generator needs to be updated based on the current number of iterations. More details about whether to update generator can be found in `should_gen_update()`.

参数

- **data** (*dict*) –Data sampled from dataloader.
- **optim_wrapper** (*OptimWrapperDict*) –OptimWrapperDict instance contains OptimWrapper of generator and discriminator.

返回 A `dict` of tensor for logging.

返回类型 Dict[str, torch.Tensor]

val_step(*data: dict*) → Sequence[mmengine.structures.base_data_element.BaseDataElement]

Gets the generated image of given data.

Calls `self.data_preprocessor(data)` and `self(inputs, data_sample, mode=None)` in order. Return the generated results which will be passed to evaluator.

参数 **data** (*dict*) –Data sampled from metric specific sampler. More detials in *Metrics* and *Evaluator*.

返回 Generated image or image dict.

返回类型 SampleList

property with_ema_gen: bool

Whether the GAN adopts exponential moving average.

返回

If *True*, means this GAN model is adopted to exponential moving average and vice versa.

返回类型 bool

class mmedit.models.base_models.**BaseMattor**(*data_preprocessor: Union[dict, mmengine.config.config.Config]*, *backbone: dict*, *init_cfg: Optional[dict] = None*, *train_cfg: Optional[dict] = None*, *test_cfg: Optional[dict] = None*)

Base class for trimap-based matting models.

A matting model must contain a backbone which produces *pred_alpha*, a dense prediction with the same height and width of input image. In some cases (such as DIM), the model has a refiner which refines the prediction of the backbone.

Subclasses should overwrite the following functions:

- `_forward_train()`, to return a loss
- `_forward_test()`, to return a prediction
- `_forward()`, to return raw tensors

For test, this base class provides functions to resize inputs and post-process pred_alphas to get predictions

参数

- **backbone** (*dict*) –Config of backbone.
- **data_preprocessor** (*dict*) –Config of data_preprocessor. See `MattorPreprocessor` for details.
- **init_cfg** (*dict, optional*) –Initialization config dict.
- **train_cfg** (*dict*) –Config of training. Customized by subclassesCustomized bu In `train_cfg`, `train_backbone` should be specified. If the model has a refiner, `train_refiner` should be specified.
- **test_cfg** (*dict*) –Config of testing. In `test_cfg`, If the model has a refiner, `train_refiner` should be specified.

forward(*inputs: torch.Tensor*, *data_samples: Optional[Union[list, torch.Tensor]] = None*, *mode: str = 'tensor'*) → List[mmedit.structures.edit_data_sample.EditDataSample]

General forward function.

参数

- **inputs** (*torch.Tensor*) –A batch of inputs. with image and trimap concatenated alone channel dimension.
- **data_samples** (*List[EditDataSample], optional*) –A list of data samples, containing: - Ground-truth alpha / foreground / background to compute loss - other meta information
- **mode** (*str*) –mode should be one of `loss`, `predict` and `tensor`. Default: 'tensor'.

- `loss`: Called by `train_step` and return loss `dict` used for logging
- `predict`: Called by `val_step` and `test_step` and return list of `BaseDataElement` results used for computing metric.
- `tensor`: Called by custom use to get `Tensor` type results.

返回 Sequence of predictions packed into EditDataElement

返回类型 List[EditDataElement]

postprocess(*batch_pred_alpha: torch.Tensor*, *data_samples: List[mmedit.structures.edit_data_sample.EditDataSample]*) → List[mmedit.structures.edit_data_sample.EditDataSample]

Post-process alpha predictions.

This function contains the following steps:

1. Restore padding or interpolation
2. Mask alpha prediction with trimap
3. Clamp alpha prediction to 0-1
4. Convert alpha prediction to uint8
5. Pack alpha prediction into EditDataSample

Currently only batch_size 1 is actually supported.

参数

- **batch_pred_alpha** (*torch.Tensor*) –A batch of predicted alpha of shape (N, 1, H, W).
- **data_samples** (*List[EditDataSample]*) –List of data samples.

返回

A list of predictions. Each data sample contains a pred_alpha, which is a torch.Tensor with dtype=uint8, device=cuda:0

返回类型 List[EditDataSample]

resize_inputs(*batch_inputs*)

Pad or interpolate images and trimaps to multiple of given factor.

restore_size(*pred_alpha*, *data_sample*)

Restore the predicted alpha to the original shape.

The shape of the predicted alpha may not be the same as the shape of original input image. This function restores the shape of the predicted alpha.

参数

- **pred_alpha** (*torch.Tensor*) –A single predicted alpha of shape (1, H, W).
- **data_sample** (*EditDataSample*) –Data sample containing original shape as meta data.

返回 The reshaped predicted alpha.

返回类型 torch.Tensor

class mmedit.models.base_models.**BaseTranslationModel**(*generator*, *discriminator*, *default_domain: str*, *reachable_domains: List[str]*, *related_domains: List[str]*, *data_preprocessor*, *discriminator_steps: int = 1*, *disc_init_steps: int = 0*, *real_img_key: str = 'real_img'*, *loss_config: Optional[dict] = None*)

Base Translation Model.

Translation models can transfer images from one domain to another. Domain information like *default_domain*, *reachable_domains* are needed to initialize the class. And we also provide query functions like *is_domain_reachable*, *get_other_domains*.

You can get a specific generator based on the domain, and by specifying *target_domain* in the forward function, you can decide the domain of generated images. Considering the difference among different image translation models, we only provide the external interfaces mentioned above. When you implement image translation with a specific method, you can inherit both *BaseTranslationModel* and the method (e.g BaseGAN) and implement abstract methods.

参数

- **default_domain** (*str*) –Default output domain.
- **reachable_domains** (*list[str]*) –Domains that can be generated by the model.
- **related_domains** (*list[str]*) –Domains involved in training and testing. *reachable_domains* must be contained in *related_domains*. However, related_domains may contain source domains that are used to retrieve source images from data_batch but not in reachable_domains.
- **discriminator_steps** (*int*) –The number of times the discriminator is completely updated before the generator is updated. Defaults to 1.
- **disc_init_steps** (*int*) –The number of initial steps used only to train discriminators.

forward(*img*, *test_mode=False*, ***kwargs*)

Forward function.

参数

- **img** (*tensor*) –Input image tensor.
- **test_mode** (*bool*) –Whether in test mode or not. Default: False.
- **kwargs** (*dict*) –Other arguments.

forward_test(*img*, *target_domain*, ***kwargs*)

Forward function for testing.

参数

- **img** (*tensor*) –Input image tensor.
- **target_domain** (*str*) –Target domain of output image.
- **kwargs** (*dict*) –Other arguments.

返回 Forward results.

返回类型 dict

forward_train(*img*, *target_domain*, ***kwargs*)

Forward function for training.

参数

- **img** (*tensor*) –Input image tensor.
- **target_domain** (*str*) –Target domain of output image.
- **kwargs** (*dict*) –Other arguments.

返回 Forward results.

返回类型 dict

get_module(*module*)

Get *nn.ModuleDict* to fit the *MMDistributedDataParallel* interface.

参数 **module** (*MMDistributedDataParallel | nn.ModuleDict*) –The input module that needs processing.

返回 The ModuleDict of multiple networks.

返回类型 nn.ModuleDict

get_other_domains(*domain*)

get other domains.

init_weights(*pretrained=None*)

Initialize weights for the model.

参数 **pretrained**(*str, optional*) –Path for pretrained weights. If given None, pretrained weights will not be loaded. Default: None.

is_domain_reachable(*domain*)

Whether image of this domain can be generated.

translation(*image*, *target_domain=None*, ***kwargs*)

Translation Image to target style.

参数

- **image** (*tensor*) –Image tensor with a shape of (N, C, H, W).
- **target_domain** (*str, optional*) –Target domain of output image. Default to None.

返回 Image tensor of target style.

返回类型 dict

class mmedit.models.base_models.**BasicInterpolator**(*generator*, *pixel_loss*, *train_cfg=None*, *test_cfg=None*, *required_frames=2*, *step_frames=1*, *init_cfg=None*, *data_preprocessor=None*)

Basic model for video interpolation.

It must contain a generator that takes frames as inputs and outputs an interpolated frame. It also has a pixel-wise loss for training.

参数

- **generator** (*dict*) –Config for the generator structure.
- **pixel_loss** (*dict*) –Config for pixel-wise loss.
- **train_cfg** (*dict*) –Config for training. Default: None.
- **test_cfg** (*dict*) –Config for testing. Default: None.
- **required_frames** (*int*) –Required frames in each process. Default: 2
- **step_frames** (*int*) –Step size of video frame interpolation. Default: 1
- **init_cfg** (*dict, optional*) –The weight initialized config for BaseModule.
- **data_preprocessor** (*dict, optional*) –The pre-process config of BaseDataPreprocessor.

init_cfg

Initialization config dict.

Type dict, optional

data_preprocessor

Used for pre-processing data sampled by dataloader to the format accepted by forward().

Type BaseDataPreprocessor

static merge_frames(*input_tensors*, *output_tensors*)

merge input frames and output frames.

Interpolate a frame between the given two frames.

Merged from [[in1, in2], [in2, in3], [in3, in4], ···] [[out1], [out2], [out3], ···]

to [in1, out1, in2, out2, in3, out3, in4, ···]

参数

- **input_tensors** (*Tensor*) –The input frames with shape [n, 2, c, h, w]
- **output_tensors** (*Tensor*) –The output frames with shape [n, 1, c, h, w].

返回 The final frames.

返回类型 list[np.array]

split_frames(*input_tensors*)

split input tensors for inference.

参数 **input_tensors** (*Tensor*) –Tensor of input frames with shape [1, t, c, h, w]

返回 Split tensor with shape [t-1, 2, c, h, w]

返回类型 Tensor

class mmedit.models.base_models.**ExponentialMovingAverage**(*model: torch.nn.modules.module.Module*, *momentum: float = 0.0002*, *interval: int = 1*, *device: Optional[torch.device] = None*, *update_buffers: bool = False*)

Implements the exponential moving average (EMA) of the model.

All parameters are updated by the formula as below:

$$Xema_{t+1} = (1 - momentum) * Xema_t + momentum * X_t$$

参数

- **model** (*nn.Module*) –The model to be averaged.
- **momentum** (*float*) –The momentum used for updating ema parameter. Defaults to 0.0002. Ema's parameter are updated with the formula $averaged\_param = (1 - momentum) * averaged\_param + momentum * source\_param$.
- **interval** (*int*) –Interval between two updates. Defaults to 1.

- **device** (*torch.device, optional*) –If provided, the averaged model will be stored on the device. Defaults to None.
- **update_buffers** (*bool*) –if True, it will compute running averages for both the parameters and the buffers of the model. Defaults to False.

avg_func(*averaged_param: torch.Tensor*, *source_param: torch.Tensor*, *steps: int*) → None

Compute the moving average of the parameters using exponential moving average.

参数

- **averaged_param** (*Tensor*) –The averaged parameters.
- **source_param** (*Tensor*) –The source parameters.
- **steps** (*int*) –The number of times the parameters have been updated.

sync_buffers(*model: torch.nn.modules.module.Module*) → None

Copy buffer from model to averaged model.

参数 **model** (*nn.Module*) –The model whose parameters will be averaged.

sync_parameters(*model: torch.nn.modules.module.Module*) → None

Copy buffer and parameters from model to averaged model.

参数 **model** (*nn.Module*) –The model whose parameters will be averaged.

class mmedit.models.base_models.**OneStageInpaintor**(*data_preprocessor: Union[dict, mmengine.config.config.Config]*, *encdec*, *disc=None*, *loss_gan=None*, *loss_gp=None*, *loss_disc_shift=None*, *loss_composed_percep=None*, *loss_out_percep=False*, *loss_l1_hole=None*, *loss_l1_valid=None*, *loss_tv=None*, *train_cfg=None*, *test_cfg=None*, *init_cfg: Optional[dict] = None*)

Standard one-stage inpaintor with commonly used losses.

An inpaintor must contain an encoder-decoder style generator to inpaint masked regions. A discriminator will be adopted when adversarial training is needed.

In this class, we provide a common interface for inpaintors. For other inpaintors, only some funcs may be modified to fit the input style or training schedule.

参数

- **data_preprocessor** (*dict*) –Config of data_preprocessor.
- **encdec** (*dict*) –Config for encoder-decoder style generator.

- **disc** (*dict*) –Config for discriminator.
- **loss_gan** (*dict*) –Config for adversarial loss.
- **loss_gp** (*dict*) –Config for gradient penalty loss.
- **loss_disc_shift** (*dict*) –Config for discriminator shift loss.
- **loss_composed_percep** (*dict*) –Config for perceptual and style loss with composed image as input.
- **loss_out_percep** (*dict*) –Config for perceptual and style loss with direct output as input.
- **loss_l1_hole** (*dict*) –Config for l1 loss in the hole.
- **loss_l1_valid** (*dict*) –Config for l1 loss in the valid region.
- **loss_tv** (*dict*) –Config for total variation loss.
- **train_cfg** (*dict*) –Configs for training scheduler. *disc_step* must be contained for indicates the discriminator updating steps in each training step.
- **test_cfg** (*dict*) –Configs for testing scheduler.
- **init_cfg** (*dict, optional*) –Initialization config dict.

forward (*inputs*, *data_samples*, *mode='tensor'*)

Forward function.

参数

- **inputs** (*torch.Tensor*) –batch input tensor collated by `data_preprocessor`.
- **data_samples** (*List[BaseDataElement], optional*) –data samples collated by `data_preprocessor`.
- **mode** (*str*) –mode should be one of `loss`, `predict` and `tensor`. Default: 'tensor'.
  - `loss`: Called by `train_step` and return loss `dict` used for logging
  - `predict`: Called by `val_step` and `test_step` and return list of `BaseDataElement` results used for computing metric.
  - `tensor`: Called by custom use to get `Tensor` type results.

返回

- If `mode == loss`, return a `dict` of loss tensor used for backward and logging.
- If `mode == predict`, return a `list` of `BaseDataElement` for computing metric and getting inference result.
- If `mode == tensor`, return a tensor or `tuple` of tensor or `dict` or tensor for custom use.

返回类型 ForwardResults

forward_dummy(*x*)

Forward dummy function for getting flops.

参数 **x** (*torch.Tensor*) –Input tensor with shape of (n, c, h, w).

返回 Results tensor with shape of (n, 3, h, w).

返回类型 torch.Tensor

forward_tensor(*inputs*, *data_samples*)

Forward function in tensor mode.

参数

- **inputs** (*torch.Tensor*) –Input tensor.
- **data_samples** (*List[dict]*) –List of data sample dict.

返回

Direct output of the generator and composition of *fake_res* and ground-truth image.

返回类型 tuple

forward_test(*inputs*, *data_samples*)

Forward function for testing.

参数

- **inputs** (*torch.Tensor*) –Input tensor.
- **data_samples** (*List[dict]*) –List of data sample dict.

返回

List of prediction saved in DataSample.

返回类型 predictions (List[DataSample])

forward_train(**args*, ***kwargs*)

Forward function for training.

In this version, we do not use this interface.

forward_train_d(*data_batch*, *is_real*, *is_disc*)

Forward function in discriminator training step.

In this function, we compute the prediction for each data batch (real or fake). Meanwhile, the standard gan loss will be computed with several proposed losses for stable training.

参数

- **data_batch** (*torch.Tensor*) –Batch of real data or fake data.

- **is_real** (*bool*) –If True, the gan loss will regard this batch as real data. Otherwise, the gan loss will regard this batch as fake data.
- **is_disc** (*bool*) –If True, this function is called in discriminator training step. Otherwise, this function is called in generator training step. This will help us to compute different types of adversarial loss, like LSGAN.

返回 Contains the loss items computed in this function.

返回类型 dict

generator_loss(*fake_res*, *fake_img*, *gt*, *mask*, *masked_img*)

Forward function in generator training step.

In this function, we mainly compute the loss items for generator with the given (fake_res, fake_img). In general, the *fake_res* is the direct output of the generator and the *fake_img* is the composition of direct output and ground-truth image.

参数

- **fake_res** (*torch.Tensor*) –Direct output of the generator.
- **fake_img** (*torch.Tensor*) –Composition of *fake_res* and ground-truth image.
- **gt** (*torch.Tensor*) –Ground-truth image.
- **mask** (*torch.Tensor*) –Mask image.
- **masked_img** (*torch.Tensor*) –Composition of mask image and ground-truth image.

返回 Dict contains the results computed within this function for visualization and dict contains the loss items computed in this function.

返回类型 tuple(dict)

train_step(*data: List[dict]*, *optim_wrapper*)

Train step function.

In this function, the inpaintor will finish the train step following the pipeline:

1. get fake res/image
2. optimize discriminator (if have)
3. optimize generator

If *self.train_cfg.disc_step > 1*, the train step will contain multiple iterations for optimizing discriminator with different input data and only one iteration for optimizing gerator after *disc_step* iterations for discriminator.

参数

- **data** (*List[dict]*) –Batch of data as input.
- **optim_wrapper** (*dict[torch.optim.Optimizer]*) –Dict with optimizers for generator and discriminator (if have).

返回

Dict with loss, information for logger, the number of samples and results for visualization.

返回类型 dict

class mmedit.models.base_models.**RampUpEMA**(*model: torch.nn.modules.module.Module*, *interval: int = 1*, *ema_kimg: int = 10*, *ema_rampup: float = 0.05*, *batch_size: int = 32*, *eps: float = 1e-08*, *start_iter: int = 0*, *device: Optional[torch.device] = None*, *update_buffers: bool = False*)

Implements the exponential moving average with ramping up momentum.

Ref: https://github.com/NVlabs/stylegan3/blob/master/training/training_loop.py # noqa

参数

- **model** (*nn.Module*) –The model to be averaged.
- **interval** (*int*) –Interval between two updates. Defaults to 1.
- **ema_kimg** (*int, optional*) –EMA kimgs. Defaults to 10.
- **ema_rampup** (*float, optional*) –Ramp up rate. Defaults to 0.05.
- **batch_size** (*int, optional*) –Global batch size. Defaults to 32.
- **eps** (*float, optional*) –Ramp up epsilon. Defaults to 1e-8.
- **start_iter** (*int, optional*) –EMA start iter. Defaults to 0.
- **device** (*torch.device, optional*) –If provided, the averaged model will be stored on the `device`. Defaults to None.
- **update_buffers** (*bool*) –if True, it will compute running averages for both the parameters and the buffers of the model. Defaults to False.

avg_func(*averaged_param: torch.Tensor*, *source_param: torch.Tensor*, *steps: int*) → None

Compute the moving average of the parameters using exponential moving average.

参数

- **averaged_param** (*Tensor*) –The averaged parameters.
- **source_param** (*Tensor*) –The source parameters.
- **steps** (*int*) –The number of times the parameters have been updated.

static rampup(*steps*, *ema_kimg=10*, *ema_rampup=0.05*, *batch_size=4*, *eps=1e-08*)

Ramp up ema momentum.

Ref: https://github.com/NVlabs/stylegan3/blob/a5a69f58294509598714d1e88c9646c3d7c6ec94/training/training_loop.py#L300-L308 # noqa

参数

- **steps** –
- **ema_kimg** (*int, optional*) –Half-life of the exponential moving average of generator weights. Defaults to 10.
- **ema_rampup** (*float, optional*) –EMA ramp-up coefficient.If set to None, then rampup will be disabled. Defaults to 0.05.
- **batch_size** (*int, optional*) –Total batch size for one training iteration. Defaults to 4.
- **eps** (*float, optional*) –Epsiolon to avoid `batch_size` divided by zero. Defaults to 1e-8.

返回 Updated momentum.

返回类型 dict

sync_buffers(*model: torch.nn.modules.module.Module*) → None

Copy buffer from model to averaged model.

参数 **model** (*nn.Module*) –The model whose parameters will be averaged.

sync_parameters(*model: torch.nn.modules.module.Module*) → None

Copy buffer and parameters from model to averaged model.

参数 **model** (*nn.Module*) –The model whose parameters will be averaged.

class mmedit.models.base_models.**TwoStageInpaintor**(*data_preprocessor: Union[dict, mmengine.config.config.Config]*, *encdec: dict*, *disc=None*, *loss_gan=None*, *loss_gp=None*, *loss_disc_shift=None*, *loss_composed_percep=None*, *loss_out_percep=False*, *loss_l1_hole=None*, *loss_l1_valid=None*, *loss_tv=None*, *train_cfg=None*, *test_cfg=None*, *init_cfg: Optional[dict] = None*, *stage1_loss_type=('loss_l1_hole',)*, *stage2_loss_type=('loss_l1_hole', 'loss_gan')*, *input_with_ones=True*, *disc_input_with_mask=False*)

Standard two-stage inpaintor with commonly used losses. A two-stage inpaintor contains two encoder-decoder style generators to inpaint masked regions. Currently, we support these loss types in each of two stage inpaintors:

['loss_gan' , 'loss_l1_hole' , 'loss_l1_valid' , 'loss_composed_percep' , 'loss_out_percep' , 'loss_tv'] The *stage1_loss_type* and *stage2_loss_type* should be chosen from these loss types.

参数

- **data_preprocessor** (*dict*) –Config of data_preprocessor.

- **encdec** (*dict*) –Config for encoder-decoder style generator.
- **disc** (*dict*) –Config for discriminator.
- **loss_gan** (*dict*) –Config for adversarial loss.
- **loss_gp** (*dict*) –Config for gradient penalty loss.
- **loss_disc_shift** (*dict*) –Config for discriminator shift loss.
- **loss_composed_percep** (*dict*) –Config for perceptual and style loss with composed image as input.
- **loss_out_percep** (*dict*) –Config for perceptual and style loss with direct output as input.
- **loss_l1_hole** (*dict*) –Config for l1 loss in the hole.
- **loss_l1_valid** (*dict*) –Config for l1 loss in the valid region.
- **loss_tv** (*dict*) –Config for total variation loss.
- **train_cfg** (*dict*) –Configs for training scheduler. *disc_step* must be contained for indicates the discriminator updating steps in each training step.
- **test_cfg** (*dict*) –Configs for testing scheduler.
- **init_cfg** (*dict, optional*) –Initialization config dict.
- **stage1_loss_type** (*tuple[str]*) –Contains the loss names used in the first stage model. Default: (‘loss_l1_hole’).
- **stage2_loss_type** (*tuple[str]*) –Contains the loss names used in the second stage model. Default: (‘loss_l1_hole’, ‘loss_gan’).
- **input_with_ones** (*bool*) –Whether to concatenate an extra ones tensor in input. Default: True.
- **disc_input_with_mask** (*bool*) –Whether to add mask as input in discriminator. Default: False.

calculate_loss_with_type(*loss_type*, *fake_res*, *fake_img*, *gt*, *mask*, *prefix='stage1_'*)

Calculate multiple types of losses.

参数

- **loss_type** (*str*) –Type of the loss.
- **fake_res** (*torch.Tensor*) –Direct results from model.
- **fake_img** (*torch.Tensor*) –Composited results from model.
- **gt** (*torch.Tensor*) –Ground-truth tensor.
- **mask** (*torch.Tensor*) –Mask tensor.

- **prefix** (*str, optional*) –Prefix for loss name. Defaults to ‘stage1_’. # noqa

返回 Contain loss value with its name.

返回类型 dict

forward_tensor(*inputs*, *data_samples*)

Forward function in tensor mode.

参数

- **inputs** (*torch.Tensor*) –Input tensor.
- **data_samples** (*List[dict]*) –List of data sample dict.

返回 Dict contains output results.

返回类型 dict

train_step(*data: List[dict]*, *optim_wrapper*)

Train step function.

In this function, the inpaintor will finish the train step following the pipeline: 1. get fake res/image 2. optimize discriminator (if have) 3. optimize generator

If *self.train_cfg.disc_step > 1*, the train step will contain multiple iterations for optimizing discriminator with different input data and only one iteration for optimizing gerator after *disc_step* iterations for discriminator.

参数

- **data** (*List[dict]*) –Batch of data as input.
- **optim_wrapper** (*dict[torch.optim.Optimizer]*) –Dict with optimizers for generator and discriminator (if have).

返回 Dict with loss, information for logger, the number of samples and results for visualization.

返回类型 dict

two_stage_loss(*stage1_data*, *stage2_data*, *gt*, *mask*, *masked_img*)

Calculate two-stage loss.

参数

- **stage1_data** (*dict*) –Contain stage1 results.
- **stage2_data** (*dict*) –Contain stage2 results..
- **gt** (*torch.Tensor*) –Ground-truth image.
- **mask** (*torch.Tensor*) –Mask image.
- **masked_img** (*torch.Tensor*) –Composition of mask image and ground-truth image.

返回 Dict contains the results computed within this function for visualization and dict contains the loss items computed in this function.

返回类型 tuple(dict)

18.2 data_preprocessors

class mmedit.models.data_preprocessors.**EditDataPreprocessor**(*mean: Sequence[Union[float, int]] = (0, 0, 0), std: Sequence[Union[float, int]] = (255, 255, 255), pad_size_divisor: int = 1, input_view=(- 1, 1, 1), output_view=None, pad_args: dict = {}*)

Basic data pre-processor used for collating and copying data to the target device in mmediting.

`EditDataPreprocessor` performs data pre-processing according to the following steps:

- Collates the data sampled from dataloader.
- Copies data to the target device.
- Stacks the input tensor at the first dimension.

and post-processing of the output tensor of model.

TODO: Most editing methods have crop inputs to a same size, batched padding will be faster.

参数

- **mean** (*Sequence[float or int]*) –The pixel mean of R, G, B channels. Defaults to (0, 0, 0). If `mean` and `std` are not specified, ImgDataPreprocessor will normalize images to [0, 1].
- **std** (*Sequence[float or int]*) –The pixel standard deviation of R, G, B channels. (255, 255, 255). If `mean` and `std` are not specified, ImgDataPreprocessor will normalize images to [0, 1].
- **pad_size_divisor** (*int*) –The size of padded image should be divisible by `pad_size_divisor`. Defaults to 1.
- **input_view** (*Tuple | List*) –Tensor view of mean and std for input (without batch). Defaults to (-1, 1, 1) for (C, H, W).
- **output_view** (*Tuple | List | None*) –Tensor view of mean and std for output (without batch). If None, output_view=input_view. Defaults: None.
- **pad_args** (*dict*) –Args of F.pad. Default: dict().

destructor(*batch_tensor: torch.Tensor*)

Destructor of data processor. Destruct padding, normalization and dissolve batch.

参数 **batch_tensor** (*Tensor*) –Batched output.

返回 Destructed output.

返回类型 Tensor

forward(*data: Sequence[dict]*, *training: bool = False*) → Tuple[torch.Tensor, Optional[list]]

Pre-process the data into the model input format.

After the data pre-processing of `collate_data()`, `forward` will stack the input tensor list to a batch tensor at the first dimension.

参数

- **data** (*Sequence[dict]*) –data sampled from dataloader.
- **training** (*bool*) –Whether to enable training time augmentation. Default: False.

返回 Data in the same format as the model input.

返回类型 Tuple[torch.Tensor, Optional[list]]

class mmedit.models.data_preprocessors.**GenDataPreprocessor**(*mean: Sequence[Union[float, int]] = (127.5, 127.5, 127.5)*, *std: Sequence[Union[float, int]] = (127.5, 127.5, 127.5)*, *pad_size_divisor: int = 1*, *pad_value: Union[float, int] = 0*, *bgr_to_rgb: bool = False*, *rgb_to_bgr: bool = False*, *non_image_keys: Optional[Tuple[str, List[str]]] = None*, *non_concentate_keys: Optional[Tuple[str, List[str]]] = None*)

Image pre-processor for generative models. This class provide normalization and bgr to rgb conversion for image tensor inputs. The input of this classes should be dict which keys are *inputs* and *data_samples*.

Besides to process tensor *inputs*, this class support dict as *inputs*. - If the value is *Tensor* and the corresponding key is not contained in `_NON_IMAGE_KEYS`, it will be processed as image tensor. - If the value is *Tensor* and the corresponding key belongs to `_NON_IMAGE_KEYS`, it will not remains unchanged. - If value is string or integer, it will not remains unchanged.

参数

- **mean** (*Sequence[float or int], optional*) –The pixel mean of image channels. If `bgr_to_rgb=True` it means the mean value of R, G, B channels. If it is not specified, images will not be normalized. Defaults None.
- **std** (*Sequence[float or int], optional*) –The pixel standard deviation of image channels. If `bgr_to_rgb=True` it means the standard deviation of R, G, B channels. If it is not specified, images will not be normalized. Defaults None.
- **pad_size_divisor** (*int*) –The size of padded image should be divisible by `pad_size_divisor`. Defaults to 1.
- **pad_value** (*float or int*) –The padded pixel value. Defaults to 0.
- **bgr_to_rgb** (*bool*) –whether to convert image from BGR to RGB. Defaults to False.
- **rgb_to_bgr** (*bool*) –whether to convert image from RGB to RGB. Defaults to False.

cast_data(*data: Union[tuple, dict, mmengine.structures.base_data_element.BaseDataElement, torch.Tensor, list]*) → Union[tuple, dict, mmengine.structures.base_data_element.BaseDataElement, torch.Tensor, list]

Copying data to the target device.

参数 **data** (*dict*) –Data returned by `DataLoader`.

返回 Inputs and data sample at target device.

返回类型 CollatedResult

forward(*data: dict*, *training: bool = False*) → dict

Performs normalization、padding and bgr2rgb conversion based on `BaseDataPreprocessor`.

参数

- **data** (*dict*) –Input data to process.
- **training** (*bool*) –Whether to enable training time augmentation. This is ignored for *GenDataPreprocessor*. Defaults to False.

返回 Data in the same format as the model input.

返回类型 dict

process_dict_inputs(*batch_inputs: dict*) → dict

Preprocess dict type inputs.

参数 **batch_inputs** (*dict*) –Input dict.

返回 Preprocessed dict.

返回类型 dict

class mmedit.models.data_preprocessors.**MattorPreprocessor**(*mean: float = [123.675, 116.28, 103.53]*, *std: float = [58.395, 57.12, 57.375]*, *bgr_to_rgb: bool = True*, *proc_inputs: str = 'normalize'*, *proc_trimap: str = 'rescale_to_zero_one'*, *proc_gt: str = 'rescale_to_zero_one'*)

DataPreprocessor for matting models.

See base class `BaseDataPreprocessor` for detailed information.

Workflow as follow :

- Collate and move data to the target device.
- Convert inputs from bgr to rgb if the shape of input is (3, H, W).
- Normalize image with defined std and mean.
- Stack inputs to batch_inputs.

参数

- **mean** (*Sequence[float or int]*) –The pixel mean of R, G, B channels. Defaults to [123.675, 116.28, 103.53].
- **std** (*Sequence[float or int]*) –The pixel standard deviation of R, G, B channels. [58.395, 57.12, 57.375].
- **bgr_to_rgb** (*bool*) –whether to convert image from BGR to RGB. Defaults to True.
- **proc_inputs** (*str*) –Methods to process inputs. Default: 'normalize'. Available options are `normalize`.
- **proc_trimap** (*str*) –Methods to process gt tensors. Default: 'rescale_to_zero_one'. Available options are `rescale_to_zero_one` and `as-is`.
- **proc_gt** (*str*) –Methods to process gt tensors. Default: 'rescale_to_zero_one'. Available options are `rescale_to_zero_one` and `ignore`.

collate_data(*data: Sequence[dict]*) → Tuple[list, list, list]

Collating and moving data to the target device.

See base class `BaseDataPreprocessor` for detailed information.

forward(*data: Sequence[dict]*, *training: bool = False*) → Tuple[torch.Tensor, list]

Pre-process input images, trimaps, ground-truth as configured.

参数

- **data** (*Sequence[dict]*) –data sampled from dataloader.
- **training** (*bool*) –Whether to enable training time augmentation. Default: False.

返回 Batched inputs and list of data samples.

返回类型 Tuple[torch.Tensor, list]

mmedit.models.data_preprocessors.**split_batch**(*batch_tensor: torch.Tensor*, *padded_sizes: torch.Tensor*)

reverse operation of `stack_batch`.

参数

- **batch_tensor** (*Tensor*) –The 4D-tensor or 5D-tensor. Tensor.dim == tensor_list[0].dim + 1
- **padded_sizes** (*Tensor*) –The padded sizes of each tensor.

返回 A list of tensors with the same dim.

返回类型 tensor_list (List[Tensor])

mmedit.models.data_preprocessors.**stack_batch**(*tensor_list: List[torch.Tensor]*, *pad_size_divisor: int = 1*, *pad_args: dict = {}*)

Stack multiple tensors to form a batch and pad the images to the max shape use the right bottom padding mode in these images.

If `pad_size_divisor > 0`, add padding to ensure the shape of each dim is divisible by `pad_size_divisor`.

参数

- **tensor_list** (*List[Tensor]*) –A list of tensors with the same dim.
- **pad_size_divisor** (*int*) –If `pad_size_divisor > 0`, add padding to ensure the shape of each dim is divisible by `pad_size_divisor`. This depends on the model, and many models need to be divisible by 32. Defaults to 1
- **pad_args** (*dict*) –The padding args.

返回 The 4D-tensor or 5D-tensor. Tensor.dim == tensor_list[0].dim + 1 padded_sizes (Tensor): The padded sizes of each tensor.

返回类型 batch_tensor (Tensor)

18.3 layers

18.4 losses

18.5 utils

mmedit.models.utils.**default_init_weights**(*module*, *scale=1*)

Initialize network weights.

参数

- **modules** (*nn.Module*) –Modules to be initialized.
- **scale** (*float*) –Scale initialized weights, especially for residual blocks. Default: 1.

mmedit.models.utils.**extract_around_bbox**(*img*, *bbox*, *target_size*, *channel_first=True*)

Extract patches around the given bbox.

参数

- **img** (*torch.Tensor | numpy.array*) –Image data to be extracted. If organized in batch dimension, the batch dimension must be the first order like (n, h, w, c) or (n, c, h, w).
- **bbox** (*np.ndarray | torch.Tensor*) –Bboxes to be modified. Bbox can be in batch or not.
- **target_size** (*List(int)*) –Target size of final bbox.
- **channel_first** (*bool*) –If True, the channel dimension of img is before height and width, e.g. (c, h, w). Otherwise, the img shape (samples in the batch) is like (h, w, c). Default: True.

返回 Extracted patches. The dimension of the output should be the same as *img*.

返回类型 (torch.Tensor | np.ndarray)

mmedit.models.utils.**extract_bbox_patch**(*bbox*, *img*, *channel_first=True*)

Extract patch from a given bbox.

参数

- **bbox** (*torch.Tensor | numpy.array*) –Bbox with (top, left, h, w). If *img* has batch dimension, the *bbox* must be stacked at first dimension. The shape should be (4,) or (n, 4).
- **img** (*torch.Tensor | numpy.array*) –Image data to be extracted. If organized in batch dimension, the batch dimension must be the first order like (n, h, w, c) or (n, c, h, w).
- **channel_first** (*bool*) –If True, the channel dimension of img is before height and width, e.g. (c, h, w). Otherwise, the img shape (samples in the batch) is like (h, w, c). Default: True.

返回 Extracted patches. The dimension of the output should be the same as *img*.

返回类型 (torch.Tensor | numpy.array)

mmedit.models.utils.**flow_warp**(*x*, *flow*, *interpolation='bilinear'*, *padding_mode='zeros'*, *align_corners=True*)

Warp an image or a feature map with optical flow.

参数

- **x** (*Tensor*) –Tensor with size (n, c, h, w).
- **flow** (*Tensor*) –Tensor with size (n, h, w, 2). The last dimension is a two-channel, denoting the width and height relative offsets. Note that the values are not normalized to [-1, 1].
- **interpolation** (*str*) –Interpolation mode: ‘nearest’ or ‘bilinear’. Default: ‘bilinear’.
- **padding_mode** (*str*) –Padding mode: ‘zeros’ or ‘border’ or ‘reflection’. Default: ‘zeros’.
- **align_corners** (*bool*) –Whether align corners. Default: True.

返回 Warped image or feature map.

返回类型 Tensor

mmedit.models.utils.**generation_init_weights**(*module*, *init_type='normal'*, *init_gain=0.02*)

Default initialization of network weights for image generation.

By default, we use normal init, but xavier and kaiming might work better for some applications.

参数

- **module** (*nn.Module*) –Module to be initialized.
- **init_type** (*str*) –The name of an initialization method: normal | xavier | kaiming | orthogonal. Default: ‘normal’.
- **init_gain** (*float*) –Scaling factor for normal, xavier and orthogonal. Default: 0.02.

mmedit.models.utils.**get_module_device**(*module*)

Get the device of a module.

参数 **module** (*nn.Module*) –A module contains the parameters.

返回 The device of the module.

返回类型 torch.device

mmedit.models.utils.**get_unknown_tensor**(*trimap*, *unknown_value=0.5019607843137255*)

Get 1-channel unknown area tensor from the 3 or 1-channel trimap tensor.

参数

- **trimap** (*Tensor*) –Tensor with shape (N, 3, H, W) or (N, 1, H, W).

- **unknown_value** (*float*) –Scalar value indicating unknown region in trimap. If trimap is pre-processed using 'rescale_to_zero_one', then 0 for bg, 128/255 for unknown, 1 for fg, and unknown_value should set to 128 / 255. If trimap is pre-processed by `FormatTrimap(to_onehot=False)()`, then 0 for bg, 1 for unknown, 2 for fg and unknown_value should set to 1. If trimap is pre-processed by `FormatTrimap(to_onehot=True)()`, then trimap is 3-channeled, and this value is not used.

返回 Unknown area mask of shape (N, 1, H, W).

返回类型 Tensor

mmedit.models.utils.**get_valid_noise_size**(*noise_size: Optional[int]*, *generator: Union[Dict, torch.nn.modules.module.Module]*) → Optional[int]

Get the value of *noise_size* from input, *generator* and check the consistency of these values. If no conflict is found, return that value.

参数

- **noise_size** (*Optional[int]*) –*noise_size* passed to *BaseGAN_refactor*'s initialize function.
- **generator** (*ModelType*) –The config or the model of generator.

返回 The noise size feed to generator.

返回类型 int | None

mmedit.models.utils.**get_valid_num_batches**(*batch_inputs: Tuple[Dict[str, Union[torch.Tensor, str, int]], torch.Tensor]*) → int

Try get the valid batch size from inputs.

- If some values in *batch_inputs* are *Tensor* and 'num_batches' is in *batch_inputs*, we check whether the value of 'num_batches' and the the length of first dimension of all tensors are same. If the values are not same, *AssertionError* will be raised. If all values are the same, return the value.
- If no values in *batch_inputs* is *Tensor*, 'num_batches' must be contained in *batch_inputs*. And this value will be returned.
- If some values are *Tensor* and 'num_batches' is not contained in *batch_inputs*, we check whether all tensor have the same length on the first dimension. If the length are not same, *AssertionError* will be raised. If all length are the same, return the length as batch size.
- If batch_inputs is a *Tensor*, directly return the length of the first dimension as batch size.

参数 **batch_inputs** (*ForwardInputs*) –Inputs passed to `forward()`.

返回 The batch size of samples to generate.

返回类型 int

mmedit.models.utils.**label_sample_fn**(*label: Optional[Union[torch.Tensor, Callable, List[int]]] = None*, **, num_batches: int = 1*, *num_classes: Optional[int] = None*, *device: Optional[str] = None*) → Optional[torch.Tensor]

Sample random label with respect to *num_batches*, *num_classes* and *device*.

参数

- **label** (*Union[Tensor, Callable, List[int], None], optional*) –You can directly give a batch of label through a `torch.Tensor` or offer a callable function to sample a batch of label data. Otherwise, the `None` indicates to use the default label sampler. Defaults to None.
- **num_batches** (*int, optional*) –The number of batch size. Defaults to 1.
- **num_classes** (*Optional[int], optional*) –The number of classes. Defaults to None.
- **device** (*Optional[str], optional*) –The target device of the label. Defaults to None.

返回 Sampled random label.

返回类型 Union[Tensor, None]

mmedit.models.utils.**make_layer**(*block*, *num_blocks*, ***kwarg*)

Make layers by stacking the same blocks.

参数

- **block** (*nn.module*) –nn.module class for basic block.
- **num_blocks** (*int*) –number of blocks.

返回 Stacked blocks in nn.Sequential.

返回类型 nn.Sequential

mmedit.models.utils.**noise_sample_fn**(*noise: Optional[Union[torch.Tensor, Callable]] = None*, **, num_batches: int = 1*, *noise_size: Optional[Union[int, Sequence[int]]] = None*, *device: Optional[str] = None*) → torch.Tensor

Sample noise with respect to the given *num_batches*, *noise_size* and *device*.

参数

- **noise** (*torch.Tensor | callable | None*) –You can directly give a batch of noise through a `torch.Tensor` or offer a callable function to sample a batch of noise data. Otherwise, the `None` indicates to use the default noise sampler. Defaults to None.
- **num_batches** (*int, optional*) –The number of batch size. Defaults to 1.

- **noise_size** (*Union[int, Sequence[int], None], optional*) –The size of random noise. Defaults to None.
- **device** (*Optional[str], optional*) –The target device of the random noise. Defaults to None.

返回 Sampled random noise.

返回类型 Tensor

mmedit.models.utils.**normalize_vecs**(*vectors: torch.Tensor*) → torch.Tensor

Normalize vector with it's lengths at the last dimension. If *vector* is two-dimension tensor, this function is same as L2 normalization.

参数 **vector** (*torch.Tensor*) –Vectors to be normalized.

返回 Vectors after normalization.

返回类型 torch.Tensor

mmedit.models.utils.**set_requires_grad**(*nets*, *requires_grad=False*)

Set requires_grad for all the networks.

参数

- **nets** (*nn.Module | list[nn.Module]*) –A list of networks or a single network.
- **requires_grad** (*bool*) –Whether the networks require gradients or not

18.6 editors

CHAPTER 19

mmedit.visualization

class mmedit.visualization.**ConcatImageVisualizer**(*fn_key: str*, *img_keys: Sequence[str]*, *pixel_range={}*, *bgr2rgb=False*, *name: str = 'visualizer'*, **args*, ***kwargs*)

Visualize multiple images by concatenation.

This visualizer will horizontally concatenate images belongs to different keys and vertically concatenate images belongs to different frames to visualize.

Image to be visualized can be:

- torch.Tensor or np.array
- Image sequences of shape (T, C, H, W)
- Multi-channel image of shape (1/3, H, W)
- Single-channel image of shape (C, H, W)

参数

- **fn_key** (*str*) –key used to determine file name for saving image. Usually it is the path of some input image. If the value is *dir/basename.ext*, the name used for saving will be basename.
- **img_keys** (*str*) –keys, values of which are images to visualize.
- **pixel_range** (*dict*) –min and max pixel value used to denormalize images, note that only float array or tensor will be denormalized, uint8 arrays are assumed to be unnormalized.
- **bgr2rgb** (*bool*) –whether to convert the image from BGR to RGB.

- **name** (*str*) –name of visualizer. Default: ‘visualizer’.
- ****kwargs** (**args and*) –Other arguments are passed to *Visualizer*. # noqa

add_datasample(*data_sample: mmedit.structures.edit_data_sample.EditDataSample*, *step=0*) → None

Concatenate image and draw.

参数

- **input** (*torch.Tensor*) –Single input tensor from data_batch.
- **data_sample** (*EditDataSample*) –Single data_sample from data_batch.
- **output** (*EditDataSample*) –Single prediction output by model.
- **step** (*int*) –Global step value to record. Default: 0.

class mmedit.visualization.**GenVisBackend**(*save_dir: str*, *img_save_dir: str = 'vis_image'*, *config_save_file: str = 'config.py'*, *scalar_save_file: str = 'scalars.json'*, *ceph_path: Optional[str] = None*, *delete_local_image: bool = True*)

Generation visualization backend class. It can write image, config, scalars, etc. to the local hard disk and ceph path. You can get the drawing backend through the experiment property for custom drawing.

实际案例

```
>>> from mmgen.visualization import GenVisBackend
>>> import numpy as np
>>> vis_backend = GenVisBackend(save_dir='temp_dir',
>>>                             ceph_path='s3://temp-bucket')
>>> img = np.random.randint(0, 256, size=(10, 10, 3))
>>> vis_backend.add_image('img', img)
>>> vis_backend.add_scalar('mAP', 0.6)
>>> vis_backend.add_scalars({'loss': [1, 2, 3], 'acc': 0.8})
>>> cfg = Config(dict(a=1, b=dict(b1=[0, 1])))
>>> vis_backend.add_config(cfg)
```

参数

- **save_dir** (*str*) –The root directory to save the files produced by the visualizer.
- **img_save_dir** (*str*) –The directory to save images. Default to ‘vis_image’.
- **config_save_file** (*str*) –The file name to save config. Default to ‘config.py’.
- **scalar_save_file** (*str*) –The file name to save scalar values. Default to ‘scalars.json’.

- **ceph_path** (*Optional[str]*) –The remote path of Ceph cloud storage. Defaults to None.
- **delete_local** (*bool*) –Whether eelete local after uploading to ceph or not. If ceph_path is None, this will be ignored. Defaults to True.

add_config(*config: mmengine.config.config.Config*, ***kwargs*) → None

Record the config to disk.

参数 **config** (*Config*) –The Config object

add_image(*name: str*, *image: numpy.array*, *step: int = 0*, ***kwargs*) → None

Record the image to disk.

参数

- **name** (*str*) –The image identifier.
- **image** (*np.ndarray*) –The image to be saved. The format should be RGB. Default to None.
- **step** (*int*) –Global step value to record. Default to 0.

add_scalar(*name: str*, *value: Union[int, float, torch.Tensor, numpy.ndarray]*, *step: int = 0*, ***kwargs*) → None

Record the scalar data to disk.

参数

- **name** (*str*) –The scalar identifier.
- **value** (*int, float, torch.Tensor, np.ndarray*) –Value to save.
- **step** (*int*) –Global step value to record. Default to 0.

add_scalars(*scalar_dict: dict*, *step: int = 0*, *file_path: Optional[str] = None*, ***kwargs*) → None

Record the scalars to disk.

The scalar dict will be written to the default and specified files if file_path is specified.

参数

- **scalar_dict** (*dict*) –Key-value pair storing the tag and corresponding values. The value must be dumped into json format.
- **step** (*int*) –Global step value to record. Default to 0.
- **file_path** (*str, optional*) –The scalar's data will be saved to the file_path file at the same time if the file_path parameter is specified. Default to None.

property experiment: ***mmedit.visualization.vis_backend.GenVisBackend***

Return the experiment object associated with this visualization backend.

class mmedit.visualization.**GenVisualizer**(*name='visualizer'*, *vis_backends: Optional[List[Dict]] = None*, *save_dir: Optional[str] = None*)

MMEditing Visualizer.

参数

- **name** (*str*) –Name of the instance. Defaults to 'visualizer'.
- **vis_backends** (*list, optional*) –Visual backend config list. Defaults to None.
- **save_dir** (*str, optional*) –Save file dir for all storage backends. If it is None, the backend storage will not save any data.

Examples:

```
>>> # Draw image
>>> vis = GenVisualizer()
>>> vis.add_datasample(
>>>     'random_noise',
>>>     gen_samples=torch.rand(2, 3, 10, 10),
>>>     gt_samples=dict(imgs=torch.randn(2, 3, 10, 10)),
>>>     gt_keys='imgs',
>>>     vis_mode='image',
>>>     n_rows=2,
>>>     step=10)
```

add_datasample(*name: str*, ***, *gen_samples: Sequence[mmedit.structures.edit_data_sample.EditDataSample]*, *target_keys: Optional[Tuple[str, List[str]]] = None*, *vis_mode: Optional[str] = None*, *n_row: Optional[int] = 1*, *color_order: str = 'bgr'*, *target_mean: Sequence[Union[float, int]] = 127.5*, *target_std: Sequence[Union[float, int]] = 127.5*, *show: bool = False*, *wait_time: int = 0*, *step: int = 0*, ***kwargs*) → None

Draw datasample and save to all backends.

If GT and prediction are plotted at the same time, they are displayed in a stitched image where the left image is the ground truth and the right image is the prediction.

If `show` is True, all storage backends are ignored, and the images will be displayed in a local window.

参数

- **name** (*str*) –The image identifier.
- **gen_samples** (*List[EditDataSample]*) –Data samples to visualize.
- **vis_mode** (*str, optional*) –Visualization mode. If not passed, will visualize results as image. Defaults to None.
- **n_rows** (*int, optional*) –Number of images in one row. Defaults to 1.
- **color_order** (*str*) –The color order of the passed images. Defaults to 'bgr'.

- **target_mean** (*Sequence[Union[float, int]]*) –The target mean of the visualization results. Defaults to 127.5.
- **target_std** (*Sequence[Union[float, int]]*) –The target std of the visualization resutts. Defaults to 127.5.
- **show** (*bool*) –Whether to display the drawn image. Default to False.
- **wait_time** (*float*) –The interval of show (s). Defaults to 0.
- **step** (*int*) –Global step value to record. Defaults to 0.

add_image(*name: str*, *image: numpy.ndarray*, *step: int = 0*, ***kwargs*) → None

Record the image. Support input kwargs.

参数

- **name** (*str*) –The image identifier.
- **image** (*np.ndarray, optional*) –The image to be saved. The format should be RGB. Default to None.
- **step** (*int*) –Global step value to record. Default to 0.

class mmedit.visualization.**PaviGenVisBackend**(*save_dir: str*, *exp_name: Optional[str] = None*, *labels: Optional[str] = None*, *project: Optional[str] = None*, *model: Optional[str] = None*, *description: Optional[str] = None*)

Visualization backend for Pavi.

add_image(*name: str*, *image: numpy.array*, *step: int = 0*, ***kwargs*) → None

Record the image to Pavi.

参数

- **name** (*str*) –The image identifier.
- **image** (*np.ndarray*) –The image to be saved. The format should be RGB. Default to None.
- **step** (*int*) –Global step value to record. Default to 0.

add_scalar(*name: str*, *value: Union[int, float, torch.Tensor, numpy.ndarray]*, *step: int = 0*, ***kwargs*) → None

Record the scalar data to Pavi.

参数

- **name** (*str*) –The scalar identifier.
- **value** (*int, float, torch.Tensor, np.ndarray*) –Value to save.
- **step** (*int*) –Global step value to record. Default to 0.

add_scalars(*scalar_dict: dict*, *step: int = 0*, *file_path: Optional[str] = None*, ***kwargs*) → None

Record the scalars to Pavi.

The scalar dict will be written to the default and specified files if `file_path` is specified.

参数

- **scalar_dict** (*dict*) –Key-value pair storing the tag and corresponding values. The value must be dumped into json format.
- **step** (*int*) –Global step value to record. Default to 0.
- **file_path** (*str, optional*) –The scalar's data will be saved to the `file_path` file at the same time if the `file_path` parameter is specified. Default to None.

property experiment: ***mmedit.visualization.vis_backend.GenVisBackend***

Return the experiment object associated with this visualization backend.

class mmedit.visualization.**TensorboardGenVisBackend**(*save_dir: str*)

add_image(*name: str*, *image: numpy.array*, *step: int = 0*, ***kwargs*)

Record the image to Tensorboard. Additional support upload gif files.

参数

- **name** (*str*) –The image identifier.
- **image** (*np.ndarray*) –The image to be saved. The format should be RGB.
- **step** (*int*) –Useless parameter. Wandb does not need this parameter. Default to 0.

class mmedit.visualization.**WandbGenVisBackend**(*save_dir: str*, *init_kwargs: Optional[dict] = None*, *define_metric_cfg: Optional[dict] = None*, *commit: Optional[bool] = True*, *log_code_name: Optional[str] = None*, *watch_kwargs: Optional[dict] = None*)

Wandb visualization backend for MMEditing.

add_image(*name: str*, *image: numpy.array*, *step: int = 0*, ***kwargs*)

Record the image to wandb. Additional support upload gif files.

参数

- **name** (*str*) –The image identifier.
- **image** (*np.ndarray*) –The image to be saved. The format should be RGB.
- **step** (*int*) –Useless parameter. Wandb does not need this parameter. Default to 0.

CHAPTER 20

mmedit.utils

mmedit.utils.**add_gaussian_noise**(*img: numpy.ndarray*, *mu*, *sigma*)

Add Gaussian Noise on the input image.

参数

- **img** (*np.ndarray*) –Input image.
- **mu** (*float*) –The mu value of the Gaussian function.
- **sigma** (*float*) –The sigma value of the Gaussian function.

返回 Gaussian noisy output image.

返回类型 noisy_img (np.ndarray)

mmedit.utils.**adjust_gamma**(*image*, *gamma=1*, *gain=1*)

Performs Gamma Correction on the input image.

This function is adopted from skimage: https://github.com/scikit-image/scikit-image/blob/7e4840bd9439d1dfb6beaf549998452c99f97fdd/skimage/exposure/ exposure.py#L439-L494

Also known as Power Law Transform. This function transforms the input image pixelwise according to the equation `O = I**gamma` after scaling each pixel to the range 0 to 1.

参数

- **image** (*np.ndarray*) –Input image.
- **gamma** (*float, optional*) –Non negative real number. Defaults to 1.
- **gain** (*float, optional*) –The constant multiplier. Defaults to 1.

返回 Gamma corrected output image.

返回类型 np.ndarray

mmedit.utils.**bbox2mask**(*img_shape*, *bbox*, *dtype='uint8'*)

Generate mask in np.ndarray from bbox.

The returned mask has the shape of (h, w, 1). '1' indicates the hole and '0' indicates the valid regions.

We prefer to use *uint8* as the data type of masks, which may be different from other codes in the community.

参数

- **img_shape** (*tuple[int]*) –The size of the image.
- **bbox** (*tuple[int]*) –Configuration tuple, (top, left, height, width)
- **np.dtype** (*str*) –Indicate the data type of returned masks. Default: 'uint8'

返回 Mask in the shape of (h, w, 1).

返回类型 mask (np.ndarray)

mmedit.utils.**brush_stroke_mask**(*img_shape*, *num_vertices=(4, 12)*, *mean_angle=1.2566370614359172*, *angle_range=0.41887902047863906*, *brush_width=(12, 40)*, *max_loops=4*, *dtype='uint8'*)

Generate free-form mask.

The method of generating free-form mask is in the following paper: Free-Form Image Inpainting with Gated Convolution.

When you set the config of this type of mask. You may note the usage of *np.random.randint* and the range of *np.random.randint* is [left, right).

We prefer to use *uint8* as the data type of masks, which may be different from other codes in the community.

TODO: Rewrite the implementation of this function.

参数

- **img_shape** (*tuple[int]*) –Size of the image.
- **num_vertices** (*int | tuple[int]*) –Min and max number of vertices. If only give an integer, we will fix the number of vertices. Default: (4, 12).
- **mean_angle** (*float*) –Mean value of the angle in each vertex. The angle is measured in radians. Default: 2 * math.pi / 5.
- **angle_range** (*float*) –Range of the random angle. Default: 2 * math.pi / 15.
- **brush_width** (*int | tuple[int]*) –(min_width, max_width). If only give an integer, we will fix the width of brush. Default: (12, 40).
- **max_loops** (*int*) –The max number of for loops of drawing strokes. Default: 4.

- **np.dtype** (*str*) –Indicate the data type of returned masks. Default: 'uint8'.

返回 Mask in the shape of (h, w, 1).

返回类型 mask (np.ndarray)

mmedit.utils.**download_from_url**(*url*, *dest_path=None*, *dest_dir='/home/docs/.cache/openmmlab/mmedit/'*, *hash_prefix=None*)

Download object at the given URL to a local path.

参数

- **url** (*str*) –URL of the object to download.
- **dest_path** (*str*) –Path where object will be saved.
- **dest_dir** (*str*) –The directory of the destination. Defaults to `'~/.cache/openmmlab/mmgen/'`.
- **hash_prefix** (*string, optional*) –If not None, the SHA256 downloaded file should start with *hash_prefix*. Default: None.

返回 path for the downloaded file.

返回类型 str

mmedit.utils.**get_box_info**(*pred_bbox*, *original_shape*, *final_size*)

参数

- **pred_bbox** –The bounding box for the instance
- **original_shape** –Original image shape
- **final_size** –Size of the final output

返回 [L_pad, R_pad, T_pad, B_pad, rh, rw]

返回类型 List

mmedit.utils.**get_irregular_mask**(*img_shape*, *area_ratio_range=(0.15, 0.5)*, ***kwargs*)

Get irregular mask with the constraints in mask ratio.

参数

- **img_shape** (*tuple[int]*) –Size of the image.
- **area_ratio_range** (*tuple(float)*) –Contain the minimum and maximum area
- **Default** (*ratio.*) –(0.15, 0.5).

返回 Mask in the shape of (h, w, 1).

返回类型 mask (np.ndarray)

mmedit.utils.**get_sampler**(*sample_kwargs: dict*, *runner: Optional[mmengine.runner.runner.Runner]*)

Get a sampler to loop input data.

参数

- **sample_kwargs** (*dict*) –_description_
- **runner** (*Optional[Runner]*) –_description_

返回 _description_

返回类型 _type_

mmedit.utils.**make_coord**(*shape*, *ranges=None*, *flatten=True*)

Make coordinates at grid centers.

参数

- **shape** (*tuple*) –shape of image.
- **ranges** (*tuple*) –range of coordinate value. Default: None.
- **flatten** (*bool*) –flatten to (n, 2) or Not. Default: True.

返回 coordinates.

返回类型 coord (Tensor)

mmedit.utils.**modify_args**()

Modify args of argparse.ArgumentParser.

mmedit.utils.**print_colored_log**(*msg*, *level=20*, *color='magenta'*)

Print colored log with default logger.

参数

- **msg** (*str*) –Message to log.
- **level** (*int*) –The root logger level. Note that only the process of rank 0 is affected, while other processes will set the level to "Error" and be silent most of the time.Log level, default to 'info'.
- **color** (*str, optional*) –Color 'magenta'.

mmedit.utils.**random_bbox**(*img_shape*, *max_bbox_shape*, *max_bbox_delta=40*, *min_margin=20*)

Generate a random bbox for the mask on a given image.

In our implementation, the max value cannot be obtained since we use *np.random.randint*. And this may be different with other standard scripts in the community.

参数

- **img_shape** (*tuple[int]*) –The size of a image, in the form of (h, w).

- **max_bbox_shape** (*int | tuple[int]*) –Maximum shape of the mask box, in the form of (h, w). If it is an integer, the mask box will be square.
- **max_bbox_delta** (*int | tuple[int]*) –Maximum delta of the mask box, in the form of (delta_h, delta_w). If it is an integer, delta_h and delta_w will be the same. Mask shape will be randomly sampled from the range of *max_bbox_shape* - *max_bbox_delta* and *max_bbox_shape*. Default: (40, 40).
- **min_margin** (*int | tuple[int]*) –The minimum margin size from the edges of mask box to the image boarder, in the form of (margin_h, margin_w). If it is an integer, margin_h and margin_w will be the same. Default: (20, 20).

返回 The generated box, (top, left, h, w).

返回类型 tuple[int]

mmedit.utils.**random_choose_unknown**(*unknown*, *crop_size*)

Randomly choose an unknown start (top-left) point for a given crop_size.

参数

- **unknown** (*np.ndarray*) –The binary unknown mask.
- **crop_size** (*tuple[int]*) –The given crop size.

返回 The top-left point of the chosen bbox.

返回类型 tuple[int]

mmedit.utils.**register_all_modules**(*init_default_scope: bool = True*) → None

Register all modules in mmedit into the registries.

参数 **init_default_scope** (*bool*) –Whether initialize the mmedit default scope. When *init_default_scope=True*, the global default scope will be set to *mmedit*, and all registries will build modules from mmedit's registry node. To understand more about the registry, please refer to https://github.com/open-mmlab/mmengine/blob/main/docs/en/tutorials/registry.md Defaults to True.

mmedit.utils.**reorder_image**(*img*, *input_order='HWC'*)

Reorder images to 'HWC' order.

If the input_order is (h, w), return (h, w, 1); If the input_order is (c, h, w), return (h, w, c); If the input_order is (h, w, c), return as it is.

参数

- **img** (*np.ndarray*) –Input image.
- **input_order** (*str*) –Whether the input order is 'HWC' or 'CHW'. If the input image shape is (h, w), input_order will not have effects. Default: 'HWC'.

返回 Reordered image.

返回类型 np.ndarray

mmedit.utils.**tensor2img**(*tensor*, *out_type=<class 'numpy.uint8'>*, *min_max=(0, 1)*)

Convert torch Tensors into image numpy arrays.

After clamping to (min, max), image values will be normalized to [0, 1].

For different tensor shapes, this function will have different behaviors:

1. **4D mini-batch Tensor of shape (N x 3/1 x H x W):** Use *make_grid* to stitch images in the batch dimension, and then convert it to numpy array.
2. **3D Tensor of shape (3/1 x H x W) and 2D Tensor of shape (H x W):** Directly change to numpy array.

Note that the image channel in input tensors should be RGB order. This function will convert it to cv2 convention, i.e., (H x W x C) with BGR order.

参数

- **tensor** (*Tensor | list[Tensor]*) –Input tensors.
- **out_type** (*numpy type*) –Output types. If `np.uint8`, transform outputs to uint8 type with range [0, 255]; otherwise, float type with range [0, 1]. Default: `np.uint8`.
- **min_max** (*tuple*) –min and max values for clamp.

返回 3D ndarray of shape (H x W x C) or 2D ndarray of shape (H x W).

返回类型 (Tensor | list[Tensor])

mmedit.utils.**to_numpy**(*img*, *dtype=<class 'numpy.float64'>*)

Convert data into numpy arrays of dtype.

参数

- **img** (*Tensor | np.ndarray*) –Input data.
- **dtype** (*np.dtype*) –Set the data type of the output. Default: np.float64

返回 Converted numpy arrays data.

返回类型 img (np.ndarray)

mmedit.utils.**try_import**(*name: str*) → Optional[module]

Try to import a module.

参数 **name** (*str*) –Specifies what module to import in absolute or relative terms (e.g. either pkg.mod or ..mod).

返回 If importing successfully, returns the imported module, otherwise returns None.

返回类型 ModuleType or None

CHAPTER 21

总览（待更新）

CHAPTER 22

贡献指南（待更新）

CHAPTER 23

生态项目（待更新）

CHAPTER 24

变更日志 (待更新)

24.1 v1.0.0rc1(23/9/2022)

MMEditing 1.0.0rc1 已经合并了 MMGeneration 1.x。

- 支持 42+ 算法, 169+ 配置文件 and 168+ 预训练模型参数文件.
- 支持 26+ loss functions, 20+ metrics.
- 支持 tensorboard, wandb.
- 支持 unconditional GANs, conditional GANs, image2image translation 以及 internal learning.

24.2 v1.0.0rc0(31/8/2022)

MMEditing 1.0.0rc0 是 MMEditing 1.x 的第一个版本，是 OpenMMLab 2.0 项目的一部分。

基于新的训练引擎, MMEditing 1.x 统一了数据、模型、评测和可视化的接口。该版本存在有一些 BC-breaking 的修改。请在迁移指南中查看更多细节。

24.3 v0.15.0 (01/06/2022)

Highlights 主要更新

1. 支持 FLAVR
2. 支持 AOT-GAN
3. 在 CAIN 中支持 ReduceLROnPlateau 策略

新功能

- 支持 AOT-GAN (#681)
- 支持 Vimeo90k-triplet 数据集 (#810)
- 为 mm-assistant 添加默认 config (#827)
- 支持 CPU demo (#848)
- 在 `LoadImageFromFileList` 中支持 `use_cache` 和 `backend` (#857)
- 支持 VFIVimeo90K7FramesDataset (#858)
- 在 VFI pipeline 中支持 ColorJitter (#859)
- 支持 ReduceLrUpdaterHook (#860)
- 在 IterBaseRunner 中支持 `after_val_epoch` (#861)
- 支持 FLAVR Net (#866, #867, #897)
- 支持 MAE 评估方式 (#871)
- 使用 mdformat (#888)
- 在 CAIN 中支持 ReduceLROnPlateau 策略 (#906)

Bug 修复

- 在 restoration_demo.py 中将 - 改为 _ (#834)
- 移除 requirements/docs.txt 中的 recommonmark (#844)
- 将 README 中的 EDVR 移动到 VSR 类别中 (#849)
- 修改 crop.py，移除跨栏 F-string 中的 , (#855)
- 修改 test_pipeline，将重复的 `lq_path` 改为 `gt_path` (#862)
- 修复 TOF-VFI 的 unittest 问题 (#873)
- 解决 VFI demo 中帧序列出错问题 (#891)
- 修复 README 中的 logo & contrib 链接 (#898)
- 修复 indexnet_dimaug_mobv2_1x16_78k_comp1k.py (#901)

改进

- 在训练和测试脚本中增加 `--cfg-options` 参数 (#826)
- 更新 MMCV_MAX 到 1.6 (#829)
- 在 README 中更新 TOFlow (#835)
- 恢复 beirf 安装步骤，合并可选要求 (#836)
- 在 citation 中使用 {MMEditing Contributors} (#838)
- 增加定制损失函数的教程 (#839)
- 在 README 中添加安装指南 (wiki ver) (#845)
- 在中文文档中添加“需要帮助翻译”的说明 (#850)
- 在 README_zh-CN.md 中添加微信二维码 (#851)
- 支持 SRFolderVideoDataset 的非零帧索引，修复拼写错误 (#853)
- 创建 docker 的 README.md (#856)
- 优化 IO 流量偏差 (#881)
- 将 wiki/installation 移到 docs (#883)
- 添加 `myst_heading_anchors` (#887)
- 在 inpainting demo 中使用预训练模型链接 (#892)

贡献者

@wangruohui @quincylin1 @nijkah @jayagami @ckkelvinchan @ryanxingql @NK-CS-ZZL @Yshuo-Li

24.4 v0.14.0 (01/04/2022)

Highlights 主要更新

1. 支持视频插帧算法 TOFlow

新功能

- 支持 AOT-GAN (#677)
- 使用 `--diff-seed` 在多卡训练中为 torch 设置不同的初始化种子 (#781)
- 在视频插帧 demo 中支持流帧读取 (#790)
- 支持非 slurm 的 dist_train (#791)
- 在 restoration_video_demo 中将 LQ 存放在 CPU (#792)
- 在 EDSR 中支持处理灰度数据 (#793)
- 支持视频插帧算法 TOFlow (#806, #811)

- 支持在 DistributedSampler 中为不同的 rank 设置不同的随机种子 (#815)

Bug 修复

- 修复 README 文件中的失效链接 (#782, #786, #819, #820)
- 修复抠图教程 (#795)
- 翻转 `RandomAffine` 中的 `flip_ratio` (#799)
- 修复 preprocess_div2k_dataset.py (#801)
- 修复 SR Colab Demo Installation 方法和 Set5 链接 (#807)
- 修正 EDSR README 中的 Y/GRB 错误 (#812)
- 将 `README(_zh-CN).md` 中的 pytorch 安装命令替换为 conda (#816)

改进

- 更新 CI (#650)
- 更新 requirements.txt (#725, #817)
- 增加 dataset (#758), pipeline (#779), model (#766) 教程
- 更新 index 和 TOC 结构树 (#767)
- 更新 update_model_index.py 以兼容 Windows (#768)
- 更新文档构建系统 (#769)
- 更新 setuptools 的关键字和分类器 (#773)
- 更新安装文档 (#776, #800)
- 更新 BasicVSR++ 和 RealBasicVSR 文档 (#778)
- 更新 citation (#785, #787)
- 重组文档 (#788)
- 在 metafile 中使用 config 的全名作为‘Name’，以支持 mim 下载 (#798)
- 更新 README 中的图片和视频示例 (#802)
- 在视频插帧测试时使用 `clamp(0, 1)` (#805)
- 在演示和工具中的命令行参数中使用连字符 (#808), 在 python 文件中为必选参数保留下划线 (#822)
- 将 `dataset.pipeline` 作为文档的单列内容 (#813)
- 更新 mmcv-full>=1.3.13 以在 CPU 中支持 DCN (#823)

贡献者

@wangruohui @ckkelvinchan @Yshuo-Li @nijkah @wdmwhh @freepoet @quincylin1

24.5 v0.13.0 (01/03/2022)

Highlights 主要更新

1. 支持 CAIN
2. 支持 EDVR-L
3. 支持在 Windows 系统中运行

New Features

- 为图像和视频添加测试时间 ensemble，并支持 BasicVSR 系列中的 ensemble (#585)
- 支持 AOT-GAN (正在进行中的工作) (#674, #675, #676)
- 支持 CAIN (#683, #691, #709, #713)
- 新增 basic interpolater (#687)
- 新增 BaseVFIDataset and VFIVimeo90KDataset (#695, #697)
- 新增 video interpolation demo (#688, #717)
- 在 RDDBNet 中支持多种 scale (#699)
- 在 demo 中支持 Ref-SR 推理 (#716)
- 在 REDS 数据集上支持 EDVR-L (#719)
- 支持 CPU 训练 (#720)
- 支持在 Windows 中运行 (#732, #738)
- 支持 CPU 中的 DCN (#735)

Bug 修复

- 修复文档中的链接问题 (#703, #704)
- 修复 Dockerfile 中的 `MMCV` 参数 (#708)
- 修复不可执行文件的文件权限 (#718)
- 修复一些与 numpy 相关的弃用警告 (#728)
- 删除 `TestVFIDataset` 中的 `__init__` (#731)
- 修复数据集说明文档中的数据类型 (#739)
- 修复说明文档中的数学符号 (#741)
- 修复 copyright commit hook 中忽略的文件夹 (#754)
- 删除加载中的重复测试 (#756)

改进

- 将 CI 中的 Pillow 版本从 6.2.2 to 更新至 8.4 (#693)
- 在 SRREDSMultipleGTDataset 中增加‘repeat’参数 (#672)
- 弃用对“python setup.py test”的支持 (#701)
- 在训练和测试中添加 `multi-processing` 设置 (#707)
- 添加 OpenMMLab 网站和平台链接 (#710)
- 重构各模型的 README 文件 (#712)
- 使用 `package.version.parse` 替代字符串版本比较 (#723)
- 添加 Ref-SR 演示和视频帧插值演示的文档 (#724)
- 重构 README.md 并增加插帧算法相关内容 (#726)
- 更新 pre-commit hook 中的 isort 版本 (#727)
- 重新设计 Linux 的 CI (#734)
- 更新 install.md (#763)
- 在 README 文件中重新组织 OpenMMLab 项目 (#764)
- 为部署工具添加弃用消息 (#765)

贡献者

@wangruohui @ckkelvinchan @Yshuo-Li @quincylin1 @Juggernaut93 @anse3832 @nijkah

24.6 v0.12.0 (31/12/2021)

主要更新

1. 支持 RealBasicVSR
2. 支持 Real-ESRGAN 预训练模型

新功能

- 支持视频恢复演示中的视频输入和输出 (#622)
- 支持 RealBasicVSR (#632, #633, #647, #680)
- 支持 Real-ESRGAN 预训练模型 (#635)
- 加载图片时支持转化到 Y 通道 (643)
- 训练时支持随机视频压缩 (#646)
- 支持裁剪序列 (#648)
- 支持 pixel_unshuffle (#684)

Bug 修复

- 将 RandomResize 的‘target_size’从列表更改为元组 (#617)
- 修复 preprocess_df2k_ost_dataset.py 中的文件夹创建问题 (#623)
- 在 README 中更改 TDAN 配置路径 (#625)
- 在 Real-ESRNet 配置中将 UnsharpMasking 的‘radius’更改为‘kernel_size’(#626)
- 修复 MATLABLikeResize 中的 Bug (#630)
- 修复‘flow_warp’注释 (#655)
- 修复文档中 Model Zoo 和 Datasets 的错误 (#664)
- 修复‘random_degradations’中的错误 (#673)
- 限制 opencv-python 版本 (#689)

改进

- 将文档翻译成中文 (#576, #577, #578, #579, #581, #582, #584, #587, #588, #589, #590, #591, #592, #593, #594, #595, #596, #641, #647, #656, #665, #666)
- 添加 UNetDiscriminatorWithSpectralNorm (#605)
- 使用 PyTorch sphinx 主题 (#607, #608)
- 在文本文档中更新 mmcv (#609), mmflow (#621), mmfewshot (#634) and mmhuman3d (#649) 的信息
- 将最低 GCC 版本转换为 5.4 (#612)
- 在 SRDataset IMG_EXTENSIONS 中添加 tiff (#614)
- 更新 metafile 和 update_model_index.py (#615)
- 更新 preprocess_df2k_ost_dataset.py (#624)
- 将摘要添加到 README (#628, #636)
- 将 NIQE 与 MATLAB 结果对齐 (#631)
- 添加官方 Markdown lint 钩子 (#639)
- 更改某些特定文件时跳过 CI (#640)
- 更新 docs/conf.py (#644, #651)
- 尝试在 Windows 上创建软链接 (#645)
- 取消之前未完成的运行 (#650)
- 更新 demo.md 和 getting_started.md 中 config 的路径 (#658, #659)
- 使用 mmcv 根模型注册表 (#660)
- 更新 README.md (#654, #663)
- 重构文档结构 (#668)

- 添加脚本以将 REDS 图像裁剪为子图像以加快 IO (#669)
- 将 metafile 中任务名称的第一个字母大写 (#678)
- 更新 FixedCrop 以支持裁剪图像序列 (#682)

24.7 v0.11.0 (03/11/2021)

亮点

1. 支持使用 GLEAN 处理人脸图像的盲超分辨率
2. 支持 Real-ESRGAN 模型 #546

新功能

- 指数移动平均线钩子 #542
- 支持 DF2K_OST 数据 #566

改进

- 增加与 MATLAB 相似的双线性插值算法 #507
- 在训练期间支持随机退化 #504
- 支持 torchserve #568

24.8 v0.10.0 (12/08/2021).

亮点

1. 支持 LIIF-RDN (CVPR’2021)
2. 支持 BasicVSR++ (NTIRE’2021)

新功能

- Video SR datasets 支持加载文件列表 (#423)
- 支持 persistent worker (#426)
- 支持 LIIF-RDN (#428, #440)
- 支持 BasicVSR++ (#451, #467)
- 支持 mim (#455)

Bug 修复

- 修复了 stat.py 中的 bug (#420)
- 修复了 tensor2img 函数中的 astype 错误 (#429)

- 修复了当 pytorch >= 1.7 时由 torch.new_tensor 导致的 device 错误 (#465)
- 修复了.mmedit/apis/train.py 中的 _non_dist_train (#473)
- 修复了多节点分布式测试函数 (#478)

兼容性更新

- 对 pytorch2onnx 重构了 LIIF (#425)

改进

- 更新了部分中文文档 (#415, #416, #418, #421, #424, #431, #442)
- 添加了 pytorch 1.9.0 的 CI (#444)
- 重写了 README.md 的 configs 文件 (#452)
- 避免在单元测试中加载 VGG 网络的预训练权重 (#466)
- 支持在 div2k 数据集预处理时指定 scales (#472)
- 支持 readthedocs 中的所有格式 (#479)
- 使用 mmcv 的 version_info (#480)
- 删除了 restoration_video_demo.py 中不必要的代码 (#484)
- 将 DistEvalIterHook 的优先级修改为‘LOW’(#489)
- 重置资源限制 (#491)
- 在 README_CN.md 中更新了 QQ 的 QR code (#494)
- 添加了 `myst_parser` (#495)
- 添加了 license 信息 (#496)
- 修正了 StyleGAN modules 中的拼写错误 (#427)
- 修正了 docs/demo.md 中的拼写错误 (#453, #454)
- 修复了 tools/data/super-resolution/reds/README.md 中的拼写错误 (#469)

24.9 v0.9.0 (30/06/2021).

主要更新

1. 支持 DIC 和 DIC-GAN (CVPR’2020)
2. 支持 GLEAN Cat 8x (CVPR’2021)
3. 支持 TTSR-GAN (CVPR’2020)
4. 增加超分辨率 colab 使用指南

新功能

- 添加 DIC (#342, #345, #348, #350, #351, #357, #363, #365, #366)
- 增加 SRFolderMultipleGTDataset (#355)
- 增加 GLEAN Cat 8x (#367)
- 增加 SRFolderVideoDataset (#370)
- 增加超分辨率 colab 使用指南 (#380)
- 增加 TTSR-GAN (#372, #381, #383, #398)
- 增加 DIC-GAN (#392, #393, #394)

Bug 修复

- 修复了 restoration_video_inference.py 中的 bug (#379)
- 修复了 LIIF 的配置文件 (#368)
- 修改了 pre-trained EDVR-M 的路径 (#396)
- 修复了 restoration_video_inference 中的 normalization (#406)
- 修复了单元测试中的 [brush_stroke_mask] 错误 (#409)

兼容性更新

- 更改 mmcv 最低兼容版本为 v1.3 (#378)

改进

- 修正了代码中的拼写错误 (#371)
- 添加了 Custom_hooks (#362)
- 重构了 unittest 的目录结构 (#386)
- 添加了 Vid4 数据集的文档和下载链接 (#399)
- 更新了文档中的 model zoo (#400)
- 更新了 metafile (407)

24.10 v0.8.0 (31/05/2021).

主要更新

1. 支持 GLEAN (CVPR’2021)
2. 支持 TTSR (CVPR’2020)
3. 支持 TDAN (CVPR’2020)

新功能

- 添加了 GLEAN (#296, #332)

- 支持 PWD metafile (#298)
- 支持 CropLike in pipeline (#299)
- 添加了 TTSR (#301, #304, #307, #311, #311, #312, #313, #314, #321, #326, #327)
- 添加了 TDAN (#316, #334, #347)
- 添加了 onnx2tensorrt (#317)
- 添加了 tensorrt evaluation (#328)
- 添加了 SRFacicalLandmarkDataset (#329)
- 添加了对 DIC 的 key point 辅助模型 (#336, #341)
- 添加了对视频超分辨率方法的演示 (#275)
- 添加了 SR Folder Ref Dataset (#292)
- 支持对视频超分辨率模型的 FLOPs 计算 (#309)

Bug 修复

- 修复了 find_unused_parameters in PyTorch 1.8 for BasicVSR (#290)
- 修复了 error in publish_model.py for pt>=1.6 (#291)
- 修复了 PSNR when input is uint8 (#294)

改进

- 支持 LoadImageFromFile 的 backend (#293, #303)
- 更新了视频超分数据集的 `metric_average_mode` (#319)
- 添加了 restoration_demo.py 中的错误提示信息 (324)
- 修改了 getting_started.md (#339)
- 更新了 Vimeo90K 的描述 (#349)
- 支持在 GenerateSegmentIndices 中使用 start_index (#338)
- 支持在 GenerateSegmentIndices 使用不同的文件名模板 (#325)
- 支持使用 scale-factor 进行 resize (#295, #310)

24.11 v0.7.0 (30/04/2021).

主要更新

1. 支持 BasicVSR (CVPR’2021)
2. 支持 IconVSR (CVPR’2021)
3. 支持 RDN (CVPR’2018)
4. 添加了 onnx evaluation 工具

新功能

- 添加了 RDN (#233, #260, #280)
- 添加了 MultipleGT 数据集 (#238)
- 添加了 BasicVSR and IconVSR (#245, #252, #253, #254, #264, #274, #258, #252, #264)
- 添加了 onnx evaluation 工具 (#279)

Bug 修复

- 修复了 maxunpool2d 的 onnx 转换 (#243)
- 修正了 `demo.md` 中的 inpainting (#248)
- 修正了 EDSR 的 config 文件 (#251)
- 修正了 README 中的链接 (#256)
- 修复了 restoration_inference 中 key missing 的 bug (#270)
- 修复了 channel_order 在 `loading.py` 的使用 (#271)
- 修复了 inpainting 的 command (#278)
- 修复了 `preprocess_vimeo90k_dataset.py` 中的 args 名称 (#281)

改进

- 支持 `test.py` 中的 `empty_cache` 选项 (#261)
- 更新了 README 中的 projects (#249, #276)
- 支持计算 Y-channel 的 PSNR and SSIM (#250)
- 添加了 zh-CN README (#262)
- 更新了 pytorch2onnx doc (#265)
- 删除了 README 中多余的引文 (#268)
- 更改了 tags 以 comment (#269)
- 在 README 中列出了 `model zoo` (#284, #285, #286)

24.12 v0.6.0 (08/04/2021).

主要更新

1. 支持 Local Implicit Image Function (LIIF)
2. 支持将 DIM 和 GCA 从 Pytorch 导出到 ONNX

新功能

- 添加了 readthedocs 的配置文件以及修复了 docstring (#92)
- 添加了 github action file (#94)
- 支持将 DIM 和 GCA 从 Pytorch 导出到 ONNX (#105)
- 支持 concatenating datasets (#106)
- 支持 `non_dist_train` validation (#110)
- 添加了 matting 的 colab 教程 (#111)
- 支持 niqe metric (#114)
- 对 parrots 支持 PoolDataLoader (#134)
- 支持 collect-env (#137, #143)
- 支持 CI 中的 pt1.6 cpu/gpu (#138)
- 支持 fp16 (139, #144)
- 支持发布到 pypi (#149)
- 添加了 modelzoo 的数据 (#171, #182, #186)
- 添加了数据集的文档 (194)
- 支持扩展 foreground 选项. (#195, #199, #200, #210)
- 支持 nn.MaxUnpool2d (#196)
- 添加了一些 FBA 组件 (#203, #209, #215, #220)
- 支持在数据预处理流水线中使用 random down sampling (#222)
- 支持 SR folder GT Dataset (#223)
- 支持 Local Implicit Image Function (LIIF) (#224, #226, #227, #234, #239)

Bug 修复

- 修复了 train api 中的 `_non_dist_train` (#104)
- 修复了 setup 和 CI (#109)
- 修复了 Normalize 中会导致多余循环的 bug (#121)

- 修复了 `get_hash` in `setup.py` (#124)
- 修复了 `tool/preprocess_reds_dataset.py` (#148)
- 修复了 `getting_started.md` 中的 slurm 训练教程 (#162)
- 修复了 pip 安装的 bug (#173)
- 修复了 config file 中的 bug (#185)
- 修复了数据集中失效的链接 (#236)
- 修复了 model zoo 中失效的链接 (#242)

兼容性更新

- 重构了 data loader 配置文件 (#201)

改进

- 更新了 requirements.txt (#95, #100)
- 更新了 teaser (#96)
- 更新了 README (#93, #97, #98, #152)
- 更新了 model_zoo (#101)
- 修正了一些 typos (#102, #188, #191, #197, #208)
- 采用了 skimage 中的 adjust_gamma 以及减少了依赖 (#112)
- 移除了 `.gitlab-ci.yml` (#113)
- 更新了第一方的代码库的引入 (#115)
- 移除了引用信息和联系方式 (#122)
- 更新了版本信息文件 (#136)
- 更新了下载链接 (#141)
- 更新了 `setup.py` (#150)
- 更新了所支持的 mmcv 的最高版本 (#153, #154)
- 更新了 `Crop` 以处理一系列来自视频的帧 (#164)
- 添加了其他 mm 项目的链接 (#179, #180)
- 添加了 config type (#181)
- 重写了文档 (#184)
- 添加了 config link (#187)
- 更新了文件结构 (#192)
- 更新了配置文档 (#202)

- 更新了 `slurm_train.md` 的脚本 (#204)
- 改进了代码风格 (#206, #207)
- 在 CompositeFg 使用了 `file_client` (#212)
- 使用 `numpy.random` 代替了 `random` (#213)
- 重构了 `loader_cfg` (#214)

24.13 v0.5.0 (09/07/2020).

请注意，作为 MMEdit 的一部分，**MMSR** 已经被合并到此代码库中。我们希望通过对新框架的精心设计和细致实现，MMEditing 能够为您提供更好的体验。

CHAPTER 25

常见问题解答（待更新）

CHAPTER 26

English

CHAPTER 27

简体中文

CHAPTER 28

Indices and tables

- genindex
- modindex
- search

Python 模块索引

m

索引

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W

模块