flowvision

OneFlow

Mar 29, 2023

TUTORIALS

CHAPTER

ONE

GETTING STARTED

1.1 Installation

- To install latest stable release of flowvision:

```
pip install flowvision==0.1.0
```

- For an editable install

```
git clone https://github.com/Oneflow-Inc/vision.git
cd vision
pip install -e .
```

1.2 Usage

1.2.1 Create a model

In flowvision we support two ways to create a model.

- Import the target model from `flowvision.models`, e.g., create `alexnet` from flowvision

```
from flowvision.models.alexnet import alexnet
model = alexnet()

# will download the pretrained model
model = alexnet(pretrained=True)

# customize model to fit different number of classes
model = alexnet(num_classes=100)
```

- Or create model in an easier way by using `ModelCreator`, e.g., create `alexnet` model by `ModelCreator`

```
from flowvision.models import ModelCreator
alexnet = ModelCreator.create_model("alexnet")

# will download the pretrained model
alexnet = ModelCreator.create_model("alexnet", pretrained=True)

# customize model to fit different number of classes
alexnet = ModelCreator.create_model("alexnet", num_classes=100)
```

1.2.2 Tabulate all models with pretrained weights

`ModelCreator.model_table()` returns a tabular results of available models in `flowvision`. To check all of pretrained models, pass in `pretrained=True` in `ModelCreator.model_table()`.

```
from flowvision.models import ModelCreator
all_pretrained_models = ModelCreator.model_table(pretrained=True)
print(all_pretrained_models)
```

You can get the results like:

```
| Supported Models                              | Pretrained   |
| alexnet                                       | true         |
| convmixer_1024_20                             | true         |
| convmixer_1536_20                             | true         |
| convmixer_768_32_relu                         | true         |
| crossformer_base_patch4_group7_224            | true         |
| crossformer_large_patch4_group7_224           | true         |
| crossformer_small_patch4_group7_224           | true         |
| crossformer_tiny_patch4_group7_224            | true         |
|                    ...                        | ...          |
| wide_resnet101_2                              | true         |
| wide_resnet50_2                               | true         |
```

1.2.3 Search for supported model by Wildcard

It is easy to search for model architectures by using Wildcard as below:

```
from flowvision.models import ModelCreator
all_efficientnet_models = ModelCreator.model_table("**efficientnet**")
print(all_efficientnet_models)
```

You can get the results like:

```
| Supported Models    | Pretrained   |
| efficientnet_b0     | true         |
| efficientnet_b1     | true         |
| efficientnet_b2     | true         |
```

(continues on next page)

(continued from previous page)

```
| efficientnet_b3    | true         |
|--------------------|--------------|
| efficientnet_b4    | true         |
|--------------------|--------------|
| efficientnet_b5    | true         |
|--------------------|--------------|
| efficientnet_b6    | true         |
|--------------------|--------------|
| efficientnet_b7    | true         |
```

1.2.4 List all models supported in flowvision

`ModelCreator.model_list` has similar function as `ModelCreator.model_table` but return a list object, which gives the user a more flexible way to check the supported model in flowvision.

- List all models with pretrained weights

```
from flowvision.models import ModelCreator
all_pretrained_models = ModelCreator.model_list(pretrained=True)
print(all_pretrained_models[:5])
```

You can get the results like:

```
['alexnet',
 'convmixer_1024_20',
 'convmixer_1536_20',
 'convmixer_768_32_relu',
 'crossformer_base_patch4_group7_224']
```

- Support wildcard search

```
from flowvision.models import ModelCreator
all_efficientnet_models = ModelCreator.model_list("**efficientnet**")
print(all_efficientnet_models)
```

You can get the results like:

```
['efficientnet_b0',
 'efficientnet_b1',
 'efficientnet_b2',
 'efficientnet_b3',
 'efficientnet_b4',
 'efficientnet_b5',
 'efficientnet_b6',
 'efficientnet_b7']
```

CHAPTER

TWO

FLOWVISION.DATA

Advanced data augmentations

class flowvision.data.**AugMixAugment**(*ops*, *alpha=1.0*, *width=3*, *depth=- 1*, *blended=False*)
AugMix Transform Adapted and improved from impl here: https://github.com/google-research/augmix/blob/master/imagenet.py

From paper AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty

class flowvision.data.**AutoAugment**(*policy*)
Auto Augmentation

From paper AutoAugment: Learning Augmentation Policies from Data

class flowvision.data.**Mixup**(*mixup_alpha=1.0*, *cutmix_alpha=0.0*, *cutmix_minmax=None*, *prob=1.0*, *switch_prob=0.5*, *mode='batch'*, *correct_lam=True*, *label_smoothing=0.1*, *num_classes=1000*)
Mixup/Cutmix that applies different params to each element or whole batch

Parameters

- **mixup_alpha** (*float*) – Mixup alpha value, mixup is active if > 0
- **cutmix_alpha** (*float*) – Cutmix alpha value, cutmix is active if > 0
- **cutmix_minmax** (*List[float]*) – Cutmix min/max image ratio, cutmix is active and uses this vs alpha if not None
- **prob** (*float*) – Probability of applying mixup or cutmix per batch or element
- **switch_prob** (*float*) – Probability of switching to cutmix instead of mixup when both are active
- **mode** (*str*) – How to apply mixup/cutmix params (per 'batch', 'pair' (pair of elements), 'elem' (element)
- **correct_lam** (*bool*) – Apply lambda correction when cutmix bbox clipped by image borders
- **label_smoothing** (*float*) – Apply label smoothing to the mixed target tensor
- **num_classes** (*int*) – Number of classes for target

class flowvision.data.**RandAugment**(*ops*, *num_layers=2*, *choice_weights=None*)
Random Augmentation

From paper RandAugment: Practical automated data augmentation

class flowvision.data.**RandomErasing**(*probability=0.5*, *min_area=0.02*, *max_area=0.3333333333333333*, *min_aspect=0.3*, *max_aspect=None*, *mode='const'*, *min_count=1*, *max_count=None*, *num_splits=0*, *device='cuda'*)

Randomly selects a rectangle region in an image and erases its pixels. 'Random Erasing Data Augmentation' by Zhong et al. See https://arxiv.org/pdf/1708.04896.pdf

This variant of RandomErasing is intended to be applied to either a batch or single image tensor after it has been normalized by dataset mean and std.

Parameters

- **probability** – Probability that the Random Erasing operation will be performed
- **min_area** – Minimum percentage of erased area wrt input image area
- **max_area** – Maximum percentage of erased area wrt input image area
- **min_aspect** – Minimum aspect ratio of erased area
- **mode** – Pixel color mode, one of 'const', 'rand', or 'pixel'
  - 'const' - erase block is constant color of 0 for all channels
  - 'rand' - erase block is same per-channel random (normal) color
  - 'pixel' - erase block is per-pixel random (normal) color
- **max_count** – Maximum number of erasing blocks per image, area per box is scaled by count. per-image count is randomly chosen between 1 and this value

flowvision.data.**augment_and_mix_transform**(*config_str*, *hparams*)

Create AugMix OneFlow transform

Parameters

- **config_str** – String defining configuration of random augmentation. Consists of multiple sections separated by dashes ('-'). The first section defines the specific variant of rand augment (currently only 'rand'). The remaining sections, not order sepecific determine
  - 'm' - integer magnitude (severity) of augmentation mix (default: 3)
  - 'w' - integer width of augmentation chain (default: 3)
  - 'd' - integer depth of augmentation chain (-1 is random [1, 3], default: -1)
  - 'b' - integer (bool), blend each branch of chain into end result without a final blend, less CPU (default: 0)
  - 'mstd' - float std deviation of magnitude noise applied (default: 0)

  Example: 'augmix-m5-w4-d2' results in AugMix with severity 5, chain width 4, chain depth 2
- **hparams** – Other hparams (kwargs) for the Augmentation transforms

Returns An OneFlow compatible Transform

flowvision.data.**auto_augment_transform**(*config_str*, *hparams*)

Create a AutoAugment transform

Parameters

- **config_str** – String defining configuration of auto augmentation. Consists of multiple sections separated by dashes ('-'). The first section defines the AutoAugment policy (one of 'v0', 'v0r', 'original', 'originalr'). The remaining sections, not order sepecific determine

– 'mstd' - float std deviation of magnitude noise applied

Example: 'original-mstd0.5' results in AutoAugment with original policy, magnitude_std 0.5

- **hparams** – Other hparams (kwargs) for the AutoAugmentation scheme

Returns An OneFlow compatible Transform

flowvision.data.**cutmix_bbox_and_lam**(*img_shape*, *lam*, *ratio_minmax=None*, *correct_lam=True*, *count=None*)

Generate bbox and apply lambda correction.

flowvision.data.**mixup_target**(*target*, *num_classes*, *lam=1.0*, *smoothing=0.0*, *device='cuda'*)

Mixup the targets with label-smoothing

flowvision.data.**rand_augment_transform**(*config_str*, *hparams*)

Create a RandAugment transform

Parameters

- **config_str** – String defining configuration of random augmentation. Consists of multiple sections separated by dashes ('-'). The first section defines the specific variant of rand augment (currently only 'rand'). The remaining sections, not order sepecific determine

  – 'm' - integer magnitude of rand augment

  – 'n' - integer num layers (number of transform ops selected per image)

  – 'w' - integer probabiliy weight index (index of a set of weights to influence choice of op)

  – 'mstd' - float std deviation of magnitude noise applied, or uniform sampling if infinity (or > 100)

  – 'mmax' - set upper bound for magnitude to something other than default of _LEVEL_DENOM (10)

  – 'inc' - integer (bool), use augmentations that increase in severity with magnitude (default: 0)

  Example: 'rand-m9-n3-mstd0.5' results in RandAugment with magnitude 9, num_layers 3, magnitude_std 0.5 'rand-mstd1-w0' results in magnitude_std 1.0, weights 0, default magnitude of 10 and num_layers 2

- **hparams** – Other hparams (kwargs) for the RandAugmentation scheme

Returns An OneFlow compatible Transform

flowvision.data.**rand_bbox**(*img_shape*, *lam*, *margin=0.0*, *count=None*)

Standard CutMix bounding-box Generates a random square bbox based on lambda value. This impl includes support for enforcing a border margin as percent of bbox dimensions.

Parameters

- **img_shape** (*tuple*) – Image shape as tuple
- **lam** (*float*) – Cutmix lambda value
- **margin** (*float*) – Percentage of bbox dimension to enforce as margin (reduce amount of box outside image)
- **count** (*int*) – Number of bbox to generate

flowvision.data.**rand_bbox_minmax**(*img_shape*, *minmax*, *count=None*)

Min-Max CutMix bounding-box Inspired by Darknet cutmix impl, generates a random rectangular bbox based

on min/max percent values applied to each dimension of the input image. Typical defaults for minmax are usually in the .2-.3 for min and .8-.9 range for max.

Parameters

- **img_shape** (*tuple*) – Image shape as tuple
- **minmax** (*tuple or list*) – Min and max bbox ratios (as percent of image size)
- **count** (*int*) – Number of bbox to generate

CHAPTER

THREE

FLOWVISION.DATASETS

3.1 Dataset Utils for Visual Tasks

class flowvision.datasets.**CIFAR10**(*root: str*, *train: bool = True*, *transform: Optional[Callable] = None*, *target_transform: Optional[Callable] = None*, *download: bool = False*, *source_url: Optional[str] = None*)

CIFAR10 Dataset.

Parameters

- **root** (*string*) – Root directory of dataset where directory `cifar-10-batches-py` exists or will be saved to if download is set to True.
- **train** (*bool, optional*) – If True, creates dataset from training set, otherwise creates from test set.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.RandomCrop`
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.
- **download** (*bool, optional*) – If true, downloads the dataset from the internet and puts it in root directory. If dataset is already downloaded, it is not downloaded again.

class flowvision.datasets.**CIFAR100**(*root: str*, *train: bool = True*, *transform: Optional[Callable] = None*, *target_transform: Optional[Callable] = None*, *download: bool = False*, *source_url: Optional[str] = None*)

CIFAR100 Dataset.

This is a subclass of the *CIFAR10* Dataset.

class flowvision.datasets.**CocoCaptions**(*root: str*, *annFile: str*, *transform: Optional[Callable] = None*, *target_transform: Optional[Callable] = None*, *transforms: Optional[Callable] = None*)

MS Coco Captions Dataset.

Parameters

- **root** (*string*) – Root directory where images are downloaded to.
- **annFile** (*string*) – Path to json annotation file.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.ToTensor`

- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.
- **transforms** (*callable, optional*) – A function/transform that takes input sample and its target as entry and returns a transformed version.

Example

```
import flowvision.datasets as dset
import flowvision.transforms as transforms
cap = dset.CocoCaptions(root = 'dir where images are',
                        annFile = 'json annotation file',
                        transform=transforms.ToTensor())
print('Number of samples: ', len(cap))
img, target = cap[3] # load 4th sample
print("Image Size: ", img.size())
print(target)
```

Output:

```
Number of samples: 82783
Image Size: (3L, 427L, 640L)
[u'A plane emitting smoke stream flying over a mountain.',
u'A plane darts across a bright blue sky behind a mountain covered in snow',
u'A plane leaves a contrail above the snowy mountain top.',
u'A mountain that has a plane flying overheard in the distance.',
u'A mountain view with a plume of smoke in the background']
```

class flowvision.datasets.**CocoDetection**(*root: str, annFile: str, transform: Optional[Callable] = None, target_transform: Optional[Callable] = None, transforms: Optional[Callable] = None*)

MS Coco Detection Dataset.

Parameters

- **root** (*string*) – Root directory where images are downloaded to.
- **annFile** (*string*) – Path to json annotation file.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.ToTensor`
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.
- **transforms** (*callable, optional*) – A function/transform that takes input sample and its target as entry and returns a transformed version.

class flowvision.datasets.**DatasetFolder**(*root: str, loader: Callable[[str], Any], extensions: Optional[Tuple[str, ...]] = None, transform: Optional[Callable] = None, target_transform: Optional[Callable] = None, is_valid_file: Optional[Callable[[str], bool]] = None*)

A generic data loader. This default directory structure can be customized by overriding the *find_classes()* method.

Parameters

- **root** (*string*) – Root directory path.

- **loader** (*callable*) – A function to load a sample given its path.
- **extensions** (*tuple[string]*) – A list of allowed extensions. both extensions and is_valid_file should not be passed.
- **transform** (*callable, optional*) – A function/transform that takes in a sample and returns a transformed version. E.g, `transforms.RandomCrop` for images.
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.
- **is_valid_file** – A function that takes path of a file and check if the file is a valid file (used to check of corrupt files) both extensions and is_valid_file should not be passed.

find_classes (*directory: str*) → Tuple[List[str], Dict[str, int]]

Find the class folders in a dataset structured as follows:

```
directory/
├── class_x
│   ├── xxx.ext
│   ├── xxy.ext
│   └── ...
│       └── xxz.ext
└── class_y
    ├── 123.ext
    ├── nsdf3.ext
    └── ...
    └── asd932_.ext
```

This method can be overridden to only consider a subset of classes, or to adapt to a different dataset directory structure.

Parameters **directory** (*str*) – Root directory path, corresponding to `self.root`

Raises **FileNotFoundError** – If `dir` has no class folders.

Returns List of all classes and dictionary mapping each class to an index.

Return type (Tuple[List[str], Dict[str, int]])

static make_dataset (*directory: str*, *class_to_idx: Dict[str, int]*, *extensions: Optional[Tuple[str, ...]] = None*, *is_valid_file: Optional[Callable[[str], bool]] = None*) → List[Tuple[str, int]]

Generates a list of samples of a form (path_to_sample, class). This can be overridden to e.g. read files from a compressed zip file instead of from the disk.

Parameters

- **directory** (*str*) – root dataset directory, corresponding to `self.root`.
- **class_to_idx** (*Dict[str, int]*) – Dictionary mapping class name to class index.
- **extensions** (*optional*) – A list of allowed extensions. Either extensions or is_valid_file should be passed. Defaults to None.
- **is_valid_file** (*optional*) – A function that takes path of a file and checks if the file is a valid file (used to check of corrupt files) both extensions and is_valid_file should not be passed. Defaults to None.

Raises

- **ValueError** – In case `class_to_idx` is empty.

- **ValueError** – In case `extensions` and `is_valid_file` are None or both are not None.
- **FileNotFoundError** – In case no valid file was found for any class.

Returns samples of a form (path_to_sample, class)

Return type List[Tuple[str, int]]

class flowvision.datasets.**FashionMNIST**(*root: str*, *train: bool = True*, *transform: Optional[Callable] = None*, *target_transform: Optional[Callable] = None*, *download: bool = False*, *source_url: Optional[str] = None*)

Fashion-MNIST Dataset.

Parameters

- **root** (*string*) – Root directory of dataset where `FashionMNIST/processed/training.pt` and `FashionMNIST/processed/test.pt` exist.
- **train** (*bool, optional*) – If True, creates dataset from `training.pt`, otherwise from `test.pt`.
- **download** (*bool, optional*) – If true, downloads the dataset from the internet and puts it in root directory. If dataset is already downloaded, it is not downloaded again.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.RandomCrop`
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.

class flowvision.datasets.**ImageFolder**(*root: str, transform: Optional[Callable] = None, target_transform: Optional[Callable] = None, loader: Callable[[str], Any] = <function default_loader>, is_valid_file: Optional[Callable[[str], bool]] = None*)

A generic data loader where the images are arranged in this way by default:

```
root/dog/xxx.png
root/dog/xxy.png
root/dog/[...]/xxz.png
root/cat/123.png
root/cat/nsdf3.png
root/cat/[...]/asd932_.png
```

This class inherits from `DatasetFolder` so the same methods can be overridden to customize the dataset.

Parameters

- **root** (*string*) – Root directory path.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.RandomCrop`
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.
- **loader** (*callable, optional*) – A function to load an image given its path.
- **is_valid_file** – A function that takes path of an Image file and check if the file is a valid file (used to check of corrupt files)

class flowvision.datasets.**ImageNet**(*root: str*, *split: str = 'train'*, *download: Optional[str] = None*, ***kwargs: Any*)

ImageNet 2012 Classification Dataset.

Parameters

- **root** (*string*) – Root directory of the ImageNet Dataset.
- **split** (*string, optional*) – The dataset split, supports `train`, or `val`.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.RandomCrop`
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.
- **loader** – A function to load an image given its path.

class flowvision.datasets.**MNIST**(*root: str*, *train: bool = True*, *transform: Optional[Callable] = None*, *target_transform: Optional[Callable] = None*, *download: bool = False*, *source_url: Optional[str] = None*)

MNIST Dataset.

Parameters

- **root** (*string*) – Root directory of dataset where `MNIST/processed/training.pt` and `MNIST/processed/test.pt` exist.
- **train** (*bool, optional*) – If True, creates dataset from `training.pt`, otherwise from `test.pt`.
- **download** (*bool, optional*) – If true, downloads the dataset from the internet and puts it in root directory. If dataset is already downloaded, it is not downloaded again.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.RandomCrop`
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.

download() → None

Download the MNIST data if it doesn't exist already.

class flowvision.datasets.**VOCDetection**(*root: str*, *year: str = '2012'*, *image_set: str = 'train'*, *download: bool = False*, *transform: Optional[Callable] = None*, *target_transform: Optional[Callable] = None*, *transforms: Optional[Callable] = None*)

Pascal VOC Detection Dataset.

Parameters

- **root** (*string*) – Root directory of the VOC Dataset.
- **year** (*string, optional*) – The dataset year, supports years `"2007"` to `"2012"`.
- **image_set** (*string, optional*) – Select the image_set to use, `"train"`, `"trainval"` or `"val"`. If `year=="2007"`, can also be `"test"`.
- **download** (*bool, optional*) – If true, downloads the dataset from the internet and puts it in root directory. If dataset is already downloaded, it is not downloaded again. (default: alphabetic indexing of VOC's 20 classes).
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, `transforms.RandomCrop`

- **target_transform** (*callable, required*) – A function/transform that takes in the target and transforms it.
- **transforms** (*callable, optional*) – A function/transform that takes input sample and its target as entry and returns a transformed version.

class flowvision.datasets.**VOCSegmentation**(*root: str*, *year: str = '2012'*, *image_set: str = 'train'*, *download: bool = False*, *transform: Optional[Callable] = None*, *target_transform: Optional[Callable] = None*, *transforms: Optional[Callable] = None*)

Pascal VOC Segmentation Dataset.

Parameters

- **root** (*string*) – Root directory of the VOC Dataset.
- **year** (*string, optional*) – The dataset year, supports years "2007" to "2012".
- **image_set** (*string, optional*) – Select the image_set to use, "train", "trainval" or "val". If year=="2007", can also be "test".
- **download** (*bool, optional*) – If true, downloads the dataset from the internet and puts it in root directory. If dataset is already downloaded, it is not downloaded again.
- **transform** (*callable, optional*) – A function/transform that takes in an PIL image and returns a transformed version. E.g, transforms.RandomCrop
- **target_transform** (*callable, optional*) – A function/transform that takes in the target and transforms it.
- **transforms** (*callable, optional*) – A function/transform that takes input sample and its target as entry and returns a transformed version.

CHAPTER

FOUR

FLOWVISION.LAYERS

Plug and Play Modules or Functions that are specific for Computer Vision Tasks

4.1 flowvision.layers.blocks module

class flowvision.layers.blocks.**FeaturePyramidNetwork**(*in_channels_list: List[int]*, *out_channels: int*, *extra_blocks: Optional[flowvision.layers.blocks.feature_pyramid_networ* = *None*)

Module that adds a FPN from on top of a set of feature maps. This is based on "Feature Pyramid Network for Object Detection".

The feature maps are currently supposed to be increasing depth order.

The input to the model is expected to be an OrderedDict[Tensor], containing the feature maps on top of which the FPN will be added.

Parameters

- **in_channels_list** (*list[int]*) – number of channels for each feature map that is passed to the module
- **out_channels** (*int*) – number of channels of the FPN representation
- **extra_blocks** (*ExtraFPNBlock or None*) – if provided, extra operations will be performed. It is expected to take the fpn features, the original features and the names of the original features as input, and returns a new list of feature maps and their corresponding names

forward(*x: Dict[str, oneflow.Tensor]*) → Dict[str, oneflow.Tensor]

Computes the FPN for a set of feature maps.

Parameters **x** (*OrderedDict[Tensor]*) – feature maps for each feature level.

Returns

feature maps after FPN layers. They are ordered from highest resolution first.

Return type results (OrderedDict[Tensor])

get_result_from_inner_blocks(*x: oneflow.Tensor*, *idx: int*) → oneflow.Tensor

This is equivalent to self.inner_blocks[idx](x)

get_result_from_layer_blocks(*x: oneflow.Tensor*, *idx: int*) → oneflow.Tensor

This is equivalent to self.layer_blocks[idx](x)

class flowvision.layers.blocks.**MultiScaleRoIAlign**(*featmap_names: List[str]*, *output_size: Union[int, Tuple[int], List[int]]*, *sampling_ratio: int*, ***, *canonical_scale: int = 224*, *canonical_level: int = 4*)

Multi-scale RoIAlign pooling, which is useful for detection with or without FPN.

It infers the scale of the pooling via the heuristics specified in eq. 1 of the Feature Pyramid Network paper. They keyword-only parameters `canonical_scale` and `canonical_level` correspond respectively to `224` and `k0=4` in eq. 1, and have the following meaning: `canonical_level` is the target level of the pyramid from which to pool a region of interest with `w x h = canonical_scale x canonical_scale`.

Parameters

- **featmap_names** (*List[str]*) – the names of the feature maps that will be used for the pooling.
- **output_size** (*List[Tuple[int, int]] or List[int]*) – output size for the pooled region
- **sampling_ratio** (*int*) – sampling ratio for ROIAlign
- **canonical_scale** (*int, optional*) – canonical_scale for LevelMapper
- **canonical_level** (*int, optional*) – canonical_level for LevelMapper

forward(*x: Dict[str, oneflow.Tensor]*, *boxes: List[oneflow.Tensor]*, *image_shapes: List[Tuple[int, int]]*) → oneflow.Tensor

Parameters

- **x** (*OrderedDict[Tensor]*) – feature maps for each level. They are assumed to have all the same number of channels, but they can have different sizes.
- **boxes** (*List[Tensor[N, 4]]*) – boxes to be used to perform the pooling operation, in (x1, y1, x2, y2) format and in the image reference size, not the feature map reference. The coordinate must satisfy `0 <= x1 < x2` and `0 <= y1 < y2`.
- **image_shapes** (*List[Tuple[height, width]]*) – the sizes of each image before they have been fed to a CNN to obtain feature maps. This allows us to infer the scale factor for each one of the levels to be pooled.

Returns result (Tensor)

flowvision.layers.blocks.**batched_nms**(*boxes: oneflow.Tensor*, *scores: oneflow.Tensor*, *idxs: oneflow.Tensor*, *iou_threshold: float*) → oneflow.Tensor

Performs non-maximum suppression in a batched fashion.

Each index value correspond to a category, and NMS will not be applied between elements of different categories.

Parameters

- **boxes** (*Tensor[N, 4]*) – boxes where NMS will be performed. They are expected to be in `(x1, y1, x2, y2)` format with `0 <= x1 < x2` and `0 <= y1 < y2`.
- **scores** (*Tensor[N]*) – scores for each one of the boxes
- **idxs** (*Tensor[N]*) – indices of the categories for each one of the boxes.
- **iou_threshold** (*float*) – discards all overlapping boxes with IoU > iou_threshold

Returns int64 tensor with the indices of the elements that have been kept by NMS, sorted in decreasing order of scores

Return type Tensor

flowvision.layers.blocks.**box_iou**(*boxes1: oneflow.Tensor*, *boxes2: oneflow.Tensor*) → oneflow.Tensor

Return intersection-over-union (Jaccard index) between two sets of boxes.

Both sets of boxes are expected to be in `(x1, y1, x2, y2)` format with `0 <= x1 < x2` and `0 <= y1 < y2`.

Parameters

- **boxes1** (*Tensor[N, 4]*) – first set of boxes
- **boxes2** (*Tensor[N, 4]*) – second set of boxes

Returns the NxM matrix containing the pairwise IoU values for every element in boxes 1 and boxes2

Return type Tensor[N, M]

flowvision.layers.blocks.**nms**(*boxes: oneflow.Tensor*, *scores: oneflow.Tensor*, *iou_threshold: float*) → oneflow.Tensor

Performs non-maximum suppression (NMS) on the boxes according to their intersection-over-union (IoU).

NMS iteratively removes lower scoring boxes which have an IoU greater than iou_threshold with another (higher scoring) box.

Parameters

- **boxes** (*Tensor[N, 4]*) – boxes to perform NMS on. They are expected to be in `(x1, y1, x2, y2)` format with `0 <= x1 < x2` and `0 <= y1 < y2`.
- **scores** (*Tensor[N]*) – scores for each one of the boxes
- **iou_threshold** (*float*) – discards all overlapping boxes with IoU > iou_threshold

Returns int64 tensor with the indices of the elements that have been kept by NMS, sorted in decreasing order of scores

Return type Tensor

4.2 flowvision.layers.attention module

class flowvision.layers.attention.**SEModule**(*channels: int*, *reduction: int = 16*, *rd_channels: Optional[int] = None*, *act_layer: Optional[oneflow.nn.modules.activation.ReLU] = <class 'oneflow.nn.modules.activation.ReLU'>*, *gate_layer: Optional[oneflow.nn.modules.activation.Sigmoid] = <class 'oneflow.nn.modules.activation.Sigmoid'>*, *mlp_bias=True*)

"Squeeze-and-Excitation" block adaptively recalibrates channel-wise feature responses. This is based on "Squeeze-and-Excitation Networks". This unit is designed to improve the representational capacity of a network by enabling it to perform dynamic channel-wise feature recalibration.

Parameters

- **channels** (*int*) – The input channel size

- **reduction** (*int*) – Ratio that allows us to vary the capacity and computational cost of the SE Module. Default: 16
- **rd_channels** (*int or None*) – Number of reduced channels. If none, uses reduction to calculate
- **act_layer** (*Optional[ReLU]*) – An activation layer used after the first FC layer. Default: flow.nn.ReLU
- **gate_layer** (*Optional[Sigmoid]*) – An activation layer used after the second FC layer. Default: flow.nn.Sigmoid
- **mlp_bias** (*bool*) – If True, add learnable bias to the linear layers. Default: True

CHAPTER

FIVE

FLOWVISION.LOSS

Loss Functions that are specific for Computer Vision Tasks

5.1 Losses

class flowvision.loss.**LabelSmoothingCrossEntropy**(*smoothing=0.1*)
NLL Loss with label smoothing

class flowvision.loss.**SoftTargetCrossEntropy**
Soft target CrossEntropy loss

CHAPTER

SIX

FLOWVISION.MODELS

Pretrain Models for Visual Tasks

6.1 Classification

The models subpackage contains definitions for the following model architectures for image classification:

- AlexNet
- SqueezeNet
- VGG
- GoogLeNet
- InceptionV3
- ResNet
- ResNeXt
- ResNeSt
- SENet
- DenseNet
- ShuffleNetV2
- MobileNetV2
- MobileNetV3
- MNASNet
- GhostNet
- Res2Net
- EfficientNet
- RegNet
- ReXNet
- ViT
- DeiT
- PVT
- Swin-Transformer

- CSwin-Transformer
- CrossFormer
- PoolFormer
- Mlp_Mixer
- ResMLP
- gMLP
- ConvMixer
- ConvNeXt
- LeViT
- RegionViT
- VAN
- MobileViT
- DeiT-III
- CaiT
- DLA
- GENet
- HRNet
- FAN

6.1.1 Alexnet

`flowvision.models.`**`alexnet`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.alexnet.AlexNet

Constructs the AlexNet model.

Note: AlexNet model architecture from the One weird trick... paper. The required minimum input size of this model is 63x63.

Parameters

- **`pretrained`** (*`bool`*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **`progress`** (*`bool`*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> alexnet = flowvision.models.alexnet(pretrained=False, progress=True)
```

6.1.2 SqueezeNet

flowvision.models.**squeezenet1_0**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.squeezenet.SqueezeNet

Constructs the SqueezeNet model.

Note: SqueezeNet model from the SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> squeezenet1_0 = flowvision.models.squeezenet1_0(pretrained=False,␣
↪progress=True)
```

flowvision.models.**squeezenet1_1**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.squeezenet.SqueezeNet

Constructs the SqueezeNet 1.1 model.

Note: SqueezeNet 1.1 model from the SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size paper. Note that SqueezeNet 1.1 has 2.4x less computation and slightly fewer parameters than SqueezeNet 1.0, without sacrificing accuracy.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> squeezenet1_1 = flowvision.models.squeezenet1_1(pretrained=False,␣
↪progress=True)
```

6.1.3 VGG

flowvision.models.**vgg11**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-11 model (configuration "A").

Note: VGG 11-layer model (configuration "A") from "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg11 = flowvision.models.vgg11(pretrained=False, progress=True)
```

flowvision.models.**vgg11_bn**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-11 model with batch normalization (configuration "A").

Note: VGG 11-layer model (configuration "A") with batch normalization "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg11_bn = flowvision.models.vgg11_bn(pretrained=False, progress=True)
```

flowvision.models.**vgg13**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-13 model (configuration "B").

Note: VGG 13-layer model (configuration "B") from "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg13 = flowvision.models.vgg13(pretrained=False, progress=True)
```

flowvision.models.**vgg13_bn**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-13 model with batch normalization (configuration "B").

Note: VGG 13-layer model (configuration "B") with batch normalization from "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg13_bn = flowvision.models.vgg13_bn(pretrained=False, progress=True)
```

flowvision.models.**vgg16**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-16 model (configuration "D").

Note: VGG 16-layer model (configuration "D") from "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg16 = flowvision.models.vgg16(pretrained=False, progress=True)
```

flowvision.models.**vgg16_bn**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-16 model (configuration "D") with batch normalization.

Note: VGG 16-layer model (configuration "D") with batch normalization from "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg16_bn = flowvision.models.vgg16_bn(pretrained=False, progress=True)
```

flowvision.models.**vgg19**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-19 model (configuration "E").

Note: VGG 19-layer model (configuration "E") from "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg19 = flowvision.models.vgg19(pretrained=False, progress=True)
```

flowvision.models.**vgg19_bn**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.vgg.VGG

Constructs the VGG-19 model (configuration "E") with batch normalization.

Note: VGG 19-layer model (configuration "E") with batch normalization from "Very Deep Convolutional Networks For Large-Scale Image Recognition". The required minimum input size of the model is 32x32.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vgg19_bn = flowvision.models.vgg19_bn(pretrained=False, progress=True)
```

6.1.4 GoogLeNet

flowvision.models.**googlenet**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.googlenet.GoogLeNet

Constructs the GoogLeNet (Inception v1) model.

Note: GoogLeNet (Inception v1) model from the Going Deeper with Convolutions paper. The required minimum input size of the model is 15x15.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`
- **aux_logits** (*bool*) – If True, adds two auxiliary branches that can improve training. Default: `False` when pretrained is True otherwise `True`
- **transform_input** (*bool*) – If True, preprocesses the input according to the method with which it was trained on ImageNet. Default: `False`

For example:

```
>>> import flowvision
>>> googlenet = flowvision.models.googlenet(pretrained=False, progress=True)
```

6.1.5 InceptionV3

flowvision.models.**inception_v3**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*)

Constructs Inception v3 model.

Note: Inception v3 model from the Rethinking the Inception Architecture for Computer Vision paper. In contrast to the other models the inception_v3 expects tensors with a size of N x 3 x 299 x 299, so ensure your images are sized accordingly.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`
- **aux_logits** (*bool*) – If True, add an auxiliary branch that can improve training. Default: `True`

- **transform_input** (*bool*) – If True, preprocesses the input according to the method with which it was trained on ImageNet. Default: `False`

For example:

```
>>> import flowvision
>>> inception_v3 = flowvision.models.inception_v3(pretrained=False, progress=True)
```

6.1.6 ResNet

`flowvision.models.`**`resnet101`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the ResNet-101 model.

Note: Deep Residual Learning for Image Recognition.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnet101 = flowvision.models.resnet101(pretrained=False, progress=True)
```

`flowvision.models.`**`resnet152`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the ResNet-152 model.

Note: Deep Residual Learning for Image Recognition.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnet152 = flowvision.models.resnet152(pretrained=False, progress=True)
```

`flowvision.models.`**`resnet18`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the ResNet-18 model.

Note: Deep Residual Learning for Image Recognition.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnet18 = flowvision.models.resnet18(pretrained=False, progress=True)
```

flowvision.models.**resnet34**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the ResNet-34 model.

Note: Deep Residual Learning for Image Recognition.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnet34 = flowvision.models.resnet34(pretrained=False, progress=True)
```

flowvision.models.**resnet50**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the ResNet-50 model.

Note: Deep Residual Learning for Image Recognition.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnet50 = flowvision.models.resnet50(pretrained=False, progress=True)
```

flowvision.models.**resnext101_32x8d**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the ResNeXt-101 32x8d model.

Note: Aggregated Residual Transformation for Deep Neural Networks.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnext101_32x8d = flowvision.models.resnext101_32x8d(pretrained=False,␣
↪progress=True)
```

flowvision.models.**resnext50_32x4d**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the ResNeXt-50 32x4d model.

Note: Aggregated Residual Transformation for Deep Neural Networks.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnext50_32x4d = flowvision.models.resnext50_32x4d(pretrained=False,␣
↪progress=True)
```

flowvision.models.**wide_resnet101_2**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the Wide ResNet-101-2 model.

Note: Wide Residual Networks. The model is the same as ResNet except for the bottleneck number of channels which is twice larger in every block. The number of channels in outer 1x1 convolutions is the same, e.g.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> wide_resnet101_2 = flowvision.models.wide_resnet101_2(pretrained=False,␣
↪progress=True)
```

flowvision.models.**wide_resnet50_2**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.resnet.ResNet

Constructs the Wide ResNet-50-2 model.

Note: Wide Residual Networks. The model is the same as ResNet except for the bottleneck number of channels which is twice larger in every block. The number of channels in outer 1x1 convolutions is the same, e.g. last block in ResNet-50 has 2048-512-2048 channels, and in Wide ResNet-50-2 has 2048-1024-2048.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> wide_resnet50_2 = flowvision.models.wide_resnet50_2(pretrained=False,␣
↪progress=True)
```

6.1.7 ResNeSt

flowvision.models.**resnest101**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResNeSt-101 model trained on ImageNet2012.

Note: ResNeSt-101 model from *"ResNeSt: Split-Attention Networks" <https://arxiv.org/abs/2004.08955> _*. The required input size of the model is 256x256.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnest101 = flowvision.models.resnest101(pretrained=False, progress=True)
```

flowvision.models.**resnest200**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResNeSt-200 model trained on ImageNet2012.

Note: ResNeSt-200 model from *"ResNeSt: Split-Attention Networks" <https://arxiv.org/abs/2004.08955> _*. The required input size of the model is 320x320.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnest200 = flowvision.models.resnest200(pretrained=False, progress=True)
```

`flowvision.models.`**`resnest269`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResNeSt-269 model trained on ImageNet2012.

Note: ResNeSt-269 model from *"ResNeSt: Split-Attention Networks" <https://arxiv.org/abs/2004.08955> _*. The required input size of the model is 416x416.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnest269 = flowvision.models.resnest269(pretrained=False, progress=True)
```

`flowvision.models.`**`resnest50`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResNeSt-50 model trained on ImageNet2012.

Note: ResNeSt-50 model from *"ResNeSt: Split-Attention Networks" <https://arxiv.org/abs/2004.08955> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resnest50 = flowvision.models.resnest50(pretrained=False, progress=True)
```

6.1.8 DenseNet

flowvision.models.**densenet121**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.densenet.DenseNet

Constructs the DenseNet-121 model.

Note: DenseNet-121 model architecture from the Densely Connected Convolutional Networks paper. The required minimum input size of the model is 29x29.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> densenet121 = flowvision.models.densenet121(pretrained=False, progress=True)
```

flowvision.models.**densenet161**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.densenet.DenseNet

Constructs the DenseNet-161 model.

Note: DenseNet-161 model architecture from the Densely Connected Convolutional Networks paper. The required minimum input size of the model is 29x29.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> densenet161 = flowvision.models.densenet161(pretrained=False, progress=True)
```

flowvision.models.**densenet169**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.densenet.DenseNet

Constructs the DenseNet-169 model.

Note: DenseNet-169 model architecture from the Densely Connected Convolutional Networks paper. The required minimum input size of the model is 29x29.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> densenet169 = flowvision.models.densenet169(pretrained=False, progress=True)
```

flowvision.models.**densenet201**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.densenet.DenseNet

Constructs the DenseNet-201 model.

Note: DenseNet-201 model architecture from the Densely Connected Convolutional Networks paper. The required minimum input size of the model is 29x29.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> densenet201 = flowvision.models.densenet201(pretrained=False, progress=True)
```

6.1.9 ShuffleNetV2

flowvision.models.**shufflenet_v2_x0_5**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.shufflenet_v2.ShuffleNetV2

Constructs the ShuffleNetV2(0.5x) model.

Note: ShuffleNetV2 with 0.5x output channels model architecture from the ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> shufflenet_v2_x0_5 = flowvision.models.shufflenet_v2_x0_5(pretrained=False,␣
↪progress=True)
```

flowvision.models.**shufflenet_v2_x1_0**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.shufflenet_v2.ShuffleNetV2

Constructs the ShuffleNetV2(1.0x) model.

Note: ShuffleNetV2 with 1.0x output channels model architecture from the ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> shufflenet_v2_x1_0 = flowvision.models.shufflenet_v2_x1_0(pretrained=False,␣
↪progress=True)
```

flowvision.models.**shufflenet_v2_x1_5**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.shufflenet_v2.ShuffleNetV2

Constructs the ShuffleNetV2(1.5x) model.

Note: ShuffleNetV2 with 1.5x output channels model architecture from the ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> shufflenet_v2_x1_5 = flowvision.models.shufflenet_v2_x1_5(pretrained=False,␣
↪progress=True)
```

flowvision.models.**shufflenet_v2_x2_0**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.shufflenet_v2.ShuffleNetV2

Constructs the ShuffleNetV2(2.0x) model.

Note: ShuffleNetV2 with 2.0x output channels model architecture from the ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> shufflenet_v2_x2_0 = flowvision.models.shufflenet_v2_x2_0(pretrained=False,␣
↪progress=True)
```

6.1.10 MobileNetV2

flowvision.models.**mobilenet_v2**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.mobilenet_v2.MobileNetV2

Constructs the MobileNetV2 model.

Note: MobileNetV2 model architecture from the MobileNetV2: Inverted Residuals and Linear Bottlenecks paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mobilenet_v2 = flowvision.models.mobilenet_v2(pretrained=False, progress=True)
```

6.1.11 MobileNetV3

flowvision.models.**mobilenet_v3_large**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.mobilenet_v3.MobileNetV3

Constructs the MobileNetV3-Large model.

Note: MobileNetV3-Large model architecture from the Searching for MobileNetV3 paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mobilenet_v3_large = flowvision.models.mobilenet_v3_large(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mobilenet_v3_small**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.mobilenet_v3.MobileNetV3

Constructs the MobileNetV3-Small model.

Note: MobileNetV3-Small model architecture from the Searching for MobileNetV3 paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mobilenet_v3_small = flowvision.models.mobilenet_v3_small(pretrained=False,␣
↪progress=True)
```

6.1.12 MNASNet

flowvision.models.**mnasnet0_5**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the MNASNet model with depth multiplier of 0.5.

Note: MNASNet model with depth multiplier of 0.5 from the MnasNet: Platform-Aware Neural Architecture Search for Mobile paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mnasnet0_5 = flowvision.models.mnasnet0_5(pretrained=False, progress=True)
```

flowvision.models.**mnasnet0_75**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the MNASNet model with depth multiplier of 0.75.

Note: MNASNet model with depth multiplier of 0.75 from the MnasNet: Platform-Aware Neural Architecture Search for Mobile paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mnasnet0_75 = flowvision.models.mnasnet0_75(pretrained=False, progress=True)
```

flowvision.models.**mnasnet1_0**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the MNASNet model with depth multiplier of 1.0.

Note: MNASNet model with depth multiplier of 1.0 from the MnasNet: Platform-Aware Neural Architecture Search for Mobile paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mnasnet1_0 = flowvision.models.mnasnet1_0(pretrained=False, progress=True)
```

flowvision.models.**mnasnet1_3**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the MNASNet model with depth multiplier of 1.3.

Note: MNASNet model with depth multiplier of 1.3 from the MnasNet: Platform-Aware Neural Architecture Search for Mobile paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mnasnet1_3 = flowvision.models.mnasnet1_3(pretrained=False, progress=True)
```

6.1.13 GhostNet

flowvision.models.**ghostnet**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*)
Constructs the GhostNet model.

Note: GhostNet model from GhostNet: More Features from Cheap Operations.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> ghostnet = flowvision.models.ghostnet(pretrained=True, progress=True)
```

6.1.14 Res2Net

flowvision.models.**res2net101_26w_4s**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the Res2Net-101_26w_4s model.

Note: Res2Net-101_26w_4s model from the Res2Net: A New Multi-scale Backbone Architecture paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> res2net101_26w_4s = flowvision.models.res2net101_26w_4s(pretrained=False,␣
↪progress=True)
```

flowvision.models.**res2net50_14w_8s**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the Res2Net-50_14w_8s model.

Note: Res2Net-50_14w_8s model from the Res2Net: A New Multi-scale Backbone Architecture paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> res2net50_14w_8s = flowvision.models.res2net50_14w_8s(pretrained=False,␣
↪progress=True)
```

flowvision.models.**res2net50_26w_4s**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Res2Net-50_26w_4s model.

Note: Res2Net-50_26w_4s model from the Res2Net: A New Multi-scale Backbone Architecture paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> res2net50_26w_4s = flowvision.models.res2net50_26w_4s(pretrained=False,␣
↪progress=True)
```

flowvision.models.**res2net50_26w_6s**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Res2Net-50_26w_6s model.

Note: Res2Net-50_26w_6s model from the Res2Net: A New Multi-scale Backbone Architecture paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> res2net50_26w_6s = flowvision.models.res2net50_26w_6s(pretrained=False,␣
↪progress=True)
```

flowvision.models.**res2net50_26w_8s**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Res2Net-50_26w_8s model.

Note: Res2Net-50_26w_8s model from the Res2Net: A New Multi-scale Backbone Architecture paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> res2net50_26w_8s = flowvision.models.res2net50_26w_8s(pretrained=False,␣
↪progress=True)
```

flowvision.models.**res2net50_48w_2s**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Res2Net-50_48w_2s model.

Note: Res2Net-50_48w_2s model from the Res2Net: A New Multi-scale Backbone Architecture paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> res2net50_48w_2s = flowvision.models.res2net50_48w_2s(pretrained=False,␣
↪progress=True)
```

6.1.15 EfficientNet

flowvision.models.**efficientnet_b0**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B0 model.

Note: EfficientNet B0 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (256, 224) for efficientnet-b0 model when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b0 = flowvision.models.efficientnet_b0(pretrained=False,␣
↪progress=True)
```

(continues on next page)

(continued from previous page)

flowvision.models.**efficientnet_b1**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B1 model.

Note: EfficientNet B1 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (256, 240) for efficientnet-b1 model when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b1 = flowvision.models.efficientnet_b1(pretrained=False,␣
↪progress=True)
```

flowvision.models.**efficientnet_b2**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B2 model.

Note: EfficientNet B2 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (288, 288) for efficientnet-b2 model when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b2 = flowvision.models.efficientnet_b2(pretrained=False,␣
↪progress=True)
```

flowvision.models.**efficientnet_b3**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B3 model.

Note: EfficientNet B3 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (320, 300) for efficientnet-b3 model

when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b3 = flowvision.models.efficientnet_b3(pretrained=False,␣
↪progress=True)
```

flowvision.models.**efficientnet_b4**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B4 model.

Note: EfficientNet B4 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (384, 380) for efficientnet-b4 model when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b4 = flowvision.models.efficientnet_b4(pretrained=False,␣
↪progress=True)
```

flowvision.models.**efficientnet_b5**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B5 model.

Note: EfficientNet B5 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (489, 456) for efficientnet-b5 model when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b5 = flowvision.models.efficientnet_b5(pretrained=False,␣
↪progress=True)
```

flowvision.models.**efficientnet_b6**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B6 model.

Note: EfficientNet B6 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (561, 528) for efficientnet-b6 model when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b6 = flowvision.models.efficientnet_b6(pretrained=False,␣
↪progress=True)
```

flowvision.models.**efficientnet_b7**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.efficientnet.EfficientNet

Constructs the EfficientNet B7 model.

Note: EfficientNet B7 model architecture from the EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks paper. Note that the (resize-size, crop-size) should be (633, 600) for efficientnet-b7 model when training and testing.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> efficientnet_b7 = flowvision.models.efficientnet_b7(pretrained=False,␣
↪progress=True)
```

6.1.16 RegNet

flowvision.models.**regnet_x_16gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetX-16GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_x_16gf = flowvision.models.regnet_x_16gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_x_1_6gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetX-1.6GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_x_1_6gf = flowvision.models.regnet_x_1_6gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_x_32gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetX-32GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_x_32gf = flowvision.models.regnet_x_32gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_x_3_2gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetX-3.2GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_x_3_2gf = flowvision.models.regnet_x_3_2gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_x_400mf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetX-400MF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_x_400mf = flowvision.models.regnet_x_400mf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_x_800mf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetX-800MF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_x_800mf = flowvision.models.regnet_x_800mf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_x_8gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetX-8GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_x_8gf = flowvision.models.regnet_x_8gf(pretrained=False, progress=True)
```

flowvision.models.**regnet_y_16gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetY-16GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_y_16gf = flowvision.models.regnet_y_16gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_y_1_6gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetY-1.6GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_y_1_6gf = flowvision.models.regnet_y_1_6gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_y_32gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetY-32GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_y_32gf = flowvision.models.regnet_y_32gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_y_3_2gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetY-3.2GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_y_3_2gf = flowvision.models.regnet_y_3_2gf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_y_400mf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetY-400MF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_y_400mf = flowvision.models.regnet_y_400mf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_y_800mf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetY-800MF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_y_800mf = flowvision.models.regnet_y_800mf(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regnet_y_8gf**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs: Any*) → flowvision.models.regnet.RegNet

Constructs a RegNetY-8GF architecture from "Designing Network Design Spaces".

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regnet_y_8gf = flowvision.models.regnet_y_8gf(pretrained=False, progress=True)
```

6.1.17 ReXNet

flowvision.models.**rexnetv1_1_0**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ReXNet model with width multiplier of 1.0.

Note: ReXNet model with width multiplier of 1.0 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: True

For example:

```
>>> import flowvision
>>> rexnetv1_1_0 = flowvision.models.rexnetv1_1_0(pretrained=False, progress=True)
```

flowvision.models.**rexnetv1_1_3**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ReXNet model with width multiplier of 1.3.

Note: ReXNet model with width multiplier of 1.3 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: True

For example:

```
>>> import flowvision
>>> rexnetv1_1_3 = flowvision.models.rexnetv1_1_3(pretrained=False, progress=True)
```

flowvision.models.**rexnetv1_1_5**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ReXNet model with width multiplier of 1.5.

Note: ReXNet model with width multiplier of 1.5 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: True

For example:

```
>>> import flowvision
>>> rexnetv1_1_5 = flowvision.models.rexnetv1_1_5(pretrained=False, progress=True)
```

flowvision.models.**rexnetv1_2_0**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ReXNet model with width multiplier of 2.0.

Note: ReXNet model with width multiplier of 2.0 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> rexnetv1_2_0 = flowvision.models.rexnetv1_2_0(pretrained=False, progress=True)
```

flowvision.models.**rexnetv1_3_0**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ReXNet model with width multiplier of 3.0.

Note: ReXNet model with width multiplier of 3.0 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> rexnetv1_3_0 = flowvision.models.rexnetv1_3_0(pretrained=False, progress=True)
```

flowvision.models.**rexnet_lite_1_0**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ReXNet-lite model with width multiplier of 1.0.

Note: ReXNet-lite model with width multiplier of 1.0 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> rexnet_lite_1_0 = flowvision.models.rexnet_lite_1_0(pretrained=False,␣
↪progress=True)
```

flowvision.models.**rexnet_lite_1_3**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ReXNet-lite model with width multiplier of 1.3.

Note: ReXNet-lite model with width multiplier of 1.3 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> rexnet_lite_1_3 = flowvision.models.rexnet_lite_1_3(pretrained=False,␣
↪progress=True)
```

`flowvision.models.`**rexnet_lite_1_5**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ReXNet-lite model with width multiplier of 1.5.

Note: ReXNet-lite model with width multiplier of 1.5 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> rexnet_lite_1_5 = flowvision.models.rexnet_lite_1_5(pretrained=False,␣
↪progress=True)
```

`flowvision.models.`**rexnet_lite_2_0**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ReXNet-lite model with width multiplier of 2.0.

Note: ReXNet-lite model with width multiplier of 2.0 from the Rethinking Channel Dimensions for Efficient Model Design paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> rexnet_lite_2_0 = flowvision.models.rexnet_lite_2_0(pretrained=False,␣
↪progress=True)
```

6.1.18 SENet

flowvision.models.**se_resnet101**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the SE-ResNet101 model trained on ImageNet2012.

Note: SE-ResNet101 model from Squeeze-and-Excitation Networks. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> se_resnet101 = flowvision.models.se_resnet101(pretrained=False, progress=True)
```

flowvision.models.**se_resnet152**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the SE-ResNet152 model trained on ImageNet2012.

Note: SE-ResNet152 model Squeeze-and-Excitation Networks. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> se_resnet152 = flowvision.models.se_resnet152(pretrained=False, progress=True)
```

flowvision.models.**se_resnet50**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the SE-ResNet50 model trained on ImageNet2012.

Note: SE-ResNet50 model from Squeeze-and-Excitation Networks. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> se_resnet50 = flowvision.models.se_resnet50(pretrained=False, progress=True)
```

flowvision.models.**se_resnext101_32x4d**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the SE-ResNeXt101-32x4d model trained on ImageNet2012.

Note: SE-ResNeXt101-32x4d model from Squeeze-and-Excitation Networks. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> se_resnext101_32x4d = flowvision.models.se_resnext101_32x4d(pretrained=False,␣
↪progress=True)
```

flowvision.models.**se_resnext50_32x4d**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the SE-ResNeXt50-32x4d model trained on ImageNet2012.

Note: SE-ResNeXt50-32x4d model from Squeeze-and-Excitation Networks. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> se_resnext50_32x4d = flowvision.models.se_resnext50_32x4d(pretrained=False,␣
↪progress=True)
```

flowvision.models.**senet154**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the SENet-154 model trained on ImageNet2012.

Note: seneSENet-154t154 model from Squeeze-and-Excitation Networks. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> senet154 = flowvision.models.senet154(pretrained=False, progress=True)
```

6.1.19 ViT

`flowvision.models.`**vit_base_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch16-224 model.

Note: ViT-Base-patch16-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch16_224 = flowvision.models.vit_base_patch16_
↪224(pretrained=False, progress=True)
```

`flowvision.models.`**vit_base_patch16_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch16-224 ImageNet21k pretrained model.

Note: ViT-Base-patch16-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch16_224_in21k = flowvision.models.vit_base_patch16_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_base_patch16_224_miil**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch16-224-miil model.

Note: ViT-Base-patch16-224-miil model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch16_224_miil = flowvision.models.vit_base_patch16_224_
↪miil(pretrained=False, progress=True)
```

flowvision.models.**vit_base_patch16_224_miil_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch16-224-miil ImageNet21k pretrained model.

Note: ViT-Base-patch16-224-miil ImageNet21k pretrained model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch16_224_miil_in21k = flowvision.models.vit_base_patch16_224_miil_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_base_patch16_224_sam**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch16-224-sam model.

Note: ViT-Base-patch16-224-sam model from "When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch16_224_sam = flowvision.models.vit_base_patch16_224_
↪sam(pretrained=False, progress=True)
```

`flowvision.models.`**vit_base_patch16_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch16-384 model.

Note: ViT-Base-patch16-384 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch16_384 = flowvision.models.vit_base_patch16_
↪384(pretrained=False, progress=True)
```

`flowvision.models.`**vit_base_patch32_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch32-224 model.

Note: ViT-Base-patch32-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch32_224 = flowvision.models.vit_base_patch32_
↪224(pretrained=False, progress=True)
```

flowvision.models.**vit_base_patch32_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch32-224 ImageNet21k pretrained model.

Note: ViT-Base-patch32-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch32_224_in21k = flowvision.models.vit_base_patch32_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_base_patch32_224_sam**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch32-224-sam model.

Note: ViT-Base-patch32-224-sam model from "When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch32_224_sam = flowvision.models.vit_base_patch32_224_
↪sam(pretrained=False, progress=True)
```

flowvision.models.**vit_base_patch32_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Base-patch32-384 model.

Note: ViT-Base-patch32-384 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch32_384 = flowvision.models.vit_base_patch32_
↪384(pretrained=False, progress=True)
```

flowvision.models.**vit_base_patch8_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ViT-Base-patch8-224 model.

Note: ViT-Base-patch8-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch8_224 = flowvision.models.vit_base_patch8_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**vit_base_patch8_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ViT-Base-patch8-224 ImageNet21k pretrained model.

Note: ViT-Base-patch8-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_base_patch8_224_in21k = flowvision.models.vit_base_patch8_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_giant_patch14_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ViT-Giant-patch14-224 model.

Note: ViT-Giant-patch14-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_giant_patch14_224 = flowvision.models.vit_giant_patch14_
↪224(pretrained=False, progress=True)
```

`flowvision.models.`**vit_gigantic_patch14_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ViT-Gigantic-patch14-224 model.

Note: ViT-Giant-patch14-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_gigantic_patch14_224 = flowvision.models.vit_gigantic_patch14_
↪224(pretrained=False, progress=True)
```

`flowvision.models.`**vit_huge_patch14_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ViT-Huge-patch14-224 model.

Note: ViT-Huge-patch14-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_huge_patch14_224 = flowvision.models.vit_huge_patch14_
↪224(pretrained=False, progress=True)
```

flowvision.models.**vit_huge_patch14_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Huge-patch14-224 ImageNet21k pretrained model.

Note: ViT-Huge-patch14-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_huge_patch14_224_in21k = flowvision.models.vit_huge_patch14_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_large_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Large-patch16-224 model.

Note: ViT-Large-patch16-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_large_patch16_224 = flowvision.models.vit_large_patch16_
↪224(pretrained=False, progress=True)
```

flowvision.models.**vit_large_patch16_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Large-patch16-224 ImageNet21k pretrained model.

Note: ViT-Large-patch16-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_large_patch16_224_in21k = flowvision.models.vit_large_patch16_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_large_patch16_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Large-patch16-384 model.

Note: ViT-Large-patch16-384 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_large_patch16_384 = flowvision.models.vit_large_patch16_
↪384(pretrained=False, progress=True)
```

flowvision.models.**vit_large_patch32_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Large-patch32-224 model.

Note: ViT-Large-patch32-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_large_patch32_224 = flowvision.models.vit_large_patch32_
↪224(pretrained=False, progress=True)
```

flowvision.models.**vit_large_patch32_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Large-patch32-224 ImageNet21k pretrained model.

Note: ViT-Large-patch32-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_large_patch32_224_in21k = flowvision.models.vit_large_patch32_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_large_patch32_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Large-patch32-384 model.

Note: ViT-Large-patch32-384 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_large_patch32_384 = flowvision.models.vit_large_patch32_
↪384(pretrained=False, progress=True)
```

flowvision.models.**vit_small_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Small-patch16-224 model.

Note: ViT-Small-patch16-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_small_patch16_224 = flowvision.models.vit_small_patch16_
↪224(pretrained=False, progress=True)
```

flowvision.models.**vit_small_patch16_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Small-patch16-224 ImageNet21k pretrained model.

Note: ViT-Small-patch16-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_small_patch16_224_in21k = flowvision.models.vit_small_patch16_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_small_patch16_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Small-patch16-384 model.

Note: ViT-Small-patch16-384 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_small_patch16_384 = flowvision.models.vit_small_patch16_
↪384(pretrained=False, progress=True)
```

flowvision.models.**vit_small_patch32_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Small-patch32-224 model.

Note: ViT-Small-patch32-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_small_patch32_224 = flowvision.models.vit_small_patch32_
↪224(pretrained=False, progress=True)
```

`flowvision.models.`**`vit_small_patch32_224_in21k`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Small-patch32-224 ImageNet21k pretrained model.

Note: ViT-Small-patch32-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_small_patch32_224_in21k = flowvision.models.vit_small_patch32_224_
↪in21k(pretrained=False, progress=True)
```

`flowvision.models.`**`vit_small_patch32_384`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Small-patch32-384 model.

Note: ViT-Small-patch32-384 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_small_patch32_384 = flowvision.models.vit_small_patch32_
↪384(pretrained=False, progress=True)
```

`flowvision.models.`**`vit_tiny_patch16_224`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Tiny-patch16-224 model.

Note: ViT-Tiny-patch16-224 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_tiny_patch16_224 = flowvision.models.vit_tiny_patch16_
↪224(pretrained=False, progress=True)
```

flowvision.models.**vit_tiny_patch16_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Tiny-patch16-224 ImageNet21k pretrained model.

Note: ViT-Tiny-patch16-224 ImageNet21k pretrained model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_tiny_patch16_224_in21k = flowvision.models.vit_tiny_patch16_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**vit_tiny_patch16_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ViT-Tiny-patch16-384 model.

Note: ViT-Tiny-patch16-384 model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> vit_tiny_patch16_384 = flowvision.models.vit_tiny_patch16_
↪384(pretrained=False, progress=True)
```

6.1.20 DeiT

flowvision.models.**deit_base_distilled_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-224 distilled model.

Note: DeiT-Base-patch16-224 distilled model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_distilled_patch16_224 = flowvision.models.deit_base_distilled_
↪patch16_224(pretrained=False, progress=True)
```

flowvision.models.**deit_base_distilled_patch16_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-384 distilled model.

Note: DeiT-Base-patch16-384 distilled model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_distilled_patch16_384 = flowvision.models.deit_base_distilled_
↪patch16_384(pretrained=False, progress=True)
```

flowvision.models.**deit_base_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-224 model.

Note: DeiT-Base-patch16-224 model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_patch16_224 = flowvision.models.deit_base_patch16_
↪224(pretrained=False, progress=True)
```

flowvision.models.**deit_base_patch16_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-384 model.

Note: DeiT-Base-patch16-384 model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_patch16_384 = flowvision.models.deit_base_patch16_
↪384(pretrained=False, progress=True)
```

flowvision.models.**deit_small_distilled_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Small-patch16-224 distilled model.

Note: DeiT-Small-patch16-224 distilled model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_small_distilled_patch16_224 = flowvision.models.deit_small_distilled_
↪patch16_224(pretrained=False, progress=True)
```

flowvision.models.**deit_small_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Small-patch16-224 model.

Note: DeiT-Small-patch16-224 model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_small_patch16_224 = flowvision.models.deit_small_patch16_
↪224(pretrained=False, progress=True)
```

flowvision.models.**deit_tiny_distilled_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Tiny-patch16-224 distilled model.

Note: DeiT-Tiny-patch16-224 distilled model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_tiny_distilled_patch16_224 = flowvision.models.deit_tiny_distilled_
↪patch16_224(pretrained=False, progress=True)
```

flowvision.models.**deit_tiny_patch16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Tiny-patch16-224 model.

Note: DeiT-Tiny-patch16-224 model from "Training data-efficient image transformers & distillation through attention". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_tiny_patch16_224 = flowvision.models.deit_tiny_patch16_
↪224(pretrained=False, progress=True)
```

6.1.21 PVT

`flowvision.models.`**`pvt_large`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the PVT-large model.

Note: PVT-large model from "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> pvt_large = flowvision.models.pvt_large(pretrained=False, progress=True)
```

`flowvision.models.`**`pvt_medium`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the PVT-medium model.

Note: PVT-medium model from "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> pvt_medium = flowvision.models.pvt_medium(pretrained=False, progress=True)
```

flowvision.models.**pvt_small**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the PVT-small model.

Note: PVT-small model from "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> pvt_small = flowvision.models.pvt_small(pretrained=False, progress=True)
```

flowvision.models.**pvt_tiny**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the PVT-tiny model.

Note: PVT-tiny model from "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> pvt_tiny = flowvision.models.pvt_tiny(pretrained=False, progress=True)
```

6.1.22 Swin-Transformer

flowvision.models.**swin_base_patch4_window12_384**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs Swin-B 384x384 model trained on ImageNet-1k.

Note: Swin-B 384x384 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_base_patch4_window12_384 = flowvision.models.swin_base_patch4_window12_
↪384(pretrained=False, progress=True)
```

flowvision.models.**swin_base_patch4_window12_384_in22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs Swin-B 384x384 model pretrained on ImageNet-22k and fine tuned on ImageNet-1k.

Note: Swin-B 384x384 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_base_patch4_window12_384_in22k_to_1k = flowvision.models.swin_base_
↪patch4_window12_384_in22k_to_1k(pretrained=False, progress=True)
```

flowvision.models.**swin_base_patch4_window7_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs Swin-B 224x224 model trained on ImageNet-1k.

Note: Swin-B 224x224 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_base_patch4_window7_224 = flowvision.models.swin_base_patch4_window7_
↪224(pretrained=False, progress=True)
```

flowvision.models.**swin_base_patch4_window7_224_in22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs Swin-B 224x224 model pretrained on ImageNet-22k and fine tuned on ImageNet-1k.

Note: Swin-B 224x224 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_base_patch4_window7_224_in22k_to_1k = flowvision.models.swin_base_patch4_
↪window7_224_in22k_to_1k(pretrained=False, progress=True)
```

flowvision.models.**swin_large_patch4_window12_384_in22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs Swin-L 384x384 model pretrained on ImageNet-22k and fine tuned on ImageNet-1k.

Note: Swin-L 384x384 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_large_patch4_window12_384_in22k_to_1k = flowvision.models.swin_large_
↪patch4_window12_384_in22k_to_1k(pretrained=False, progress=True)
```

flowvision.models.**swin_large_patch4_window7_224_in22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs Swin-L 224x224 model pretrained on ImageNet-22k and fine tuned on ImageNet-1k.

Note: Swin-L 224x224 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_large_patch4_window7_224_in22k_to_1k = flowvision.models.swin_large_
↪patch4_window7_224_in22k_to_1k(pretrained=False, progress=True)
```

flowvision.models.**swin_small_patch4_window7_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs Swin-S 224x224 model trained on ImageNet-1k.

Note: Swin-S 224x224 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_small_patch4_window7_224 = flowvision.models.swin_small_patch4_window7_
↪224(pretrained=False, progress=True)
```

flowvision.models.**swin_tiny_patch4_window7_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs Swin-T 224x224 model trained on ImageNet-1k.

Note: Swin-T 224x224 model from "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> swin_tiny_patch4_window7_224 = flowvision.models.swin_tiny_patch4_window7_
↪224(pretrained=False, progress=True)
```

6.1.23 CSwin-Transformer

flowvision.models.**cswin_base_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs CSwin-B 224x224 model.

Note: CSwin-B 224x224 model from "CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cswin_base_224 = flowvision.models.cswin_base_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**cswin_base_384**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs CSwin-B 384x384 model.

Note: CSwin-B 384x384 model from "CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cswin_base_384 = flowvision.models.cswin_base_384(pretrained=False,␣
↪progress=True)
```

flowvision.models.**cswin_large_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs CSwin-L 224x224 model.

Note: CSwin-L 224x224 model from "CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cswin_large_224 = flowvision.models.cswin_large_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**cswin_large_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs CSwin-L 384x384 model.

Note: CSwin-L 384x384 model from "CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cswin_large_384 = flowvision.models.cswin_large_384(pretrained=False,␣
↪progress=True)
```

flowvision.models.**cswin_small_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs CSwin-S 224x224 model.

Note: CSwin-S 224x224 model from "CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cswin_small_224 = flowvision.models.cswin_small_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**cswin_tiny_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs CSwin-T 224x224 model.

Note: CSwin-T 224x224 model from "CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cswin_tiny_224 = flowvision.models.cswin_tiny_224(pretrained=False,␣
↪progress=True)
```

6.1.24 CrossFormer

flowvision.models.**crossformer_base_patch4_group7_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs CrossFormer-B 224x224 model.

Note: CrossFormer-B 224x224 model from "CrossFormer: A Versatile Vision Transformer Based on Cross-scale Attention".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> crossformer_base_patch4_group7_224 = flowvision.models.crossformer_base_
↪patch4_group7_224(pretrained=False, progress=True)
```

flowvision.models.**crossformer_large_patch4_group7_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs CrossFormer-L 224x224 model.

Note: CrossFormer-L 224x224 model from "CrossFormer: A Versatile Vision Transformer Based on Cross-scale Attention".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> crossformer_large_patch4_group7_224 = flowvision.models.crossformer_large_
↪patch4_group7_224(pretrained=False, progress=True)
```

flowvision.models.**crossformer_small_patch4_group7_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs CrossFormer-S 224x224 model.

Note: CrossFormer-S 224x224 model from "CrossFormer: A Versatile Vision Transformer Based on Cross-scale Attention".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> crossformer_small_patch4_group7_224 = flowvision.models.crossformer_small_
↪patch4_group7_224(pretrained=False, progress=True)
```

flowvision.models.**crossformer_tiny_patch4_group7_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs CrossFormer-T 224x224 model.

Note: CrossFormer-T 224x224 model from "CrossFormer: A Versatile Vision Transformer Based on Cross-scale Attention".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> crossformer_tiny_patch4_group7_224 = flowvision.models.crossformer_tiny_
↪patch4_group7_224(pretrained=False, progress=True)
```

6.1.25 PoolFormer

flowvision.models.**poolformer_m36**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the PoolFormer-M36 model.

Note: PoolFormer-M36 model. From *"MetaFormer is Actually What You Need for Vision" <https://arxiv.org/abs/2111.11418> _*.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> poolformer_m36 = flowvision.models.poolformer_m36(pretrained=False,␣
↪progress=True)
```

flowvision.models.**poolformer_m48**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the PoolFormer-M48 model.

Note: PoolFormer-M48 model. From *"MetaFormer is Actually What You Need for Vision" <https://arxiv.org/abs/2111.11418> _*.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> poolformer_m48 = flowvision.models.poolformer_m48(pretrained=False,␣
↪progress=True)
```

flowvision.models.**poolformer_s12**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the PoolFormer-S12 model.

Note: PoolFormer-S12 model. From *"MetaFormer is Actually What You Need for Vision" <https://arxiv.org/abs/2111.11418> _*.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> poolformer_s12 = flowvision.models.poolformer_s12(pretrained=False,␣
↪progress=True)
```

flowvision.models.**poolformer_s24**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the PoolFormer-S24 model.

Note: PoolFormer-S24 model. From *"MetaFormer is Actually What You Need for Vision" <https://arxiv.org/abs/2111.11418> _*.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> poolformer_s24 = flowvision.models.poolformer_s24(pretrained=False,␣
↪progress=True)
```

flowvision.models.**poolformer_s36**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the PoolFormer-S36 model.

Note: PoolFormer-S36 model. From *"MetaFormer is Actually What You Need for Vision" <https://arxiv.org/abs/2111.11418> _*.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> poolformer_s36 = flowvision.models.poolformer_s36(pretrained=False,␣
↪progress=True)
```

6.1.26 UniFormer

flowvision.models.**uniformer_base**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the UniFormer-Base model.

Note:

UniFormer-Base model from *UniFormer: Unified Transformer for Efficient Spatiotemporal Representation Learning* - https://arxiv.org/abs/2201.04676

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> uniformer_base= flowvision.models.uniformer_base(pretrained=False,␣
↪progress=True)
```

flowvision.models.**uniformer_base_ls**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the UniFormer-Base-Ls model.

Note:

UniFormer-Base-Ls model from *UniFormer: Unified Transformer for Efficient Spatiotemporal Representation Learning* - https://arxiv.org/abs/2201.04676

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> uniformer_base_ls = flowvision.models.uniformer_base_ls(pretrained=False,␣
↪progress=True)
```

flowvision.models.**uniformer_small**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the UniFormer-Small model.

Note:

UniFormer-Small model from *UniFormer: Unified Transformer for Efficient Spatiotemporal Representation Learning* - https://arxiv.org/abs/2201.04676

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> uniformer_small = flowvision.models.uniformer_small(pretrained=False,␣
↪progress=True)
```

`flowvision.models.`**uniformer_small_plus** (*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the UniFormer-Small-Plus model.

Note:

UniFormer-Small-Plus model from *UniFormer: Unified Transformer for Efficient Spatiotemporal Representation Learning* https://arxiv.org/abs/2201.04676

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> uniformer_small_plus = flowvision.models.uniformer_small_
↪plus(pretrained=False, progress=True)
```

6.1.27 Mlp-Mixer

`flowvision.models.`**mlp_mixer_b16_224** (*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Mixer-B/16 224x224 model.

Note: Mixer-B/16 224x224 model from "MLP-Mixer: An all-MLP Architecture for Vision".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_b16_224 = flowvision.models.mlp_mixer_b16_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mlp_mixer_b16_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Mixer-B/16 224x224 ImageNet21k pretrained model.

Note: Mixer-B/16 224x224 ImageNet21k pretrained model from "MLP-Mixer: An all-MLP Architecture for Vision". Note that this model is the pretrained model for fine-tune on different datasets.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_b16_224_in21k = flowvision.models.mlp_mixer_b16_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**mlp_mixer_b16_224_miil**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Mixer-B/16 224x224 model with different weights.

Note: Mixer-B/16 224x224 model from "MLP-Mixer: An all-MLP Architecture for Vision". Weights taken from: https://github.com/Alibaba-MIIL/ImageNet21K.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_b16_224_miil = flowvision.models.mlp_mixer_b16_224_
↪miil(pretrained=False, progress=True)
```

flowvision.models.**mlp_mixer_b16_224_miil_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Mixer-B/16 224x224 ImageNet21k pretrained model.

Note: Mixer-B/16 224x224 ImageNet21k pretrained model from "MLP-Mixer: An all-MLP Architecture for Vision". Weights taken from: https://github.com/Alibaba-MIIL/ImageNet21K

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_b16_224_miil_in21k = flowvision.models.mlp_mixer_b16_224_miil_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**mlp_mixer_b32_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the Mixer-B/32 224x224 model.

Note: Mixer-B/32 224x224 model from "MLP-Mixer: An all-MLP Architecture for Vision".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_b32_224 = flowvision.models.mlp_mixer_b32_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mlp_mixer_l16_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the Mixer-L/16 224x224 model.

Note: Mixer-L/16 224x224 model from "MLP-Mixer: An all-MLP Architecture for Vision".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_l16_224 = flowvision.models.mlp_mixer_l16_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mlp_mixer_l16_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the Mixer-L/16 224x224 ImageNet21k pretrained model.

Note: Mixer-L/16 224x224 ImageNet21k pretrained model from "MLP-Mixer: An all-MLP Architecture for Vision". Note that this model is the pretrained model for fine-tune on different datasets.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_l16_224_in21k = flowvision.models.mlp_mixer_l16_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**mlp_mixer_l32_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Mixer-L/32 224x224 model.

Note: Mixer-L/32 224x224 model from "MLP-Mixer: An all-MLP Architecture for Vision".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_l32_224 = flowvision.models.mlp_mixer_l32_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mlp_mixer_s16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the Mixer-S/16 224x224 model.

Note: Mixer-S/16 224x224 model from "MLP-Mixer: An all-MLP Architecture for Vision".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_s16_224 = flowvision.models.mlp_mixer_s16_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mlp_mixer_s32_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the Mixer-S/32 224x224 model.

Note: Mixer-S/32 224x224 model from "MLP-Mixer: An all-MLP Architecture for Vision".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mlp_mixer_s32_224 = flowvision.models.mlp_mixer_s32_224(pretrained=False,␣
↪progress=True)
```

6.1.28 ResMLP

flowvision.models.**resmlp_12_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ResMLP-12 model.

Note: ResMLP-12 model from "ResMLP: Feedforward networks for image classification with data-efficient training".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_12_224 = flowvision.models.resmlp_12_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**resmlp_12_224_dino**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ResMLP-12 model trained under DINO proposed in "Emerging Properties in Self-Supervised Vision Transformers".

Note: ResMLP-12 model with distillation from "ResMLP: Feedforward networks for image classification with data-efficient training". Note that this model is the same as resmlp_12 but the pretrained weight is different.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_12_224_dino = flowvision.models.resmlp_12_224_dino(pretrained=False,␣
↪progress=True)
```

`flowvision.models.`**resmlp_12_distilled_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResMLP-12 model with distillation.

Note: ResMLP-12 model with distillation from "ResMLP: Feedforward networks for image classification with data-efficient training". Note that this model is the same as resmlp_12 but the pretrained weight is different.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_12_distilled_224 = flowvision.models.resmlp_12_distilled_
↪224(pretrained=False, progress=True)
```

`flowvision.models.`**resmlp_24_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResMLP-24 model.

Note: ResMLP-24 model from "ResMLP: Feedforward networks for image classification with data-efficient training".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_24_224 = flowvision.models.resmlp_24_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**resmlp_24_224_dino**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ResMLP-24 model trained under DINO proposed in "Emerging Properties in Self-Supervised Vision Transformers".

Note: ResMLP-24 model with distillation from "ResMLP: Feedforward networks for image classification with data-efficient training". Note that this model is the same as resmlp_24 but the pretrained weight is different.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_24_224_dino = flowvision.models.resmlp_24_224_dino(pretrained=False,␣
↪progress=True)
```

flowvision.models.**resmlp_24_distilled_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ResMLP-24 model with distillation.

Note: ResMLP-24 model with distillation from "ResMLP: Feedforward networks for image classification with data-efficient training". Note that this model is the same as resmlp_24 but the pretrained weight is different.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_24_distilled_224 = flowvision.models.resmlp_24_distilled_
↪224(pretrained=False, progress=True)
```

flowvision.models.**resmlp_36_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ResMLP-36 model.

Note: ResMLP-36 model from "ResMLP: Feedforward networks for image classification with data-efficient training".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_36_224 = flowvision.models.resmlp_36_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**resmlp_36_distilled_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResMLP-36 model with distillation.

Note: ResMLP-36 model with distillation from "ResMLP: Feedforward networks for image classification with data-efficient training". Note that this model is the same as resmlp_36 but the pretrained weight is different.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_36_distilled_224 = flowvision.models.resmlp_36_distilled_
↪224(pretrained=False, progress=True)
```

flowvision.models.**resmlp_big_24_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResMLP-Big-24 model.

Note: ResMLP-Big-24 model from "ResMLP: Feedforward networks for image classification with data-efficient training".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_big_24_224 = flowvision.models.resmlp_big_24_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**resmlp_big_24_224_in22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ImageNet22k pretrained ResMLP-B-24 model.

Note: ImageNet22k pretrained ResMLP-B-24 model from "ResMLP: Feedforward networks for image classification with data-efficient training". Note that this model is the same as resmlpB_24 but the pretrained weight is different.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_big_24_224_in22k_to_1k = flowvision.models.resmlp_big_24_224_in22k_to_
↪1k(pretrained=False, progress=True)
```

flowvision.models.**resmlp_big_24_distilled_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ResMLP-B-24 model with distillation.

Note: ResMLP-B-24 model with distillation from "ResMLP: Feedforward networks for image classification with data-efficient training". Note that this model is the same as resmlpB_24 but the pretrained weight is different.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> resmlp_big_24_distilled_224 = flowvision.models.resmlp_big_24_distilled_
↪224(pretrained=False, progress=True)
```

6.1.29 gMLP

flowvision.models.**gmlp_b16_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the gMLP-base-16 224x224 model.

Note: gMLP-base-16 224x224 model from "Pay Attention to MLPs".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> gmlp_b16_224 = flowvision.models.gmlp_b16_224(pretrained=False, progress=True)
```

flowvision.models.**gmlp_s16_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the gMLP-small-16 224x224 model.

Note: gMLP-small-16 224x224 model from "Pay Attention to MLPs".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> gmlp_s16_224 = flowvision.models.gmlp_s16_224(pretrained=False, progress=True)
```

flowvision.models.**gmlp_ti16_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the gMLP-tiny-16 224x224 model.

Note: gMLP-tiny-16 224x224 model from "Pay Attention to MLPs".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> gmlp_ti16_224 = flowvision.models.gmlp_ti16_224(pretrained=False,␣
↪progress=True)
```

6.1.30 ConvMixer

flowvision.models.**convmixer_1024_20**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs the ConvMixer model with 20 depth and 1024 hidden size.

Note: ConvMixer model with 20 depth and 1024 hidden size from the Patched Are All You Need? paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> convmixer_1024_20 = flowvision.models.convmixer_1024_20(pretrained=False,␣
↪progress=True)
```

flowvision.models.**convmixer_1536_20**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs the ConvMixer model with 20 depth and 1536 hidden size.

Note: ConvMixer model with 20 depth and 1536 hidden size from the Patched Are All You Need? paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> convmixer_1536_20 = flowvision.models.convmixer_1536_20(pretrained=False,␣
↪progress=True)
```

flowvision.models.**convmixer_768_32_relu**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs the ConvMixer model with 32 depth and 768 hidden size and ReLU activation layer.

Note: ConvMixer model with 32 depth and 768 hidden size and ReLU activation layer from the Patched Are All You Need? paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: True

For example:

```
>>> import flowvision
>>> convmixer_768_32_relu = flowvision.models.convmixer_768_32_
↪relu(pretrained=False, progress=True)
```

6.1.31 ConvNeXt

flowvision.models.**convnext_base_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Base model trained on ImageNet2012.

Note: ConvNext-Base model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> convnext_base_224 = flowvision.models.convnext_base_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**convnext_base_224_22k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Base model pretrained on ImageNet22k.

Note: ConvNext-Base model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> convnext_base_224_22k = flowvision.models.convnext_base_224_
↪22k(pretrained=False, progress=True)
```

flowvision.models.**convnext_base_224_22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Base model pretrained on ImageNet22k and finetuned on ImageNet2012.

Note: ConvNext-Base model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_base_224_22k_to_1k = flowvision.models.convnext_base_224_22k_to_
↪1k(pretrained=False, progress=True)
```

flowvision.models.**convnext_base_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Base model trained on ImageNet2012.

Note: ConvNext-Base model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_base_384 = flowvision.models.convnext_base_384(pretrained=False,␣
↪progress=True)
```

flowvision.models.**convnext_base_384_22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Base model pretrained on ImageNet22k and finetuned on ImageNet2012.

Note: ConvNext-Base model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> convnext_base_384_22k_to_1k = flowvision.models.convnext_base_384_22k_to_
↪1k(pretrained=False, progress=True)
```

flowvision.models.**convnext_iso_base_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ConvNext-Isotropic-Base model trained on ImageNet2012.

Note: ConvNext-Isotropic-Base model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> convnext_iso_base_224 = flowvision.models.convnext_iso_base_
↪224(pretrained=False, progress=True)
```

flowvision.models.**convnext_iso_large_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ConvNext-Isotropic-Large model trained on ImageNet2012.

Note: ConvNext-Isotropic-Large model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: True

For example:

```
>>> import flowvision
>>> convnext_iso_large_224 = flowvision.models.convnext_iso_large_
↪224(pretrained=False, progress=True)
```

flowvision.models.**convnext_iso_small_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ConvNext-Isotropic-Small model trained on ImageNet2012.

Note: ConvNext-Isotropic-Small model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_iso_small_224 = flowvision.models.convnext_iso_small_
↪224(pretrained=False, progress=True)
```

`flowvision.models.`**convnext_large_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ConvNext-Large model trained on ImageNet2012.

Note: ConvNext-Large model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_large_224 = flowvision.models.convnext_large_224(pretrained=False,␣
↪progress=True)
```

`flowvision.models.`**convnext_large_224_22k**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ConvNext-Large model trained on ImageNet22k.

Note: ConvNext-Large model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_large_224_22k = flowvision.models.convnext_large_224_
↪22k(pretrained=False, progress=True)
```

flowvision.models.**convnext_large_224_22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Large model pretrained on ImageNet22k and finetuned on ImageNet2012.

Note: ConvNext-Large model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_large_224_22k_to_1k = flowvision.models.convnext_large_224_22k_to_
↪1k(pretrained=False, progress=True)
```

flowvision.models.**convnext_large_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Large model trained on ImageNet2012.

Note: ConvNext-Large model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_large_384 = flowvision.models.convnext_large_384(pretrained=False,␣
↪progress=True)
```

flowvision.models.**convnext_large_384_22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-Large model pretrained on ImageNet22k and finetuned on ImageNet2012.

Note: ConvNext-Large model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545>* _. The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_large_384_22k_to_1k = flowvision.models.convnext_large_384_22k_to_
↪1k(pretrained=False, progress=True)
```

flowvision.models.**convnext_small_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ConvNext-Small model trained on ImageNet2012.

Note: ConvNext-Small model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_small_224 = flowvision.models.convnext_small_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**convnext_tiny_224**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the ConvNext-Tiny model trained on ImageNet2012.

Note: ConvNext-Tiny model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_tiny_224 = flowvision.models.convnext_tiny_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**convnext_xlarge_224_22k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-xLarge model pretrained on ImageNet22k.

Note: ConvNext-xLarge model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_xlarge_224_22k = flowvision.models.convnext_xlarge_224_
↪22k(pretrained=False, progress=True)
```

flowvision.models.**convnext_xlarge_224_22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-xLarge model pretrained on ImageNet22k and finetuned on ImageNet2012.

Note: ConvNext-xLarge model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_xlarge_224_22k_to_1k = flowvision.models.convnext_xlarge_224_22k_to_
↪1k(pretrained=False, progress=True)
```

flowvision.models.**convnext_xlarge_384_22k_to_1k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the ConvNext-xLarge model pretrained on ImageNet22k and finetuned on ImageNet2012.

Note: ConvNext-xLarge model from *"A ConvNet for the 2020s" <https://arxiv.org/abs/2201.03545> _*. The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderrt. Default: `True`

For example:

```
>>> import flowvision
>>> convnext_xlarge_384_22k_to_1k = flowvision.models.convnext_xlarge_384_22k_to_
↪1k(pretrained=False, progress=True)
```

6.1.32 RegionViT

`flowvision.models.`**regionvit_base_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Base-224 model.

Note: RegionViT-Base-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_base_224 = flowvision.models.regionvit_base_224(pretrained=False,␣
↪progress=True)
```

`flowvision.models.`**regionvit_base_w14_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Base-w14-224 model.

Note: RegionViT-Base-w14-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_base_w14_224 = flowvision.models.regionvit_base_w14_
↪224(pretrained=False, progress=True)
```

flowvision.models.**regionvit_base_w14_peg_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Base-w14-peg-224 model.

Note: RegionViT-Base-w14-peg-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_base_w14_peg_224 = flowvision.models.regionvit_base_w14_peg_
↪224(pretrained=False, progress=True)
```

flowvision.models.**regionvit_medium_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Medium-224 model.

Note: RegionViT-Medium-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_medium_224 = flowvision.models.regionvit_medium_
↪224(pretrained=False, progress=True)
```

flowvision.models.**regionvit_small_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Small-224 model.

Note: RegionViT-Small-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_small_224 = flowvision.models.regionvit_small_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**regionvit_small_w14_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Small-w14-224 model.

Note: RegionViT-Small-w14-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_small_w14_224 = flowvision.models.regionvit_small_w14_
↪224(pretrained=False, progress=True)
```

flowvision.models.**regionvit_small_w14_peg_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Small-w14-peg-224 model.

Note: RegionViT-Small-w14-peg-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_small_w14_peg_224 = flowvision.models.regionvit_small_w14_peg_
↪224(pretrained=False, progress=True)
```

flowvision.models.**regionvit_tiny_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the RegionViT-Tiny-224 model.

Note: RegionViT-Tiny-224 model from "RegionViT: Regional-to-Local Attention for Vision Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> regionvit_tiny_224 = flowvision.models.regionvit_tiny_224(pretrained=False,␣
↪progress=True)
```

6.1.33 VAN

`flowvision.models.`**`van_base`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs the VAN-Base model trained on ImageNet-1k.

Note: VAN-Base model from "Visual Attention Network". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> van_base = flowvision.models.van_base(pretrained=False, progress=True)
```

`flowvision.models.`**`van_large`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs the VAN-Large model trained on ImageNet-1k.

Note: VAN-Large model from "Visual Attention Network". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> van_large = flowvision.models.van_large(pretrained=False, progress=True)
```

flowvision.models.**van_small**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs the VAN-Small model trained on ImageNet-1k.

Note: VAN-Small model from "Visual Attention Network". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> van_small = flowvision.models.van_small(pretrained=False, progress=True)
```

flowvision.models.**van_tiny**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs the VAN-Tiny model trained on ImageNet-1k.

Note: VAN-Tiny model from "Visual Attention Network". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> van_tiny = flowvision.models.van_tiny(pretrained=False, progress=True)
```

6.1.34 LeViT

flowvision.models.**levit_128**(*num_classes=1000*, *distillation=True*, *pretrained=False*)
Constructs the LeViT-128 model.

Note: LeViT-128 model architecture from the LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> levit_128 = flowvision.models.levit_128(pretrained=False, progress=True)
```

flowvision.models.**levit_128s**(*num_classes=1000*, *distillation=True*, *pretrained=False*)
Constructs the LeViT-128S model.

Note: LeViT-128S model architecture from the LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> levit_128s = flowvision.models.levit_128s(pretrained=False, progress=True)
```

flowvision.models.**levit_192**(*num_classes=1000*, *distillation=True*, *pretrained=False*)
Constructs the LeViT-192 model.

Note: LeViT-192 model architecture from the LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> levit_192 = flowvision.models.levit_192(pretrained=False, progress=True)
```

flowvision.models.**levit_256**(*num_classes=1000*, *distillation=True*, *pretrained=False*)
Constructs the LeViT-256 model.

Note: LeViT-256 model architecture from the LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> levit_256 = flowvision.models.levit_256(pretrained=False, progress=True)
```

flowvision.models.**levit_384**(*num_classes=1000*, *distillation=True*, *pretrained=False*)

Constructs the LeViT-384 model.

Note: LeViT-384 model architecture from the LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference paper.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> levit_384 = flowvision.models.levit_384(pretrained=False, progress=True)
```

6.1.35 MobileViT

flowvision.models.**mobilevit_small**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs MobileViT-S 224x224 model pretrained on ImageNet-1k.

Note: MobileViT-S 224x224 model from "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mobilevit_s = flowvision.models.mobilevit_small(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mobilevit_x_small**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs MobileViT-XS 224x224 model pretrained on ImageNet-1k.

Note: MobileViT-XS 224x224 model from "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mobilevit_xs = flowvision.models.mobilevit_x_small(pretrained=False,␣
↪progress=True)
```

flowvision.models.**mobilevit_xx_small**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs MobileViT-XXS 224x224 model pretrained on ImageNet-1k.

Note: MobileViT-XXS 224x224 model from "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> mobilevit_xxs = flowvision.models.mobilevit_xx_small(pretrained=False,␣
↪progress=True)
```

6.1.36 DeiT-III

flowvision.models.**deit_base_patch16_LS_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-LS-224 model.

Note: DeiT-Base-patch16-LS-224 model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_patch16_LS_224 = flowvision.models.deit_base_patch16_LS_
↪224(pretrained=False, progress=True)
```

flowvision.models.**deit_base_patch16_LS_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-LS-224 ImageNet21k pretrained model.

Note: DeiT-Base-patch16-LS-224 ImageNet21k pretrained model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_patch16_LS_224_in21k = flowvision.models.deit_base_patch16_LS_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**deit_base_patch16_LS_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-LS-384 model.

Note: DeiT-Base-patch16-LS-384 model from "DeiT III: Revenge of the ViT". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_patch16_LS_384 = flowvision.models.deit_base_patch16_LS_
↪384(pretrained=False, progress=True)
```

flowvision.models.**deit_base_patch16_LS_384_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Base-patch16-LS-384 ImageNet21k pretrained model.

Note: DeiT-Base-patch16-LS-384 ImageNet21k pretrained model from "DeiT III: Revenge of the ViT". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_base_patch16_LS_384_in21k = flowvision.models.deit_base_patch16_LS_384_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**deit_huge_patch14_LS_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Huge-patch14-LS-224 model.

Note: DeiT-Huge-patch14-LS-224 model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_huge_patch14_LS_224 = flowvision.models.deit_huge_patch14_LS_
↪224(pretrained=False, progress=True)
```

flowvision.models.**deit_huge_patch14_LS_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Huge-patch14-LS-224 ImageNet21k pretrained model.

Note: DeiT-Huge-patch14-LS-224 ImageNet21k pretrained model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_huge_patch14_LS_224_in21k = flowvision.models.deit_huge_patch14_LS_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**deit_large_patch16_LS_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Large-patch16-LS-224 model.

Note: DeiT-Large-patch16-LS-224 model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_large_patch16_LS_224 = flowvision.models.deit_large_patch16_LS_
↪224(pretrained=False, progress=True)
```

flowvision.models.**deit_large_patch16_LS_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Large-patch16-LS-224 ImageNet21k pretrained model.

Note: DeiT-Large-patch16-LS-224 ImageNet21k pretrained model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_large_patch16_LS_224_in21k = flowvision.models.deit_large_patch16_LS_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**deit_large_patch16_LS_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Large-patch16-LS-384 model.

Note: DeiT-Large-patch16-LS-384 model from "DeiT III: Revenge of the ViT". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_large_patch16_LS_384 = flowvision.models.deit_large_patch16_LS_
↪384(pretrained=False, progress=True)
```

flowvision.models.**deit_large_patch16_LS_384_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Large-patch16-LS-384 ImageNet21k pretrained model.

Note: DeiT-Large-patch16-LS-384 ImageNet21k pretrained model from "DeiT III: Revenge of the ViT". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_large_patch16_LS_384_in21k = flowvision.models.deit_large_patch16_LS_384_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**deit_small_patch16_LS_224**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Small-patch16-LS-224 model.

Note: DeiT-Small-patch16-LS-224 model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_small_patch16_LS_224 = flowvision.models.deit_small_patch16_LS_
↪224(pretrained=False, progress=True)
```

flowvision.models.**deit_small_patch16_LS_224_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Small-patch16-LS-224 ImageNet21k pretrained model.

Note: DeiT-Small-patch16-LS-224 ImageNet21k pretrained model from "DeiT III: Revenge of the ViT". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_small_patch16_LS_224_in21k = flowvision.models.deit_small_patch16_LS_224_
↪in21k(pretrained=False, progress=True)
```

flowvision.models.**deit_small_patch16_LS_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Small-patch16-LS-384 model.

Note: DeiT-Small-patch16-LS-384 model from "DeiT III: Revenge of the ViT". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_small_patch16_LS_384 = flowvision.models.deit_small_patch16_LS_
↪384(pretrained=False, progress=True)
```

flowvision.models.**deit_small_patch16_LS_384_in21k**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the DeiT-Small-patch16-LS-384 ImageNet21k pretrained model.

Note: DeiT-Small-patch16-LS-384 ImageNet21k pretrained model from "DeiT III: Revenge of the ViT". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> deit_small_patch16_LS_384_in21k = flowvision.models.deit_small_patch16_LS_384_
↪in21k(pretrained=False, progress=True)
```

6.1.37 CaiT

`flowvision.models.`**`cait_M36_384`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the CaiT-M36-384 model.

Note: CaiT-M36-384 model from "Going Deeper With Image Transformers". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cait_M36_384 = flowvision.models.cait_M36_384(pretrained=False, progress=True)
```

`flowvision.models.`**`cait_M48_448`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the CaiT-M48-448 model.

Note: CaiT-M48-448 model from "Going Deeper With Image Transformers". The required input size of the model is 448x448.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cait_M48_448 = flowvision.models.cait_M48_448(pretrained=False, progress=True)
```

`flowvision.models.`**`cait_S24_224`**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the CaiT-S24-224 model.

Note: CaiT-S24-224 model from "Going Deeper With Image Transformers". The required input size of the model is 224x224.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cait_S24_224 = flowvision.models.cait_S24_224(pretrained=False, progress=True)
```

flowvision.models.**cait_S24_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the CaiT-S24-384 model.

Note: CaiT-S24-384 model from "Going Deeper With Image Transformers". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cait_S24_384 = flowvision.models.cait_S24_384(pretrained=False, progress=True)
```

flowvision.models.**cait_S36_384**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the CaiT-S36-384 model.

Note: CaiT-S36-384 model from "Going Deeper With Image Transformers". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cait_S36_384 = flowvision.models.cait_S36_384(pretrained=False, progress=True)
```

flowvision.models.**cait_XS24_384**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs the CaiT-XS24-384 model.

Note: CaiT-XS24-384 model from "Going Deeper With Image Transformers". The required input size of the model is 384x384.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> cait_XS24_384 = flowvision.models.cait_XS24_384(pretrained=False,␣
↪progress=True)
```

6.1.38 DLA

flowvision.models.**dla102**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs DLA102 224x224 model trained on ImageNet-1k.

Note: DLA102 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla102 = flowvision.models.dla102(pretrained=False, progress=True)
```

flowvision.models.**dla102x**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs DLA102x 224x224 model trained on ImageNet-1k.

Note: DLA102x 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla102x = flowvision.models.dla102x(pretrained=False, progress=True)
```

flowvision.models.**dla102x2**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs DLA102x2 224x224 model trained on ImageNet-1k.

Note: DLA102x2 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla102x2 = flowvision.models.dla102x2(pretrained=False, progress=True)
```

flowvision.models.**dla169**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs DLA169 224x224 model trained on ImageNet-1k.

Note: DLA169 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla169 = flowvision.models.dla169(pretrained=False, progress=True)
```

flowvision.models.**dla34**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs DLA34 224x224 model trained on ImageNet-1k.

Note: DLA34 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla34 = flowvision.models.dla34(pretrained=False, progress=True)
```

flowvision.models.**dla46_c**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs DLA46_c 224x224 model trained on ImageNet-1k.

Note: DLA46_c 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla46_c = flowvision.models.dla46_c(pretrained=False, progress=True)
```

flowvision.models.**dla46x_c**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs DLA46x_c 224x224 model trained on ImageNet-1k.

Note: DLA46x_c 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla46x_c = flowvision.models.dla46x_c(pretrained=False, progress=True)
```

flowvision.models.**dla60**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs DLA60 224x224 model trained on ImageNet-1k.

Note: DLA60 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla60 = flowvision.models.dla60(pretrained=False, progress=True)
```

flowvision.models.**dla60x**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs DLA60x 224x224 model trained on ImageNet-1k.

Note: DLA60x 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla60x = flowvision.models.dla60x(pretrained=False, progress=True)
```

flowvision.models.**dla60x_c**(*pretrained=False*, *progress=True*, ***kwargs*)
Constructs DLA60x_c 224x224 model trained on ImageNet-1k.

Note: DLA60x_c 224x224 model from "Deep Layer Aggregation".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> dla60x_c = flowvision.models.dla60x_c(pretrained=False, progress=True)
```

6.1.39 GENet

flowvision.models.**genet_large**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs GENet-large 256x256 model pretrained on ImageNet-1k.

Note: GENet-large 256x256 model from "Neural Architecture Design for GPU-Efficient Networks".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: True

For example:

```
>>> import flowvision
>>> genet_large = flowvision.models.genet_large(pretrained=False, progress=True)
```

flowvision.models.**genet_normal**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs GENet-normal 192x192 model pretrained on ImageNet-1k.

Note: GENet-normal 192x192 model from "Neural Architecture Design for GPU-Efficient Networks".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: True

For example:

```
>>> import flowvision
>>> genet_normal = flowvision.models.genet_normal(pretrained=False, progress=True)
```

flowvision.models.**genet_small**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs GENet-small 192x192 model pretrained on ImageNet-1k.

Note: GENet-small 192x192 model from "Neural Architecture Design for GPU-Efficient Networks".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: False
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: True

For example:

```
>>> import flowvision
>>> genet_small = flowvision.models.genet_small(pretrained=False, progress=True)
```

6.1.40 HRNet

flowvision.models.**hrnet_w18**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs HRNet-w18 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w18 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w18 = flowvision.models.hrnet_w18(pretrained=False, progress=True)
```

flowvision.models.**hrnet_w18_small**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs HRNet-w18-small 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w18-small 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w18_small = flowvision.models.hrnet_w18_small(pretrained=False,␣
↪progress=True)
```

flowvision.models.**hrnet_w18_small_v2**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)
Constructs HRNet-w18-small-v2 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w18-small-v2 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w18_small_v2 = flowvision.models.hrnet_w18_small_v2(pretrained=False,␣
↪progress=True)
```

`flowvision.models.`**`hrnet_w30`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs HRNet-w30 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w30 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w30 = flowvision.models.hrnet_w30(pretrained=False, progress=True)
```

`flowvision.models.`**`hrnet_w32`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs HRNet-w32 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w32 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w32 = flowvision.models.hrnet_w32(pretrained=False, progress=True)
```

`flowvision.models.`**`hrnet_w40`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs HRNet-w40 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w40 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w40 = flowvision.models.hrnet_w40(pretrained=False, progress=True)
```

`flowvision.models.`**`hrnet_w44`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs HRNet-w44 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w44 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w44 = flowvision.models.hrnet_w44(pretrained=False, progress=True)
```

`flowvision.models.`**`hrnet_w48`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs HRNet-w48 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w48 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w48 = flowvision.models.hrnet_w48(pretrained=False, progress=True)
```

`flowvision.models.`**`hrnet_w64`**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs HRNet-w64 224x224 model pretrained on ImageNet-1k.

Note: HRNet-w64 224x224 model from "Deep High-Resolution Representation Learning for Visual Recognition".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> hrnet_w64 = flowvision.models.hrnet_w64(pretrained=False, progress=True)
```

6.1.41 FAN

flowvision.models.**fan_base_16_p4_hybrid**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-base 224x224 model pretrained on ImageNet-1k.

Note: FAN-Hybrid-base 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_base_16_p4_hybrid = flowvision.models.fan_base_16_p4_
↪hybrid(pretrained=False, progress=True)
```

flowvision.models.**fan_base_16_p4_hybrid_in22k_1k**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-base 224x224 model pretrained on ImageNet-21k.

Note: FAN-Hybrid-base 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_base_16_p4_hybrid_in22k_1k = flowvision.models.fan_base_16_p4_hybrid_
↪in22k_1k(pretrained=False, progress=True)
```

flowvision.models.**fan_base_16_p4_hybrid_in22k_1k_384**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-base 384x384 model pretrained on ImageNet-21k.

Note: FAN-Hybrid-base 384x384 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_base_16_p4_hybrid_in22k_1k_384 = flowvision.models.fan_base_16_p4_hybrid_
↪in22k_1k_384(pretrained=False, progress=True)
```

flowvision.models.**fan_base_18_p16_224**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-ViT-base 224x224 model pretrained on ImageNet-1k.

Note: FAN-ViT-base 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_base_18_p16_224 = flowvision.models.fan_base_18_p16_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**fan_large_16_p4_hybrid**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-large 224x224 model pretrained on ImageNet-1k.

Note: FAN-Hybrid-large 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_large_16_p4_hybrid = flowvision.models.fan_large_16_p4_
↪hybrid(pretrained=False, progress=True)
```

flowvision.models.**fan_large_16_p4_hybrid_in22k_1k**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-large 224x224 model pretrained on ImageNet-21k.

Note: FAN-Hybrid-large 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_large_16_p4_hybrid_in22k_1k = flowvision.models.fan_large_16_p4_hybrid_
↪in22k_1k(pretrained=False, progress=True)
```

flowvision.models.**fan_large_16_p4_hybrid_in22k_1k_384**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-large 384x384 model pretrained on ImageNet-21k.

Note: FAN-Hybrid-large 384x384 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_large_16_p4_hybrid_in22k_1k_384 = flowvision.models.fan_large_16_p4_
↪hybrid_in22k_1k_384(pretrained=False, progress=True)
```

flowvision.models.**fan_large_24_p16_224**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-ViT-large 224x224 model pretrained on ImageNet-1k.

Note: FAN-ViT-large 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_large_24_p16_224 = flowvision.models.fan_large_24_p16_
↪224(pretrained=False, progress=True)
```

flowvision.models.**fan_small_12_p16_224**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-ViT-small 224x224 model pretrained on ImageNet-1k.

Note: FAN-ViT-small 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_small_12_p16_224 = flowvision.models.fan_small_12_p16_
↪224(pretrained=False, progress=True)
```

flowvision.models.**fan_small_12_p4_hybrid**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-small 224x224 model pretrained on ImageNet-1k.

Note: FAN-Hybrid-small 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_small_12_p4_hybrid = flowvision.models.fan_small_12_p4_
↪hybrid(pretrained=False, progress=True)
```

flowvision.models.**fan_tiny_12_p16_224**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-ViT-tiny 224x224 model pretrained on ImageNet-1k.

Note: FAN-ViT-tiny 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_tiny_12_p16_224 = flowvision.models.fan_tiny_12_p16_224(pretrained=False,␣
↪progress=True)
```

flowvision.models.**fan_tiny_8_p4_hybrid**(*pretrained: bool = False*, *progress: bool = True*, ***kwargs*)

Constructs FAN-Hybrid-tiny 224x224 model pretrained on ImageNet-1k.

Note: FAN-Hybrid-tiny 224x224 model from "Understanding The Robustness in Vision Transformers".

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> fan_tiny_8_p4_hybrid = flowvision.models.fan_tiny_8_p4_
↪hybrid(pretrained=False, progress=True)
```

6.2 Neural Style Transfer

flowvision.models.style_transfer.**fast_neural_style**(*pretrained: bool = False*, *progress: bool = True*, *style_model: str = 'sketch'*, ***kwargs: Any*) → flowvision.models.style_transfer.stylenet.FastNeuralStyle

Constructs the Fast Neural Style Transfer model.

Note: Perceptual Losses for Real-Time Style Transfer and Super-Resolution. The required minimum input size of the model is 256x256. For more details for how to use this model, users can refer to: neural_style_transfer project.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on ImageNet. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`
- **style_model** (*str*) – Which pretrained style model to download, user can choose from [sketch, candy, mosaic, rain_princess, udnie]. Default: `sketch`

For example:

```
>>> import flowvision
>>> stylenet = flowvision.models.style_transfer.fast_neural_style(pretrained=True,
↪ progress=True, style_model = "sketch")
```

6.3 Face Recognition

flowvision.models.face_recognition.**iresnet101**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the IResNet-101 model trained on Glint360K(https://github.com/deepinsight/insightface/tree/master/recognition/partial_fc#4-download).

Note: The required input size of the model is 112x112.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on Glint360K. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> iresnet101 = flowvision.models.face_recognition.iresnet101(pretrained=False,␣
↪progress=True)
```

(continues on next page)

(continued from previous page)

flowvision.models.face_recognition.**iresnet50**(*pretrained=False*, *progress=True*, ***kwargs*)

Constructs the IResNet-50 model trained on webface600K(https://www.face-benchmark.org/download.html).

Note: The required input size of the model is 112x112.

Parameters

- **pretrained** (*bool*) – Whether to download the pre-trained model on webface600K. Default: `False`
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr. Default: `True`

For example:

```
>>> import flowvision
>>> iresnet50 = flowvision.models.face_recognition.iresnet50(pretrained=False,␣
↪progress=True)
```

6.4 Semantic Segmentation

6.4.1 FCN

flowvision.models.segmentation.**fcn_resnet101_coco**(*pretrained=False*, *progress=True*, *num_classes=21*, *aux_loss=None*, ***kwargs*)

Constructs a Fully-Convolutional Network model with a ResNet-101 backbone.

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017 which contains the same classes as Pascal VOC
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **aux_loss** (*bool*) – If True, it uses an auxiliary loss

For example:

```
>>> import flowvision
>>> deeplabv3_mobilenet_v3_large_coco = flowvision.models.segmentation.fcn_
↪resnet101_coco(pretrained=True, progress=True)
```

flowvision.models.segmentation.**fcn_resnet50_coco**(*pretrained=False*, *progress=True*, *num_classes=21*, *aux_loss=None*, ***kwargs*)

Constructs a Fully-Convolutional Network model with a ResNet-50 backbone.

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017 which contains the same classes as Pascal VOC
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **aux_loss** (*bool*) – If True, it uses an auxiliary loss

For example:

```
>>> import flowvision
>>> deeplabv3_mobilenet_v3_large_coco = flowvision.models.segmentation.fcn_
↪resnet50_coco(pretrained=True, progress=True)
```

6.4.2 DeepLabV3

flowvision.models.segmentation.**deeplabv3_mobilenet_v3_large_coco**(*pretrained=False*, *progress=True*, *num_classes=21*, *aux_loss=None*, ***kwargs*)

Constructs a DeepLabV3 model with a MobileNetV3-Large backbone. :param pretrained: If True, returns a model pre-trained on COCO train2017 which

contains the same classes as Pascal VOC

Parameters

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **aux_loss** (*bool*) – If True, it uses an auxiliary loss

For example:

```
>>> import flowvision
>>> deeplabv3_mobilenet_v3_large_coco = flowvision.models.segmentation.deeplabv3_
↪mobilenet_v3_large_coco(pretrained=True, progress=True)
```

flowvision.models.segmentation.**deeplabv3_resnet101_coco**(*pretrained=False*, *progress=True*, *num_classes=21*, *aux_loss=None*, ***kwargs*)

Constructs a DeepLabV3 model with a ResNet-101 backbone. :param pretrained: If True, returns a model pre-trained on COCO train2017 which

contains the same classes as Pascal VOC

Parameters

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – The number of classes
- **aux_loss** (*bool*) – If True, include an auxiliary classifier

For example:

```
>>> import flowvision
>>> deeplabv3_resnet101_coco = flowvision.models.segmentation.deeplabv3_resnet101_
↪coco(pretrained=True, progress=True)
```

flowvision.models.segmentation.**deeplabv3_resnet50_coco**(*pretrained=False*, *progress=True*, *num_classes=21*, *aux_loss=None*, ***kwargs*)

Constructs a DeepLabV3 model with a ResNet-50 backbone. :param pretrained: If True, returns a model pre-trained on COCO train2017 which

contains the same classes as Pascal VOC

Parameters

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **aux_loss** (*bool*) – If True, it uses an auxiliary loss

For example:

```
>>> import flowvision
>>> deeplabv3_resnet50_coco = flowvision.models.segmentation.deeplabv3_resnet50_
↪coco(pretrained=True, progress=True)
```

6.4.3 LRASPP

flowvision.models.segmentation.**lraspp_mobilenet_v3_large_coco**(*pretrained=False*, *progress=True*, *num_classes=21*, ***kwargs*)

Constructs a Lite R-ASPP Network model with a MobileNetV3-Large backbone. :param pretrained: If True, returns a model pre-trained on COCO train2017 which

contains the same classes as Pascal VOC

Parameters

- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)

For example:

```
>>> import flowvision
>>> lraspp_mobilenet_v3_large_coco = flowvision.models.segmentation.lraspp_
↪mobilenet_v3_large_coco(pretrained=True, progress=True)
```

6.5 Object Detection

6.5.1 Faster R-CNN

flowvision.models.detection.**fasterrcnn_mobilenet_v3_large_320_fpn**(*pretrained: bool = False, progress: bool = True, num_classes: Optional[int] = 91, pretrained_backbone: bool = True, trainable_backbone_layers: Optional[int] = None, **kwargs*)

Constructs a low resolution Faster R-CNN model with a MobileNetV3-Large FPN backbone tunned for mobile use-cases. It works similarly to Faster R-CNN with ResNet-50 FPN backbone. See *fasterrcnn_resnet50_fpn()* for more details.

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **pretrained_backbone** (*bool*) – If True, returns a model with backbone pre-trained on Imagenet
- **trainable_backbone_layers** (*int*) – number of trainable (not frozen) resnet layers starting from final block. Valid values are between 0 and 6, with 6 meaning all backbone layers are trainable. If `None` is passed (the default) this value is set to 3.

For example:

```
>>> import flowvision
>>> fasterrcnn_mobilenet_v3_large_320_fpn = flowvision.models.detection.
↪fasterrcnn_mobilenet_v3_large_320_fpn(pretrained=False, progress=True)
```

flowvision.models.detection.**fasterrcnn_mobilenet_v3_large_fpn**(*pretrained: bool = False*, *progress: bool = True*, *num_classes: Optional[int] = 91*, *pretrained_backbone: bool = True*, *trainable_backbone_layers: Optional[int] = None*, ***kwargs*)

Constructs a high resolution Faster R-CNN model with a MobileNetV3-Large FPN backbone. It works similarly to Faster R-CNN with ResNet-50 FPN backbone. See *fasterrcnn_resnet50_fpn()* for more details.

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **pretrained_backbone** (*bool*) – If True, returns a model with backbone pre-trained on Imagenet
- **trainable_backbone_layers** (*int*) – number of trainable (not frozen) resnet layers starting from final block. Valid values are between 0 and 6, with 6 meaning all backbone layers are trainable. If `None` is passed (the default) this value is set to 3.

For example:

```
>>> import flowvision
>>> fasterrcnn_mobilenet_v3_large_fpn = flowvision.models.detection.fasterrcnn_
↪mobilenet_v3_large_fpn(pretrained=False, progress=True)
```

flowvision.models.detection.**fasterrcnn_resnet50_fpn**(*pretrained: bool = False*, *progress: bool = True*, *num_classes: Optional[int] = 91*, *pretrained_backbone: bool = True*, *trainable_backbone_layers: Optional[int] = None*, ***kwargs*)

Constructs a Faster R-CNN model with a ResNet-50-FPN backbone.

Reference: "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks".

The input to the model is expected to be a list of tensors, each of shape `[C, H, W]`, one for each images, and should be in `0-1` range. Different images can have different sizes.

The behavior of the model changes depending if it is in training or evaluation mode.

During training, the model expects both the input tensors, as well as a targets (list of dictionary), containing:

- boxes (`FloatTensor[N, 4]`): the ground-truth boxes in `[x1, y1, x2, y2]` format, with `0 <= x1 < x2 <= W` and `0 <= y1 < y2 <= H`.
- labels (`Int64Tensor[N]`): the class label for each ground-truth box

The model returns a `Dict[Tensor]` during training, containing the classification and regression losses for both the RPN and the R-CNN.

During inference, the model requires only the input tensors, and returns the psot-processed predictions as a `List[Dict[Tensor]]`, one for each input image. The fields of the `Dict` are as follows, where `N` is the number of detections:

- boxes (`FloatTensor[N, 4]`): the predicted boxes in `[x1, y1, x2, y2]` format, with `0 <= x1 < x2 <= W` and `0 <= y1 < y2 <= H`.
- labels (`Int64Tensor[N]`): the predicted labels for each detection
- scores (`Tensor[N]`): the scores of each detection

For more details on the output, you may refer to instance_seg_output.

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **pretrained_backbone** (*bool*) – If True, returns a model with backbone pre-trained on Imagenet
- **trainable_backbone_layers** (*int*) – number of trainable (not frozen) resnet layers starting from final block. Valid values are between 0 and 5, with 5 meaning all backbone layers are trainable. If `None` is passed (the default) this value is set to 3.

For example:

```
>>> import flowvision
>>> fasterrcnn_resnet50_fpn = flowvision.models.detection.fasterrcnn_resnet50_
↪fpn(pretrained=False, progress=True)
```

6.5.2 RetinaNet

`flowvision.models.detection.`**retinanet_resnet50_fpn**(*pretrained: bool = False, progress: bool = True, num_classes: Optional[int] = 91, pretrained_backbone: bool = True, trainable_backbone_layers: Optional[int] = None, **kwargs*)

Constructs a RetinaNet model with a ResNet-50-FPN backbone.

Reference: "Focal Loss for Dense Object Detection".

The input to the model is expected to be a list of tensors, each of shape `[C, H, W]`, one for each image, and should be in `0-1` range. Different images can have different sizes.

The behavior of the model changes depending if it is in training or evaluation mode.

During training, the model expects both the input tensors, as well as a targets (list of dictionary), containing:

- boxes (`FloatTensor[N, 4]`): the ground-truth boxes in `[x1, y1, x2, y2]` format, with `0 <= x1 < x2 <= W` and `0 <= y1 < y2 <= H`.
- labels (`Int64Tensor[N]`): the class label for each ground-truth box

The model returns a `Dict[Tensor]` during training, containing the classification and regression losses.

During inference, the model requires only the input tensors, and returns the post-processed predictions as a `List[Dict[Tensor]]`, one for each input image. The fields of the `Dict` are as follows, where `N` is the number of detections:

- boxes (`FloatTensor[N, 4]`): the predicted boxes in `[x1, y1, x2, y2]` format, with `0 <= x1 < x2 <= W` and `0 <= y1 < y2 <= H`.
- labels (`Int64Tensor[N]`): the predicted labels for each detection
- scores (`Tensor[N]`): the scores of each detection

For more details on the output, you may refer to instance_seg_output.

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **pretrained_backbone** (*bool*) – If True, returns a model with backbone pre-trained on Imagenet
- **trainable_backbone_layers** (*int*) – number of trainable (not frozen) resnet layers starting from final block. Valid values are between 0 and 5, with 5 meaning all backbone layers are trainable.

For example:

```
>>> import flowvision
>>> retinanet_resnet50_fpn = flowvision.models.detection.retinanet_resnet50_
↪fpn(pretrained=False, progress=True)
```

6.5.3 SSD

`flowvision.models.detection.`**`ssd300_vgg16`**(*pretrained: bool = False*, *progress: bool = True*, *num_classes: int = 91*, *pretrained_backbone: bool = True*, *trainable_backbone_layers: Optional[int] = None*, ***kwargs: Any*)

Constructs an SSD model with input size 300x300 and a VGG16 backbone.

Reference: "SSD: Single Shot MultiBox Detector".

The input to the model is expected to be a list of tensors, each of shape [C, H, W], one for each image, and should be in 0-1 range. Different images can have different sizes but they will be resized to a fixed size before passing it to the backbone.

The behavior of the model changes depending if it is in training or evaluation mode.

During training, the model expects both the input tensors, as well as a targets (list of dictionary), containing:

- boxes (`FloatTensor[N, 4]`): the ground-truth boxes in `[x1, y1, x2, y2]` format, with `0 <= x1 < x2 <= W` and `0 <= y1 < y2 <= H`.
- labels (Int64Tensor[N]): the class label for each ground-truth box

The model returns a Dict[Tensor] during training, containing the classification and regression losses.

During inference, the model requires only the input tensors, and returns the post-processed predictions as a List[Dict[Tensor]], one for each input image. The fields of the Dict are as follows, where `N` is the number of detections:

- boxes (`FloatTensor[N, 4]`): the predicted boxes in `[x1, y1, x2, y2]` format, with `0 <= x1 < x2 <= W` and `0 <= y1 < y2 <= H`.
- labels (Int64Tensor[N]): the predicted labels for each detection

- scores (Tensor[N]): the scores for each detection

Example:

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **pretrained_backbone** (*bool*) – If True, returns a model with backbone pre-trained on Imagenet
- **trainable_backbone_layers** (*int*) – number of trainable (not frozen) resnet layers starting from final block. Valid values are between 0 and 5, with 5 meaning all backbone layers are trainable.

For example:

```
>>> import flowvision
>>> ssd300_vgg16 = flowvision.models.detection.ssd300_vgg16(pretrained=False,␣
↪progress=True)
```

6.5.4 SSDLite

flowvision.models.detection.**ssdlite320_mobilenet_v3_large**(*pretrained: bool = False*, *progress: bool = True*, *num_classes: int = 91*, *pretrained_backbone: bool = False*, *trainable_backbone_layers: Optional[int] = None*, *norm_layer: Optional[Callable[[...], oneflow.nn.modules.module.Module]] = None*, ***kwargs: Any*)

Constructs an SSDlite model with input size 320x320 and a MobileNetV3 Large backbone, as described at "Searching for MobileNetV3" and "MobileNetV2: Inverted Residuals and Linear Bottlenecks".

See *ssd300_vgg16()* for more details.

Parameters

- **pretrained** (*bool*) – If True, returns a model pre-trained on COCO train2017
- **progress** (*bool*) – If True, displays a progress bar of the download to stderr
- **num_classes** (*int*) – number of output classes of the model (including the background)
- **pretrained_backbone** (*bool*) – If True, returns a model with backbone pre-trained on Imagenet
- **trainable_backbone_layers** (*int*) – number of trainable (not frozen) resnet layers starting from final block. Valid values are between 0 and 6, with 6 meaning all backbone layers are trainable.

- **norm_layer** (*callable, optional*) – Module specifying the normalization layer to use.

For example:

```
>>> import flowvision
>>> ssdlite320_mobilenet_v3_large = flowvision.models.detection.ssdlite320_
↪mobilenet_v3_large(pretrained=False, progress=True)
```

CHAPTER

SEVEN

FLOWVISION.SCHEDULER

class flowvision.scheduler.**Scheduler**(*optimizer: oneflow.optim.optimizer.Optimizer*, *param_group_field: str*, *noise_range_t=None*, *noise_type='normal'*, *noise_pct=0.67*, *noise_std=1.0*, *noise_seed=None*, *initialize: bool = True*)

Parameter Scheduler Base Class Borrowed from pytorch-image-models A scheduler base class that can be used to schedule any optimizer parameter groups. Unlike the builtin PyTorch schedulers, this is intended to be consistently called

- At the END of each epoch, before incrementing the epoch count, to calculate next epoch's value
- At the END of each optimizer update, after incrementing the update count, to calculate next update's value

The schedulers built on this should try to remain as stateless as possible (for simplicity). This family of schedulers is attempting to avoid the confusion of the meaning of 'last_epoch' and -1 values for special behaviour. All epoch and update counts must be tracked in the training code and explicitly passed in to the schedulers on the corresponding step or step_update call. Based on ideas from:

- https://github.com/pytorch/fairseq/tree/master/fairseq/optim/lr_scheduler
- https://github.com/allenai/allennlp/tree/master/allennlp/training/learning_rate_schedulers
- https://github.com/rwightman/pytorch-image-models/tree/master/timm/scheduler

class flowvision.scheduler.**CosineLRScheduler**(*optimizer: oneflow.optim.optimizer.Optimizer*, *t_initial: int*, *t_mul: float = 1.0*, *lr_min: float = 0.0*, *decay_rate: float = 1.0*, *warmup_t=0*, *warmup_lr_init=0*, *warmup_prefix=False*, *cycle_limit=0*, *t_in_epochs=True*, *noise_range_t=None*, *noise_pct=0.67*, *noise_std=1.0*, *noise_seed=42*)

Cosine decay with restarts borrowed from timm. This is described in the paper https://arxiv.org/abs/1608.03983.

Inspiration from https://github.com/allenai/allennlp/blob/master/allennlp/training/learning_rate_schedulers/cosine.py

Parameters

- **optimizer** – The optimizer will be used for the training process
- **t_initial** – The initial number of epochs. Example, 50, 100 etc.
- **t_mul** – updates the SGDR schedule annealing.
- **lr_min** – Defaults to 1e-5. The minimum learning rate to use during the scheduling. The learning rate does not ever go below this value.

- **decay_rate** – When decay rate > 0 and < 1., at every restart the learning rate is decayed by new learning rate which equals `lr * decay_rate`. If decay_rate=0.5, then in that case, the new learning rate becomes half the initial lr.
- **warmup_t** – Defines the number of warmup epochs.
- **warmup_lr_init** – The initial learning rate during warmup.

class flowvision.scheduler.**LinearLRScheduler**(*optimizer: oneflow.optim.optimizer.Optimizer*, *t_initial: int*, *lr_min_rate: float*, *warmup_t=0*, *warmup_lr_init=0.0*, *t_in_epochs=True*, *noise_range_t=None*, *noise_pct=0.67*, *noise_std=1.0*, *noise_seed=42*, *initialize=True*)

Linear warmup and linear decay scheduler

Inspiration from https://github.com/microsoft/Swin-Transformer/blob/main/lr_scheduler.py

Parameters

- **optimizer** – The optimizer will be used for the training process
- **t_initial** – The initial number of epochs. Example, 50, 100 etc.
- **t_mul** – updates the SGDR schedule annealing.
- **lr_min_rate** – The minimum learning rate factor to use during the scheduling. The learning rate does not ever go below to `lr * lr_min_rate`.
- **warmup_t** – Defines the number of warmup epochs.
- **warmup_lr_init** – The initial learning rate during warmup.

class flowvision.scheduler.**StepLRScheduler**(*optimizer: oneflow.optim.optimizer.Optimizer*, *decay_t: float*, *decay_rate: float = 1.0*, *warmup_t=0*, *warmup_lr_init=0*, *t_in_epochs=True*, *noise_range_t=None*, *noise_pct=0.67*, *noise_std=1.0*, *noise_seed=42*, *initialize=True*)

Step LRScheduler Decays the learning rate of each parameter group by decay_rate every decay_t steps.

Parameters

- **optimizer** – The optimizer will be used for the training process
- **decay_t** – Period of learning rate decay.
- **decay_rate** – Multiplicative factor of learning rate decay. Default: 1.0.
- **warmup_t** – Defines the number of warmup epochs.
- **warmup_lr_init** – The initial learning rate during warmup.

class flowvision.scheduler.**MultiStepLRScheduler**(*optimizer: oneflow.optim.optimizer.Optimizer*, *decay_t: List[int]*, *decay_rate: float = 1.0*, *warmup_t=0*, *warmup_lr_init=0*, *t_in_epochs=True*, *noise_range_t=None*, *noise_pct=0.67*, *noise_std=1.0*, *noise_seed=42*, *initialize=True*)

MultiStep LRScheduler Decays the learning rate of each parameter group by decay_rate once the number of step reaches one of the decay_t.

Parameters

- **optimizer** – The optimizer will be used for the training process
- **decay_t** – List of epoch indices. Must be increasing.
- **decay_rate** – Multiplicative factor of learning rate decay. Default: 1.0.
- **warmup_t** – Defines the number of warmup epochs.
- **warmup_lr_init** – The initial learning rate during warmup.

class flowvision.scheduler.**PolyLRScheduler**(*optimizer: oneflow.optim.optimizer.Optimizer*, *t_initial: int*, *power: float = 0.5*, *lr_min: float = 0.0*, *cycle_mul: float = 1.0*, *cycle_decay: float = 1.0*, *cycle_limit: int = 1*, *warmup_t=0*, *warmup_lr_init=0*, *warmup_prefix=False*, *t_in_epochs=True*, *noise_range_t=None*, *noise_pct=0.67*, *noise_std=1.0*, *noise_seed=42*, *k_decay=1.0*, *initialize=True*)

Polynomial LR Scheduler w/ warmup, noise, and k-decay k-decay option based on *k-decay: A New Method For Learning Rate Schedule* - https://arxiv.org/abs/2004.05909

Parameters

- **optimizer** – The optimizer will be used for the training process
- **t_initial** – The initial number of epochs. Example, 50, 100 etc.
- **power** – The power of polynomial. Defaults to 0.5.
- **lr_min** – Defaults to 1e-5. The minimum learning rate to use during
- **scheduling. The learning rate does not ever go below this value.** (*the*) –
- **warmup_t** – Defines the number of warmup epochs.
- **warmup_lr_init** – The initial learning rate during warmup.

class flowvision.scheduler.**TanhLRScheduler**(*optimizer: oneflow.optim.optimizer.Optimizer*, *t_initial: int*, *lb: float = - 7.0*, *ub: float = 3.0*, *lr_min: float = 0.0*, *cycle_mul: float = 1.0*, *cycle_decay: float = 1.0*, *cycle_limit: int = 1*, *warmup_t=0*, *warmup_lr_init=0*, *warmup_prefix=False*, *t_in_epochs=True*, *noise_range_t=None*, *noise_pct=0.67*, *noise_std=1.0*, *noise_seed=42*, *initialize=True*)

Hyberbolic-Tangent decay with restarts. This is described in the paper https://arxiv.org/abs/1806.01593

CHAPTER

EIGHT

FLOWVISION.TRANSFORMS

8.1 Utils for Image Transforms

class flowvision.transforms.**CenterCrop**(*size*)

Crops the given image at the center. If the image is oneflow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions. If image size is smaller than output size along any edge, image is padded with 0 and then center cropped.

Parameters **size** (*sequence or int*) – Desired output size of the crop. If size is an int instead of sequence like (h, w), a square crop (size, size) is made. If provided a sequence of length 1, it will be interpreted as (size[0], size[0]).

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be cropped.

Returns Cropped image.

Return type PIL Image or Tensor

class flowvision.transforms.**Compose**(*transforms*)

Composes several transforms together. Please, see the note below. :param transforms: list of transforms to compose. :type transforms: list of `Transform` objects

Example

```
>>> transforms.Compose([
>>>     transforms.CenterCrop(10),
>>>     transforms.ToTensor(),
>>> ])
```

Note: In order to script the transformations, please use `flow.nn.Sequential` as below. >>> transforms = flow.nn.Sequential(>>> transforms.CenterCrop(10), >>> transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)), >>>) Make sure to use only scriptable transformations, i.e. that work with `flow.Tensor`, does not require *lambda* functions or `PIL.Image`.

class flowvision.transforms.**ConvertImageDtype**(*dtype: oneflow._oneflow_internal.dtype*)

Convert a tensor image to the given `dtype` and scale the values accordingly This function does not support PIL Image.

Parameters **dtype** (*flow.dtype*) – Desired data type of the output

Note: When converting from a smaller to a larger integer `dtype` the maximum values are **not** mapped exactly. If converted back and forth, this mismatch has no effect.

Raises **RuntimeError** – When trying to cast `flow.float32` to `flow.int32` or `flow.int64` as well as for trying to cast `flow.float64` to `flow.int64`. These conversions might lead to overflow errors since the floating point `dtype` cannot store consecutive integers over the whole range of the integer `dtype`.

class flowvision.transforms.**FiveCrop**(*size*)

Crop the given image into four corners and the central crop. If the image is flow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions

Note: This transform returns a tuple of images and there may be a mismatch in the number of inputs and targets your Dataset returns. See below for an example of how to deal with this.

Parameters **size** (*sequence or int*) – Desired output size of the crop. If size is an `int` instead of sequence like (h, w), a square crop of size (size, size) is made. If provided a sequence of length 1, it will be interpreted as (size[0], size[0]).

Example

```
>>> transform = Compose([
>>>     FiveCrop(size), # this is a list of PIL Images
>>>     Lambda(lambda crops: flow.stack([ToTensor()(crop) for crop in crops])) #␣
↪returns a 4D tensor
>>> ])
>>> #In your test loop you can do the following:
>>> input, target = batch # input is a 5d tensor, target is 2d
>>> bs, ncrops, c, h, w = input.size()
>>> result = model(input.view(-1, c, h, w)) # fuse batch size and ncrops
>>> result_avg = result.view(bs, ncrops, -1).mean(1) # avg over crops
```

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be cropped.

Returns tuple of 5 images. Image can be PIL Image or Tensor

class flowvision.transforms.**GaussianBlur**(*kernel_size*, *sigma=(0.1, 2.0)*)

Blurs image with randomly chosen Gaussian blur. If the image is oneflow Tensor, it is expected to have [..., C, H, W] shape, where ... means an arbitrary number of leading dimensions.

Parameters

- **kernel_size** (*int or sequence*) – Size of the Gaussian kernel.
- **sigma** (*float or tuple of float (min, max)*) – Standard deviation to be used for creating kernel to perform blurring. If float, sigma is fixed. If it is tuple of float (min, max), sigma is chosen uniformly at random to lie in the given range.

Returns Gaussian blurred version of the input image.

Return type PIL Image or Tensor

forward(*img: oneflow.Tensor*) → oneflow.Tensor

Parameters **img** (*PIL Image or Tensor*) – image to be blurred.

Returns Gaussian blurred image

Return type PIL Image or Tensor

static get_params(*sigma_min: float*, *sigma_max: float*) → float
Choose sigma for random gaussian blurring.

Parameters

- **sigma_min** (*float*) – Minimum standard deviation that can be chosen for blurring kernel.
- **sigma_max** (*float*) – Maximum standard deviation that can be chosen for blurring kernel.

Returns Standard deviation to be passed to calculate kernel for gaussian blurring.

Return type float

class flowvision.transforms.**Grayscale**(*num_output_channels=1*)
Convert image to grayscale. If the image is oneflow Tensor, it is expected to have [..., 3, H, W] shape, where ... means an arbitrary number of leading dimensions

Parameters **num_output_channels** (*int*) – (1 or 3) number of channels desired for output image

Returns Grayscale version of the input. - If `num_output_channels == 1` : returned image is single channel - If `num_output_channels == 3` : returned image is 3 channel with r == g == b

Return type PIL Image

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be converted to grayscale.

Returns Grayscaled image.

Return type PIL Image or Tensor

class flowvision.transforms.**InterpolationMode**(*value*)
Interpolation modes

class flowvision.transforms.**Lambda**(*lambd*)
Apply a user-defined lambda as a transform.

Parameters **lambd** (*function*) – Lambda/function to be used for transform.

class flowvision.transforms.**Normalize**(*mean*, *std*, *inplace=False*)
Normalize a tensor image with mean and standard deviation. This transform does not support PIL Image. Given mean: `(mean[1],...,mean[n])` and std: `(std[1],..,std[n])` for `n` channels, this transform will normalize each channel of the input `flow.*Tensor` i.e., `output[channel] = (input[channel] - mean[channel]) / std[channel]`

Note: This transform acts out of place, i.e., it does not mutate the input tensor.

Parameters

- **mean** (*sequence*) – Sequence of means for each channel.
- **std** (*sequence*) – Sequence of standard deviations for each channel.

- **inplace** (*bool, optional*) – Bool to make this operation in-place.

forward(*tensor: oneflow.Tensor*) → oneflow.Tensor

Parameters **tensor** (*Tensor*) – Tensor image to be normalized.

Returns Normalized Tensor image.

Return type Tensor

class flowvision.transforms.**PILToTensor**

Convert a PIL Image to a tensor of the same type

Converts a PIL Image (H x W x C) to a Tensor of shape (C x H x W).

class flowvision.transforms.**Pad**(*padding*, *fill=0*, *padding_mode='constant'*)

Pad the given image on all sides with the given "pad" value. If the image is oneflow Tensor, it is expected to have [..., H, W] shape, where ... means at most 2 leading dimensions for mode reflect and symmetric, at most 3 leading dimensions for mode edge, and an arbitrary number of leading dimensions for mode constant

Parameters

- **padding** (*int or sequence*) – Padding on each border. If a single int is provided this is used to pad all borders. If sequence of length 2 is provided this is the padding on left/right and top/bottom respectively. If a sequence of length 4 is provided this is the padding for the left, top, right and bottom borders respectively.
- **fill** (*number or str or tuple*) – Pixel fill value for constant fill. Default is 0. If a tuple of length 3, it is used to fill R, G, B channels respectively. This value is only used when the padding_mode is constant. Only number is supported for oneflow Tensor. Only int or str or tuple value is supported for PIL Image.
- **padding_mode** (*str*) – Type of padding. Should be: constant, edge, reflect or symmetric. Default is constant.
  - constant: pads with a constant value, this value is specified with fill
  - edge: pads with the last value at the edge of the image. If input a 5D oneflow Tensor, the last 3 dimensions will be padded instead of the last 2
  - reflect: pads with reflection of image without repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in reflect mode will result in [3, 2, 1, 2, 3, 4, 3, 2]
  - symmetric: pads with reflection of image repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in symmetric mode will result in [2, 1, 1, 2, 3, 4, 4, 3]

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be padded.

Returns Padded image.

Return type PIL Image or Tensor

class flowvision.transforms.**RandomApply**(*transforms*, *p=0.5*)

Apply randomly a list of transformations with a given probability.

Note: In order to script the transformation, please use flow.nn.ModuleList as input instead of list/tuple of transforms as shown below:

```
>>> transforms = transforms.RandomApply(flow.nn.ModuleList([
>>>     transforms.ColorJitter(),
>>> ]), p=0.3)
```

Make sure to use only scriptable transformations, i.e. that work with `flow.Tensor`, does not require *lambda* functions or `PIL.Image`.

Parameters

- **transforms** (*sequence or Module*) – list of transformations
- **p** (*float*) – probability

class flowvision.transforms.**RandomChoice**(*transforms*)

Apply single transformation randomly picked from a list.

class flowvision.transforms.**RandomCrop**(*size*, *padding=None*, *pad_if_needed=False*, *fill=0*, *padding_mode='constant'*)

Crop the given image at a random location. If the image is oneflow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions, but if non-constant padding is used, the input is expected to have at most 2 leading dimensions

Parameters

- **size** (*sequence or int*) – Desired output size of the crop. If size is an int instead of sequence like (h, w), a square crop (size, size) is made. If provided a sequence of length 1, it will be interpreted as (size[0], size[0]).
- **padding** (*int or sequence, optional*) – Optional padding on each border of the image. Default is None. If a single int is provided this is used to pad all borders. If sequence of length 2 is provided this is the padding on left/right and top/bottom respectively. If a sequence of length 4 is provided this is the padding for the left, top, right and bottom borders respectively.
- **pad_if_needed** (*boolean*) – It will pad the image if smaller than the desired size to avoid raising an exception. Since cropping is done after padding, the padding seems to be done at a random offset.
- **fill** (*number or str or tuple*) – Pixel fill value for constant fill. Default is 0. If a tuple of length 3, it is used to fill R, G, B channels respectively. This value is only used when the padding_mode is constant. Only number is supported for flow Tensor. Only int or str or tuple value is supported for PIL Image.
- **padding_mode** (*str*) – Type of padding. Should be: constant, edge, reflect or symmetric. Default is constant.
  - constant: pads with a constant value, this value is specified with fill
  - edge: pads with the last value at the edge of the image. If input a 5D flow Tensor, the last 3 dimensions will be padded instead of the last 2
  - reflect: pads with reflection of image without repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in reflect mode will result in [3, 2, 1, 2, 3, 4, 3, 2]
  - symmetric: pads with reflection of image repeating the last value on the edge. For example, padding [1, 2, 3, 4] with 2 elements on both sides in symmetric mode will result in [2, 1, 1, 2, 3, 4, 4, 3]

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be cropped.

Returns Cropped image.

Return type PIL Image or Tensor

static get_params(*img: oneflow.Tensor*, *output_size: Tuple[int, int]*) → Tuple[int, int, int, int]
Get parameters for `crop` for a random crop.

Parameters

- **img** (*PIL Image or Tensor*) – Image to be cropped.
- **output_size** (*tuple*) – Expected output size of the crop.

Returns params (i, j, h, w) to be passed to `crop` for random crop.

Return type tuple

class flowvision.transforms.**RandomGrayscale**(*p=0.1*)
Randomly convert image to grayscale with a probability of p (default 0.1). If the image is flow Tensor, it is expected to have [..., 3, H, W] shape, where ... means an arbitrary number of leading dimensions

Parameters **p** (*float*) – probability that image should be converted to grayscale.

Returns Grayscale version of the input image with probability p and unchanged with probability (1-p). - If input image is 1 channel: grayscale version is 1 channel - If input image is 3 channel: grayscale version is 3 channel with r == g == b

Return type PIL Image or Tensor

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be converted to grayscale.

Returns Randomly grayscaled image.

Return type PIL Image or Tensor

class flowvision.transforms.**RandomHorizontalFlip**(*p=0.5*)
Horizontally flip the given image randomly with a given probability. If the image is flow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions

Parameters **p** (*float*) – probability of the image being flipped. Default value is 0.5

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be flipped.

Returns Randomly flipped image.

Return type PIL Image or Tensor

class flowvision.transforms.**RandomOrder**(*transforms*)
Apply a list of transformations in a random order.

class flowvision.transforms.**RandomResizedCrop**(*size*, *scale=(0.08, 1.0)*, *ratio=(0.75, 1.3333333333333333)*, *interpolation=<InterpolationMode.BILINEAR: 'bilinear'>*)
Crop a random portion of image and resize it to a given size.

If the image is flow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions

A crop of the original image is made: the crop has a random area (H * W) and a random aspect ratio. This crop is finally resized to the given size. This is popularly used to train the Inception networks.

Parameters

- **size** (*int or sequence*) – expected output size of the crop, for each edge. If size is an int instead of sequence like (h, w), a square output size `(size, size)` is made. If provided a sequence of length 1, it will be interpreted as (size[0], size[0]).
- **scale** (*tuple of float*) – Specifies the lower and upper bounds for the random area of the crop, before resizing. The scale is defined with respect to the area of the original image.
- **ratio** (*tuple of float*) – lower and upper bounds for the random aspect ratio of the crop, before resizing.
- **interpolation** (`InterpolationMode`) – Desired interpolation enum defined by *flowvision.transforms.InterpolationMode*. Default is `InterpolationMode.BILINEAR`. If input is Tensor, only `InterpolationMode.NEAREST`, `InterpolationMode.BILINEAR` and `InterpolationMode.BICUBIC` are supported. For backward compatibility integer values (e.g. `PIL.Image.NEAREST`) are still acceptable.

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be cropped and resized.

Returns Randomly cropped and resized image.

Return type PIL Image or Tensor

static get_params(*img: oneflow.Tensor*, *scale: List[float]*, *ratio: List[float]*) → Tuple[int, int, int, int]

Get parameters for `crop` for a random sized crop.

Parameters

- **img** (*PIL Image or Tensor*) – Input image.
- **scale** (*list*) – range of scale of the origin size cropped
- **ratio** (*list*) – range of aspect ratio of the origin aspect ratio cropped

Returns params (i, j, h, w) to be passed to `crop` for a random sized crop.

Return type tuple

class flowvision.transforms.**RandomSizedCrop**(**args*, ***kwargs*)

Note: This transform is deprecated in favor of RandomResizedCrop.

class flowvision.transforms.**RandomTransforms**(*transforms*)

Base class for a list of transformations with randomness

Parameters **transforms** (*sequence*) – list of transformations

class flowvision.transforms.**RandomVerticalFlip**(*p=0.5*)

Vertically flip the given image randomly with a given probability. If the image is flow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions

Parameters **p** (*float*) – probability of the image being flipped. Default value is 0.5

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be flipped.

Returns Randomly flipped image.

Return type PIL Image or Tensor

class flowvision.transforms.**Resize**(*size*, *interpolation=<InterpolationMode.BILINEAR: 'bilinear'>*)

Resize the input image to the given size. If the image is oneflow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions

Parameters

- **size** (*sequence or int*) – Desired output size. If size is a sequence like (h, w), output size will be matched to this. If size is an int, smaller edge of the image will be matched to this number. i.e, if height > width, then image will be rescaled to (size * height / width, size).
- **interpolation** (InterpolationMode) – Desired interpolation enum defined by *flowvision.transforms.InterpolationMode*. Default is `InterpolationMode.BILINEAR`. If input is Tensor, only `InterpolationMode.NEAREST`, `InterpolationMode.BILINEAR` and `InterpolationMode.BICUBIC` are supported. For backward compatibility integer values (e.g. `PIL.Image.NEAREST`) are still acceptable.

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be scaled.

Returns Rescaled image.

Return type PIL Image or Tensor

class flowvision.transforms.**Scale**(**args*, ***kwargs*)

Note: This transform is deprecated in favor of Resize.

class flowvision.transforms.**Solarization**(*p=0.1*)

Apply Solarization to the input PIL Image.

Parameters **p** (*float*) – probability that image should be applied with solarization operation.

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be applied with solarization operation.

Returns selorization image.

Return type PIL Image or Tensor

class flowvision.transforms.**TenCrop**(*size*, *vertical_flip=False*)

Crop the given image into four corners and the central crop plus the flipped version of these (horizontal flipping is used by default). If the image is flow Tensor, it is expected to have [..., H, W] shape, where ... means an arbitrary number of leading dimensions

---

Note: This transform returns a tuple of images and there may be a mismatch in the number of inputs and targets your Dataset returns. See below for an example of how to deal with this.

Parameters

- **size** (*sequence or int*) – Desired output size of the crop. If size is an int instead of sequence like (h, w), a square crop (size, size) is made. If provided a sequence of length 1, it will be interpreted as (size[0], size[0]).
- **vertical_flip** (*bool*) – Use vertical flipping instead of horizontal

Example

```
>>> transform = Compose([
>>>     TenCrop(size), # this is a list of PIL Images
>>>     Lambda(lambda crops: flow.stack([ToTensor()(crop) for crop in crops])) #␣
↪returns a 4D tensor
>>> ])
>>> #In your test loop you can do the following:
>>> input, target = batch # input is a 5d tensor, target is 2d
>>> bs, ncrops, c, h, w = input.size()
>>> result = model(input.view(-1, c, h, w)) # fuse batch size and ncrops
>>> result_avg = result.view(bs, ncrops, -1).mean(1) # avg over crops
```

forward(*img*)

Parameters **img** (*PIL Image or Tensor*) – Image to be cropped.

Returns tuple of 10 images. Image can be PIL Image or Tensor

class flowvision.transforms.**ToPILImage**(*mode=None*)

Convert a tensor or an ndarray to PIL Image.

Converts a flow.Tensor of shape C x H x W or a numpy ndarray of shape H x W x C to a PIL Image while preserving the value range.

Parameters **mode** (PIL.Image mode) – color space and pixel depth of input data (optional). If `mode` is `None` (default) there are some assumptions made about the input data: - If the input has 4 channels, the `mode` is assumed to be `RGBA`. - If the input has 3 channels, the `mode` is assumed to be `RGB`. - If the input has 2 channels, the `mode` is assumed to be `LA`. - If the input has 1 channel, the `mode` is determined by the data type (i.e `int`, `float`, `short`).

class flowvision.transforms.**ToTensor**

Convert a `PIL Image` or `numpy.ndarray` to tensor.

Converts a PIL Image or numpy.ndarray (H x W x C) in the range [0, 255] to a flow.FloatTensor of shape (C x H x W) in the range [0.0, 1.0] if the PIL Image belongs to one of the modes (L, LA, P, I, F, RGB, YCbCr, RGBA, CMYK, 1) or if the numpy.ndarray has dtype = np.uint8 In the other cases, tensors are returned without scaling.

Note: Because the input image is scaled to [0.0, 1.0], this transformation should not be used when transforming target image masks.

CHAPTER

NINE

FLOWVISION.UTILS

Useful utils for deep learning tasks

class flowvision.utils.**AverageMeter**
Computes and stores the average and current value

class flowvision.utils.**ModelEmaV2**(*model*, *decay=0.9999*, *device=None*)
Model Exponential Moving Average V2 borrowed from: https://github.com/rwightman/pytorch-image-models/blob/master/timm/utils/model_ema.py

Keep a moving average of everything in the model state_dict (parameters and buffers). V2 of this module is simpler, it does not match params/buffers based on name but simply iterates in order.

This is intended to allow functionality like https://www.tensorflow.org/api_docs/python/tf/train/ExponentialMovingAverage

A smoothed version of the weights is necessary for some training schemes to perform well. E.g. Google's hyper-params for training MNASNet, MobileNet-V3, EfficientNet, etc that use RMSprop with a short 2.4-3 epoch decay period and slow LR decay rate of .96-.99 requires EMA smoothing of weights to match results. Pay attention to the decay constant you are using relative to your update count per epoch.

To keep EMA from using GPU resources, set device='cpu'. This will save a bit of memory but disable validation of the EMA weights. Validation will have to be done manually in a separate process, or after the training stops converging.

This class is sensitive where it is initialized in the sequence of model init, GPU assignment and distributed training wrappers.

flowvision.utils.**accuracy**(*output*, *target*, *topk=(1)*)
Computes the accuracy over the k top predictions for the specified values of k

flowvision.utils.**dispatch_clip_grad**(*parameters*, *value: float*, *mode: str = 'norm'*, *norm_type: float = 2.0*)
Dispatch to gradient clipping method

Parameters

- **parameters** (*Iterable*) – model parameters to clip
- **value** (*float*) – clipping value/factor/norm, mode dependant
- **mode** (*str*) – clipping mode, one of 'norm', 'value', 'agc'
- **norm_type** (*float*) – p-norm, default 2.0

flowvision.utils.**make_grid**(*tensor: Union[oneflow.Tensor, List[oneflow.Tensor]]*, *nrow: int = 8*, *padding: int = 2*, *normalize: bool = False*, *range: Optional[Tuple[int, int]] = None*, *scale_each: bool = False*, *pad_value: int = 0*) → oneflow.Tensor
Make a grid of images.

Parameters

- **tensor** (*Tensor or list*) – 4D mini-batch Tensor of shape (B x C x H x W) or a list of images all of the same size.
- **nrow** (*int, optional*) – Number of images displayed in each row of the grid. The final grid size is `(B / nrow, nrow)`. Default: `8`.
- **padding** (*int, optional*) – amount of padding. Default: `2`.
- **normalize** (*bool, optional*) – If True, shift the image to the range (0, 1), by the min and max values specified by `range`. Default: `False`.
- **range** (*tuple, optional*) – tuple (min, max) where min and max are numbers, then these numbers are used to normalize the image. By default, min and max are computed from the tensor.
- **scale_each** (*bool, optional*) – If `True`, scale each image in the batch of images separately rather than the (min, max) over all images. Default: `False`.
- **pad_value** (*float, optional*) – Value for the padded pixels. Default: `0`.

Example: See this notebook here

flowvision.utils.**save_image**(*tensor: Union[oneflow.Tensor, List[oneflow.Tensor]]*, *fp: Union[str, pathlib.Path, BinaryIO]*, *nrow: int = 8*, *padding: int = 2*, *normalize: bool = False*, *range: Optional[Tuple[int, int]] = None*, *scale_each: bool = False*, *pad_value: int = 0*, *format: Optional[str] = None*) → None

Save a given Tensor into an image file.

Parameters

- **tensor** (*Tensor or list*) – Image to be saved. If given a mini-batch tensor, saves the tensor as a grid of images by calling `make_grid`.
- **fp** (*string or file object*) – A filename or a file object
- **format** (*Optional*) – If omitted, the format to use is determined from the filename extension. If a file object was used instead of a filename, this parameter should always be used.
- ****kwargs** – Other arguments are documented in `make_grid`.

CHAPTER

TEN

CHANGELOG

10.1 V0.2.1

New Features

- Support `transforms.Grayscale` and `transforms.Solarization` transforms #220
- Support TorchHub functionality #258
- Support `transforms.RandomAffine` method #259
- Support `transforms.RandomRotation` method #261

Bug Fixes

- fix hsv2rgb bug #256
- fix rgb2hsv bug #257

Improvements

- refine TODO items and align transforms.ColorJitter to torchvision(0.13.1):#264

Docs Update

Contributors A total of 3 developers contributed to this release. Thanks @BBuf, @rentainhe, @Flowingsun007

10.2 V0.2.0

New Features

- Support SENet model and pretrained weight #149
- Support ResNeSt model and pretrained weight #156
- Support PoolFormer model and pretrained weight #137
- Support RegionViT model and pretrained weight #144
- Support UniFormer model and pretrained weight #147

- Support `IResNet` model for face recognition #160
- Support VAN model and pretrained weight #166
- Support Dynamic convolution module #166
- Support `transforms.RandomGrayscale` method #171
- Support RegNet model and pretrained weight #166
- Support LeViT model and pretrained weight #177
- Support `transforms.GaussianBlur` method #188
- Support `SUN397`,`Country211`, dataset #215
- Support `Flowers102`,`FGVCAircraft`,`OxfordIIITPet`,`DTD`,`Food101`,`RenderedSST2`,`StanfordCars`,`PCAM`,`Euro`
  dataset #217
- Support MobileViT model and pretrained weight #231
- Support DeiT III model and pretrained weight #239
- Support CaiT model and pretrained weight #239
- Support DLA model and pretrained weight #239
- Support GENet model and pretrained weight #250
- Support HRNet model and pretrained weight #250
- Support FAN model and pretrained weight #250

Bug Fixes

- Fix benchmark normalize mode error #146

Improvements

Docs Update

- Add `SEModule` Docs #143
- Add `Scheduler` and `Data` Docs #189

Contributors A total of x developers contributed to this release.

10.3 v0.1.0 (10/02/2022)

New Features

- Support `trunc_normal_` in `flowvision.layers.weight_init` #92
- Support DeiT model #115
- Support `PolyLRScheduler` and `TanhLRScheduler` in `flowvision.scheduler` #85
- Add `resmlp_12_224_dino` model and pretrained weight #128
- Support ConvNeXt model #93
- Add ReXNet weights #132

Bug Fixes

- Fix `F.normalize` usage in SSD #116
- Fix bug in `EfficientNet` and `Res2Net` #122

- Fix error pretrained weight usage in `vit_small_patch32_384` and `res2net50_48w_2s` #128

Improvements

- Refator `trunc_normal_` and `linspace` usage in Swin-T, Cross-Former, PVT and CSWin models #100
- Refator `Vision Transformer` model #115
- Refine `flowvision.models.ModelCreator` to support `ModelCreator.model_list` func #123
- Refator README #124
- Refine `load_state_dict_from_url` in `flowvision.models.utils` to support downloading pretrained weights to cache dir `~/.oneflow/flowvision_cache` #127
- Rebuild a cleaner model zoo and test all the model with pretrained weights released in flowvision #128

Docs Update

- Update `Vision Transformer` docs #115
- Add `Getting Started` docs #124
- Add `resmlp_12_224_dino` docs #128
- Fix `VGG` docs bug #128
- Add `ConvNeXt` docs #93

Contributors

A total of 5 developers contributed to this release. Thanks @rentainhe, @simonJJJ, @kaijieshi7, @lixiang007666, @Ldpe2G

PYTHON MODULE INDEX

f

INDEX

D

E

F

G

H

I

L

M

N

P

R

S

T

U

V

W