segmentation_models_pytorch Documentation

Release 0.1.0

Pavel Yakubovskiy

Jan 03, 2020

Contents:

CHAPTER 1

API

1.1 Unet

class segmentation_models_pytorch.**Unet**(*encoder_name: str = 'resnet34'*, *encoder_depth: int = 5*, *encoder_weights: str = 'imagenet'*, *decoder_use_batchnorm: bool = True*, *decoder_channels: List[int] = (256, 128, 64, 32, 16)*, *decoder_attention_type: Optional[str] = None*, *in_channels: int = 3*, *classes: int = 1*, *activation: Union[str, callable, None] = None*, *aux_params: Optional[dict] = None*)

Unet is a fully convolution neural network for image semantic segmentation

Parameters

- **encoder_name** – name of classification model (without last dense layers) used as feature extractor to build segmentation model.
- **encoder_depth** (*int*) – number of stages used in decoder, larger depth - more features are generated. e.g. for depth=3 encoder will generate list of features with following spatial shapes [(H,W), (H/2, W/2), (H/4, W/4), (H/8, W/8)], so in general the deepest feature tensor will have spatial resolution (H/(2^depth), W/(2^depth)]
- **encoder_weights** – one of `None` (random initialization), `imagenet` (pre-training on ImageNet).
- **decoder_channels** – list of numbers of `Conv2D` layer filters in decoder blocks
- **decoder_use_batchnorm** – if `True`, `BatchNormalisation` layer between `Conv2D` and `Activation` layers is used. If 'inplace' InplaceABN will be used, allows to decrease memory consumption. One of [True, False, 'inplace']
- **decoder_attention_type** – attention module used in decoder of the model One of [`None`, `scse`]
- **in_channels** – number of input channels for model, default is 3.

- **classes** – a number of classes for output (output shape - `(batch, classes, h, w)`).
- **activation** – activation function to apply after final convolution; One of [`sigmoid`, `softmax`, `logsoftmax`, `identity`, callable, None]
- **aux_params** – if specified model will have additional classification auxiliary output build on top of encoder, supported params:
  - classes (int): number of classes
  - pooling (str): one of 'max', 'avg'. Default is 'avg'.
  - dropout (float): dropout factor in [0, 1)
  - activation (str): activation function to apply "sigmoid"/"softmax" (could be None to return logits)

Returns Unet

Return type `torch.nn.Module`

1.2 Linknet

class segmentation_models_pytorch.**Linknet**(*encoder_name: str = 'resnet34', encoder_depth: int = 5, encoder_weights: Optional[str] = 'imagenet', decoder_use_batchnorm: bool = True, in_channels: int = 3, classes: int = 1, activation: Union[str, callable, None] = None, aux_params: Optional[dict] = None*)

Linknet is a fully convolution neural network for fast image semantic segmentation

Note: This implementation by default has 4 skip connections (original - 3).

Parameters

- **encoder_name** – name of classification model (without last dense layers) used as feature extractor to build segmentation model.
- **encoder_depth** (*int*) – number of stages used in decoder, larger depth - more features are generated. e.g. for depth=3 encoder will generate list of features with following spatial shapes [(H,W), (H/2, W/2), (H/4, W/4), (H/8, W/8)], so in general the deepest feature will have spatial resolution (H/(2^depth), W/(2^depth)]
- **encoder_weights** – one of `None` (random initialization), `imagenet` (pre-training on ImageNet).
- **decoder_use_batchnorm** – if `True`, `BatchNormalisation` layer between `Conv2D` and `Activation` layers is used. If 'inplace' InplaceABN will be used, allows to decrease memory consumption. One of [True, False, 'inplace']
- **in_channels** – number of input channels for model, default is 3.
- **classes** – a number of classes for output (output shape - `(batch, classes, h, w)`).

- **activation** – activation function used in `.predict(x)` method for inference. One of [`sigmoid`, `softmax`, callable, None]
- **aux_params** – if specified model will have additional classification auxiliary output build on top of encoder, supported params:
  - classes (int): number of classes
  - pooling (str): one of 'max', 'avg'. Default is 'avg'.
  - dropout (float): dropout factor in [0, 1)
  - activation (str): activation function to apply "sigmoid"/"softmax" (could be None to return logits)

Returns Linknet

Return type `torch.nn.Module`

1.3 FPN

class segmentation_models_pytorch.**FPN**(*encoder_name: str = 'resnet34'*, *encoder_depth: int = 5*, *encoder_weights: Optional[str] = 'imagenet'*, *decoder_pyramid_channels: int = 256*, *decoder_segmentation_channels: int = 128*, *decoder_merge_policy: str = 'add'*, *decoder_dropout: float = 0.2*, *in_channels: int = 3*, *classes: int = 1*, *activation: Optional[str] = None*, *upsampling: int = 4*, *aux_params: Optional[dict] = None*)

FPN is a fully convolution neural network for image semantic segmentation :param encoder_name: name of classification model (without last dense layers) used as feature

extractor to build segmentation model.

Parameters

- **encoder_depth** – number of stages used in decoder, larger depth - more features are generated. e.g. for depth=3 encoder will generate list of features with following spatial shapes [(H,W), (H/2, W/2), (H/4, W/4), (H/8, W/8)], so in general the deepest feature will have spatial resolution (H/(2^depth), W/(2^depth)]
- **encoder_weights** – one of `None` (random initialization), `imagenet` (pre-training on ImageNet).
- **decoder_pyramid_channels** – a number of convolution filters in Feature Pyramid of FPN.
- **decoder_segmentation_channels** – a number of convolution filters in segmentation head of FPN.
- **decoder_merge_policy** – determines how to merge outputs inside FPN. One of [`add`, `cat`]
- **decoder_dropout** – spatial dropout rate in range (0, 1).
- **in_channels** – number of input channels for model, default is 3.
- **classes** – a number of classes for output (output shape - `(batch, classes, h, w)`).

- **activation** (*str, callable*) – activation function used in `.predict(x)` method for inference. One of [`sigmoid`, `softmax2d`, callable, None]
- **upsampling** – optional, final upsampling factor (default is 4 to preserve input -> output spatial shape identity)
- **aux_params** – if specified model will have additional classification auxiliary output build on top of encoder, supported params:
  - classes (int): number of classes
  - pooling (str): one of 'max', 'avg'. Default is 'avg'.
  - dropout (float): dropout factor in [0, 1)
  - activation (str): activation function to apply "sigmoid"/"softmax" (could be None to return logits)

Returns FPN

Return type `torch.nn.Module`

1.4 PSPNet

class segmentation_models_pytorch.**PSPNet**(*encoder_name: str = 'resnet34', encoder_weights: Optional[str] = 'imagenet', encoder_depth: int = 3, psp_out_channels: int = 512, psp_use_batchnorm: bool = True, psp_dropout: float = 0.2, in_channels: int = 3, classes: int = 1, activation: Union[str, callable, None] = None, upsampling: int = 8, aux_params: Optional[dict] = None*)

PSPNet is a fully convolution neural network for image semantic segmentation

Parameters

- **encoder_name** – name of classification model used as feature extractor to build segmentation model.
- **encoder_depth** – number of stages used in decoder, larger depth - more features are generated. e.g. for depth=3 encoder will generate list of features with following spatial shapes [(H,W), (H/2, W/2), (H/4, W/4), (H/8, W/8)], so in general the deepest feature will have spatial resolution (H/(2^depth), W/(2^depth)]
- **encoder_weights** – one of `None` (random initialization), `imagenet` (pre-training on ImageNet).
- **psp_out_channels** – number of filters in PSP block.
- **psp_use_batchnorm** – if `True`, `BatchNormalisation` layer between `Conv2D` and `Activation` layers is used. If 'inplace' InplaceABN will be used, allows to decrease memory consumption. One of [True, False, 'inplace']
- **psp_dropout** – spatial dropout rate between 0 and 1.
- **in_channels** – number of input channels for model, default is 3.
- **classes** – a number of classes for output (output shape - `(batch, classes, h, w)`).

- **activation** – activation function used in `.predict(x)` method for inference. One of [`sigmoid`, `softmax`, callable, None]
- **upsampling** – optional, final upsampling factor (default is 8 for depth=3 to preserve input -> output spatial shape identity)
- **aux_params** – if specified model will have additional classification auxiliary output build on top of encoder, supported params:
  - classes (int): number of classes
  - pooling (str): one of 'max', 'avg'. Default is 'avg'.
  - dropout (float): dropout factor in [0, 1)
  - activation (str): activation function to apply "sigmoid"/"softmax" (could be None to return logits)

Returns PSPNet

Return type `torch.nn.Module`

1.5 PAN

class segmentation_models_pytorch.pan.model.**PAN**(*encoder_name: str = 'resnet34'*, *encoder_weights: str = 'imagenet'*, *encoder_dilation: bool = True*, *decoder_channels: int = 32*, *in_channels: int = 3*, *classes: int = 1*, *activation: Union[str, callable, None] = None*, *upsampling: int = 4*, *aux_params: Optional[dict] = None*)

Implementation of _PAN (Pyramid Attention Network). Currently works with shape of input tensor >= [B x C x 128 x 128] for pytorch <= 1.1.0 and with shape of input tensor >= [B x C x 256 x 256] for pytorch == 1.3.1

Parameters

- **encoder_name** – name of classification model (without last dense layers) used as feature extractor to build segmentation model.
- **encoder_weights** – one of `None` (random initialization), `imagenet` (pre-training on ImageNet).
- **encoder_dilation** – Flag to use dilation in encoder last layer. Doesn't work with [`*ception*`, `vgg*`, `densenet*`] backbones, default is True.
- **decoder_channels** – Number of `Conv2D` layer filters in decoder blocks
- **in_channels** – number of input channels for model, default is 3.
- **classes** – a number of classes for output (output shape - `(batch, classes, h, w)`).
- **activation** – activation function to apply after final convolution; One of [`sigmoid`, `softmax`, `logsoftmax`, `identity`, callable, None]
- **upsampling** – optional, final upsampling factor (default is 4 to preserve input -> output spatial shape identity)
- **aux_params** – if specified model will have additional classification auxiliary output build on top of encoder, supported params:

- classes (int): number of classes
- pooling (str): one of 'max', 'avg'. Default is 'avg'.
- dropout (float): dropout factor in [0, 1)
- activation (str): activation function to apply "sigmoid"/"softmax" (could be None to return logits)

Returns PAN

Return type `torch.nn.Module`

Python library with Neural Networks for Image Segmentation based on PyTorch

The main features of this library are:

- High level API (just two lines to create neural network)
- 5 models architectures for binary and multi class segmentation (including legendary Unet)
- 46 available encoders for each architecture
- All encoders have pre-trained weights for faster and better convergence

Contents

CHAPTER 2

Quick start

Since the library is built on the PyTorch framework, created segmentation model is just a PyTorch nn.Module, which can be created as easy as:

```
import segmentation_models_pytorch as smp

model = smp.Unet()
```

Depending on the task, you can change the network architecture by choosing backbones with fewer or more parameters and use pretrainded weights to initialize it:

```
model = smp.Unet('resnet34', encoder_weights='imagenet')
```

Change number of output classes in the model:

```
model = smp.Unet('resnet34', classes=3, activation='softmax')
```

All models have pretrained encoders, so you have to prepare your data the same way as during weights pretraining:

```
from segmentation_models_pytorch.encoders import get_preprocessing_fn

preprocess_input = get_preprocessing_fn('resnet18', pretrained='imagenet')
```

CHAPTER 3

Examples

- Training model for cars segmentation on CamVid dataset here.
- Training SMP model with Catalyst (high-level framework for PyTorch), Ttach (TTA library for PyTorch) and Albumentations (fast image augmentation library) - here

CHAPTER 4

Models

4.1 Architectures

- Unet
- Linknet
- FPN
- PSPNet
- PAN

class segmentation_models_pytorch.**Unet**(*encoder_name: str = 'resnet34'*, *encoder_depth: int = 5*, *encoder_weights: str = 'imagenet'*, *decoder_use_batchnorm: bool = True*, *decoder_channels: List[int] = (256, 128, 64, 32, 16)*, *decoder_attention_type: Optional[str] = None*, *in_channels: int = 3*, *classes: int = 1*, *activation: Union[str, callable, None] = None*, *aux_params: Optional[dict] = None*)

Unet is a fully convolution neural network for image semantic segmentation

Parameters

- **encoder_name** – name of classification model (without last dense layers) used as feature extractor to build segmentation model.
- **encoder_depth** (*int*) – number of stages used in decoder, larger depth - more features are generated. e.g. for depth=3 encoder will generate list of features with following spatial shapes [(H,W), (H/2, W/2), (H/4, W/4), (H/8, W/8)], so in general the deepest feature tensor will have spatial resolution (H/(2^depth), W/(2^depth)]
- **encoder_weights** – one of `None` (random initialization), `imagenet` (pre-training on ImageNet).
- **decoder_channels** – list of numbers of `Conv2D` layer filters in decoder blocks

- **decoder_use_batchnorm** – if `True`, `BatchNormalisation` layer between `Conv2D` and `Activation` layers is used. If 'inplace' InplaceABN will be used, allows to decrease memory consumption. One of [True, False, 'inplace']
- **decoder_attention_type** – attention module used in decoder of the model One of [`None`, `scse`]
- **in_channels** – number of input channels for model, default is 3.
- **classes** – a number of classes for output (output shape - `(batch, classes, h, w)`).
- **activation** – activation function to apply after final convolution; One of [`sigmoid`, `softmax`, `logsoftmax`, `identity`, callable, None]
- **aux_params** – if specified model will have additional classification auxiliary output build on top of encoder, supported params:
  - classes (int): number of classes
  - pooling (str): one of 'max', 'avg'. Default is 'avg'.
  - dropout (float): dropout factor in [0, 1)
  - activation (str): activation function to apply "sigmoid"/"softmax" (could be None to return logits)

Returns Unet

Return type `torch.nn.Module`

4.2 Encoders

Encoder	Weights	Params, M
resnet18	imagenet	11M
resnet34	imagenet	21M
resnet50	imagenet	23M
resnet101	imagenet	42M
resnet152	imagenet	58M
resnext50_32x4d	imagenet	22M
resnext101_32x8d	imagenetinstagram	86M
resnext101_32x16d	instagram	191M
resnext101_32x32d	instagram	466M
resnext101_32x48d	instagram	826M
dpn68	imagenet	11M
dpn68b	imagenet+5k	11M
dpn92	imagenet+5k	34M
dpn98	imagenet	58M
dpn107	imagenet+5k	84M
dpn131	imagenet	76M
vgg11	imagenet	9M
vgg11_bn	imagenet	9M
vgg13	imagenet	9M
vgg13_bn	imagenet	9M
vgg16	imagenet	14M

Continued on next page

Table 1 – continued from previous page

Encoder	Weights	Params, M
vgg16_bn	imagenet	14M
vgg19	imagenet	20M
vgg19_bn	imagenet	20M
senet154	imagenet	113M
se_resnet50	imagenet	26M
se_resnet101	imagenet	47M
se_resnet152	imagenet	64M
se_resnext50_32x4d	imagenet	25M
se_resnext101_32x4d	imagenet	46M
densenet121	imagenet	6M
densenet169	imagenet	12M
densenet201	imagenet	18M
densenet161	imagenet	26M
inceptionresnetv2	imagenetimagenet+background	54M
inceptionv4	imagenetimagenet+background	41M
efficientnet-b0	imagenet	4M
efficientnet-b1	imagenet	6M
efficientnet-b2	imagenet	7M
efficientnet-b3	imagenet	10M
efficientnet-b4	imagenet	17M
efficientnet-b5	imagenet	28M
efficientnet-b6	imagenet	40M
efficientnet-b7	imagenet	63M
mobilenet_v2	imagenet	2M
xception	imagenet	22M

CHAPTER 5

Models API

- `model.encoder` - pretrained backbone to extract features of different spatial resolution
- `model.decoder` - depends on models architecture (`Unet`/`Linknet`/`PSPNet`/`FPN`)
- `model.segmentation_head` - last block to produce required number of mask channels (include also optional upsampling and activation)
- `model.classification_head` - optional block which create classification head on top of encoder
- `model.forward(x)` - sequentially pass `x` through model's encoder, decoder and segmentation head (and classification head if specified)

Input channels parameter allow you to create models, which process tensors with arbitrary number of channels. If you use pretrained weights from imagenet - weights of first convolution will be reused for 1- or 2- channels inputs, for input channels > 4 weights of first convolution will be initialized randomly.

```
model = smp.FPN('resnet34', in_channels=1)
mask = model(torch.ones([1, 1, 64, 64]))
```

All models support `aux_params` parameters, which is default set to `None`. If `aux_params = None` than classification auxiliary output is not created, else model produce not only `mask`, but also `label` output with shape `NC`. Classification head consist of GlobalPooling->Dropout(optional)->Linear->Activation(optional) layers, which can be configured by `aux_params` as follows:

```
aux_params=dict(
    pooling='avg',             # one of 'avg', 'max'
    dropout=0.5,               # dropout ratio, default is None
    activation='sigmoid',      # activation function, default is None
    classes=4,                 # define number of output labels
)
model = smp.Unet('resnet34', classes=4, aux_params=aux_params)
mask, label = model(x)
```

Depth parameter specify a number of downsampling operations in encoder, so you can make your model lighted if specify smaller `depth`.

```
model = smp.Unet('resnet34', encoder_depth=4)
```

CHAPTER 6

Installation

PyPI version:

```
$ pip install segmentation-models-pytorch
```

Latest version from source:

```
$ pip install git+https://github.com/qubvel/segmentation_models.pytorch
```

CHAPTER 7

Competitions won with the library

`Segmentation Models` package is widely used in the image segmentation competitions. Here you can find competitions, names of the winners and links to their solutions.

CHAPTER 8

License

Project is distributed under MIT License

CHAPTER 9

Contributing

```
$ docker build -f docker/Dockerfile.dev -t smp:dev . && docker run --rm smp:dev␣
↪pytest -p no:cacheprovider
```

```
$ docker build -f docker/Dockerfile.dev -t smp:dev . && docker run --rm smp:dev␣
↪python misc/generate_table.py
```

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Index