Segmentation Models

Contents

Print to PDF

Unet
Unet++
FPN
PSPNet
DeepLabV3
DeepLabV3+
Linknet
MAnet
PAN
UPerNet
Segformer

Back to top

latest

Unet

class segmentation_models_pytorch.**Unet**(*encoder_name='resnet34'*, *encoder_depth=5*, *encoder_weights='imagenet'*, *decoder_use_batchnorm=True*, *decoder_channels=(256, 128, 64, 32, 16)*, *decoder_attention_type=None*, *in_channels=3*, *classes=1*, *activation=None*, *aux_params=None*, ***kwargs*) [source]

Unet_ is a fully convolution neural network for image semantic segmentation. Consist of *encoder* and *decoder* parts connected with *skip connections*. Encoder extract features of different spatial resolution (skip connections) which are used by decoder to define accurate segmentation mask. Use *concatenation* for fusing decoder blocks with skip connections.

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **“imagenet”** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **decoder_channels** (*Tuple[int, ...]*) – List of integers which specify **in_channels** parameter for convolutions used in decoder. Length of the list should be the same as **encoder_depth**
- **decoder_use_batchnorm** (*bool*) – If **True**, BatchNorm2d layer between Conv2D and Activation layers is used. If **“inplace”** InplaceABN will be used, allows to decrease memory consumption. Available options are **True, False, “inplace”**
- **decoder_attention_type** (*str* | *None*) – Attention module used in decoder of the model. Available options are **None** and **scse** (https://arxiv.org/abs/1808.08127).
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *Callable* | *None*) –

latest

An activation function to apply after the final convolution layer. Available options are **“sigmoid”**, **“softmax”**, **“logsoftmax”**, **“tanh”**, **“identity”**,

> **callable** and **None**.

Default is **None**

- **aux_params** (*dict | None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

  > - classes (int): A number of classes
  > - pooling (str): One of “max”, “avg”. Default is “avg”
  > - dropout (float): Dropout factor in [0, 1)
  > - **activation (str): An activation function to apply “sigmoid”/”softmax”**
  >
  >   (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

Unet

Return type:

`torch.nn.Module`

Unet++

class **segmentation_models_pytorch.UnetPlusPlus**(***encoder_name***=*'resnet34'*, ***encoder_depth***=*5*, ***encoder_weights***=*'imagenet'*, ***decoder_use_batchnorm***=*True*, ***decoder_channels***=*(256, 128, 64, 32, 16)*, ***decoder_attention_type***=*None*, ***in_channels***=*3*, ***classes***=*1*, ***activation***=*None*, ***aux_params***=*None*, ****kwargs***) [source]

Unet++ is a fully convolution neural network for image semantic segmentation. Consist of *encoder* and *decoder* parts connected with *skip connections*. Encoder extract different spatial resolution (skip connections) which are used by decoder to define accurate segmentation mask. Decoder of Unet++ is more complex than in usual Unet.

latest

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **"imagenet"** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **decoder_channels** (*List[int]*) – List of integers which specify **in_channels** parameter for convolutions used in decoder. Length of the list should be the same as **encoder_depth**
- **decoder_use_batchnorm** (*bool*) – If **True**, BatchNorm2d layer between Conv2D and Activation layers is used. If **"inplace"** InplaceABN will be used, allows to decrease memory consumption. Available options are **True, False, "inplace"**
- **decoder_attention_type** (*str* | *None*) – Attention module used in decoder of the model. Available options are **None** and **scse** (https://arxiv.org/abs/1808.08127).
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *callable* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **"sigmoid"**, **"softmax"**, **"logsoftmax"**, **"tanh"**, **"identity"**,

  > **callable** and **None**.

  Default is **None**
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

  > - classes (int): A number of classes
  > - pooling (str): One of "max", "avg". Default is "avg"

latest

- dropout (float): Dropout factor in [0, 1)
- **activation (str): An activation function to apply "sigmoid"/"softmax"**
  (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

Unet++

Return type:

`torch.nn.Module`

Reference:

https://arxiv.org/abs/1807.10165

FPN

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

FPN_ is a fully convolution neural network for image semantic segmentation.

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **"imagenet"** (pre-training on ImageNet) and other pretrained weights (see table

- **decoder_pyramid_channels** (*int*) – A number of convolution filters in Feature Pyramid of FPN_
- **decoder_segmentation_channels** (*int*) – A number of convolution filters in segmentation blocks of FPN_
- **decoder_merge_policy** (*str*) – Determines how to merge pyramid features inside FPN. Available options are **add** and **cat**
- **decoder_dropout** (*float*) – Spatial dropout rate in range (0, 1) for feature pyramid in FPN_
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **"sigmoid"**, **"softmax"**, **"logsoftmax"**, **"tanh"**, **"identity"**,

  > **callable** and **None**.

  Default is **None**
- **upsampling** (*int*) – Final upsampling factor. Default is 4 to preserve input-output spatial shape identity
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

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

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class __: function. Applies only to `timm` models. Keys with `None` values are passing.

latest

Returns:

FPN

Return type:

`torch.nn.Module`

PSPNet

class **segmentation_models_pytorch.PSPNet**(***encoder_name*****='resnet34'**, ***encoder_weights*****='imagenet'**, ***encoder_depth*****=3**, ***psp_out_channels*****=512**, ***psp_use_batchnorm*****=True**, ***psp_dropout*****=0.2**, ***in_channels*****=3**, ***classes*****=1**, ***activation*****=None**, ***upsampling*****=8**, ***aux_params*****=None**, ****kwargs**) [source]

PSPNet_ is a fully convolution neural network for image semantic segmentation. Consist of *encoder* and *Spatial Pyramid* (decoder). Spatial Pyramid build on top of encoder and does not use "fine-features" (features of high spatial resolution). PSPNet can be used for multiclass segmentation of high resolution images, however it is not good for detecting small objects and producing accurate, pixel-level mask.

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **"imagenet"** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **psp_out_channels** (*int*) – A number of filters in Spatial Pyramid
- **psp_use_batchnorm** (*bool*) – If **True**, BatchNorm2d layer between Conv2D and Activation layers is used. If **"inplace"** InplaceABN will be used, allows to decrease memory consumption. Available options are **True, False, "inplace"**
- **psp_dropout** (*float*) – Spatial dropout rate in [0, 1) used in Spatial Pyramid
- **in_channels** (*int*) – A number of input channels for the model, defau images)

latest

- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *callable* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **“sigmoid”**, **“softmax”**, **“logsoftmax”**, **“tanh”**, **“identity”**,

  > **callable** and **None**.

  Default is **None**
- **upsampling** (*int*) – Final upsampling factor. Default is 8 to preserve input-output spatial shape identity
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

  > - classes (int): A number of classes
  > - pooling (str): One of “max”, “avg”. Default is “avg”
  > - dropout (float): Dropout factor in [0, 1)
  > - **activation (str): An activation function to apply “sigmoid”/”softmax”**
  >
  > (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

PSPNet

Return type:

`torch.nn.Module`

DeepLabV3

class segmentation_models_pytorch.DeepLabV3(*encoder_name='resne* latest
encoder_depth=5, encoder_weights='imagenet', encoder_output_stride=8,
decoder_channels=256, decoder_atrous_rates=(12, 24, 36),

decoder_aspp_separable*=*False*, *decoder_aspp_dropout*=*0.5*, *in_channels*=*3*, *classes*=*1*, *activation*=*None*, *upsampling*=*None*, *aux_params*=*None*, ***kwargs)

DeepLabV3_ implementation from "Rethinking Atrous Convolution for Semantic Image Segmentation" [source]

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **"imagenet"** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **decoder_channels** (*int*) – A number of convolution filters in ASPP module. Default is 256
- **encoder_output_stride** (*Literal[8, 16]*) – Downsampling factor for last encoder features (see original paper for explanation)
- **decoder_atrous_rates** (*Iterable[int]*) – Dilation rates for ASPP module (should be an iterable of 3 integer values)
- **decoder_aspp_separable** (*bool*) – Use separable convolutions in ASPP module. Default is False
- **decoder_aspp_dropout** (*float*) – Use dropout in ASPP module projection layer. Default is 0.5
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **"sigmoid"**, **"softmax"**, **"logsoftmax"**, **"tanh"**, **"identity"**,

  > **callable** and **None**.

Default is **None**

- **upsampling** (*int* | *None*) – Final upsampling factor. Default is **None** to preserve input-output spatial shape identity
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

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

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

DeepLabV3

Return type:

`torch.nn.Module`

DeepLabV3+

class **segmentation_models_pytorch.DeepLabV3Plus**(***encoder_name***=*'resnet34'*, ***encoder_depth***=*5*, ***encoder_weights***=*'imagenet'*, ***encoder_output_stride***=*16*, ***decoder_channels***=*256*, ***decoder_atrous_rates***=*(12, 24, 36)*, ***decoder_aspp_separable***=*True*, ***decoder_aspp_dropout***=*0.5*, ***in_channels***=*3*, ***classes***=*1*, ***activation***=*None*, ***upsampling***=*4*, ***aux_params***=*None*, ****kwargs***) [source]

DeepLabV3+ implementation from "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation"

Parameters:

latest

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution

- **encoder_depth** (*Literal[3, 4, 5]*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **"imagenet"** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **encoder_output_stride** (*Literal[8, 16]*) – Downsampling factor for last encoder features (see original paper for explanation)
- **decoder_atrous_rates** (*Iterable[int]*) – Dilation rates for ASPP module (should be an iterable of 3 integer values)
- **decoder_aspp_separable** (*bool*) – Use separable convolutions in ASPP module. Default is True
- **decoder_aspp_dropout** (*float*) – Use dropout in ASPP module projection layer. Default is 0.5
- **decoder_channels** (*int*) – A number of convolution filters in ASPP module. Default is 256
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **"sigmoid"**, **"softmax"**, **"logsoftmax"**, **"tanh"**, **"identity"**,

  > **callable** and **None**.

  Default is **None**
- **upsampling** (*int*) – Final upsampling factor. Default is 4 to preserve input-output spatial shape identity.
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (defau

  latest

  params:

  > - classes (int): A number of classes

  - pooling (str): One of “max”, “avg”. Default is “avg”
  - dropout (float): Dropout factor in [0, 1)
  - **activation (str): An activation function to apply “sigmoid”/”softmax”**
    (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

DeepLabV3Plus

Return type:

`torch.nn.Module`

Reference:

https://arxiv.org/abs/1802.02611v3

Linknet

class **segmentation_models_pytorch.Linknet**(***encoder_name***=*'resnet34'*, ***encoder_depth***=*5*, ***encoder_weights***=*'imagenet'*, ***decoder_use_batchnorm***=*True*, ***in_channels***=*3*, ***classes***=*1*, ***activation***=*None*, ***aux_params***=*None*, ****kwargs***) [source]

Linknet_ is a fully convolution neural network for image semantic segmentation. Consist of *encoder* and *decoder* parts connected with *skip connections*. Encoder extract features of different spatial resolution (skip connections) which are used by decoder to define accurate segmentation mask. Use *sum* for fusing decoder blocks with skip connections.

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

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial reso
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one

(e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5

- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **“imagenet”** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **decoder_use_batchnorm** (*bool*) – If **True**, BatchNorm2d layer between Conv2D and Activation layers is used. If **“inplace”** InplaceABN will be used, allows to decrease memory consumption. Available options are **True, False, “inplace”**
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *callable* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **“sigmoid”**, **“softmax”**, **“logsoftmax”**, **“tanh”**, **“identity”**,

  > **callable** and **None**.

  Default is **None**
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

  > - classes (int): A number of classes
  > - pooling (str): One of “max”, “avg”. Default is “avg”
  > - dropout (float): Dropout factor in [0, 1)
  > - **activation (str): An activation function to apply “sigmoid”/”softmax”**
  >
  > (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

latest

Linknet

Return type:

`torch.nn.Module`

MAnet

class **segmentation_models_pytorch.MAnet**(***encoder_name***=*'resnet34'*, ***encoder_depth***=*5*, ***encoder_weights***=*'imagenet'*, ***decoder_use_batchnorm***=*True*, ***decoder_channels***=*(256, 128, 64, 32, 16)*, ***decoder_pab_channels***=*64*, ***in_channels***=*3*, ***classes***=*1*, ***activation***=*None*, ***aux_params***=*None*, ****kwargs***) [source]

MAnet_ : Multi-scale Attention Net. The MA-Net can capture rich contextual dependencies based on the attention mechanism, using two blocks:

- Position-wise Attention Block (PAB), which captures the spatial dependencies between pixels in a global view
- Multi-scale Fusion Attention Block (MFAB), which captures the channel dependencies between any feature map by multi-scale semantic feature fusion

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **“imagenet”** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **decoder_channels** (*List[int]*) – List of integers which specify **in_channels** parameter for convolutions used in decoder. Length of the list should be the same as **encoder_depth**
- **decoder_use_batchnorm** (*bool*) – If **True**, BatchNorm2d layer between Conv2D and Activation layers is used. If **“inplace”** InplaceABN will be used, allow memory consumption. Available options are **True, False, “inplace”**

latest

- **decoder_pab_channels** (*int*) – A number of channels for PAB module in decoder. Default is 64.
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *callable* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **“sigmoid”**, **“softmax”**, **“logsoftmax”**, **“tanh”**, **“identity”**,

  > **callable** and **None**.

  Default is **None**
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

  > - classes (int): A number of classes
  > - pooling (str): One of “max”, “avg”. Default is “avg”
  > - dropout (float): Dropout factor in [0, 1)
  > - **activation (str): An activation function to apply “sigmoid”/”softmax”**
  >
  > (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

MAnet

Return type:

`torch.nn.Module`

PAN

latest

class segmentation_models_pytorch.PAN(*encoder_name='resnet34', encoder_depth=5, encoder_weights='imagenet', encoder_output_stride=16,*

***decoder_channels*=*32*, *in_channels*=*3*, *classes*=*1*, *activation*=*None*, *upsampling*=*4*, *aux_params*=*None*, ***kwargs*)** [source]

Implementation of PAN_ (Pyramid Attention Network).

> **Note**
>
> 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** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*Literal[3, 4, 5]*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **“imagenet”** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **encoder_output_stride** (*Literal[16, 32]*) – 16 or 32, if 16 use dilation in encoder last layer. Doesn’t work with ***ception***, **vgg***, **densenet***` backbones.Default is 16.
- **decoder_channels** (*int*) – A number of convolution layer filters in decoder blocks
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *Callable* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **“sigmoid”**, **“softmax”**, **“logsoftmax”**, **“tanh”**, **“identity”**,

  > **callable** and **None**.

  Default is **None**
- **upsampling** (*int*) – Final upsampling factor. Default is 4 to preserve spatial shape identity

latest

- **aux_params** (*dict | None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

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

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

PAN

Return type:

`torch.nn.Module`

UPerNet

class **segmentation_models_pytorch.UPerNet**(***encoder_name*** *='resnet34',* ***encoder_depth*** *=5,* ***encoder_weights*** *='imagenet',* ***decoder_pyramid_channels*** *=256,* ***decoder_segmentation_channels*** *=64,* ***in_channels*** *=3,* ***classes*** *=1,* ***activation*** *=None,* ***aux_params*** *=None,* *****kwargs***) [source]

UPerNet is a unified perceptual parsing network for image segmentation.

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5

- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **“imagenet”** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **decoder_pyramid_channels** (*int*) – A number of convolution filters in Feature Pyramid, default is 256
- **decoder_segmentation_channels** (*int*) – A number of convolution filters in segmentation blocks, default is 64
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *callable* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **“sigmoid”**, **“softmax”**, **“logsoftmax”**, **“tanh”**, **“identity”**,

  > **callable** and **None**.

  Default is **None**
- **aux_params** (*dict* | *None*) –

  Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

  > - classes (int): A number of classes
  > - pooling (str): One of “max”, “avg”. Default is “avg”
  > - dropout (float): Dropout factor in [0, 1)
  > - **activation (str): An activation function to apply “sigmoid”/”softmax”** (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

UPerNet

Return type:

`torch.nn.Module`

Segformer

class segmentation_models_pytorch.**Segformer**(***encoder_name***=*'resnet34'*, ***encoder_depth***=*5*, ***encoder_weights***=*'imagenet'*, ***decoder_segmentation_channels***=*256*, ***in_channels***=*3*, ***classes***=*1*, ***activation***=*None*, ***aux_params***=*None*, ****kwargs***) [source]

Segformer is simple and efficient design for semantic segmentation with Transformers

Parameters:

- **encoder_name** (*str*) – Name of the classification model that will be used as an encoder (a.k.a backbone) to extract features of different spatial resolution
- **encoder_depth** (*int*) – A number of stages used in encoder in range [3, 5]. Each stage generate features two times smaller in spatial dimensions than previous one (e.g. for depth 0 we will have features with shapes [(N, C, H, W),], for depth 1 - [(N, C, H, W), (N, C, H // 2, W // 2)] and so on). Default is 5
- **encoder_weights** (*str* | *None*) – One of **None** (random initialization), **“imagenet”** (pre-training on ImageNet) and other pretrained weights (see table with available weights for each encoder_name)
- **decoder_segmentation_channels** (*int*) – A number of convolution filters in segmentation blocks, default is 256
- **in_channels** (*int*) – A number of input channels for the model, default is 3 (RGB images)
- **classes** (*int*) – A number of classes for output mask (or you can think as a number of channels of output mask)
- **activation** (*str* | *Callable* | *None*) –

  An activation function to apply after the final convolution layer. Available options are **“sigmoid”**, **“softmax”**, **“logsoftmax”**, **“tanh”**, **“identity”**,

  > **callable** and **None**.

  Default is **None**
- **aux_params** (*dict* | *None*) –

Dictionary with parameters of the auxiliary output (classification head). Auxiliary output is build on top of encoder if **aux_params** is not **None** (default). Supported params:

- classes (int): A number of classes
- pooling (str): One of “max”, “avg”. Default is “avg”
- dropout (float): Dropout factor in [0, 1)
- **activation (str): An activation function to apply “sigmoid”/”softmax”**
  (could be **None** to return logits)

- **kwargs** (*dict[str, Any]*) – Arguments passed to the encoder class `__init__()` function. Applies only to `timm` models. Keys with `None` values are pruned before passing.

Returns:

Segformer

Return type:

`torch.nn.Module`