le DigitalPathology library

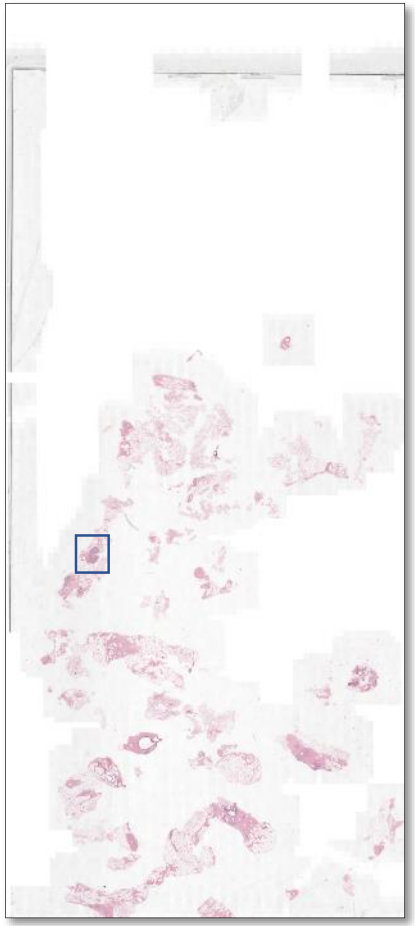
DigitalPathology library

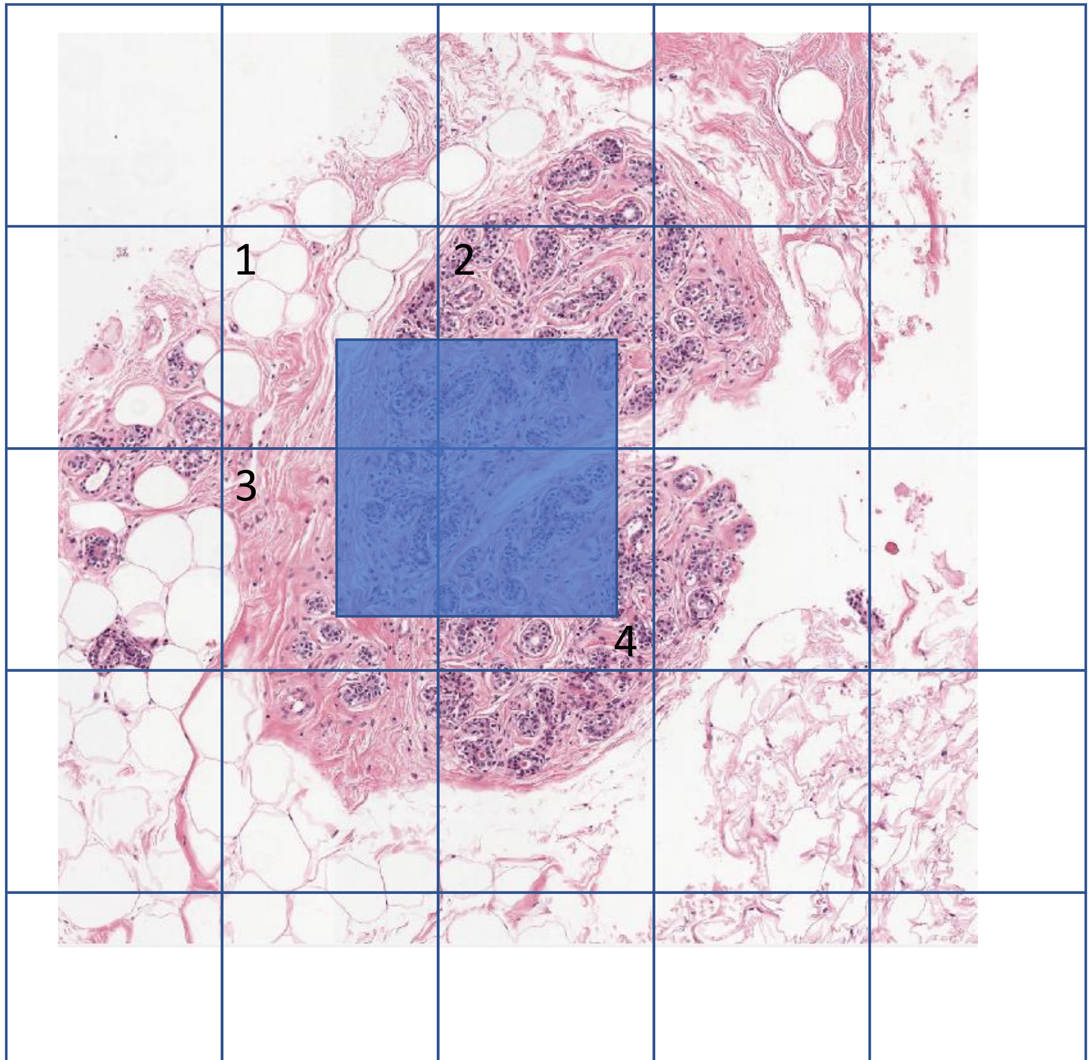
- https://github.com/DIAGNijmegen/DigitalPathology
- 110 files
- 27032 lines of code
- Main purpose:
  1. **Data extraction** from multiresolution images
  2. **Saving/loading network models**
  3. **Executing experiments**
  4. **Collection of handy scripts**
  5. **Some examples**
- Documentation: docstring only *(for now)*
- Testing: No *(for now)*

Main components

1. **BatchGenerator:** for loading patch of images from collection of multiresolution images.
2. **ModelBase:** reusable representation of networks.
3. **NetworkTrainer:** for executing, logging and saving deep learning experiments.
4. **scripts:** collection of useful scripts.

Multi-resolution images

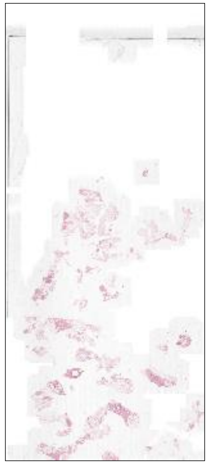




- Images are stored in tiles
- Tiles are JPEG compressed
- One collection of tiles per level

Multi-resolution images

0.25 µm



0.5 µm

1.0 µm

2.0 µm

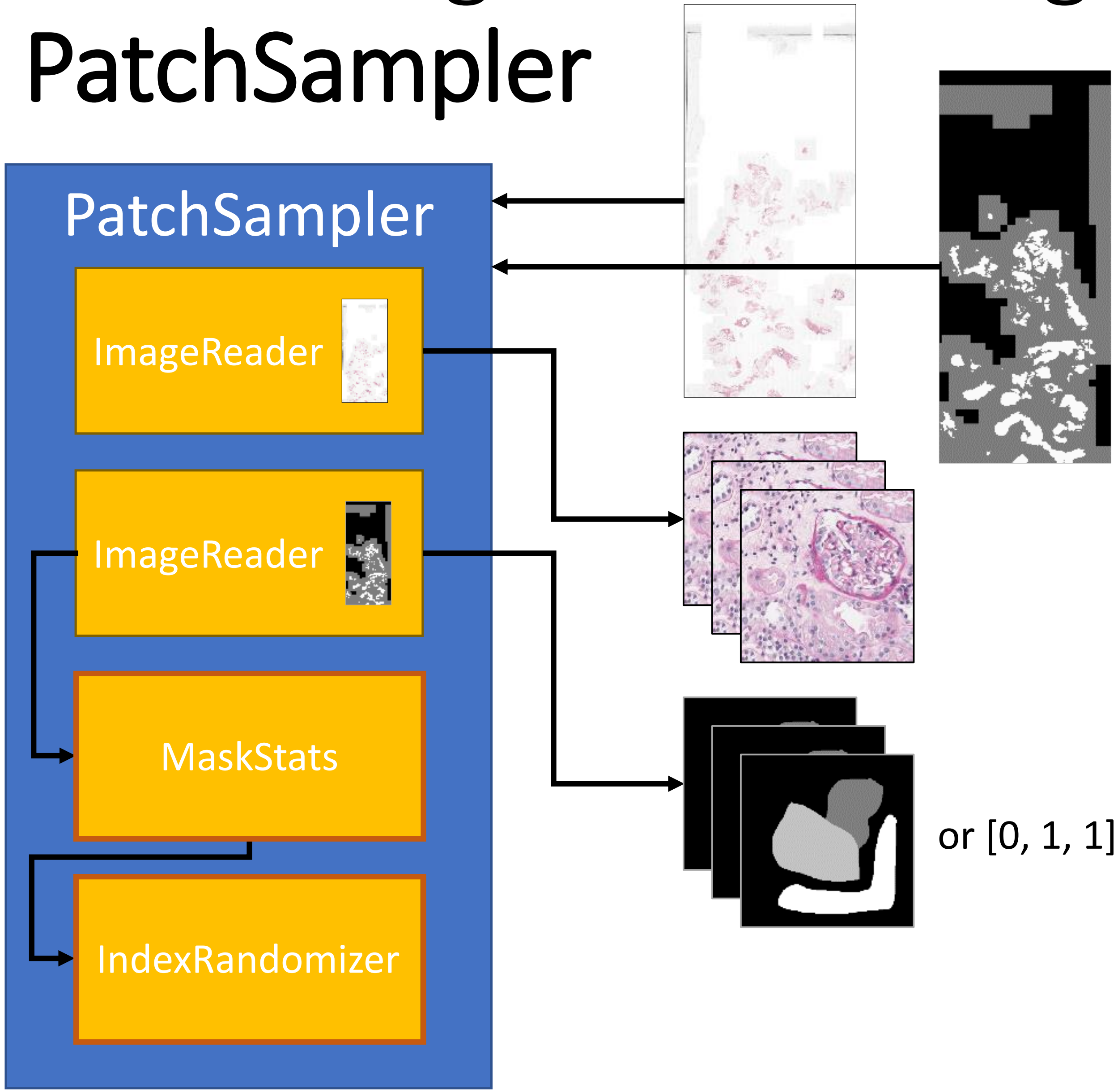
- The levels cover the same spatial area, but with increasing pixel size
- Use pixel spacing instead of zoom
- Levels: arbitrary
- Identify levels by pixel spacing

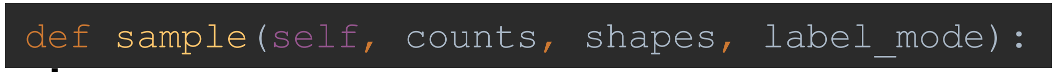
Multi-resolution images

- multiresolutionimageinterface of ASAP
  - Uses OpenSlide and LibTIFF
  - C++ library
  - Not pythonic
- digitalpathology.image.imagereader.ImageReader
  digitalpathology.image.imagewriter.ImageWriter
  - Pythonic
  - Exceptions
  - Returns numpy array of the right size and data type

1. BatchGenerator

Generating batch of image patches: **PatchSampler**

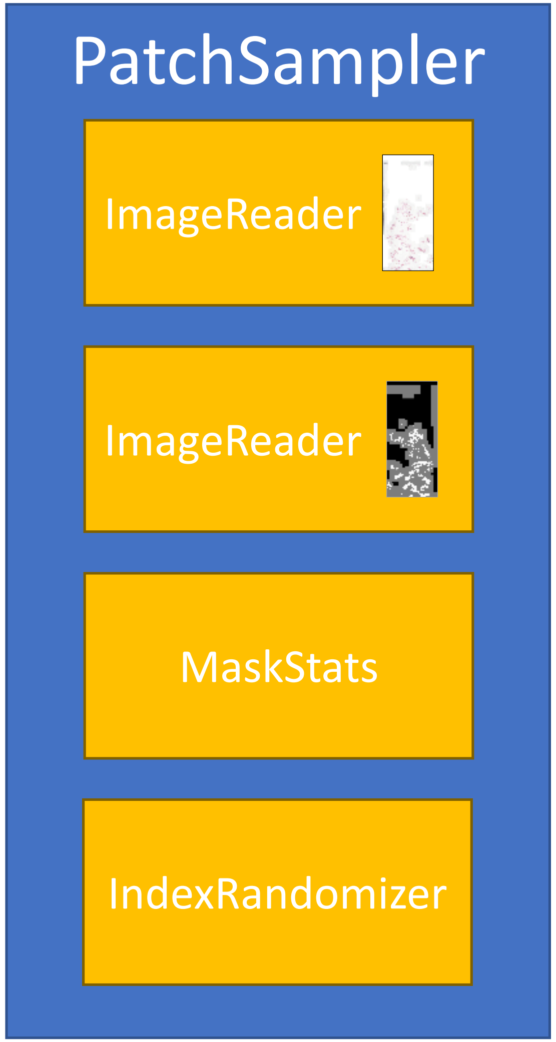




```
def sample(self, counts, shapes, label_mode):
```

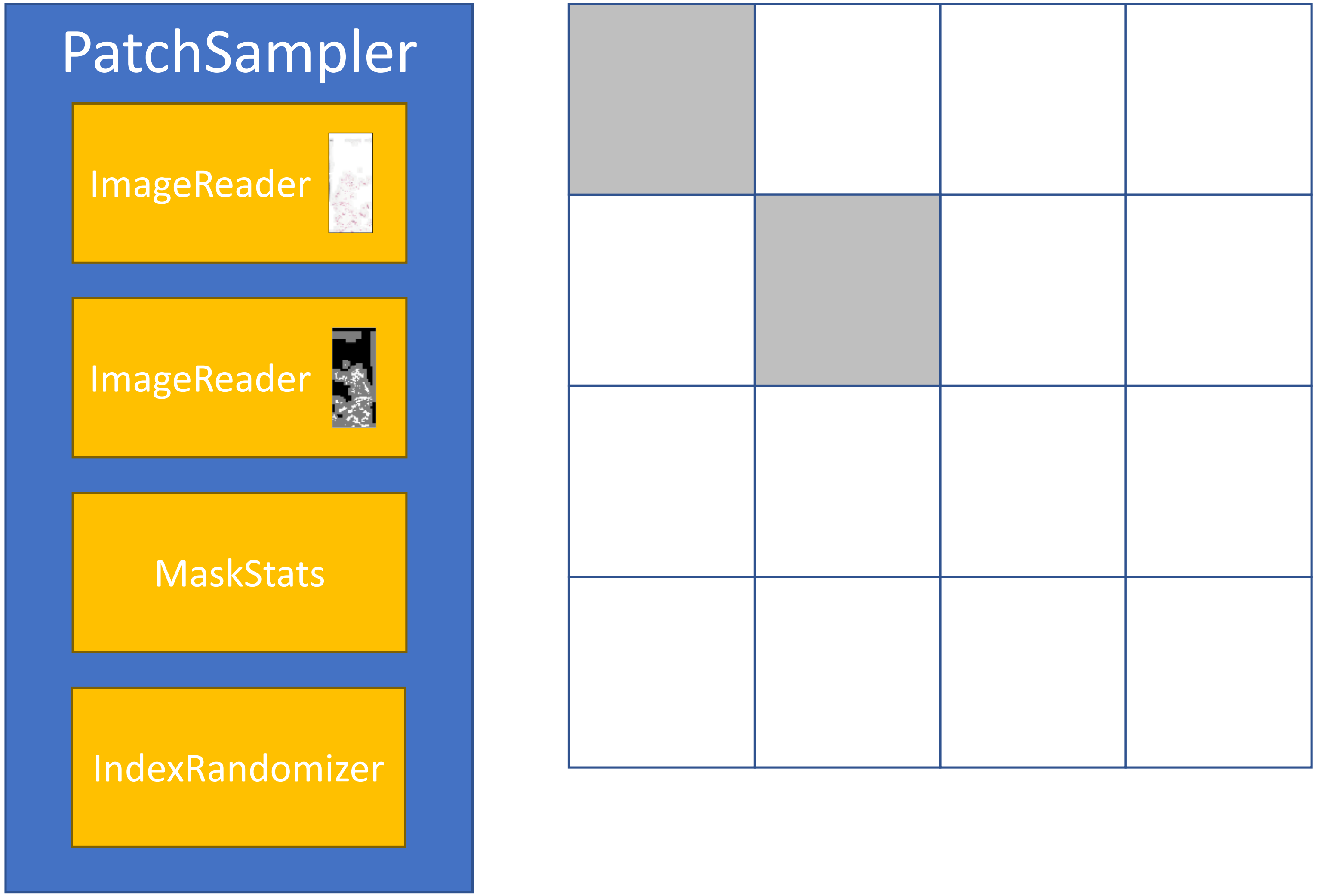
- **counts:** label value to label count mapping
- **shapes:** pixel spacings to patch shape mapping
- **label_mode:** label generation mode
  - central
  - synthesize
  - load

Generating batch of image patches: **PatchSampler**



- Mask level is arbitrary
- The lower the level to more memory the MaskStats consume
- Matching of image and mask is based on pixel spacing
- Fallback to image shape matching
- Parsing of mask files is slow, MaskStats can be saved/loaded (.stat files)
- The extracted image patch is centered on the randomized position. Always truncated to the upper left.
- In case of pixel spacing mismatch, the central pixel is used.

Generating batch of image patches: **PatchSampler**



PatchSampler

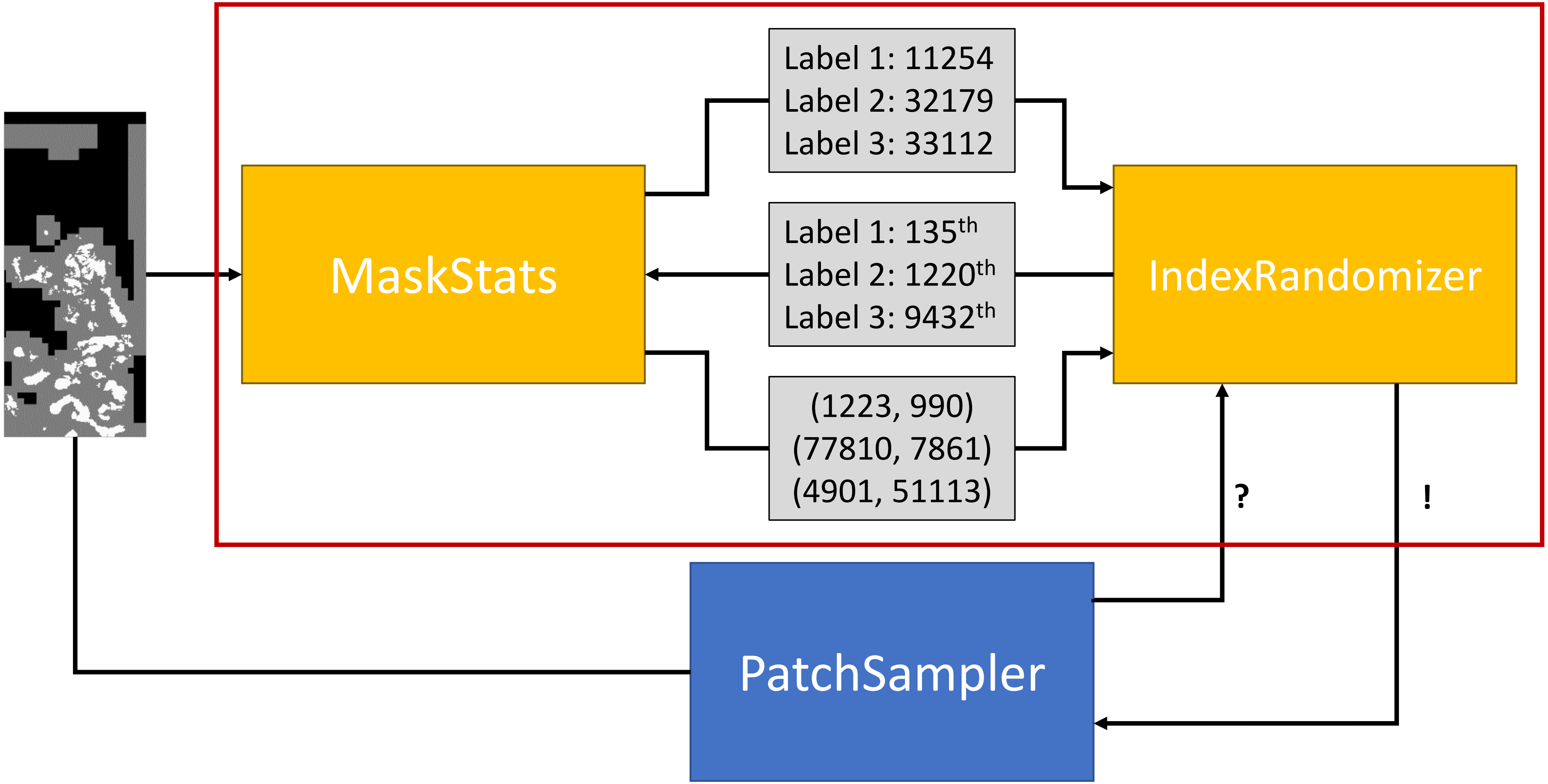
ImageReader

ImageReader

MaskStats

IndexRandomizer

Generating batch of image patches: **MaskStats** and **IndexRandomizer**



MaskStats

Label 1: 11254
Label 2: 32179
Label 3: 33112

Label 1: 135th
Label 2: 1220th
Label 3: 9432th

(1223, 990)
(77810, 7861)
(4901, 51113)

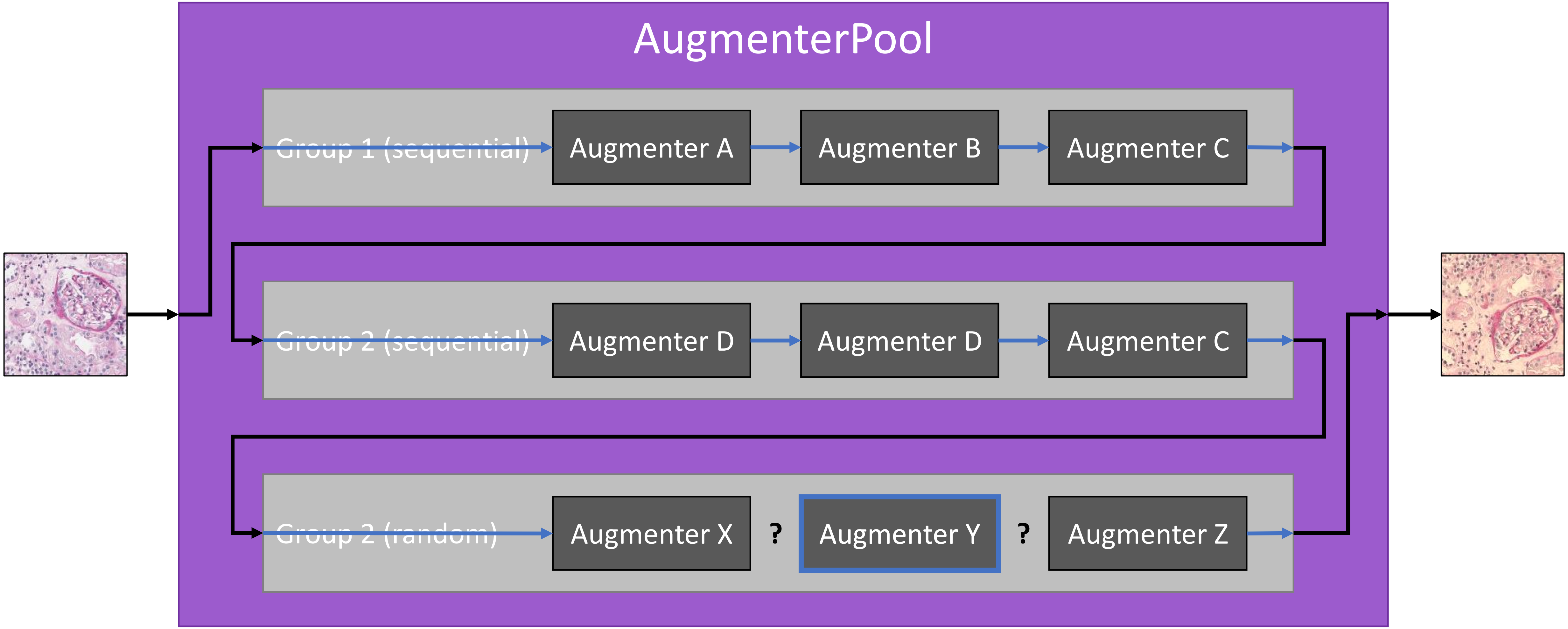
IndexRandomizer

?

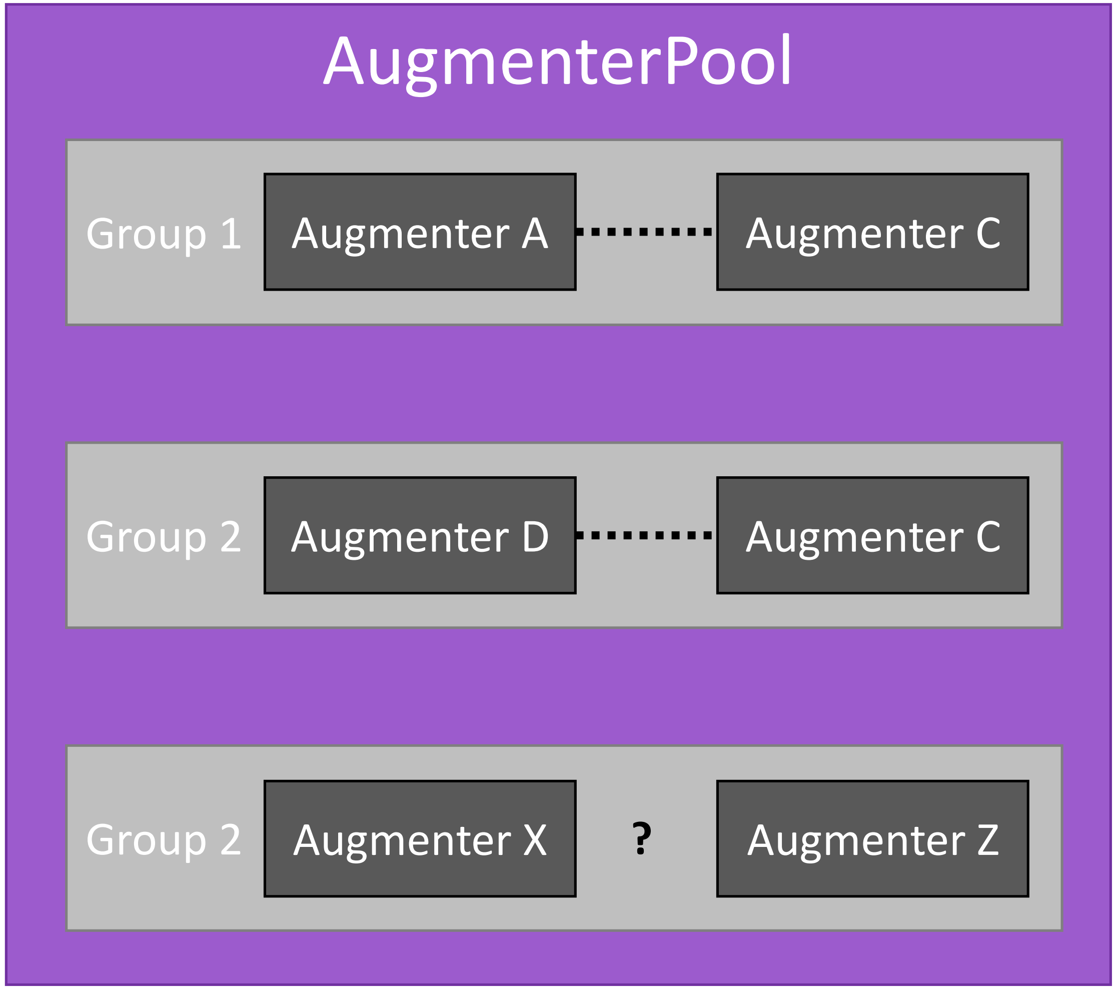
!

PatchSampler

Generating batch of image patches: **AugmenterPool**



Generating batch of image patches: **AugmenterPool**



```
def appendgroup(self, group, randomized):
```

- Append an empty group to the pool
- Groups are executed in order
- Groups can be sequential or random

```
def appendaugmenter(self, augmenter, group, ratio=0.0):
```

- Append an augmenter to an existing group
- Set probability for random selection

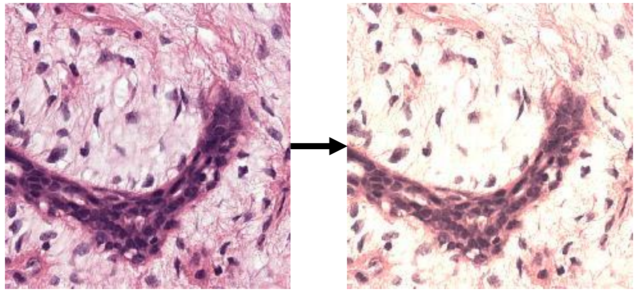
```
def process(self, patches, shapes=None, randomize=True):
```

Process a patch collection (as extracted by PatchSampler)

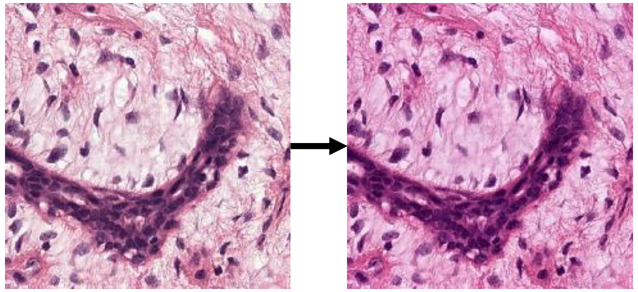
```
def randomize(self):
```

Randomize parameters for each augmenter object

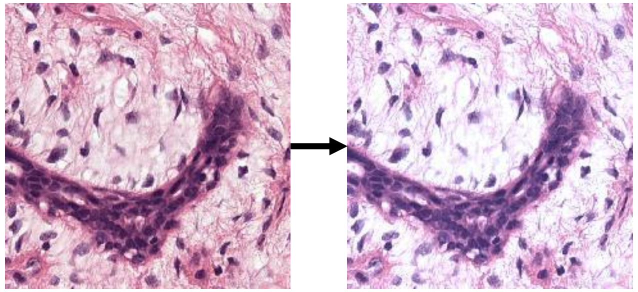
Generating batch of image patches: HedColorAugmenter, HsbColorAugmenter



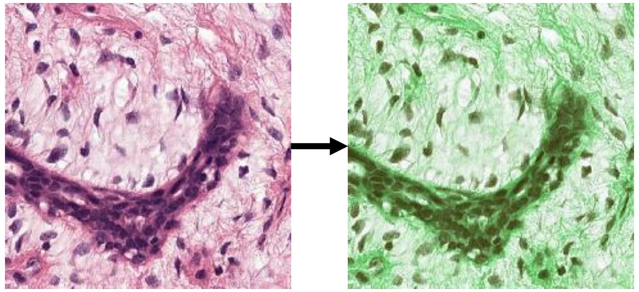
HedColorAugmenter
Haematoxylin adjustment



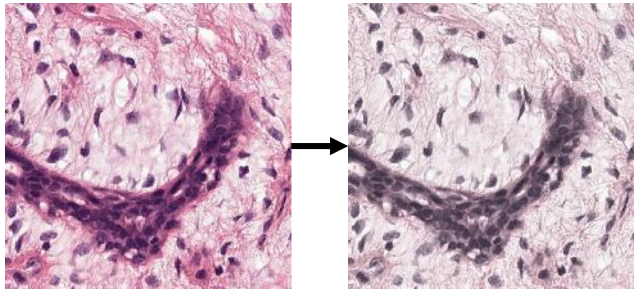
HedColorAugmenter
Eusin adjustment



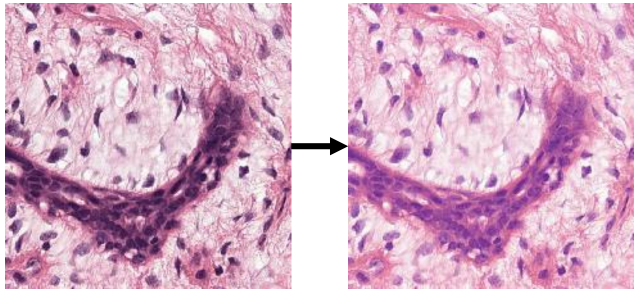
HedColorAugmenter
DAB adjustment



HsbColorAugmenter
Hue adjustment



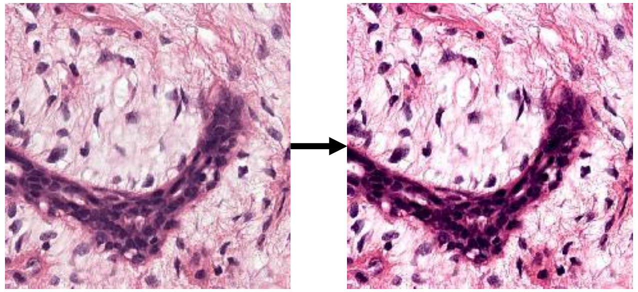
HsbColorAugmenter
Saturation adjustment



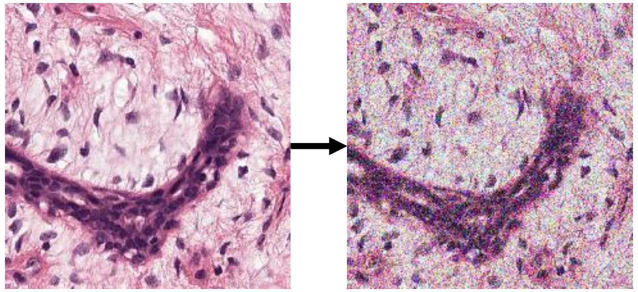
HsbColorAugmenter
Brightness adjustment

Generating batch of image patches:

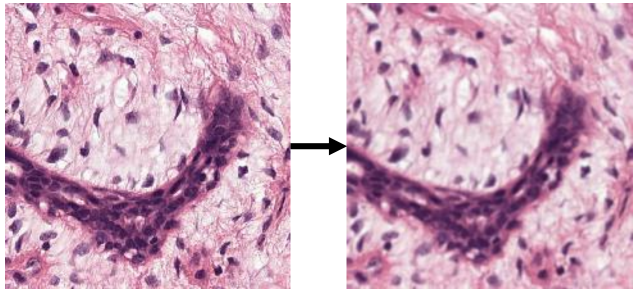
ContrastAugmenter, AdditiveGaussianNoiseAugmenter, GaussianBlurAugmenter, FlipAugmenter, Rotate90Augmenter, ScalingAugmenter



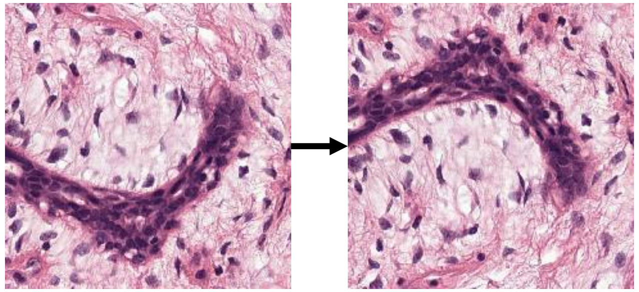
ContrastAugmenter
Contrast adjustment



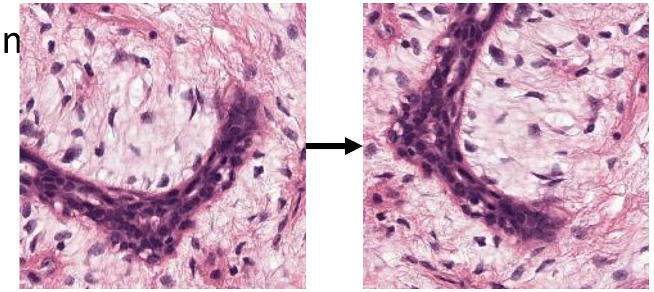
AdditiveGaussianNoiseAugn
Gaussian noise addition



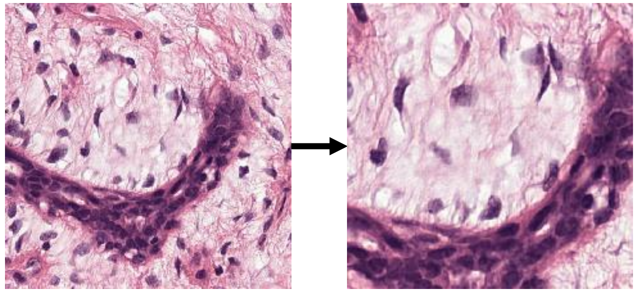
GaussianBlurAugmenter
Gaussian blurring



FlipAugmenter
Flipping



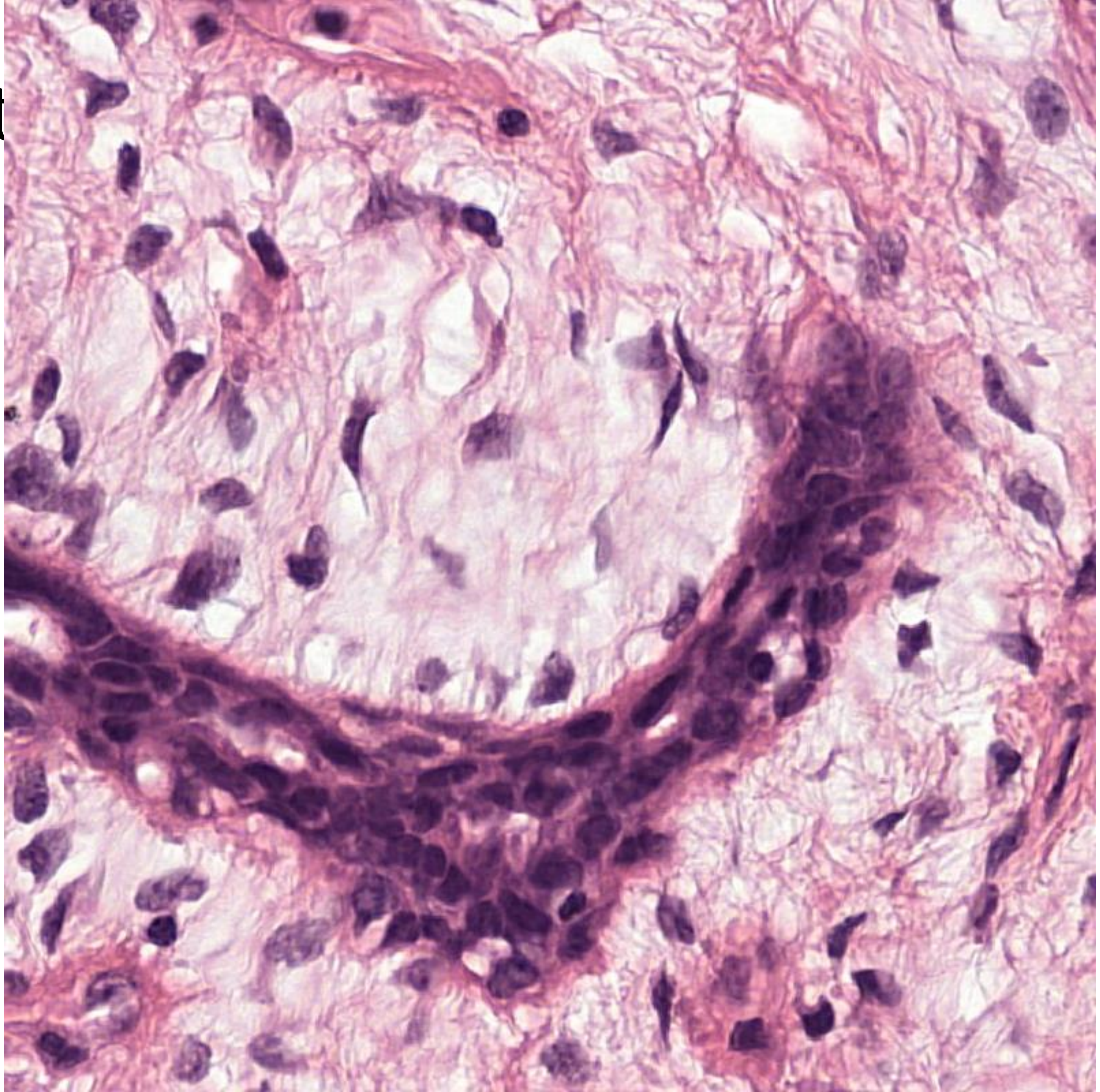
Rotate90Augmenter
Rotate by 90˚, 180˚, 270˚



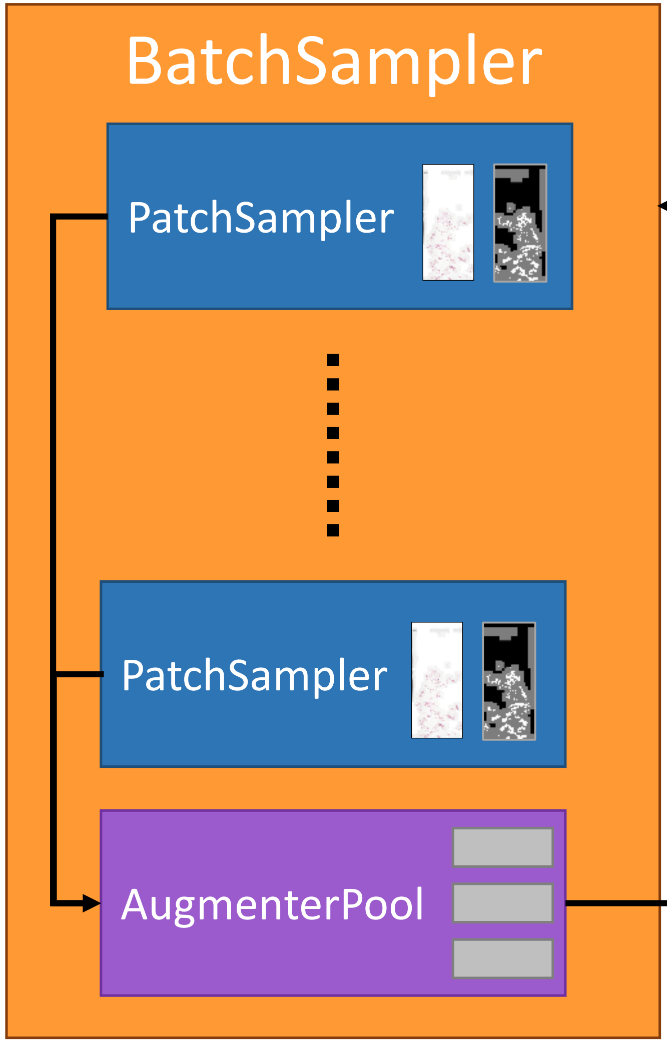
ScalingAugmenter
Scaling

Generat

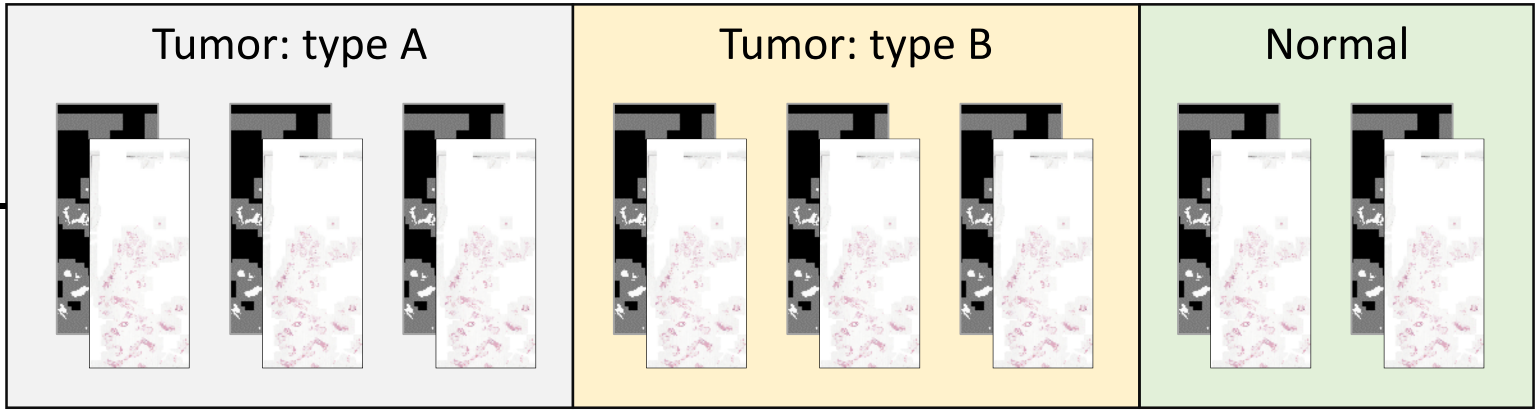
ElasticA



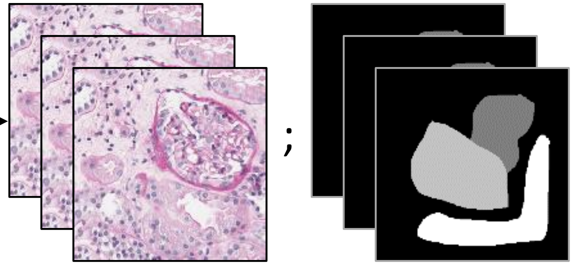
Generating batch of image patches: **BatchSampler**



Data configuration YAML file

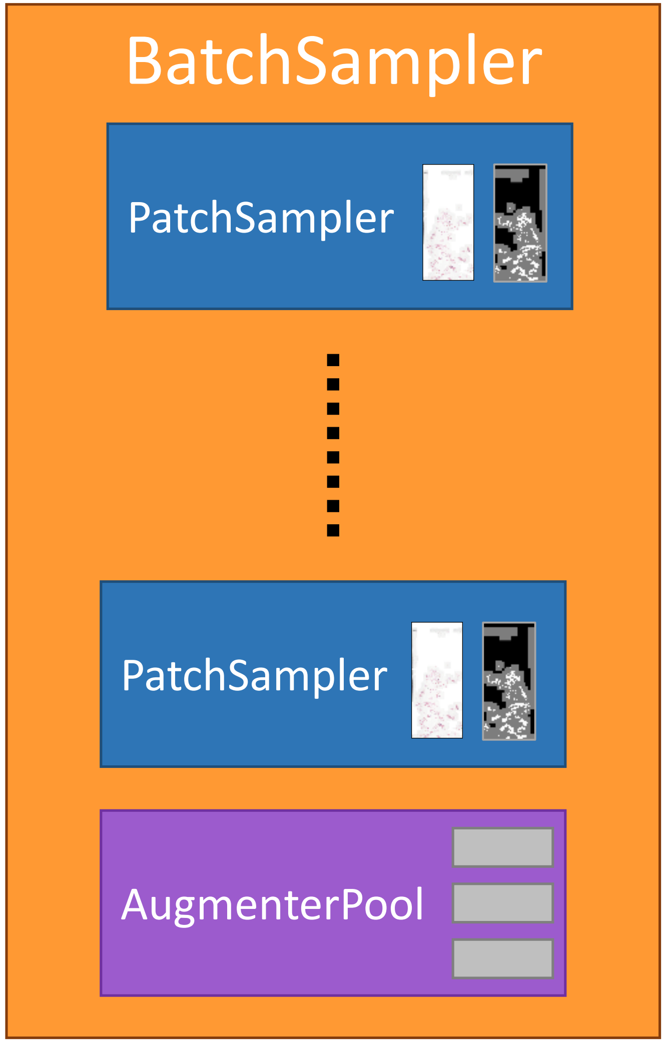


- Most of the **parameters** that the **PatchSampler** has are fixed in **BatchSampler**
- **Multiprocessing:** the patch samplers are delegated to separate processes
- **Pool size:** Number of PatchSampler instances
- **Category distribution:** distribution of images from different categories in the pool of PatchSamplers



or [0, 1, 1]

Generating batch of image patches: **BatchSampler**



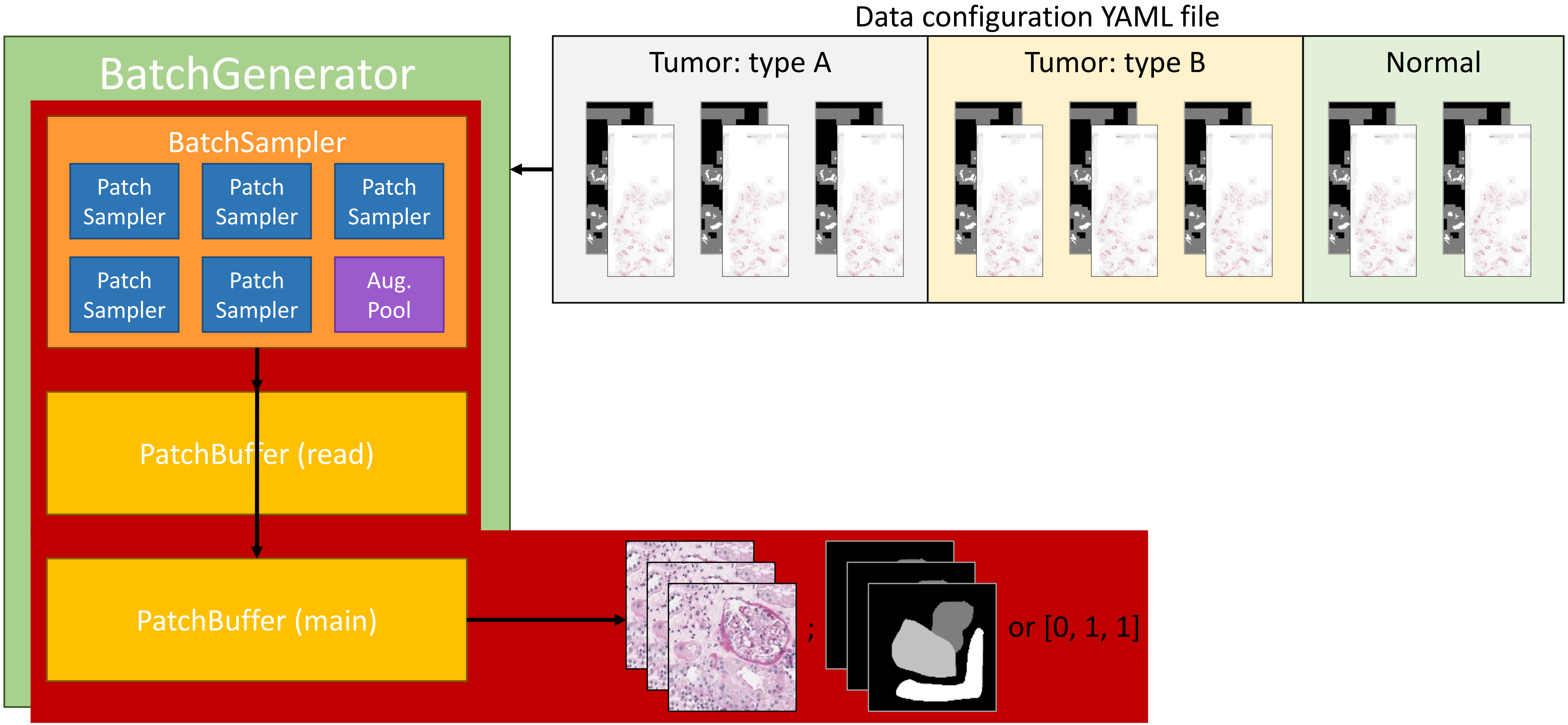
```
def step(self):
```

- After instantiation the BatchSampler is in **invalid state**: it has no PatchSampler instances
- The step function **randomizes the images** to open based on the distribution of categories
- Opens a collection images, the number is limited by the **pool size**

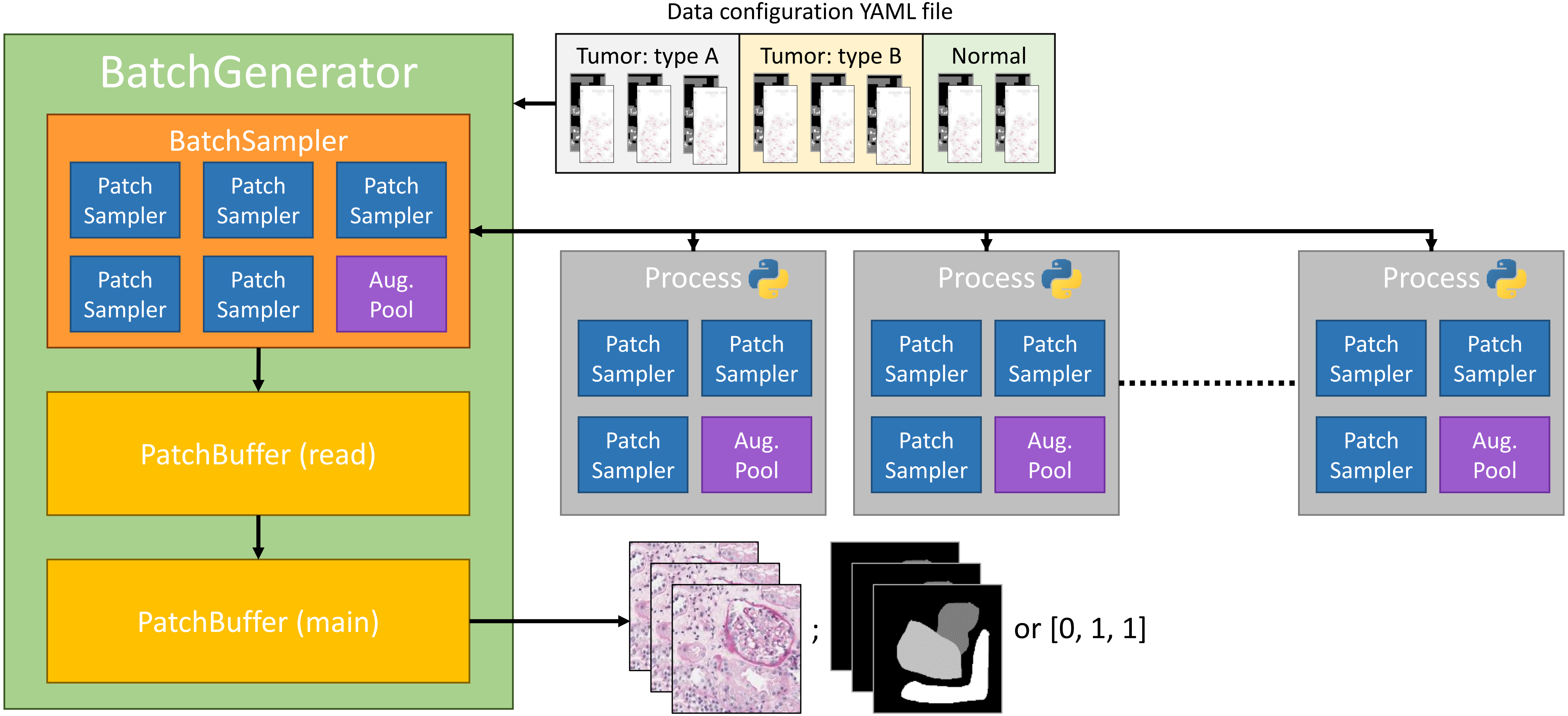
```
def batch(self, batch_size):
```

- **Load a batch of patches** from the pool of PatchSamplers
- All other parameters for the PatchSampler are fixed by the contructor.

Generating batch of image patches: **BatchGenerator**



Generating batch of image patches: **BatchGenerator**



Data configuration YAML file

Tumor: type A

Tumor: type B

Normal

BatchGenerator

BatchSampler

Patch Sampler

Patch Sampler

Patch Sampler

Patch Sampler

Patch Sampler

Aug. Pool

PatchBuffer (read)

PatchBuffer (main)

Process

Patch Sampler

Patch Sampler

Patch Sampler

Aug. Pool

Process

Patch Sampler

Patch Sampler

Patch Sampler

Aug. Pool

Process

Patch Sampler

Patch Sampler

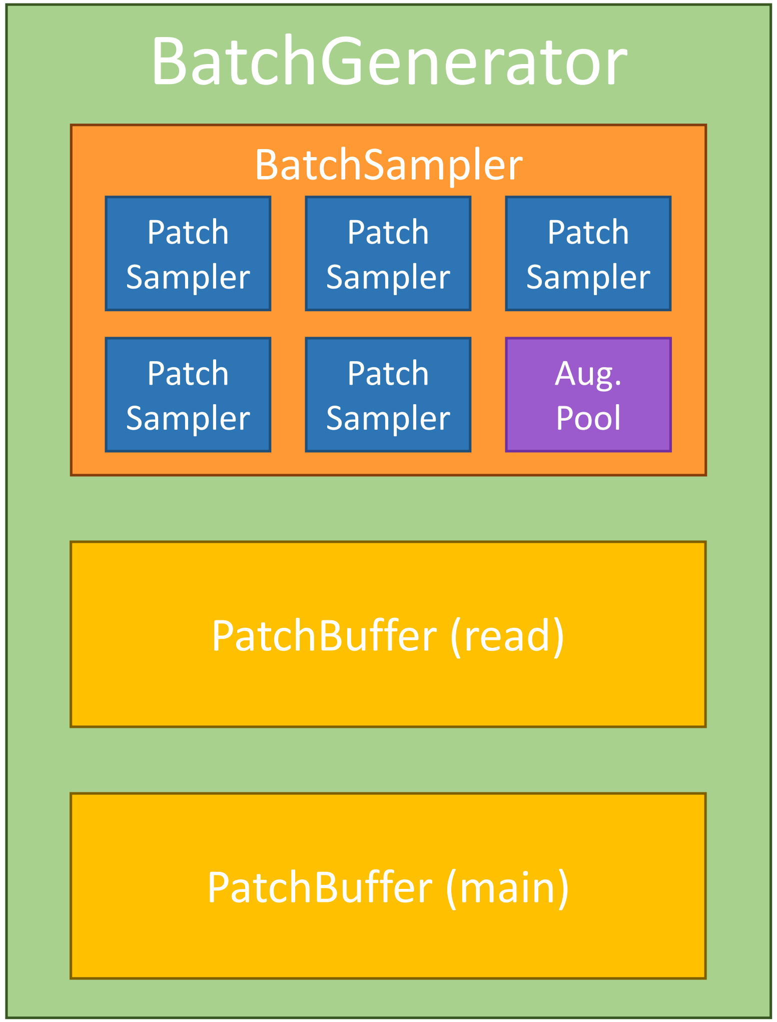
Patch Sampler

Aug. Pool

;

or [0, 1, 1]

Generating batch of image patches: **BatchGenerator**



```
def start(self):
```
→ Instantiate BatchSampler

```
def stop(self):
```
→ Shut down everything

```
def step(self):
```
→ BatchSampler:step, open a randomized set of images

```
def batch(self, batch_size):
```
→ Get a batch from the main buffer

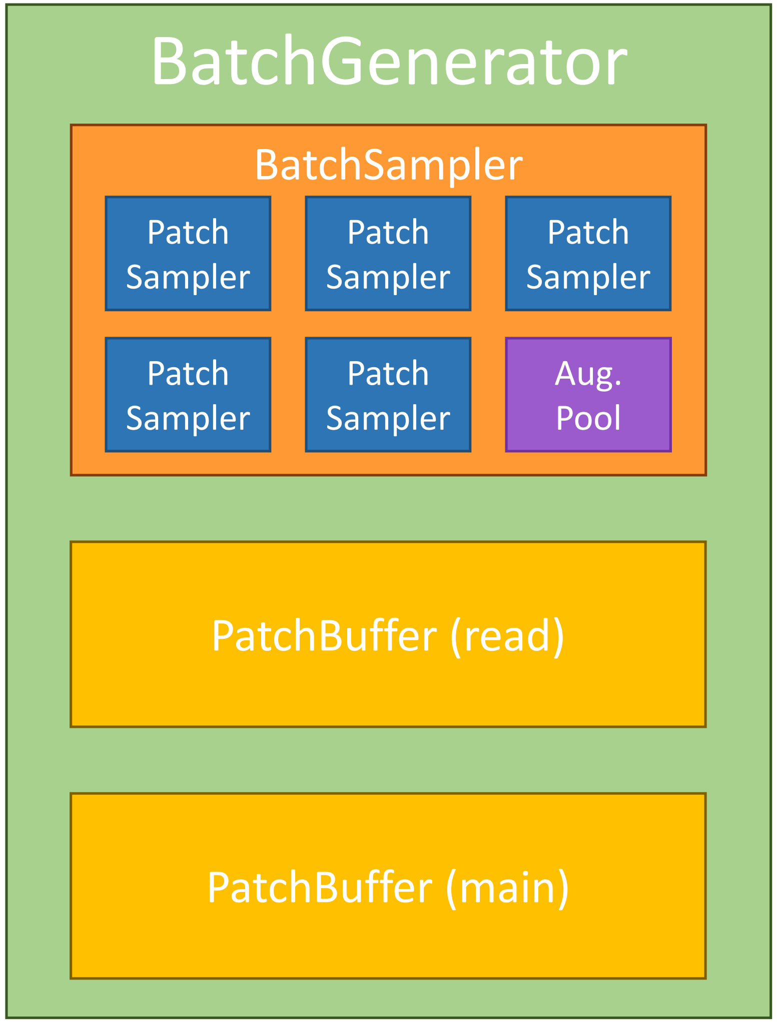
```
def fill(self):
```
→ Fill up the **main** buffer directly

```
def load(self, batch_size=0):
```
→ Load patches to the **read** buffer

```
def transfer(self, batch_size=0, difficult_threshold=0.0):
```
↳ Transfer patches from the **read** to the **main** buffer

```
def wait(self):
```
→ Wait for the previous async job to finish

Generating batch of image patches: **BatchGenerator**



BatchGenerator

BatchSampler

Patch Sampler

Patch Sampler

Patch Sampler

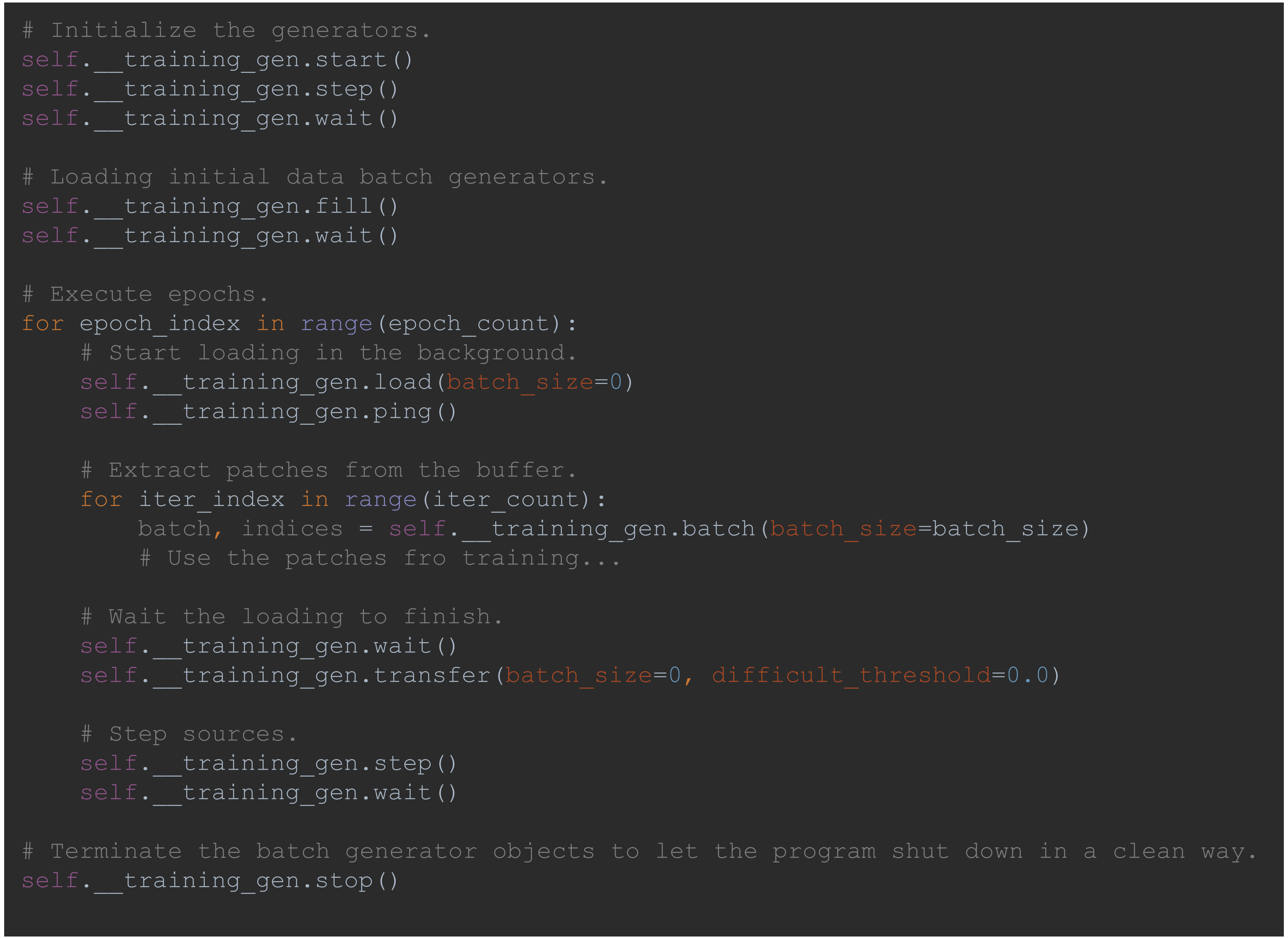
Patch Sampler

Patch Sampler

Aug. Pool

PatchBuffer (read)

PatchBuffer (main)



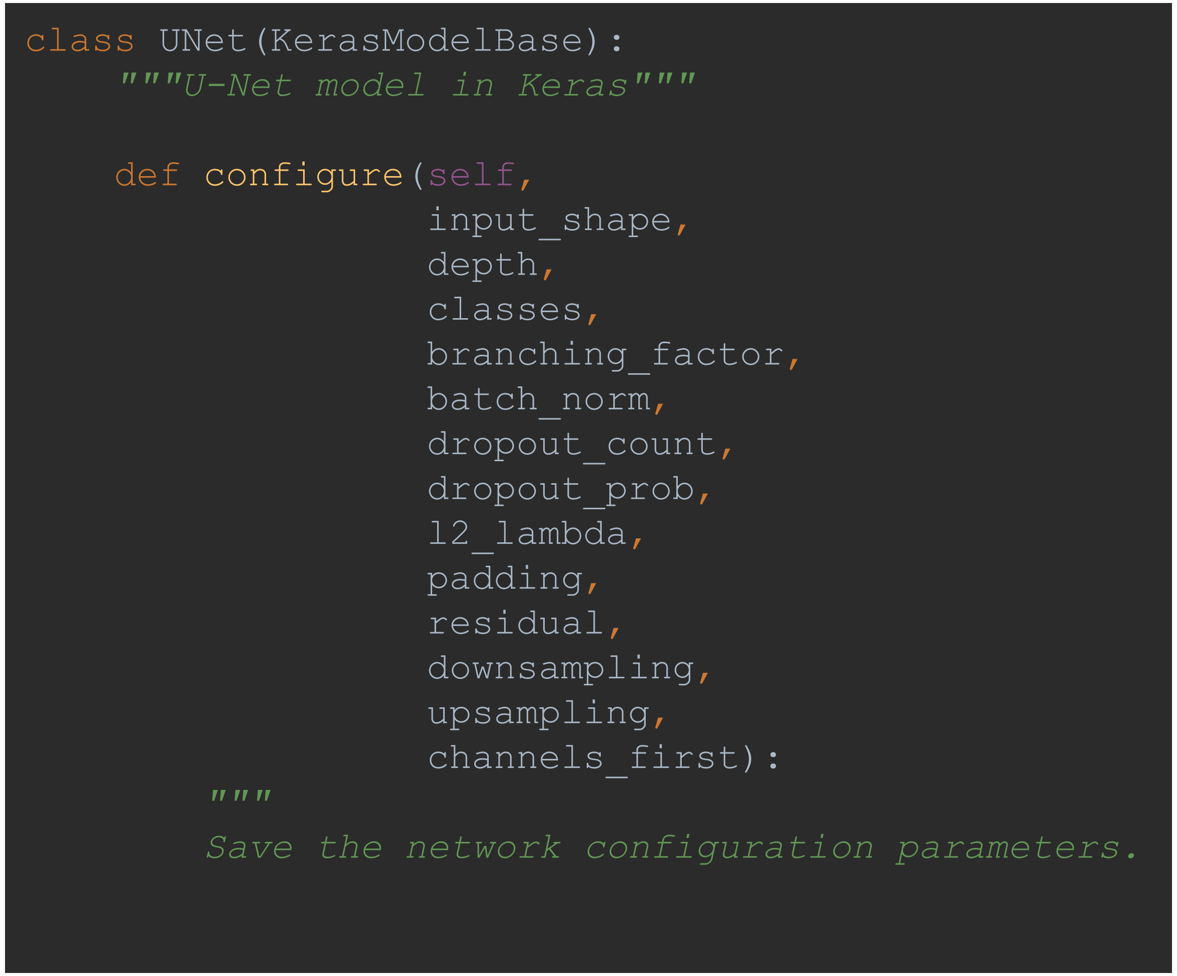
```
# Initialize the generators.
self.__training_gen.start()
self.__training_gen.step()
self.__training_gen.wait()

# Loading initial data batch generators.
self.__training_gen.fill()
self.__training_gen.wait()

# Execute epochs.
for epoch_index in range(epoch_count):
    # Start loading in the background.
    self.__training_gen.load(batch_size=0)
    self.__training_gen.ping()

    # Extract patches from the buffer.
    for iter_index in range(iter_count):
        batch, indices = self.__training_gen.batch(batch_size=batch_size)
        # Use the patches fro training...

    # Wait the loading to finish.
    self.__training_gen.wait()
    self.__training_gen.transfer(batch_size=0, difficult_threshold=0.0)

    # Step sources.
    self.__training_gen.step()
    self.__training_gen.wait()

# Terminate the batch generator objects to let the program shut down in a clean way.
self.__training_gen.stop()
```

2. ModelBase

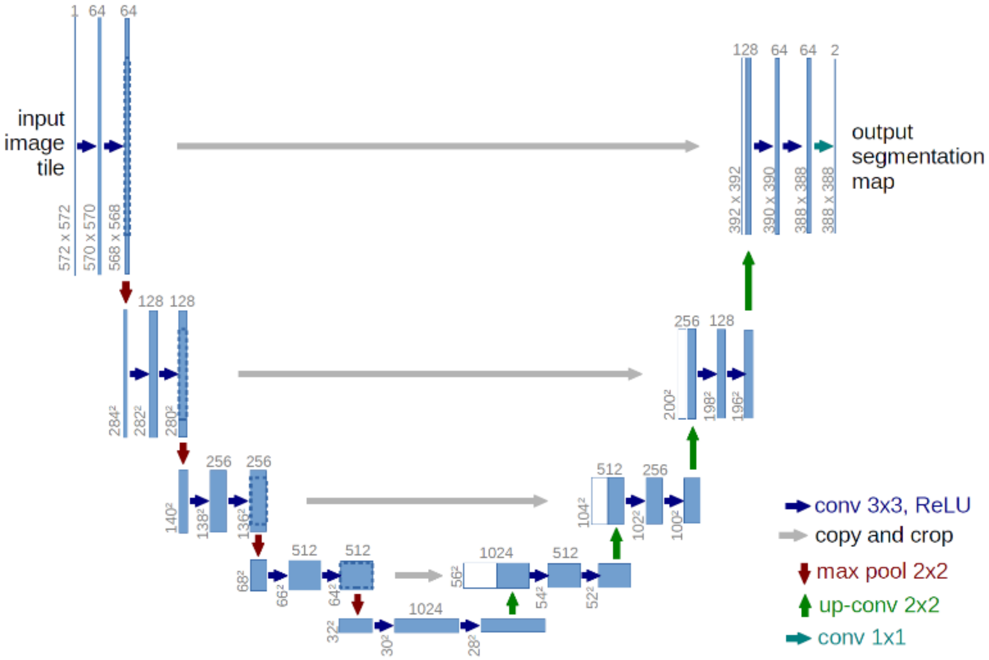
Base class for network models: ModelBase

- Base class for network models
- Wrapper around libraries, currently:
  - Theano/Lasagna (probably already broken)
  - Keras + TensorFlow
- In the future:
  - TensorFlow
  - PyThorc
- The library provides a standard way to **build, save, load, train** and **use** the networks.
- Provides an easy way to configure existing network architectures.

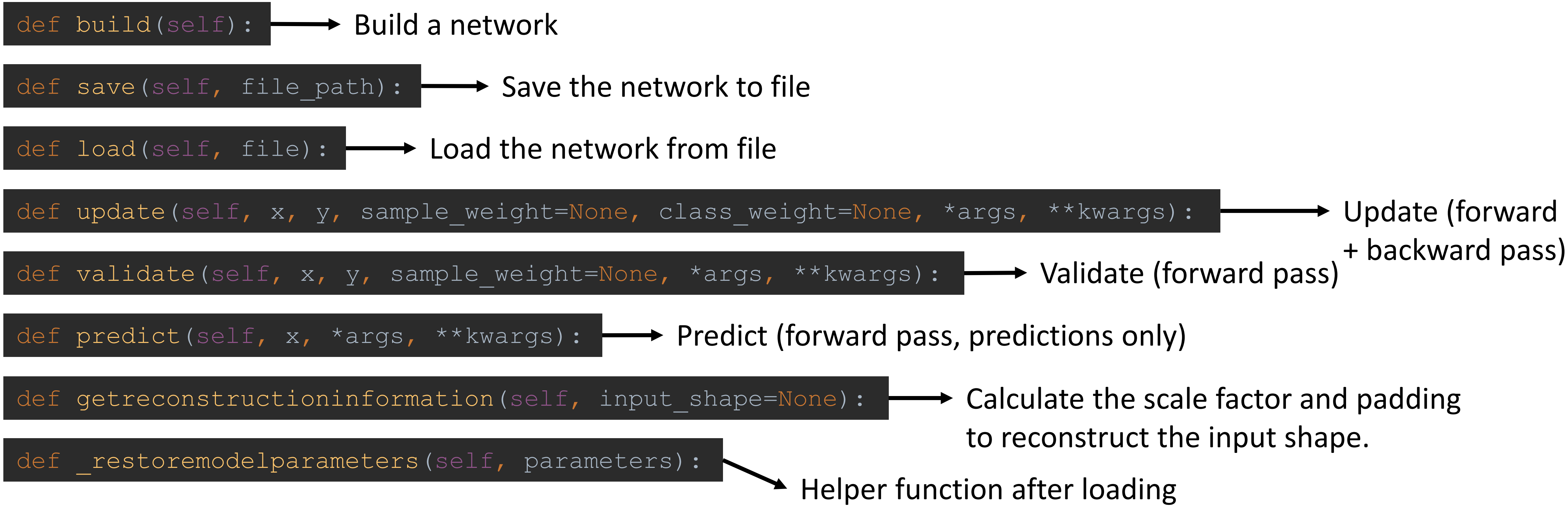
Base class for network models: **ModelBase**



```
class UNet(KerasModelBase):
    """U-Net model in Keras"""

    def configure(self,
                  input_shape,
                  depth,
                  classes,
                  branching_factor,
                  batch_norm,
                  dropout_count,
                  dropout_prob,
                  l2_lambda,
                  padding,
                  residual,
                  downsampling,
                  upsampling,
                  channels_first):
        """
        Save the network configuration parameters.
```



Base class for network models: **ModelBase**



```
def build(self):
```

Build a network

```
def save(self, file_path):
```

Save the network to file

```
def load(self, file):
```

Load the network from file

```
def update(self, x, y, sample_weight=None, class_weight=None, *args, **kwargs):
```

Update (forward + backward pass)

```
def validate(self, x, y, sample_weight=None, *args, **kwargs):
```

Validate (forward pass)

```
def predict(self, x, *args, **kwargs):
```

Predict (forward pass, predictions only)

```
def getreconstructioninformation(self, input_shape=None):
```

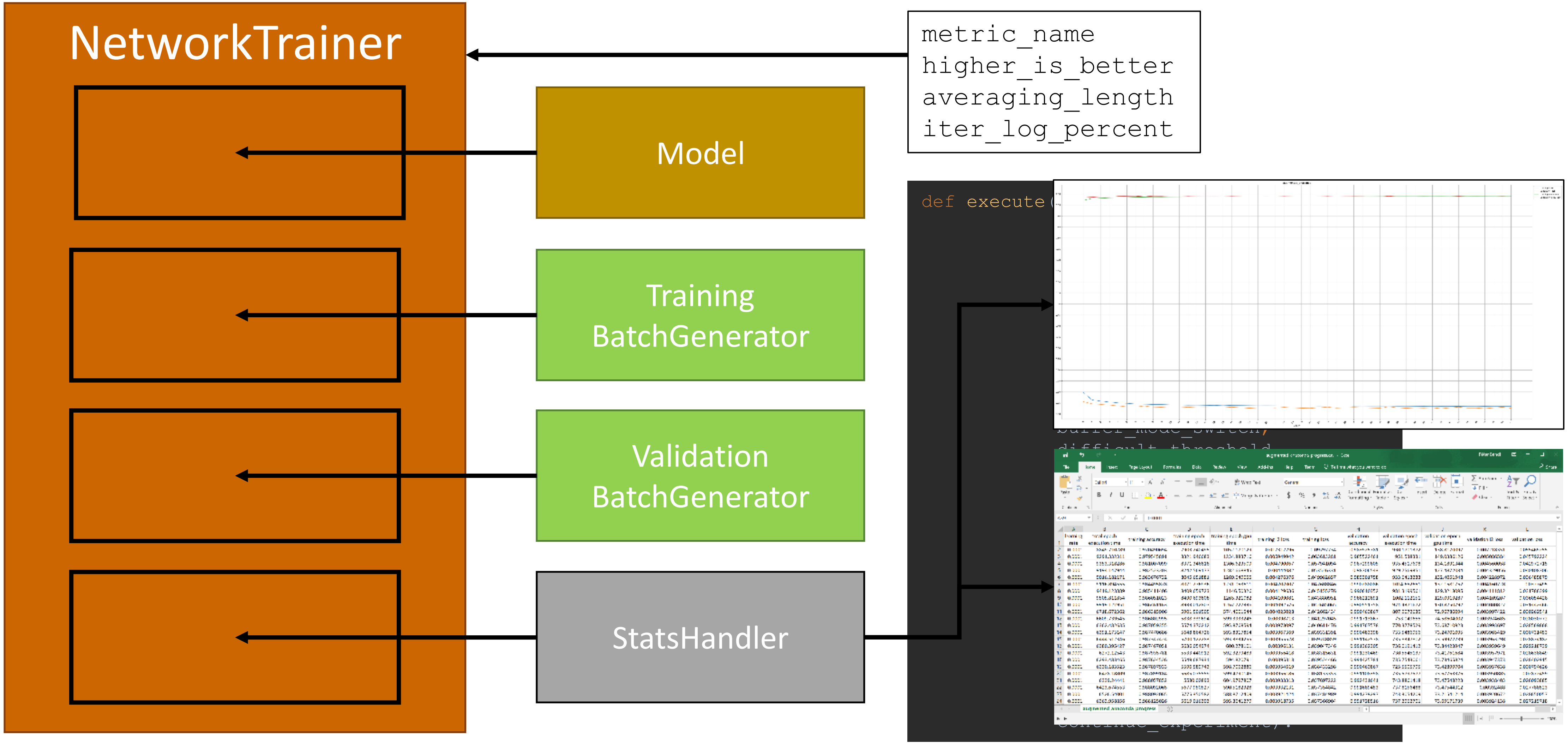
Calculate the scale factor and padding to reconstruct the input shape.

```
def _restoremodelparameters(self, parameters):
```

Helper function after loading

3. NetworkTrainer

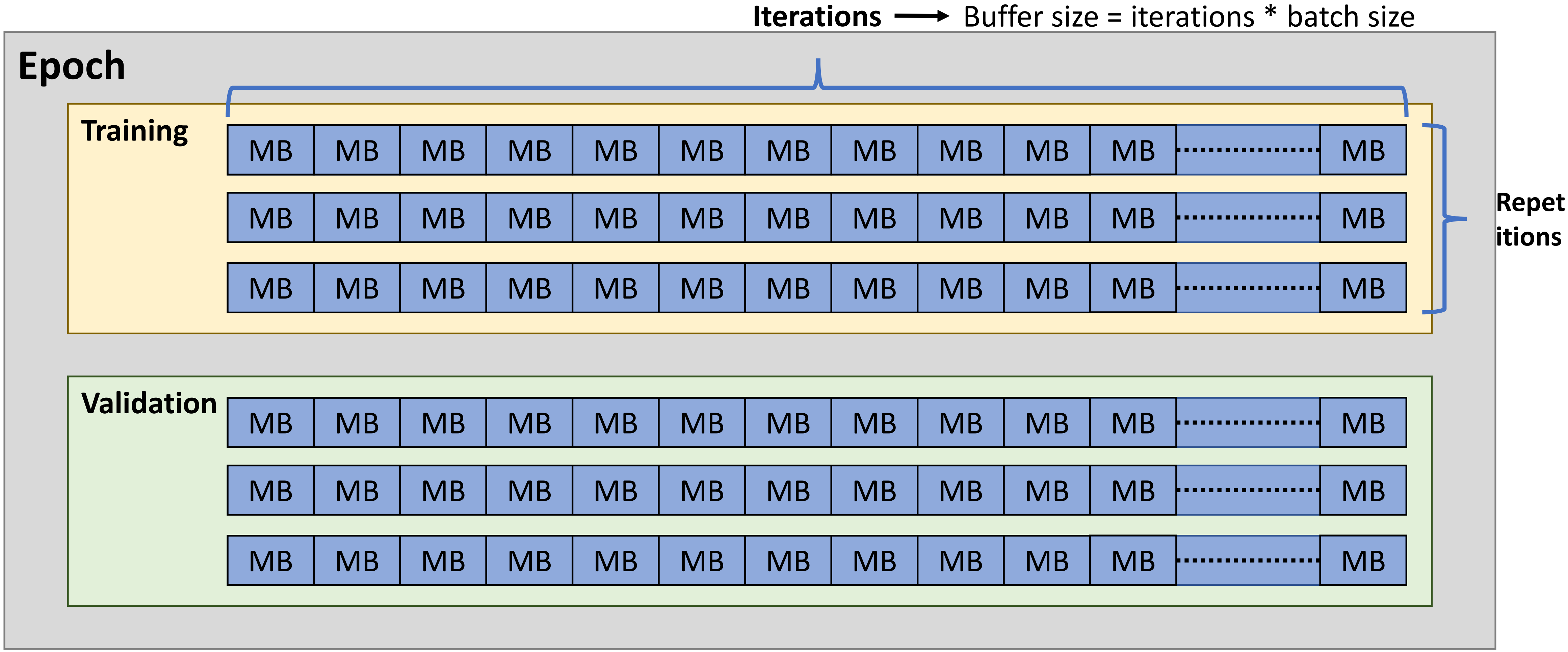
Training networks: **NetworkTrainer**



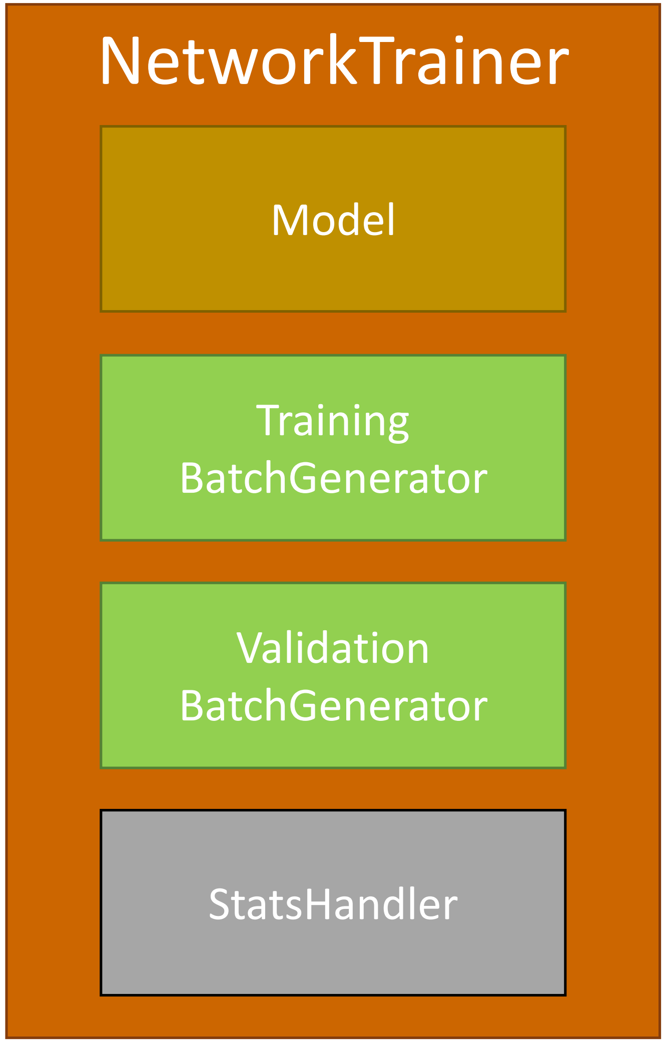
```
metric_name
higher_is_better
averaging_length
iter_log_percent
```

```
def execute(
buffer_mode_switch,
difficult_threshold
continue_experiment):
```

Training networks: **NetworkTrainer**

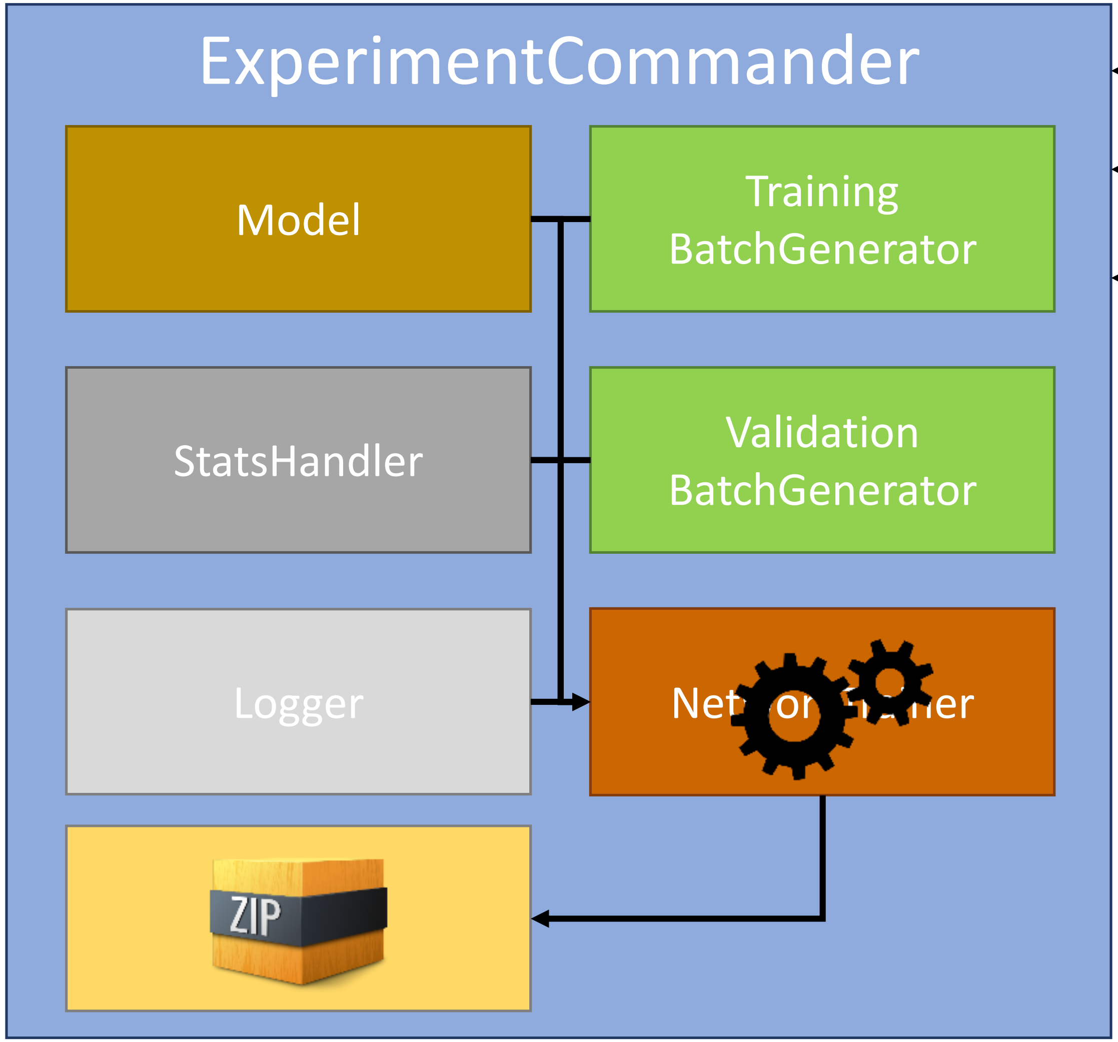


Training networks: **NetworkTrainer**

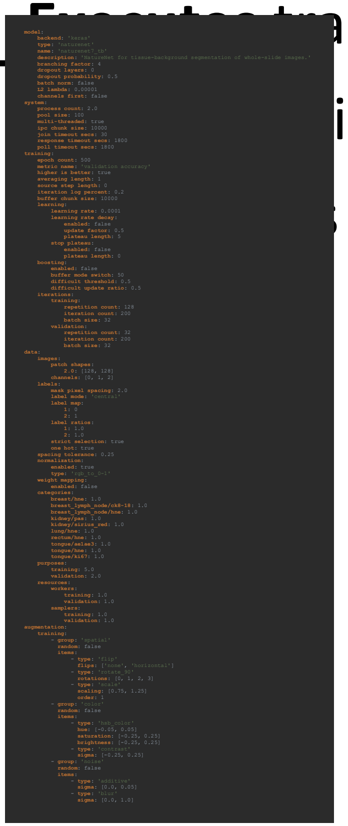


- **Executes** the experiment
- Saves the **best** and the **last** network **model**
- Saves the **metrics** table and the plot
- Saves the **status** of the training
- Can pick up an **continue** an experiment

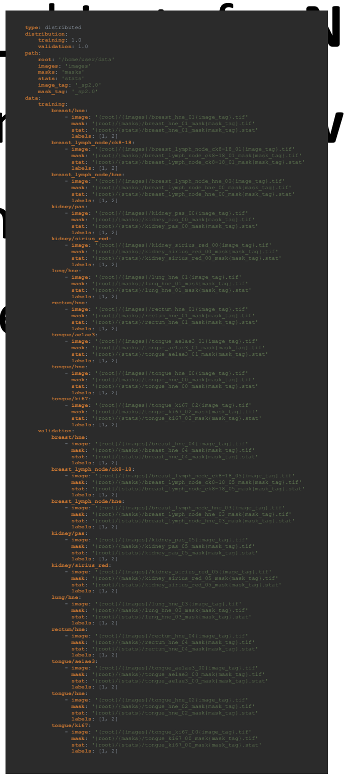
Training networks: **ExperimentCommander**



- **Parses** the data and parameter config files
- **Instantiates** all **NetworkTrainer**
- **Executes** trainin **workTrainer**
- inin
- s eve single **archive**



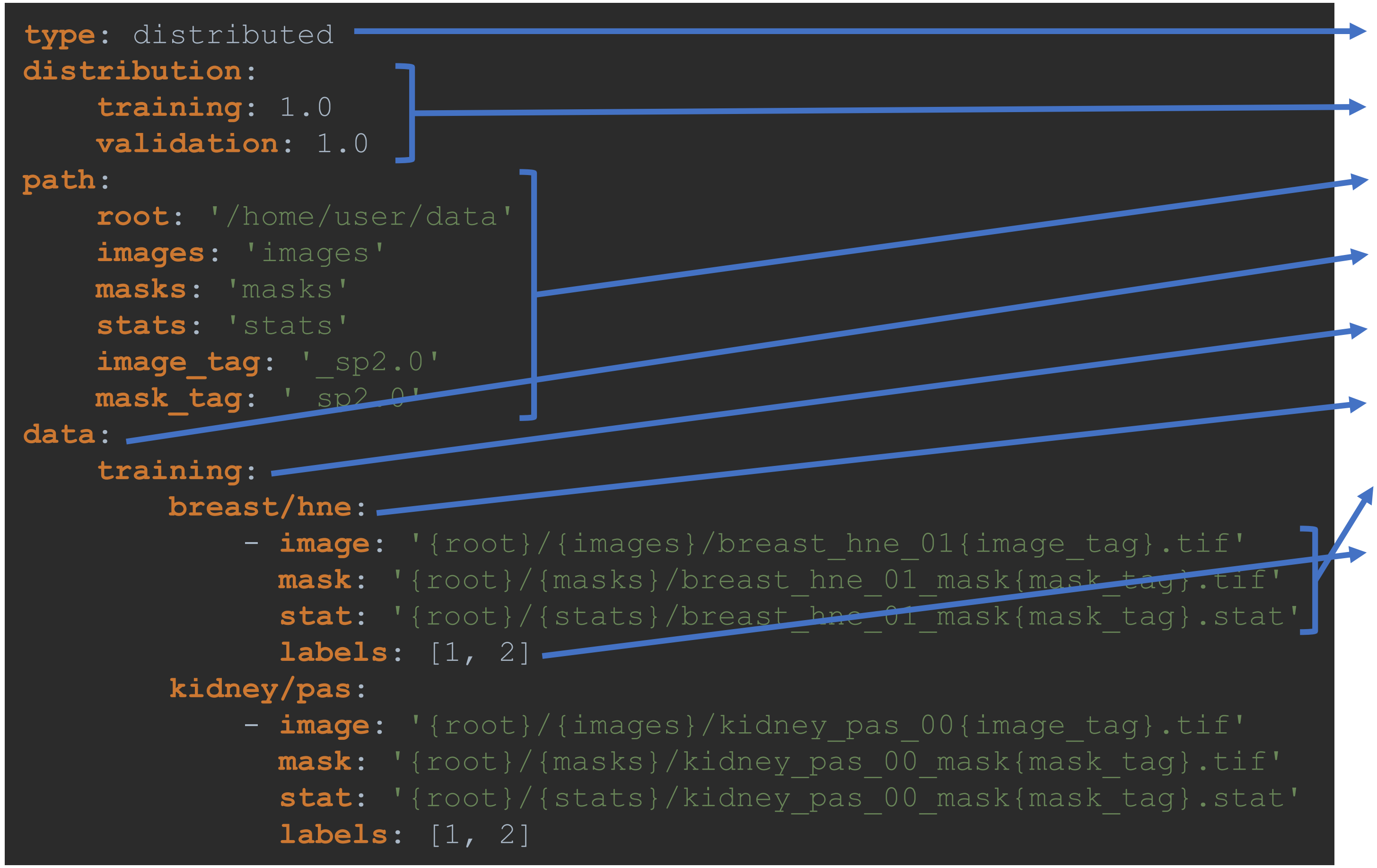
```
model:
    backend: 'keras'
    type: 'naturenet'
    name: 'naturenet7_tb'
    description: 'NatureNet for tissue-background segmentation of whole-slide images.'
    branching factor: 4
    dropout layers: 0
    dropout probability: 0.5
    batch norm: false
    L2 lambda: 0.00001
    channels first: false
system:
    process count: 2.0
    pool size: 100
    multi-threaded: true
    ipc chunk size: 10000
    join timeout secs: 30
    response timeout secs: 1800
    poll timeout secs: 1800
training:
    epoch count: 500
    metric name: 'validation accuracy'
    higher is better: true
    averaging length: 1
    source step length: 0
    iteration log percent: 0.2
    buffer chunk size: 10000
    learning:
        learning rate: 0.0001
        learning rate decay:
            enabled: false
            update factor: 0.5
            plateau length: 5
        stop plateau:
            enabled: false
            plateau length: 0
    boosting:
        enabled: false
        buffer mode switch: 50
        difficult threshold: 0.5
        difficult update ratio: 0.5
    iterations:
        training:
            repetition count: 128
            iteration count: 200
            batch size: 32
        validation:
            repetition count: 32
            iteration count: 200
            batch size: 32
data:
    images:
        patch shapes:
            2.0: [128, 128]
        channels: [0, 1, 2]
    labels:
        mask pixel spacing: 2.0
        label mode: 'central'
        label map:
            1: 0
            2: 1
        label ratios:
            1: 1.0
            2: 1.0
        strict selection: true
        one hot: true
    spacing tolerance: 0.25
    normalization:
        enabled: true
        type: 'rgb_to_0-1'
    weight mapping:
        enabled: false
    categories:
        breast/hne: 1.0
        breast_lymph_node/ck8-18: 1.0
        breast_lymph_node/hne: 1.0
        kidney/pas: 1.0
        kidney/sirius_red: 1.0
        lung/hne: 1.0
        rectum/hne: 1.0
        tongue/ae1ae3: 1.0
        tongue/hne: 1.0
        tongue/ki67: 1.0
    purposes:
        training: 5.0
        validation: 2.0
    resources:
        workers:
            training: 1.0
            validation: 1.0
        samplers:
            training: 1.0
            validation: 1.0
augmentation:
    training:
        - group: 'spatial'
          random: false
          items:
              - type: 'flip'
                flips: ['none', 'horizontal']
              - type: 'rotate_90'
                rotations: [0, 1, 2, 3]
              - type: 'scale'
                scaling: [0.75, 1.25]
                order: 1
        - group: 'color'
          random: false
          items:
              - type: 'hsb_color'
                hue: [-0.05, 0.05]
                saturation: [-0.25, 0.25]
                brightness: [-0.25, 0.25]
              - type: 'contrast'
                sigma: [-0.25, 0.25]
        - group: 'noise'
          random: false
          items:
              - type: 'additive'
                sigma: [0.0, 0.05]
              - type: 'blur'
                sigma: [0.0, 1.0]
```



```
type: distributed
distribution:
    training: 1.0
    validation: 1.0
path:
    root: '/home/user/data'
    images: 'images'
    masks: 'masks'
    stats: 'stats'
    image_tag: '_sp2.0'
    mask_tag: '_sp2.0'
data:
    training:
        breast/hne:
            - image: '{root}/{images}/breast_hne_01{image_tag}.tif'
              mask: '{root}/{masks}/breast_hne_01_mask{mask_tag}.tif'
              stat: '{root}/{stats}/breast_hne_01_mask{mask_tag}.stat'
              labels: [1, 2]
        breast_lymph_node/ck8-18:
            - image: '{root}/{images}/breast_lymph_node_ck8-18_01{image_tag}.tif'
              mask: '{root}/{masks}/breast_lymph_node_ck8-18_01_mask{mask_tag}.tif'
              stat: '{root}/{stats}/breast_lymph_node_ck8-18_01_mask{mask_tag}.stat'
              labels: [1, 2]
        breast_lymph_node/hne:
            - image: '{root}/{images}/breast_lymph_node_hne_00{image_tag}.tif'
              mask: '{root}/{masks}/breast_lymph_node_hne_00_mask{mask_tag}.tif'
              stat: '{root}/{stats}/breast_lymph_node_hne_00_mask{mask_tag}.stat'
              labels: [1, 2]
        kidney/pas:
            - image: '{root}/{images}/kidney_pas_00{image_tag}.tif'
              mask: '{root}/{masks}/kidney_pas_00_mask{mask_tag}.tif'
              stat: '{root}/{stats}/kidney_pas_00_mask{mask_tag}.stat'
              labels: [1, 2]
        kidney/sirius_red:
            - image: '{root}/{images}/kidney_sirius_red_00{image_tag}.tif'
              mask: '{root}/{masks}/kidney_sirius_red_00_mask{mask_tag}.tif'
              stat: '{root}/{stats}/kidney_sirius_red_00_mask{mask_tag}.stat'
              labels: [1, 2]
        lung/hne:
            - image: '{root}/{images}/lung_hne_01{image_tag}.tif'
              mask: '{root}/{masks}/lung_hne_01_mask{mask_tag}.tif'
              stat: '{root}/{stats}/lung_hne_01_mask{mask_tag}.stat'
              labels: [1, 2]
        rectum/hne:
            - image: '{root}/{images}/rectum_hne_01{image_tag}.tif'
              mask: '{root}/{masks}/rectum_hne_01_mask{mask_tag}.tif'
              stat: '{root}/{stats}/rectum_hne_01_mask{mask_tag}.stat'
              labels: [1, 2]
        tongue/ae1ae3:
            - image: '{root}/{images}/tongue_ae1ae3_01{image_tag}.tif'
              mask: '{root}/{masks}/tongue_ae1ae3_01_mask{mask_tag}.tif'
              stat: '{root}/{stats}/tongue_ae1ae3_01_mask{mask_tag}.stat'
              labels: [1, 2]
        tongue/hne:
            - image: '{root}/{images}/tongue_hne_00{image_tag}.tif'
              mask: '{root}/{masks}/tongue_hne_00_mask{mask_tag}.tif'
              stat: '{root}/{stats}/tongue_hne_00_mask{mask_tag}.stat'
              labels: [1, 2]
        tongue/ki67:
            - image: '{root}/{images}/tongue_ki67_02{image_tag}.tif'
              mask: '{root}/{masks}/tongue_ki67_02_mask{mask_tag}.tif'
              stat: '{root}/{stats}/tongue_ki67_02_mask{mask_tag}.stat'
              labels: [1, 2]
    validation:
        breast/hne:
            - image: '{root}/{images}/breast_hne_04{image_tag}.tif'
              mask: '{root}/{masks}/breast_hne_04_mask{mask_tag}.tif'
              stat: '{root}/{stats}/breast_hne_04_mask{mask_tag}.stat'
              labels: [1, 2]
        breast_lymph_node/ck8-18:
            - image: '{root}/{images}/breast_lymph_node_ck8-18_05{image_tag}.tif'
              mask: '{root}/{masks}/breast_lymph_node_ck8-18_05_mask{mask_tag}.tif'
              stat: '{root}/{stats}/breast_lymph_node_ck8-18_05_mask{mask_tag}.stat'
              labels: [1, 2]
        breast_lymph_node/hne:
            - image: '{root}/{images}/breast_lymph_node_hne_03{image_tag}.tif'
              mask: '{root}/{masks}/breast_lymph_node_hne_03_mask{mask_tag}.tif'
              stat: '{root}/{stats}/breast_lymph_node_hne_03_mask{mask_tag}.stat'
              labels: [1, 2]
        kidney/pas:
            - image: '{root}/{images}/kidney_pas_05{image_tag}.tif'
              mask: '{root}/{masks}/kidney_pas_05_mask{mask_tag}.tif'
              stat: '{root}/{stats}/kidney_pas_05_mask{mask_tag}.stat'
              labels: [1, 2]
        kidney/sirius_red:
            - image: '{root}/{images}/kidney_sirius_red_05{image_tag}.tif'
              mask: '{root}/{masks}/kidney_sirius_red_05_mask{mask_tag}.tif'
              stat: '{root}/{stats}/kidney_sirius_red_05_mask{mask_tag}.stat'
              labels: [1, 2]
        lung/hne:
            - image: '{root}/{images}/lung_hne_03{image_tag}.tif'
              mask: '{root}/{masks}/lung_hne_03_mask{mask_tag}.tif'
              stat: '{root}/{stats}/lung_hne_03_mask{mask_tag}.stat'
              labels: [1, 2]
        rectum/hne:
            - image: '{root}/{images}/rectum_hne_04{image_tag}.tif'
              mask: '{root}/{masks}/rectum_hne_04_mask{mask_tag}.tif'
              stat: '{root}/{stats}/rectum_hne_04_mask{mask_tag}.stat'
              labels: [1, 2]
        tongue/ae1ae3:
            - image: '{root}/{images}/tongue_ae1ae3_00{image_tag}.tif'
              mask: '{root}/{masks}/tongue_ae1ae3_00_mask{mask_tag}.tif'
              stat: '{root}/{stats}/tongue_ae1ae3_00_mask{mask_tag}.stat'
              labels: [1, 2]
        tongue/hne:
            - image: '{root}/{images}/tongue_hne_02{image_tag}.tif'
              mask: '{root}/{masks}/tongue_hne_02_mask{mask_tag}.tif'
              stat: '{root}/{stats}/tongue_hne_02_mask{mask_tag}.stat'
              labels: [1, 2]
        tongue/ki67:
            - image: '{root}/{images}/tongue_ki67_00{image_tag}.tif'
              mask: '{root}/{masks}/tongue_ki67_00_mask{mask_tag}.tif'
              stat: '{root}/{stats}/tongue_ki67_00_mask{mask_tag}.stat'
              labels: [1, 2]
```

experiment_name
progress_dir_path
archive_dir_path
data_file_path
data_overrides
data_copy_config
param_file_path
param_overrides
continue_experiment
random_seed
cpu_count_enforce
caller_dir_paths
logging_scope
file_log_level
console_log_level

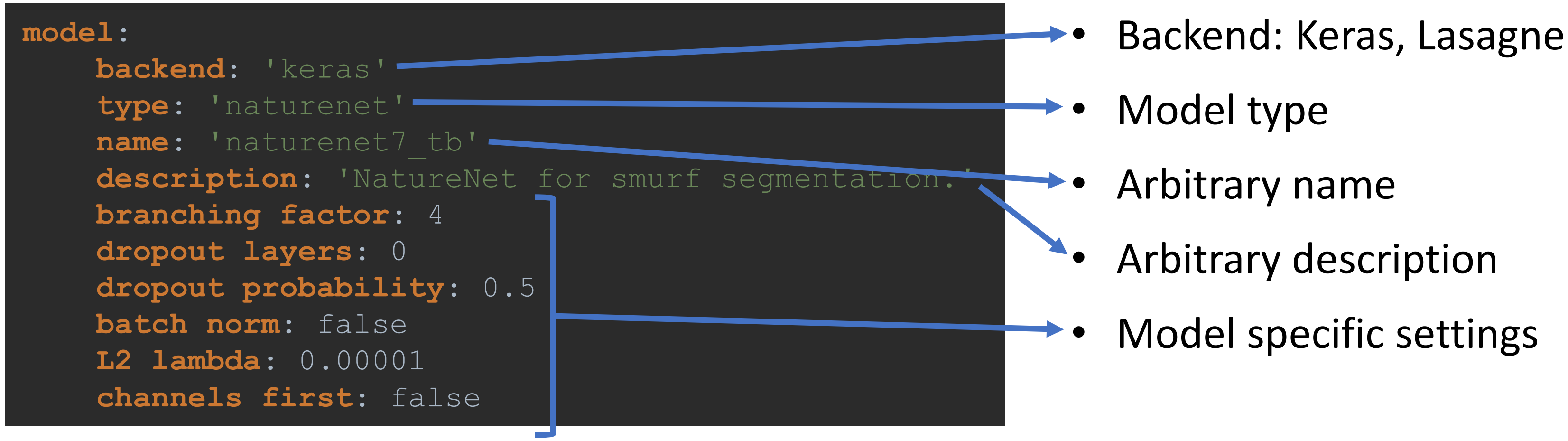
Training networks: **data config**



```yaml
type: distributed
distribution:
    training: 1.0
    validation: 1.0
path:
    root: '/home/user/data'
    images: 'images'
    masks: 'masks'
    stats: 'stats'
    image_tag: '_sp2.0'
    mask_tag: '_sp2.0'
data:
    training:
        breast/hne:
            - image: '{root}/{images}/breast_hne_01{image_tag}.tif'
              mask: '{root}/{masks}/breast_hne_01_mask{mask_tag}.tif'
              stat: '{root}/{stats}/breast_hne_01_mask{mask_tag}.stat'
              labels: [1, 2]
        kidney/pas:
            - image: '{root}/{images}/kidney_pas_00{image_tag}.tif'
              mask: '{root}/{masks}/kidney_pas_00_mask{mask_tag}.tif'
              stat: '{root}/{stats}/kidney_pas_00_mask{mask_tag}.stat'
              labels: [1, 2]
```

- Distributed, list (e.g. training, validation)
- Distribution ratio
- Replacement strings
- Data part
- Purpose: “training”
- Class: “breast/hne”
- Image, mask, stat files
- Labels in mask

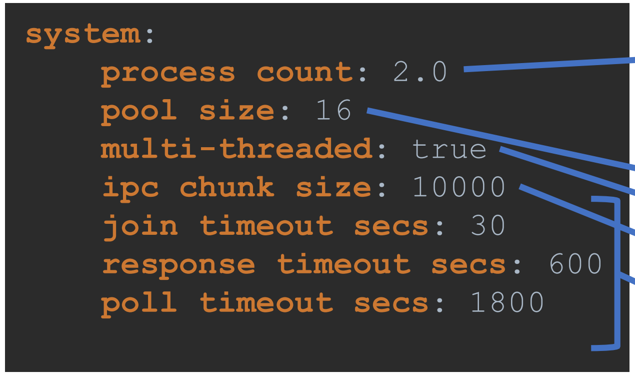
Training networks: **parameters: model**



```
model:
    backend: 'keras'
    type: 'naturenet'
    name: 'naturenet7_tb'
    description: 'NatureNet for smurf segmentation.'
    branching factor: 4
    dropout layers: 0
    dropout probability: 0.5
    batch norm: false
    L2 lambda: 0.00001
    channels first: false
```

- Backend: Keras, Lasagne
- Model type
- Arbitrary name
- Arbitrary description
- Model specific settings

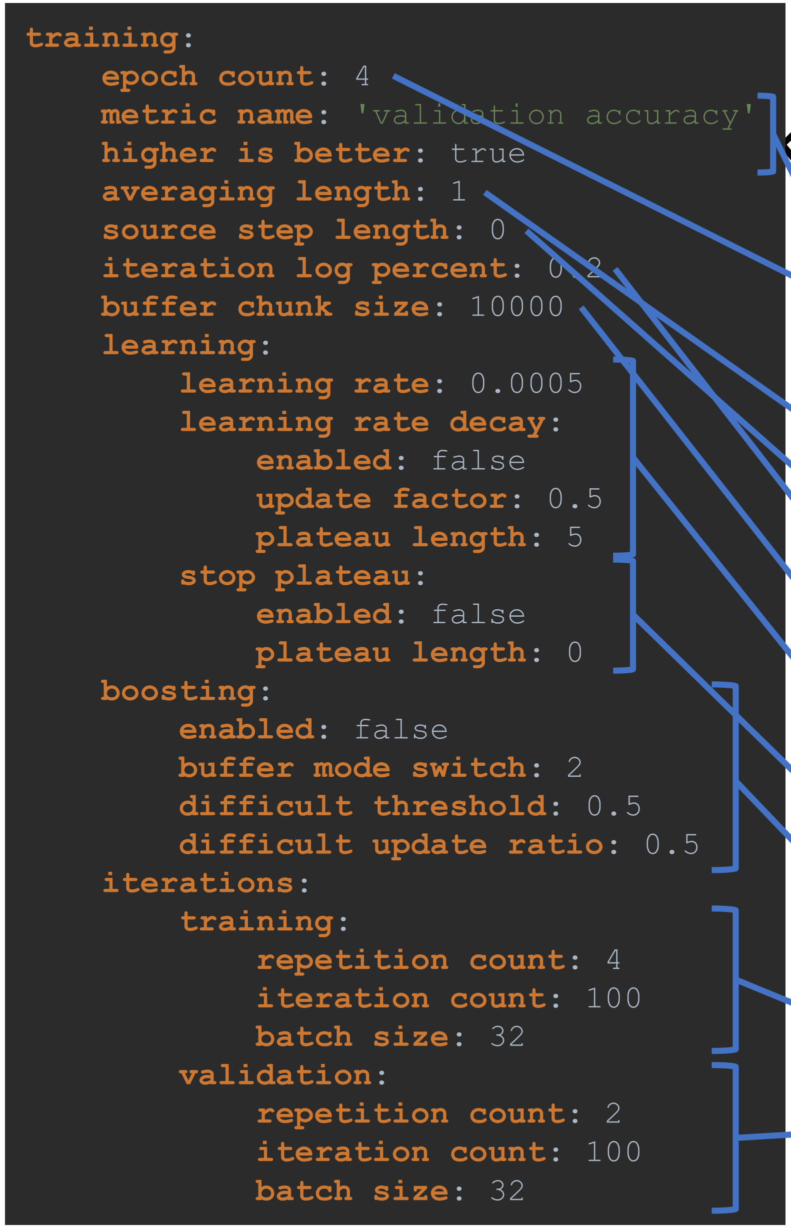
Training networks: **parameters: system**



```
system:
    process count: 2.0
    pool size: 16
    multi-threaded: true
    ipc chunk size: 10000
    join timeout secs: 30
    response timeout secs: 600
    poll timeout secs: 1800
```

- Worker process count
  - Float: multiplier for CPU count
  - Integer: fixed number
  - Use **--cpu** option with float
- Pool size: Max number of PatchSampler objects at any time
- Multi threaded: BatchSampler runs on separate thread
- Chunk size for transferring patches between processes
- Timeouts to prevent the docker hanging

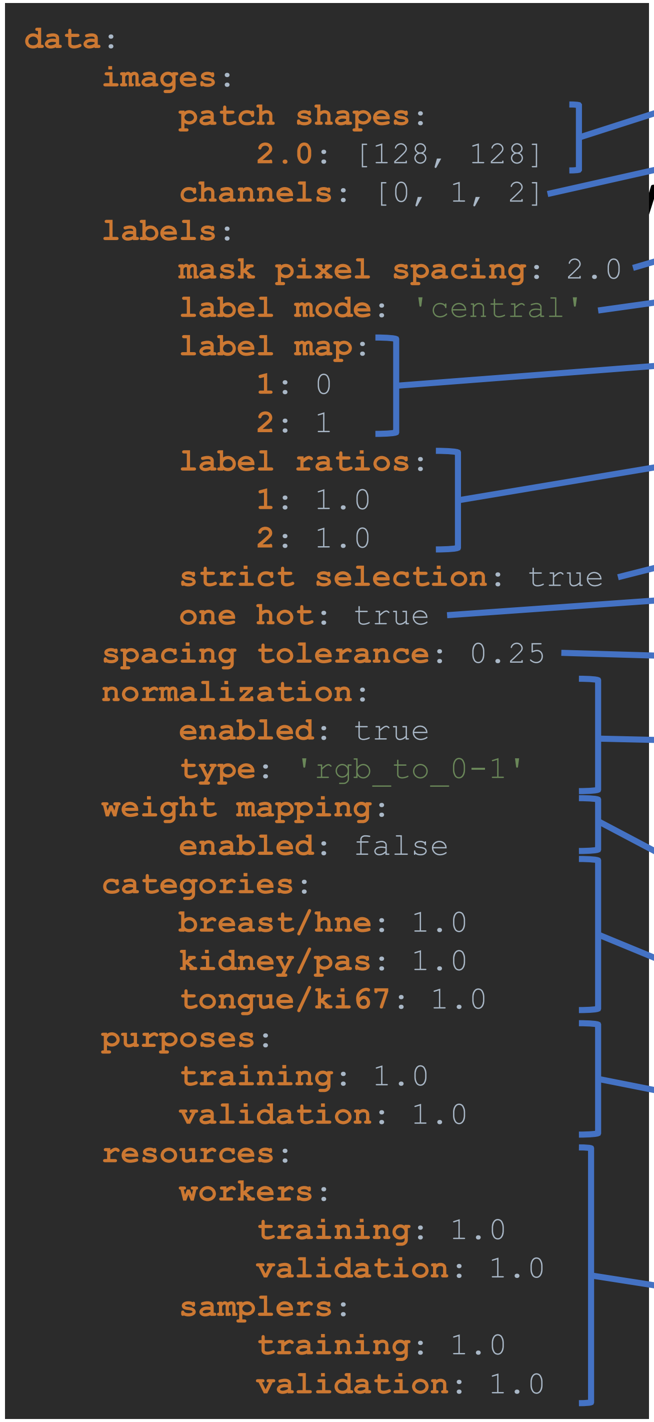
ks: parameters: training



```
training:
    epoch count: 4
    metric name: 'validation accuracy'
    higher is better: true
    averaging length: 1
    source step length: 0
    iteration log percent: 0.2
    buffer chunk size: 10000
    learning:
        learning rate: 0.0005
        learning rate decay:
            enabled: false
            update factor: 0.5
            plateau length: 5
        stop plateau:
            enabled: false
            plateau length: 0
    boosting:
        enabled: false
        buffer mode switch: 2
        difficult threshold: 0.5
        difficult update ratio: 0.5
    iterations:
        training:
            repetition count: 4
            iteration count: 100
            batch size: 32
        validation:
            repetition count: 2
            iteration count: 100
            batch size: 32
```

- Epoch count
- Metric name, higher is better
- Average the last X epoch of the metric
- Step the sources in every X epoch
- Log (and print) the progress in every 20% of the iterations
- Chunk size for transferring the read buffer to the main
- Learning rate, and learning rate decay settings
- Stop criteria
- Boosting settings
- Training iteration and repetition counts and batch sizes
- Validation iteration and repetition counts and batch sizes

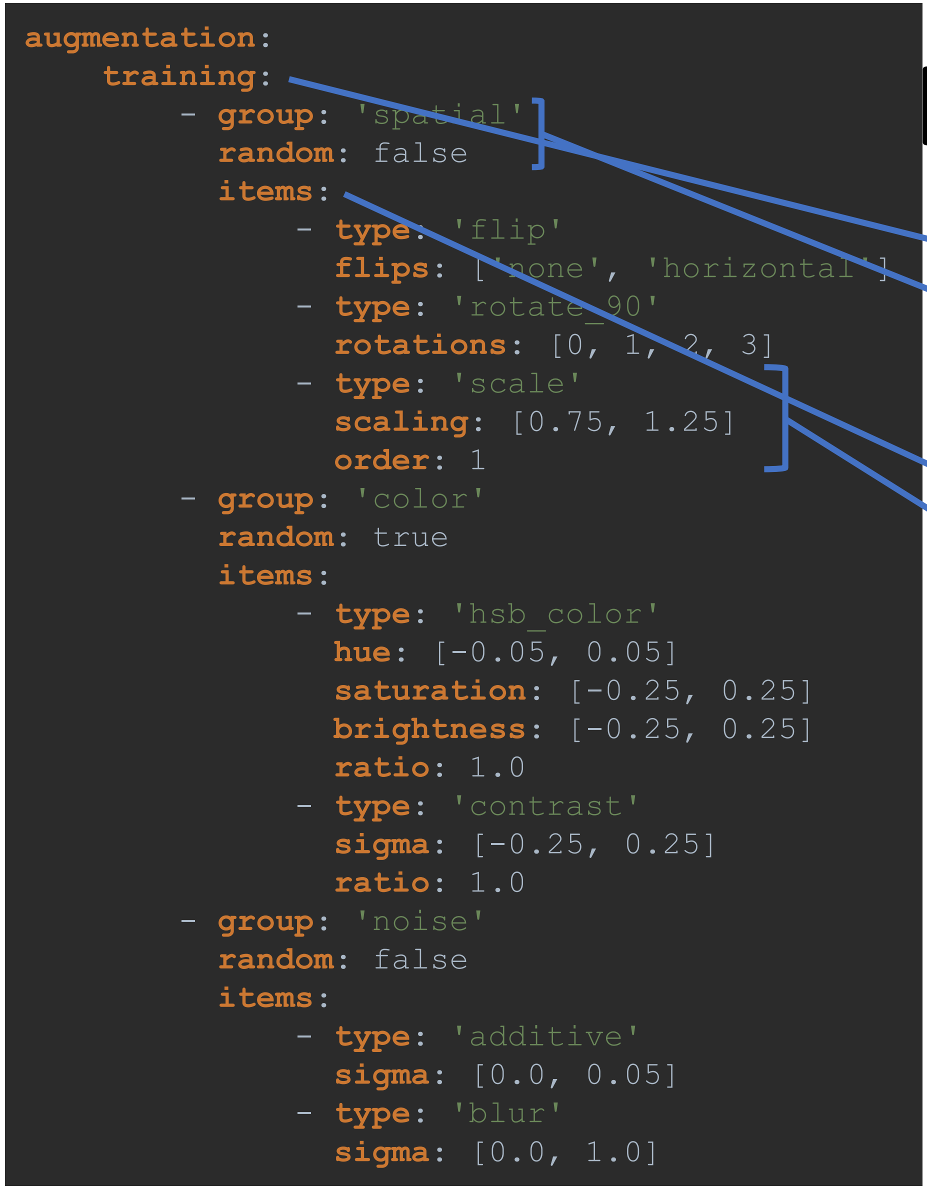
orks: parameters: data



```
data:
    images:
        patch shapes:
            2.0: [128, 128]
        channels: [0, 1, 2]
    labels:
        mask pixel spacing: 2.0
        label mode: 'central'
        label map:
            1: 0
            2: 1
        label ratios:
            1: 1.0
            2: 1.0
        strict selection: true
        one hot: true
    spacing tolerance: 0.25
    normalization:
        enabled: true
        type: 'rgb_to_0-1'
    weight mapping:
        enabled: false
    categories:
        breast/hne: 1.0
        kidney/pas: 1.0
        tongue/ki67: 1.0
    purposes:
        training: 1.0
        validation: 1.0
    resources:
        workers:
            training: 1.0
            validation: 1.0
        samplers:
            training: 1.0
            validation: 1.0
```

- Patch shapes, per pixel spacing (same location)
- Channels to extract from image
- Mask processing pixel spacing
- Label mode: **central, synthetize,** or **load**
- Label mapping from mask values to network values
- Label ratios in buffer (~batch)
- Check if all labels can be load, at all time
- Convert labels to **one hot** representation
- Spacing tolerance (percentage)
- Normalization of image values:
    - `rgb_to_0-1`: [0, 255] → [0.0, 1.0]
    - `rgb`: [0, 255] → [?, ?]
    - `general`: [?, ?] → [?, ?]
- Weight mapping (for U-Net like networks)
- **Distribution of patches** in the buffer from categories
- **Distribution of images** for different purposes. The input data YAML file should be either be list type, or the ratios should match the ratios in the YAML file.
- **Distribution of resources** (workers: processes, sampler: PatchSampler objects) for different purposes.

parameters: augmentation

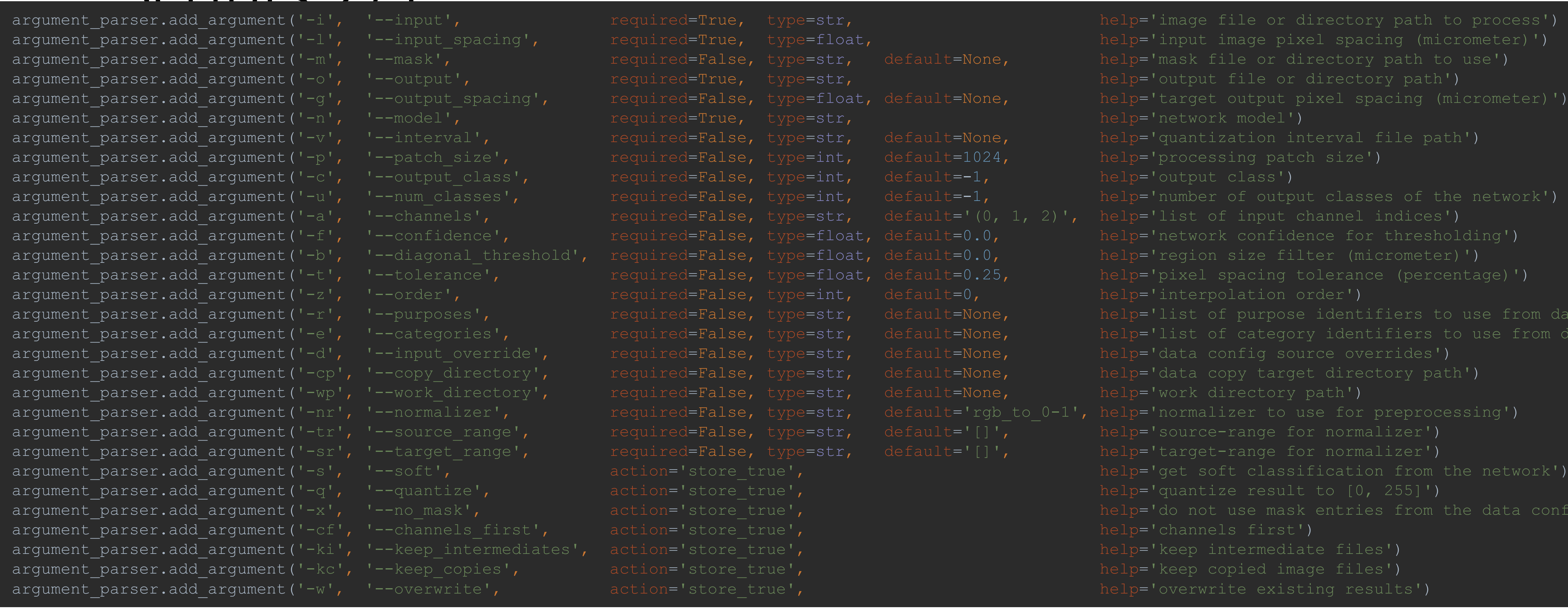


```
augmentation:
    training:
        - group: 'spatial'
          random: false
          items:
              - type: 'flip'
                flips: ['none', 'horizontal']
              - type: 'rotate_90'
                rotations: [0, 1, 2, 3]
              - type: 'scale'
                scaling: [0.75, 1.25]
                order: 1
        - group: 'color'
          random: true
          items:
              - type: 'hsb_color'
                hue: [-0.05, 0.05]
                saturation: [-0.25, 0.25]
                brightness: [-0.25, 0.25]
                ratio: 1.0
              - type: 'contrast'
                sigma: [-0.25, 0.25]
                ratio: 1.0
        - group: 'noise'
          random: false
          items:
              - type: 'additive'
                sigma: [0.0, 0.05]
              - type: 'blur'
                sigma: [0.0, 1.0]
```

- Separate settings for training and (possibly) validation
- Group with arbitrary name and the type:
    - Sequential (random = false)
    - Random (random = true)
- Augmenter list
- Augmenter type and its parameters

4. Scritps

Scripts 2/1



```
argument_parser.add_argument('-i',  '--input',               required=True,  type=str,                         help='image file or directory path to process')
argument_parser.add_argument('-l',  '--input_spacing',       required=True,  type=float,                       help='input image pixel spacing (micrometer)')
argument_parser.add_argument('-m',  '--mask',                required=False, type=str,   default=None,         help='mask file or directory path to use')
argument_parser.add_argument('-o',  '--output',              required=True,  type=str,                         help='output file or directory path')
argument_parser.add_argument('-g',  '--output_spacing',      required=False, type=float, default=None,         help='target output pixel spacing (micrometer)')
argument_parser.add_argument('-n',  '--model',               required=True,  type=str,                         help='network model')
argument_parser.add_argument('-v',  '--interval',            required=False, type=str,   default=None,         help='quantization interval file path')
argument_parser.add_argument('-p',  '--patch_size',          required=False, type=int,   default=1024,         help='processing patch size')
argument_parser.add_argument('-c',  '--output_class',        required=False, type=int,   default=-1,           help='output class')
argument_parser.add_argument('-u',  '--num_classes',         required=False, type=int,   default=-1,           help='number of output classes of the network')
argument_parser.add_argument('-a',  '--channels',            required=False, type=str,   default='(0, 1, 2)',  help='list of input channel indices')
argument_parser.add_argument('-f',  '--confidence',          required=False, type=float, default=0.0,          help='network confidence for thresholding')
argument_parser.add_argument('-b',  '--diagonal_threshold',  required=False, type=float, default=0.0,          help='region size filter (micrometer)')
argument_parser.add_argument('-t',  '--tolerance',           required=False, type=float, default=0.25,         help='pixel spacing tolerance (percentage)')
argument_parser.add_argument('-z',  '--order',               required=False, type=int,   default=0,            help='interpolation order')
argument_parser.add_argument('-r',  '--purposes',            required=False, type=str,   default=None,         help='list of purpose identifiers to use from da
argument_parser.add_argument('-e',  '--categories',          required=False, type=str,   default=None,         help='list of category identifiers to use from d
argument_parser.add_argument('-d',  '--input_override',      required=False, type=str,   default=None,         help='data config source overrides')
argument_parser.add_argument('-cp', '--copy_directory',      required=False, type=str,   default=None,         help='data copy target directory path')
argument_parser.add_argument('-wp', '--work_directory',      required=False, type=str,   default=None,         help='work directory path')
argument_parser.add_argument('-nr', '--normalizer',          required=False, type=str,   default='rgb_to_0-1', help='normalizer to use for preprocessing')
argument_parser.add_argument('-tr', '--source_range',        required=False, type=str,   default='[]',         help='source-range for normalizer')
argument_parser.add_argument('-sr', '--target_range',        required=False, type=str,   default='[]',         help='target-range for normalizer')
argument_parser.add_argument('-s',  '--soft',                action='store_true',                              help='get soft classification from the network')
argument_parser.add_argument('-q',  '--quantize',            action='store_true',                              help='quantize result to [0, 255]')
argument_parser.add_argument('-x',  '--no_mask',             action='store_true',                              help='do not use mask entries from the data conf
argument_parser.add_argument('-cf', '--channels_first',      action='store_true',                              help='channels first')
argument_parser.add_argument('-ki', '--keep_intermediates',  action='store_true',                              help='keep intermediate files')
argument_parser.add_argument('-kc', '--keep_copies',         action='store_true',                              help='keep copied image files')
argument_parser.add_argument('-w',  '--overwrite',           action='store_true',                              help='overwrite existing results')
```

Scripts 2/2

- **`imageinfo.py:`** Print image shape and spacing information.
- **`normalizemasks.py:`** Map mask values to different values.
- **`preprocessmasks.py:`** Calculate STAT files from mask images.
- **`saveimagesatlevel.py:`** Save TIFF image at a given level (still multiresolution).
- **`savemrimageasimage.py:`** Save TIFF image at a given level as PNG.
- **`setspacing.py:`** Set the pixel spacing on level 0 for a TIFF image (recompression).
- **`summarizelogs.py:`** Get the best values from the logs of ExperimentCommander.
- **`thresholdimage.py:`** Low-threshold a TIFF image at a given value.
- **`trainnetwork.py:`** Callable wrapper around ExperimentCommander.
- **`updatenetwork.py:`** Update an old network file format to the newest one.
- **`zoomimage.py:`** Zoom image.

Developing DigitalPathology

Developing DigitalPathology

- GitHub: https://github.com/DIAGNijmegen/DigitalPathology
- Latest stable branch: **master**
- Latest development branch: **develop**
- To add something to DigitalPathology:
  1. Create a branch from develop, called
     - `feature/feature_name`
     - `experiment/feature_name`
  2. Commit changes
  3. Create pull request
- Follow **PEP 8** recommendations **and** DigitalPathology/documents/**style.md**

Configuration advices

Configuration advices

1. Look for examples in DigitalPathology/examples
2. The example annotation configuration is **not a good one!**
3. The resource configuration is a **good one**
   - ***system: process count***
   - ***system: pool size*** (set it for your project!)
   - ***data: resources***
4. Configurating the resource use:
   - There is only **either** the training **or** the validation BatchGenerator active at once
   - Use ***process count: 2.0*** setting to use all the available CPUs
   - Use ***--cpu=X*** command line option for the trainnetwork.py script on the cluster
   - Each worker process has its own python interpreter (~200Mb of memory)
   - Test how much memory a single image/mask pair consumes when loaded to a PatchSampler
   - Log in to the executing machine and use ***docker stats*** command to monitor the memory usage

Errors

Errors

1. Look for the error reporting in the logs
2. Most of the errors has their unique exception
3. The stack trace is reported with the exception
4. If a process disappears during training it is most likely due to **insufficient memory**
5. Always find the first error in the logs
6. Try remote debugging if you cannot reproduce the error locally

Errors

```
[training] Exception raised: 'FailedSourceSelectionError("Not all [1, 2, 3] labels can be sampled from the current source selection:
                                   ['c:\TissueBackground\masks\tongue_ae1ae3_01_mask_sp2.0.tif',
                                    'c:\TissueBackground\masks\kidney_pas_00_mask_sp2.0.tif',
                                    'c:\TissueBackground\masks\tongue_ki67_02_mask_sp2.0.tif',
                                    'c:\TissueBackground\masks\breast_lymph_node_ck8-18_01_mask_sp2.0.tif',
                                    'c:\TissueBackground\masks\breast_lymph_node_hne_00_mask_sp2.0.tif',
                                    'c:\TissueBackground\masks\tongue_hne_00_mask_sp2.0.tif',
                                    'c:\TissueBackground\masks\rectum_hne_01_mask_sp2.0.tif',
                                    'c:\TissueBackground\masks\lung_hne_01_mask_sp2.0.tif'].",)'.
           Trace: 'batchsamplerdaemon.py:130:batchsampler_daemon_loop/batchsampler.py:938:step/batchsampler.py:757:__samplesources'.
           Shutdown is imminent

[training] Unknown response from batch sampler:
           {'response': 'error’,
            'exception': 'FailedSourceSelectionError("Not all [1, 2, 3] labels can be sampled from the current source selection:
                               ['c:\TissueBackground\masks\tongue_ae1ae3_01_mask_sp2.0.tif’,
                                'c:\TissueBackground\masks\kidney_pas_00_mask_sp2.0.tif’,
                                'c:\TissueBackground\masks\tongue_ki67_02_mask_sp2.0.tif’,
                                'c:\TissueBackground\masks\breast_lymph_node_ck8-18_01_mask_sp2.0.tif’,
                                'c:\TissueBackground\masks\breast_lymph_node_hne_00_mask_sp2.0.tif’,
                                'c:\TissueBackground\masks\tongue_hne_00_mask_sp2.0.tif’,
                                'c:\TissueBackground\masks\rectum_hne_01_mask_sp2.0.tif’,
                                'c:\TissueBackground\masks\lung_hne_01_mask_sp2.0.tif'].",)’,
            'trace':
'batchsamplerdaemon.py:130:batchsampler_daemon_loop/batchsampler.py:938:step/batchsampler.py:757:__samplesources’,
            'command': {'command': 'step'},
            'tid': 15396}
```

Questions?