```yaml
# affix version
version:
  {
    minimum: 0.0.14,
    maximum: 0.0.14 # this should NOT be made a variable, but should be tested
after every tag is created
  }
# Choose the model parameters here
model:
  {
    dimension: 4, # the dimension of the model and dataset: defines
dimensionality of computations
    base_filters: 30, # 30 is for a GPU with 11GB VRAM - can be
decreased/increased appropriately
    architecture: unet, # options: unet, resunet, fcn, uinc
    final_layer: sigmoid, # can be either sigmoid, softmax or none (none ==
regression)
    class_list: [0,1,2,3,4], # Set the list of labels the model should train
on and predict
    amp: True, # Set if you want to use Automatic Mixed Precision for your
operations or not - options: True, False
    # n_channels: 3, # set the input channels - useful when reading RGB or
images that have vectored pixel types
  }
# metrics to evaluate the validation performance
metrics:
  - 'dice'
# Set the Modality : rad for radiology, path for histopathology
modality: rad
weighted_loss: True
# Patch size during training - 2D patch for breast images since third
dimension is not patched
patch_size: [128,128]
# uniform: UniformSampler or label: LabelSampler
patch_sampler: uniform
# Number of epochs
num_epochs: 500
# Set the patience - measured in number of epochs after which, if the
performance metric does not improve, exit the training loop - defaults to the
number of epochs
patience: 50
# Set the batch size
batch_size: 1 # change according to available GPU memory - this is for 11GB
# Set the initial learning rate
learning_rate: 0.1
# Learning rate scheduler - options: triangle, exp, reduce-on-lr, step, more
to come soon - default hyperparameters can be changed thru code
scheduler: triangle_modified
```

```
# Set which loss function you want to use - options : 'dc' - for dice only,
'dcce' - for sum of dice and CE and you can guess the next (only lower-case
please)
# options: dc (dice only), dc_log (-log of dice), ce (), dcce (sum of dice and
ce), mse () ...
# mse is the MSE defined by torch and can define a variable 'reduction'; see
https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html#torch.nn.MSELo
ss
# use mse_torch for regression/classification problems and dice for
segmentation
loss_function: dcce
#loss_function:
#  {
#    'mse':{
#      'reduction': 'mean' # see
https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html#torch.nn.MSELo
ss for all options
#    }
#  }
# Which optimizer do you want to use - adam/sgd
optimizer: adam
# this parameter controls the nested training process
# performs randomized k-fold cross-validation
# split is performed using sklearn's KFold method
# for single fold run, use '-' before the fold number
nested_training:
  {
    testing: 5, # this controls the testing data splits for final model
evaluation; use '1' if this is to be disabled
    validation: 5 # this controls the validation data splits for model
training
  }
## pre-processing
# this constructs an order of transformations, which is applied to all images
in the data loader
# order: resize --> threshold/clip --> resample --> normalize
# 'threshold': performs intensity thresholding; i.e., if x[i] < min: x[i] = 0;
and if x[i] > max: x[i] = 0
# 'clip': performs intensity clipping; i.e., if x[i] < min: x[i] = min; and if
x[i] > max: x[i] = max
# 'threshold'/'clip': if either min/max is not defined, it is taken as the
minimum/maximum of the image, respectively
# 'normalize': performs z-score normalization:
https://torchio.readthedocs.io/transforms/preprocessing.html?highlight=ToCanon
ical#torchio.transforms.ZNormalization
# 'resample: resolution: X,Y,Z': resample the voxel resolution:
https://torchio.readthedocs.io/transforms/preprocessing.html?highlight=ToCanon
ical#torchio.transforms.Resample
```

```
# 'resample: resolution: X': resample the voxel resolution in an isotropic
manner:
https://torchio.readthedocs.io/transforms/preprocessing.html?highlight=ToCanon
ical#torchio.transforms.Resample
# resize the image(s) and mask (this should be greater than or equal to
patch_size); resize is done ONLY when resample is not defined
# crop_external_zero_planes: crops all non-zero planes from input tensor to
reduce image search space
data_preprocessing:
  {
    'normalize_nonZero',
    'crop_external_zero_planes',
  }
# various data augmentation techniques
# options: affine, elastic, downsample, motion, ghosting, bias, blur,
gaussianNoise, swap
# keep/edit as needed
# all transforms:
https://torchio.readthedocs.io/transforms/transforms.html?highlight=transforms
# 'kspace': one of motion, ghosting or spiking is picked (randomly) for
augmentation
# 'probability' subkey adds the probability of the particular augmentation
getting added during training (this is always 1 for normalize and resampling)
data_augmentation:
  {
    'affine':{
      'probability': 0
    },
    'elastic':{
      'probability': 0
    },
    'kspace':{
      'probability': 0
    },
    'bias':{
      'probability': 0
    },
    'blur':{
      'probability': 0
    },
    'noise':{
      'probability': 0
    },
    'rotate_90':{
      'probability': 0
    },
    'rotate_180':{
      'probability': 0
    },
  }
# parallel training on HPC - here goes the command to prepend to send to a
high performance computing
# cluster for parallel computing during multi-fold training
# not used for single fold training
# this gets passed before the training_loop, so ensure enough memory is
provided along with other parameters
# that your HPC would expect
# ${outputDir} will be changed to the outputDir you pass in CLI +
'/${fold_number}'
# ensure that the correct location of the virtual environment is getting
invoked, otherwise it would pick up the system python, which might not have
all dependencies
#parallel_compute_command: 'qsub -b y -l gpu -l h_vmem=32G -cwd -o
${outputDir}/\$JOB_ID.stdout -e ${outputDir}/\$JOB_ID.stderr `pwd`/sge_wrapper
_correct_location_of_virtual_environment_/venv/bin/python'
## queue configuration -
https://torchio.readthedocs.io/data/patch_training.html?#queue
# this determines the maximum number of patches that can be stored in the
queue. Using a large number means that the queue needs to be filled less
often, but more CPU memory is needed to store the patches
q_max_length: 100
# this determines the number of patches to extract from each volume. A small
number of patches ensures a large variability in the queue, but training will
be slower
q_samples_per_volume: 10
# this determines the number subprocesses to use for data loading; '0' means
main process is used
q_num_workers: 16 # change according to available cpu threads
# used for debugging
q_verbose: False
```