# Philosophy

HistoQC consists of a pipeline of modules being sequentially applied to an image. These modules act on the image to (a) produce metrics, and (b) produce output images after running classifiers or applying thresholds.

When an image is loaded it is initially assigned a True mask of the same size called *img\_mask\_use*, indicating that every pixel in the image is artifact free and “useful” for analysis.

The HistoQC approach uses its pipeline to sequentially refine the *img\_mask\_use* mask. For example, while initially the entire image is considered useful, after the “ LightDarkModule.getIntensityThresholdPercent:tissue” module is run, the background of the tissue should now be set to false, refining the locations in the image which are suitable for computation.

As such, the order of the events in the pipeline is important. Most modules have the option to “limit\_to\_mask” which implies that the module’s operations will only take place in the regions currently identified as accepted by *img\_mask\_use*. For example, when computing image color distributions, one would like to only operate on the part of the image that has tissue and avoid the white background which will artificially inflate the white value of the distribution, thus placing “HistogramModule.getHistogram” after “LightDarkModule.getIntensityThresholdPercent:tissue” is ideal.

# Suggested workflow

Through various experiments, we have come to the following suggested workflow. Depending on your task and the expected homogeneity of your dataset, this approach may be rather extreme, so it is suggested that you modify your approach accordingly.

1. Run HistoQC overall images with only the “BasicModule.getMag” module enabled. (e.g., config\_mag.ini). Identify all images which are maximally scanned at different magnifications, e.g., 20x and 40x images. Split the cohort into sub-cohorts based on these values as (a) various modules are likely to function differently at different magnifications, (b) the *image level* which is loaded by openslide will be different, implying potential memory issues (loading too big of an image accidently) or attempting to open a level which doesn’t exist. Ideally, one wants sub-cohorts which are all of the same magnification and contain the same number of levels.

2.

2) detect overall contrast differences and break into groups (light group, dark group) expect same paramters to work for both?low

In case of errors: For example, if some images caused errors because of out of memory, you can rerun pipeline simply by deleting their output directories. they are easily found because they don't have thumbnail images. matlab code to do this:

files=dir('\*.svs');

for zz=1:length(files)

fname=files(zz).name;

if(~exist(sprintf('%s/%s\_thumb.png',fname,fname),'file'))

fprintf('%s\n',fname);

rmdir(fname,'s');

end

end

# Pipeline configuration

## Pipeline module order

The pipeline configuration is specified using a configuration file. A default config.ini is supplied in the repository. The configuration syntax is that of pythons configparser [https://docs.python.org/3/library/configparser.html]. In brief this means that the configuration file has sections, and each section has key value parameters. In the HistoQC setting, the sections are named for their associated module.

There is only a single required section, which is called “[pipeline]”. This section defines, again in sequential order, the steps which will be taken on a per image basis. An example pipeline configuration is presented here:

**[pipeline]**steps= BasicModule.getBasicStats  
 BasicModule.getMag  
 ClassificationModule.byExampleWithFeatures:pen\_markings  
 #ClassificationModule.byExampleWithFeatures:pen\_markings\_red  
 ClassificationModule.byExampleWithFeatures:coverslip\_edge  
 #LightDarkModule.getIntensityThresholdPercent:bubble  
 LightDarkModule.getIntensityThresholdPercent:tissue  
 #BubbleRegionByRegion.pixelWise  
 LightDarkModule.getIntensityThresholdPercent:darktissue  
 MorphologyModule.removeSmallObjects  
 MorphologyModule.fillSmallHoles  
 BlurDetectionModule.identifyBlurryRegions  
 BasicModule.finalProcessingSpur  
 BasicModule.finalProcessingArea  
 HistogramModule.compareToTemplates  
 HistogramModule.getHistogram  
 BrightContrastModule.getContrast  
 BrightContrastModule.getBrightness  
 DeconvolutionModule.seperateStains  
 SaveModule.saveFinalMask  
 SaveModule.saveThumbnail  
 BasicModule.finalComputations

We note here that it is possible to use the *same* module multiple times, with different settings and assign it a different name. For example getIntensityThresholdPercent applies a threshold to the image, and “getIntensityThresholdPercent:tissue” applies a high threshold to remove the background on the slide, while “getIntensityThresholdPercent:darktissue” applies a low threshold to identify regions which may contain artifacts such as folded tissue or drastic overstaining. Each instance of the module is defined as the base module name (getIntensityThresholdPercent) plus a double colon followed by the specific instance name of that module (e.g., “:darktissue”). Later on in the configuration file, we can see the associated sections are named exactly the same ([LightDarkModule.getIntensityThresholdPercent:darktissue]), and that each section contains a “name:” parameter, which is used as the output name of the image as well as the column name in the tsv results file.

## Pipeline image size

BaseImage’s image\_work\_size parameter specifies the default size of the internal representation of the image to be used in the pipeline. Most modules, unless otherwise specified, will use an image of this size to perform their operation, thus setting a suitable size is important. In most cases, it is infeasible to load an entire 40x whole slide mount, but even in doing so will not provide greater specificity in most of the metrics (e.g., color distributions). As such a default of “3” is recommend, which when using Aperio svs files typically coincides with a 40x magnified image being loaded at 1.25x (1:32 reduction).

There are 3 ways to specify the desired image size

1. When image\_work\_size < 1 and is a floating point number, it is considered a downscaling factor of the original image (e.g., new.image.dimensions = image.dimensions \* image\_work\_size)
2. if image\_work\_size < 100, it is considered to indicate the *level* of image to load using the openslide pointer. In the case of Aperio Svs, this typically coincides with {0=Base, 1 = 4:1, 2=16:1, 3=32:1}
3. if image\_work\_size > 100, this is considered to be the *exact* longest dimension desired (e.g., an image of size 1234 x 2344, if image\_work\_size is set to 500, the output will be 263 x 500)

BEWARE: these operations are not free! In cases #1 and #3, we leverage the openslide “get\_thumbnail” function to produce the requested image. This function works by taking the *next* largest image layer, loading it, and then downsizing it to the requested size. One can image that if the image\_work\_size size is not properly set, the whole uncompressed image will be loaded, likely exploding available resources.

# Adding classification type modules

Most of the modules are implemented using statistics or thresholds and are thus relatively easy to setup. The classification modules represent a departure from that simplicity and are not only the most sophisticated modules in HistoQC, but also the most powerful. The classification approach consists of first loading exemplar images from which to create a model. Each exemplar should consists of 2 images of the same size. Each set is specified under the “examples” parameter, one per line, and each separated by a double colon like so:

examples: ./pen/1k\_version/pen\_green.png:./pen/1k\_version/pen\_green\_mask.png  
 #./pen/1k\_version/pen\_red.png:./pen/1k\_version/pen\_red\_mask.png

Which indicates that relative or absolute location of 2 exemplars (pen\_green and pen\_red), and their associated mask (pen\_green\_mask.png and pen\_red\_mask.png). The mask is a binary image (i.e., only containing the values 0,1) which identifies which pixels should be used as the positive class in the image (e.g., 1), and the pixels which should be used as the negative class (e.g., 0). It usually makes sense for these images to be of the same magnification specified by “image\_work\_size”, as this will improve the performance of the classifier.

In the second step, after the images are loaded, a classifier is trained. To improve the robustness of the classifier, we allow for the computation of a number of different pixel-features to augment the original RGB space. These features are those implemented in skimage.filters [http://scikit-image.org/docs/dev/api/skimage.filters.html] and include:

features: frangi  
 laplace  
 rgb  
 #lbp  
 #gabor  
 #median  
 #gaussian

Each of their parameters can be set by using the feature name as the prefix to the parameter, for example: “frangi\_black\_ridges: True”, sets the “black\_ridges” parameter of the frangi filter to true. A single model is trained and shared by all individual the processes which request access to it.

After the model is trained, it is retained in memory, and is applied at the appropriate time to the images identified by HistoQC. Internally, the output from this is a probability likihood that a particular pixel belongs to the trained positive class, but as a real value output is not suitable here, we accept a parameter “tresh” which will apply a threshold to the probability map to provide the final binary value mask which is used in downstream analysis.

# Extending HistoQC

Internal Representation

output images need to have same format, will automatically appear in dropdown "[filename].svs\_[format].png”. Is this always the case?

How to add to print list?

As such, HistoQC was designed specifically to be easily extendible by even novice programmers. The best way to get started it to identify a module which is most similar to the desired added functionality, copy the module and then start modifying.

Passed to each module are 2 parameters, “s” and “params”.

Params contains the parameters for that specific module as specified in the configuration file. Any values added to here will be lost.

“s” is a hold-all structure for an image and is of type BaseImage. It contains all of the metrics, metadata, and masks. Most importantly it contains an open openslide pointer for usage in loading the slide.

s.addToPrintList

require prerequisites

discuss idea of requiring certain things to be completed (e.g., histogram)