

PyHIST: A Histological Image Segmentation Tool

Authors: Manuel Muñoz-Aguirre^{†1,2}, Vasilis F. Ntasis^{†1}, Roderic Guigó^{*1,3}

Affiliations:

¹Centre for Genomic Regulation (CRG), The Barcelona Institute for Science and Technology, Barcelona, Catalonia, Spain

²Department of Statistics and Operations Research, Universitat Politècnica de Catalunya (UPC), Barcelona, Catalonia, Spain

³Department of Experimental and Health Sciences (DCEXS), Universitat Pompeu Fabra (UPF), Barcelona, Catalonia, Spain

[†]Contributed equally to this work.

^{*}Corresponding author. Correspondence to: manuel.munoz@crg.eu

1 **Abstract**

2 The development of increasingly sophisticated methods to acquire high resolution images has led
3 to the generation of large collections of biomedical imaging data, including images of tissues and
4 organs. Many of the current machine learning methods that aim to extract biological knowledge
5 from histopathological images require several data preprocessing stages, creating an overhead
6 before the proper analysis. Here we present PyHIST ([https://github.com/manuel-munoz-](https://github.com/manuel-munoz-aguirre/PyHIST)
7 [aguirre/PyHIST](https://github.com/manuel-munoz-aguirre/PyHIST)), an easy-to-use, open source whole slide histological image tissue segmentation
8 and preprocessing tool aimed at data preparation for machine learning applications.

1 **Main text**

2 In histopathology, Whole Slide Images (WSI) are high resolution images of tissue sections
3 obtained by scanning conventional glass slides. Histopathological images are routinely used in
4 the diagnosis of many diseases, notably cancer. The increasing automation of WSI acquisition
5 has led to the development of computational methods to process the images and help clinicians
6 and pathologists in diagnosis and disease classification. As an increasing number of larger WSI
7 datasets became available, methods have been developed for a wide array of tasks, such as the
8 classification of breast cancer metastases, Gleason scoring for prostate cancer, tumor
9 segmentation, nuclei detection and segmentation, bladder cancer diagnosis, mutated gene
10 prediction based on pathology images, among others ¹⁻⁶. Besides of being important diagnostic
11 tools, histopathological images capture endophenotypes (of organs and tissues) that, when
12 correlated with molecular and cellular data on the one hand, and higher order phenotypic traits
13 on the other, can provide crucial information on the biological pathways that mediate between the
14 sequence of the genome and the biological traits of the organisms (including diseases).

15
16 Because of the complexity of the information typically contained in WSIs, Machine Learning (ML)
17 methods that can infer, without prior assumptions, the relevant features that they encode are
18 becoming the preferred analytical tools. These features may be clinically relevant but challenging
19 to spot even for expert pathologists, and thus, ML methods can prove valuable in healthcare
20 decision-making ⁷.

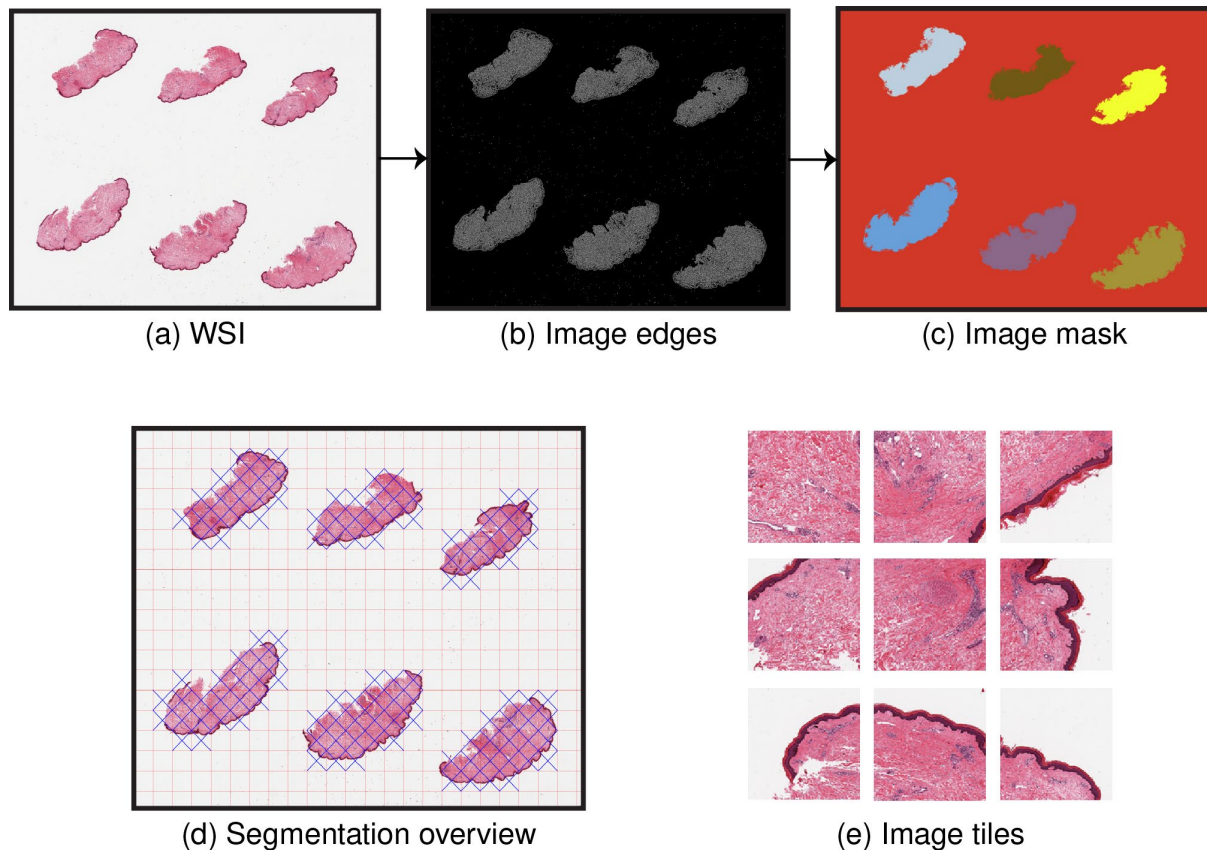
21
22 In most ML tasks, data preprocessing remains a fundamental step. Indeed, in the domain of
23 histological images, there are a number of issues when preprocessing the data before an
24 analysis: due to the large dimensions of WSIs, many deep learning applications have to break
25 them down into smaller-sized square pieces called tiles ⁸. Furthermore, a significant fraction of
26 the area in a WSI is often background, which does not contain useful information and can cause
27 computing overhead in downstream analyses. To circumvent this, some applications apply a
28 series of image transformations to identify the foreground from the background (see, for example,
29 ⁹), and perform relevant operations only over regions with tissue content. However, this process
30 is not standardized, and customized scripts have to be frequently developed to deal with data
31 preparation stages. This is cumbersome and may introduce dataset specific-biases, which can
32 prevent integration across multiple datasets.

1 To systematize the WSI preprocessing procedure for ML applications, we developed PyHIST, a
2 pipeline to segment the regions of a histological image into tiles with relevant tissue content
3 (foreground) with little human intervention.

4
5 PyHIST is a Python tool based on OpenSlide ¹⁰, a library to read high-resolution histological
6 images in a memory-efficient way. PyHIST's input is a WSI encoded in SVS format (Fig. 1a), and
7 the main output is a series of image tiles retrieved from regions with tissue content (Fig. 1e).

8
9 The PyHIST pipeline involves three main steps: first, tissue edges inside the WSI are identified
10 using a Canny edge detector (Fig. 1b), generating an alternate version of the image with
11 diminished noise and an enhanced distinction between the background and the tissue foreground.
12 Second, these edges are processed by a graph-based segmentation algorithm ¹¹, which is used
13 here to identify tissue content. In short, this step evaluates the boundaries between different
14 regions of an image as defined by the edges; different parts of the image are represented as
15 connected components of a graph and the "within" and "in-between" variations of neighbouring
16 components are assessed in order to decide if the examined image regions should be merged or
17 not into a single component. From this, a mask is obtained in which the background and the
18 different tissue slices are separated and marked as distinct objects using different colors (Fig. 1c).
19 Finally, the selected regions are divided into tiles at a user-specified size. Optionally, it is possible
20 to only select tiles that contain a proportion of tissue above a given threshold with respect to the
21 total area of the tile.

22
23 Of note, tile generation can be performed at the native resolution of the WSI, but downsampling
24 factors can also be specified to generate tiles at lower resolutions. Additionally, edge detection
25 and mask generation can also be performed on downsampled versions of WSI - reducing
26 segmentation runtimes (Fig. S1, Methods). A segmentation overview image is generated at the
27 end of the segmentation procedure for the user to visually inspect the selected tiles (Fig. 1d). With
28 the set of parameters available in PyHIST (Supplementary text), the user can specify regions to
29 ignore when performing the masking and segmentation (Fig. S2), and have a fine grained control
30 over specific use-cases. PyHIST also has a random tile sampling mode for those applications that
31 do not necessarily need to distinguish the background from the foreground. In this mode, tiles at
32 a user-specified size and resolution will be extracted from random starting positions in the WSI.



1 **Figure 1: PyHIST pipeline.** (a) The input to the pipeline is a Whole Slide Image (WSI). Within PyHIST, the
2 user can decide to scale down the image to perform the segmentation and tile extraction at lower
3 resolutions. The WSI shown is of a skin tissue sample (GTEx-1117F-0126) from the Genotype-Tissue
4 Expression (GTEx) project ¹² (b) An alternate version of the input image is generated, where the tissue
5 edges are highlighted using a Canny edge detector. A graph segmentation algorithm is employed over this
6 image, in order to generate the mask shown in (c). PyHIST extracts tiles of specific dimensions from the
7 masked regions, and provides an overview image to inspect the output of the segmentation and masking
8 procedure, as shown in (d), where the red lines indicate the grid generated by tiling the image at user-
9 specified tile dimensions, while the blue crosses indicate the selected tiles meeting a certain user-specified
10 threshold of tissue content with respect to the total area of the tile. In (e), examples of selected tiles are
11 shown.

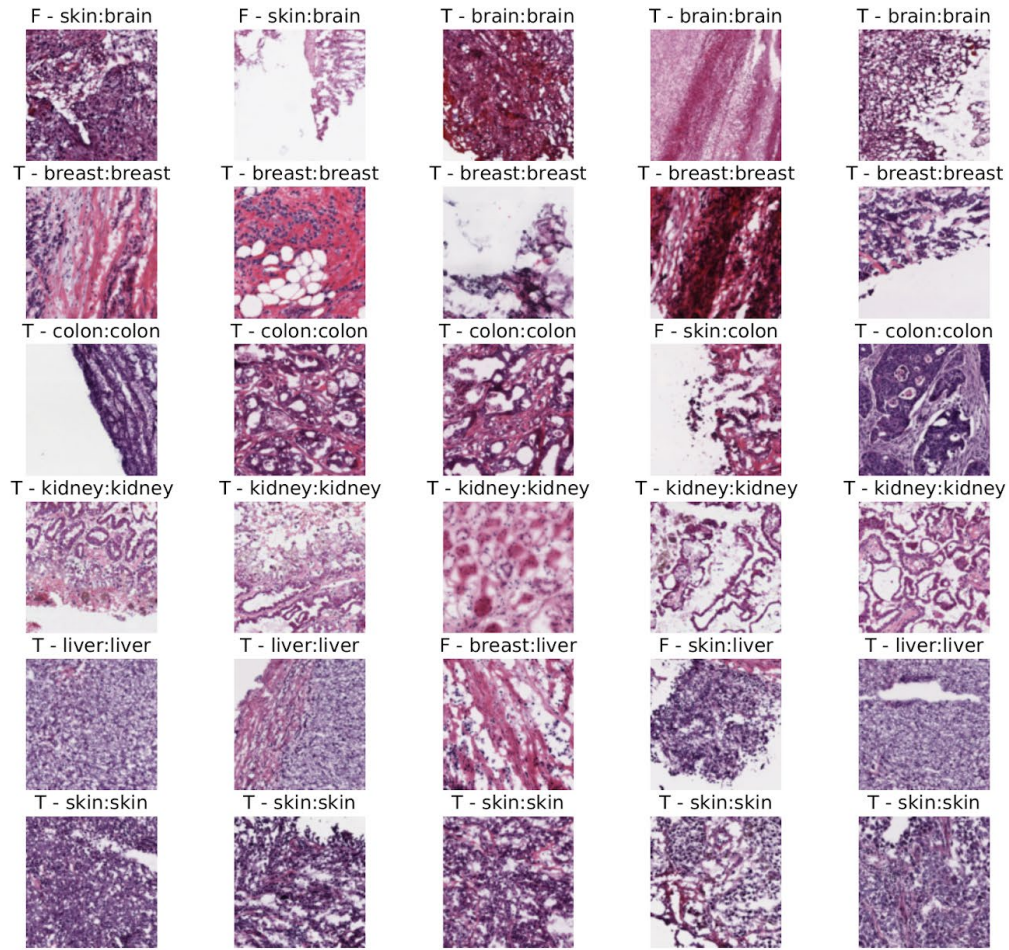
12 To demonstrate how PyHIST can be used to preprocess WSIs for usage in a ML application, we
13 generated a use case example with the goal of building a classifier at the tile-level that allows us
14 to determine the tissue of origin based on the histological patterns encoded in these tiles. To this
15 end, we first retrieved a total of 30 publicly available WSIs, 5 from each of the following human
16 tissues hosted in The Cancer Genome Atlas (TCGA) ¹³: Brain, Breast, Colon, Kidney, Liver, and
17 Skin. Second, these WSIs were preprocessed with PyHIST, generating a total of 4720 tiles with

1 dimensions 512x512. These tiles were then partitioned into training and test sets, and we then fit
2 a deep learning model over these tiles, achieving a classification accuracy of 94% (Fig. 2a, Table
3 S1). We also inspected the feature vectors generated by the deep learning model: for each tile,
4 we retrieved the features corresponding to the last layer of the network, and performed
5 dimensionality reduction (tSNE) over the stacked matrix of these vectors. From here, we infer that
6 the learned features clearly recapitulate tissue morphology since tile clusters corresponding to
7 each tissue are formed (Figs. 2b, S3).

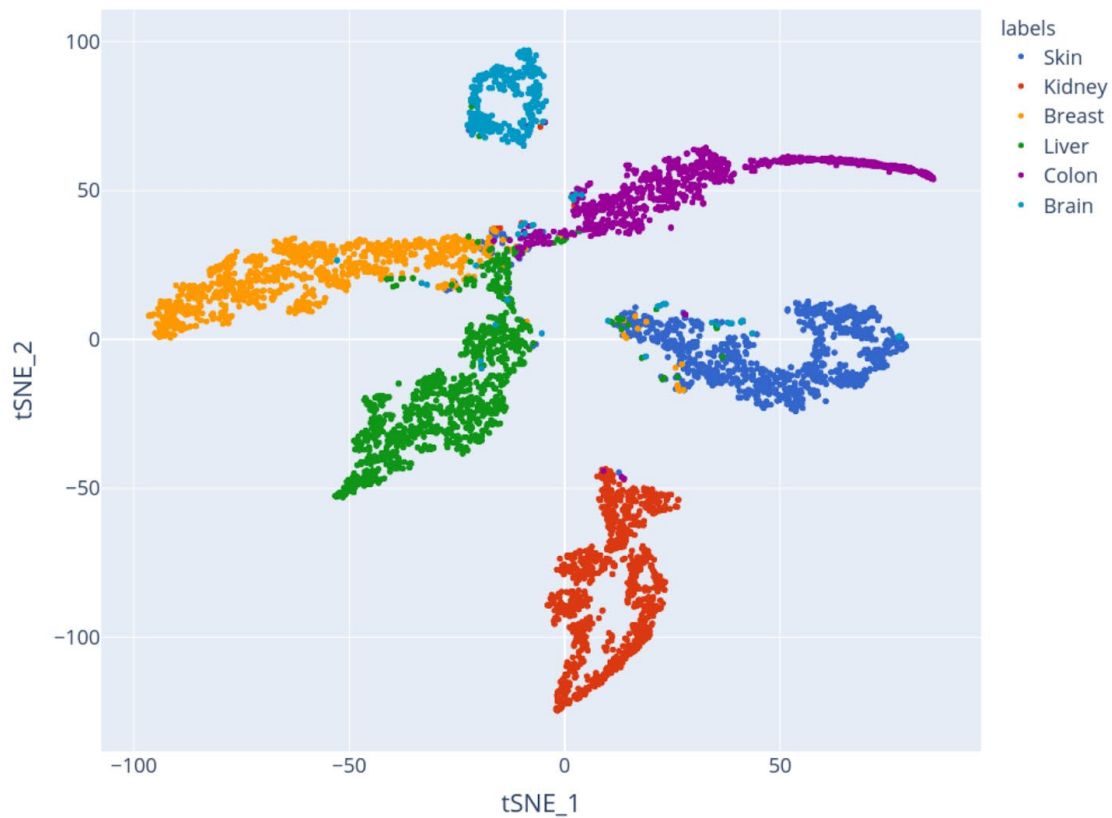
8
9 With this use case, we have shown how to quickly prepare WSI data using PyHIST without
10 manually preprocessing the images, reducing the overhead to start performing downstream
11 analyses. The example use case described here is documented and fully available at
12 <https://pyhist.readthedocs.io/en/latest/testcase/>, and divided into three Jupyter notebooks: 1)
13 Data preprocessing with PyHIST, 2) Constructing a deep learning tissue classifier, and 3) Feature
14 exploration.

15
16 PyHIST is a generic tool to segment histological images automatically: it allows for easy and rapid
17 WSI cleaning and preprocessing with minimal effort to generate image tiles geared towards usage
18 in ML analyses. The tool is highly customizable, enabling the user to tune the segmentation
19 process in order to suit the needs of any particular application that relies on histological image
20 tiles. PyHIST and all of its dependencies have been packaged in a Docker image, ensuring
21 portability across different systems. PyHIST can also be used locally on a regular computing
22 environment with minimal requirements. Finally, PyHIST is open source software: all the code
23 and reproducible notebooks for the example use case are available in GitHub and will continue
24 to be improved on the basis of user feedback.

A



B



1 **Figure 2: TCGA use case. (a)** Random examples of tile predictions from TCGA tissue tiles. 30 random
2 tiles are shown, 5 tiles per tissue in each row. Above each tile, "T" indicates that the prediction was correct
3 and "F" incorrect, followed by the predicted tissue label, and the ground truth tissue label for that tile. **(b)**
4 Dimensionality reduction of TCGA tiles. tSNE performed with the feature vectors (activations) of each tile
5 that were derived from the deep learning classifier model. Each dot corresponds to an image tile.

6 **Methods**

7 **Overview of PyHIST:** PyHIST is a high-resolution histological image segmentation pipeline. It
8 takes as input a Whole Slide Image (WSI) encoded in SVS format (Fig. 1a) and produces a series
9 of image tiles (Fig. 1e), with dimensions specified by the user. These tiles are usually extracted
10 from the regions of the WSI that have tissue content, after discarding the background, although it
11 is also possible to extract all the tiles from the WSI. Tile extraction is performed as if overlaying a
12 grid with tiles of a fixed size over the original image (Fig. 1d), and then keeping the relevant tiles.
13 It is also possible to sample tiles from random starting positions in the WSI. In both cases, tile
14 extraction can be performed either at the original resolution of the image, or at a downsampled
15 version.

16
17 **WSI segmentation:** In order to identify the sections of the WSI that contain tissue, PyHIST first
18 uses a Canny edge detector ¹⁴ to generate a version of the image that highlights edges within
19 tissue fragments. This image is then used in combination with a graph segmentation algorithm ¹¹
20 to select connected components in the image, obtaining a clean mask that corresponds to tissue
21 segments (Fig. 1c). The mask will be then divided into tiles. Each tile will be inspected to
22 determine if it's composed by at least a certain percentage of tissue content with respect to the
23 total area of the tile (see argument *--content-threshold*, Supplementary text). If the tile is kept,
24 then the corresponding area of the original WSI will be retrieved. Tiles can be saved in JPG or
25 PNG formats.

26
27 Although it is possible to perform the segmentation at the highest resolution encoded in the WSI,
28 by default PyHIST will output tiles from a downsampled version (i.e. a lower resolution version of
29 the image, scaled by a factor). The user can specify a downsampling factor (powers of 2) to
30 perform tile extraction at a lower resolution (see argument *--output-downsample*, Supplementary
31 text). Of note, incomplete tiles can be generated towards the right and lower borders of the image
32 due to the selected tile size not being a multiple of the WSI size. In this case, the size of these
33 tiles will be as large as possible, without exceeding the specified tile dimension. When the process

1 is finished, a tab-delimited text file will be also produced indicating tile coordinates and size of
2 each tile, as well as an indicator column stating if the tile passed the content threshold to be
3 considered foreground.

4
5 Several options are provided to tune PyHIST for specific segmentation use-cases. Usually, tissue
6 segments are placed towards the center of the slide during the imaging process. However, in
7 some cases, tissue can be located in the borders of the slide, and depending on the application,
8 the user would like to keep or remove these regions. PyHIST provides parameters to deal with
9 these types of cases (Supplementary text).

10
11 It is also possible to extract a given number of tiles randomly from the WSI at any given resolution.
12 In this case, since the tiles are sampled from any position, there is no need to perform
13 segmentation.

14
15 **Intermediate scaling:** To speed up the segmentation process, the input image (Fig. S1a) can be
16 downscaled at different resolutions, at different steps of the pipeline, not only initially. If the mask
17 is downscaled (Fig. S1b), the edge generation and segmentation process will be performed at a
18 lower resolution, reducing the time needed to complete the process. The output resolution used
19 to generate the tiles (Fig. S1c) is independent of all other choices: for example, a user can decide
20 to generate the mask at 16x, but output tiles at native (1x) or any other resolution. If the mask is
21 closer to the native resolution, a more precise segmentation is obtained, however, for most
22 applications, such level of detail is not necessary. The same is applicable for the segmentation
23 overview image (Fig. S1d), which can be generated at any resolution independently of the other
24 choices. All these arguments are available in the "Downsampling" section of the Supplementary
25 text.

26
27 **Test mode:** A test mode (argument `--test-mode`) is available in PyHIST to inspect how a mask
28 will look with a given combination of segmentation parameters, as well as to verify the tiling grid
29 that will be generated at the selected tile dimensions (Fig. S2). This is to aid the user in inspecting
30 the output before producing the individual tile files. When the test mode is invoked, no tile selection
31 will be performed, as it only serves to assess how the tiles will be generated.

32
33 **Execution times:** We benchmarked PyHIST's execution time both for random tile sampling and
34 segmentation. Using the WSI in Fig. S1 with native dimensions 47807x38653, we performed

1 random sampling of a varying number of tiles at different downsampling factors (Fig. S4a),
2 observing a linear trend in runtime with respect to the number of tiles. Efficiency will be determined
3 by the resolution levels natively encoded in the WSI generation process: for this WSI, 1x, 4x and
4 16x are available. If the requested downsampling factors are available in the image layers of the
5 WSI file, segmentation will be faster. On the other hand, if the requested downsampling factor is
6 not encoded in the WSI (like 8x in this example), sampling will run for a longer time since resizing
7 operations need to be performed. We also examined the runtime of performing random sampling
8 of 1000 tiles at a fixed tile size, and at different downsampling levels (Fig. S4b). As expected,
9 runtime increases with tile size. Similarly to the previous benchmark, here we also observe the
10 effect that the native encodings in the WSI have on the runtime speed: since 8x is not encoded
11 in the WSI, resizing operations need to be performed, leading to longer runtimes when compared
12 to 1x and 4x.

13
14 To benchmark the segmentation process, we use 50 different WSIs of stomach tissue (with
15 different dimensions) from the Genotype-Tissue Expression (GTEx) project ¹² and perform
16 segmentation at different downsampling factors at a fixed tile size of 256x256. Runtime variability
17 within a single downsampling factor will be determined by slide size; and segmentation runtime
18 decreases almost linearly with respect to resolution (Fig. S4c), while the runtime will increase
19 linearly with the dimension of the slide (Fig. S4d).

20
21 **TCGA tissue classification use case:** We downsample the original WSIs by a factor of 4x, and
22 require that a tile is composed of at least 40% of tissue content in order to consider it for further
23 analysis. From the 30 WSIs, a total of 4720 tiles with dimensions 512x512 were produced. We
24 performed data augmentation by applying a set of transformations to the image tiles (rotations,
25 resizing, crops, and flips). We then use a ResNet152 ¹⁵ deep learning architecture pretrained on
26 ImageNet ¹⁶ to classify the tissue of origin for each tile (Fig. 2a). Transfer learning is performed
27 by changing the last fully connected layer of the model, freezing the rest of the network, and
28 training only the last layer.

Main references

1. Ehteshami Bejnordi, B. *et al.* Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *JAMA* **318**, 2199–2210 (2017).
2. Nagpal, K. *et al.* Development and validation of a deep learning algorithm for improving Gleason scoring of prostate cancer. *npj Digital Med.* **2**, 48 (2019).
3. Qaiser, T. *et al.* Fast and accurate tumor segmentation of histology images using persistent homology and deep convolutional features. *Med. Image Anal.* **55**, 1–14 (2019).
4. Hou, L. *et al.* Sparse autoencoder for unsupervised nucleus detection and representation in histopathology images. *Pattern Recognit.* **86**, 188–200 (2019).
5. Zhang, Z. *et al.* Pathologist-level interpretable whole-slide cancer diagnosis with deep learning. *Nat. Mach. Intell.* **1**, 236–245 (2019).
6. Coudray, N. *et al.* Classification and mutation prediction from non-small cell lung cancer histopathology images using deep learning. *Nat. Med.* **24**, 1559–1567 (2018).
7. Serag, A. *et al.* Translational AI and deep learning in diagnostic pathology. *Front Med (Lausanne)* **6**, 185 (2019).
8. Srinidhi, C. L., Ciga, O. & Martel, A. L. Deep neural network models for computational histopathology: A survey. *arXiv* (2019).
9. Gertych, A. *et al.* Convolutional neural networks can accurately distinguish four histologic growth patterns of lung adenocarcinoma in digital slides. *Sci. Rep.* **9**, 1483 (2019).
10. Goode, A., Gilbert, B., Harkes, J., Jukic, D. & Satyanarayanan, M. OpenSlide: A vendor-neutral software foundation for digital pathology. *J. Pathol. Inform.* **4**, 27 (2013).
11. Felzenszwalb, P. F. & Huttenlocher, D. P. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vis.* **59**, 167–181 (2004).
12. GTEx Consortium. Genetic effects on gene expression across human tissues. *Nature* **550**,

204–213 (2017).

13. Cancer Genome Atlas Research Network *et al.* The Cancer Genome Atlas Pan-Cancer analysis project. *Nat. Genet.* **45**, 1113–1120 (2013).

Methods references

14. Canny, J. in *Readings in computer vision* 184–203 (Elsevier, 1987). doi:10.1016/B978-0-08-051581-6.50024-6
15. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 770–778 (IEEE, 2016). doi:10.1109/CVPR.2016.90
16. Deng, J. *et al.* ImageNet: A large-scale hierarchical image database. in *2009 IEEE Conference on Computer Vision and Pattern Recognition* 248–255 (IEEE, 2009). doi:10.1109/CVPR.2009.5206848

Acknowledgments

We acknowledge Kaiser Co and Valentin Wucher for testing PyHIST, and the colleagues at the lab for useful feedback. This work was supported by FPU15/03635, Ministerio de Educación, Cultura y Deporte to M.M.-A. All authors acknowledge the support of the Spanish Ministry of Science, Innovation and Universities to the EMBL partnership, the Centro de Excelencia Severo Ochoa, and the CERCA Programme / Generalitat de Catalunya.

Author contributions

M.M.-A. and R.G. conceived the idea of PyHIST. V.F.N. implemented the core features of PyHIST. M.M.-A. implemented usability features, performance upgrades and packaged the Docker image for PyHIST. V.F.N. and M.M.-A. conceived the TCGA example use case and implemented reproducible notebooks. R.G. provided conceptual advice. All authors participated in writing the manuscript.

Competing interests

The authors declare no competing interests.

Code availability

PyHIST is available at <https://github.com/manuel-munoz-aguirre/PyHIST> and released under a GPL license. Updated documentation and a tutorial can be found at <https://pyhist.readthedocs.io/>

Data availability

The TCGA WSIs in the Use Case were downloaded from the Genomic Data Commons (GDC) repository (<https://gdc.cancer.gov/>) using the GDC Data-transfer tool (<https://gdc.cancer.gov/access-data/gdc-data-transfer-tool>).