ValidPath

Release 2.0.0

Seyed Kahaki

Dec 26, 2023

CONTENTS

CHAPTER ONE

ABOUT VALIDPATH

The Whole Slide Image Processing and Machine Learning Performance Assessment Tool is a software program written in Python for analyzing whole slide images (WSIs), assisting pathologists in the assessment of machine learning (ML) results, and assessment of ML performance. The tool currently contains three modules that accept WSIs to generate image patches for AI/ML models, accept image patches (e.g., ML detected ROIs) to generate an Aperio ImageScope annotation file for validation of ML model results by pathologists, and accept outputs of ML models to generate performance results and their confidence intervals.

The Whole Slide Image Processing and Performance Assessment Tool code has been used in the following publications:

- Kahaki, Seyed, et al. "Weakly supervised deep learning for predicting the response to hormonal treatment of women with atypical endometrial hyperplasia: a feasibility study." Medical Imaging 2023: Digital and Computational Pathology. Vol. 12471. SPIE, 2023.
- Kahaki, Seyed, et al. "Supervised deep learning model for ROI detection of atypical endometrial hyperplasia and endometrial cancer on histopathology whole slide images for predicting hormonal treatment response." Medical Imaging 2024: Digital and Computational Pathology.
- Kahaki, Seyed, et al. "End-to-End Deep Learning Method for Predicting Hormonal Treatment Response in Women with Atypical Endometrial Hyperplasia or Endometrial Cancer." Journal of Medical Imaging, Journal of Medical Imaging, Under Review
- Mariia Sidulova, et al. "Contextual unsupervised deep clustering of digital pathology dataset", Submitted to ISBI 2024

CHAPTER

TWO

MODULES

There are several modules in this package including:

1. WSI handler: includes functions and classes for general WSI analysis such as read whole slide images, tissue segmentation, and normalization.
2. Annotation Extraction: this module includes several functions for processing annotations such as annotation extraction.
3. Patch Extraction: which assist pathologist and developers in extracting image patches from whole slide images region of interest.
4. Aperio ImageScope Annotation File Generator: to enable pathologist validation of the AI/ML results.
5. Performance Assessment: to assess the performance of ML models in classification tasks.

To see a demo of the functions in this toolbox, please refer to the Jupyter Notebooks files in the root folder of this package.

- 01_read_wsi.ipynb
- 02_annotation_extraction.ipynb
- 03_patch_extraction.ipynb
- 4_annotation_generator.ipynb
- 05_performance_assessment.ipynb

2.1 Installation

This installation guide provides a detailed explanation and step by step guide to install packages required for SlidePro toolbox.

2.1.1 Required Packages

In order to use ValidPath, you need to install some python packages. It is recommended to install the same version specified in this section (and in the requirement.txt).

ValidPath was tested on the following environments :

- Linux System (Tested on Ubuntu 18.04.3 LTS) and Python 3.8.8
- Windows 10 and Python 3.11.5
- To install a python package with specific version of a package using pip, you can use the syntax "pip install package==version" in the command line.

For example, in ValidPath we are using lxml which is one of the fastest and feature-rich libraries for processing XML and HTML in Python.

To install lxml version 4.9.1, run the following command:

```
pip install lxml==4.9.1
```

Please follow the same procedure to install these python packages:

```
python -m venv ValidPath
pip install lxml==4.9.1
pip install opencv-python==4.8.1.78
pip install openslide-python==1.1.2
pip install scikit-image==0.18.1
pip install Shapely==1.7.1
pip install sharepy==2.0.0
pip install matplotlib==3.6.2
pip install Pillow==9.3.0
pip install tifffile==2022.10.10
pip install mpmath==1.2.1
pip install h5py
pip install scikit-learn
pip install openpyxl
pip install pandas
```

Alternatively, the required packages can be installed at once, rather than installing them one by one, using the following command:

```
pip install -r requirements.txt
```

For the full list of the requirements, please see the requirement.txt file in the project root directory

In order to check the current package version installed on you system, you can use "pip freeze" or ".__version__" as follows:

```
pip freeze | findstr lxml
```

or

```
import lxml

print(lxml.__version__)
```

2.1.2 Installation Using Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.Wikipedia

There are few steps to complete the installation. Firstly, you need to install Anaconda Navigator. This allows you to access to different Python IDEs and Python packages. When you install Anaconda Navigator, you may install your favorite IDEs such as Spider, PyCharm, and etc. You also will be able to create environment to have specific IDEs and Python packages for each project separately. Let's start with Anaconda Navigator.

Anaconda Navigator

In order to install Anaconda Navigator, download the Anaconda distribution from the following URL:

https://www.anaconda.com/products/distribution

2.1.3 Installing ValidPath using Anaconda

Open a terminal window.

```
$ cd ValidPath ROOT DIRECTORY
```

Download a complete copy of the ** ValidPath **.

```
$ git clone https://github.com/mousavikahaki/ValidPath
```

Change directory to ValidPath

```
$ cd ValidPath
```

Create virtual environment for ** ValidPath** using

```
$ conda env create -f requirements.dev.conda.yml
```

```
$ conda activate ValidPath-dev
```

or

```
$ conda create -n ValidPath python=3.8
```

```
$ conda activate ValidPath
```

```
$ pip install -r requirements.txt
```

To use the packages installed in the environment, run the command:

```
$ conda activate ValidPath-dev
```

2.1.4 Direct Installation of ValidPath

You can install required packages and then use pip to install the ValidPath.

Windows

1. Download OpenSlide binaries from this page. Extract the folder and add `bin` and `lib` subdirectories to Windows system path.

2. Install OpenSlide. The easiest way is to install OpenSlide is through pip using

```
C:\> pip install OpenSlide
```

3. Install ValidPath.

```
C:\> pip install ValidPath
```

Linux (Ubuntu)

On Linux the prerequisite software can be installed using the command

```
$ apt-get -y install libopenjp2-7-dev libopenjp2-tools openslide-tools
```

2.1.5 From Source

The source code of the slidepro toolbox can be accessed from the GitHub.

You can either clone the public repository:

```
$ git clone https://github.com/mousavikahaki/ValidPath.git
```

after downloading the source code of the slidepro toolbox, you can install it using the following command:

```
$ python setup.py install
```

2.2 Data (Input/Output) Examples

Example of these files can be found in the /data/_ directory.

Here are the input and output examples for each module:

2.2.1 WSI Handler

Input: Whole Slide Image file (.SVS,DICOM,TIFF)

Link to the Data File Example: SVSFile

Link to the Jupyter Notebook Example Script: WSI_Handler

Output: Processed file such as sub image region

2.2.2 Annotation Extraction

Input: Whole Slide Image file (.SVS,DICOM,TIFF) and annotation file (.xml)

Link to the WSI File Example: SVSFile

Link to the XML File Example: XMLFile

Link to the Jupyter Notebook Example Script: AnnotationExtraction

Output: Extracted annotation file

Link to the Output Data File Example: Annotation

2.2.3 Patch Extraction

Input: Extracted annotation file generated from Annotation Extraction Step

Link to the Input Data File Example: Annotation

Link to the Jupyter Notebook Example Script: PatchExtraction

Output: Extracted Patches

Link to the Output Data File Example: Patches

2.2.4 Annotation File Generator

Input: Extracted Patches generated from Patch Extraction Step

Link to the Output Data File Example: Patches

Link to the Jupyter Notebook Example Script: AnnotationGenerator

Output: Generated Annotation File (.xml)

Link to the Output Data File Example: xml_file

2.2.5 Performance Assessment

Input: Continuous classification scores generated by ML model and the binary truth

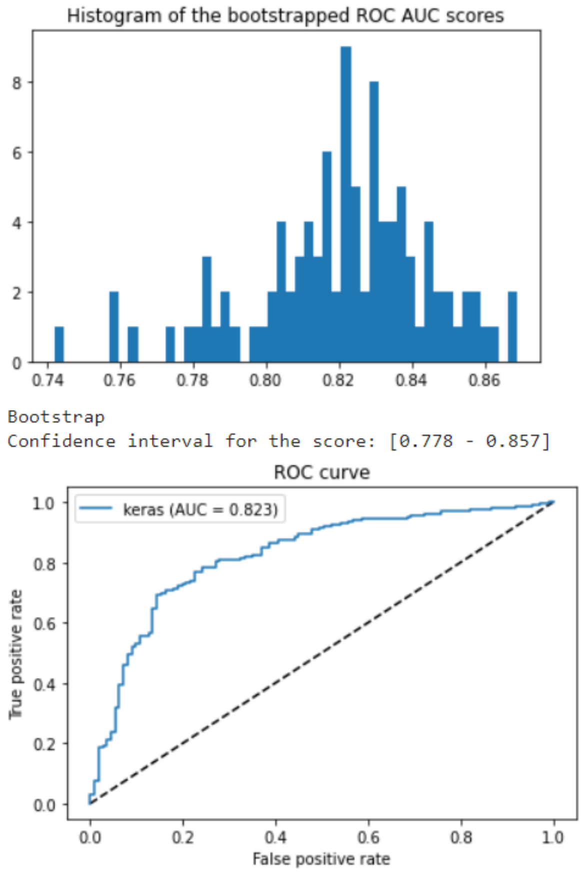
Link to the Output Data File Example: MLResultExample

Link to the Jupyter Notebook Example Script: PerformanceAssessment

Output: Performance Assessment Results

Example of the results output:

```
####################
Results for Test
          pred:yes  pred:no
true:yes       285       14
true:no         79       32
Precision:  0.782967032967033
Precision_CI:  (0.7406191220140836, 0.8253149439199823)
Recall:  0.9531772575250836
Recall_CI:  (0.9292315454409011, 0.9771229696092661)
Delong Method
AUC: 0.8234354756093887
AUC COV: 0.0005619434162873773
95% AUC CI: [0.77697385 0.8698971 ]
```



2.3 WSI Handler

2.3.1 WSI.readwsi module

Title: ValidPath Toolbox - WSI Handler module

Description: This is the WSI Handler module for the ValidPath toolbox. It is includes ReadWsi class and several methods

Classes: WSIReader

Methods: There are three methods in the ReadWSI module as follows:

- Reader: wsi_obj = WSIReader.wsi_reader(path)
- Region Extractor: WSIReader.extract_region(wsi_obj,location,level,size)
- Extract Bounds: WSIReader.extract_bounds(wsi_obj,bounds,level)

`class WSI.readwsi.WSIReader`

Bases: `object`

`extract_bounds`(*bounds*, *level*)

This method process the WSIs and extract image.

Parameters

wsi_obj
[object] recieve the WSI object

bounds
[tuple] recieve the locations for extracting image from WSI

level
[int] WSI level to extract image from

Returns

IMG
[Image] Image data

`extract_region`(*location*, *level*, *size*)

This method process the WSIs and extract regions.

Parameters

wsi_obj
[object] recieve the WSI object

Returns

IMG
[Image] Image data

`wsi_reader()`

This code read a WSI and return the WSI object. This code can read the WSIs with the following formats: Aperio (.svs, .tif) Hamamatsu (.vms, .vmu, .ndpi) Leica (.scn) MIRAX (.mrxs) Philips (.tiff) Sakura (.svslide) Trestle (.tif) Ventana (.bif, .tif) Generic tiled TIFF (.tif)

Parameters

WSI_path
[string] The address to the WSI file.

Returns

wsi_obj
[object] WSI object

`wsi_xml_list()`

This code process the WSIs and XML list and returns these lists. Only WSI are included if there is an XML file with the same name.

Parameters

wsis_dir
[string] Input Directory which has the original WSIs and XML files

Returns

WSIs
[list] List of included WSIs

xml_
[list] List of XML files associated with included WSIs

2.4 Annotation Extraction

2.4.1 WSI.annotation module

Title: ValidPath Toolbox - Annotation Extraction Module

Description: This is the Annotation Extraction module for the ValidPath toolbox. It is includes AnnotationExtractor class and several methods.

Classes: AnnotationExtractor

Methods: AnnotationExtractor.extract_ann (Save_dir: str,XMLs: array , WSIs: array)

`class WSI.annotation.AnnotationExtractor`

Bases: `object`

`extract_ann`(*save_dir*, *XMLs*, *WSIs*, *vis=False*, *save_mask=False*)

This method extracts different types for annotations from Whole Slide Images. It can save the extracted annottions to the output directory as defined in inputs. This code also handles several annotations per slide. The output directory will be generated based on the strucutr of the input directories.

Parameters

save_dir
[string] The path to the directory in order to save the annotations

WSIs
[list] List of whole slide image files

XMLs
[list] List of annotation files in XML format

Returns
Image (array) – annotation files

`make_folder`(*directory*)

This method creates a directory if not exist.

Parameters

directory
[string] Directory to be created.

Returns

None
[None] None.

2.5 Patch Extraction

2.5.1 WSI.patch module

Title: ValidPath Toolbox - patch extraction module

Description: This is the patch extraction module for the ValidPath toolbox. It is includes two classes and several methods

Classes: WSIpatch_extractor, PatchExtractor

Methods: There are two methods in the patch extraction module as follows:

- PatchExtractor.gen_patch(INPUTDIR: str, PatchSize: tuple, Number_of_Patches: int, intensity_check: boolean, OUTPUTDIR:str)
- WSIpatch_extractor.patch_extraction(wsi_obj: object, PatchSize: tuple, OUTPUTDIR:str, Random: boolean, Visualize: boolean, Intensity_check: boolean, Number_of_Patches: (int)

`class WSI.patch.PatchExtractor`

Bases: `object`

`find_between`(*s*, *first*, *last*)

`gen_patch`(*INPUTDIR*, *PatchSize*, *Number_of_Patches*, *intensity_check*, *intensity_threshold*, *OUTPUTDIR*)

This function extracts a number of pactches from extracted annotations. It can save the extracted annottions to the output directory as defined in inputs. Before running this function, please call annotation.ann_extractor.extract_ann(save_dir, XMLs, WSIs) to generate annotations. The output directory will be generated based on the strucutr of the input directories. IF the WSI Magnification is 13X or 20X, this code will automaticall convert to 20X.

Parameters

INPUTDIR
[string] the path to the input directory

PatchSize
[tuple] the size of image patches to be extracted

Number_of_Patches
[int] the number of patches per annotation to be extracted

intensity_check
[boolean] to filter the image patches and eliminate empty ones

- OUTPUTDIR : string the path to the output directory to save image patches

Returns

Image – extracted image patches from the annotated area.

`class WSI.patch.WSIpatch_extractor`

Bases: `object`

`patch_extraction`(*patch_size*, *output_folder*, *random_state*, *visualize*, *intensity_check*, *intensity_threshold*, *std_threshold*, *patch_number=-1*)

this function Generate object for tiles using the DeepZoomGenerator and divided the svs file into tiles of size 256 with no overlap. then processing and saving each tile to local directory.

Parameters

wsi_obj
[object] an object containing WSI file and its information

patch_size: integer
the size of image patches to be extracted

output_folder
[string] the path to the output directory to save image patches

random_state
[boolean] extract patches randomly or in order

visualize: boolean
either to plot extracted patches or not

intensity_check: boolean
to filter the image patches and eliminate empty ones

intensity_threshold: integer
the threshold to include image patches

std_threshold: integer
the standard deviation threhold to include image patches

patch_number
[boolean] the number of patches to be extracted. Set to '-1' to extract all possible image patches

`patch_extraction_of_tissue`(*patch_size*, *output_folder*, *number_of_patches=1*, *vis=False*)

`patch_extraction_with_normalized_tiles`(*patch_size*, *output_folder*, *random_state=True*, *patch_number=-1*)

this function Generate object for tiles using the DeepZoomGenerator and divided the svs file into tiles of size 256 with no overlap. then processing and saving each tile to local directory.

Parameters

wsi_obj
[object] WSI object.

patch_size: integer
size tiles

output_folder
[string] path root folder to save tiles

perform_segmentation_state: boolean random_state : boolean

2.6 Annotation File Generator

2.6.1 assessment.annotation module

Title: ValidPath Toolbox - Annotation File Generation Module

Description: This is the Annotation File Generator module for the ValidPath toolbox. It is includes Annotation_Generator class and several methods

Classes: Annotation_Generator

Methods: There are three methods in the Annotation File Generation module as follows:

- ROI_Generator.generate_map_file(input_DIR: str, output_DIR: str, file_Name: str)
- ROI_Generator.create_xml(input_DIR,file_Name,path_size,ROI_output_DIR,tag_name)
- make_region(self, x , y , id , txt,path_size,Regions)

`class assessment.annotation.`**`Annotation_Generator`**

Bases: `object`

`create_xml`(*input_DIR*, *file_path*, *path_size*, *save_xml_path*)

This method reads the map file generated uisng the ROI_Generator.generate_map_file and generated the XML annotation file based on Aperio ImageScope standard.

Parameters

input_DIR
[string] the path to the input directory of mapping file

file_path
[string] map file name (csv)

path_size
[integer] Size of image patch

save_xml_path: string
output directory

Returns
XML – the XML files

`generate_map_file`(*input_DIR*, *output_DIR*, *file_Name*, *tag_name*)

This method extracts different types for annotations from Whole Slide Images. It can save the extracted annottions to the output directory as defined in inputs. This code also handles several annotations per slide. The output directory will be generated based on the strucutr of the input directories.

Parameters

input_DIR
[string] the path to the input directory of image patches

output_DIR
[str] the path to the output directory to save the map file

file_Name
[string] map file name (csv)

tag_name
[string] Tag name

Returns
CSV – the map file

make_region(*x*, *y*, *id*, *txt*, *path_size*, *Regions*)
This method generate the XMl file structure and fill the content based on the Aperio ImageScope standard

Parameters

x
[integer] Output Directory to save the extracted annotations

x
[integer] List of included WSIs

txt
[string] List of XML files associated with included WSIs

path_size
[integer] Patch size

Regions
[object] the corresponsing XML region object

Returns
XML strycture

2.7 ML Assessment

2.7.1 assessment.uncertainty module

Title: ValidPath Toolbox - Uncertainty Analysis module

Description: This is the Uncertainty Analysis module of the ValidPath toolbox. It is includes Uncertainty_Analysis class and several methods

Classes: Uncertainty_Analysis

Methods: get_report, **auc_keras_**, **ci_**, Delong_CI, compute_midrank, compute_midrank_weight, calc_pvalue, compute_ground_truth_statistics, delong_roc_variance, bootstrapping

---

class assessment.uncertainty.**Uncertainty_Analysis**
Bases: object

Delong_CI(*y_pred*, *y_truth*)
A Python implementation of an algorithm for computing the statistical significance of comparing two sets of predictions by ROC AUC. Also can compute variance of a single ROC AUC estimate. X. Sun and W. Xu, "Fast Implementation of DeLong's Algorithm for Comparing the Areas Under Correlated Receiver Operating Characteristic Curves," in IEEE Signal Processing Letters, vol. 21, no. 11, pp. 1389-1393, Nov. 2014, doi: 10.1109/LSP.2014.2337313.

Parameters
y_truth: ground_truth - np.array of 0 and 1 y_pred: predictions - np.array of floats of the probability of being class 1

Returns
auc, ci, lower_upper_q, auc_cov, auc_std

`auc_keras_`(*fpr_keras*, *tpr_keras*)

Estimates confidence interval for Bernoulli p

Parameters
fpr_keras: False Positive Rate Values tpr_keras: True Positive Rate Values

Returns
AUC: Area Under the ROC Curve

`bootstrapping`(*y_true*, *y_pred*)

Computes ROC AUC variance for a single set of predictions

Parameters
ground_truth: np.array of 0 and 1 predictions: np.array of floats of the probability of being class 1

`calc_pvalue`(*aucs*, *sigma*)

Computes log(10) of p-values.

Parameters
aucs: 1D array of AUCs sigma: AUC DeLong covariances

Returns
log10(pvalue)

`ci_`(*tp*, *n*, *alpha=0.05*)

Estimates confidence interval for Bernoulli p

Parameters
tp: number of positive outcomes, TP in this case n: number of attemps, TP+FP for Precision, TP+FN for Recall alpha: confidence level

Returns
Tuple[float, float]: lower and upper bounds of the confidence interval

`compute_ground_truth_statistics`(*ground_truth*, *sample_weight*)

`compute_midrank`(*x*)

Computes midranks.

Parameters
x - a 1D numpy array

Returns
array of midranks

`compute_midrank_weight`(*x*, *sample_weight*)

Computes midranks.

Parameters
x - a 1D numpy array

Returns
array of midranks

`delong_roc_variance`(*ground_truth*, *predictions*, *sample_weight=None*)

Computes ROC AUC variance for a single set of predictions

Parameters
ground_truth: np.array of 0 and 1 predictions: np.array of floats of the probability of being class 1

`fastDeLong`(*predictions_sorted_transposed*, *label_1_count*, *sample_weight*)

`fastDeLong_no_weights`(*predictions_sorted_transposed*, *label_1_count*)

The fast version of DeLong's method for computing the covariance of unadjusted AUC.

Parameters

predictions_sorted_transposed: a 2D numpy.array[n_classifiers, n_examples]
sorted such as the examples with label "1" are first

Returns
(AUC value, DeLong covariance)

Reference:

@article{sun2014fast,
title={Fast Implementation of DeLong's Algorithm for Comparing the Areas Under Correlated Receiver Oerating Characteristic Curves}, author={Xu Sun and Weichao Xu}, journal={IEEE Signal Processing Letters}, volume={21}, number={11}, pages={1389–1393}, year={2014}, publisher={IEEE}

}

`fastDeLong_weights`(*predictions_sorted_transposed*, *label_1_count*, *sample_weight*)

The fast version of DeLong's method for computing the covariance of unadjusted AUC.

Parameters

predictions_sorted_transposed: a 2D numpy.array[n_classifiers, n_examples]
sorted such as the examples with label "1" are first

Returns
(AUC value, DeLong covariance)

Reference

@article{sun2014fast,
title={Fast Implementation of DeLong's Algorithm for Comparing the Areas Under Correlated Receiver Oerating Characteristic Curves}, author={Xu Sun and Weichao Xu}, journal={IEEE Signal Processing Letters}, volume={21}, number={11}, pages={1389–1393}, year={2014}, publisher={IEEE}

}

`get_report`(*y_pred*, *y_truth*)

This method recieve the machine learning prediction output and the ground truth and report several metrics. This is the main metod of the Uncertainty_Analysis class which calls other methods to procude results.

Parameters
y_truth: ground_truth - np.array of 0 and 1 y_pred: predictions - np.array of floats of the probability of being class 1

Returns
precision Precision Conficenc Interval Recall Recall Conficenc Interval AUC based on delong method and its Conficenc Interval and COV False Positive Rate True Positive Rate AUC Confusion Matrix

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

w

INDEX

P

U

W