There is more than one way to run PyLC: on a personal computer or on a Compute Canada server.

Coding 101: **All folders and filenames must NOT have spaces in them.** Use hyphens, underscores, or capital letters to separate words!

`Note: anything in this font is what you type into the terminal`
`Anything highlighted needs to be changed`
All lines provided are one line that goes into the terminal press enter after you have input all of the text `in this font`

Additional resources
Linux commands:
https://www.hostinger.com/tutorials/linux-commands

Add instructions for installing anaconda and setting up Python

How to: Using Github

Last updated August 31st, 2023 by Kristyn Lang

Cloning PyLC to home computer
The easiest way to use PyLC (for merging, testing, training) is by cloning the Github repository for PyLC. This will ensure you have all the necessary files for every step.

1. Open the terminal and navigate to the directory (folder) you would like pylc to live in
    a. Ex: mine lives in /Users/kristynlang/Desktop/kristyn/python
2. Type git clone --branch master https://github.com/kristynlang/pylc_v2.0.git
    a. This version has the updated model and mask merging functions

`git clone --branch master https://github.com/kristynlang/pylc_v2.0.git`
Or
`git clone --branch master https://github.com/scrose/pylc.git` if cloning the original version

3. This will create a folder called pylc_v2.0 (or pylc if cloning the original version) in the directory you chose in step 1
4. You will need to create a virtual environment in Python to run pylc containing the python packages listed in requirements.txt
    a. `conda create -n myenv` (where myenv is the name of your environment, can be anything you want ex. pylc or mlp)
    b. `conda activate myenv`
    c. `cd pylc_v2.0`
    d. `pip install -r requirements.txt`
    e. If using original pylc from scrose you will need to individually install each package in the requirements.txt file
        i. Ex. `pip install numpy`

Using Github to update the Model
If you want to make changes to the code base (update the model), you need to **fork** the repository before cloning it:

Instructions for getting started with Github:
https://docs.github.com/en/desktop/overview/creating-your-first-repository-using-github-desktop

1. Create a Github account if you do not have one already: https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home
2. Navigate to https://github.com/kristynlang/pylc_v2.0/
3. Click "fork" on the top right corner of the webpage
    a. This creates a copy of the main branch that you can edit and push changes to without affecting the original repository
4. Download Github for desktop: https://desktop.github.com/
5. Then you will want to clone your forked repository to your desktop either through the Github gui (graphical user interface) or through the terminal
    a. You can choose the path that Cloned repositories will save to, such as: /Users/username/Desktop/Github/repo_name
6. To clone via the terminal:
7. Create a folder for Github code somewhere that makes sense, mine goes in my desktop folder
    a. `cd /Users/username/Desktop/`
    b. Create github folder
    `mkdir Github/`
    Navigate to the github folder
    `cd Github/`
    c. Clone:

`git clone --branch master https://github.com/username/repo_name.git`

Where repo_name is the name you choose for your forked repository (can match original ex. pylc_v2.0)

8. This will create a folder called repo_name (whatever you decided to name the repository, ex. pylc_v2.0) in the directory `/Users/username/Desktop/Github/`
9. You will need to create a virtual environment in Python to run pylc containing the python packages listed in requirements.txt
    a. `conda create -n myenv` (where myenv is the name of your environment, can be anything you want ex. pylc or mlp)
    b. `conda activate myenv`
    c. `cd repo_name`
    d. `pip install -r requirements.txt`
    e. If using original pylc from scrose you will need to individually install each package in the requirements.txt file - **Things like this is why you should fork https://github.com/kristynlang/pylc_v2.0/ if you are continuing work on the model
        i. Ex. `pip install numpy`

10. You can edit and run code from your home computer, once changes are finalised open Github desktop and click “fetch origin”. Any changes you have made will show up in the left hand side under changes. When you are done making sure all the changes are good press “commit to master” to push your changes to your forked repository on your Gitub. This will not affect the original repository from which you created your fork. You first must title and leave a description for the changes you have made.
11. You can edit code directly through the terminal or by using Jupyter notebook.
12. NOTE: You won’t ever be able to commit the file **resnet101-5d3b4d8f.pth** in the **./models/** folder. The file size is too large. You can either commit this change to .gitignore or uncheck this change when committing new changes to the master. This file **must be downloaded** to run training (see steps 3 and 4 in “How to run pylc training on Cedar”). Please leave the instructions and link for download in the main README file under training.

How to: Running PyLC on a Personal Computer (using Github method)

1. Either clone the original repository (see steps 1 - 4 in **Cloning PyLC to home computer**) or your forked repository (depending on your purpose – see steps 1 - 8 in **Using Github to update the Model**)
    a. Follow steps all the way through to set up your python environment with all of the requirements necessary to run PyLC
2. Put all images and masks in a folder (ex. training or testing). Note: all images and masks should have the following filename convention: Dataset_Location_H.png for masks and Dataset_Location_H.tif (or .jpg) for images. The filename for each mask/image pair MUST be the same.
    a. Filepath example: /Users/username/Desktop/Images/training
    b. If filenames DO NOT MATCH use the **Script for renaming files**(this works if the files are Mask_Dataset_Location_H.png for masks and Img_Dataset_Location_H.tif for images) other filename formats will need to be renamed manually or code needs to be updated accordingly
3. Make sure all masks from different datasets have the same categorization scheme, if not use the merging tool: https://github.com/kristynlang/pylc_v2.0/tree/master/merging
    a. Follow the steps outlined in README.md
    b. Note: if you have cloned a repository you **do not** need to download these files, you can open merge_class.ipynb in Jupyter notebook and edit/run the script from there. The necessary .py files will be in the same folder on your computer and accessible by the script.
    c. Make sure to make your environment accessible to Jupyter notebook by creating a kernel:
        i. `pip install ipykernel`
        ii. `python -m ipykernel install --user --name=myenv`

Training

Note: these steps are outlined in the README.md on Github: https://github.com/kristynlang/pylc_v2.0/

1. Download Resnet pretrained weights from https://download.pytorch.org/models/resnet101-5d3b4d8f.pth
2. Copy file into the pylc_v2.0/models/ directory (first navigate to your downloads folder on your home computer)
    a. `cd /Users/username/Downloads`
    b. `cp *f.pth [pylc/models/location]`
        i. Ex. `cp *f.pth /Users/username/Desktop/pylc_v2.0/models`
        ii. Filepath will depend on where you cloned the pylc repository to previously and how you named the repo
3. Navigate to the directory containing pylc:
    a. Ex: `cd /Users/username/Desktop/Github/pylc_v2.0`
    b. ***To find the file path for any directory you can navigate to it using `ls` to list what is inside the directory you are in in the terminal and `cd dirname` (where dirname is the name of a folder) to navigate
    c. Once you are in the directory/folder of interest you can type `pwd` to get the full file path
4. Extract image tiles for training:
    a. `yes | python pylc.py extract --ch [number of channels] --img [path/to/image(s)] --mask [path/to/mask(s)]`
    b. Yes automatically grayscales any images that are RGB if --ch 1 is selected
    c. Number of channels is 1 for historic (black and white) and 3 for repeat (colour)
    d. Path to images and masks is the filepath for the folder you created with image/mask pairs
        i. Ex. /Users/username/Desktop/Images/training

Example for historical training (all one line):
```
python pylc.py extract --ch 1 --img
/Users/username/Desktop/Images/training --mask
/Users/username/Desktop/Images/training
```

5. Path to database file needed for next step is output at the bottom of the terminal once the extraction step is completed: `./data/db/db_filename.h5`
    a. Copy this file path for next step
6. Augmentation **this step can be skipped but is not recommended
    a. `python pylc.py augment --db ./data/db/db_filename.h5`
    b. This will also output a file path for a .h5 file that is needed for training in the format: `./data/db/db_filename.h5`
7. To avoid an OS error when performing training type the following into the terminal:
    a. `ulimit -Sn 2048`
8. Training
    a. `python pylc.py train --db ./data/db/db_filename.h5 --ce_weight 0.269 --dice_weight 0.196 --focal_weight 0.059 --n_workers [number of cores]`

    b. Weights can be changed but requires analysis (produce different accuracies). The ones selected here had the highest accuracy out of the ones tested for historic images in Spencer's thesis. These will be different for repeats.
        i. Find other options and weights for repeats (Tables 6.7 and 6.8) here: https://dspace.library.uvic.ca/handle/1828/12156
        ii. Highest accuracy repeat weights: `--ce_weight 0.215 --dice_weight 0.136 --focal_weight 0.036`
    c. The number of cores on your computer can be found googling the technical specifications of your computer (ex. 2023 mac mini has a 12-core CPU, so n_workers should be 12)
9. Trained model will be saved as a .pth file in a folder like ./data/save/pylc_deeplab_ch1_schema_a
    a. Note if using a different architecture or number of channels the folder name will be a little different (i.e. pylc_deeplab_ch3_schema_a for repeats)
    b. Make sure to update folder name to something that is distinguishable, if you run training again without changing the folder name it will be overwritten by the newer trained model. You want to be able to remember what was different about each trained model (images, parameters, etc.)
    c. You can navigate to these folders using the gui on your computer (ex. Finder on mac) or through the terminal
        i. `cd data/save/pylc_deeplab_ch1_schema_a`

Testing

1. Navigate to folder containing pylc
    a. Ex: `cd /Users/username/Desktop/Github/pylc_v2.0`
    b. ***To find the file path for any directory you can navigate to it using `ls` to list what is inside the directory you are in in the terminal and `cd dirname` (where dirname is the name of a folder) to navigate
    c. Once you are in the directory/folder of interest you can type `pwd` to get the full file path
2. `python pylc.py test --model [model/path] --img [image/path] --mask [mask/path] --aggregate_metrics [True/False]`
    a. If using a new trained model, the model path will be ./data/save/renamed_pylc_deeplab_ch1_schema_a
    b. If using one of Spencer's trained models the model file path will be wherever you saved the model to on your computer
        i. Download PyLC pre-trained models from the Read Me section of the GitHub (https://github.com/scrose/pylc).
    c. Model and image path will be wherever your test set folder is located. Use PyLC-SummerTestSet for consistency with previous work, these images have been analysed for accuracy and spread of classes.
        i. Ex. `Users/username/Desktop/Images/PyLC-SummerTestSet`
    d. Aggregate_metrics = False gives accuracy metrics for each individual image/mask pair, aggregate_metrics = True gives accuracy metrics for the whole dataset. True is recommended for faster comparison of improvements between models.

e. Output is saved to ./data/outputs/pylc_deeplab_ch1_schema_a
    i. Different folders for output masks and metrics
    ii. Update folder name after testing each model (if this is not done the folder will be overwritten every time a new test is done) and change the folder name to match the model folder name for consistency

How to: Running PyLC on a Compute Canada server (Cedar, Graham, etc.)

Last updated September 28th, 2023 by Kristyn Lang

Using Python on Cedar and running jobs:
https://docs.alliancecan.ca/wiki/Python
https://docs.alliancecan.ca/wiki/Running_jobs

NOTE: Cedar clears your **scratch** directory (the directory you will run the model in) once in a while so it is a good idea to keep a backup in your projects directory!

From main directory: type `cd`
`cp scratch /home/username/projects/def-ehiggs/username/scratch_copy`

"Files on the scratch filesystem that have not been used in over 60 days are automatically purged as per our filesystem policies described at https://docs.computecanada.ca/wiki/Storage_and_file_management#Filesystem_Quotas_and_Policies"

Setting up Cedar:

1. Open the terminal (command prompt on Windows, terminal on Mac)

Log onto Cedar via the terminal: `ssh username@cedar.computecanada.ca`

   a. `Note: anything in this font is what you type into the terminal`
   b. `Anything highlighted needs to be changed`
   c. If using another server, replace cedar with the name of that server (ex. Graham)
   d. Press enter to run in the terminal

2. Say yes to "are you sure you want to continue connecting"
3. Enter password
4. Load in Python *steps 5-7 **must be done in the main directory** (to navigate here type `cd`)
   a. Check directory with `pwd` should be /home/username
   b. `module load python`
5. Create a virtual environment:
   a. `virtualenv --no-download env_name` (note env_name is whatever you want, ex. MLP)
6. Activate your environment
   a. `source env_name/bin/activate`
7. Upgrade pip
   a. `pip install --no-index --upgrade pip`

Getting PyLC set up on Cedar:

8. Navigate to the scratch directory
9. `cd scratch`
    a. (the directory you will run the model in) - type out, wont work if copy and paste

`git clone --branch master https://github.com/kristynlang/pylc_v2.0.git`

    b. This version has the updated model and mask merging functions
    c. git clone --branch master https://github.com/scrose/pylc.git if cloning the original version of pylc
10. This will create a folder called pylc_v2.0 (or pylc if cloning the original version) in your scratch directory
11. Navigate to the directory containing pylc
    a. `cd pylc_v2.0`
12. Installing python package requirements for PyLC:
    a. `pip install -r requirements.txt --no-index`
    b. You should only have to do this once
    c. No index ensures the dependencies on Cedar are consistent (without it won't work)
    d. You will get an error about opencv - follow steps below
13. Also need to install cv2 (this step needs to be done every time you run pylc on Cedar)
    a. `module load StdEnv/2020 gcc/9.3.0 opencv/4.7.0 python`
    b. `python -c "import cv2"`

How to set up pylc testing on Cedar

(with pre-existing models trained by Spencer):

14. Move images and presaved models from Spencer onto Cedar, these need to be on your home computer first
    a. Download PyLC pre-trained models from the Read Me section of the GitHub (https://github.com/scrose/pylc) - click on the filename to download
        i. Move the pretrained models to a folder in your home directory that makes sense - I keep mine in a folder called models in the same directory as pylc
        ii. Ex. /Users/kristynlang/Desktop/kristyn/python/models
    b. Create a directory on Cedar in your scratch directory to hold the presaved models
        i. Navigate back to your scratch directory using `cd ..` or `cd /home/username/scratch/`
        ii. `mkdir presaved_models`
        iii. `cd presaved_models`
        iv. `pwd` to grab filepath
        v. Open a new terminal window on your home computer
        vi. ls to list what is in your directory
        vii. Navigate to the filepath container the models using cd
        viii. `cd desired/filepath`

c. Copy the presaved models from your home computer to Cedar using the command:

`scp * username@cedar.computecanada.ca:/desired/filepath/`

i. Make sure you are in the directory containing the models (ex. /Users/kristynlang/Desktop/kristyn/python/models for me)
ii. Ex.kristynl@cedar.computecanada.ca:/home/kristynl/scratch/MLP/models/
iii. Can also scp a specific filename, typing * means everything in that directory will be copied over, typing test* will copy over every file starting with the word test, *test* will copy over every file with the word test somewhere in the filename, to copy over every file with a certain file extension use *.ext (ex. *.png)

d. Enter password
e. Now copy the images for your test set over to Cedar
i. Images should live in your projects folder: /home/username/projects/def-ehiggs/username
f. Navigate to directory container the desired folder of images on you home computer using cd
i. Ex: `cd /Users/kristynlang/Desktop/kristyn/Images`
g. Copy desired folder to Cedar

`scp -r folder_name /home/username/projects/def-ehiggs/username/`

i. **Update username to yours**
ii. -r means recursive - necessary since you are copying a folder and not a file

15. Go back to open Cedar terminal
16. Set up an sbatch script to submit jobs in the queue (pylc testing must be submitted as a job)
17. Open a .sl file
a. `vi pylc_test.sl`
b. Quick how to vi in Linux:
i. To type in the new file hit the “i” key
ii. Press esc to exit out of typing mode
iii. To save type `:w` To quit type `:q` To save and quit type `:wq` To quit without saving type `:q!`
c. Copy and paste the following into the new file (make sure you hit insert key i)

```
#!/bin/bash -l

#SBATCH --job-name=desired_job_name
#SBATCH --account=def-ehiggs
#SBATCH --time=0-05:00:00
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=6G
#SBATCH --output=pylc.log
#SBATCH --mail-user=email@email.com
#SBATCH --mail-type=all
```

yes | python pylc.py test --model /home/username/scratch/presaved_models/pylc_2-1_deeplab_ch1_schema_a.pth --img /home/username/projects/def-ehiggs/username/ImageFolder --mask /home/username/projects/def-ehiggs/username/ImageFolder

18. Edit the job name and email, everything else listed in #SBATCH should stay the same
    a. Cpus per task and mem-per-cpu can be changed to see if you can get a faster/more efficient run time (experimental)
19. Edit the directories for the model, images, and masks to match the locations on your own Cedar account (should be similar if you followed my steps above)
    a. **If using a model you trained** instead of one on github from Spencer the file path will be something like ./data/save/pylc_deeplab_ch1_schema_a (see step 2 in **Testing** and step 6 in **Training** in **How to: Running PyLC on a Personal Computer**)
        i. Make sure to rename this folder if training more models!
20. Time is how much time you think the job will complete by, I usually go over a bit. PyLC takes about 2 hours to test the PyLC-SummerTestSet on Whitebark
21. To run type `sbatch pylc_test.sl`
22. To check on the job status type `sq`
23. You will get a SLURM email when the job begins and will receive emails if the job fails or is complete
24. If the job fails you can check the error by opening the pylc.log file in the pylc_v2.0 directory on Cedar
    a. `vi pylc.log`

Now that everything is set up fewer steps need to be taken to run further testing…
Running testing on Cedar after set up:

1. Log onto Cedar via the terminal: `ssh username@cedar.computecanada.ca`
2. Enter password
3. Load in Python
    a. `module load StdEnv/2020 gcc/9.3.0 opencv/4.7.0 python`
4. Activate your environment
    a. `source env_name/bin/activate`
5. Load in open cv
    a. `python -c "import cv2"`
6. Navigate to directory pylc lives in
    a. `cd /home/username/scratch/pylc_v2.0`
7. Run sbatch script `sbatch pylc_test.sl`
    a. Edit sbatch script file paths if using different a different model or different image folder (will have to follow step 13 from last section if adding new images or models)
8. This puts you in the queue on Cedar, so be patient sometimes it is quick and sometimes it is not!

How to run pylc training on Cedar:

1. Follow the same steps (1 - 6) as running testing (above) if already set up on Cedar

2. If not set up on Cedar, follow steps 1 - 12 for setting up Cedar (from top of document)
3. Download Resnet pretrained weights from https://download.pytorch.org/models/resnet101-5d3b4d8f.pth
4. IF you get an error and safari cannot connect to the webpage then Connect to the UVic VPN via Cisco Any Connect
5. Open new terminal window
6. Copy file to Cedar into the pylc_v2.0/models/ directory (first navigate to your downloads folder on your home computer)
    a. `cd /Users/username/Downloads`

```
scp *f.pth
username@cedar.computecanada.ca:/home/username/scratch/pylc_v2.0/models
```

    b. The above is one line in the terminal
7. Copy training images from home computer to Cedar
    a. Navigate to directory containing folder for training images on your computer
    b. Ex. `cd /Users/kristynlang/Desktop/kristyn/Images`

```
scp -r folder_name
username@cedar.computecanada.ca:/home/username/projects/def-ehiggs/user
name/
```

    c. This will copy over the folder into this directory (with the same folder name)
8. Create sbatch scripts for tile extraction, augmentation, and training
9. `vi pylc_extract.sl`
    a. Copy and paste into open terminal window:

```
#!/bin/bash -l

#SBATCH --job-name=PyLC_extract
#SBATCH --account=def-ehiggs
#SBATCH --time=0-01:00:00
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=6G
#SBATCH --output=pylc_extract.log
#SBATCH --mail-user=email
#SBATCH --mail-type=all
yes | python pylc.py extract --ch 1 --img /home/username/projects/def-ehiggs/username/training_foldername --mask
/home/username/projects/def-ehiggs/username/training_foldername
```

    b. Update email and folder names to match your own
        i. Press “i” key to insert and then you can delete and type
    c. Esc and type “:wq” to save and quit
10. `sbatch pylc_extract.sl` to run
11. This will submit tile extraction as a job, wait for SLURM email that says job is complete (use sq to check job status in the terminal)
12. Once job is completed open pylc.log
13. `vi pylc_extract.log`
14. Shift + g to move to the bottom of the log file
15. Copy file path `./data/db/db_filename.h5` provided in ~line 551 of pylc_extract.log file
16. esc then :q to quit
17. `vi pylc_augment.sl`

Copy and paste the following into the blank terminal window

```
#!/bin/bash -l

#SBATCH --job-name=PyLC_augment
#SBATCH --account=def-ehiggs
#SBATCH --time=0-01:00:00
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --cpus-per-task=32
#SBATCH --mem-per-cpu=6G
#SBATCH --output=pylc_augment.log
#SBATCH --mail-user=email
#SBATCH --mail-type=all

python pylc.py augment --db ./data/db/db_filename.h5
```

   a. Replace `./data/db/db_filename.h5` with filepath provided in line 551 of pylc_extract.log file

18. Update email
19. Esc :wq to save and quit
20. `sbatch pylc_augment.sl` to run
21. Create sbatch training file
22. `vi pylc_train.sl`

Copy and paste the following into the blank terminal window

```
#!/bin/bash -l

#SBATCH --job-name=PyLC_train
#SBATCH --account=def-ehiggs
#SBATCH --time=0-48:00:00
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --cpus-per-task=32
#SBATCH --mem-per-cpu=6G
#SBATCH --output=pylc_train.log
#SBATCH --mail-user=email
#SBATCH --mail-type=all
#edit path to database

python pylc.py train  --db ./data/db/db_filename.h5 --ce_weight 0.269 --dice_weight 0.196 --focal_weight 0.059 --n_workers 32
```

23. Update email
24. --n_workers is an integer that represents the number of processors/cores you want to use, can set to a maximum of 32 on cedar (32 cores per node)
25. Replace `./data/db/db_filename.h5` with filepath provided in line 118 of pylc_augment.log file
26. Esc :wq to quit
27. `vi pylc_augment.log`
28. Press shift + g to move to the end of the file (line118 is here) and copy, esc :q to quit
29. Reopen pylc_train.sl `vi pylc_train.sl`
30. Paste --db filepath
31. Esq :wq to quit
32. Run training `sbatch pylc_train.sl`
33. Trained model will be saved as a .pth file in a folder like ./data/save/pylc_deeplab_ch1_schema_a

a. Note if using a different architecture or number of channels the folder name will be a little different (i.e. pylc_deeplab_ch3_schema_a for repeats)
b. Make sure to update folder name to something that is distinguishable, if you run training again without changing the folder name it will be overwritten by the newer trained model. You want to be able to remember what was different about each trained model (images, parameters, etc.)
c. You can navigate to these folders through the terminal
    i. `cd ./data/save/pylc_deeplab_ch1_schema_a` from the pylc_v2.0 directory

Script for renaming files

named Mask_Dataset_Location_H.png and Image_Dataset_Location_H.tif (or .jpg) to have correct convention for use with PyLC (Dataset_Location_H.png and Dataset_Location_H.tif)

1. This script can go anywhere on your personal computer but pick a spot that makes sense, navigate to where you want the shell script to be saved using `cd`
2. `vi rename.sh`
3. Copy the following and paste into the blank window in the terminal:

```
# Check if the folder path argument is provided
if [ $# -eq 0 ]; then
    echo "Please provide the folder path as an argument."
    exit 1
fi
# Assign the folder path from the argument
folder_path="$1"
# Change to the directory
cd "$folder_path" || exit 1
# Rename the image files
for file in Img_*; do
    new_name="${file#Img_}"
    mv "$file" "$new_name"
done
# Rename the mask files
for file in Mask_*; do
    new_name="${file#Mask_}"
    mv "$file" "$new_name"
done
```

4. Press esc then type `:wq` to save and quit
5. To run the shell script type `chmod a+x rename.sh` into the terminal
6. `./rename.sh [image/mask/folderpath]` to run

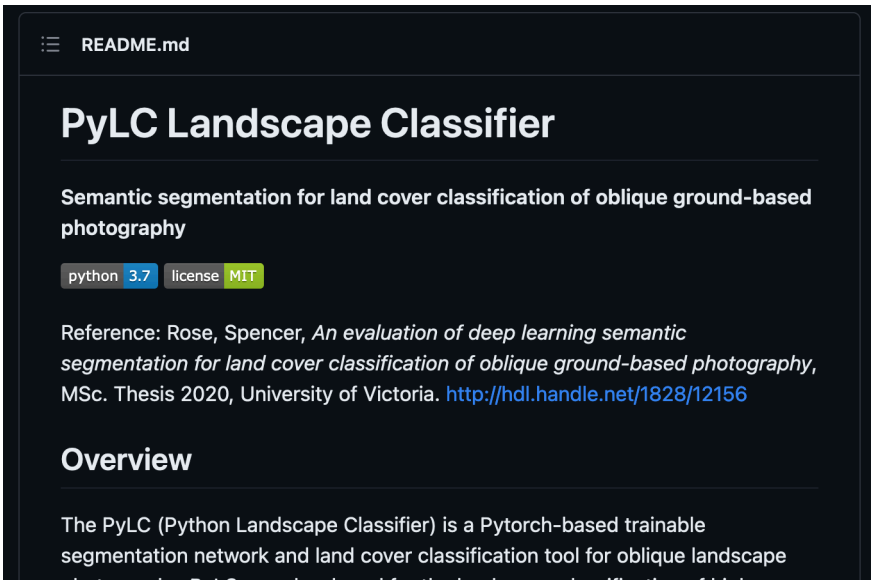
Ex. `./rename.sh /Users/kristynlang/Desktop/kristyn/Images/3_Rose2020Training`

How to: Running PyLC on a Personal Computer for Testing Spencer's Original Models Only – without Github (Mac)

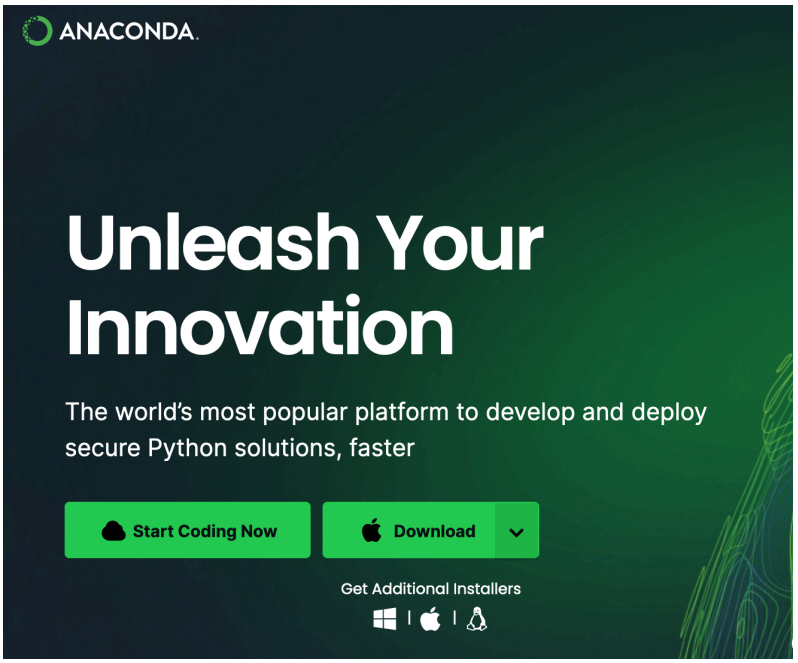
Last updated August 10th, 2023 by Larissa Bron

Steps to run PyLC if you are running a pre-existing model:

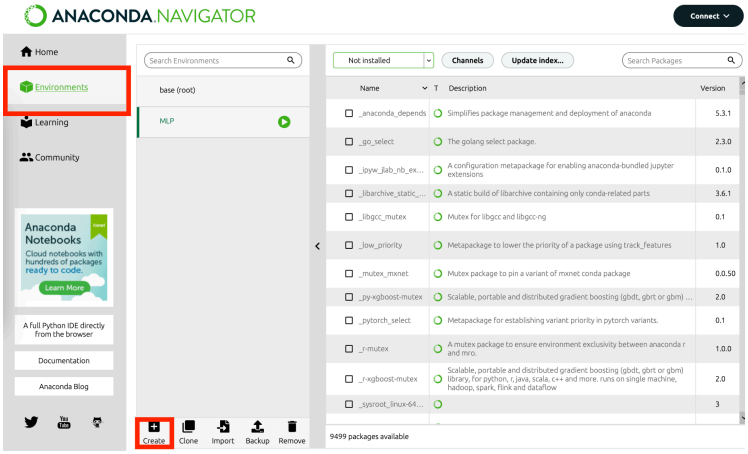
1. Familiarise yourself with Spencer Rose's GitHub where PyLC is located: https://github.com/scrose/pylc. The ReadMe contains most of the information we will work with.



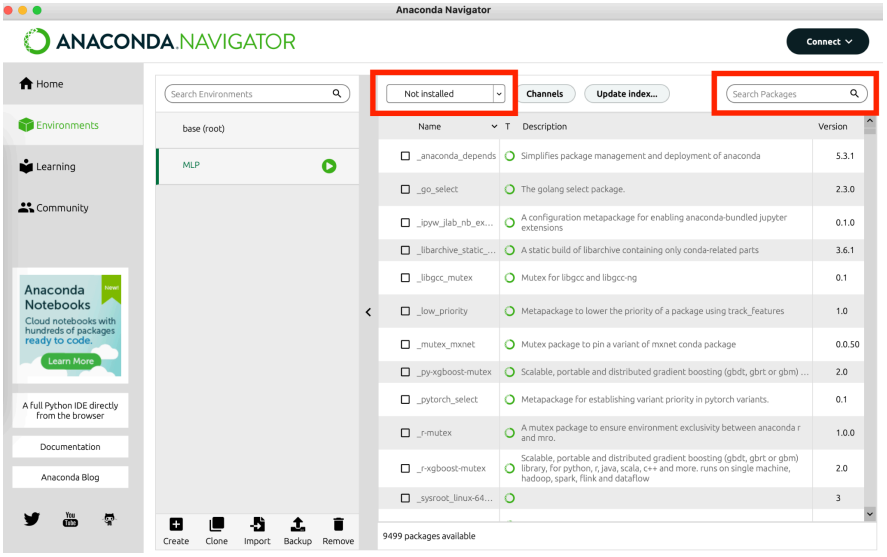
2. Download Anaconda Navigator: https://www.anaconda.com/.



3. Create a virtual environment in Anaconda by opening the application -> Environments -> Create. Pick a name like MLP or PyLC.

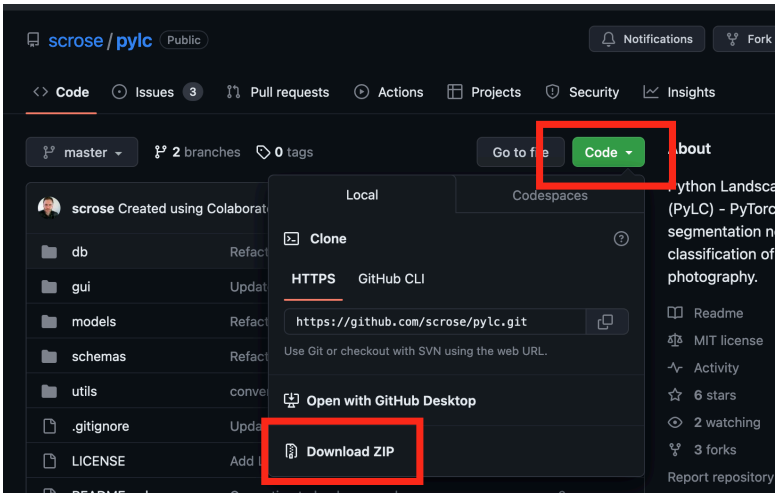


4. Download the requirements to run PyLC. Select the environment name from Step 3, select “not installed” from the drop down menu, and the search for each of the packages below and download them into the virtual environment.

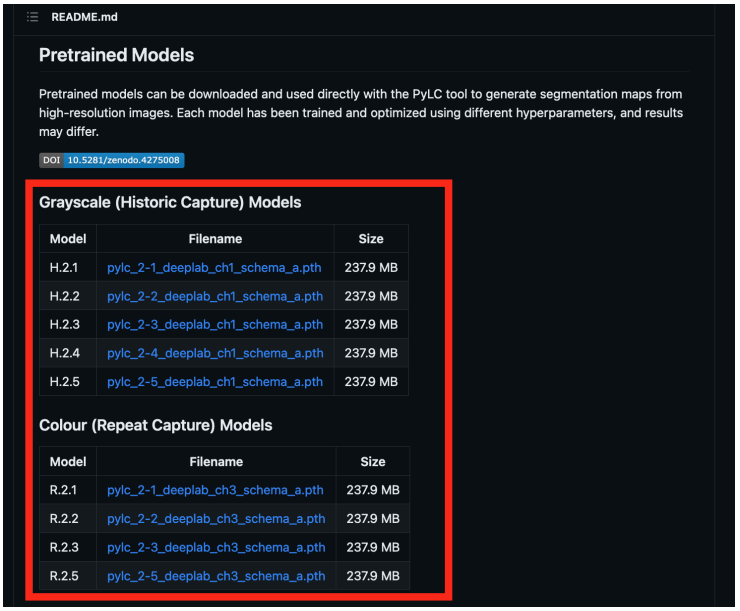


Requirements are listed in the ReadMe. As of writing, the requirements are: numpy >=1.18.5, h5py >= 2.8.0, opencv >=3.4.1, torch >=1.6.0, seaborn >=0.11.0(optional - evaluation), matplotlib >=3.2.2 (optional - evaluation), scikit-learn >=0.23.1(optional - evaluation), and tqdm >=4.47.1.

5. Download PyLC from GitHub (https://github.com/scrose/pylc) by clicking on the Code button and selecting download zip file. Then uncompress the zip file, should be named “pylc-master,” and put it into a permanent location on your computer.



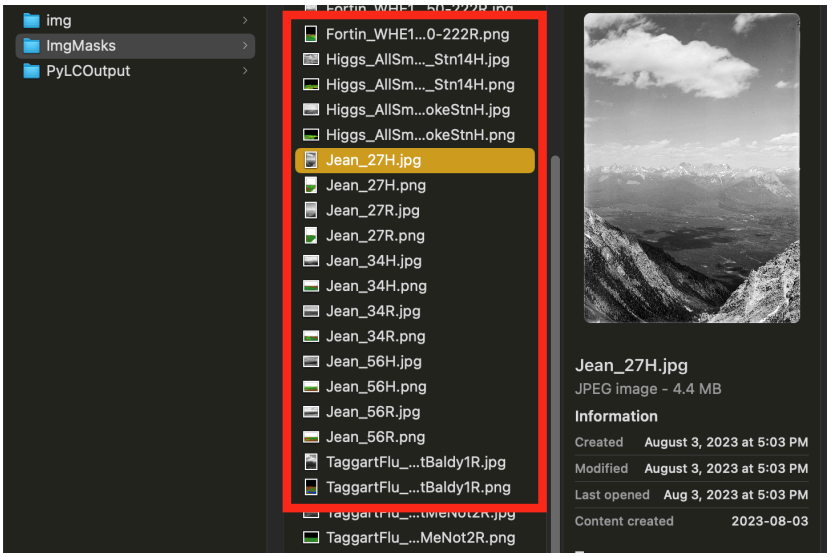
6. Download PyLC pre-trained models from the Read Me section of the GitHub (https://github.com/scrose/pylc).



7. Prepare your folder of images for sending through PyLC, depending on what you want PyLC to do for you. It can:

    a. Classify ecosystems in an image if you create a folder of just images.

    b. Additionally, it can compare its own classification with a set of user-created land cover masks. To do this, the images and masks must have the exact same file name, except the images will have a .tif or .jpg extension and the masks will be .png.

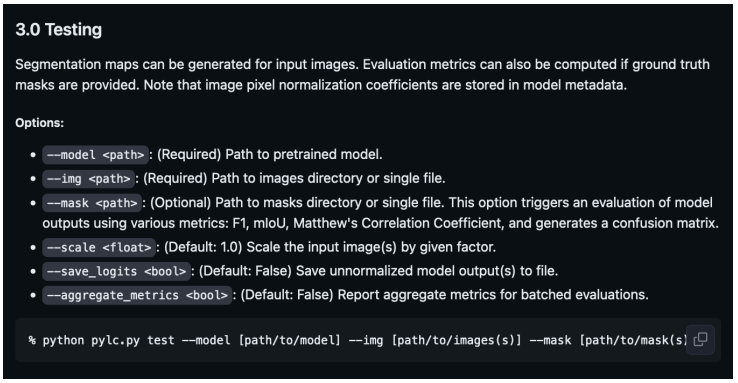


As well, the masks must follow the same colour scheme as PyLC understands, which can be done usually through the mask merging utility (https://github.com/kristynlang/pylc_v2.0/tree/master/merging) or manually (How to: Manual PyLC Mask Correction (Mar 2022)).

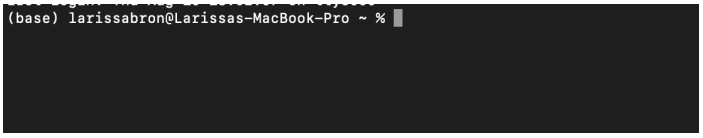
LC Category	Colour	HEX	Description
Coniferous forest		228b22	Greater than 75% coniferous forest.
Barren Rock		8b4513	Soil, sand, gravel, or rock.
Water		0000ff	6% or greater flowing or standing water
Regenerating Area		ff0004	Fire boundaries clearly identified or clear sign of recent timber harvesting.
Upland herbaceous/shrub		7cfc00	Upland herbaceous: Less than 25% shrub cover; less than 6% tree cover Upland shrub: Greater than 25% shrub cover, less than 6% tree cover.
Wetland		5f9ea0	"Wet" or aquatic moisture regime
Broadleaf/Mixed wood		ffa500	Broadleaf: Greater than 75% broadleaf trees Mixedwood forest: 26%-74% broadleaf trees, rest predominantly conifer
Snow/Ice		2dbdff	Permanent ice and snow.
Not Categorized		000000	Not included in mask creation - typically foreground, sky, and image imperfections - things like crease marks on historic scans.

8. Open terminal and input the following (replace anything in red):
    a. Conda update conda
    b. Conda activate name of your env
    c. cd path to the pylc-master folder
    d. python pylc.py test --model path to model of choice --img path to your folder of images

You can add more options to the script in d. based on Section 3.0 in the GitHub Read Me (https://github.com/scrose/pylc).



9. When PyLC is finished running the terminal screen will no longer show PyLC progress changing and the text input will be open again.



10. PyLC created masks and accuracy metrics (if prompted) will be available in the pylc-master folder -> data -> outputs.