Creating A Mask-RCNN Worm Training Dataset

6/8/2021

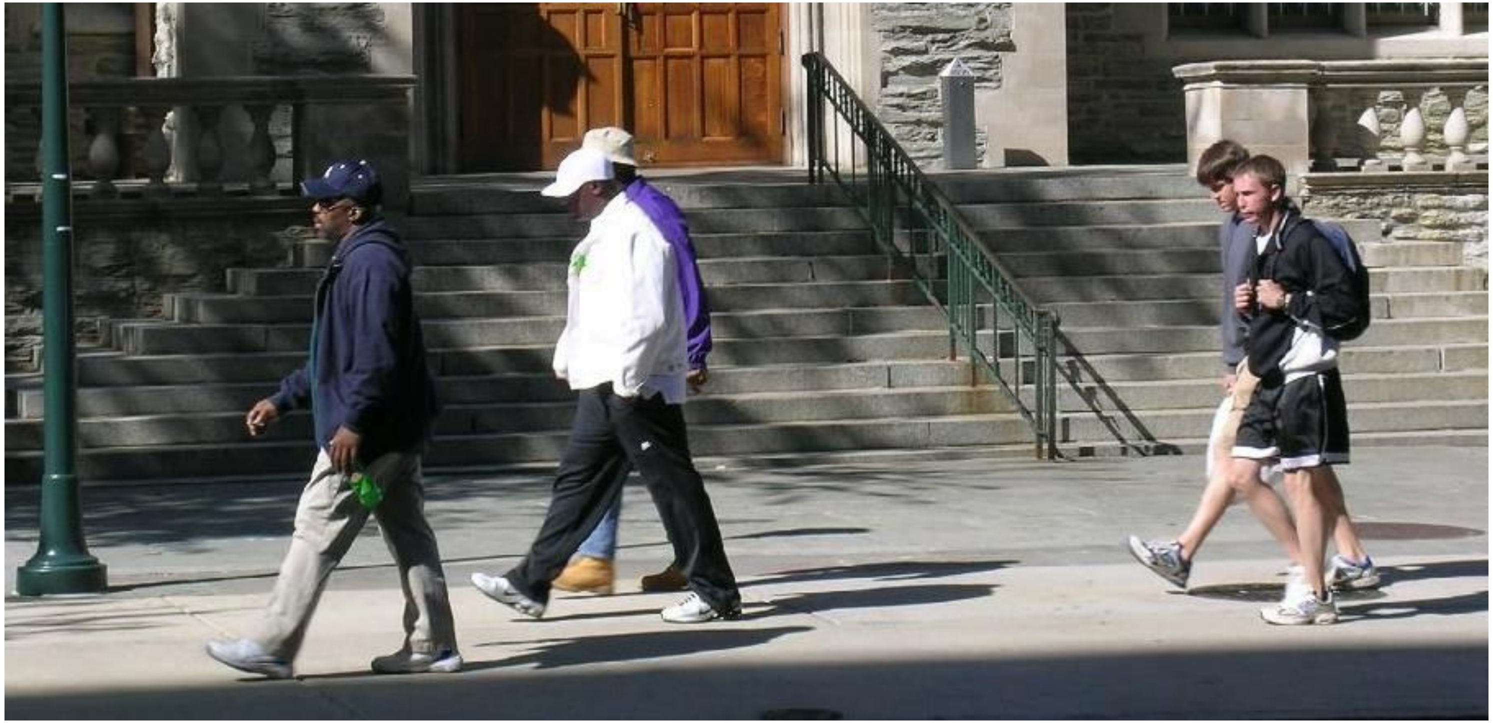
Mask RCNN Expected Input

The network expects training/test data to consist of the following items:

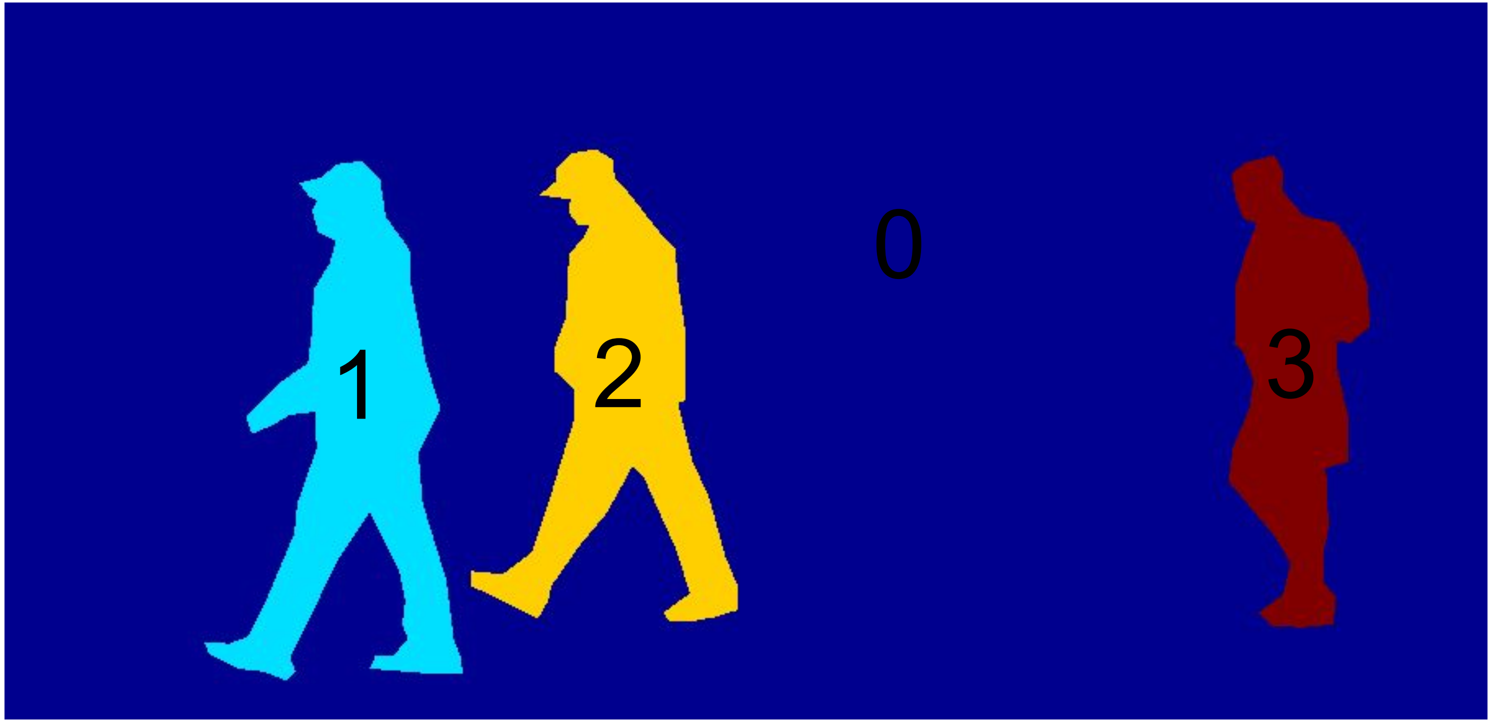
- Images on which to train and test
- For every image in the dataset a corresponding mask image must be provided
    - The mask images are black and white images in which each object in the image is segmented and has its own unique pixel value.
    - We run a python script called CNN_DrawLabelsInImages.py in order to manually create the masks.
    - The dataset class we use in the mask rcnn will automatically create the bounding boxes for each object based on the segmentations, so we don’t need to manually create them.

The mask rcnn dataset will automatically split the images/masks into a training set and a smaller testing set. So we do not need to separate them ourselves.

For instance, if your dataset had this as an input image



The corresponding segmentation image is something like this:



0 - background. All background pixels have intensity 0
1 - The first mask. All pixels in this mask have intensity 1
2 - The second mask. All pixels in this mask have intensity 2
3 - The third mask. All pixels in this mask have intensity 3
In actuality the intensity of the pixels may change based on the label but the general idea is the same.

How to set up the training data

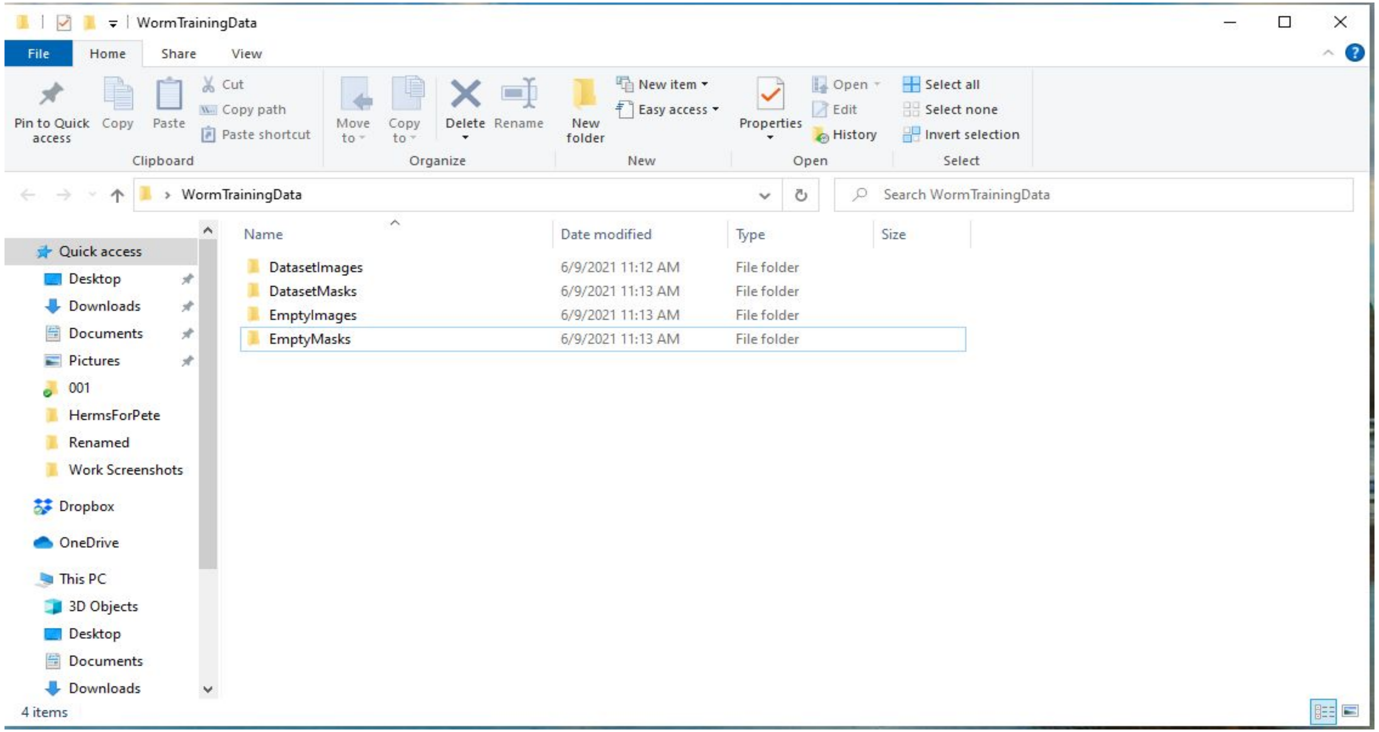
1. Gather good worm images to be used while training
2. Place them all in their own folder named “DatasetImages”
3. Use the CNN_DrawLabelsInImages.py file in the ActivityAnalyzer solution to generate masks for each image you gathered
4. Move all the mask images to their own folder named “DatasetMasks”, which is in the same parent folder as the DatasetImages folder.

Folder hierarchy:

```
ParentFolder/
        DatasetImages/
                Image1.png
                Image2.png
                …
        DatasetMasks/
                Image1_mask.png
                Image2_mask.png
                ...
```

Create the folders for the Dataset Images and Masks

Create a folder for your dataset. Then inside that folder create 4 other folders. Name the folders DatasetImages, DatasetMasks, EmptyImages, and EmptyMasks. Put the training images you gathered inside the DatasetImages folder.



Name	Date modified	Type	Size
DatasetImages	6/9/2021 11:12 AM	File folder	
DatasetMasks	6/9/2021 11:13 AM	File folder	
EmptyImages	6/9/2021 11:13 AM	File folder	
EmptyMasks	6/9/2021 11:13 AM	File folder	

Creating Masks with CNN_DrawLabelsInImages.py

First you'll need python installed, along with the following libraries: OpenCV, Numpy, matplotlib (version 3.2), and natsort.

Once python is installed, although it's not required, I recommend setting up a virtual environment via the command line:

```
python3 -m venv /path/to/new/virtual/environment
```

Activate your new virtual environment with :

```
\path\to\new\virtual\environment\Scripts\activate
```

You'll need to install the necessary dependencies for labeling. Inside the command prompt, navigate to the Activity Analyzer folder where the labeling_requirements.txt file is located (\Autogans\ActivityAnalyzer\ActivityAnalyzer) then run the following command:

```
pip install -r labeling_requirements.txt
```

See https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/ for details on venvs.

Creating Masks with CNN_DrawLabelsInImages.py

Now that your environment is set up you can navigate within the command prompt to the Activity analyzer solution where the CNN_DrawLabelsInImages.py file is located (\Autogans\ActivityAnalyzer\ActivityAnalyzer).

Once there, use the following command to run the file:

```
python CNN_DrawLabelsInImages.py --extension=_mask --mrcnn_format
```

The “--extension=_mask” flag will append _mask to the resulting mask file.
i.e. if the original image is “image1.png” the mask will be “image1_mask.png”. You can make this whatever you would like the extension to be… it doesnt need to be _mask.

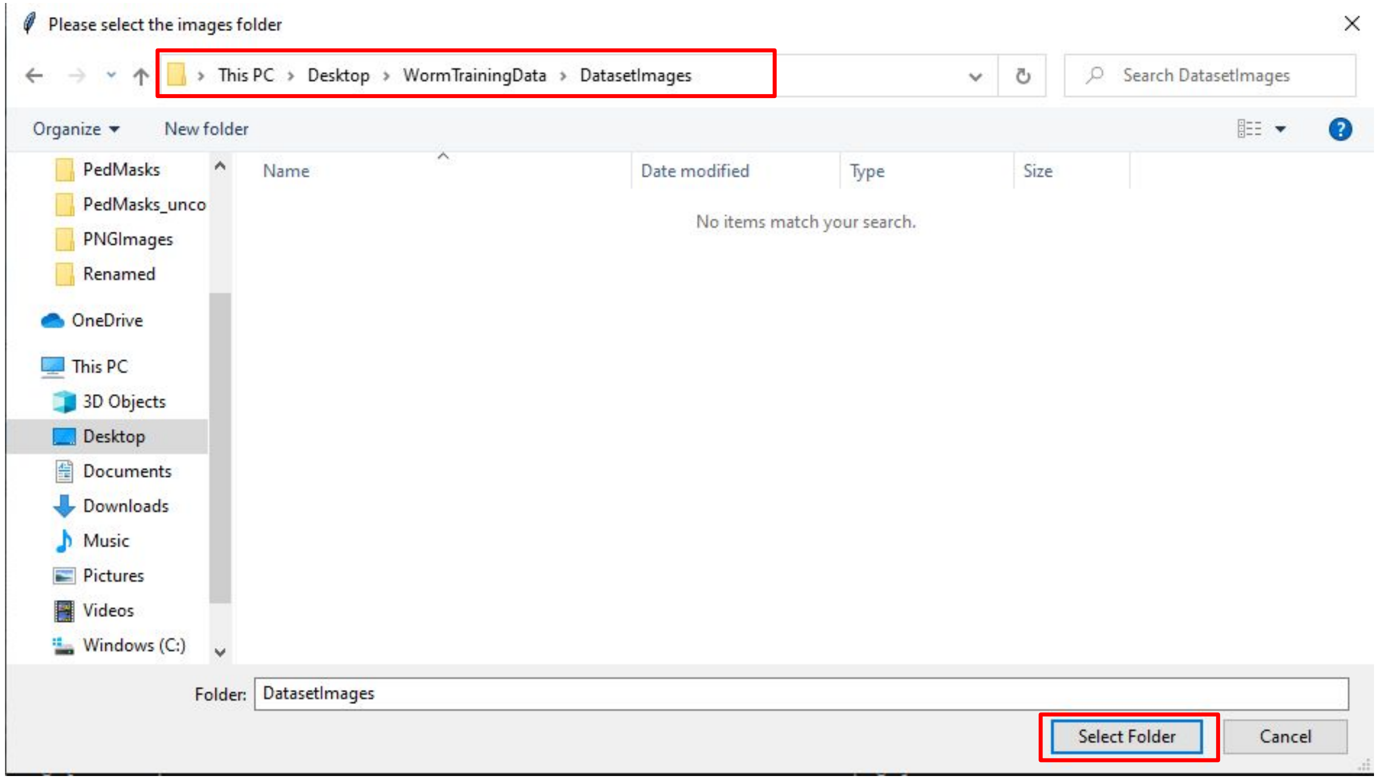
The “--mrcnn_format” flag ensures that the formatting of the resulting mask images is appropriate for our mask_rcnn network as described in the earlier slides.

If you are doing gender labeling of heads and tails you will also add the flag --gender_labels like so:

```
python CNN_DrawLabelsInImages.py --extension=_mask --mrcnn_format --gender_labels
```

Creating Masks with CNN_DrawLabelsInImages.py

Once the program is run, it will prompt you to select the images folder. Navigate to the DatasetImages folder with the worm training images and select it.



No items match your search.

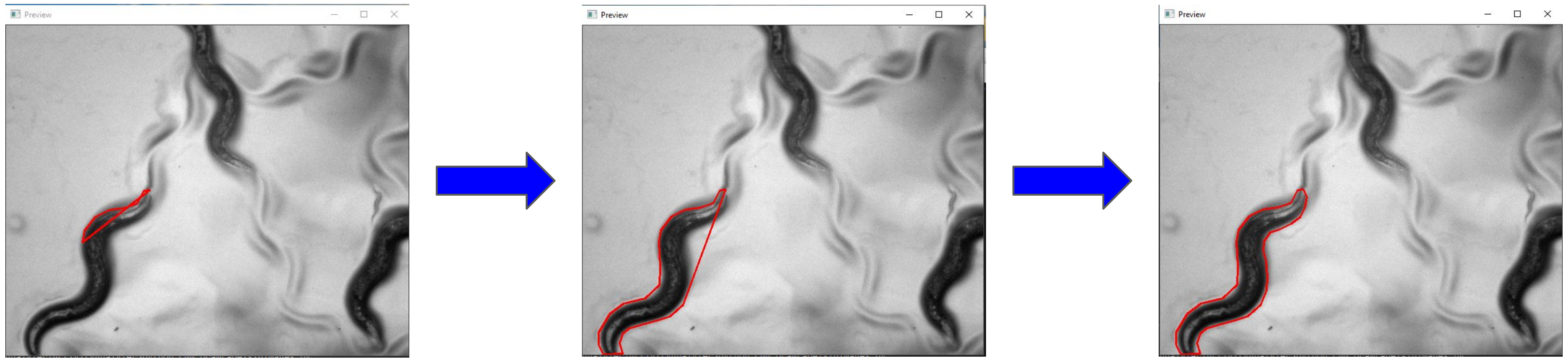
Folder: DatasetImages

Creating Masks with CNN_DrawLabelsInImages.py

The program will now show you each image in the folder, in sequence, for you to mask.

Step 1

Create a mask by carefully clicking around a worm. (If you make a mistake you can remove points by pressing the backspace key.)

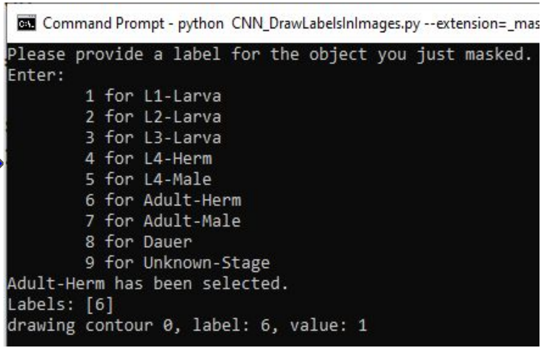


Creating Masks with CNN_DrawLabelsInImages.py

The program will now show you each image in the folder, in sequence, for you to mask.

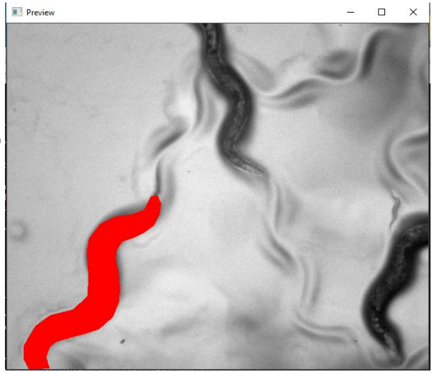
Step 2

Press spacebar when the mask is complete. Then give your mask a label by pressing the number corresponding to the desired label from the list in the command prompt.


Command Prompt - python CNN_DrawLabelsInImages.py --extension=_mas

```
Please provide a label for the object you just masked.
Enter:
        1 for L1-Larva
        2 for L2-Larva
        3 for L3-Larva
        4 for L4-Herm
        5 for L4-Male
        6 for Adult-Herm
        7 for Adult-Male
        8 for Dauer
        9 for Unknown-Stage
Adult-Herm has been selected.
Labels: [6]
drawing contour 0, label: 6, value: 1
```

6 was pressed.

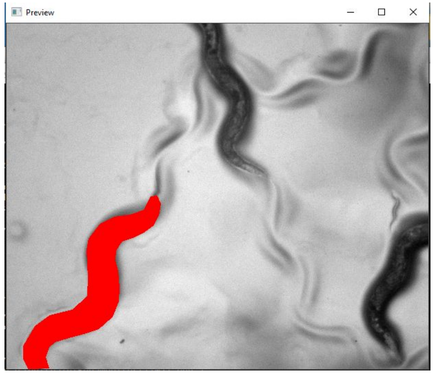


Note: If there are less than 10 label options in the command prompt list then you can simply press the desired label number, however, if there are 10 or more label options then you must enter the label numbers into the command prompt.

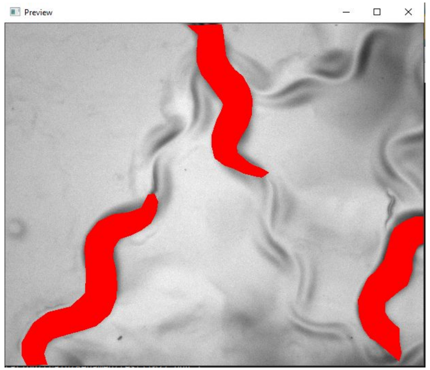
Creating Masks with CNN_DrawLabelsInImages.py

Step 3

Now you have a worm masked and labeled.



Repeat steps 1 and 2 for each worm in the image



Press escape once all worms are masked to finish the image and move to the next image

If you make a mistake with your mask or label, you can press x to delete masks you have completed, starting with the most recent mask.

Step 4

Repeat steps 1 through 3 for each image in the dataset.

Note: If you finalize an image in which you have made a mistake then you can stop the script by selecting the command prompt and pressing Ctrl+C. Next, go to the DatasetImages folder and delete any bad mask images. Then you can rerun the script and re-label the image. Note that the script will automatically skip any images that have a corresponding mask (as per the extension given by the --extension flag) so you won’t need to re-label images you have already completed.

Move Masks to DatasetMasks

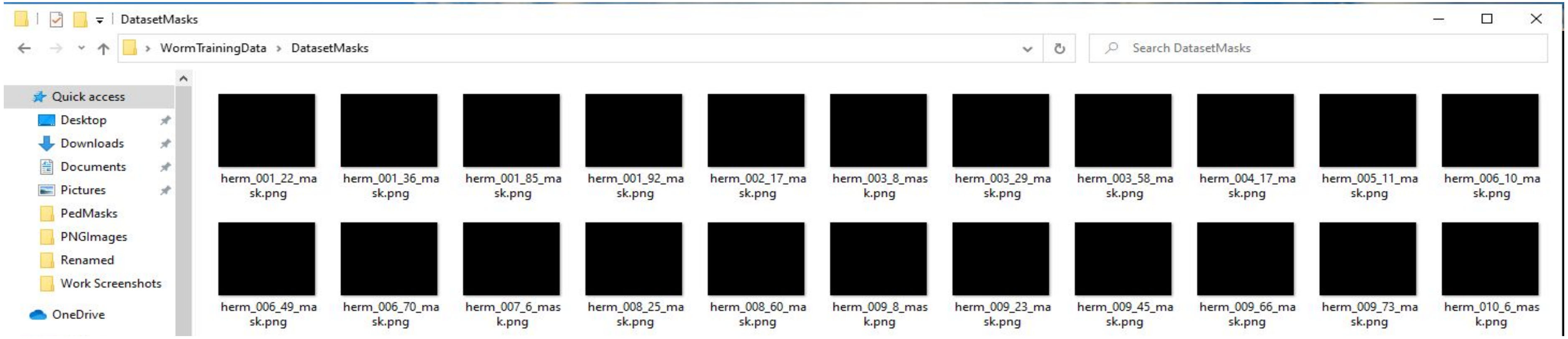
The labelling program will save all the mask images into the same folder as the original images, so the last thing you must do is move the masks from the DatasetImages folder to the DatasetMasks folder.

To do this run the RCNN_FormatLabeledDataset.py file located in the Activity Analyzer folder of the Autogans repository. (\Autogans\ActivityAnalyzer\ActivityAnalyzer).

For the first dialog box, select the folder that contains all the images and masks together.

For the second dialog box, select the folder to move the images and masks to. This should be the same folder you created in slide 6, and must contain DatasetImages, DatasetMasks, EmptyImages, and EmptyMasks folders.

The script will then sort all the images into their appropriate folders for you.



Thats it!

The dataset is now ready to be trained on the Mask RCNN network.