Learning Based Digital Matting with Ice Data

To Start

**Downloads**

* Download the learning-based-digital matting and Mask RCNN folders

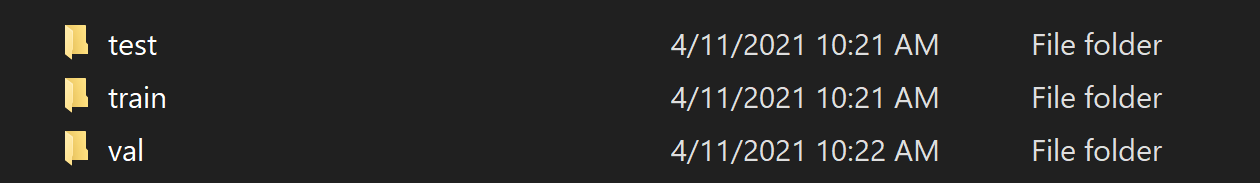
Creating Trimaps

**Introduction**

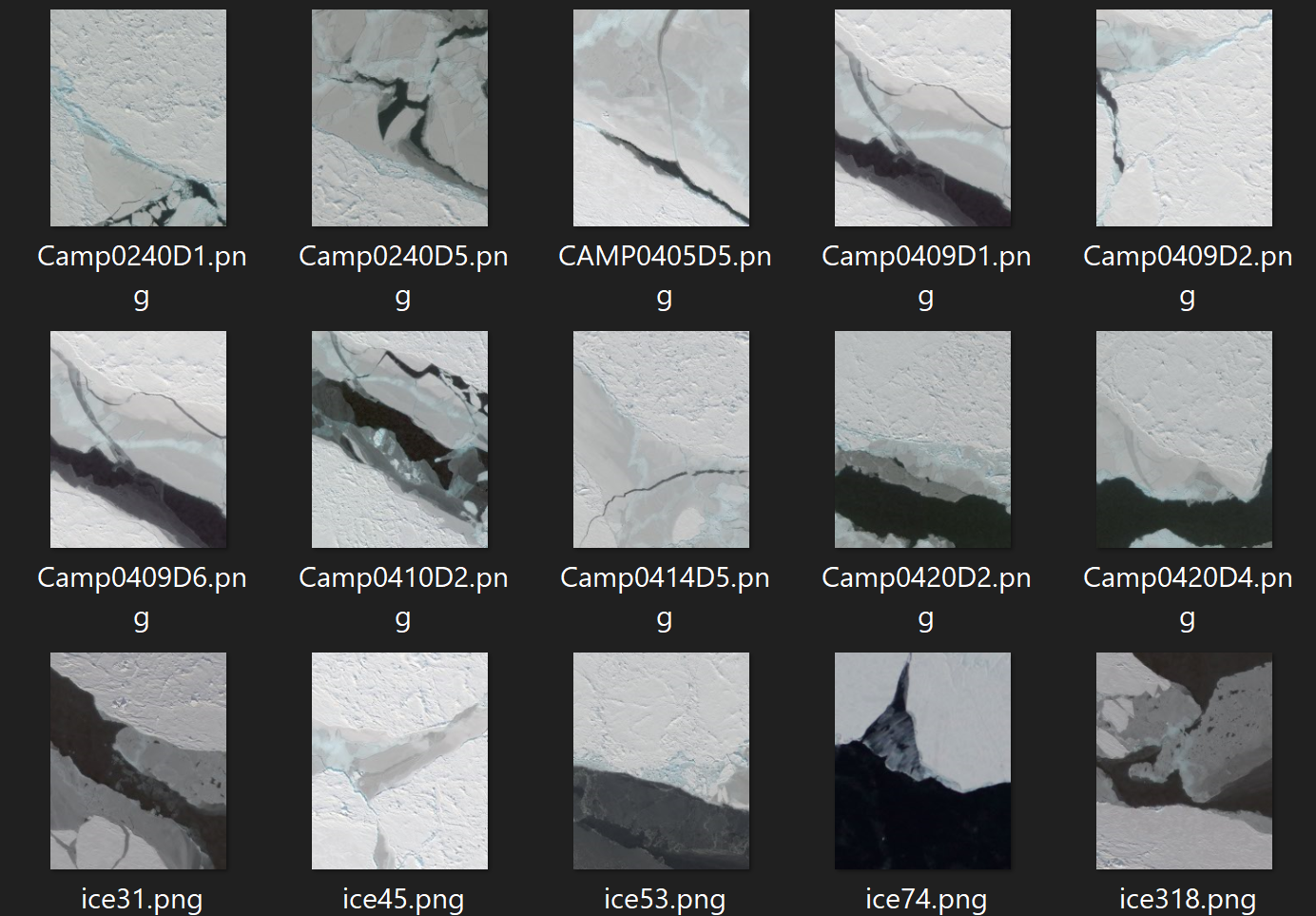
* The first step to using learning based digital matting is to create trimaps for the images in your data
* A trimap is an image segmented into three parts
  + White (object of interest/what we want the machine to learn how to detect)
  + Black (background/not involved for detection)
  + Gray (Areas such as outline and complicate shapes or patterns that learning based digital matting will determine as white or black)

**Creation**

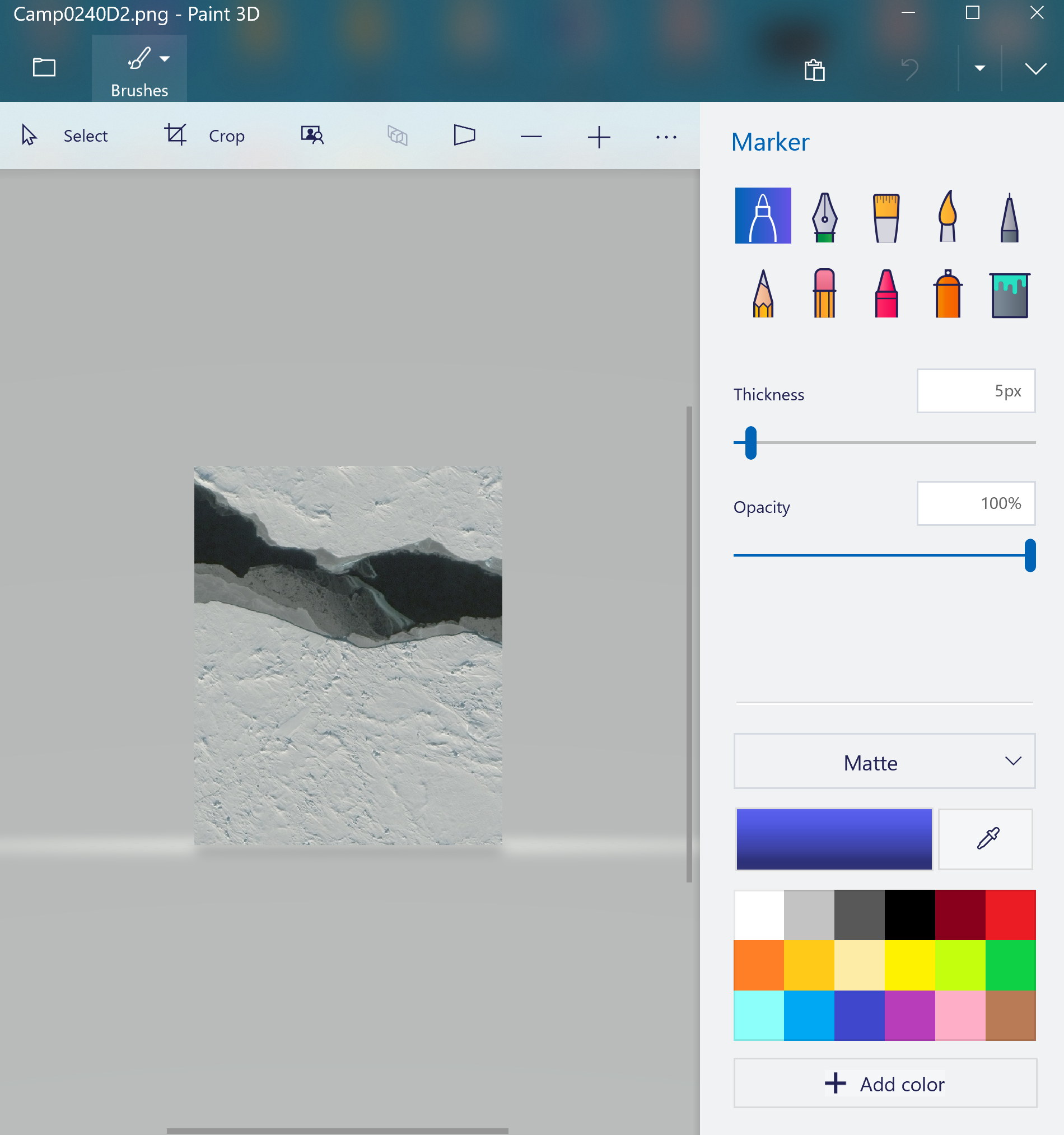
* Create three folders called test, train and val



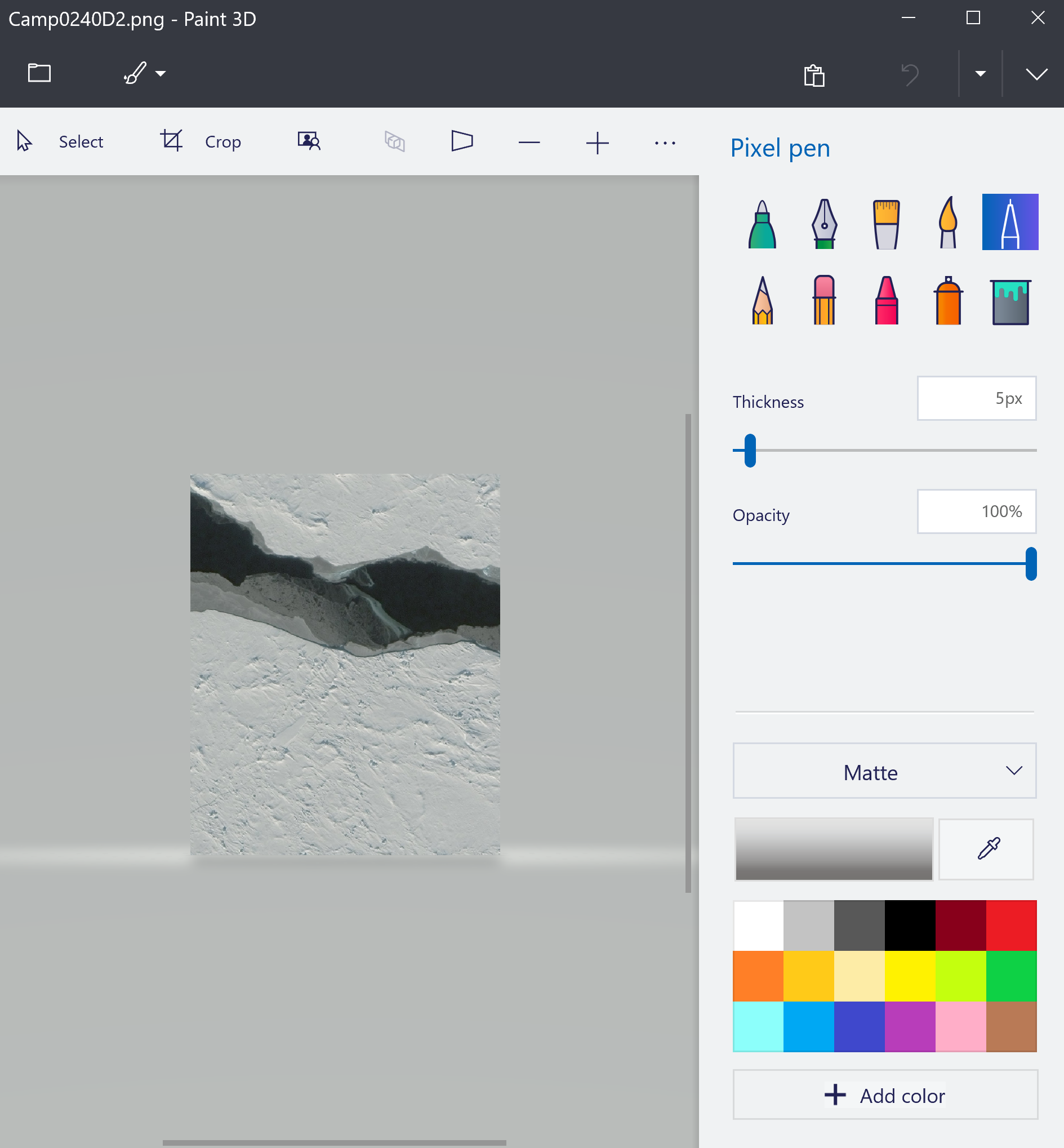
* Split the dataset into those three sections and place the images within the proper folder



* For each image within a folder, open up paint3D or any photo editing software

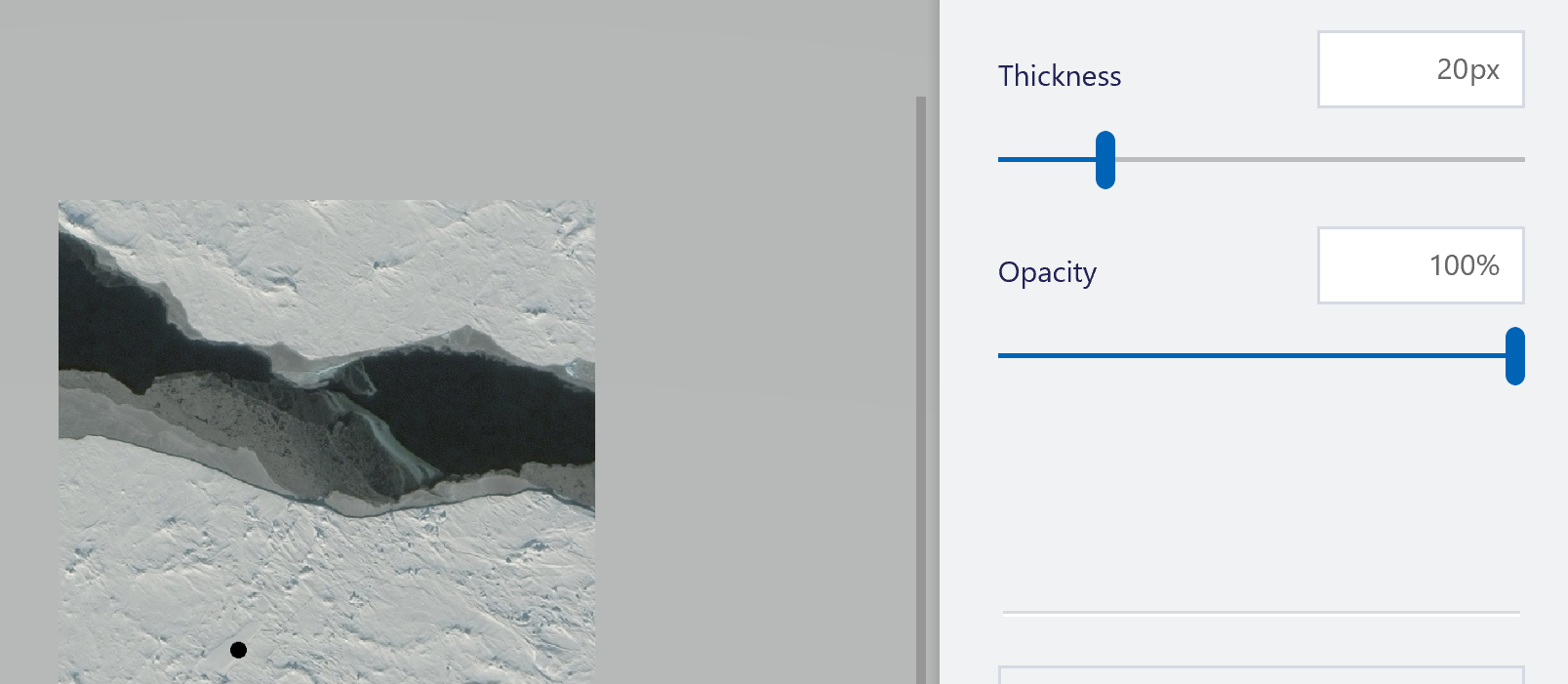


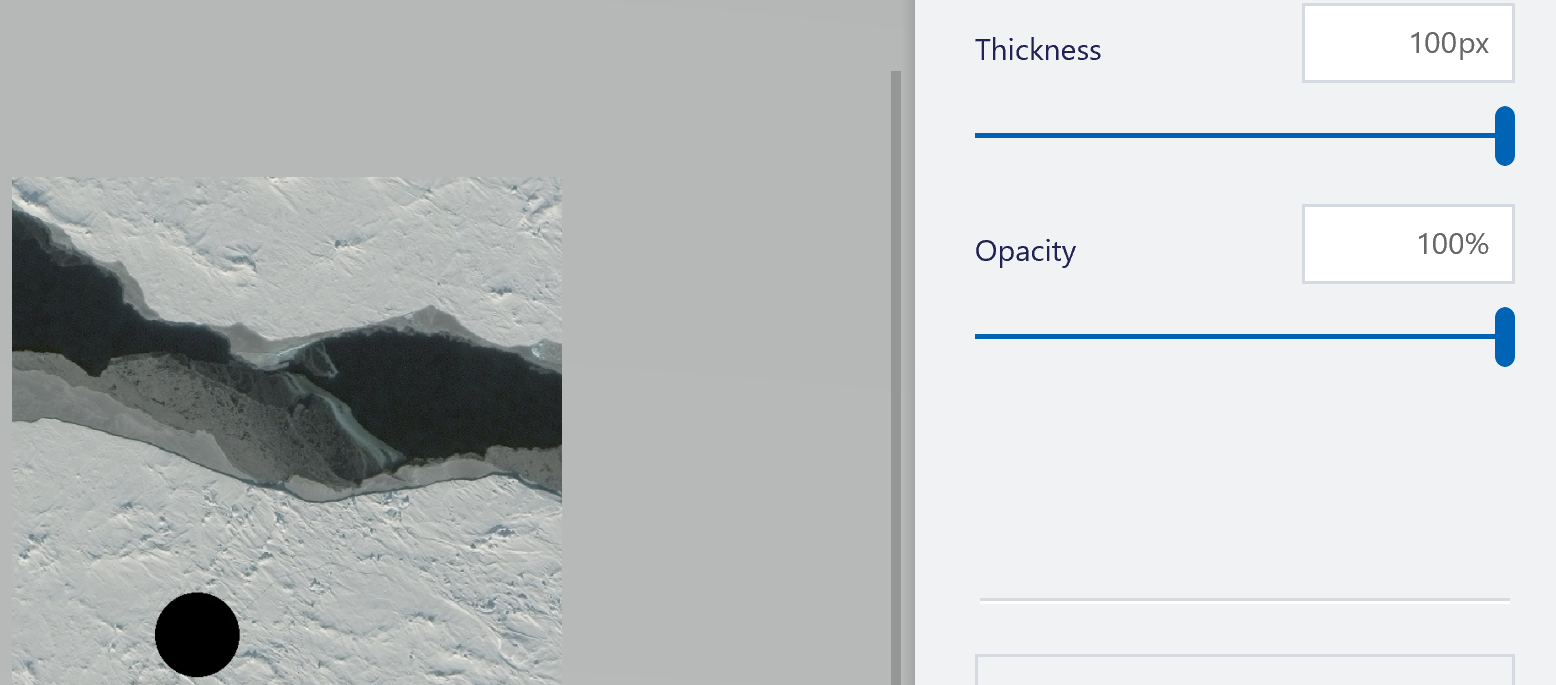
* Select the pixel pen tool and a color to start marking the image



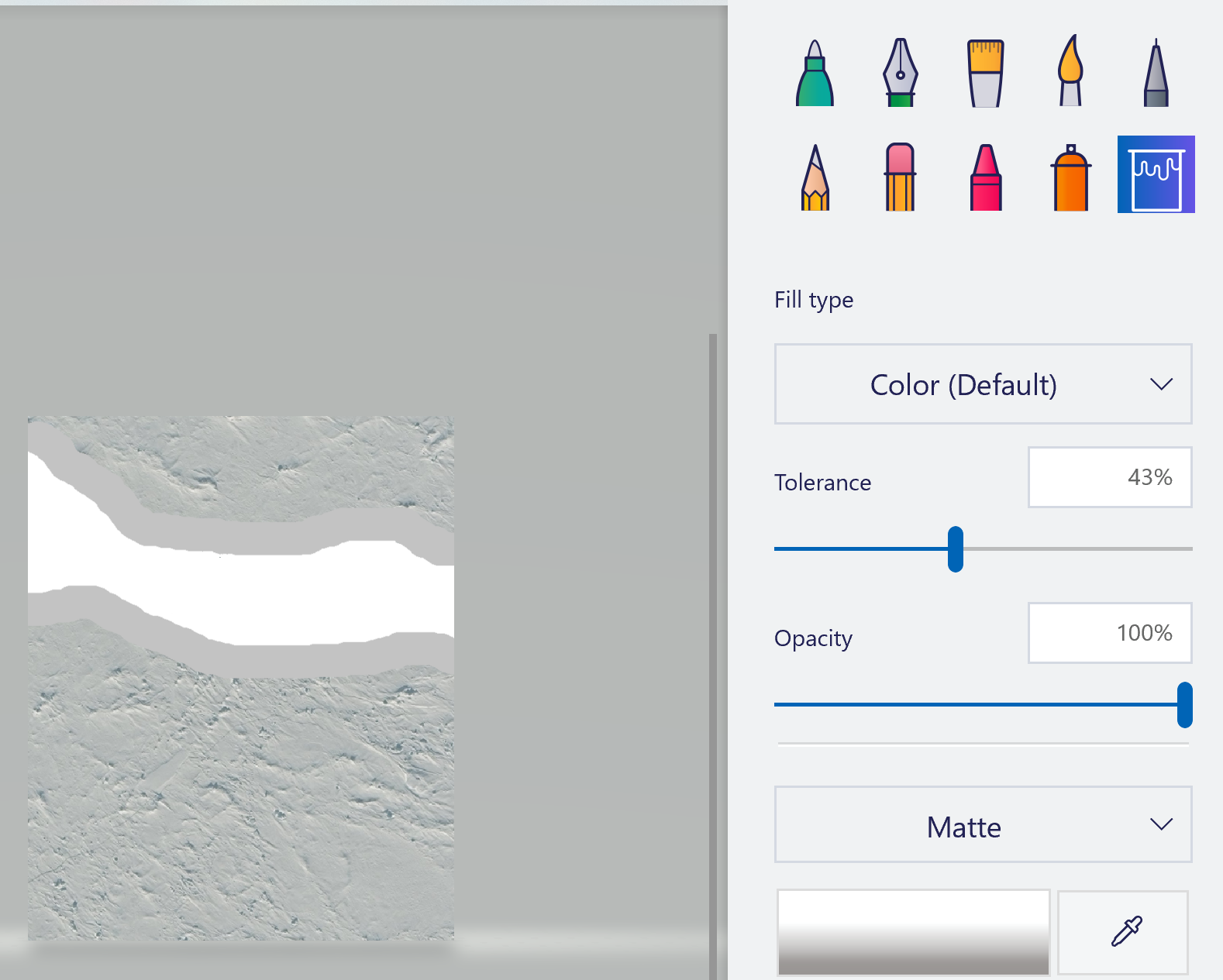
**Helpful Marking Tips**

* Moving the Thickness bar changes the size of your cursor (Higher number means move area cover when moving cursor across image)

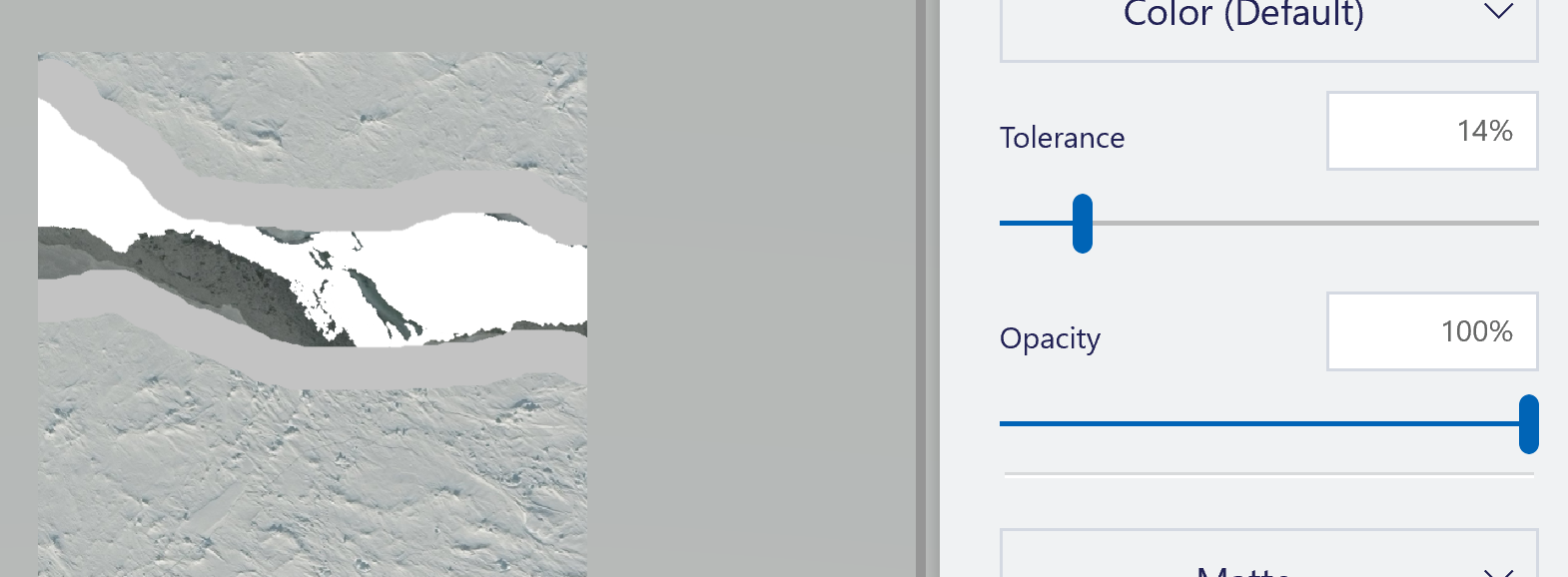


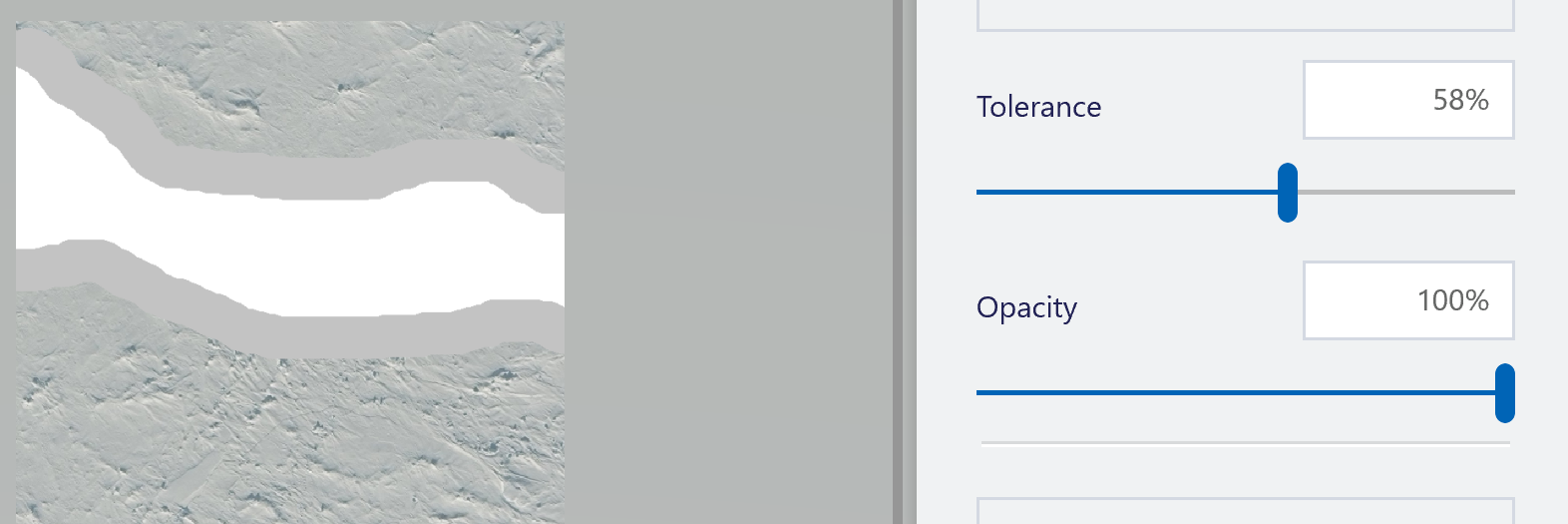


* Easier to start with gray to create an outline around the object of interest than use the fill tool to fill the inside of the gray outline in white and the rest black



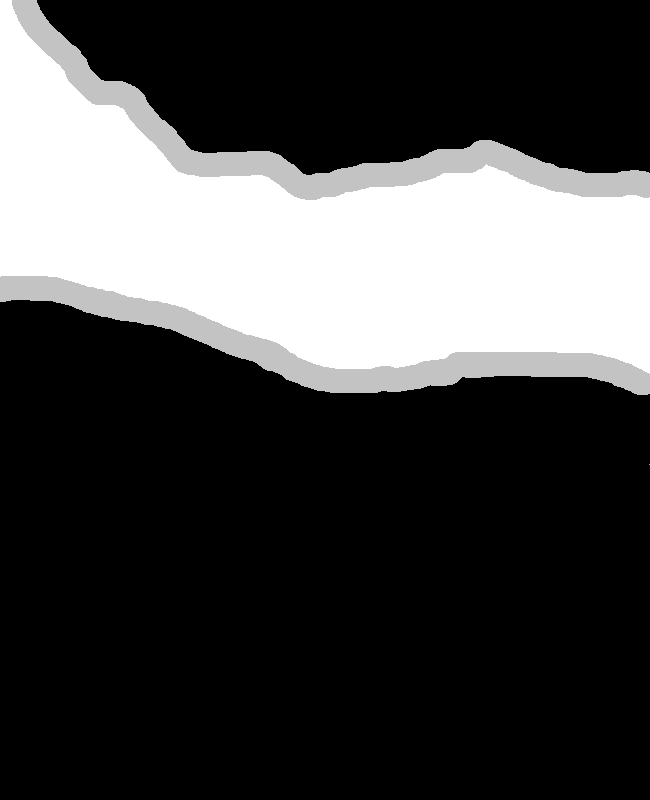
* If the fill tool doesn’t color in properly try shifting around the tolerance bar





**Finishing Trimaps**

* The trimap is considered finished when the entire image is segmented into black, white and gray



* Once the trimap is finished, save the image as a different file name then the original so you don’t override the original image. Save it to the same folder.
* If a trimap has multiple objects within it, load it back into paint3D. Save each object in a trimap as its own trimap image by coloing all objects except one in black. Save the image as a new trimap image. Undo each object being black, pick a new object that wasn’t white before and color everything else in black. Save this new object as a trimap image, undo and repeat the process until all objects have a new trimap image. Afterwards delete the original trimap image



Left image gets separated into two new trimap image (middle, right)

* Repeat this entire process until the test, training, and validation folder have image along with its respective trimaps

Using Learning Based Digital Matting

**Download/Installation**

* Make sure the learning-based-digital matting folder is downloaded
* Create a new python virtual environmental with Python version 3.6.7 along with the packages listed on the file “learnMatting pip installs.txt”

**Before Running Learning Based Digital Matting**

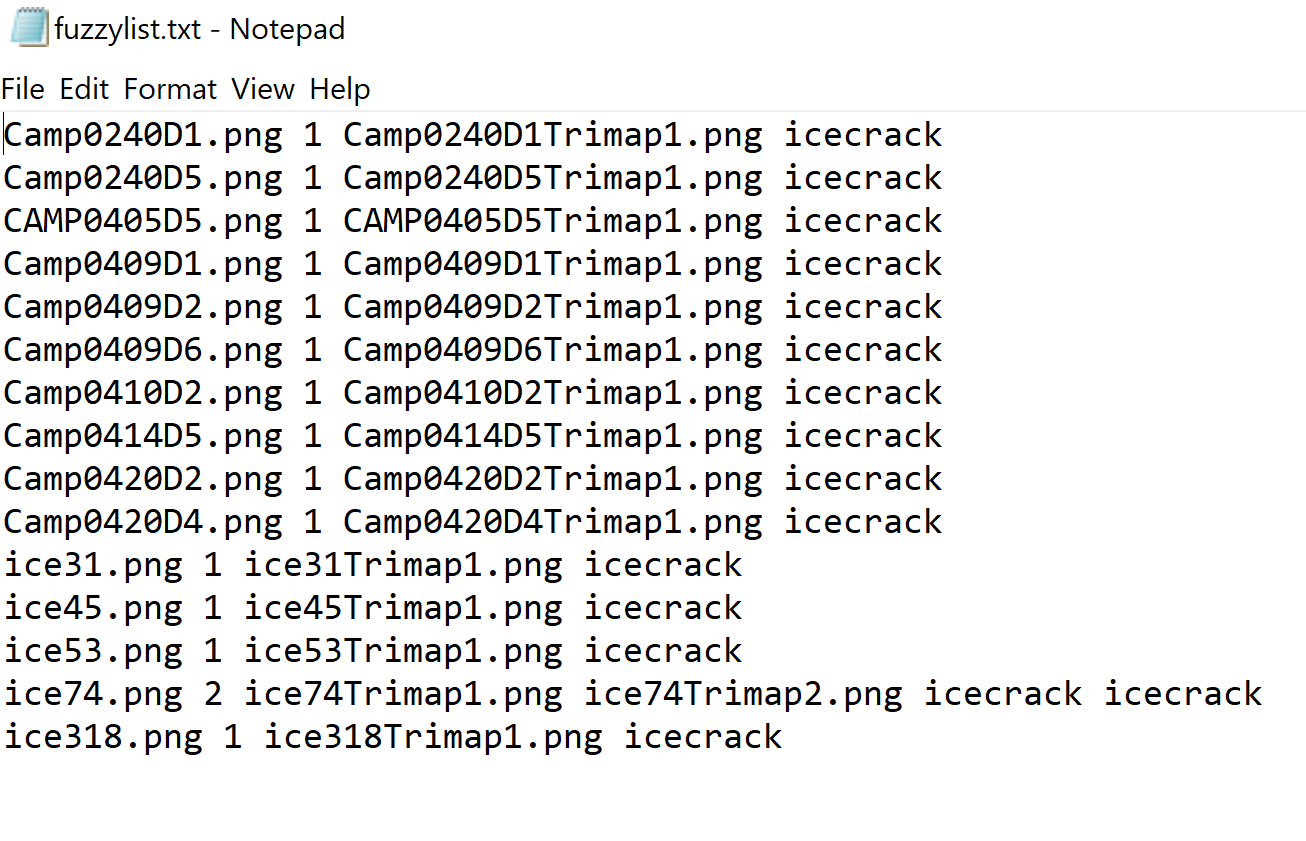
Moving Images

* Select one folder (test, training or val) and copy all the images (original and trimap) within that folder into the learning based digital matting folder

Creating Text File

* Create a new text file called “fuzzylist.txt” and place it within the learning based digital matting folder
* The format for each line of the text file will be:

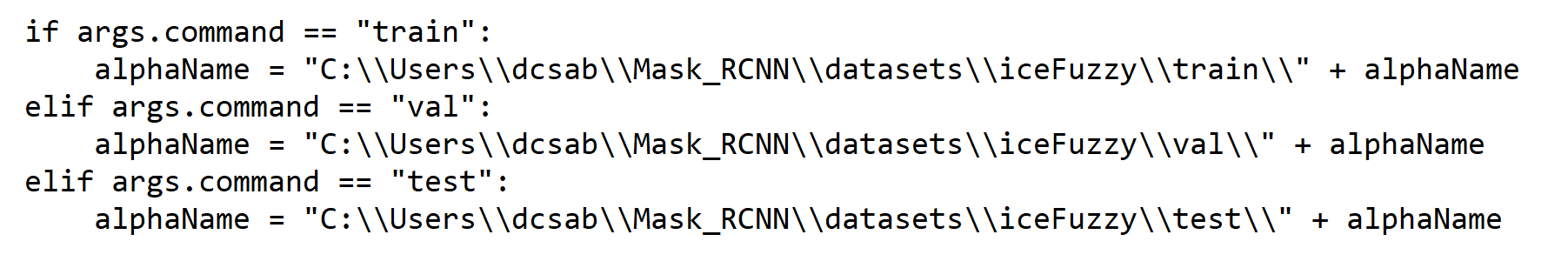
[original image filename] {# of trimaps} [trimap filename] {class of trimap}



* The number of lines within the text file will match the number of original images you had within the (test, train, or val) folder

Editing Python File

* Open up the “learning\_based\_matting\_json.py” file
* In def main() located the three if args.command == \_\_ statements



* Change the filename path to match where your Mask RCNN is located
  + Only need to change “ C:\\Users\\dcsab” to be your file path

**Running Learning Based Digital Matting**

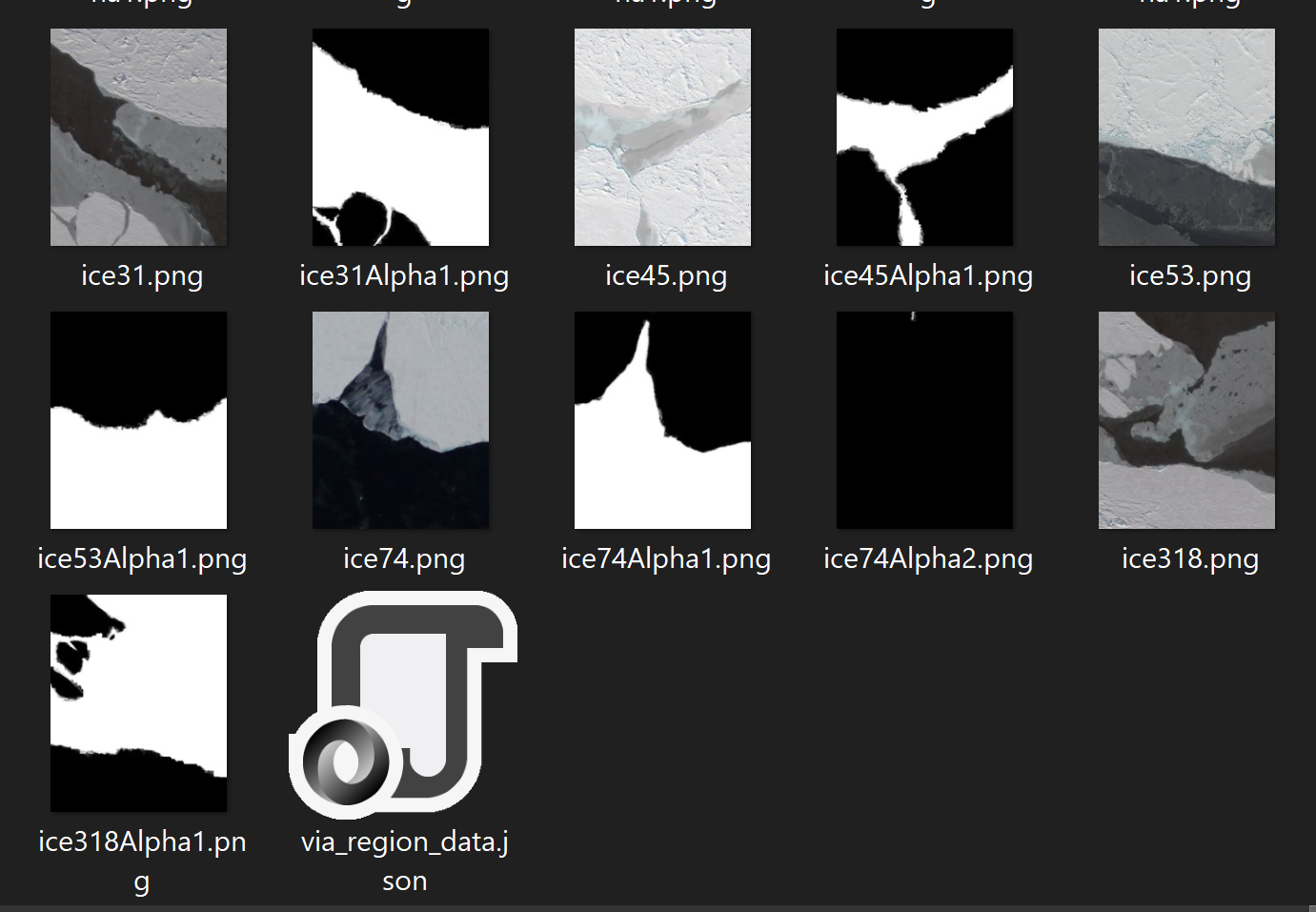
Running Code

* Once the images + trimaps + textfile are in place, call the command:

python learning\_based\_matting\_json.py [folder images came from]

* + For example: python learning\_based\_matting\_json.py train
  + if you start with putting all the test images then switch train to test. Same goes with val
* The learning based digital matting code will go through each line in the textfile, create the detailed binary images for each trimap provided, and save those images within the learning-based-digital matting folder
* A json file is also created containing information from the textfile for each image. The json file will be saved within the learning-based-digital-matting folder

Create folder to Store Image + Json

* Create a new folder outside of the learning based digital matting folder and call it “iceFuzzy”. Within this folder create three more folders called test train and val
* Take the all the original images, the binary images and json file and place them into the corresponding folder.
  + For example if you start with the test folder. Take the original image, newly generated binary images and json and place it within the iceFuzzy/test folder
* 

Repeating the Process

* Go back to the learning-based-digital matting folder, remove any old images and repeat steps “Before Running Learning Based Digital Matting “ and “Running Learning Based Digital Matting” until all folders are completed (test, train, and val)

Using Mask RCNN

**Download/Installation**

* Make sure the Mask RCNN folder is downloaded
* Create a new python virtual environmental with Python version 3.6.7 with CUDA 10.0.130 with CUDNN 7.4.2.0 along with the required packages
  + Pip
  + numpy=1.19
  + scipy=1.5.2
  + pillow=7.2.0
  + cython=0.29.21
  + matplotlib=3.3.2
  + scikit-image=0.17.2
  + tensorflow-gpu=1.14
  + keras=2.2.4
  + opencv-python=4.4.0.44
  + h5py=2.10.0
  + imgaug=0.4.0
  + IPython[all]=7.16.1
* In addition to these packages you need to install pycocotools
  + pycocotools requires Visual C++ 2015 Build Tools
  + clone repository: git clone <https://github.com/philferriere/cocoapi.git>
  + clone repository: git clone <https://github.com/philferriere/cocoapi.git>

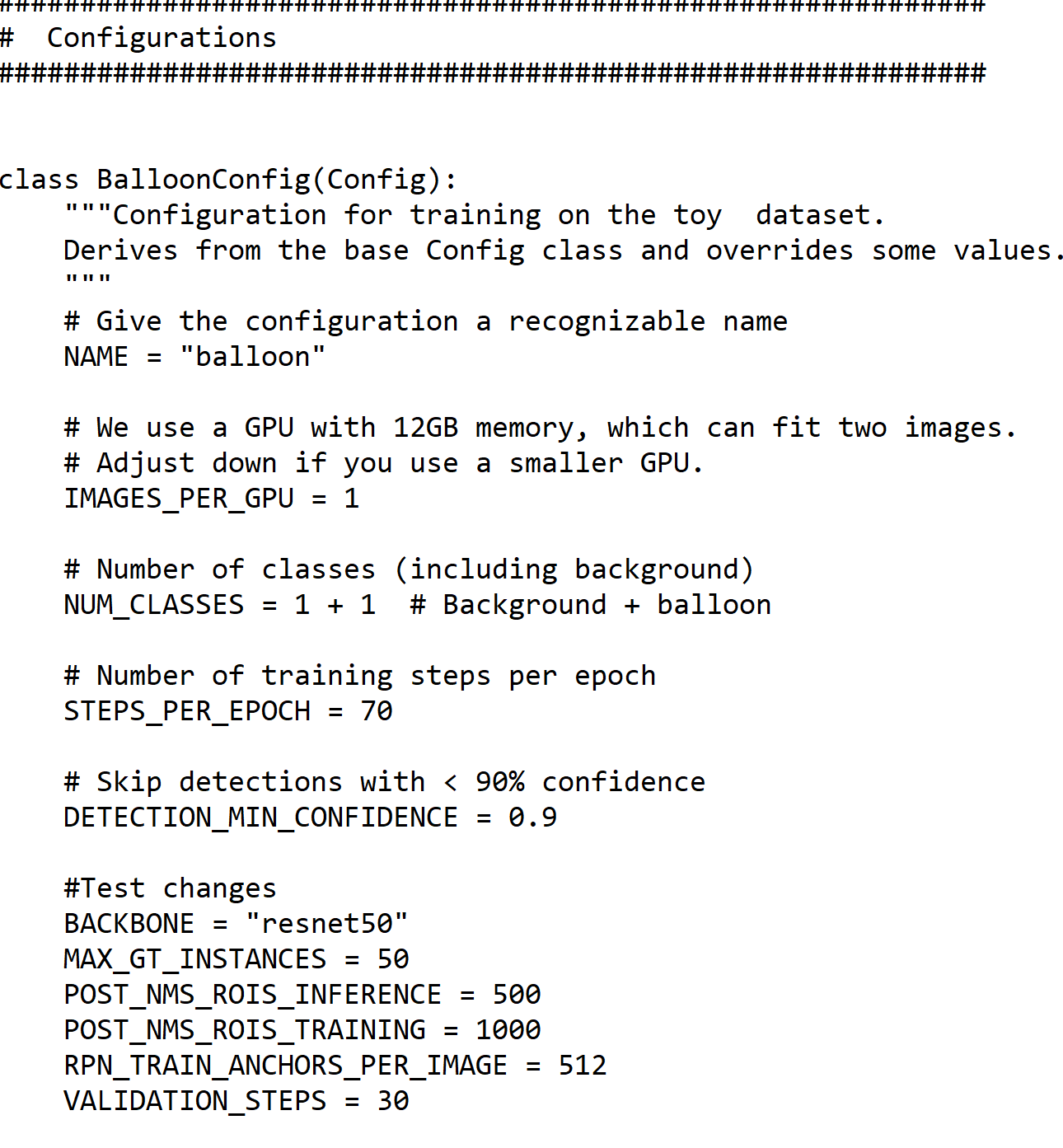
**Before Running Mask RCNN**

Uploading the dataset

* Take the iceFuzzy folder created from performing learning based digital matting and place it within the “datasets” folder

Editing the Code

* Open the “samples” folder then open the “balloon” folder then open the balloon\_Fuzzy.py file
* In class BalloonConfig(Config) change the STEPS\_PER\_EPOCH to match the number of original images in the training folder
  + Example My ice dataset has 70 images, so the STEPS\_PER\_EPOCH will be 70
* If the system you are using has a 12GB GPU or higher you can delete everything under #Test changes. Else check those configuartions or else the progam will most liekly crash



* In def train(model) if you would like the Mask RCNN model to train more, in model.train() change epochs = # to be what you feel is appropriate

**Running Mask RCNN**

* In the Mask-RCNN/Samples/Balloon directory, call the command:

python balloon\_Fuzzy.py train --dataset=”/path/to/balloon/dataset” --weights=coco

* Example:

python balloon\_Fuzzy.py train --dataset=”C:\Users\dcsab\Mask\_RCNN\datasets\iceFuzzy” --weights=coco

* The model will run until it is finished training. The runtime could take 30-45 minutes to multiple hours depending on how images are used and the number of epochs and steps per epoch.
* Once the model has finished training a folder containing the weights to the trained model will be located in the logs folder

**Evaluation and Visualized Results**

Setting up the Model

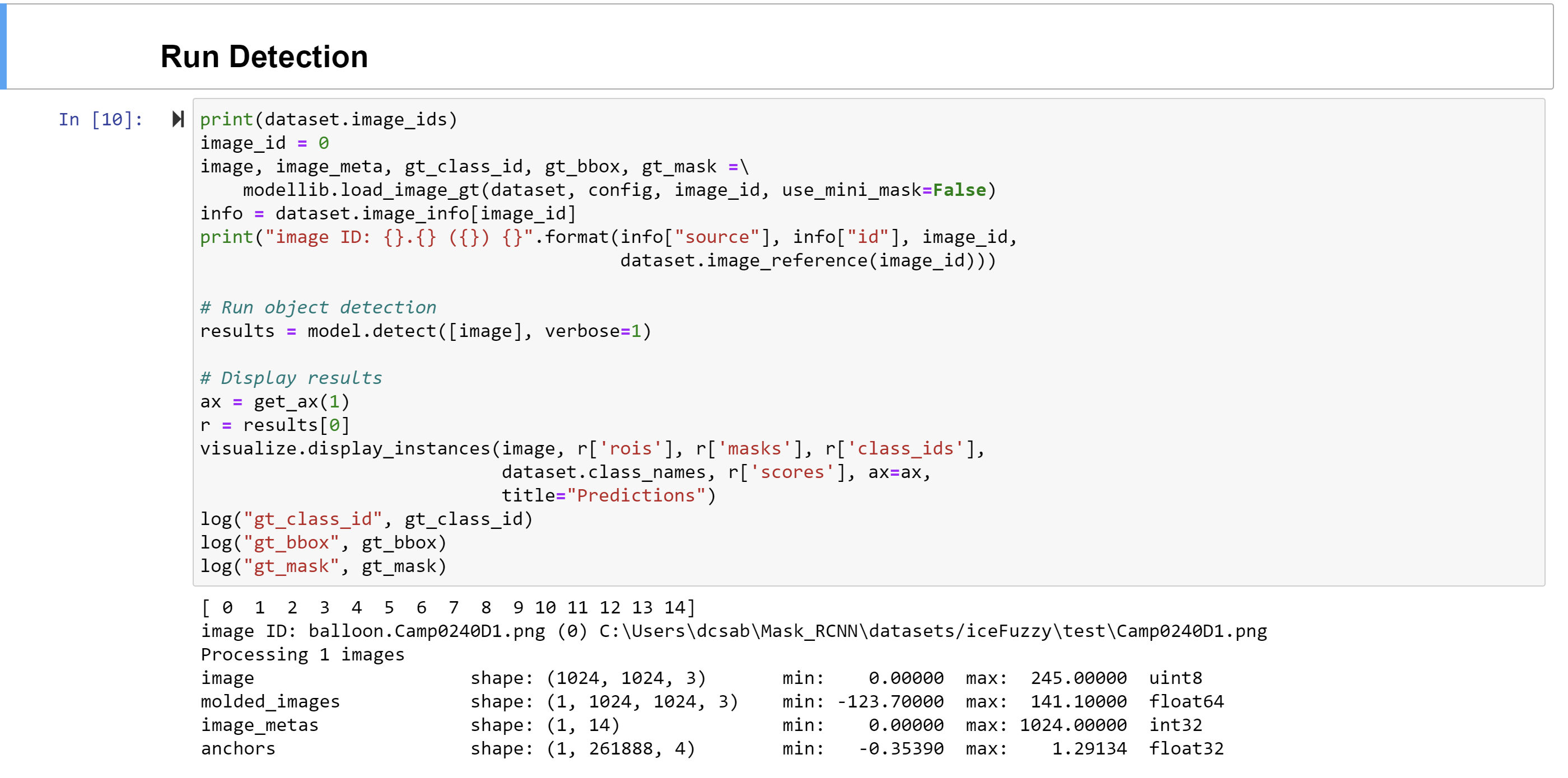
* In the Mask\_RCNN directory open up jupyter notebook
* Go to samples -> balloon -> inspect\_balloonFuzzy.ipynb
* In first coding block, change BALLOON\_WEIGHTS\_PATH to be the path of the newest weight folder in the logs folder along with the last .h5 file in the folder

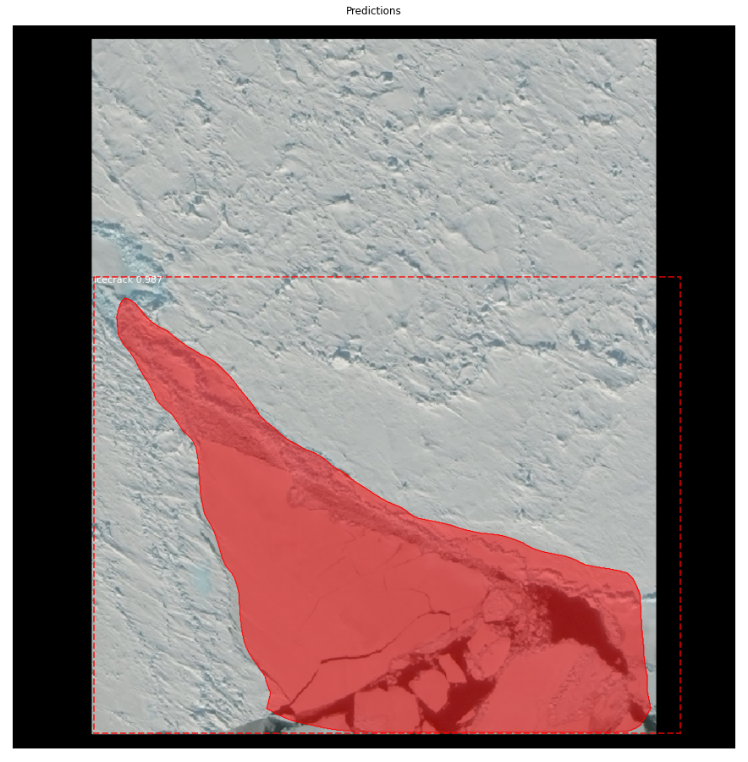


* Run all the coding blocks until Run Detection. These coding blocks set up the model to run detection

Running Evaluation

* Running the first block under “Run detection” show the result of the models detection on an image form the test set. You can switch the test image by changing image\_id = #





* Running the second code block will run detection on all test images and return the accuracy or mAP score
* 

Running Detection on any image

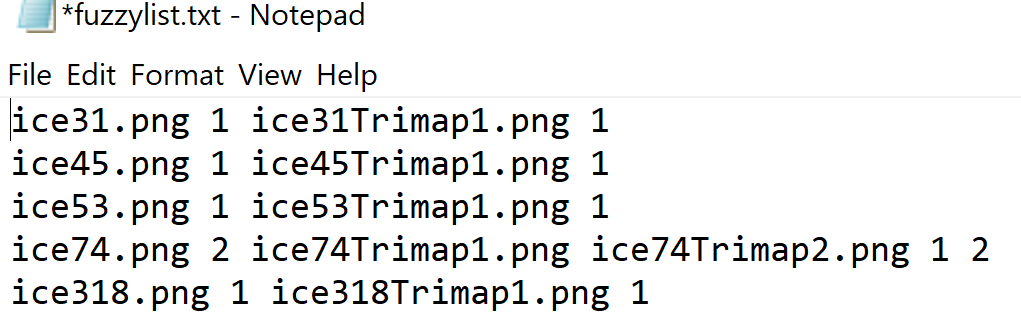
* To run detection on any image place your image into the “images” folder
* In the first code block under “Detecting Non-Test Set Images” change filename to match the file name of the image you wish to detect



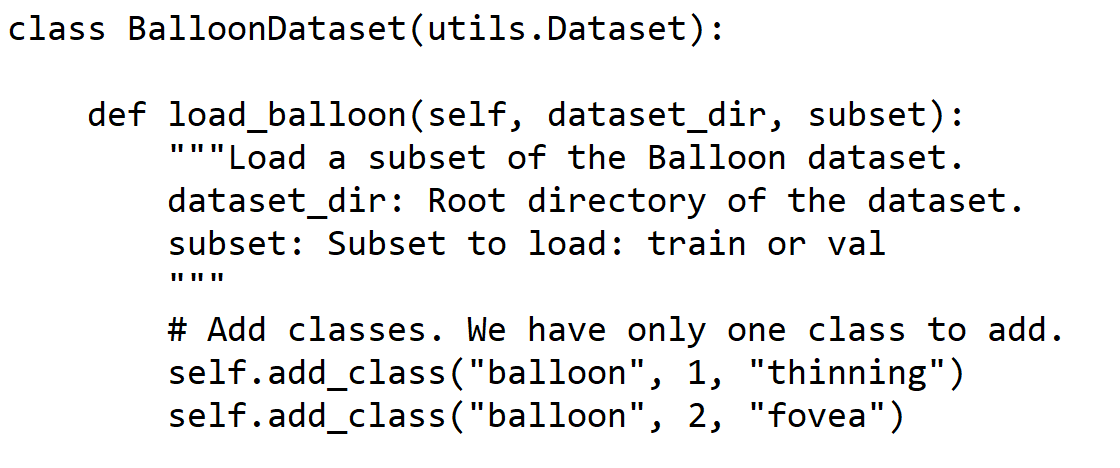
* Run the code block

Running Data with Multiple Classes

**Changes to learning Based Digital Matting**

* When creating the text file instead of writing the class name at the end of each line use a number starting with 1 up to the number of class you have
  + For example, if the you want to detect icecracks and seals write 1 (for icecracks) and 2 (for seals)
* 
* Run everything else as you would normally

**Changes to Mask RCNN**

* In the balloon\_FuzzyMulti.py file change the NUM\_CLASSES in the class BalloonConfig(Config): to be 1 + the total number of classes
* 
* Next in class BalloonDataset(utils.Dataset) add classes using the format below
* 
* Train the model by running the command with the exception of changing balloon\_Fuzzy.py to balloon\_FuzzyMulti.py

**Evaluation and Visualize Result**

* Open the inspect\_balloon\_model\_FuzzyMulti instead and follow the steps in the Mask RCNN section above