

Bizzy Moore
Professor Bridgeman
Professor Dumitriu
CS Independent Study
Spring 2020

Harmful Algal Bloom Classifier Program - User Manual

Overview

There are three main tasks that are involved in the project for classifying Harmful Algal Blooms in the Finger Lakes through machine learning. The components include acquiring images, often through extracting frames from videos, training the machine learning model, and classifying the images using the trained model. Therefore, this project is broken up into three separate programs: the VLC media player, the training program, and the classifier program.

The Harmful Algal Bloom Classifier is a program that is designed to take an image or a set of images and determine whether each image is clear, turbid, or contains blue-green algae (bga), which signifies that a Harmful Algal Bloom (HAB) was present. Upon the completion of running the program, all of the imported images are sorted into a folder according to the classification that they received: bga, clear, or turbid. In addition to sorting the images, the program creates a report that is outputted in a text file. The report supplies information about how the program classified each image. The report sorts the images from the least confident to the most confident in the program's prediction of the images. The text file can be imported into Excel for easy access.

Necessary Software

In order to use this program, you need to have two main pieces of software installed on your computer: Python and Excel.

Python is the language that is used to write the machine learning program that classifies the images. This software was selected because it has many open-source libraries for machine learning. Therefore, it was able to easily support what we wanted to do for this project. If Python is not already installed on your computer, then it can be downloaded here: <https://www.python.org/downloads/>.

Excel is a program that can be used to load in and view the text file that was created by the classifier program. Excel is recommended because of all of the tools that it offers. However, if you are more familiar with another spreadsheet program that also has the ability to import a text file, then that program can be used as well.

Acquiring Data

The HABs classifier program takes in images and sorts them according to their specific characteristics. Therefore, in order for this program to properly run, it needs to be given a group of images that are saved in the jpg format.

If you need to obtain your images by extracting them from videos and need some guidance, then see the separate document, “Extracting Images Using VLC - User Manual.”

Setup

Summary:

- You need a directory containing:
 - the classifier program file (*HABs_Classify.py*)
 - the classifier model file (*HABs_CNN_Model_FINAL.h5*)
 - a folder for the images that are to be classified, which contains 3 subfolders
 - a folder for the classified images, which contains 3 subfolders named the following:
 - *bgaClassified*
 - *clearClassified*
 - *turbidClassified*
- Advanced Option: you can specify a name for the classifier model file instead of using the default name (*HABs_CNN_Model_FINAL.h5*)

Detailed Explanation (How/Why/Specific Examples):

In order to make sure that the program runs correctly, you need to make sure that everything is set up and organized in the right way. This means that you need to confirm that all of the files that you will be using to run the program are under the same base folder in your directory. If your directory is not set up properly, then the program will not be able to find everything that it needs when it is running, and it will crash with errors.

The first step is to create a new folder somewhere in your directories on your computer that will act as the “base” folder for the project. This folder can be placed in your documents folder, on your desktop, or in any other folder that you would like. It does not matter where the folder is stored. You just need to make sure that you know the directory path of the folder because this path will be used for us to find the folder and access the programs inside of it when we run the program. For the purpose of this explanation I am going to refer to this base folder as *HABsProject*. You can name this base folder anything you would like. If I placed *HABsProject* in my Documents folder on my computer, then the directory path for *HABsProject* on a Mac would be `/Users/BizzyMoore/Documents/HABsProject`. For a Windows user this would be `C:\Users\BizzyMoore\Documents\HABsProject`.

After the base folder is set up for the project, the next step is to make sure that all of the necessary files are in this folder. As stated above, these files are the program file, *HABs_Classify.py*, and the classifier model file, *HABs_CNN_Model_FINAL.h5*. If you choose to pursue the advanced option, and specify a different name for the classifier model file, then make sure that file is included. When you run the classifier program, be sure to let it know about the different name for the classifier model file. This can

be done with the command line and is explained more in the section below about running the program. The program file, *HABs_Classify.py*, is the actual program that you will be running. This program loads in the trained model, imports your images, and then predicts the classifications of the images. This is the program that is doing all of the work for you. The h5 file, *HABs_CNN_Model_FINAL.h5*, is the trained model that is evaluating the images that you import into the program. This is the file that is loaded into *HABs_Classify.py*.

The third step for set up is creating the folders that the images will be copied to once they are classified by the program. Create a new folder under *HABsProject*. I named this folder *ClassifiedPics*, but you can name it anything that you prefer. Next, inside of *ClassifiedPics* I created three subfolders. Each subfolder represents one of the groups that the images could have been classified to: bga, clear, or turbid. The subfolders must be named the following: *bgaClassified*, *clearClassified*, and *turbidClassified*. This is because this is how they are referred to by the program. If they do not match these names exactly then the program will not be able to find them. Before you run *HABs_Classify.py* for the first time, the three folders in *ClassifiedPics* should be empty. After *HABs_Classify.py* finishes running for the first time, *bgaClassified*, *clearClassified*, and *turbidClassified* should contain the images that were assigned to their specific category.

The last step is to make sure that the base folder, *HABsProject*, contains all of the images that you want to classify. All of the pictures need to go into a separate folder under *HABsProject*. These pictures will be the images that you accumulated in the “Acquiring Data” section above. They will be the pictures that are imported into the program. Therefore, I named this base folder for the images, *ImportedPics*. You can name this folder anything you would like. Then inside of this *ImportedPics* folder, you need to have three more folders. I named these *Folder1*, *Folder2*, and *Folder3*. Again, these can be named anything that you want. Any image that you want to be classified by the program needs to be in one of these folders. It does not matter which of these three folders that the images are in. What is important is that *ImportedPics* has exactly three sub-folders, and inside of these folders are the images that are to be classified. For example, all of the images could be in *Folder1*, and *Folder2* and *Folder3* could be empty.

The reason for why *ImportedPics* needs to have exactly three subfolders is due to how the structure of the program is designed. I built my program off of a couple of machine learning tutorials that I discovered. One of the characteristics of these tutorials is that the base picture folder always contains three subfolders. This is because this tutorial was loading in the images from three sorted folders. This organization allowed the program to easily use the images for training, testing, and evaluating the model. I have not figured out how to make it so the picture folder does not need subfolders. However, the program will work as long as you include these three subfolders and have images in at least one of these subfolders.

Running the Program

Summary:

- The command line syntax to run the program is below:

```
python HABs_Classify.py --dataset folderName --classpics folderName --txtfilename reportName
```
- The 3 parameters are:
 - `--dataset folderName` - the name of the folder holding the images to classify
 - `--classpics folderName` - the name of the folder that will contain the classified images
 - `--txtfilename reportName` - the name for the text file report that the program creates
- Advanced Option: the parameter below can be used to specify the name of the classifier model file
 - `--model modelName`

Detailed Explanation (How/Why/Specific Examples):

You will be running the program through your computer's terminal and command line. You can access the terminal by going to your computer's application folder and searching for "terminal". It should pop up as an option to select.

Once you have the terminal open, the first step is to make sure that you are in the correct current directory. This means that you want to make sure that the computer is looking in the folder that contains all of the files that you will be using. In our case, HABsProject is the folder that contains everything that we need to run the program. Therefore, we need to set the current directory to the path name of *HABsProject*.

For a Mac, the command for setting the current directory in the terminal is "cd". In order to set the current directory, you need to type "cd" followed by a space and the path name of the folder that you want to access. Therefore, for our example on a Mac we would type:

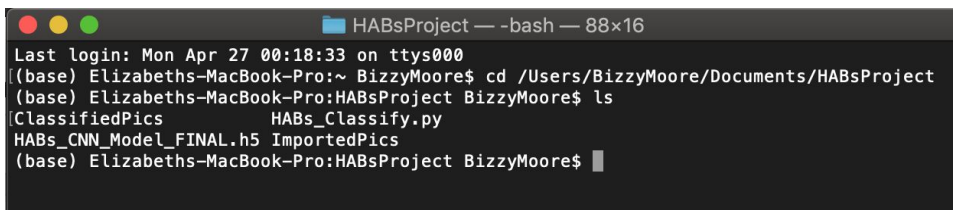
```
cd /Users/BizzyMoore/Documents/HABsProject
```

If you are using a Windows operating system, then the command for setting the current directory is below:

```
C:\Users\BizzyMoore\Documents\HABsProject
```

Press enter once you have finished typing either of these lines.

Next, you can use a specific command to view the files and folders within your current directory. This allows you to double check to make sure that all of the necessary things that are discussed in the section above are included in this directory. The picture below portrays these terminal commands and their output. For a Mac, you can use the list command, "ls", to do this. For Windows, the command for this is "dir". Below is an example of completing the above steps in a Mac terminal. I do not have access to a Windows terminal to portray these actions.



```
HABsProject — bash — 88x16
Last login: Mon Apr 27 00:18:33 on ttys000
(base) Elizabeths-MacBook-Pro:~ BizzyMoore$ cd /Users/BizzyMoore/Documents/HABsProject
(base) Elizabeths-MacBook-Pro:HABsProject BizzyMoore$ ls
ClassifiedPics      HABs_Classify.py
HABs_CNN_Model_FINAL.h5 ImportedPics
(base) Elizabeths-MacBook-Pro:HABsProject BizzyMoore$
```

Since all of the necessary parts are in the *HABsProject* directory, then it is time to run the program. You will need to know four things in order to run the program:

1. The name of the program
2. The name of the folder that holds the images you want to classify
3. The name of the folder that holds the classified images
4. The name of the text file that will include the program's report

In our example, the name of the program is *HABs_Classify.py*, the name of the folder that holds the images that we want to classify is *ImportedPics*, and the name of the folder that holds the classified images is *ClassifiedPics*. The name of the text file can be anything, but for this example, let's name it *HABsClassReport_0*. These four things will be command line arguments. This means that you can change them each time that you run the program. It gives you the flexibility to name these four things anything you want.

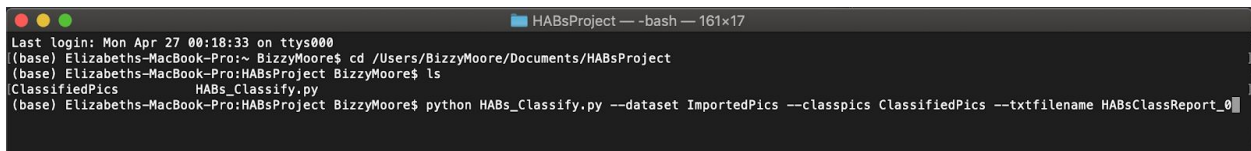
The format of running a Python program in the command line is the following:

```
python programName.py --command1 argument1 --command2 argument2 --command3 argument3
```

The commands that we are going to use are `--dataset`, `--classpics`, and `--txtfilename`. After each command, you type the respective argument after it. Therefore, for our example we would run the program by typing:

```
python HABs_Classify.py --dataset ImportedPics --classpics ClassifiedPics --txtfilename HABsClassReport_0
```

The picture below depicts the steps for running the program.

A terminal window titled 'HABsProject' with a width of 161x17. The terminal shows the following commands and output:

```
Last login: Mon Apr 27 00:18:33 on ttys000
(base) Elizabeths-MacBook-Pro:~ BizzyMoore$ cd /Users/BizzyMoore/Documents/HABsProject
(base) Elizabeths-MacBook-Pro:HABsProject BizzyMoore$ ls
ClassifiedPics      HABs_Classify.py
(base) Elizabeths-MacBook-Pro:HABsProject BizzyMoore$ python HABs_Classify.py --dataset ImportedPics --classpics ClassifiedPics --txtfilename HABsClassReport_0
```

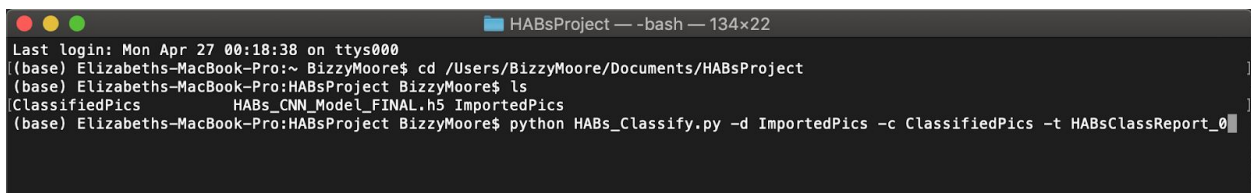
There are also shortcuts for the commands:

- `--dataset` can be shortened to `-d`
- `--classpics` can be shortened to `-c`
- `--txtfilename` can be shortened to `-t`

Therefore, we can also run our current program with the following commands:

```
python HABs_Classify.py -d ExtractedPics -c ClassifiedPics -t HABsClassReport_0
```

The picture below illustrates this.

A terminal window titled 'HABsProject' with a width of 134x22. The terminal shows the following commands and output:

```
Last login: Mon Apr 27 00:18:38 on ttys000
(base) Elizabeths-MacBook-Pro:~ BizzyMoore$ cd /Users/BizzyMoore/Documents/HABsProject
(base) Elizabeths-MacBook-Pro:HABsProject BizzyMoore$ ls
ClassifiedPics      HABs_CNN_Model_FINAL.h5 ImportedPics
(base) Elizabeths-MacBook-Pro:HABsProject BizzyMoore$ python HABs_Classify.py -d ImportedPics -c ClassifiedPics -t HABsClassReport_0
```

As mentioned in the setup section, there is an advanced option that allows you to specify a different name for the classifier model file, rather than use the default one (*HABs_CNN_Model_FINAL.h5*). The command for this is:

```
--model modelName
```

The shortcut for this command is:

```
-m modelName
```

Report in Excel

Summary:

- The report generated is a comma-delimited text file that contains the following fields:
 - *MaxValueSorter* - the program's confidence in the classification
 - *ImageName* - the original name of the image
 - *Classification* - states the image's classification group: bga, clear, or turbid
 - *bgaConfidence* - % of how confident the model was at classifying the image as bga
 - *clearConfidence* - % of how confident the model was at classifying the image as clear
 - *turbidConfidence* - % of how confident the model was at classifying the image as turbid

Importing Text File into Excel

While the program is running, it writes a report to a comma-delimited text file. Once the program is complete, this file is saved in the *HABsProject* folder. We can use Excel to look at the information that is in the text file.

In order to do this, open up a new worksheet in Excel. Click on the "File" tab that is in the menu that is at the top of the page. Then select "Import...". Select the circle next to "Text file" and select the "Import" button. Next navigate to your *HABsProject* directory and double click on the name of the text file. It should be the same name that you specified when you entered a name for the text file in the command line. Therefore, for our example, it is *HABsClassReport_0*. Once you do this, a new window should pop up. Select the "Delimited" circle and make sure that you are starting the import at row 1. Then click the "Next" button. On this new page, select the box next to "Comma". This is because each row in our text file is separated by commas. Then click the "Next" button. Lastly, make sure that the "General" circle is selected and then click "Finish". A new window will pop up and make sure that "=\$A\$1" is filled in the text box. Select "OK". You should see 6 columns of information neatly filled in. Each row represents an individual image. Therefore, there are 6 columns of information for each image.

Evaluating the Results

Once the program is finished running, and its text file has been imported into Excel, it is time to evaluate the results of how the images were classified. It is important to make sure that the images were classified correctly. This is because the program is not perfect, so there may be some outliers that were placed in the wrong folder. The text file that is created upon completion of running the images through the model helps us to be able to quickly look at the results and gauge how the program did with its classification. An example of a report once it is exported into Excel is shown below.

MaxValueSorter	ImageName	Classification	bgaConfidence	clearConfidence	turbidConfidence
6190	bgaTest2.jpg	bga	0.61905795	0.15486635	0.22607572
6204	clearTest3.jpg	turbid	0.000739711	0.37876204	0.6204983
7112	bgaTest1.jpg	bga	0.7112614	0.2885273	0.000211289
7360	turbidTest3.jpg	clear	0.000180941	0.7360357	0.2637833
8256	turbidTest1.jpg	turbid	0.000738057	0.17357405	0.82568794
9246	turbidTest2.jpg	turbid	0.012504835	0.06279887	0.92469627
9330	clearTest1.jpg	clear	0.06349413	0.9330755	0.003430344
9601	bgaTest3.jpg	bga	0.960138	0.03622552	0.003636465
9700	clearTest4.jpg	clear	2.05E-05	0.970099	0.029880516
9894	clearTest2.jpg	clear	1.23E-05	0.98945415	0.010533623

The first column *MaxValueSorter* is a value that helps us determine how confident the program was at classifying that specific image to its group. Therefore, the images are ranked in order of the value of the *MaxValueSorter*. The images with the lowest value are listed at the top of the report, and the images with the highest value are listed at the bottom of the report. Therefore, this allows us to quickly determine which images the program was least confident in classifying. When the images are saved in their new folders, “bgaClassified”, “clearClassified”, or “turbidClassified”, the *MaxValueSorter* value is concatenated to the beginning of the original filename. Therefore, this allows the images in the folders to also be sorted in order of confidence level. This makes it easier to relate the information in the report to the images in the folders. In order to make sure that the images are in sorted order in the folders, right click in the folder and make sure you are sorting by “Name”.

The *ImageName* is the original name of the image. The *Classification* column tells us which group the image got classified to: bga, clear, or turbid. The last three columns show us the percentage (out of 1) of how confident the model was at classifying the individual image to the three possible categories. The category that had the highest value is the group that the image got classified to. This information allows us to see how close the program’s decision was in predicting an image’s classification. For example, sometimes the percentage is very close between two categories, like clear and turbid. Therefore, this means that the program saw characteristics in the image that made it think that it could have been either clear or turbid. This is why it is important for us to go back and double check the results and look for outliers that could have taken place. Sorting the images by the confidence levels in both the report and the image folders allows us to do this fairly easily.