LeafMachine

User Manual

By:
William Weaver[1]
Julienne Ng[1]
Robert Laport[2]

1) University of Colorado at Boulder, Department of Ecology and Evolutionary Biology
2) Rhodes College, Department of Biology

Last Edited: January 6, 2020
Software Version – 2.0

Table of Contents

Overview

LeafMachine is a MATLAB-based software package designed to autonomously extract leaf area measurements from digitized herbarium specimen vouchers. At this time, a license to use MATLAB 2019b is required to use LeafMachine. MATLAB installation instructions are provided below; two additional MATLAB toolboxes and one MATLAB Add-On are also required to use LeafMachine.

Through a graphical user interface (GUI), a user can process a batch of images on their local computer in the three most popular environments: Mac, Windows, or Linux. LeafMachine has two image processing options designated by “2A” and “2B” in the GUI. A detailed explanation of intended use cases can be found in the section “Three Example Scenarios.” Additionally, LeafMachine contains setup tools that can be used to subset a larger dataset or download images using the LeafMachine naming convention. These tools, in section “1” of the LeafMachine GUI, are optional and are used in specific circumstances described later in this manual. LeafMachine saves numerous output files for each specimen, which is described in detail in the LeafMachine Output section of the user manual. At this time, LeafMachine saves measurements in pixels not metric areas. A conversion factor can be applied post-processing based on a manual measurement of a ruler in the image.

2A. Download and process images from a URL directly from a set of Darwin Core files

- This option allows a user to select images for processing based on the contents of an “images.csv” file and an “occurrences.csv” file that adhere to the Darwin Core standard.
- Using path 2A, LeafMachine reads the “images.csv” file and matches the coreID numbers in the first column with the corresponding id number in the “occurrences.csv” file.
- Since LeafMachine starts with the “images.csv,” we provide a tool to filter the “images.csv” based on a given “occurrences.csv” file.
- The “images.csv” file often does not contain any identifying information other than the coreID number. So if a user only wants to analyze specimens from the genus Ginkgo, they could delete all non-Ginkgo rows from their “occurrences.csv” file and use the “Generate Custom “images.csv” File” tool to create a new “images_Custom.csv” file containing only the image URLs for their desired specimens. The new “images_Custom.csv” file can then be used for path 2A with the corresponding “occurrences.csv” file.
    i. This process also checks for broken URLs.
- More information about the Darwin Core standard can be found in section “Darwin Core Standard” section of the user manual.

2B. Process images from an existing local directory

- Path 2B allows a local directory of existing images to be processed by LeafMachine.
- The following image file types are supported:
    i. .jpg
    ii. .jpeg
    iii. .png
- Any directory of images can be processed, but the existing filenames may not be desirable. It is common practice for herbaria to name image files with only a catalog number e.g. "00235240.jpg" while LeafMachine names each file in the format: HerbaiumCode_CatalogNumber_Family_Genus_Species.jpg. This file format makes manual QC and data analysis simpler for the user.
- LeafMachine uses the family name as one of the eleven variables in the SVM process. If an image is saved in the format "00235240.jpg," then the algorithm will use the ten other variables. If the LeafMachine filename format is used (by downloading images using the third tool in section 1 of the LeafMachine GUI), then LeafMachine can parse the filename and obtain the family name.
- The filename format does not need to match the LeafMachine format exactly. As long as the family name is isolated by underscores, LeafMachine will be able to parse the filename and use the family name. For example, the filename "Solanaceae_image_1.jpg" would allow LeafMachine to obtain the family name, while "SolanaceaeImage1.jpg" would not.
- A download tool at the bottom of the LeafMachine GUI section 1 allows users to download images from URLs. This function uses Darwin Core "images.csv" and "occurrences.csv" files to create an image filename that is more informative than a typical herbarium image filename.

LeafMachine System Requirements

Hardware Requirements

- CPU - Modern Intel or AMD multicore processor
    - Note: A higher single-core CPU clock speed improves performance more than a high-core count.
- RAM
    - At least 8 GB of system memory if using a GPU
    - At least 16 GB of system memory without a GPU
- GPU
    - A Nvidia CUDA-capable GPU is recommended to take advantage of GPU acceleration for the semantic segmentation process but is not required.
- Hard disk space
    - To estimate storage space requirements, multiply the total size of the input images by a factor of five. More space is required if the "save high quality images" option is selected. Less space is required if fewer output options are chosen than what is set by default.
    - If 1,000 input images take up 1 GB, then the total output size will be around 5 GB.

Estimated Processing Time per Image by Computer System Specifications					
System	**CPU**	**GPU**	**RAM**	**Storage Space**	**Avg. Processing Time per Image**
MacBook Air (2013)	Intel Core i5 2.8 GHz	N/A	8 GB	128 GB	2MP – 40 sec. 21MP – 310 sec.
MacBook Pro 15" (2017)	Intel Core i7 3.1 GHz	N/A	16 GB	1 TB	2MP – 14 sec. 21MP – 114 sec.
High-end Consumer PC	Intel Core i7-8700K 4.8 GHz	Nvidia GeForce RTX 2070 8 GB VRAM	32 GB	2 TB	2MP – 9 sec. 21MP – 71 sec.
Mid-tier Workstation	Intel Xeon E3-1245 v6 3.7 GHz	Nvidia Quadro P4000 8 GB VRAM	64 GB	10 TB	2MP – 12 sec. 21MP – 88 sec.
High-end Workstation	Intel Xeon W-2145 4.5 GHz	Nvidia Quadro P6000 24 GB VRAM	128 GB	12 TB	2MP – 8 sec. 21MP – 69 sec.

Table 1: Average processing times were calculated by running the demo validation images on each computer. The average processing time per specimen was similar with large batches. The high-end consumer grade PC averaged one specimen every 10.2 seconds for a 9,940-image low-resolution batch and the high-end workstation averaged one specimen every 204 seconds for a 9,940-image high-resolution batch. The number of leaf candidate masks (LCMs) processed by the support vector machine (SVM) drives deviations from the average.

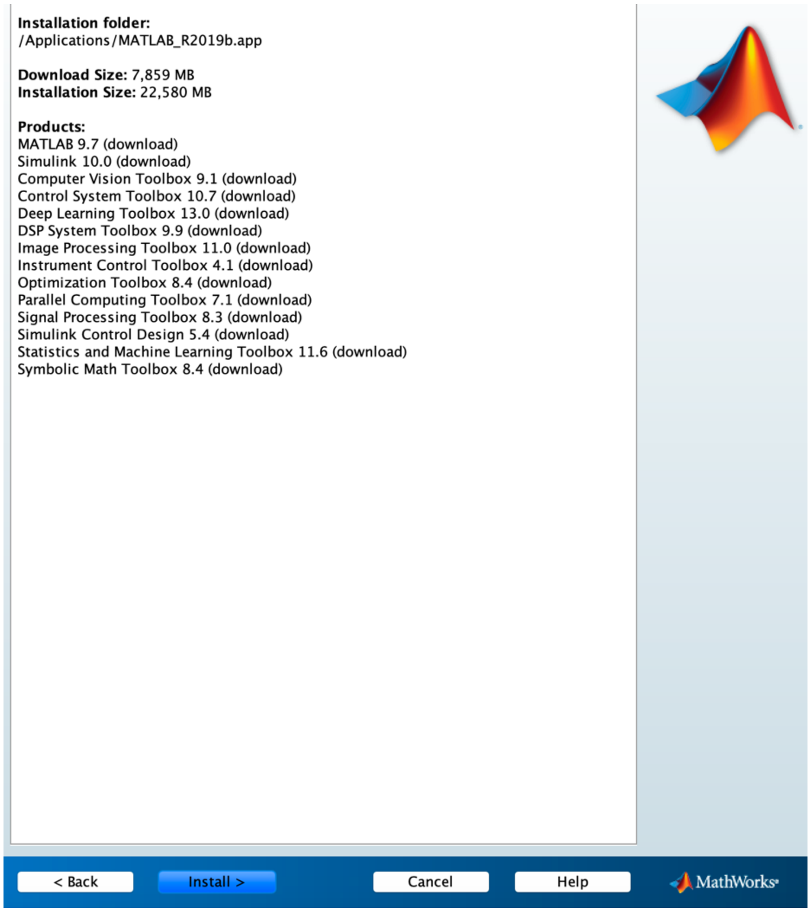
Software Requirements

LeafMachine is built with MATLAB 2019b. Running the software requires a MATLAB license. To install MATLAB on your local machine, navigate to:

https://www.mathworks.com/downloads/web_downloads/download_release?release=R2019b

Download MATLAB 2019b for your operating system. Follow the installation instructions and login to your Mathworks account when prompted. When installing on MacOS, you will also need to install Xcode from the Apple App Store. MATLAB may ask to install a C compiler, please accept and continue installation. During installation you will be asked to select which MATLAB toolboxes to install. If you have enough hard drive space (37 GB) you can install all of the toolboxes, otherwise install the MATLAB recommended toolboxes including:

- MATLAB 9.7
- Simulink 10.0
- Computer Vision Toolbox 9.1
- Control System Toolbox 10
- Deep Learning Toolbox 13.0
- DSP System Toolbox 9.9
- Image Processing Toolbox 11.0
- Instrument Control Toolbox 4.1
- Optimization Toolbox 8.4
- Parallel Computing Toolbox 7.1
- Signal Processing Toolbox 8.3
- Simulink Control Design 5.4
- Statistics and Machine Learning Toolbox 11.6
- Symbolic Math Toolbox 8.4



Installation folder:
/Applications/MATLAB_R2019b.app

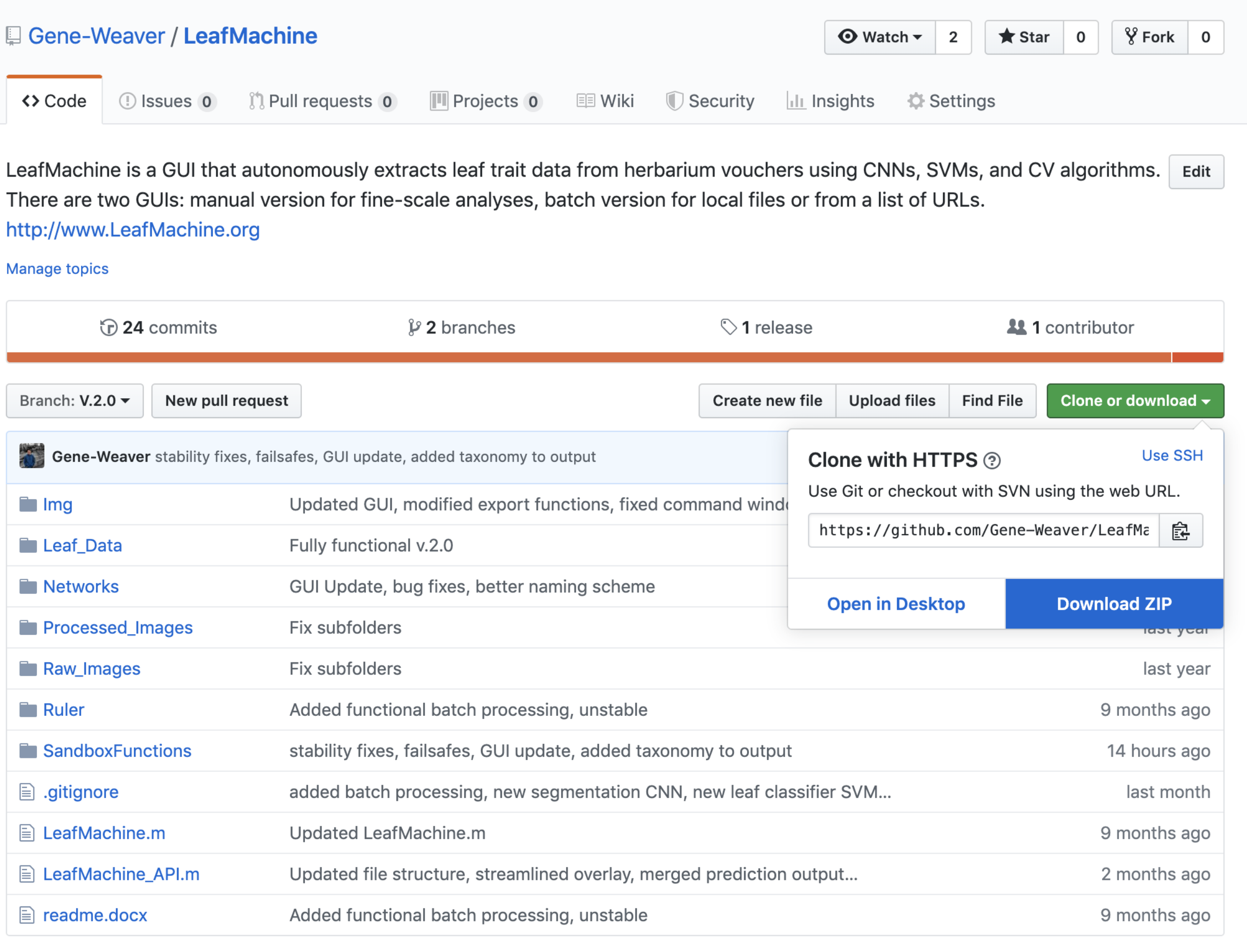
Download Size: 7,859 MB
Installation Size: 22,580 MB

Products:
MATLAB 9.7 (download)
Simulink 10.0 (download)
Computer Vision Toolbox 9.1 (download)
Control System Toolbox 10.7 (download)
Deep Learning Toolbox 13.0 (download)
DSP System Toolbox 9.9 (download)
Image Processing Toolbox 11.0 (download)
Instrument Control Toolbox 4.1 (download)
Optimization Toolbox 8.4 (download)
Parallel Computing Toolbox 7.1 (download)
Signal Processing Toolbox 8.3 (download)
Simulink Control Design 5.4 (download)
Statistics and Machine Learning Toolbox 11.6 (download)
Symbolic Math Toolbox 8.4 (download)

Screenshots used in this manual are taken from MacOS and may appear slightly different for Windows or Linux users.

Downloading LeafMachine

1. Navigate to https://github.com/Gene-Weaver/LeafMachine and click the green "Clone or download" button. Select "Download ZIP" and extract the contents using your preferred tool.



Gene-Weaver / LeafMachine

Watch 2 | Star 0 | Fork 0

<> Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

LeafMachine is a GUI that autonomously extracts leaf trait data from herbarium vouchers using CNNs, SVMs, and CV algorithms. Edit
There are two GUIs: manual version for fine-scale analyses, batch version for local files or from a list of URLs.
http://www.LeafMachine.org

Manage topics

24 commits | 2 branches | 1 release | 1 contributor

Branch: V.2.0 | New pull request | Create new file | Upload files | Find File | Clone or download

Gene-Weaver stability fixes, failsafes, GUI update, added taxonomy to output

Img	Updated GUI, modified export functions, fixed command wind	
Leaf_Data	Fully functional v.2.0	
Networks	GUI Update, bug fixes, better naming scheme	
Processed_Images	Fix subfolders	last year
Raw_Images	Fix subfolders	last year
Ruler	Added functional batch processing, unstable	9 months ago
SandboxFunctions	stability fixes, failsafes, GUI update, added taxonomy to output	14 hours ago
.gitignore	added batch processing, new segmentation CNN, new leaf classifier SVM…	last month
LeafMachine.m	Updated LeafMachine.m	9 months ago
LeafMachine_API.m	Updated file structure, streamlined overlay, merged prediction output…	2 months ago
readme.docx	Added functional batch processing, unstable	9 months ago

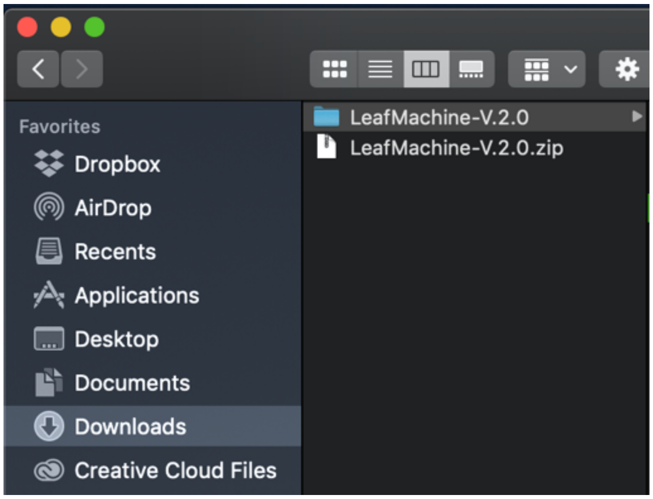
Clone with HTTPS | Use SSH

Use Git or checkout with SVN using the web URL.

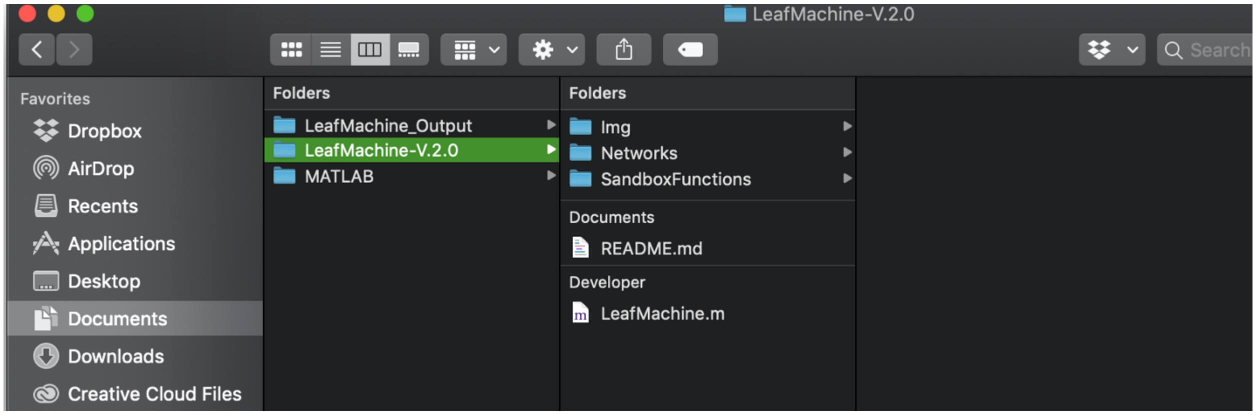
`https://github.com/Gene-Weaver/LeafMa`

Open in Desktop | Download ZIP

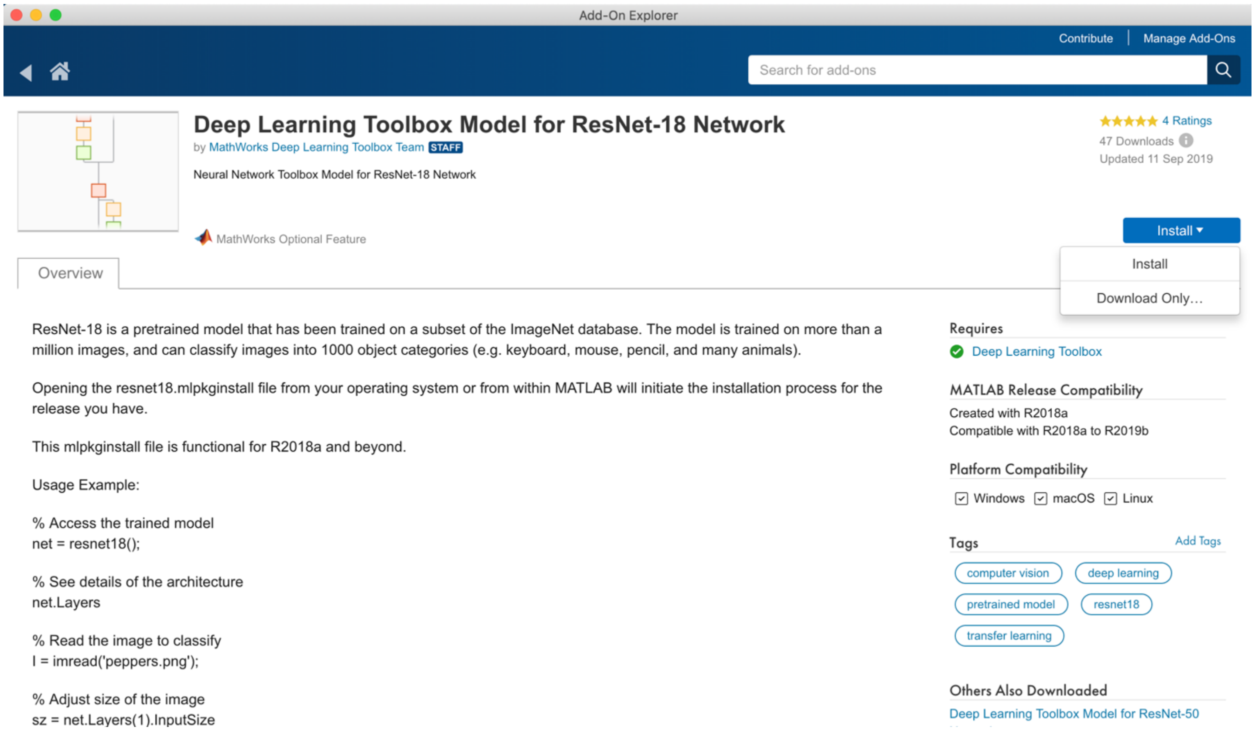
2. Move the folder called "LeafMachine-V.2.0" to a permanent location. We recommend that you create an output directory in the same location, in this case we called the output directory "LeafMachine_Output."



Move to a permanent location.



3. An additional tool is required to run LeafMachine. Open MATLAB 2019b, in the "Home" tab locate the "Add-Ons" button and click "Get Add-Ons." Search for "Deep Learning Toolbox Model for ResNet-18 Network" published by the MathWorks Deep Learning Toolbox Team. Follow the installation instructions.



Deep Learning Toolbox Model for ResNet-18 Network

by MathWorks Deep Learning Toolbox Team STAFF

Neural Network Toolbox Model for ResNet-18 Network

★★★★★ 4 Ratings
47 Downloads
Updated 11 Sep 2019

ResNet-18 is a pretrained model that has been trained on a subset of the ImageNet database. The model is trained on more than a million images, and can classify images into 1000 object categories (e.g. keyboard, mouse, pencil, and many animals).

Opening the resnet18.mlpkginstall file from your operating system or from within MATLAB will initiate the installation process for the release you have.

This mlpkginstall file is functional for R2018a and beyond.

Usage Example:

```
% Access the trained model
net = resnet18();

% See details of the architecture
net.Layers

% Read the image to classify
I = imread('peppers.png');

% Adjust size of the image
sz = net.Layers(1).InputSize
```

Requires

✓ Deep Learning Toolbox

MATLAB Release Compatibility

Created with R2018a
Compatible with R2018a to R2019b

Platform Compatibility

☑ Windows ☑ macOS ☑ Linux

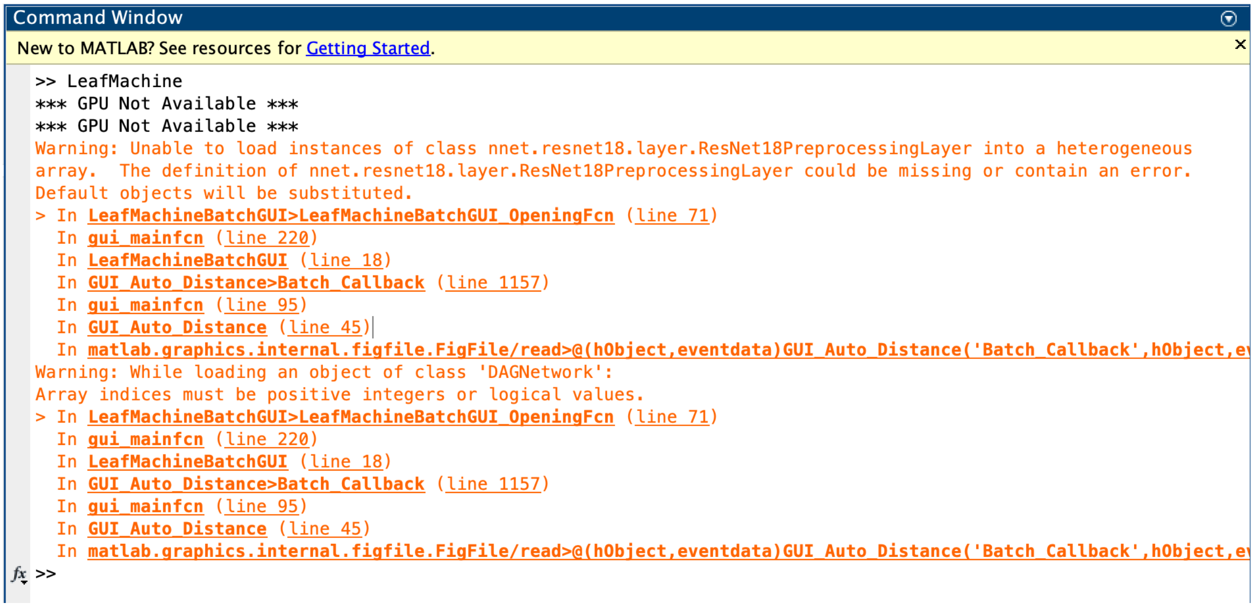
Tags

computer vision, deep learning, pretrained model, resnet18, transfer learning

Others Also Downloaded

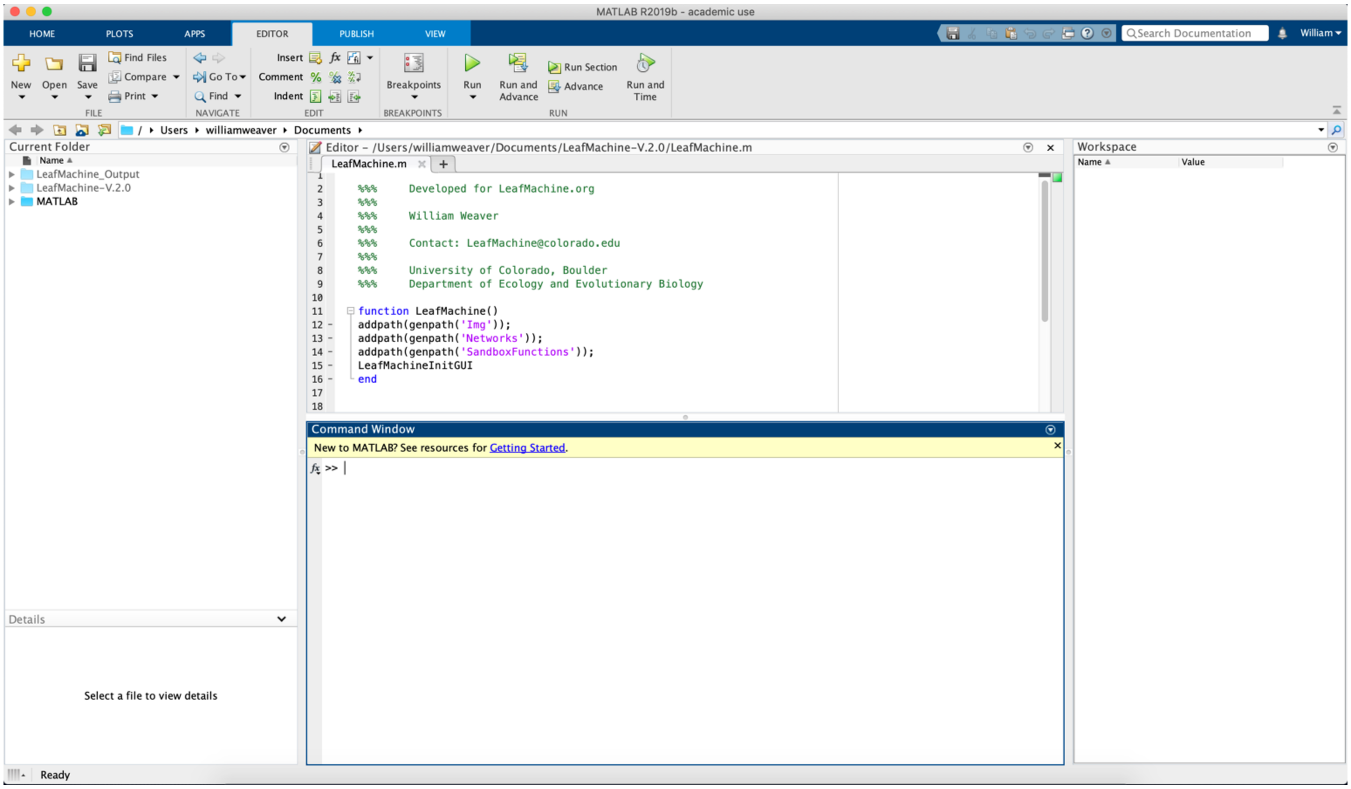
Deep Learning Toolbox Model for ResNet-50

If the error below appears when LeafMachine initiates, the ResNet18 toolbox was not installed.



```
>> LeafMachine
*** GPU Not Available ***
*** GPU Not Available ***
Warning: Unable to load instances of class nnet.resnet18.layer.ResNet18PreprocessingLayer into a heterogeneous
array.  The definition of nnet.resnet18.layer.ResNet18PreprocessingLayer could be missing or contain an error.
Default objects will be substituted.
> In LeafMachineBatchGUI>LeafMachineBatchGUI_OpeningFcn (line 71)
  In gui_mainfcn (line 220)
  In LeafMachineBatchGUI (line 18)
  In GUI_Auto_Distance>Batch_Callback (line 1157)
  In gui_mainfcn (line 95)
  In GUI_Auto_Distance (line 45)
  In matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)GUI_Auto_Distance('Batch_Callback',hObject,ev
Warning: While loading an object of class 'DAGNetwork':
Array indices must be positive integers or logical values.
> In LeafMachineBatchGUI>LeafMachineBatchGUI_OpeningFcn (line 71)
  In gui_mainfcn (line 220)
  In LeafMachineBatchGUI (line 18)
  In GUI_Auto_Distance>Batch_Callback (line 1157)
  In gui_mainfcn (line 95)
  In GUI_Auto_Distance (line 45)
  In matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)GUI_Auto_Distance('Batch_Callback',hObject,ev
fx >>
```

4. Inside the “LeafMachine-V.2.0” folder is a file called “LeafMachine.m” which will initiate LeafMachine. Right-click this file and open it with MATLAB 2019b. Doing this will set the MATLAB working directory to the “LeafMachine-V.2.0” directory. Notice that the LeafMachine folders in the Current Folder window are light gray - they are not yet in the MATLAB path.



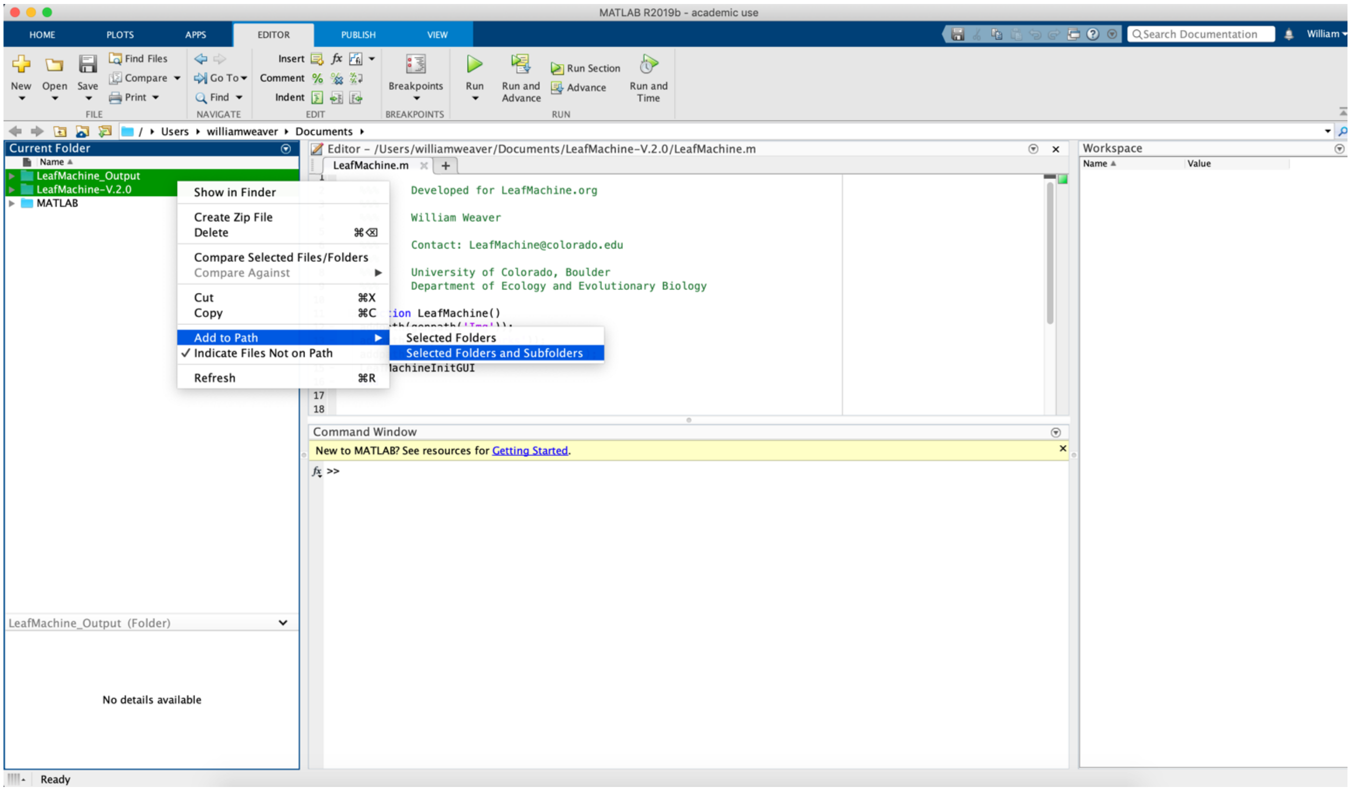
Editor – /Users/williamweaver/Documents/LeafMachine-V.2.0/LeafMachine.m

LeafMachine.m

```

%%%     Developed for LeafMachine.org
%%%
%%%     William Weaver
%%%
%%%     Contact: LeafMachine@colorado.edu
%%%
%%%     University of Colorado, Boulder
%%%     Department of Ecology and Evolutionary Biology

function LeafMachine()
addpath(genpath('Img'));
addpath(genpath('Networks'));
addpath(genpath('SandboxFunctions'));
LeafMachineInitGUI
end

```

```
fx >>
```

5. In order to run LeafMachine, all associated files must be added to the MATLAB path. Right-click “LeafMachine-V.2.0” and “LeafMachine_Output” and select “Add to Path” and then “Selected Folders and Subfolders.”

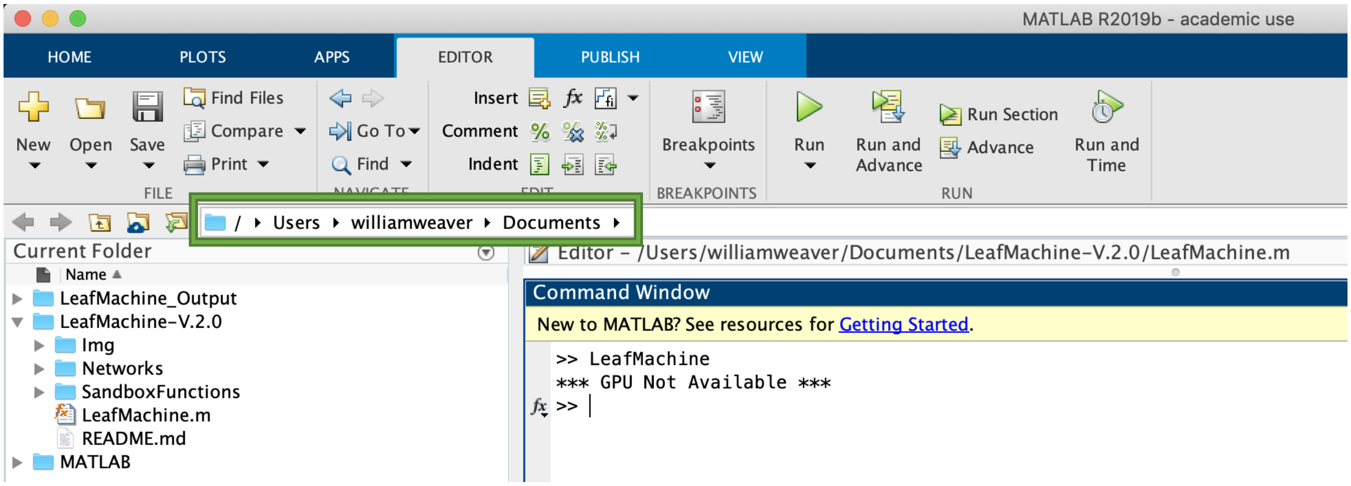


```
Developed for LeafMachine.org

William Weaver

Contact: LeafMachine@colorado.edu

University of Colorado, Boulder
Department of Ecology and Evolutionary Biology

:ion LeafMachine()

lachineInitGUI
```

New to MATLAB? See resources for Getting Started.

```
>>
```

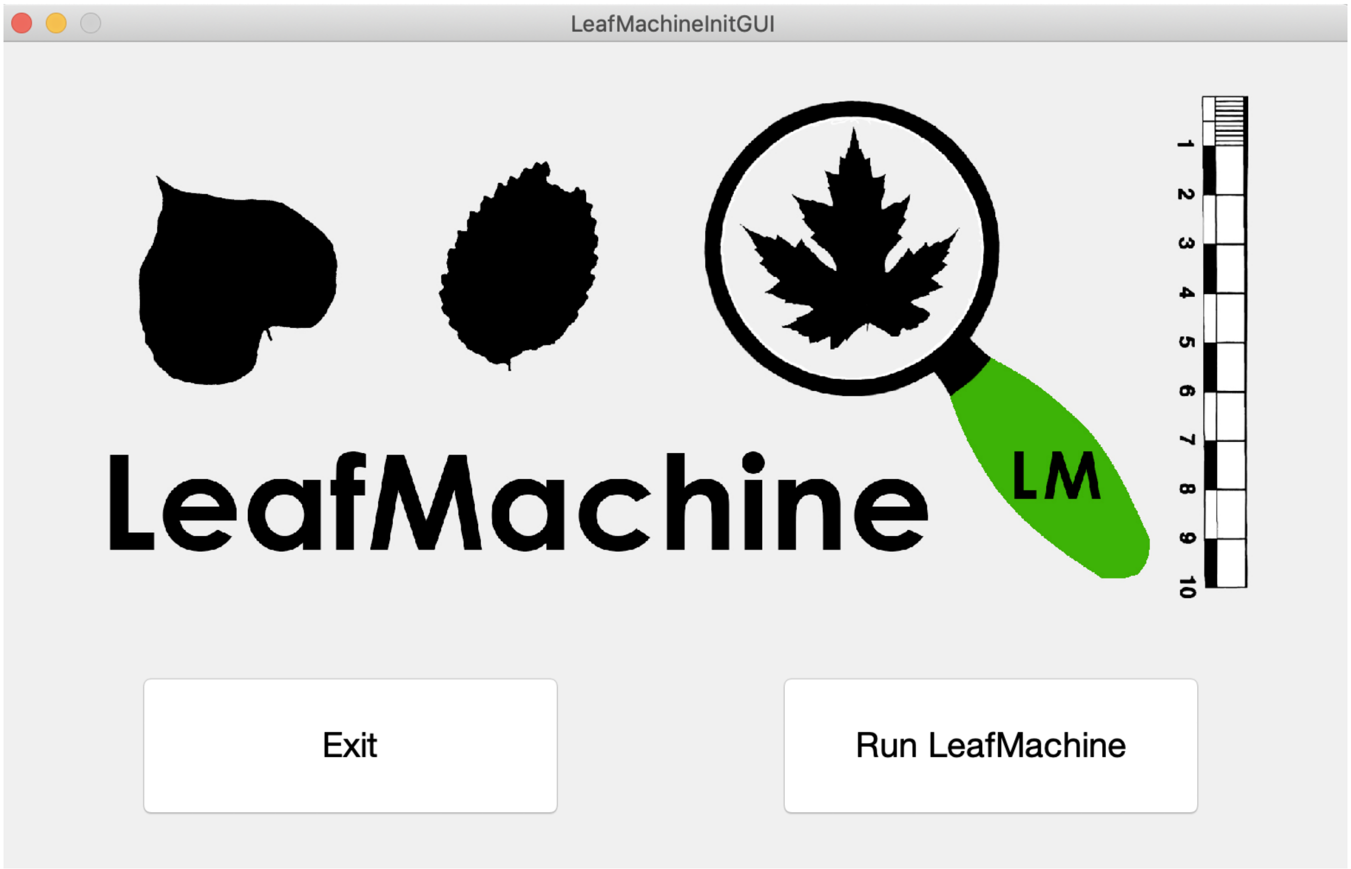
6. To start LeafMachine, either push the triangular green “Run” button or type “LeafMachine” into the Command Window. A GPU check will run and try to find an available GPU, the result will be displayed in the Command Window. If your machine has multiple GPUs, the first device will be selected. The green box indicates your working directory, make sure this is set to the location where all LeafMachine files are saved.



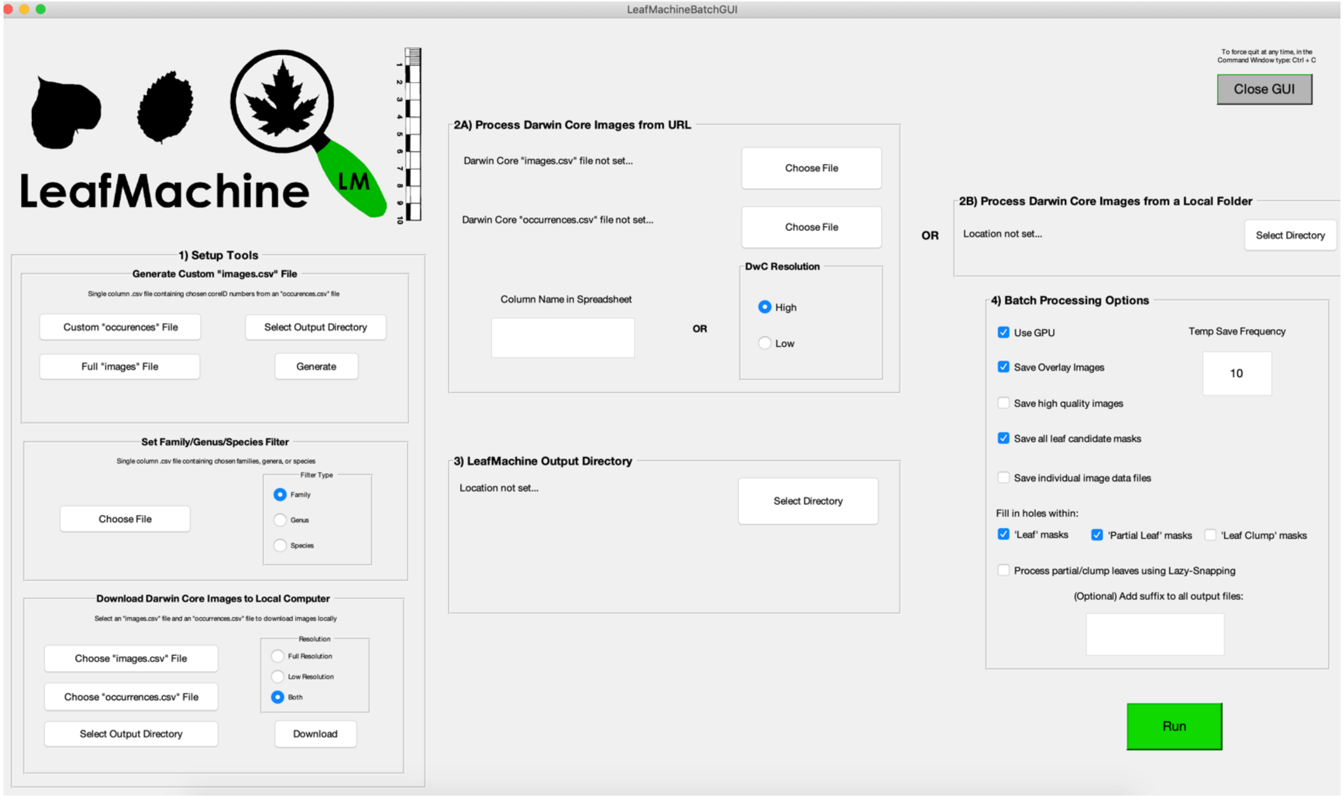
New to MATLAB? See resources for Getting Started.

```
>> LeafMachine
*** GPU Not Available ***
>> |
```

7. To run LeafMachine, press the “Run” button.



8. The main LeafMachine GUI will appear with default options selected.



1) Setup Tools

Generate Custom "images.csv" File

Single column .csv file containing chosen coreID numbers from an "occurences.csv" file

Set Family/Genus/Species Filter

Single column .csv file containing chosen families, genera, or species

Filter Type

● Family
○ Genus
○ Species

Download Darwin Core Images to Local Computer

Select an "images.csv" file and an "occurrences.csv" file to download images locally

Resolution

○ Full Resolution
○ Low Resolution
● Both

2A) Process Darwin Core Images from URL

Darwin Core "images.csv" file not set...

Darwin Core "occurrences.csv" file not set...

Column Name in Spreadsheet

DwC Resolution

● High
○ Low

3) LeafMachine Output Directory

Location not set...

To force quit at any time, in the Command Window type: Ctrl + C

2B) Process Darwin Core Images from a Local Folder

Location not set...

4) Batch Processing Options

☑ Use GPU

Temp Save Frequency: 10

☑ Save Overlay Images

□ Save high quality images

☑ Save all leaf candidate masks

□ Save individual image data files

Fill in holes within:

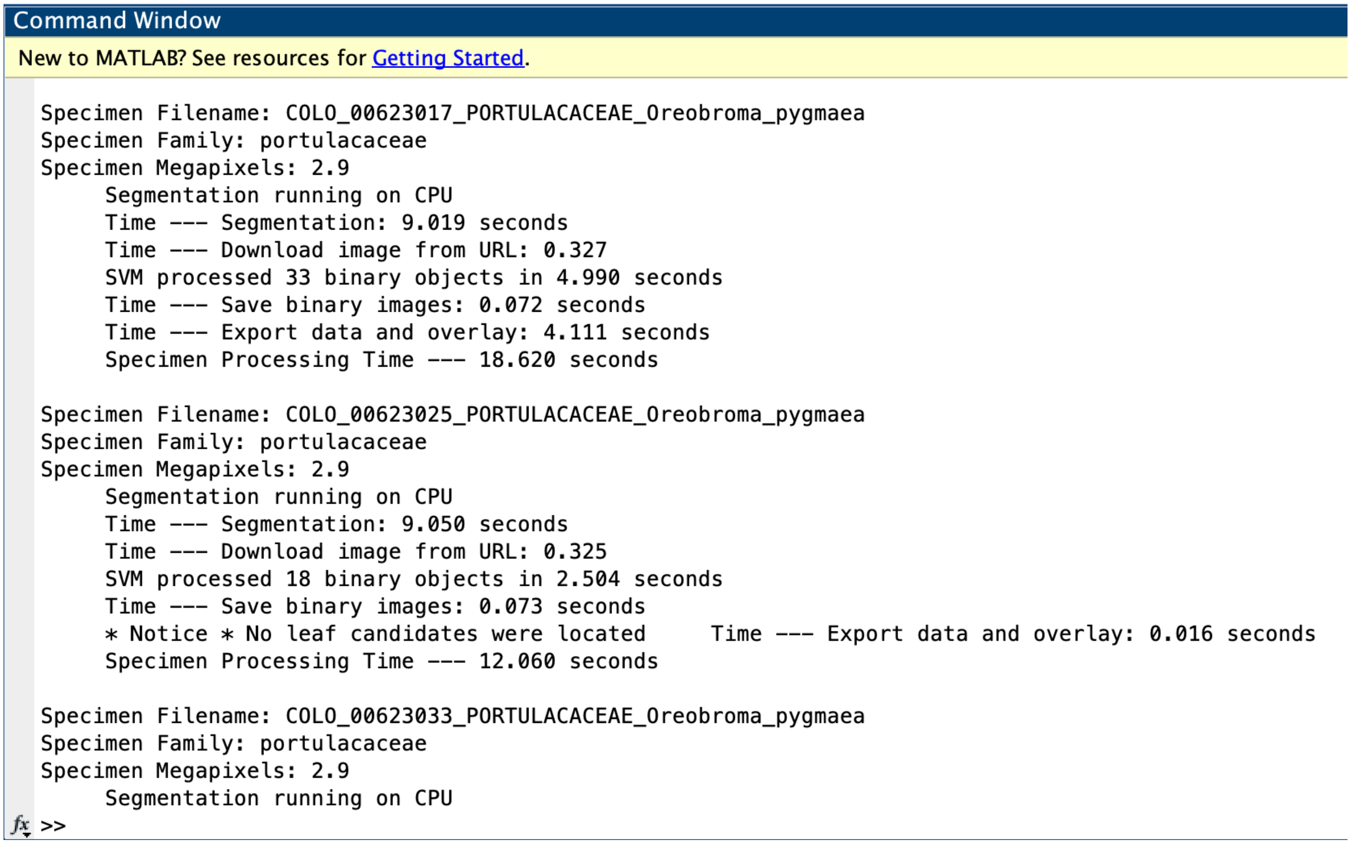
☑ 'Leaf' masks ☑ 'Partial Leaf' masks □ 'Leaf Clump' masks

□ Process partial/clump leaves using Lazy-Snapping

(Optional) Add suffix to all output files:

9. The LeafMachine GUI has five main sections described in detail in the "Using LeafMachine – GUI Overview" section of the user manual:
    1) Setup tools
    2A) Processing from Darwin Core files
    2A) Processing local images
    3) Setting output directory
    4) Setting LeafMachine parameters

10. After selecting options, the Command Window will display updates, like the example below, as LeafMachine processes each image.



```
Specimen Filename: COLO_00623017_PORTULACACEAE_Oreobroma_pygmaea
Specimen Family: portulacaceae
Specimen Megapixels: 2.9
     Segmentation running on CPU
     Time --- Segmentation: 9.019 seconds
     Time --- Download image from URL: 0.327
     SVM processed 33 binary objects in 4.990 seconds
     Time --- Save binary images: 0.072 seconds
     Time --- Export data and overlay: 4.111 seconds
     Specimen Processing Time --- 18.620 seconds

Specimen Filename: COLO_00623025_PORTULACACEAE_Oreobroma_pygmaea
Specimen Family: portulacaceae
Specimen Megapixels: 2.9
     Segmentation running on CPU
     Time --- Segmentation: 9.050 seconds
     Time --- Download image from URL: 0.325
     SVM processed 18 binary objects in 2.504 seconds
     Time --- Save binary images: 0.073 seconds
     * Notice * No leaf candidates were located     Time --- Export data and overlay: 0.016 seconds
     Specimen Processing Time --- 12.060 seconds

Specimen Filename: COLO_00623033_PORTULACACEAE_Oreobroma_pygmaea
Specimen Family: portulacaceae
Specimen Megapixels: 2.9
     Segmentation running on CPU
fx >>
```

Using LeafMachine – GUI Overview

1) Setup Tools

Generate Custom "images.csv" File

LeafMachine processes a batch of images by iterating through the "images.csv" file and matching the first column coreid to the coreid found in the associated "occurrences.csv" file. This order was chosen because many Darwin Core specimen entries do not have an associated image. When choosing images to process, a user is unlikely to need to process an entire collection, so only a portion of a full Darwin Core file will be used.

The "occurrences.csv" file contains identifying information such as family, genus, species, etc. A user can save only the rows of interest as the input for LeafMachine. However, due to the way LeafMachine processes images, a new "images.csv" file must be generated first. The only identifying information in a Darwin Core "images.csv" file is the coreid, so this LeafMachine setup tool matches a user's custom, shortened "occurences.csv" file with the full Darwin Core "images.csv" file to generate a new custom "images.csv" file that can then be used to process images with LeafMachine. During this process three files are generated:

- Images_Custom.csv
- Occurrences_NoImages.csv
- Occurrencs_WithImages.csv

The first and third files can then be used to process your images with the LeafMachine 2A) option. The second file will contain a list of the Darwin Core specimens that are not accessible by URL.

Set Family/Genus/Species Filter

Users can create a list of desired families, genera, and species to filter the inputs for the LeafMachine 2A) option. This filter will **not** work for the LeafMachine 2B) option. The input file needs to be a single column .csv file **with** a header – the name of the header does not matter.

Selecting family, genus, or species will tell LeafMachine which filter type to sort by. As a result, it is possible to have families, genera, and species mixed into the same filter file, but LeafMachine will only filter for one taxonomic rank at a time.

Download Darwin Core Images to Local Computer

This setup tool will download images from a pair of "images.csv" and "occurrences.csv" Darwin Core files, saving the images in the format HerbariumCode_catalogNumber_Family_Genus_Species.jpg using as much information as is present in the "occurrences.csv" file. Processing images from a local machine is an alternative method of using LeafMachine. We recommended that files are downloaded in this way if you are planning to use the LeafMachine 2B) option.

If you plan to run LeafMachine on a local directory that contains image files saved in a different filename format, the LeafMachine output files will be more difficult to interpret, since the LeafMachine

2B) option populates output data with information from the filename. Often, herbaria save images using the catalog number, which will make assessing the output files tedious.

2A) Process Darwin Core Images from URL

For more information about Darwin Core files, see the “Darwin Core Standard” section.

1. Choose your “images.csv” file.
   - This is either the standard Darwin Core file that ties an occurrence record to an image stored on a remote server, or a custom file generated by the “Generate Custom “images.csv” File” tool mentioned above.
2. Choose your “occurrences.csv” file.
   - This is the standard Darwin Core file containing an herbarium specimen occurrence record, or a shortened version containing only specimens of interest.
3. Select whether you want to process the high-resolution version or the low-resolution version. Note that some herbaria only share low resolution images publicly.
4. Optional: If your “image.csv” does not follow Darwin Core standards or the column header names do not exactly match the headers of the example input files provided with LeafMachine, then you can type in the header name of the column that contains the URLs. This feature is not recommended.

2B) Process Darwin Core Images from a Local Folder

Choose the directory that contains the images you want to process with LeafMachine. LeafMachine can handle most common image formats, but we recommend processing .jpg, .jpeg, or .png files if possible. This directory should contain ***only*** the images that you intend to process.

3) LeafMachine Output Directory

Choose the location where LeafMachine output files will be stored. In the file structure shown above, we would choose “LeafMachine_Output.”

4) Batch Processing Options

- Use GPU
  - If selected, LeafMachine will use a GPU to accelerate the semantic segmentation algorithm. If this option is selected but a GPU is unavailable, LeafMachine will default to processing with the CPU only. LeafMachine is not parallelized for multi-GPU setups. By default, only the first GPU will be selected.
- Save overlay image
  - This is the best way to visually inspect the LeafMachine output but can be disabled to save time or reduce the overall size of the output files.

- Save high quality image
  - This option will save the overlay images in a higher quality .png format. Only use this option if you are processing a small number of images.
- Save all leaf candidate masks
  - This option will save all rejected leaf candidate masks, which can number in the tens of thousands for large batches.
- Save individual image data files
  - This will save a .csv data file for each processed image.
- Fill in holes within
  - If selected, this uses the MATLAB function imfill() to fill in holes in the semantic segmentation mask generated by the convolutional neural network. This is helpful for reducing noise caused by less than ideal CNN predictions, but will also fill in real holes caused by damage or herbivory. We recommend that this option stay turned off unless you notice excessive gaps or holes in the overlay images. The imfill() procedure is applied after the support vector machine makes a prediction for a given leaf candidate mask.
- Add suffix to all output files
  - This will add a suffix to every output file. This can be helpful if you have multiple batches to run but want to use the same output directory. For example, if you want to run one batch without filling in holes, you can leave the suffix textbox blank. Then if you select “fill in holes for ‘leaf’ masks” you could type “filled_in” in the suffix text box, while keeping the directory chosen in the LeafMachine 3) option the same. Running a batch a second time without adding a suffix while using the same output directory may result in overwriting the output data files.
- Temp Save Frequency
  - Determines how frequently the temporary data file is saved to the “Data_Temp” folder. For example, setting this value to 10 would save a copy of the data output file every tenth image. This is useful if the power goes out or you need to stop a run for any reason. You can resume where you left off or at the minimum, not lose data.
- Other
  - Close GUI
    - Closes the LeafMachine GUI. If you want to abort mid-run, type CTRL+C into the Command Window. This will force quit the run. To reinitialize LeafMachine, click the “Run LeafMachine” button in the LeafMachine launch window.
  - Run
    - Start processing images. If 3) has not been set, you will be prompted to set an output directory first. When a run is successfully completed, the Run button will turn blue and say “Reset”. Pressing reset will reinitialize LeafMachine and you can begin a new run.

LeafMachine Demo Files

Inside LeafMachine-V.2.0 is a folder called “LeafMachine_Demo” that contains verified example datasets and demonstrates the necessary file formats. Included is an “images_Example.csv” file with 20 example images (15 specimens from COLO, 5 specimens from SWMT) and an “occurrences_Example.csv” file that has merged COLO and SWMT occurrence entries from the publicly available Darwin Core files obtained through a database such as the SERNEC Portal.

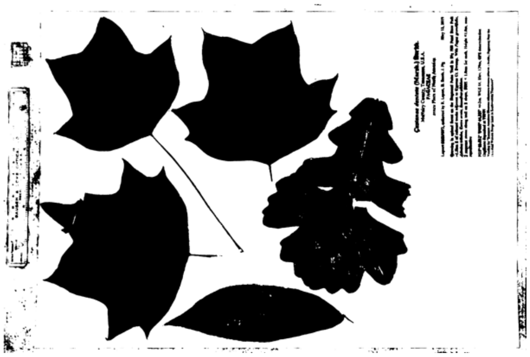
To process these files, follow the steps to implement the LeafMachine 2A) panel, set an output directory in the LeafMachine 3) panel, set processing settings in the LeafMachine 4) panel, and press “Run.” For reference, the output you should produce from running these example files is given in the “Output_HighRes” and “Output_LowRes” folders.

Additionally, we have provided 12 high-resolution and 12 low-resolution mock-vouchers that we used to validate LeafMachine’s performance located in the “Validation_Images” folder. These can be processed using the LeafMachine 2B) panel.

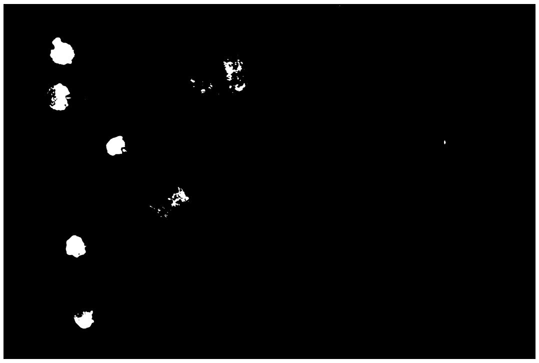
LeafMachine Output – Directories and Examples

Below is a list of the output directories, as well as an example of the files stored in each. Most use examples from the LM_Validate_2.jpg image included in the "Demo" folder. The example for fruit/flower is not from the LM_Validate set, since it contains no fruit/flowers class objects, as well as for leaf clump and leaf partial. Image orientations may be altered to minimize the space taken within this document.

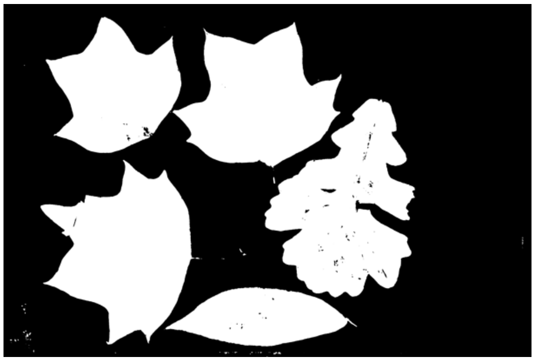
- Class_Background
  - Stores full-image binary mask of background identified pixels.



- Class_FruitFlower
  - Stores full-image binary mask of fruit/flower identified pixels.

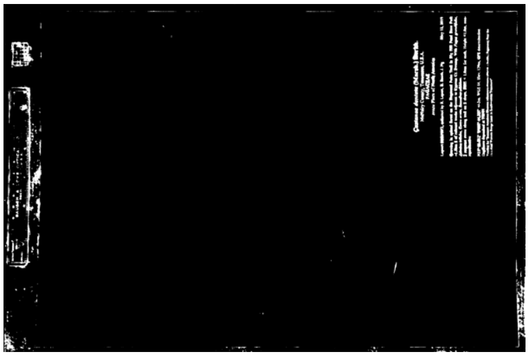


- Class_Leaf
  - Stores full-image binary mask of leaf identified pixels.

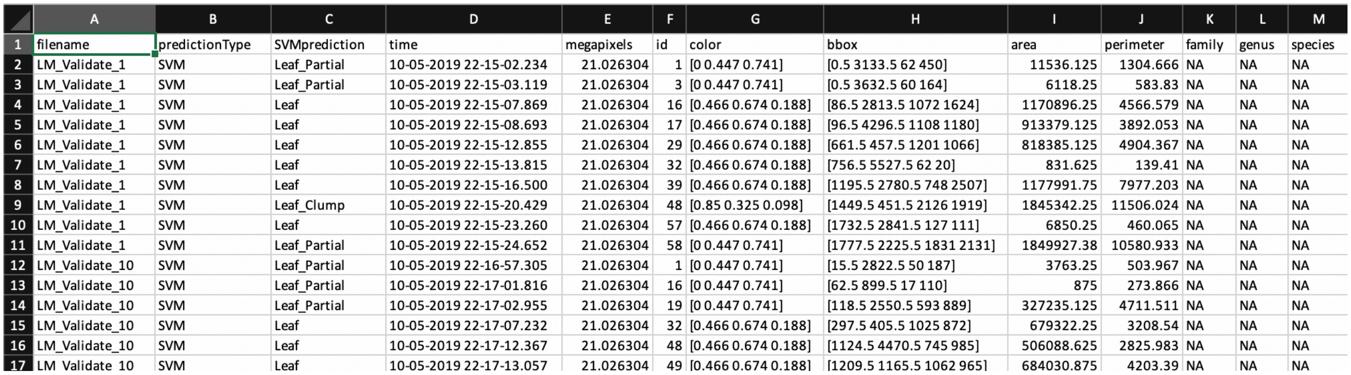


- Class_Stem
  - Stores full-image binary mask of stem identified pixels.

- Class_Text
  - Stores full-image binary mask of text identified pixels.

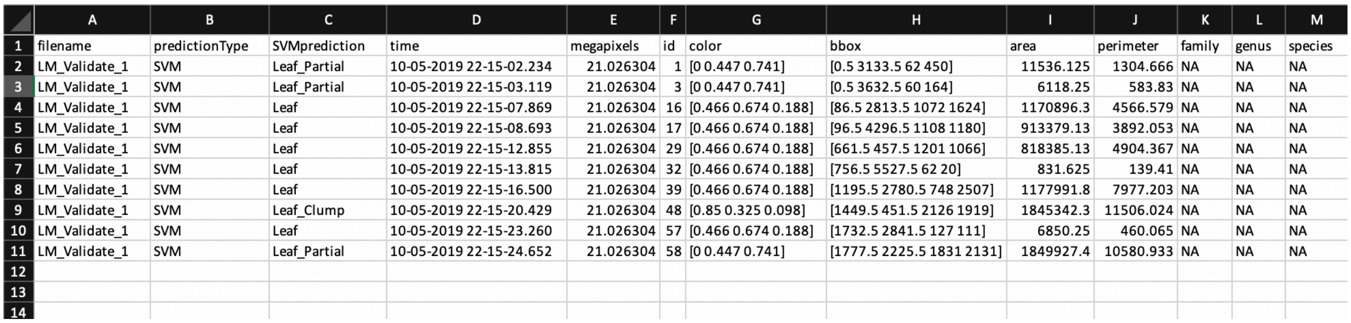


- Data
  - The final data files will be saved here. At this time, measurements are stored in pixels. Users can apply a conversion factor to get metric units by manually measuring a ruler in the image.



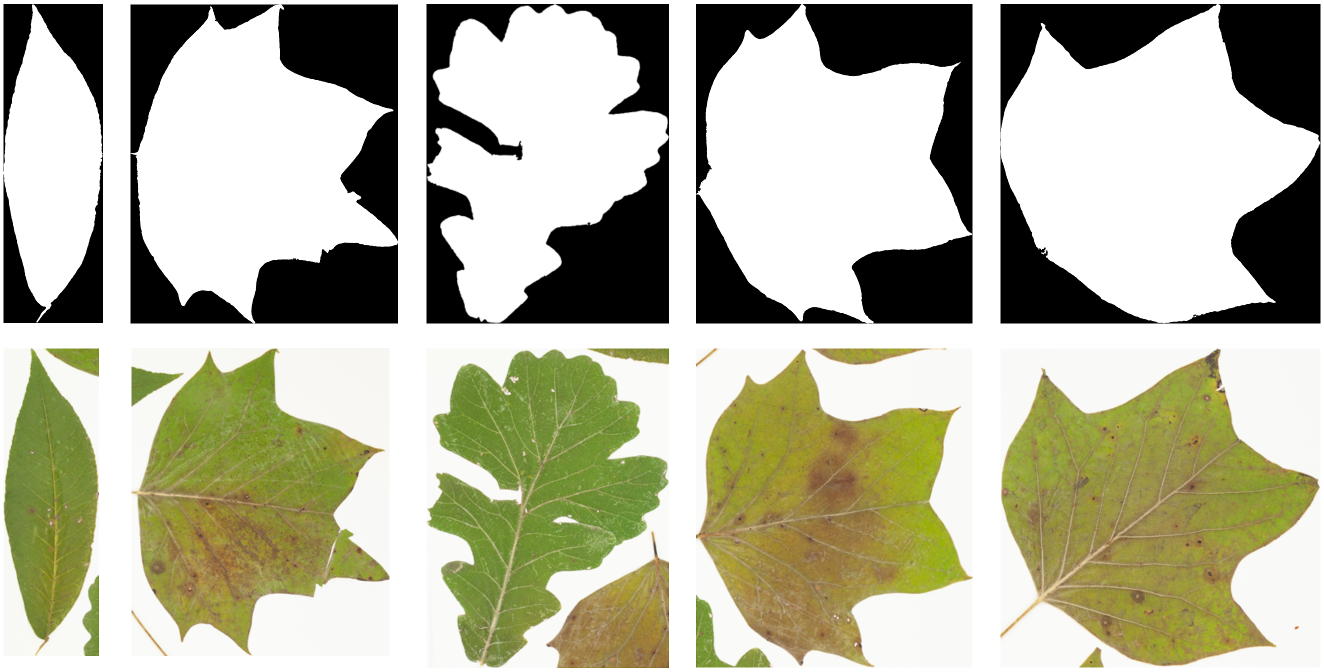
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	filename	predictionType	SVMprediction	time	megapixels	id	color	bbox	area	perimeter	family	genus	species
2	LM_Validate_1	SVM	Leaf_Partial	10-05-2019 22-15-02.234	21.026304	1	[0 0.447 0.741]	[0.5 3133.5 62 450]	11536.125	1304.666	NA	NA	NA
3	LM_Validate_1	SVM	Leaf_Partial	10-05-2019 22-15-03.119	21.026304	3	[0 0.447 0.741]	[0.5 3632.5 60 164]	6118.25	583.83	NA	NA	NA
4	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-07.869	21.026304	16	[0.466 0.674 0.188]	[86.5 2813.5 1072 1624]	1170896.25	4566.579	NA	NA	NA
5	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-08.693	21.026304	17	[0.466 0.674 0.188]	[96.5 4296.5 1108 1180]	913379.125	3892.053	NA	NA	NA
6	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-12.855	21.026304	29	[0.466 0.674 0.188]	[661.5 457.5 1201 1066]	818385.125	4904.367	NA	NA	NA
7	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-13.815	21.026304	32	[0.466 0.674 0.188]	[756.5 5527.5 62 20]	831.625	139.41	NA	NA	NA
8	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-16.500	21.026304	39	[0.466 0.674 0.188]	[1195.5 2780.5 748 2507]	1177991.75	7977.203	NA	NA	NA
9	LM_Validate_1	SVM	Leaf_Clump	10-05-2019 22-15-20.429	21.026304	48	[0.85 0.325 0.098]	[1449.5 451.5 2126 1919]	1845342.25	11506.024	NA	NA	NA
10	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-23.260	21.026304	57	[0.466 0.674 0.188]	[1732.5 2841.5 127 111]	6850.25	460.065	NA	NA	NA
11	LM_Validate_1	SVM	Leaf_Partial	10-05-2019 22-15-24.652	21.026304	58	[0 0.447 0.741]	[1777.5 2225.5 1831 2131]	1849927.38	10580.933	NA	NA	NA
12	LM_Validate_10	SVM	Leaf_Partial	10-05-2019 22-16-57.305	21.026304	1	[0 0.447 0.741]	[15.5 2822.5 50 187]	3763.25	503.967	NA	NA	NA
13	LM_Validate_10	SVM	Leaf_Partial	10-05-2019 22-17-01.816	21.026304	16	[0 0.447 0.741]	[62.5 899.5 17 110]	875	273.866	NA	NA	NA
14	LM_Validate_10	SVM	Leaf_Partial	10-05-2019 22-17-02.955	21.026304	19	[0 0.447 0.741]	[118.5 2550.5 593 889]	327235.125	4711.511	NA	NA	NA
15	LM_Validate_10	SVM	Leaf	10-05-2019 22-17-07.232	21.026304	32	[0.466 0.674 0.188]	[297.5 405.5 1025 872]	679322.25	3208.54	NA	NA	NA
16	LM_Validate_10	SVM	Leaf	10-05-2019 22-17-12.367	21.026304	48	[0.466 0.674 0.188]	[1124.5 4470.5 745 985]	506088.625	2825.983	NA	NA	NA
17	LM_Validate_10	SVM	Leaf	10-05-2019 22-17-13.057	21.026304	49	[0.466 0.674 0.188]	[1209.5 1165.5 1062 965]	684030.875	4203.39	NA	NA	NA

- Data_Temp
    - Stores a rolling copy of the final output data file. If the option to save an individual data file for each specimen is selected, the data files will be saved to this directory.

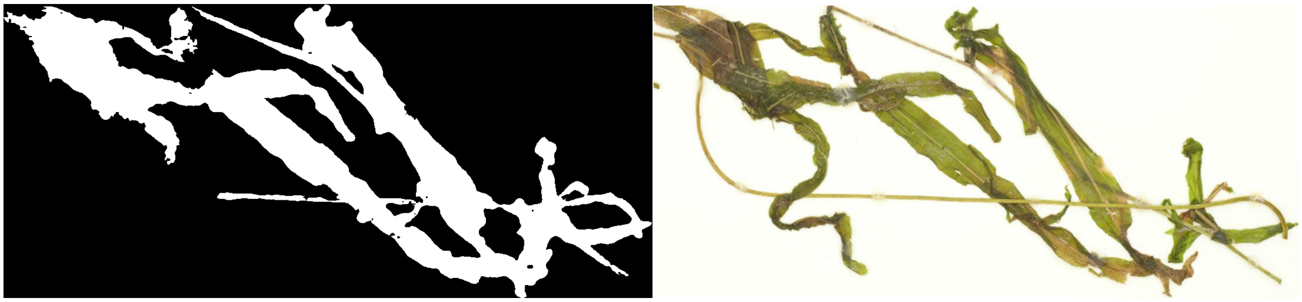


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	filename	predictionType	SVMprediction	time	megapixels	id	color	bbox	area	perimeter	family	genus	species
2	LM_Validate_1	SVM	Leaf_Partial	10-05-2019 22-15-02.234	21.026304	1	[0 0.447 0.741]	[0.5 3133.5 62 450]	11536.125	1304.666	NA	NA	NA
3	LM_Validate_1	SVM	Leaf_Partial	10-05-2019 22-15-03.119	21.026304	3	[0 0.447 0.741]	[0.5 3632.5 60 164]	6118.25	583.83	NA	NA	NA
4	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-07.869	21.026304	16	[0.466 0.674 0.188]	[86.5 2813.5 1072 1624]	1170896.3	4566.579	NA	NA	NA
5	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-08.693	21.026304	17	[0.466 0.674 0.188]	[96.5 4296.5 1108 1180]	913379.13	3892.053	NA	NA	NA
6	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-12.855	21.026304	29	[0.466 0.674 0.188]	[661.5 457.5 1201 1066]	818385.13	4904.367	NA	NA	NA
7	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-13.815	21.026304	32	[0.466 0.674 0.188]	[756.5 5527.5 62 20]	831.625	139.41	NA	NA	NA
8	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-16.500	21.026304	39	[0.466 0.674 0.188]	[1195.5 2780.5 748 2507]	1177991.8	7977.203	NA	NA	NA
9	LM_Validate_1	SVM	Leaf_Clump	10-05-2019 22-15-20.429	21.026304	48	[0.85 0.325 0.098]	[1449.5 451.5 2126 1919]	1845342.3	11506.024	NA	NA	NA
10	LM_Validate_1	SVM	Leaf	10-05-2019 22-15-23.260	21.026304	57	[0.466 0.674 0.188]	[1732.5 2841.5 127 111]	6850.25	460.065	NA	NA	NA
11	LM_Validate_1	SVM	Leaf_Partial	10-05-2019 22-15-24.652	21.026304	58	[0 0.447 0.741]	[1777.5 2225.5 1831 2131]	1849927.4	10580.933	NA	NA	NA
12													
13													
14													

- Leaf
    - Stores the binary and RGB cropped leaf candidate masks that LeafMachine deemed to be a leaf. The examples below were processed with the flood-fill option enabled.

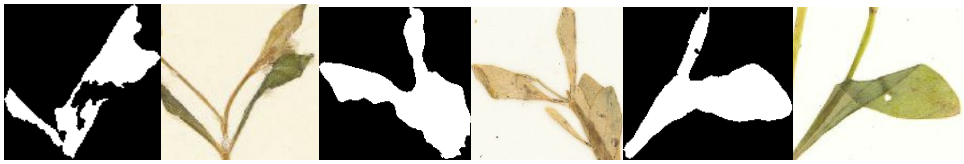


- Leaf_Clump
    - Stores the binary and RGB cropped leaf candidate masks that LeafMachine deemed to be a clump of leaves.



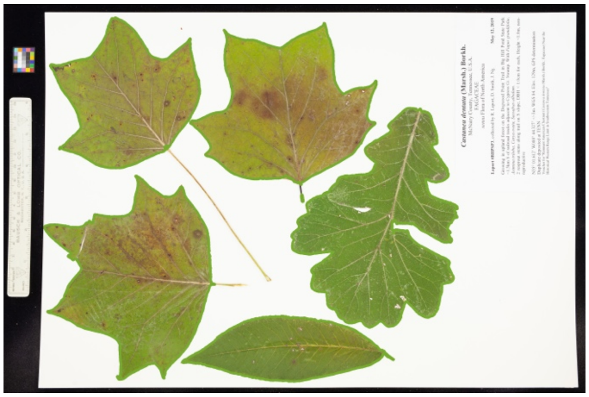
- Leaf_Fail (if selected in LeafMachine 4) panel)
    - If selected in the Batch Processing Options, this stores the binary images of the rejected leaf candidate masks. The vast majority of images in this directory are noise.

- Leaf_Partial
    - Stores the binary and RGB cropped leaf candidate masks that LeafMachine deemed to be a partial leaf.

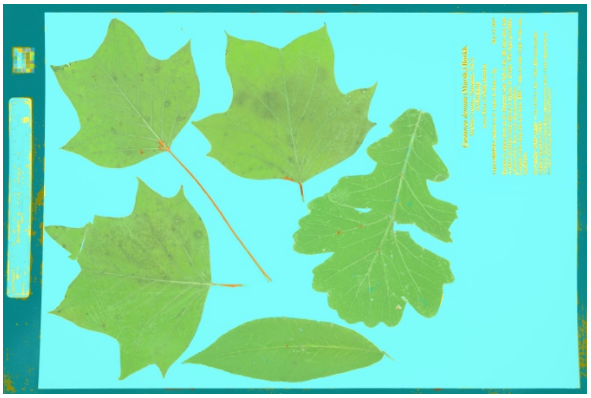


- Original (if downloading from URL)
    - Stores a copy of the original image.

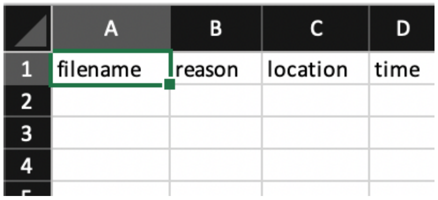
- Overlay (if selected in LeafMachine 4) panel)
  - Stores an overlay image that outlines each chosen leaf candidate mask. Green outlines denote a leaf, blue outlines denote a partial leaf, orange outlines denote a clump of leaves.



- Segmentation
  - Stores the convolutional neural network output which is a pixel-wise semantically segmented mask overlaid on the original image.

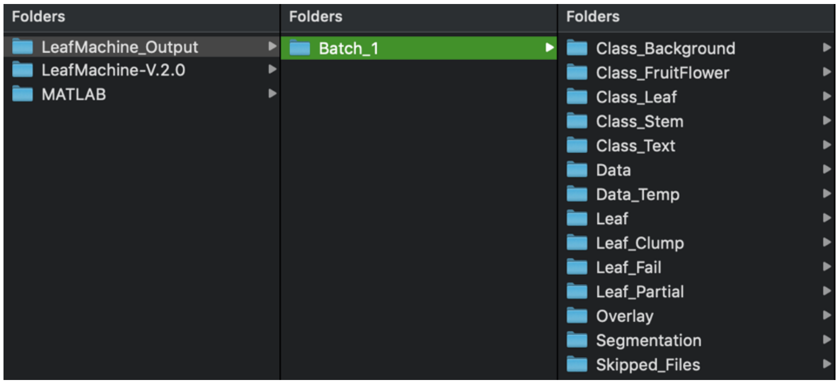


- Skipped_Files
  - Stores a spreadsheet of skipped specimens. LeafMachine may skip a file if the URL is broken or the download times out after more than 20 seconds.



	A	B	C	D
1	filename	reason	location	time
2				
3				
4				

Below is an example of the LeafMachine output file structure. The ‘Original’ directory is missing because this batch used the LeafMachine 2B) option – processing from a local directory.



Folders

LeafMachine_Output
LeafMachine-V.2.0
MATLAB

Folders

Batch_1

Folders

Class_Background
Class_FruitFlower
Class_Leaf
Class_Stem
Class_Text
Data
Data_Temp
Leaf
Leaf_Clump
Leaf_Fail
Leaf_Partial
Overlay
Segmentation
Skipped_Files

Three Example Scenarios

LeafMachine is designed and tested to process images in three scenarios:

1. Process an entire specimen voucher collection using information contained within the “occurrences.csv” and “images.csv” Darwin Core files.
2. Process a small subset of an entire collection using information contained within the “occurrences.csv” and “images.csv” Darwin Core files.
3. Process locally stored images within a directory.

Below are model procedures for completing each task.

1. Process an entire collection

a. Example scenario – A researcher looking for all leaf areas in the Sapindaceae family from the SWMT herbarium.

First, locate and download a set of “occurrences.csv” and “images.csv” Darwin Core files from one of the many herbaria data portals. Move these files to a permanent location. Initiate LeafMachine and begin with panel 2A) by loading the files into LeafMachine. If you are using a moderately powerful computer, select the high-resolution option, otherwise, select the low-resolution option.

Then in Microsoft Excel or another csv editor, create a single column table with the header ‘family’ and Sapindaceae as the first entry. Save this file and upload it to the filter option in the LeafMachine 1) panel. Click the family button to set the filter type.

Now set the output directory in the LeafMachine 3) panel. All output files will be saved here in the same file structure as described in the “LeafMachine Output Directories” section of this user manual.

Finally, set the desired parameters in the LeafMachine 4) panel. We are running this dataset for the first time, so we will just use the default values. We press run and wait for LeafMachine to finish.

2. Process subset of entire collection

a. Example scenario – A researcher looking for all leaf areas in the four counties that neighbor a field site.

We obtained a set of “occurrences.csv” and “images.csv” Darwin Core files for both SWMT and COLO to increase our chances of finding occurrence records in the four counties that are near our field site. In Excel, we merge the images files and save this copy as “images_Counties.csv”. Then we merge the occurrences files and sort by county, saving a new occurrences version that contains only the records for the four counties near our field site as “occurrences_Counties.csv”.

In LeafMachine, we need to generate a new images.csv file from the “images_Counties.csv” and “occurrences_Counties.csv” files. In the LeafMachine 1) panel we upload the “occurrences_Counties.csv” file as well as the merged “images_Counties.csv” file. Then we select an output directory and push the “generate” button.

From the output directory we upload the “images_Custom.csv” and “occurrencs_WithImages.csv” files to the LeafMachine 2A) panel. We want both high- and low-resolution results, so we start with the high-resolution option. Then we set our output directory in the LeafMachine 3) panel and add the suffix “HR” in the LeafMachine 4) panel. Using default options, we press “Run.”

Now to run the low-resolution images, we reset and replicate the same procedure, but select the low-resolution button in the LeafMachine 2A) panel and add the suffix “LR” in the LeafMachine 4) panel. The output directory can be the same as the “HR” run, or we can select a new output directory. Finally, we can look in the “Overlay” folder within the output directory and compare the results between high and low resolutions.

3. Process local images

a. Example scenario – A researcher that took their own photos of 40 specimens while visiting the COLO herbarium. The filenames are whatever the camera called them.

For the best results, we would rename the images in the format HerbariumCode_catalogNumber_Family_Genus_Species.jpg so that LeafMachine output files are more user friendly. The original filename was img_00003213.jpg. So, we look at the photo, read the label and barcode and build a new filename: COLO_01142983_Oleaceae_Fraxinus_americana.jpg. A different specimen was only identified to the family, so its new filename would be: COLO_01013002_Solanaceae__.jpg. The extra underscores help LeafMachine populate information, since LeafMachine is expecting all three taxonomic levels.

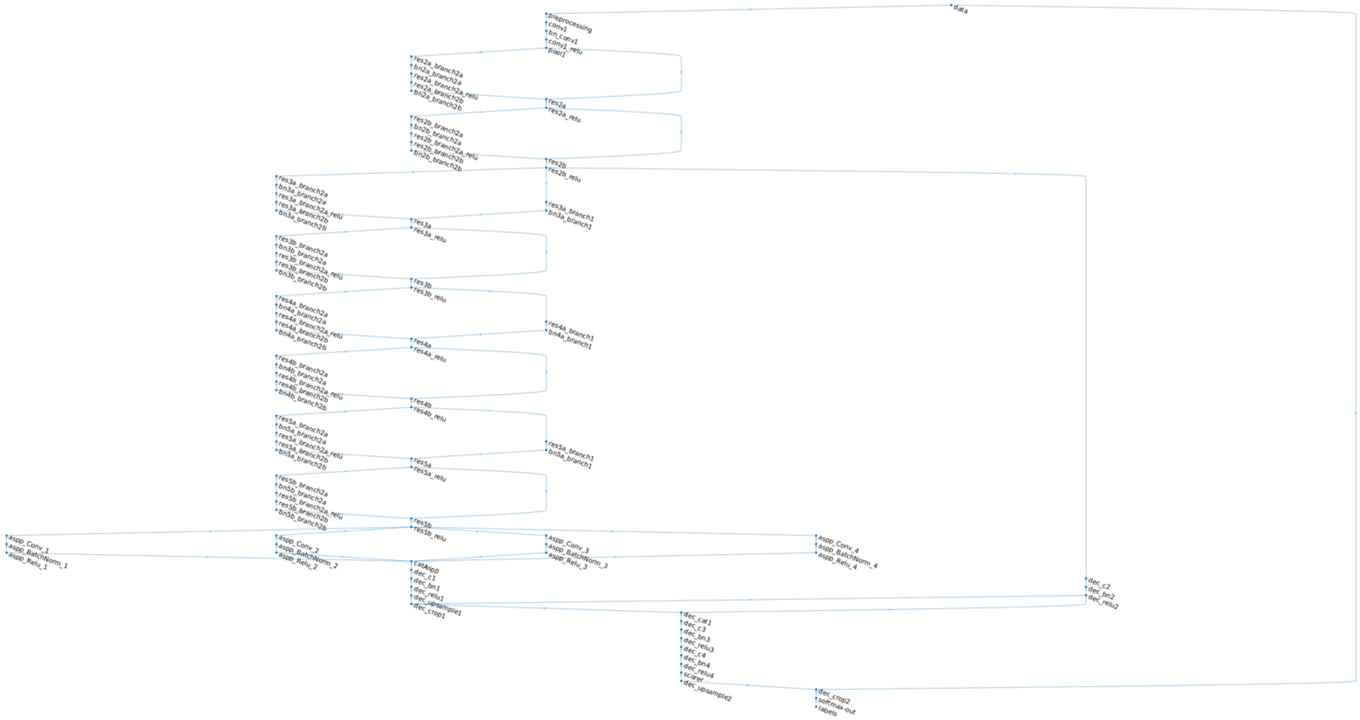
After renaming the images, we initiate LeafMachine and select the directory that contains **only** the renamed images for the LeafMachine 2B) panel. We set an output directory in the LeafMachine 3) panel and press “Run.”

LeafMachine Algorithms and Evaluation

Semantic Segmentation (CNN)

LeafMachine uses a MATLAB provided pretrained ResNet-18 feature detection network stacked in front of a DeeplabV3+ convolutional neural network (LM-CNN). This combination increased training speed and segmentation precision. The LM-CNN has 101 layers and accepts 360x360x3 dimensional inputs. We trained the network for 5 epochs with 122,000 image chips, validating every 10 iterations with 20,000 image chips, and shuffling the training data each epoch.

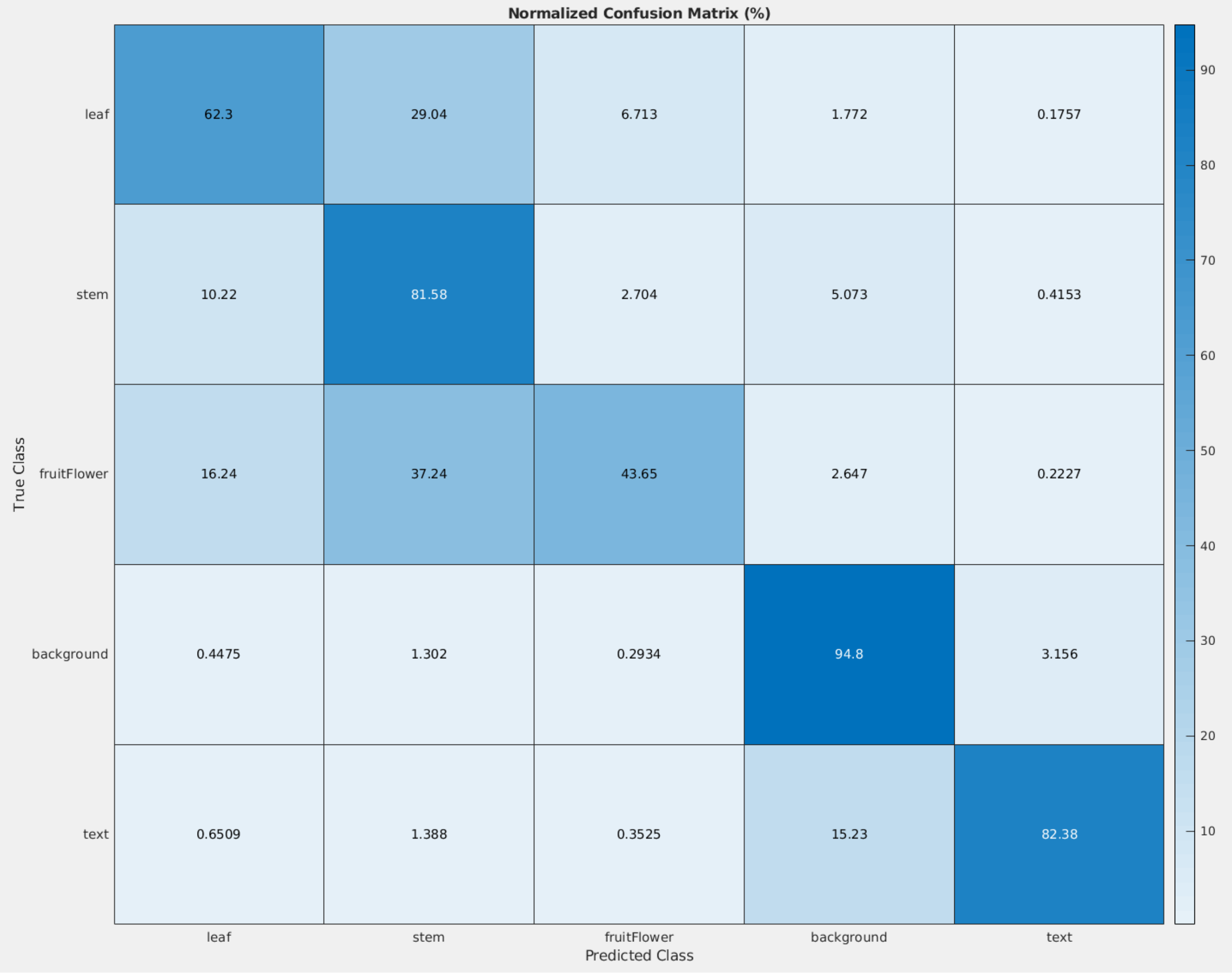
LM-CNN Network Map



Confusion Matrices

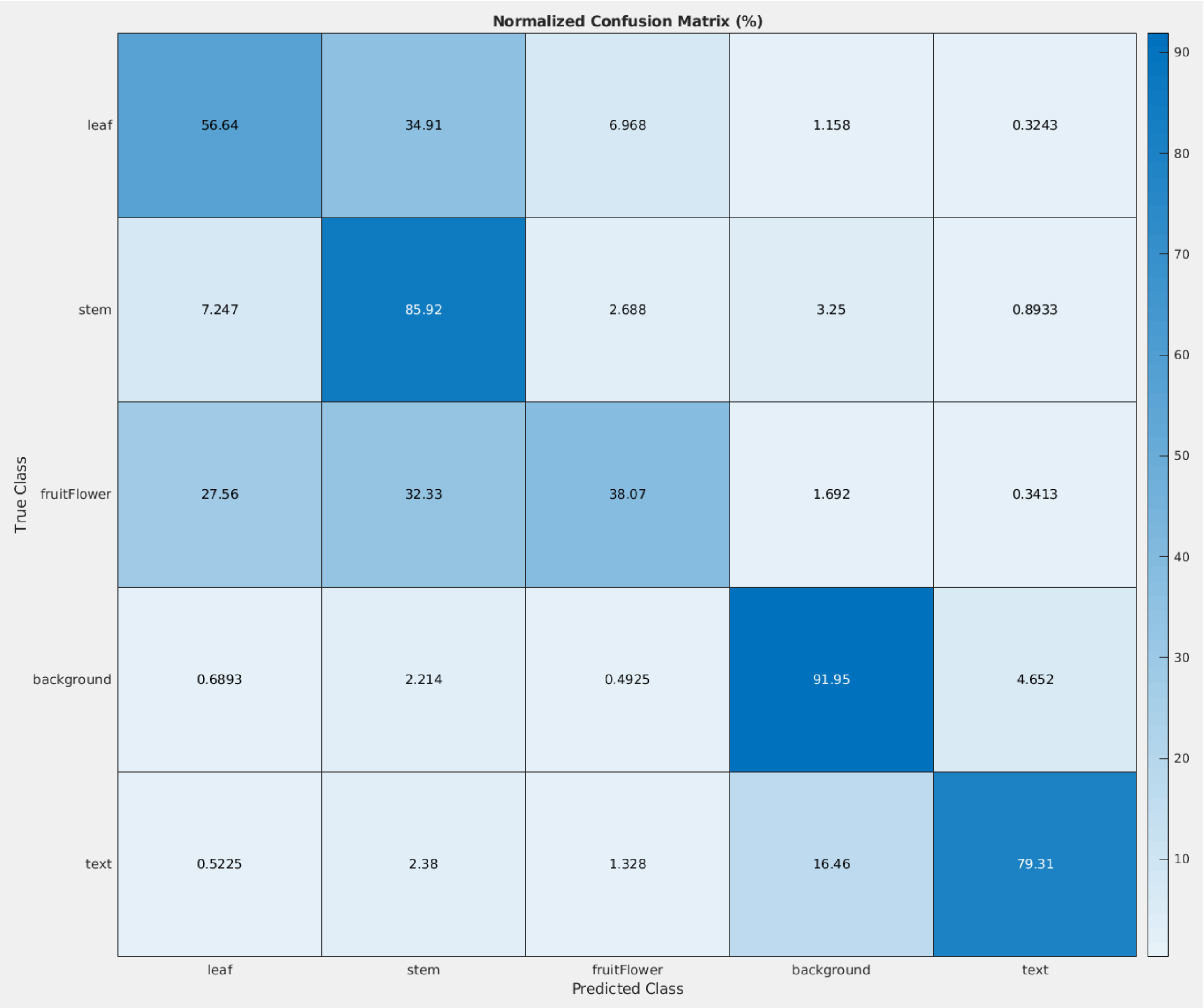
The three confusion matrices below illustrate the segmentation class prediction accuracy for a 74-image testing dataset after training completed. In this case, high-resolution images are 21 megapixels and low-resolution images are 2.4 megapixels. The LM-CNN performed best with high-resolution images. Notably, the high-resolution prediction accuracy for the 'leaf' class was 63%.

Confusion Matrix – High- and Low-Resolution

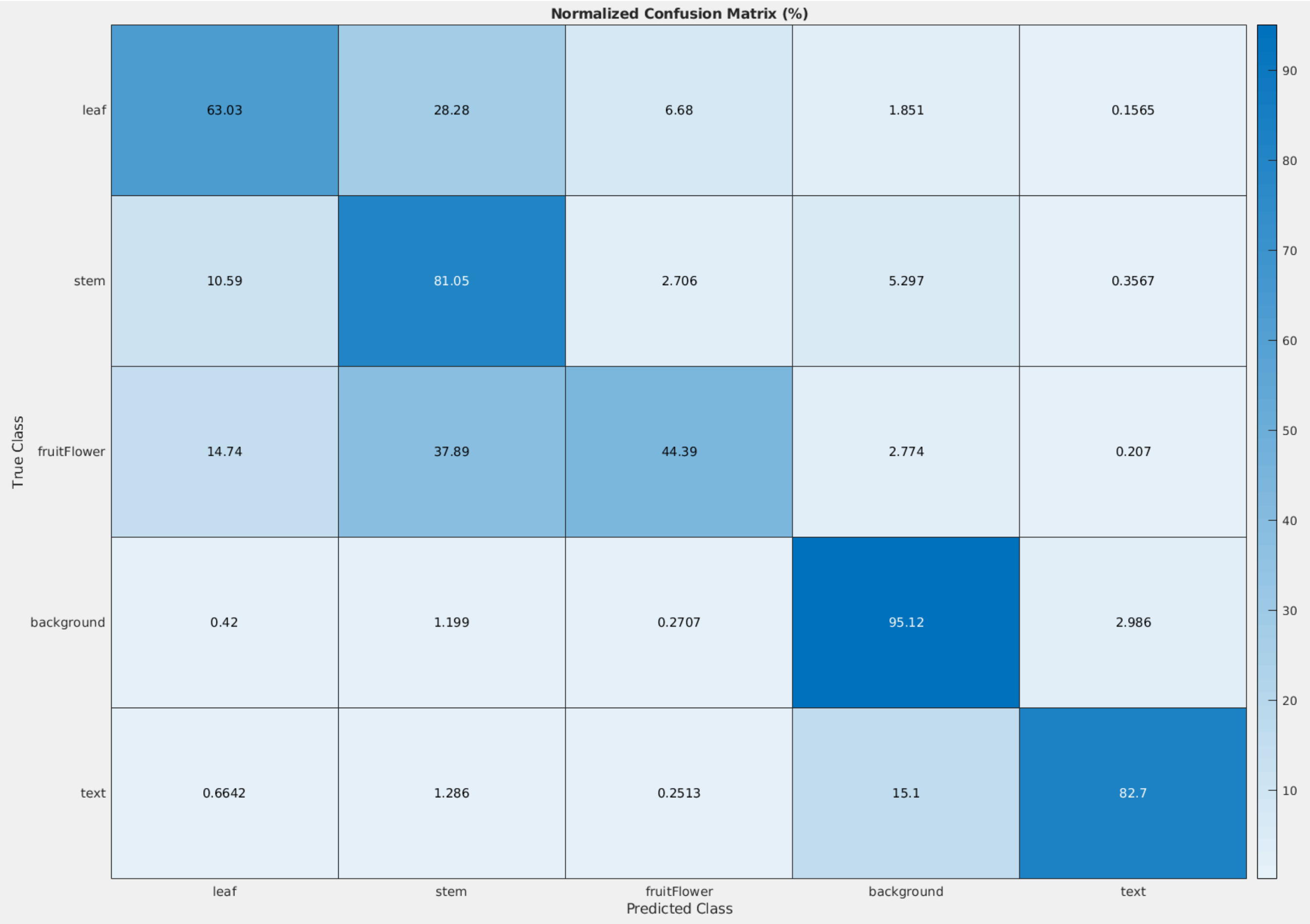


True Class \ Predicted Class	leaf	stem	fruitFlower	background	text
leaf	62.3	29.04	6.713	1.772	0.1757
stem	10.22	81.58	2.704	5.073	0.4153
fruitFlower	16.24	37.24	43.65	2.647	0.2227
background	0.4475	1.302	0.2934	94.8	3.156
text	0.6509	1.388	0.3525	15.23	82.38

Confusion Matrix – Low-Resolution



Confusion Matrix – High-Resolution



True Class \ Predicted Class	leaf	stem	fruitFlower	background	text
leaf	63.03	28.28	6.68	1.851	0.1565
stem	10.59	81.05	2.706	5.297	0.3567
fruitFlower	14.74	37.89	44.39	2.774	0.207
background	0.42	1.199	0.2707	95.12	2.986
text	0.6642	1.286	0.2513	15.1	82.7

Accuracy by Segmentation Class

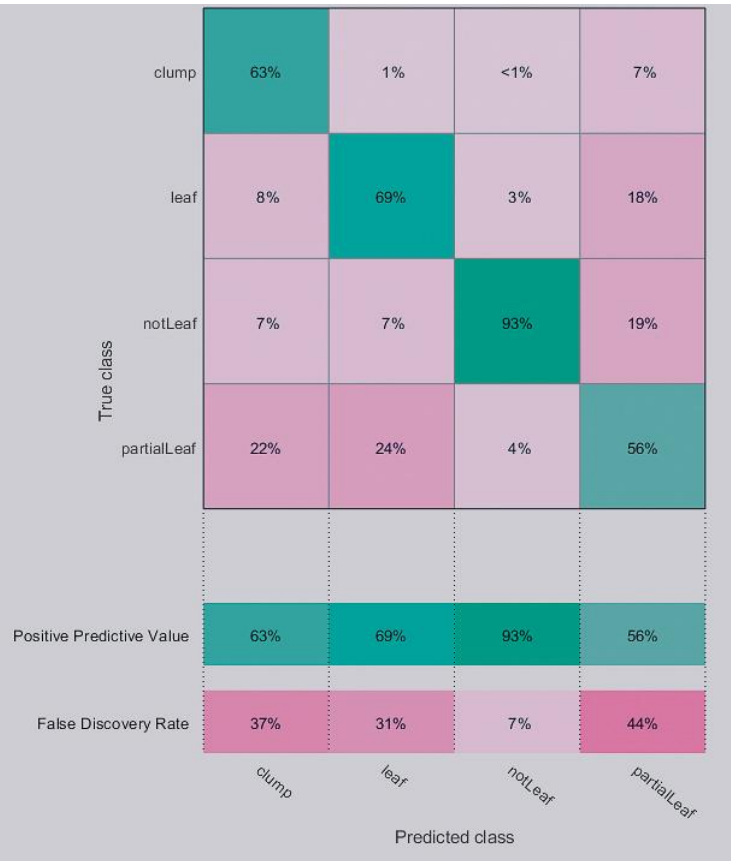
	Low-Resolution		High-Resolution	
Class	**Accuracy**	**IoU**	**Accuracy**	**IoU**
Leaf	0.566	0.486	0.630	0.561
Stem	0.859	0.260	0.811	0.305
Fruit/Flower	0.380	0.214	0.443	0.270
Background	0.919	0.915	0.951	0.946
Text	0.793	0.214	0.823	0.322
Mean	0.704	0.418	0.733	0.480

	Global Accuracy	**Mean Accuracy**	**Mean IoU**	**Weighted IoU**
High-Resolution	0.921	0.733	0.480	0.892
Low-Resolution	0.886	0.704	0.418	0.853

Support Vector Machine (SVM)

We used MATLAB’s Classification Learner App to create a support vector machine algorithm to bin each LCM into one of four categories: leaf, partial leaf, clump, and reject. For ground-truthing, we manually sorted 197,000 binary LCMs. LCMs with near-perfect leaf outlines and minimal petiole presence were binned into the ‘leaf’ class. The ‘partial leaf’ class contained LCMs with excessive petiole presence, inclusion of stems and sticks, and overlapping with no more than one other leaf. The ‘clump’ class contained LCMs with multiple overlapping leaves. The ‘reject’ class contained all other LCMs. Eleven parameters were calculated for each LCM and became a row in a table used to train the SVM. LeafMachine uses an AdaBoost decision tree support vector machine that was trained with 20% holdout validation, 50 splits, 100 learners, and a 0.1 learning rate, yielding an algorithm with an overall accuracy of 79.9%.

SVM - Confusion Matrix



True class	clump	leaf	notLeaf	partialLeaf
clump	63%	1%	<1%	7%
leaf	8%	69%	3%	18%
notLeaf	7%	7%	93%	19%
partialLeaf	22%	24%	4%	56%
Positive Predictive Value	63%	69%	93%	56%
False Discovery Rate	37%	31%	7%	44%

Predicted class

Filenames

LeafMachine begins with the images file, iterates row by row, matches the coreid number to the corresponding row in the occurrences file to gather associated information like family name and catalog number. Importantly, LeafMachine uses the family name to make predictions during the SVM step – if the family name is available. When downloading from “images” and “occurrences” files directly using option 2A), the family name is attached to the filename. If processing local images with option 2B), then LeafMachine will parse the existing filename at each underscore looking for a valid family name. If a family name is found, then it will be used in the SVM process. This can be verified by looking for a family name in the Command Window output as LeafMachine processes a local batch; if the family name is "NA," then LeafMachine could not find a valid family name. Without a valid family name, the SVM process will use 10 variables rather than all 11 variables.

The catalog number typically corresponds to the barcode on the specimen. LeafMachine finds the catalog number in two ways: (1) by taking the catalog number directly from the corresponding column in the occurrences file (2) by parsing the catalog number from the end of the URL in the images file.

While LeafMachine is running, both versions can be seen in the Command Window output log. The second method was tailored to a minority of herbaria that do not have alphabetical letters at the beginning of their catalog numbers. If a catalog number begins with zeroes, csv files have a tendency of cutting out the leading zeros. This behavior can be seen for COLO specimens. In an effort to match the catalog number to the barcode, we attempt to read the URL string and extract the catalog number from it.

For example, the catalog number SWMT00001 would follow method one, but some catalog numbers like 00008812 for a COLO specimen would get truncated to 8812 and require the second method. We chose not to add leading zeros back, since the number of characters in a catalog number vary by herbarium.

Processing Time

The three most time intensive steps are segmentation, SVM, and exporting the overlay image. The segmentation time will remain relatively constant for images of a given resolution and primarily depends on computational power – powerful GPUs will yield faster times. However, the SVM processing time is proportional to the number of LCM binary objects that need to be analyzed, which can vary greatly between specimens. Low-resolution images typically have less than 50 LCMs, while high-resolution images may have as many as 1,000 LCMs. This step is typically the lengthiest step in the LeafMachine algorithm set. In order to maintain the original image quality, exporting the image overlay utilizes a custom algorithm rather than using any built-in MATLAB functions. This algorithm takes the skeleton perimeter boundary of all SVM-selected LCMs, expands the thickness of the perimeter boundary proportional to the image resolution, and replaces the RGB values in the original image with the thickened perimeter boundaries.

Darwin Core Standard

Darwin Core is a standardized method for storing data that is used by many herbaria. LeafMachine is designed to use Darwin Core files. Example formats can be seen in the LeafMachine “Demo” folder. It is possible to format your own data to match Darwin Core standard spreadsheets and then use those files inputs for LeafMachine. The two file-types used by LeafMachine are the “image.csv” and “occurrences.csv” files, which reference the stored image URLs and the specimen record data, respectively.

For more detailed information about Darwin Core, please follow this link and read more: https://dwc.tdwg.org/