Automatic Tree Tagger User Manual

Author: Daniel Butt, NTP 2022
dbutt7@uwo.ca

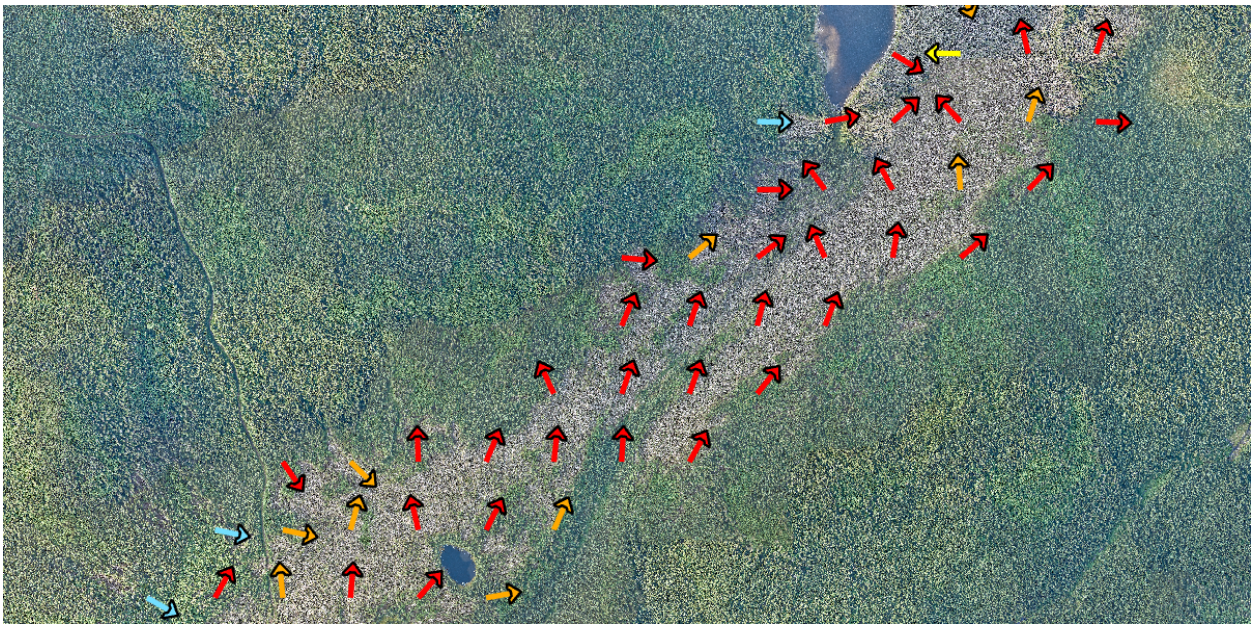


Table Of Contents

1. Overview

This software allows for the automatic detection, "tagging" (fitting a line), and direction analysis of fallen trees in large scale aerial imagery. It was designed for the Northern Tornadoes Project, mainly for analysis of fallen trees due to severe storms such as a Tornadoes. The software leverages machine learning algorithms primarily written in Python code, however it is intended that a user will utilize the software through an ArcGIS Pro add-in built in C# with the ArcGIS SDK. Currently, the software only supports analysis of image files (Image file types supported by OpenCV) which have been georeferenced to have both coordinates and an image pixel to coordinate scale.

1.1. Recommended System Specs

CPU: i5/Ryzen 5 - i7/Ryzen 7 made after 2017, but any half decent modern cpu should work

Ram: $\geqslant$ 16GB, but at least 8GB should work. Note: main test system had 64GB

GPU: NVIDIA, gtx 1060 6GB or better. Note: gpu's tested, 1070, p4000, 3070, 3080

Storage: nvme ssd or at least ssd recommended, but a standard hard drive will work

1.2. Installation

In order to install and setup the software please refer to the installation guide on the GitHub, Installation Guide

2. User Instructions

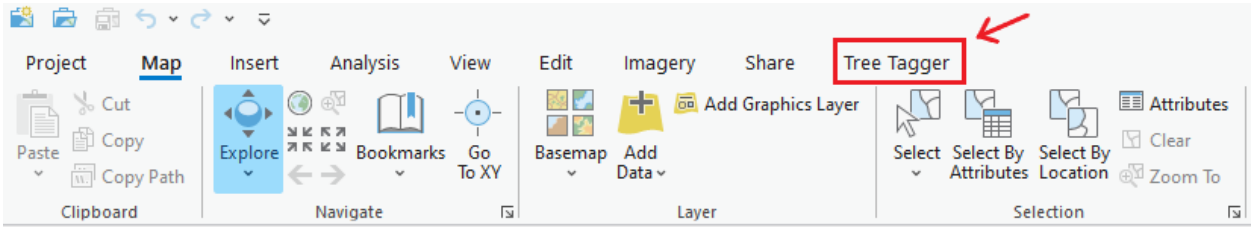
2.1. Important Notes

Primarily the machine learning models were trained and tested on 5cm imagery and it is not recommended to run the model on lower quality imagery (10cm, 20cm, etc). If you attempt to run the software on non-5cm imagery, a copy of the images will automatically be re-scaled to match 5cm imagery as the models expect a tree to look a certain way and be a certain size. *Using non-5cm imagery may also result in runtime errors.*

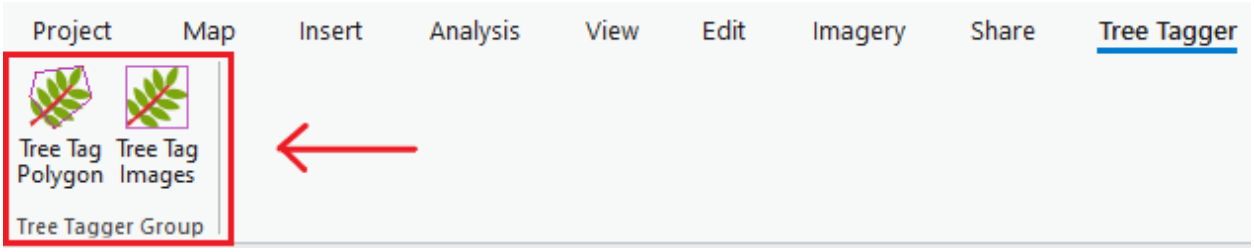
In order for the software to run optimally (speed of analysis), it is strongly recommended to run the software using a Nvidia GPU with Nvidia Cuda to speed up the machine learning model's prediction time (see installation guide). Also, having your imagery on a fast drive (nvme ssd > ssd > hard drive) will greatly improve image loading times.

2.2. Basic Usage

Once installed correctly, you should now see an additional tab in the top ribbon of any ArcGIS Pro project labelled "Tree Tagger".

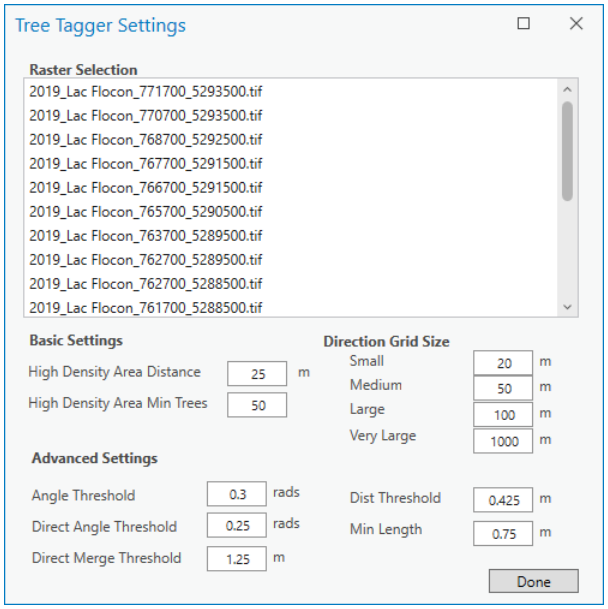


Selecting this tab will display two options for you to select, either "Tree Tag Polygon" or "Tree Tag Image".



If you know the specific image(s) you wish to process, or you plan to process the entire event, it it recommended that you use Tree Tag Images. If you only want to process a specific region, for example the damage track, you can instead select a polygon region using the Tree Tag Polygon option.

Once you have finished selecting a polygon region, or if you select Tree Tag Images, the Tree Tagger settings menu will open.



From here you can select the specific images you wish to process using the "Raster Selection" menu and change any other settings required (see the various settings sections). If you selected a polygon region, the images will be selected for you automatically and the Raster Selection menu will be disabled.

Tips:

- Clicking on an image in ArcGIS will tell you the image's name
- You can use shortcuts for selecting multiple images in the Raster Selection menu
    * Hold ctrl to select multiple images
    * Select an image, then hold shift and select another image to select a range
    * Press ctrl + A to select all images

Once you have finished selecting images and changing settings, press "Done" to start the analysis. A command prompt window should appear, displaying progress information. (note: you must select an image or polygon for the software to continue)

While the program is running, you most likely won't be able to use ArcGIS and it is not recommended to perform any other computationally intensive tasks in the background. Do not attempt to close the command prompt window (minimize is fine) or the Tree Tag settings window while the program is running. The analysis this program performs is very computationally intensive and may take a long time to run depending on your computer specifications, how many images were selected, and the size of the images.

In the event the program crashes, you should receive an error message describing the error and providing potential solutions (see trouble shooting section). Following a crash, if analysis had already been performed on the same project, ArcGIS might load the same results from the previous analysis a second time.

Once the analysis has completed, both the command prompt and Tree Tag settings window will close and ArcGIS should automatically load the results. You should now see that 4 new raster layers have been created, labelled "Lines", "Directions", "Refined_Lines", and "Regions". Each file should also include a time stamp in the name so that the results from different analyses can be differentiated. The results are saved as shape files (and csv in the case of directions) and can be located in a created folder labelled "Tree Tagger" in your project's directory.

2.2.1. Lines

The Lines shape file contains the coordinates of the end points of all lines detected from the original analysis of the imagery. Generally it is recommended to turn these off except for comparison with "Refined Lines".

2.2.2. Directions

The Directions shape file (or csv) contains the end points of the averaged direction vectors for each grid size (small, med, large, very large), drawn from first end point to second. Make sure you select arrow symbols which are drawn from first to last end point in ArcGIS or all the directions will be backwards. The shape file also contains a consistency and scale field for each vector. The scale field tells you which scale the vector belongs to (1:small - 4:very large). The consistency field gives the vector a score of 1-4 describing how consistent the directions of the trees are in the grid, 1 meaning very inconsistent and 4 meaning very consistent (see Tree Direction Averaging Calculation section for more info). It is suggested to use a discrete colour scheme to change the vectors based on their consistency value. Also it is recommended to filter the vector sizes using scale based display filters.

2.2.3. Refined_Lines

The Refined_Lines shape file contains the end points of all refined lines. The lines were refined using the analyzed line directions and removal of inconclusive lines (lines which are either a detected false positive or whose direction couldn't be extracted).

2.2.4. Regions

The Regions shape file contains the points of polygons which enclose detected high density areas (areas which have a high number of fallen trees, see Basic Setting section for more info).

2.3. Basic Settings

The basic settings menu includes two options focused on high density area detection, "High Density Area Distance" and "High Density Area Min Trees". The software detects high density areas by clustering the end points of refined lines using the DBSCAN algorithm.

2.3.1. High Density Area Distance

Any endpoints which exceed the high density area distance from each other will not initially be considered in the same cluster by the DBSCAN algorithm. Generally, it is not recommended to use a value smaller than the length of the trees in your imagery as the algorithm will have a hard time generating clusters. The default value of 25m was chosen as it is around 3-4x the length of the trees tested.

TLDR: Larger distance value = more/larger clusters.

2.3.2. High Density Area Min Trees

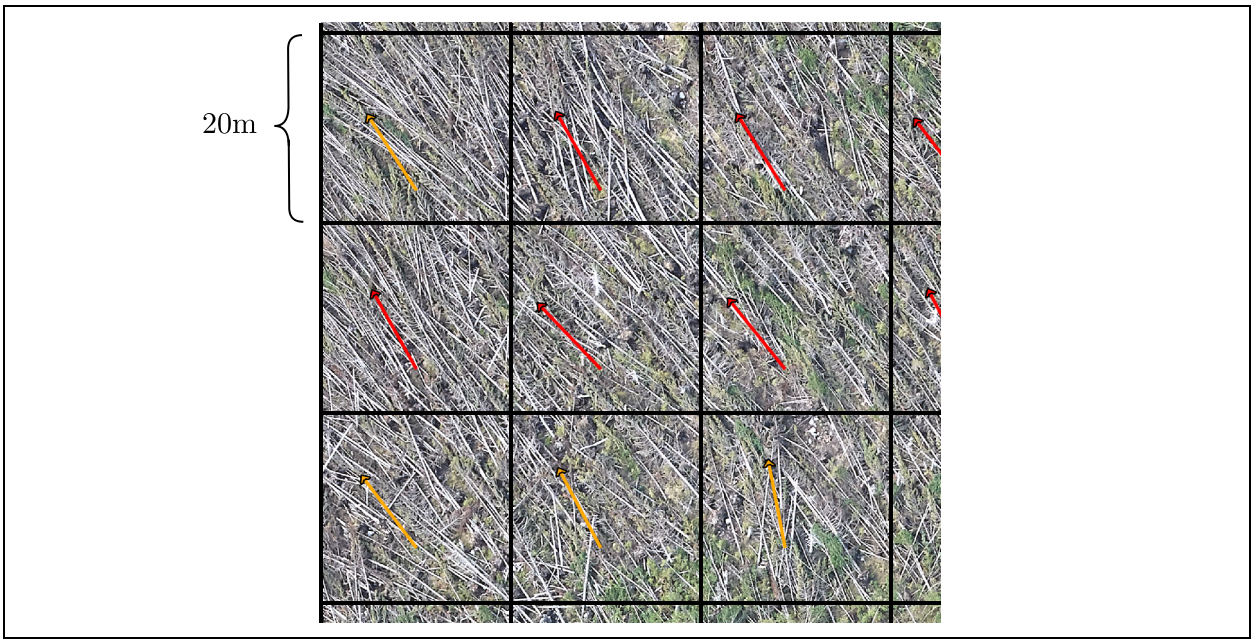
The high density area min trees refers to the minimum number for trees within the high density area distance threshold in order to produce a cluster. Basically how dense does the area need to be in order to be considered a high density area.

TLDR: Larger min trees value = fewer but more dense clusters

2.4. Direction Grid Size Settings

The direction grid size settings refer to the sizes of the grid squares in which the individual tree directions will be averaged. For each size, the analyzed images are split into a grid of the specified size. Next, all the tagged trees which have predicted directions, will be sorted into the grid squares based on the location of their line's mid-point. From here a weighted average of all the tree directions is calculated (see Tree Direction Averaging Calculation sections) for each grid square.

The default grid sizes are small: 20m x 20m, medium: 50m x 50m, large: 100m x 100m and very large: 1000m x 1000m. Generally it is not recommend to set a size smaller than 20m as there may not be enough trees to average per grid square, leading to inaccurate results. Moreover, it is not recommended to exceed a size larger then that of your imagery (say your images were 1km x 1km and you picked a size of 2km x 2km) as the direction vectors may not scale appropriately. The default values were selected as they generally work for average size events, If your event is very large or very small, you should adjust the sizes accordingly.



2.5. Advanced Settings

The advanced settings focus on parameters for joining lines together where multiple lines may have been detected over the same tree. It is not recommended that you adjust these values unless you really know what you are doing, as you may get very strange results.

The not joined lines come as the output of an edge-based line detection algorithm which extrapolates lines from the segmentation mask (only the pixels of the trees are left, everything else should be blacked out) of the trees generated from a Fully Convolutional Neural Network (FCN) (see FCN section).

The current algorithm for join lines together is as follows. First the total image area containing all the lines is split into a grid where each grid square is 256 x 256px. Next, lines are sorted into the grid square containing them, if the lines intersect multiple grid squares then the lines are added to every square they're contained in. Then, each line is compared to all other lines contained with in a 3x3 grid surrounding the line. If a pair of lines meet the criteria specified below, the lines are joined into one line which is then added to the grid and the process of checking pairs of lines continues.

2.5.1. Criteria for joining lines

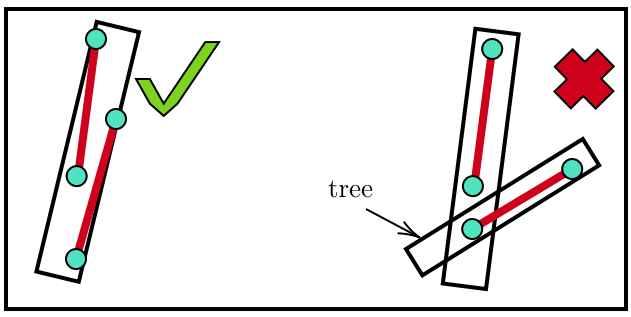
All lines are sorted such at that the endpoint with a lower x coordinate comes first. The orientation (more or less angle) of the lines is then calculated using

$\theta = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1 + \epsilon}\right)$, $\epsilon = 0.000001$ to prevent division by zero.

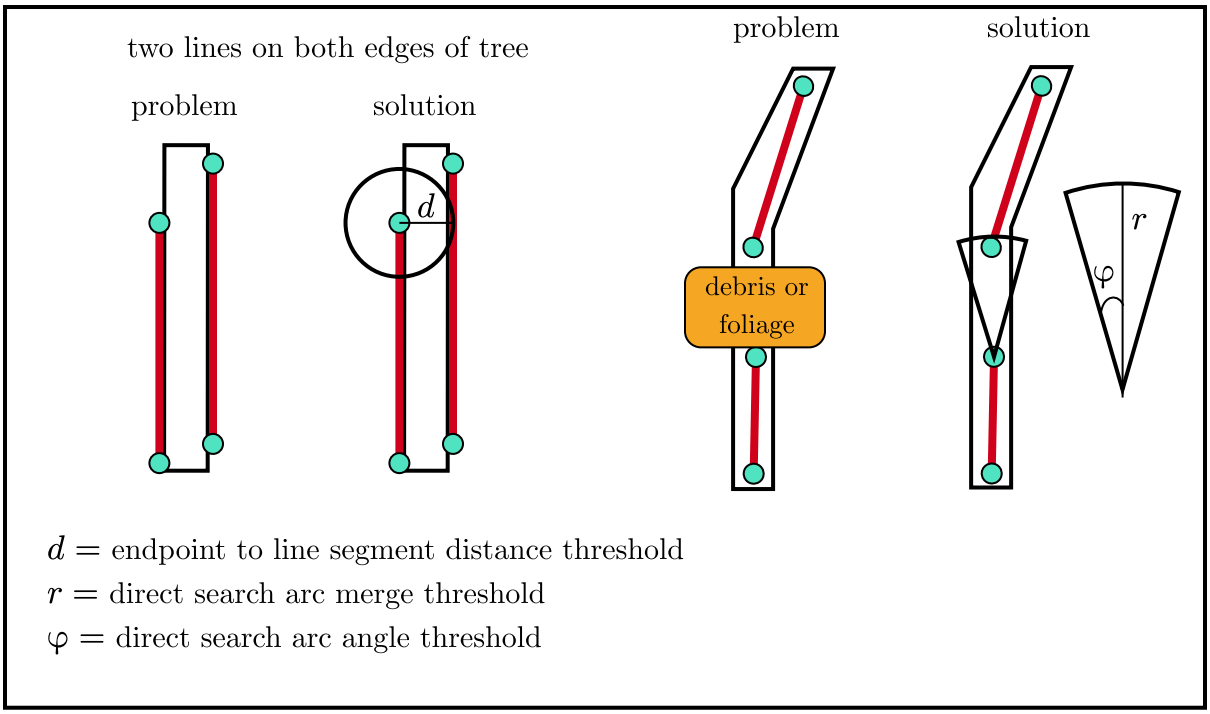
This ensures $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$, $(-90°,\ 90°)$.

The first criteria is that the pair of lines must have an angle difference of less than the "Angle Threshold". Mathematically, this is a little more complicated since for example, lines with orientations of $89°$ and $-89°$ looks very similar. So, the following calculation is used,

Angle Threshold $> \min[\ |\theta_2 - \theta_1|,\ |(\theta_2\ -\ \pi * \mathrm{sgn}(\theta_2)) - \theta_1|\ ]$
By default the angle threshold is 0.3 rad ~17°



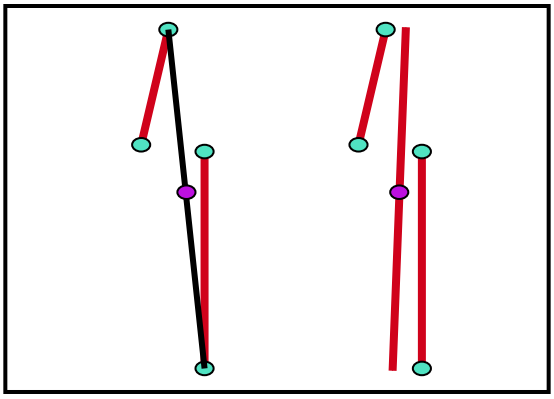
Next, the lines must satisfy one of the following two criteria. Either the endpoints of one line fall into the "search arc" of the other line. Or the distance between the endpoint of one line and the opposite line segment must be less than a threshold.



Finally, after all the lines have been checked and joined, any lines smaller than a "min length" are treated as false positives and deleted.

2.5.2. Joining a pair of lines

Once a pair of lines meet the criteria to be join, a new line is created from the pair of farthest line endpoints (one endpoint per line). Then this line is rotated about its midpoint to have an orientation equal to the weighted average orientation of the original two lines, weighted to be closer to the longer line's orientation.



3. Trouble Shooting

Have you tried turning it off and then on again? :)

In general if the Tree Tagger software encounters an error, you should be given an error message describing the error and potential problems/solutions. If the error message doesn't make any sense, or isn't really human readable, then most likely the error is with ArcGIS and can be solved by restarting ArcGIS.

3.1. ArcGISPro-py3 Error

A common error which occurs is that ArcGIS will not load the specific virtual python environment which the Tree Tagger software requires, instead loading ArcGIS's own proprietary python environment. If this occurs you should get a message tell you to save and restart ArcGIS, as this usually fixes the problem. This is a common problem which occurs the first time you attempt to use the Tree Tagger software or after using an ArcGIS python toolkit.

3.2. Done button not working

Make sure you have selected an image to process or that your selected polygon region intersects an image that can be processed (see Overview for supported file types).

3.3. Direction vectors are all pointing the opposite direction

Most likely you have selected an ArcGIS shape file symbol which draws vectors (or lines) from last to first endpoint, instead of the required first to last endpoint. If this is not the case then a catastrophic error as occurred, please check your images files are of supported types and their coordinates and coordinate system make sense. Note, this software has not been rigorously tested with every supported coordinate system ArcGIS offers.

3.4. Crashed or won't run with no clear error message

If it won't start at all, try saving and restarting ArcGIS. If this is the first time you're using the Tree Tagger software on your specific computer, you may want to double check you have installed the software correctly (see installation guide). Otherwise, see the "Done button not working" section above. If this doesn't fix the problem you may have restart your computer. If this still doesn't fix the problem, try reinstalling Tree Tagger.

If it crashed during the analysis process and you didn't receive a helpful error message, the following is a list of potential issues,

- ArcGIS it's self has had an error and maybe even crashed, try restarting ArcGIS
- The Tree Tagger python virtual environment was not setup correctly, see installation guide
- Cuda has not been installed correctly, or your using an incompatible gpu, see installation guide
- Your images are not of supported file types, of an irregular scale/coordinate system, not accessible by Tree Tagger (in use by another program), or corrupted
- Your trying to process a very large event and your computer doesn't have enough ram,
    * try running on fewer images / a smaller section
- Your storage drive is full and Tree Tagger can't save the result files.

If non of these seem to be the issue or you're really stuck, you can send me an email at dbutt7@uwo.ca.

4. Extra Info

4.1. Tree Direction Averaging Calculation

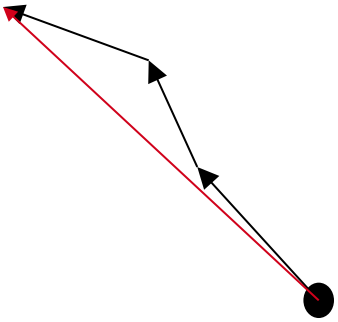
$\hat{v}$ = unit vector of detected tree in grid square
$\vec{V}_{wavg}$ = weighted average of all unit vectors in grid square

$$[\text{left},\ \text{inc},\ \text{right}]$$

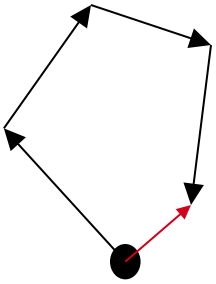
example direction prediction: $[0.71,\ 0.05,\ 0.24] \rightarrow\ w = \text{confidence} = 0.71$

$$\vec{V}_{wavg} = \frac{\sum_{i=1}^{n} \hat{v}_i w_i}{\sum_{i=1}^{n} w_i}$$

high consistency



low consistency



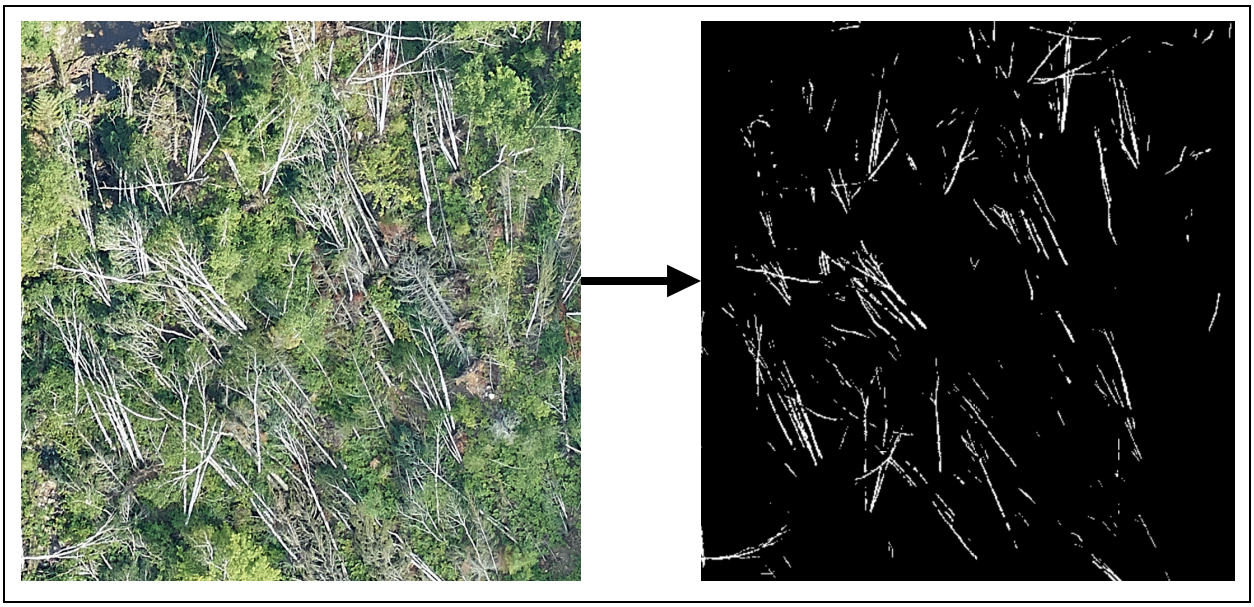
$\|\vec{V}_{wavg}\|$ = "consistency score"

- **4** consistency score $> \frac{2}{3}$
- **3** consistency score $> \frac{1}{3}$
- **2** consistency score $< \frac{1}{3}$
- **1** Only 1 detected line in grid square or if 0, extrapolated from surrounding squares

4.2. Machine Learning Models' Descriptions

4.2.1. Tree Mask Segmentation FCN

Fully Convolutional Neural Network used to extract a binary segmentation mask of the trees.



Architecture: ResNet50 → Unet
Input Size: 256 x 256px
Output Size: 128 x 128px (Bicubic interpolation to 256 x 256px)
Data Set: ~220,000, 256 x 256px images and corresponding 128 x 128px masks, post augmentation (80% training, 20% validation split)
Initial Weights: ImageNet
Optimizer: Adam
Batch Size: 16
Epochs: 30
Loss Function: Weighted Binary Cross-Entropy + Focal-Tversky,

The loss function was chosen based off of two important factors specific to the data set.

1. There are substantially more background pixels than tree pixels $\sim 50:1$ ratio
   - Thus, weighted binary cross-entropy was used to weight tree pixels as more important than background pixels by a 50:1 ratio
2. It is better to have more false positives than false negatives since false negatives will result in trees being lost, however false positives will add noise which can be removed using image filtering techniques such as Non-Local Means Denoising.
   - Thus, Tversky was used to weight false negatives as more detrimental than false positive. This also has the benefits of IOU for imbalanced data sets.

$$\textbf{WBCE}(Y,\ \widehat{Y}) = \text{Weighted BCE} = -\frac{1}{N}\sum_{i=0}^{N}[w_1 \cdot Y_i \cdot \ln(\widehat{Y}_i) + w_0 \cdot (1 - Y_i)(1 - \ln(\widehat{Y}_i))]$$

$$\textbf{Tversky}(Y,\ \widehat{Y}) = \frac{T_P}{T_P + \alpha F_P + \beta F_N}$$

$$\textbf{Loss}(Y,\ \widehat{Y}) = (1 - \textbf{WBCE}(Y,\ \widehat{Y}))\ +\ (1 - \textbf{Tversky}(Y,\ \widehat{Y}))^{\gamma}$$

$$w_1 = 1,\ w_0 = 0.02,\ \alpha = 0.5,\ \beta = 1,\ \gamma = 0.75$$

4.2.2. Tree Direction CNN

Convolutional Neural Network used to predict the direction of horizontal trees.

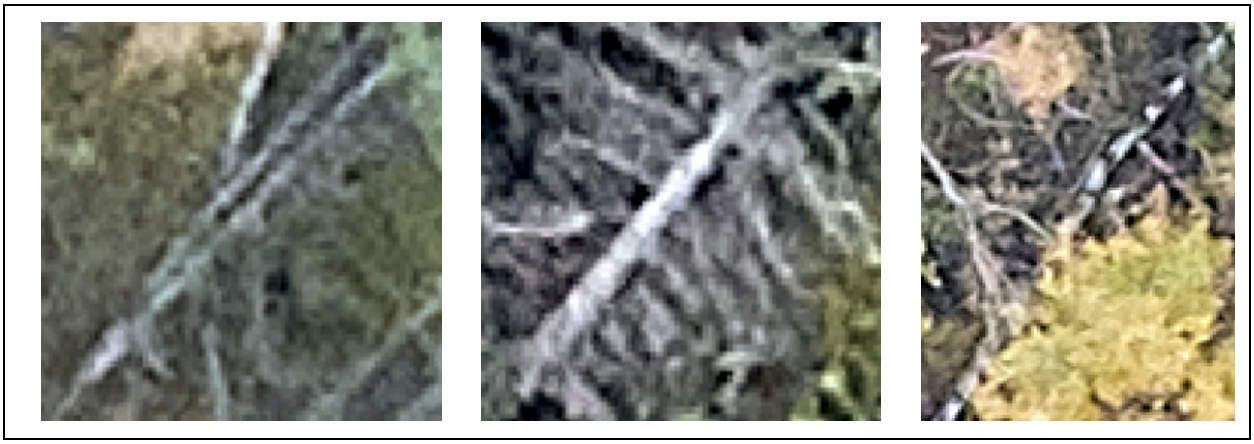
Architecture: ResNet50→Dense 128, relu→Dense 64, relu→Dense 32, relu→Dense 3, softmax
Input Size: 128 x 256px (h x w)
Output Size: [left, inconclusive/false positive, right]
Data Set: ~40,000, 128 x 256px images and corresponding labels,
post augmentation (80% training, 20% validation split)
Initial Weights: ImageNet
Optimizer: Adam
Batch Size: 16
Epochs: 9
Loss Function: Categorical Cross-Entropy

Since we already have a line over the tree and know the orientation (angle) of the line, all we really need to know is whether the line is pointing in one direction or the other. As a result, the problem of extracting direction can be significantly simplified to whether the tree is pointing to the left or right. As an added bonus, we can also filter out false positives by adding an inconclusive category since generally, you can't extract direction from debris.

4.3. Tree Detection/Tagging Problems

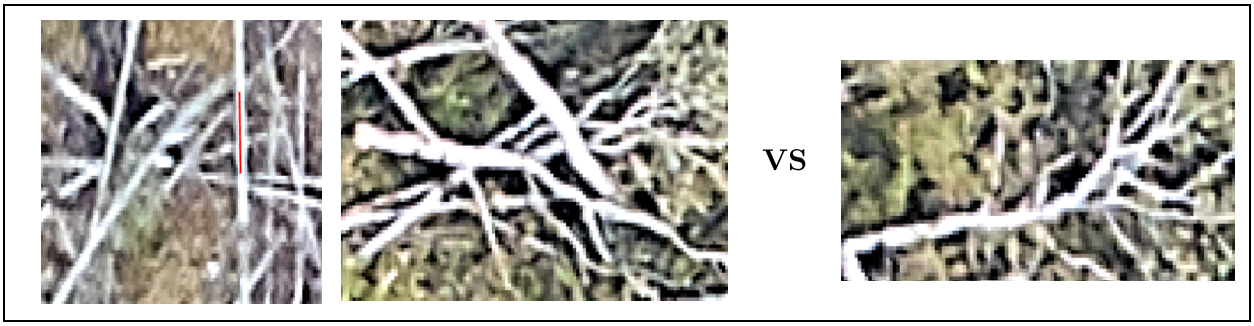
Generally, it was found that if the machine learning model failed to detect a tree, then the tree will fall into one or multiple of the following categories.

1. The tree is an off colour, is in heavy shadow, or is otherwise obstructed



This is primarily due to the fact that the data used to train the machine learning model was obtained by using image filtering/processing techniques, most notability filtering by the most common colours of the trees. This was done instead of manually tagging trees as it was substantially faster (a matter of days vs months) and we had limited time. As a result the model wasn't trained to pick up less common trees like the ones shown above. The only real solution to this issue would be to retrain the model on manually tagged trees.

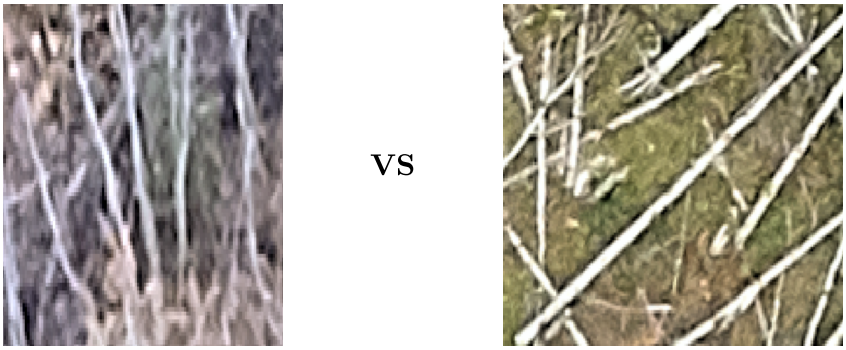
2. The tree is in a cluster of trees and or branches



The model was trained to extract the main tree trunks and to avoid branches, since ultimately the goal was to fit lines to the tree trunks and not every individual branch. However, this poses the problem of if there is a dense cluster of trees which have fallen all over each other, or a mixture of trees and branches, how can the model tell what is a tree trunk and what is a branch? Given that there are substantially more background pixels than tree pixels (regardless of the loss function discussed in the FCN section), the model is incentivised in these situations to avoid branches and say that everything must be

background. As of the time writing this manual, we have not come up with a solution to this problem except that maybe manually tagging trees would help.

3. The tree is blurred or of low quality



The model was trained on over 220,000 5cm 256 x 256px images. As a result, it expects a tree to look a certain way and be of a certain size. Giving the model lower quality ($>$ 10cm) imagery will result in the model missing a lot of trees. Nevertheless, the intention is to use this model on high quality imagery of $\leqslant$ 5cm and as a result this is not a very common issue.