AN OPEN-SOURCE CANOPY CLASSIFICATION SYSTEM USING MACHINE-LEARNING TECHNIQUES WITHIN A PYTHON FRAMEWORK

Owen Smith [1], Huidae Cho [2]*

Institute for Environmental and Spatial Analysis, University of North Georgia, Oakwood, GA 30566, USA
[1] ocsmit7654@ung.edu, [2] huidae.cho@ung.edu

KEY WORDS: Canopy Classification, Remote Sensing Analysis, Machine Learning, Open Source, Python Module

ABSTRACT:

Studying deforestation has been an important topic in forestry research. Especially, canopy classification using remotely sensed data plays an essential role in monitoring tree canopy on a large scale. As remote sensing technologies advance, the quality and resolution of satellite imagery have significantly improved. Oftentimes, leveraging high-resolution imagery such as the National Agriculture Imagery Program (NAIP) imagery requires proprietary software. However, the lack of insight into the inner workings of such software and the inability of modifying its code lead many researchers towards open-source solutions. In this research, we introduce CanoClass, an open-source cross-platform canopy classification system written in Python. CanoClass utilizes the Random Forest and Extra Trees algorithms provided by scikit-learn to classify canopy using remote sensing imagery. Based on our benchmark tests, this new canopy classification system was 283 % to 464 % faster than commercial Feature Analyst, but it produced comparable results with a similarity of 87.56 % to 87.62 %.

1. INTRODUCTION

Forested areas play an integral role in the maintenance of both local and global environments. They are the bulk of Earth's carbon sequestration for mitigating anthropogenic processes (Bala et al., 2007; Platz, 2015; Reed and Kaye, 2020; Shen et al., 2020), provide natural erosion and runoff control for flooding events, which have been growing in frequency because of climate change (Benito et al., 2003; Sriwongsitanon and Taesombat, 2011), and can offer respite for urban heat islands (Wong and Yu, 2005; Rani et al., 2018; Bosch et al., 2020). The effective creation of canopy data is of utmost importance to analyze the aforementioned processes in addition to forest patterns such as disturbance, mortality, and the societal and economic effects forests can provide (Senf et al., 2018; Senf and Seidl, 2020). Deforestation monitoring is an essential part of maintaining any environment as the loss of forested lands leads to increased carbon dioxide being placed into the atmosphere while simultaneously eliminating carbon storage (Bala et al., 2007). At smaller scales, deforestation leads to increased runoff rates and subsequently increased erosion, especially in areas where no plant reclamation is initiated (Benito et al., 2003). As improvements are made in the fields of geospatial science and remote sensing, an increasing emphasis is put on accurate forest canopy detection, among other ecological factors, for the purpose of monitoring and predicting canopy change (Franklin, 2001). However, monitoring forest ecosystems accurately to mitigate these effects on a large scale can be a time-consuming and difficult process to complete (Basu et al., 2015) and there can be many inhibiting factors such as access to high resolution imagery data and access to software capable of processing the amount of data required. Subsequently, with the increase of high spatial resolution data available both publicly and commercially, a need arises for implementations capable of reproducible and efficient classification schemes designed specifically for tree canopy detection.

In this research, we developed the CanoClass Python module for canopy classification built on top of scikit-learn (Pedregosa et al., 2011) and the Geospatial Data Abstraction Library (GDAL) (GDAL/OGR Contributors, 2020). Dallaqua et al. (2018) have used scikit-learn for deforestation monitoring in which a committee system was developed. CanoClass is built around the classification of the National Agricultural Imagery Program (NAIP) imagery and is capable of classifying both 1-m and the submeter resolution imagery made available in the 2019 iterations of the NAIP program leading to increased accuracies (USDA, 2020). Section 2 introduces our Python module and elaborates on how it works. Sections 3 and 4, respectively, present data and discuss results for a case study. Finally, we conclude our research in Section 5.

*Corresponding author

2. METHODS

2.1 Challenges in Canopy Classification

In this study, we used remote sensing data to classify canopy. However, as the resolution of remote sensing data increases and the extent of the study area grows, classifying geospatial data as canopy versus non-canopy becomes computationally challenging. Challenges faced in classifying canopy include the misclassification of water as canopy, noisy or cluttered outputs in higher resolution data sets, and linear artifact creation along data set edges. Furthermore, as data sets grow, so does the computational time required to process them. For example, as we will discuss later, we used 1-m NAIP imagery for our case study. However, since a single 1-m NAIP raster tile contains approximately 50 million individual cells, processing multiple NAIP tiles for a large study area can lead to considerable computational times if the workflow is not optimized. To address these issues, we employed machine-learning techniques where we first compute vegetation indices using multi-band remote sensing data, and apply classification and post-processing algorithms designed specifically for canopy classification problems.

2.2 Vegetation Indices

Vegetation indices are extensively used when one attempts to separate vegetation from other types of land cover. These indices typically use the near-infrared (NIR) band in their equations as the wavelength of 0.75 µm to 0.8 µm in this band is absorbed by photo-synthetically active vegetation and reflected by bodies of water and impervious surfaces (Tucker, 1979). To account for atmospheric effects, we used the Atmospherically Resistant Vegetation Index (ARVI) (Kaufman and Tanre, 1992). The ARVI is written as

$$\text{ARVI} = \frac{\text{NIR} - 2 \times \text{Red} + \text{Blue}}{\text{NIR} + 2 \times \text{Red} + \text{Blue}} \tag{1}$$

where it uses the blue band in conjunction with the red and NIR bands to provide correction for atmospheric effects. The ARVI is four times less sensitive on average to atmospheric effects than the NDVI, the most widely used vegetation index (Kaufman and Tanre, 1992).

2.3 Remote Sensing Data

We used four-band (red, green, blue, and NIR) NAIP imagery for developing and testing our canopy classification framework. Previous studies that utilized open-source classification systems have not been able to achieve accuracy at a level that NAIP imagery can provide, having been limited to using only public access imagery such as Landsat 8 (Roy et al., 2014) (30-m resolution) or Moderate Resolution Imaging Spectroradiometer (MODIS) (NASA, 2020) (250-m to 1000-m resolution) (Šimić de Torres, 2016; Dallaqua et al., 2018). While NAIP imagery is on a 3-year cycle and cannot match the temporal frequency in which satellite imagery is taken, it is taken during seasons in which agriculture is growing in the United States ensuring similar characteristics between data sets (USDA, 2020). Furthermore, cloud masking will not be needed for processing as NAIP imagery's quality control removes any image that has more than 10 % cloud cover per quarter quad (QQ), rendering the need for a cloud mask negligible (USDA, 2020). The lack of cloud cover in NAIP imagery will remove issues that previous studies had with utilizing vegetation indices because the presence of clouds would render the index increasingly unreliable the more cloud cover the scene contained. It is important to note that, while NAIP was used to develop and test the canopy classification framework and most batch processing functions are developed to use NAIP imagery, the individual classification and training modules are capable of working with any remotely sensed vegetation indices developed from other satellites such as Landsat 8 and Sentinel-2 (Drusch et al., 2012).

2.4 Classification Algorithms

We considered two classification methods including the Random Forest and Extra Trees classifiers. Both are capable of utilizing multi-core processing for increased computational speed, putting them at an advantage over other algorithms.

2.4.1 Random Forest Algorithm The Random Forest (RF) algorithm is a combined multi-tree predictor built upon bootstrap aggregating where each decision node is split using a random selection of features and the most popular class is subsequently chosen based on a vote after the specified number of trees are generated (Breiman, 2001). In cases of land-cover classification, the RF algorithm is found to be as effective, if not more effective, as other popular similar ensemble algorithms such as boosting and bagging (Breiman, 2001; Gislason et al., 2006). In addition, the RF algorithm has been found to have lighter computational load than the popular AdaBoost algorithm (Freund and Schapire, 1996). The lighter computational load of the RF classifier is thanks to the random selection of variables to split decision trees. It minimizes the correlation between trees and utilizes bootstrapping, which means that a portion as opposed to the entire data set is used for each tree. However, the RF algorithm can use a considerable amount of memory because a matrix of the number of samples by the number of trees is stored in memory (Gislason et al., 2006). With memory usage in mind, the RF classifier is still an ideal algorithm for use with large data sets since it does not overfit as the algorithm follows the law of large numbers (Etemadi, 1981) and is considerably less sensitive to noise than other boosting or bagging algorithms (Breiman, 2001). RF algorithms have been used with success when classifying vegetation and land cover, and, in the case of canopy classification, are often found to outperform other algorithms (Coulston et al., 2012).

2.4.2 Extra Trees Algorithm The Extra Trees (ET) algorithm is similar to the RF algorithm in that it is a multi-tree predictor built using an ensemble of decision trees and the most popular class is chosen based on an aggregation of trees. However, in contrast to RF, the ET algorithm splits the nodes of the tree completely at random whereas the RF algorithm cuts the node at the locally optimal combination of features and split (Breiman, 2001; Geurts et al., 2006). Additionally, the ET classifier uses the entirety of the sample and not just the bootstrap to grow trees, which means that each tree is independent or uncorrelated to the last (Geurts et al., 2006). The ET algorithm has higher bias and lower variance than the standard RF algorithm because of the increased randomness of the split nodes (Geurts et al., 2006). These differences lead to the ET algorithm's biggest strength, which is its computational efficiency that can be attributed to its increased randomness and simplistic approach to node splitting when compared to the RF algorithm. On average when empirically compared to the RF algorithm, the ET algorithm is computationally about three times faster than an RF algorithm applied to the same data set (Geurts et al., 2006). The computational efficiency in addition to its usefulness when classifying high-dimensional objects such as imagery makes the ET algorithm an ideal algorithm for an efficient classification system (Xu et al., 2010; Lawson et al., 2017).

2.5 CanoClass Python Module

CanoClass was developed to bridge the gaps between remotely sensed imagery, classification, and post-processing required for large data sets through the development of multi-phase processing modules. As a Python module, CanoClass is separated into two sections being NAIP classification and the classification of all other remote sensing products that offer 4-band imagery. The separation of NAIP from other remote sensing imagery is due to the differences in processing created for NAIP imagery, in particular the batch processing created for NAIP imagery to allow for its scalable application. Batch processes for imagery such as Landsat 8 and MODIS are not created chiefly because of the difference in scale between NAIP and other imagery. The extent of a single Landsat 8 image is 185 km by 180 km while that of a single NAIP image is approximately 7 km by 7 km, making the scalability of Landsat 8 less important than other remote imagery as it already encompasses such a large area in comparison. The processing of both NAIP and other imagery is shown in Figure 1 and described in detail below.

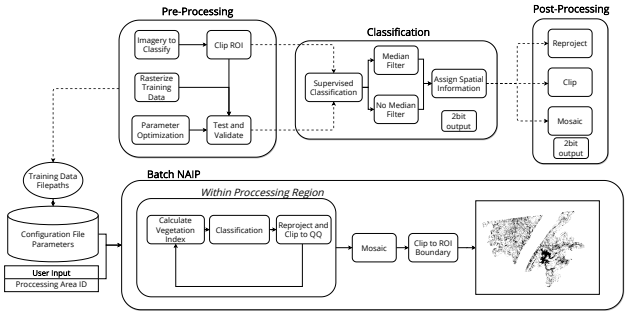


Figure 1. Workflow of CanoClass.

2.5.1 Pre-processing As the classification algorithms are to be utilized with vegetation indices, GDAL and NumPy (van der Walt et al., 2011) are used. For training data, CanoClass can convert vector training data into a raster format matching the grid and extent of the imagery that the training data is applied to. Matching extents are integral to the classification process as both the training data set and its corresponding data will be converted into matching one-dimensional arrays where training data exists. An additional region of interest (ROI) can be created to clip the imagery to a custom extent. The ROI will allow for focused analysis and additionally cut the computational processing time for larger image data sets.

CanoClass also provides parameter optimization and cross-validation before undergoing classification. Each uses a one-third data split that separates a third of the data to be used as testing data. Cross-validation allows the end user to receive the estimator performance and accuracy of their training data set while parameter optimization utilizes a generated random parameter matrix to compute aggregated cross-validation scores with computational time to return the optimal parameters to use without large sacrifices to other aspects of classification. Both features allow for optimization of accuracy and computational time, which becomes increasingly important as data sets grow.

2.5.2 Classification The classification algorithms are implemented using scikit-learn and, to keep spatial information intact when undergoing classification, all data is read through GDAL before being converted to arrays. In an effort to reduce the size of the output classified raster, each cell in the raster is allotted 2 bits (values 0 to 3) for representing non-canopy (1), canopy (2), and no data (3). However, because of limitations in both NumPy and GDAL, it is not a true 2-bit raster file as the smallest file size that both modules can save and read a raster as is 8 bits or 1 byte. For this reason, while CanoClass results in a smaller file size than an 8-bit image, the 2-bit output raster is still read as an 8-bit image by the file system, resulting in the need for cell allotment to be enforced throughout further processing.

In addition to measures taken to preserve spatial information and reduce the file size, an optional median filter for the output data is integrated in an attempt to develop a noise-reduction system such as those available in proprietary software such as Feature Analyst (Textron Systems, 2020). The median filter was created to reduce noise in imagery and has been shown to be effective while maintaining computational efficiency (Huang et al., 1979). The median filter runs a 3-by-3 cell window over each cell in the data set and can mimic a smoothing effect on the data and reduce noise that may be prevalent in high resolution rasters such as NAIP imagery.

2.5.3 Post-processing Several post-processing functions are offered with the goal to allow classified canopy imagery to have a linear workflow available from the beginning to the end of canopy classification. All post-processing functions enforce 2-bit allotment in order to maintain small file sizes after classification. Classified imagery can be converted to user-defined projections to fit different needs and additionally allow all classified outputs to be reprojected into the same spatial reference. This process is important when multiple output tiles need to be mosaicked. Clipping is provided to allow the further division of an ROI or to remove excess boundaries in the case of linear artifacts or overlapping with other imagery. Clipping becomes integral to processing of NAIP imagery in particular as the QQ seamline file that USDA provides can be used to remove the approximately 300-m overlap on each side of a NAIP raster. Mosaicking is integrated by calling GDAL through the system to allow for the synthesis of multiple classified canopy images into one continuous raster. The single mosaicked raster can enable more complete usage of the data for cell statistics and mapping purposes.

2.5.4 Batch Processing All three sections of CanoClass are utilized in the creation of batch processing functions to enable the scalable classification of NAIP imagery. A configuration file is provided to determine input and output file paths and allow the creation of a clean folder structure for every process output. Using the methods described for pre-processing, training data can be properly converted to a raster format and subsequently tested and validated. The additional ability to optimize parameters is increasingly important for batch processing as an optimal time-accuracy split can both improve accuracy and reduce computational time across a large data set.

Using the specified ROI and NAIP QQ seamline file, the file paths of all NAIP imagery within the ROI are iterated over and the specified vegetation index can be computed. The use of the NAIP QQ seamline file in conjunction with the ROI effectively allows for only those files within the ROI's spatial extent to be processed thus saving computational time. The same iterative method is used throughout the batch process to read and write files that only fall within the ROI spatial extent.

Using the trained raster to create a predictive data set, all other rasters can be classified in the ROI. The process to do so is the same as described for classification, but it includes an extra step to integrate over the required files. The spatial reference of the raster being classified is gathered in each iteration, ensuring the classified output is saved with the appropriate spatial output and not that of either the training data or another raster.

Clipping, mosaicking, and reprojection are enacted after classification. Clipping, as previously mentioned, is important because of the overlap between NAIP tiles. Without the removal of overlapping areas, errors appear in the subsequent ROI mosaicking. Each tile is clipped to its corresponding QQ polygon and is saved in a user-defined projection. As NAIP imagery is provided in a Universal Transverse Mercator (UTM) zone projection and a NAIP data set can consist of more than one UTM zone, the user-defined projection is integral to ensure that all classified canopy outputs are within the same spatial reference. Mosaicking and clipping to the ROI boundary is enacted after all rasters are clipped and reprojected, and the final 2-bit raster is saved. Thanks to the removal of overlapping areas, a mosaicking order does not need to be established.

3. CASE STUDY

3.1 Study Area

The area of study is the state of Georgia in the United States. Georgia has an area of approximately 153,910 km^2 and has a rapidly increasing population leading to large potential changes in land features (Lo and Yang, 2002). This state contains a large variety of different land-use and land-cover types with the Appalachian mountain belt starting in northeast Georgia, increasing urban sprawl of Atlanta, farming in central and south Georgia, and the wetlands that make up the coastal areas. For our case study, we chose six physiographic districts (McCaysville Basin, Lookout Mountain, Dougherty Plain, Winder Slope, Barrier Island Sequence, and Vidalia Uplands) to measure computational times and one physiographic district (Blue Ridge Mountains) to compare classification quality with Feature Analyst. Figure 2 shows all these seven physiographic districts.

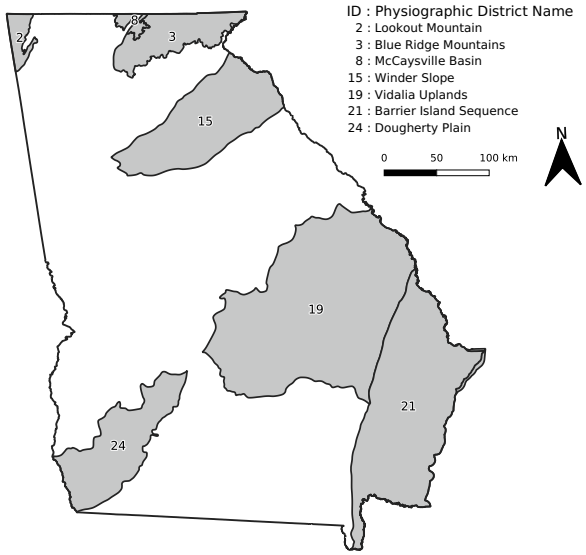


Figure 2. All physiographic districts used for the case study. For computational time comparisons, McCaysville Basin, Lookout Mountain, Dougherty Plain, Winder Slope, Barrier Island Sequence, and Vidalia Uplands districts were used. For classification quality comparisons, the Blue Ridge Mountains district was used.

3.2 Data and Training

We used 2015 NAIP imagery data for our case study. The NAIP data set used has 3913 QQ tiles each of which is 3.75° by 3.75° in size. Each tile is at a resolution of 1 m and holds all four bands offered by the USDA. The entire data set is 731 GB in size. The USDA additionally offers the accompanying seamline QQ polygon shapefile for each state containing spatial and descriptive identifying information for each QQ.

Training data was drawn using QGIS 3.10 (QGIS Development Team, 2019), a freely available open-source Geographic Information System (GIS) software. The data was drawn as a vector shapefile with the values of 1 for non-canopy and 2 for canopy. The data set created was drawn over NAIP tile `m_3408326_ne_17_1_20150915` and rasterized within CanoClass before being applied to all mentioned districts to measure computational time. It has a blend of built up, barren, pasture, and water for non-canopy labels in addition to significant amount of canopy. The training data set yielded an average cross-validation score from a 5 fold cross-validation of 0.948 indicating that the data used for training classification algorithms is accurate. The cross-validation scores were computed with functionality incorporated within the CanoClass framework.

All training and validation were done within CanoClass, and further classification and post-processing were also performed within the CanoClass batch processing environment. Table **??** shows the system specifications that we used for CanoClass processing and Cho et al. (2020) used for their Feature Analyst processing. Note that we used a Linux system while Cho et al. (2020) used two different Windows systems, which we will referred to as the Windows 1 and 2 systems below. According to a CPU benchmarking website (UserBenchmark, 2020), in terms of effective speed, the Windows system 1 is 9 % faster than the Linux system which is in turn 3 % faster than the Windows system 2. Overall, the three systems are within 12 % difference in effective speed. However, the Linux system is equipped with a hard disk drive (HDD) with 5400 rev/min, which is slower than the solid-state drives (SSDs) that the other Windows systems have.

OS	CPU	Disk Type	RAM	Swap	Software
Linux 5.4.0-40	AMD Ryzen 7 2700X @ 3.7 GHz	HDD*	32 GB	18.5 GB	CanoClass QGIS 3.10.7
Windows 10[1]	Intel Core i7-8700 @ 3.2 GHz	SSD†	32 GB	4.75 GB	Feature Analyst 5.2.0.7 ArcGIS Desktop 10.5.1
Windows 10[2]	Intel Core i7-7700 @ 3.6 GHz	SSD‡	8 GB	1.25 GB	same as above

Table 1. System specifications for our case study with CanoClass (Linux) and Feature Analyst by Cho et al. (2020) (two Windows 10s). All systems are 64 bits. 1: Windows system 1. 2: Windows system 2. *: 5400 rev/min. †: 3200 MB/s sequential read, 2400 MB/s sequential write. ‡: 550 MB/s sequential read, 535 MB/s sequential write.

3.3 Comparisons with Other Canopy Data Sets

The most notable and recent canopy creation processes utilizing Feature Analyst and NAIP imagery, and in particular within Georgia, have been those commissioned by the Georgia Forestry Commission (GFC) for the years 2015 (Bailey and Bailey, 2019) and 2009 (Cho et al., 2020) improved upon the 2015 study. We conducted two different types of comparisons between CanoClass and Feature Analyst including computational times and classification quality.

3.3.1 Computational Time Comparison For computational time comparison, it is important to accurately measure run times. Unfortunately, since there was no way to reliably calculate the computational time of creating the 2015 results that Bailey and Bailey (2019) produced, we used the study of Cho et al. (2020). Cho et al. (2020) ran each of the six districts in a single session within Feature Analyst, so the timestamps of the 2009 study outputs are a good indicator of run times. Classification times for Feature Analyst were gathered using the timestamps of the output raster files in the 2009 GFC study (Cho et al., 2020). However, there were unexpected circumstances when the Barrier Island Sequence and Vidalia Uplands districts were being processed at the end of each Feature Analyst run, and one tile and two tiles for the former and latter districts, respectively, had to be processed in separate sessions of Feature Analyst. For these three tiles in separate Feature Analyst sessions, we used the mean tile processing time of each district excluding these outliers. To be consistent, we used the same timestamp method to measure the time for CanoClass. Because

of limited computational resources and time, we only ran CanoClass for the year 2015, but not for 2009. Therefore, we compared CanoClass runs for 2015 and Feature Analyst runs for 2009.

3.3.2 Classification Quality Comparison For classification quality comparison, we used runs for the same year 2015 between CanoClass and Feature Analyst. The Blue Ridge Mountains district was used to compare the spatial results of CanoClass and Feature Analyst as training data from the 2015 iteration of the GFC study (Bailey and Bailey, 2019) in addition to the outputs were made available to us. The Blue Ridge Mountains district was classified with CanoClass utilizing only the training data created by Bailey and Bailey (2019) to ensure both data sets were created from the same training data. The 2015 data set was created utilizing Feature Analyst and completed with an estimated accuracy of 91 % (Bailey and Bailey, 2019), making an ideal proprietary data set to compare our results with. Feature Analyst operates using an Automated Feature Extraction model which uses texture, ancillary, and spectral data in conjunction with an ensemble classification method built around artificial neural network, decision trees, Bayesian learning, and imagery segmentation (O'Brien, 2003; Filchev, 2010). The difference in classification methods makes for a good comparison in classification systems between those utilized by CanoClass and those created from a proprietary setting.

For comparison, 20 QQs and their corresponding classified tiles within the Blue Ridge Mountains district were randomly chosen to eliminate bias. Quality comparison was enacted through the implementation of a moving window algorithm for raster comparisons (Costanza, 1989; Kuhnert et al., 2005; Kassawmar et al., 2019) was explored. Compared to pixel-by-pixel-based comparison methods, the moving window comparison has the advantage of detecting spatial patterns within the data, which is especially important when we compare two data sets created with two different methods and in the same resolution (Costanza, 1989; Kuhnert et al., 2005). The comparison index F_w for the moving window with the window size w can be written as

$$F_w = \frac{\sum_{s=1}^{t_w} \left[1 - \frac{\sum_{i=1}^{p} |a_{1,i} - a_{2,i}|}{2w^2} \right]_s}{t_w} \qquad (2)$$

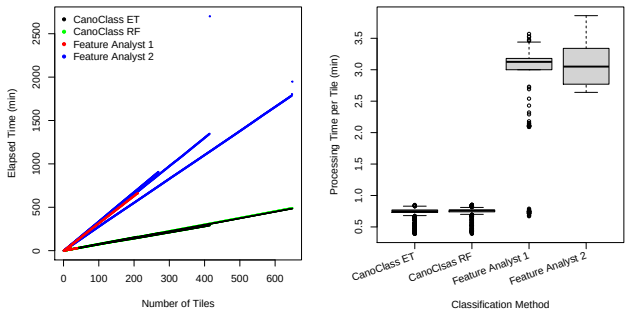
where s is the index for moving windows, t_w is the number of windows with the window size w, $a_{1,i}$ and $a_{2,i}$ represent the numbers of cells with category i in rasters 1 and 2, respectively, and p is the number of categories. The value of F_w ranges between 0 and 1 where 0 means no similarity while 1 means completely the same with the window size w. The algorithm was implemented in Python and will be added to CanoClass after more development is undertaken to increase the speed of the algorithm.

For this case study, this implementation of Eq. (2) was used outside the CanoClass environment. The F_w value was calculated for each tile with a window size of $w = 3$ to compare the similarity of both the RF and ET classified outputs to those created with Feature Analyst.

4. RESULTS AND DISCUSSION

4.1 Computational Times

Figure 3a shows elapsed times as each method processes multiple tiles for each physiographic district. Plots for Feature Analyst 1 and 2 indicate Feature Analyst runs on the Windows systems 1 and 2, respectively, as shown in Table **??** (three plots for each system). Overall, CanoClass with both ET and RF classifiers outperformed Feature Analyst except for one case where, for the Lookout Mountain district, Feature Analyst was faster than CanoClass with the RF classifier by 0.04 min. The means and standard deviations of the processing time per tile for CanoClass ET, RF, Feature Analyst 1, and 2 were 44.3 ± 4.1 s, 44.5 ± 4.1 s, 164.0 ± 52.1 s, and 182.7 ± 17.6 s, respectively. As can be seen in Figure 3b, CanoClass exhibited more consistent and faster tile processing times compared to Feature Analyst except for this one exceptional case. Thanks to this consistent performance of CanoClass, the 12 plots for CanoClass in Figure 3a (six plots each for CanoClass ET and RF) are close to each other while the six plots for Feature Analyst show varying linear trends. The more consistent performance of CanoClass indicates that this module is less sensitive to the size of the district. In contrast, the slopes of the elapsed time of Feature Analyst are different depending upon the district size. Figure 4 also shows these trends in the total computational time. CanoClass ET and RF show a linear trend with a y-intercept of close to 0 while Feature Analyst 1 shows a steeper slope and Feature Analyst 2 a linear trend with a significantly higher y-intercept implying that this software on the Windows system 2 adds a constant processing time regardless of the district size. The reliability and consistency of CanoClass's computational times is important as larger ROIs can be processed without fear of time taken growing exponentially or unpredictably.



(a) Elapsed time versus the number of tiles **(b)** Box plots of processing times per tile

Figure 3. (a) Elapsed time as the number of tiles processed grows for each of the six districts for each method. Both CanoClass ET and RF plot all the six districts in the same black and green colors, respectively. Feature Analyst 1 plots the three districts produced by the Windows system 1 in red while Feature Analyst 2 plots the other three districts by the Windows system 2 in blue. The three blue outlier points indicate that additional Feature Analyst sessions were needed to complete those districts because of unexpected circumstances such as weekly maintenance reboots. The 415th tile for the Barrier Island Sequence district, and the 648th and 649th tiles for the Vidalia Uplands district were run in separate Feature Analyst sessions. (b) The three outliers were excluded from the box plot for Feature Analyst 2.

Table **??** shows the total computational times of the three methods. CanoClass with both ET and RF classifiers was competitive with Feature Analyst. Overall, CanoClass was about 283 % to 464 % faster than Feature Analyst. For the largest physiographic district, Vidalia Uplands, classification took 1792.10 min. Classification implemented with CanoClass took 485.12 min utilizing the ET classifier and 490.01 min uti-

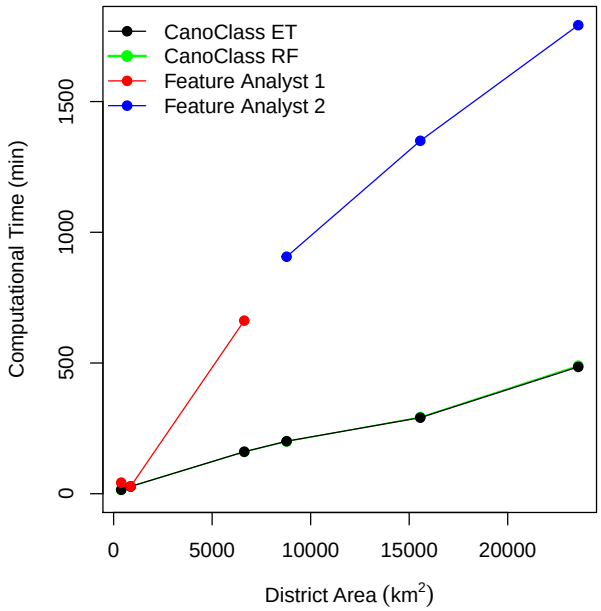


Figure 4. Computational times versus district areas.

lizing the RF classifier. As shown in Figure 4 and Table **??**, computational times for individual districts were similar between the ET and RF algorithms with each algorithm being only marginally faster than the other. While usual comparisons between the RF and ET classifiers find the ET algorithm to be three times faster on average (Geurts et al., 2006), at scales so large as in this case study, it ultimately made little difference in computational time whichever algorithm was chosen. More importantly, CanoClass with both algorithms outperformed Feature Analyst even when time is added for the creation of indices with the exception of the Lookout Mountain district for which Feature Analyst was marginally faster than the CanoClass RF algorithm by 0.04 min.

Physiographic District	Area (km^2)	Index Creation	CanoClass ET	CanoClass RF	Feature Analyst
McCaysville Basin	386	1.11	15.15	15.03	$42.47^{1,*}$
Lookout Mountain	866	3.72	27.67	27.85	$27.81^{1,*}$
Dougherty Plain	6630	22.21	160.63	160.00	$662.08^{1,\dagger,\ddagger}$
Winder Slope	8780	28.27	200.97	199.63	$906.51^{2,\dagger}$
Barrier Island Sequence	15563	40.68	290.85	293.04	$1349.84^{2,*,\S}$
Vidalia Uplands	23579	68.43	485.12	490.01	$1792.10^{2,\dagger,\S}$

Table 2. Computational times in minutes. Index creation times are shown separately to give context to the additional times the method used by CanoClass utilizes. CanoClass ET/RF times do not include the time to create the indices and is only the time to classify and save the raster. Feature Analyst times similarly shows the amount of time to classify and save the output. 1: Windows system 1. 2: Windows system 2. *: Shapefile outputs. †: GeoTIFF outputs. ‡: Simultaneous runs with other two districts that were not used in this study. §: The three outliers from Figure 3a were replaced by the average tile processing time.

Considering that the Linux system, where all CanoClass runs were conducted, is equipped with the second best CPU but with the slowest drive, the drive type did not act as a bottleneck. In other words, SSDs in the Windows systems did not help. Also, taking into account the three CPUs within 12 % difference in effective speed, the performance improvement of CanoClass between 283 % to 464 % can be considered exceptional assuming that it has produced comparable classification quality to Feature Analyst, which we will discuss in the next section.

4.2 Classification Quality

Table **??** shows the F_w values of the 20 QQs randomly chosen within the Blue Ridge Mountains district. The mean F_w values for the ET are RF algorithms when compared with Feature Analyst were 0.87617 and 0.87563, respectively. In other words, CanoClass with the two different classifiers produced results that are similar to the results of Feature Analyst within 87.56 % to 87.62 % similarity. The most notable differences between the two data sets can be observed in areas with a high amount of farm land. Feature Analyst was better at reliably separating low growing crops with similar spectral signatures from tree canopy partly because of Feature Analyst's segmentation abilities and combined ensemble classifiers. While CanoClass struggled with separating low growing green crops from canopy, it performed better in areas of extremely dense forest as can be seen in Figure 5. Notably in tile 2, while neither system performed perfectly for the noisy NAIP tile, CanoClass was a better predictor than Feature Analyst with the inaccuracy of the latter constituting to a poor F_w of only 0.7327 when compared to the ET classifier, and a lower F_w of 0.7307 when compared to the RF classifier, which is not shown in Figure 5. Overall however, the high similarity of 87.56 % to 87.62 % between the two data sets indicates a high level of confidence in the capabilities of CanoClass.

Tile	1	2	3	4	5	6	7	8	9	10
F_w ET	0.9225	0.7327	0.9409	0.8571	0.9328	0.8968	0.8474	0.9361	0.9167	0.9838
F_w RF	0.9221	0.7307	0.9409	0.8563	0.9325	0.8970	0.8470	0.9356	0.9165	0.9838
Tile	11	12	13	14	15	16	17	18	19	20
F_w ET	0.9414	0.9290	0.9302	0.8189	0.9079	0.7906	0.8172	0.8957	0.7337	0.7920
F_w RF	0.9413	0.9287	0.9302	0.8178	0.9080	0.7888	0.8163	0.8954	0.7325	0.7912

Table 3. F_w values of the 20 random QQs chosen within the Blue Ridge Mountains district.

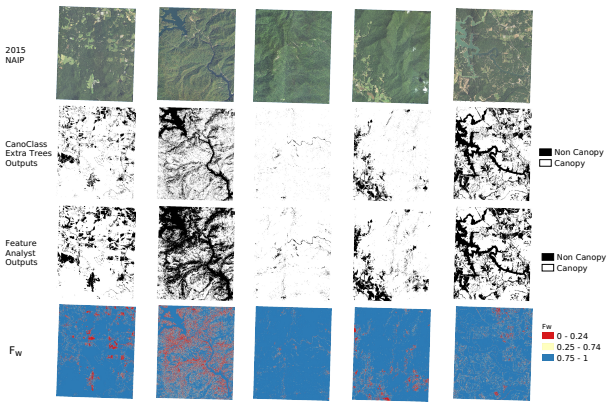


Figure 5. Tiles 1, 2, 10, 11, and 13 from Table **??** in that order from left to right showing the 2015 NAIP tiles, the raster outputs of the ET classification within CanoClass, Feature Analyst, and the raster visualizing the F_w values of each cell showing the differences between the classified outputs with a window of $w = 3$.

5. CONCLUSIONS

We developed an open-source Python framework titled CanoClass for classifying canopy using remotely sensed imagery. As an increasing amount of imagery becomes publicly available, so does the need to leverage the new data for studying forestry. Canopy classification is becoming more integral to create up-to-date data sets to influence policy, assist in research, and enable efficient environmental monitoring. To tackle issues presented in canopy classification, we developed CanoClass through the integration of GDAL and scikit-learn within

a Python framework. CanoClass is an open-source system that is both scalable and reproducible. This system provides training, classification, and post-processing utilities with the goal of bridging gaps left by proprietary and open-source systems in creating canopy data sets while addressing the issues inherent in canopy classification problems. CanoClass can be applied to any remotely sensed imagery while batch processing functions are provided specifically for NAIP imagery. The resulting process was found to be computationally 283 % to 464 % faster than Textron Systems' Feature Analyst, proprietary classification software, while producing comparable results with a high similarity of 87.56 % to 87.62 %. The system introduced in this research looks to provide an open and accessible framework to build canopy data for the future without the need to pay excessive cost for commercial proprietary software. Future work includes reduction of data requirements such as eliminating the need for the near-infrared band, which may not be readily and freely available for most researchers and areas.

SOFTWARE AVAILABILITY

CanoClass:

- `https://github.com/ocsmit/canoclass`
- Free under the GNU GPL 3.0 license

System requirements:

- Operating systems: Microsoft Windows XP or newer, macOS 10.4.10 or newer, recent GNU/Linux or a UNIX variant.
- Tested on Microsoft Windows 10 and GNU/Linux 5.4.0-40.
- Python 3.5 or newer

Required Python modules:

- Rindcalc: `https://github.com/ocsmit/rindcalc`
- Scikit-learn: `https://scikit-learn.org`
- SciPy: `https://scipy.org`
- NumPy: `https://numpy.org`
- GDAL: `https://gdal.org`

ACKNOWLEDGEMENTS

The authors acknowledge Joan Scales and Joe Burgess from the Georgia Forestry Commission (GFC) for kindly providing the 2015 NAIP imagery for this research. The authors also thank Dr. Allison J. Bailey from the University of North Georgia and her team for providing the 2015 canopy results and trained models produced by Feature Analyst through their 2016 GFC grant. The 2015 NAIP imagery was also obtained through the same grant. They do not necessarily agree with our work and the inferences drawn in this study. The 2009 canopy data set produced by Feature Analyst was funded by the project 42015377 / fund source 15377 through another GFC grant awarded to the authors. The CanoClass software itself is an independent unfunded project.

References

Bailey, A. J., Bailey, Jr., C. O., 2019. Using Feature Analyst & NAIP imagery to conduct a statewide tree canopy assessment of Georgia. *Forestry Research and Engineering: International Journal*, 3(1), 15–18.

Bala, G., Caldeira, K., Wickett, M., Phillips, T. J., Lobell, D. B., Delire, C., Mirin, A., 2007. Combined climate and carbon-cycle effects of large-scale deforestation. *Proceedings of the National Academy of Sciences*, 104(16), 6550–6555.

Basu, S., Ganguly, S., Nemani, R. R., Mukhopadhyay, S., Zhang, G., Milesi, C., Michaelis, A., Votava, P., Dubayah, R., Duncanson, L., Duncanson, L., Cook, B., Yu, Y., Saatchi, S., DiBiano, R., Karki, M., Boyda, E., Kumar, U., Li, S., 2015. A semiautomated probabilistic framework for tree-cover delineation from 1-m NAIP imagery using a high-performance computing architecture. *IEEE Transactions on Geoscience and Remote Sensing*, 53(10), 5690–5708.

Benito, E., Santiago, J. L., De Blas, E., Varela, M. E., 2003. Deforestation of water-repellent soils in Galicia (NW Spain): Effects on surface runoff and erosion under simulated rainfall. *Earth Surface Processes and Landforms: The Journal of the British Geomorphological Research Group*, 28(2), 145–155.

Bosch, M., Locatelli, M., Hamel, P., Jaligot, R., Chenal, J., Joost, S., 2020. Evaluating urban greening scenarios for urban heat mitigation: A spatially-explicit approach. *bioRxiv*, 1–20.

Breiman, L., 2001. Random forests. *Machine Learning*, 45(1), 5–32.

Cho, H., Smith, O., McCollum, J., 2020. Georgia statewide assessment of 2009 canopy. Unpublished technical report submitted to the Georgia Forestry Commission.

Costanza, R., 1989. Model goodness of fit: A multiple resolution procedure. *Ecological Modelling*, 47(3–4), 199–215.

Coulston, J. W., Moisen, G. G., Wilson, B. T., Finco, M. V., Cohen, W. B., Brewer, C. K., 2012. Modeling percent tree canopy cover: A pilot study. *Photogrammetric Engineering & Remote Sensing*, 78(7), 715–727.

Dallaqua, F., Faria, F. A., Fazenda, A. L., 2018. Active learning approaches for deforested area classification. *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, IEEE, 48–55.

Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., Hoersch, B., Isola, C., Laberinti, P., Martimort, P. et al., 2012. Sentinel-2: ESA's optical high-resolution mission for GMES operational services. *Remote Sensing of Environment*, 120, 25–36.

Etemadi, N., 1981. An elementary proof of the strong law of large numbers. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 55(1), 119–122.

Filchev, L., 2010. Land use/land cover classification of the Teyna river basin using automated feature extraction (AFE) algorithms. *Sixth Scientific Conference with International Participation*.

Franklin, S. E., 2001. *Remote sensing for sustainable forest management*. CRC press.

Freund, Y., Schapire, R. E., 1996. Experiments with a new boosting algorithm. *International Conference in Machine Learning*, 96, Citeseer, 148–156.

GDAL/OGR Contributors, 2020. GDAL/OGR Geospatial Data Abstraction software Library. Open Source Geospatial Foundation.

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. *Machine Learning*, 63(1), 3–42.

Gislason, P. O., Benediktsson, J. A., Sveinsson, J. R., 2006. Random forests for land cover classification. *Pattern Recognition Letters*, 27(4), 294–300.

Huang, T., Yang, G., Tang, G., 1979. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1), 13–18.

Kassawmar, T., Murty, K., Abraha, L., Bantider, A., 2019. Making more out of pixel-level change information: Using a neighbourhood approach to improve land change characterization across large and heterogeneous areas. *Geocarto International*, 34(9), 977–999.

Kaufman, Y., Tanre, D., 1992. Atmospherically resistant vegetation index (ARVI) for EOS-MODIS. *IEEE transactions on Geoscience and Remote Sensing*, 30(2), 261–270.

Kuhnert, M., Voinov, A., Seppelt, R., 2005. Comparing raster map comparison algorithms for spatial modeling and analysis. *Photogrammetric Engineering & Remote Sensing*, 71(8), 975–984.

Lawson, E., Smith, D., Sofge, D., Elmore, P., Petry, F., 2017. Decision forests for machine learning classification of large, noisy seafloor feature sets. *Computers & Geosciences*, 99, 116–124.

Lo, C. P., Yang, X., 2002. Atlanta, Georgia Metropolitan Area. *Photogrammetric Engineering & Remote Sensing*, 68(10), 1073–1082.

NASA, 2020. Moderate Resolution Imaging Spectroradiometer (MODIS).

O'Brien, M. A., 2003. Feature extraction with the VLS Feature Analyst system. *Proceedings of the American Society for Photogrammetry and Remote Sensing*, Citeseer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Platz, D., 2015. R eports. 96(2), 348–354.

QGIS Development Team, 2019. QGIS Geographic Information System. Open Source Geospatial Foundation Project.

Rani, M., Kumar, P., Pandey, P. C., Srivastava, P. K., Chaudhary, B. S., Tomar, V., Mandal, V. P., 2018. Multi-temporal NDVI and surface temperature analysis for Urban Heat Island inbuilt surrounding of sub-humid region: A case study of two geographical regions. *Remote Sensing Applications: Society and Environment*, 10(March), 163–172. https://doi.org/10.1016/j.rsase.2018.03.007.

Reed, W. P., Kaye, M. W., 2020. Bedrock type drives forest carbon storage and uptake across the mid-Atlantic Appalachian Ridge and Valley, U.S.A. *Forest Ecology and Management*, 460.

Roy, D. P., Wulder, M. A., Loveland, T. R., Woodcock, C., Allen, R. G., Anderson, M. C., Helder, D., Irons, J. R., Johnson, D. M., Kennedy, R. et al., 2014. Landsat-8: Science and product vision for terrestrial global change research. *Remote Sensing of Environment*, 145, 154–172.

Senf, C., Pflugmacher, D., Zhiqiang, Y., Sebald, J., Knorn, J., Neumann, M., Hostert, P., Seidl, R., 2018. Canopy mortality has doubled in Europe's temperate forests over the last three decades. *Nature Communications*, 9(1), 1–8.

Senf, C., Seidl, R., 2020. Mapping the forest disturbance regimes of Europe. *Nature Sustainability*. http://dx.doi.org/10.1038/s41893-020-00609-y.

Shen, G., Wang, Z., Liu, C., Han, Y., 2020. Mapping aboveground biomass and carbon in Shanghai's urban forest using Landsat ETM+ and inventory data. *Urban Forestry and Urban Greening*, 51(February), 126655. https://doi.org/10.1016/j.ufug.2020.126655.

Sriwongsitanon, N., Taesombat, W., 2011. Effects of land cover on runoff coefficient. *Journal of Hydrology*, 410(3-4), 226–238. http://dx.doi.org/10.1016/j.jhydrol.2011.09.021.

Textron Systems, 2020. Feature Analyst.

Tucker, C. J., 1979. Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sensing of Environment*, 8(2), 127–150.

USDA, 2020. National Agriculture Imagery Program (NAIP) Imagery.

UserBenchmark, 2020. UserBenchmark: speed test your PC in less than a minute.

van der Walt, S., Colbert, S. C., Varoquaux, G., 2011. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22–30.

Šimić de Torres, I., 2016. Analysis of satellite images to track deforestation. B.S. thesis, Universitat Politècnica de Catalunya.

Wong, N. H., Yu, C., 2005. Study of green areas and urban heat island in a tropical city. *Habitat International*, 29(3), 547–558.

Xu, H., Yang, M., Liang, L., 2010. An improved random decision trees algorithm with application to land cover classification. *2010 18th International Conference on Geoinformatics*, IEEE, 1–4.