Article

AI-Based Tree Species Classification Using Pseudo Tree Crown Derived From UAV Imagery

Shengjie Miao , Kongwen Frank Zhang * , Hongda Zeng , Jane Liu

Posted Date: 15 February 2024

doi: 10.20944/preprints202402.0786.v1

Keywords: Pseudo Tree Crown (PTC); PyTorch; Artificial Intelligence (AI); Unmanned Aerial Vehicle (UAV); individual tree species (ITS) classification

Article

AI-Based Tree Species Classification Using Pseudo Tree Crown Derived from UAV Imagery

Shengjie Miao [1,†], K. Frank Zhang [2,*], Hongda Zeng [1] and Jane Liu [1,3]

[1] Fujian Normal University, Fujian, China 1; qsx20221095@student.fjnu.edu.cn (S.M.), zeng.hd@fjnu.edu.cn (H.Z.)
[2] University of the Fraser Valley, Abbotsford, Canada 2; Frank.Zhang@ufv.ca
[3] University of Toronto, Toronto, Canada 3; jane.liu@utoronto.ca
[*] Correspondence: Frank.Zhang@ufv.ca; Tel.: 604-504-7441 x4401
[†] Current address: #19 Ninghua Rd, Fuzhou, Fujian, China

Abstract: Urban tree classification is pivotal in urban planning and management, facilitating informed decision-making processes. In this study, we introduce a novel data presentation method termed Pseudo Tree Crown, designed to enhance the accuracy and efficiency of urban tree classification. Leveraging the latest advancements in artificial intelligence (AI), we employ a state-of-the-art classification scheme, PyTorch, to maximize the accuracy of tree classification. Our results demonstrate a robust classification accuracy of over 95% from high spatial resolution imagery from Unmanned Aerial Vehicle (UAV), underscoring our proposed approach's effectiveness. Moreover, the adaptability of our method renders it applicable to various study areas, highlighting its versatility and potential for widespread implementation in urban planning and management initiatives.

Keywords: Pseudo Tree Crown (PTC), PyTorch, Artificial Intelligence (AI), Unmanned Aerial Vehicle (UAV), individual tree species (ITS) classification

1. Introduction

Urban trees play a vital role in fostering sustainable development within cities. They enhance the quality of the living environment, supply oxygen for organisms, and clean urban air. [1] Typically, urban trees grow in isolation amidst various urban facilities. Therefore, examining specific tree species becomes a fundamental aspect of urban management. [2] Traditional methods of classifying tree species heavily depend on on-site observation, relying on the visual recognition of inspectors, which is subjective and prone to errors. This approach is not only time-consuming but also incurs significant costs. [3] The rapid advancement of Unmanned Aerial Vehicle (UAV) remote sensing technology has introduced high-resolution and multi-source data in a timely, effective, and cost-efficient manner. Developing a suitable algorithm to harness this valuable data is essential for delivering accurate results promptly in individual tree classification. [4]

Recently, numerous advancements have emerged in the classification of urban tree species, encompassing various data sources such as hyperspectral images, LiDAR data, and high-resolution images from UAV using two primary classification approaches: 1) Random Forest (RF) and Support Vector Machine (SVM); 2) Artificial Neural Network (ANN) and its derivatives, including Convolutional Neural Networks (CNN) and Hierarchical Convolutional Neural Network (H-CNN). [5–9]

Liu et al. [10] conducted supervised data classification by extracting eight textures, including mean and variance, utilizing Maximum Likelihood Classification (MLC) and RF. Their findings indicated that MLC exhibited significantly lower time consumption compared to RF. Slavík et al. [11] proposed a novel approach for tree species classification using Generalized Linear Models (GLM) and RF on UAV laser scanning 3D point clouds. Through a method based on the Clark-Evans spatial aggregation index (CE), they demonstrated that RF achieved significantly higher classification accuracy than GLM. Rochdi et al. [12] utilized RF and SVM to assess the classification performance of LiDAR and RapidEye data, individually and in combination. Their results indicated a significant enhancement in classification accuracy with the joint use of LiDAR and RapidEye data, with the RF classifier outperforming SVM in tree species classification. Adelabu et al. [13] employed RF and SVM based on EnMap Box for the tree species classification of 5-band RapidEye satellite data. Their study revealed that SVM outperformed RF when dealing with limited training samples. Freeman et al. [14] investigated the performance of RF in addressing species imbalances across strata in Nevada. They proposed strategies to mitigate imbalances in sampling intensity and target species within training data. However, it's noted that traditional machine learning methods still rely on manually acquiring relevant information and manually creating necessary features, such as leaf shape, tree crown, various vegetation indices, and texture features.

In 1998, LeCun et al. [15] utilized CNN to classify two-dimensional shape changes, highlighting the significant advantages of CNN in automatically extracting features, capturing various visual structures, and facilitating classification. In 2017, Krizhevsky et al. [16] introduced the deeper CNN architecture, AlexNet, to the ImageNet dataset. This model, trained extensively on GPU, achieved breakthrough results on the dataset. Li et al. [26] employed the Caffe platform and a dual-task Gabor CNN on the Caffe platform for rapid tree species identification, presenting intuitive results in the application interface. Lei et al. [18] utilized a Hierarchical Convolutional Neural Network (H-CNN) for multi-temporal image classification, leveraging rich individual tree phenological features to enhance classification outcomes. With the rapid development of UAVs, acquiring high-resolution aerial images with superior detail and texture compared to satellite images has become more accessible. Qin et al. [19] developed a system in 2018 employing deep neural networks to classify high-resolution images, significantly reducing manual costs. Egli [20] utilized a shallow CNN for tree species classification based on high-resolution drone images, demonstrating consistent classification results exceeding 92%, regardless of external conditions. Lee et al. [21] compared the performance of two deep learning algorithms, illustrating that incorporating mixed tree species improves model classification performance. Li et al. [22] applied three CNN models to classify single tree images in high-resolution images, revealing a substantial advantage over RF and SVM methods. Liang et al. [23] employed a Spectral-Spatial Parallel Convolutional Neural Network (SSPCNN) for feature classification in hyperspectral images, demonstrating strong competitiveness. Nezami et al. [24] investigated tree species classification using a 3D Convolutional Neural Network (3D-CNN), with the best 3D-CNN classifier achieving approximately 5% higher accuracy than Multilayer Perceptron (MLP) at all levels. Shi et al. [25] employed an adjusted asymmetric convolution transfer learning model for hyperspectral image classification, significantly improving local tree species classification accuracy. Li et al. [26] enhanced various CNN models for identifying complex forest tree species, improving accuracy by introducing edge loss and accelerating model convergence. Furthermore, Chen et al. [27] integrated deep learning algorithms with UAV LIDAR data to segment single tree crowns, offering a comprehensive framework for accurate tree crown segmentation in forest conditions. However, it's crucial to note that the intertwining growth of trees and variations in canopy images can impact final classification results. Hardware limitations of UAV imaging equipment and natural factors during drone flights, such as weather and sunlight, can also affect classification data due to boundary weakening and blurring.

Upon meticulous examination and assessment of the documented research findings, it becomes evident that CNN emerges as highly effective in individual tree classification. Nevertheless, the varied outcomes associated with different CNN variations and the inherent challenges of utilizing two-dimensional image data for tree species classification prompted us to explore alternative approaches. We are pleased to report a successful discovery in addressing these challenges. We pursued a classification methodology based on a Pseudo tree crown (PTC) perspective by employing the latest and most advanced CNN-based artificial intelligence model, specifically PyTorch.

The exploration of PTC traces back to earlier efforts, notably highlighted in Fourier et al. [28] and Zhang and Hu [29] identified a promising avenue by leveraging the longitudinal profile of tree crowns, marking an initial attempt to correlate physical crown attributes with their optical nadir views. Results underscored a robust correlation. However, this approach lacked flexibility across various contexts due to computational constraints and reliance on parameterized equations and failed to gain widespread adoption.

In 2017, Balkenhol and Zhang [30] further delved into the potential expansion of three-dimensional correlations between the physical tree crown and the tree crown greyscale three-dimensional view. Limited by classification methods of the time, the study primarily showcased individual physical crown attributes alongside their 3D grayscale representations, laying the groundwork for the PTC concept. Another prototype was reported in Miao et al. [31].

The utilization of the PyTorch model yielded a commendable classification result of 85%, with an impressive accomplishment reaching 97% when adopting the PTC approach.

2. Data and Preprocessing

2.1. Study Area, Instruments and Data Acquisition

The research area of this study is located at the Cangshan Campus of Fujian Normal University (FNU) (26°02′05"N to 26°02′35"N and from 119°17′50"E to 119°18′29"E). It belongs to a subtropical monsoon climate, as shown in Figure 1. The optical image was captured on March 18, 2022, under overcast weather conditions. The D2000 UAV from Feima company, equipped with Hesai XT32 LiDAR, was used for the survey at a flight altitude of 100 meters. The camera model is SONY a6000, with a sensor size of 23.5mm by 15.6mm (APS-C), effective pixels of 24.3 million, and a 25mm fixed-focus lens. The image uses the CGCS2000 coordinate system and is in the Gauss-Krüger projection with the 3° zone and 100th band. The final image is saved in TIFF format with a resolution of 0.03m.

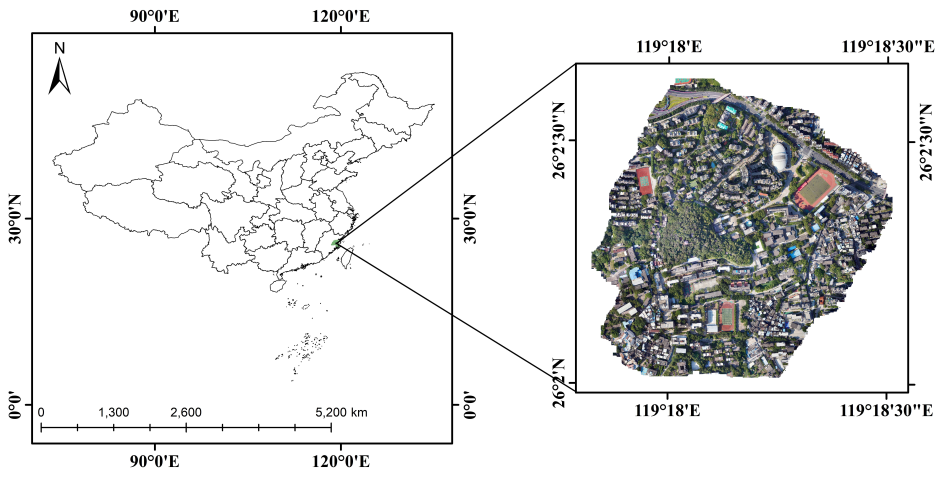


Figure 1. Study Area: Fujian Normal University Cangshan Campus, with the location map on the left and a drone orthophoto on the right.

In our study area, we selected five dominant tree species, namely Archontophoenix alexandrae (Aa), Mangifera indica (Mi), Livistona chinensis (Lc), Ficus microcarpa (Fm), and Sago palm (Sp), as the main research objects, as shown in Figure 2.

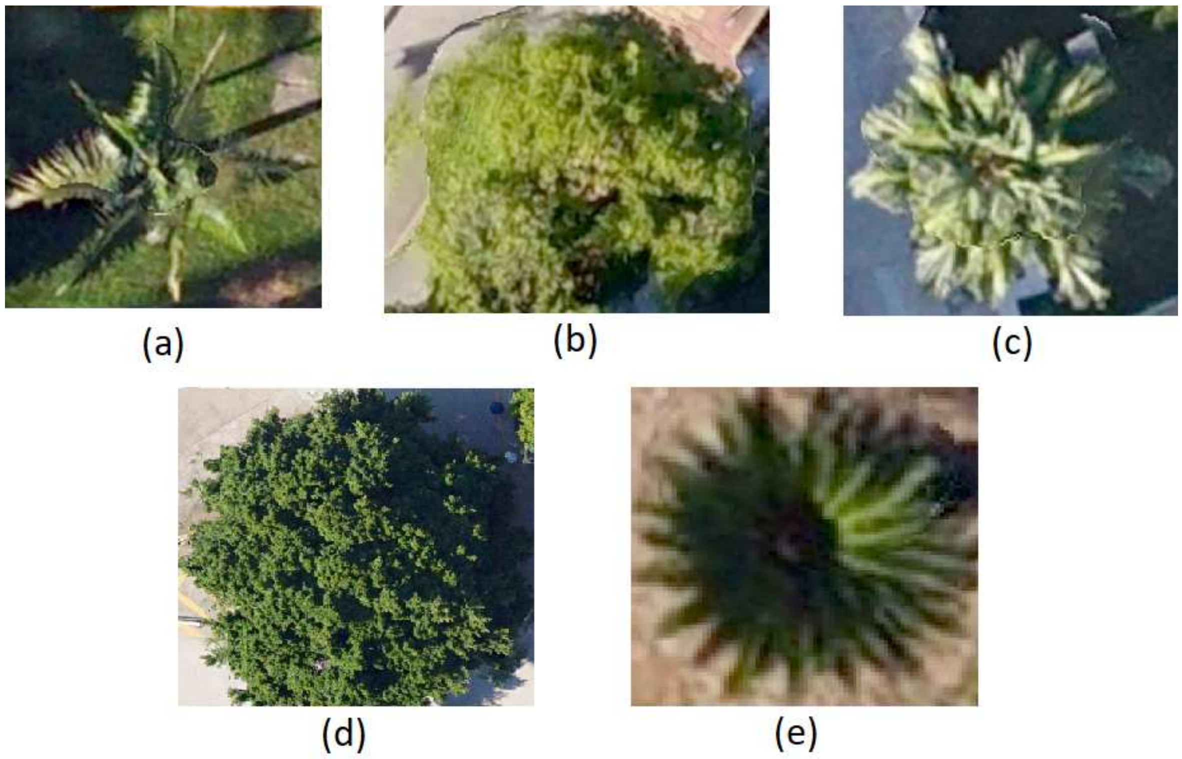


Figure 2. The nadir view of (a) Archontophoenix alexandrae (Aa) (b) Mango indica (Mi) (c) Livistona chinensis (Lc) (d) Ficus microcarpa (Fm) (e) Sago palm (Sp)

2.2. Data Pre-Processing

We first imported images into ArcGIS 10.2 and constructed an image pyramid to gather precise information for our experimental tree samples. We then manually identified and selected the shapefiles representing the target tree species, cropping corresponding image tiles accordingly. Validation of the tree species for every tree we used as samples was conducted through on-site visits. Finally, we got a total of 696 samples, with Aa comprising 226, Mi 134, Lc 143, Fm 131, and Sp 62, respectively.

Despite the relatively confined geographical area and uniform growing conditions, it's important to acknowledge that these tree samples have demonstrated in-class variation. Various factors can influence their growth, such as sunlight exposure, weather patterns, and shooting angles. Consequently, discernible variations exist between different sample categories and within the same category.

The general flowchart of pre-processing is illustrated in Figure 3.

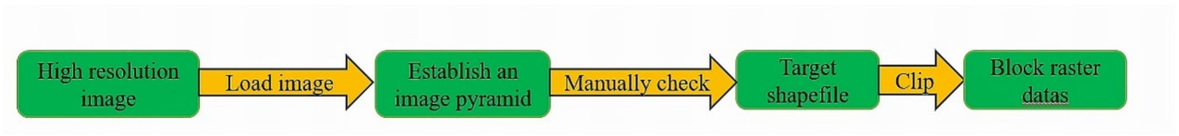


Figure 3. The workflow of individual tree sample in data pre-processng.

3. Methodology

Our study comprises two primary components, representing the main contributions of this research. Firstly, we introduced the Pseudo Tree Crown (PTC) as a new input image instead of standard nadir view images, marking its first incorporation in classification literature. Secondly, we compared three prominent AI-based classification methodologies - PyTorch, TensorFlow 2.0, and YOLOv5 - alongside the conventional Random Forest classification approach.

3.1. Pseudo Tree Crown (PTC)

The first step involves generating the PTC from the nadir view tree crown images, as shown in Figure 4. All collected sample data underwent parsing to extract information regarding the number of rows, columns, and associated bands for each sample. Subsequently, an affine matrix was formulated, and the image's projection details were extracted to retrieve pixel data for the green band. Upon acquiring this data, it was converted into an array format and organized into a grid. Any pixel values exceeding 255 were reset to nadir values, and the subplot's projection mode was configured for three-dimensional representation.

For the creation of our PTC, azimuth angles of -120° and an elevation angle of 75° were used. It started as a conventional default setting from Python 3D plot. However, we conducted a study to verify the variance in azimuth and elevation angles, elaborated in Section 4.2. Additionally, a comparative analysis was undertaken using different spatial resolutions of PTC and using original nadir view images instead of PTC, with findings detailed in sections 4.3 and 4.4, respectively.

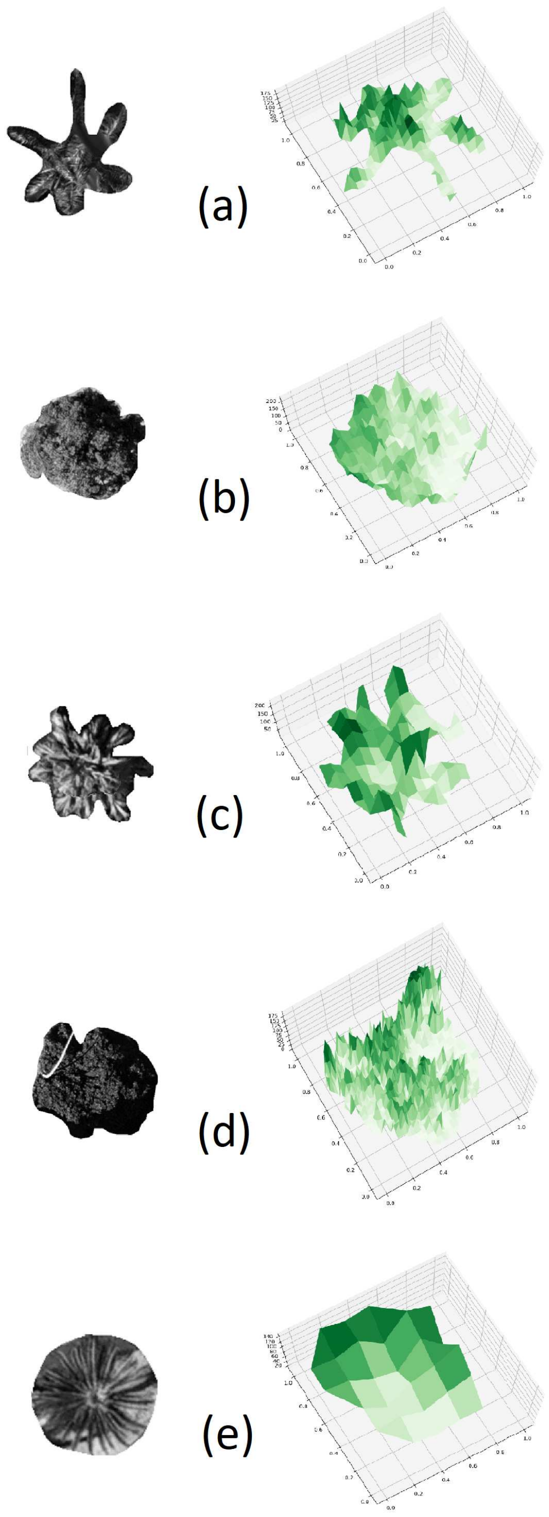


Figure 4. PTC of (a) Aa (b) Mi (c) Lc (d) Fm) (e) Sp

3.2. Convolutional Neural Networks (CNN)

The increase in CNN depth can extract better features and deliver better classification results. However, traditional CNN faces issues such as network degradation, vanishing gradients, and exploding gradients with the increasing depth of layers. This results in the performance of deeper networks being inferior to shallower ones. The ResNet introduces residual blocks, as shown in Figure 5.

ResNet, short for Residual Network, is mainly used for image classification and has wide applications in areas such as image segmentation and object detection.

ResNet addresses the challenges of gradient vanishing and accuracy degradation in deep networks by incorporating residual learning into traditional CNN. This innovation allows the network to become deeper, ensuring accuracy while controlling speed and effectively solving the problem of network degradation in deep networks. ResNet's main innovative points include the introduction of the Batch Normalization (BN) layer, which replaces Dropout to solve the vanishing/exploding gradient problem. Additionally, ResNet introduces the concept of residual learning to address network degradation.

Among various deep residual networks, we chose ResNet-50 to perform three-dimensional model image classification in the PyTorch and Tensorflow 2.0 frameworks. Through experiments, it was found that ResNet-50 successfully balanced model depth and performance. As the name suggests, ResNet-50 consists of 50 layers, as shown in Figure 6. The ResNet-50 network flows from input -> stage0 -> stage1 -> stage2 -> stage3 -> stage4 -> output. Stages 1 to 4 consist of 3, 4, 6, and 3 convolutional blocks.

In stage0, the input (3, 224, 224) undergoes a convolution operation with 64 filters of size (7, 7) and a stride of 2. Batch Normalization (BN) is applied, followed by the Rectified Linear Unit (ReLU) activation function. Finally, max-pooling with a kernel size of 3×3 and a stride of 2 is performed, resulting in an output size of 56. After this stage, the image shape changes from (3, 224, 224) to (64, 56, 56).

In stage1, three bottleneck blocks are used to process the output from stage0. Each bottleneck block consists of a series of convolution operations, including 1×1 convolution with 64 output channels, 3×3 convolution with 64 output channels, and 1×1 convolution with 256 output channels. These operations reshape the image to (256, 56, 56).

Similar operations are performed in stages 2, 3, and 4, gradually reducing the spatial dimensions and increasing the number of channels. In the end, global average pooling is applied to the (2048, 7, 7) image, resulting in an output of (2048, 1, 1). The feature map is then flattened to a one-dimensional vector and processed through fully connected layers for classification.

Considering this experiment involves five tree species, accuracy values for each category are provided in the end. The detailed architecture specifications of ResNet-50 are shown in Table 1.

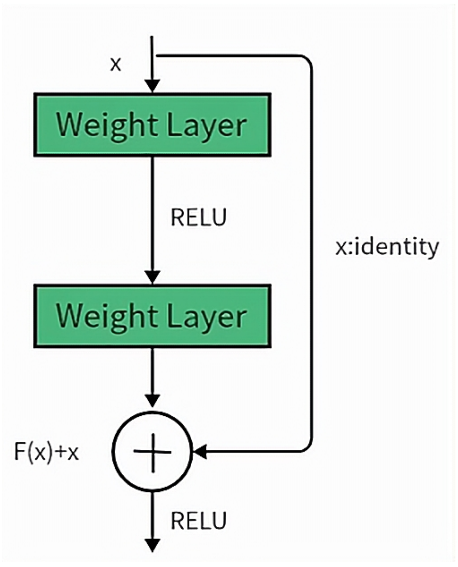


Figure 5. CNN Residual Block, where F(x) represents the residual, x is the identity mapping, and RELU is the activation function.

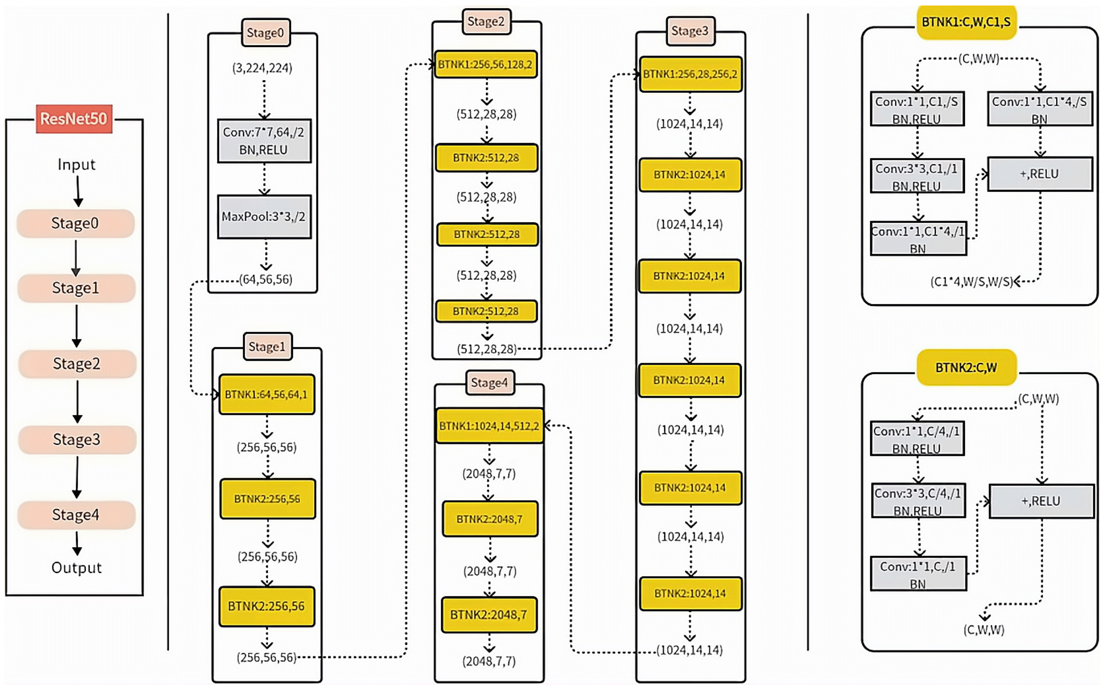


Figure 6. ResNet-50 network structure, where C represents the number of channels, W represents the input size, C1 represents the output number of channels, and S represents the convolutional stride.

Table 1. ResNet-50 network parameters and matrice size

Stage	Output Size	ResNet-50
Stage0	112 × 112	7 × 7, stride 2
Stage1	56 × 56	3 × 3 max pooling, stride 2 $\begin{bmatrix} 1X164 \\ 3X364 \\ 1X1256 \end{bmatrix}$ X 3
Stage2	14 × 14	$\begin{bmatrix} 1X1128 \\ 3X3128 \\ 1X1512 \end{bmatrix}$ X 4
Stage3	7 × 7	$\begin{bmatrix} 1X1256 \\ 3X3256 \\ 1X11024 \end{bmatrix}$ X 6
Stage4	1 × 1	$\begin{bmatrix} 1X1512 \\ 3X3512 \\ 1X12028 \end{bmatrix}$ X 4

3.3. Deep Learning Framework

3.3.1. Pytorch

In most cases, deep learning frameworks tend to focus on usability or speed, making it challenging to balance the two. PyTorch is a machine learning library indicating that these two goals can be somewhat compatible. It provides an imperative and Python programming style, supporting code as a model, making debugging easy, and maintaining compatibility with other popular scientific computing libraries. Additionally, it remains efficient and supports GPU acceleration for computations. Previous efforts recognized the value of dynamic eager execution in deep learning, and some recent frameworks have implemented this run-time-defined approach. However, they either sacrifice performance or use faster but less expressive languages, limiting their applicability. Through careful implementation and design choices, PyTorch achieves dynamic, eager execution without sacrificing performance. It enables dynamic tensor computations using automatic differentiation and GPU acceleration, maintaining performance comparable to the fastest deep learning libraries. This combination has gained popularity in the research community. PyTorch provides an array-based programming model accelerated by GPUs and allows differentiation through automatic differentiation integrated into the Python ecosystem.

One of the highlights of PyTorch is its simple and efficient interoperability, opening up possibilities to leverage the rich Python library ecosystem as part of user programs. PyTorch allows bidirectional data exchange with external libraries. It provides a mechanism using the torch.from_numpy() function and .numpy(), facilitating the conversion of tensors between NumPy arrays and PyTorch tensors. These exchanges occur without any data copying, making these operations very convenient and efficient, regardless of the size of the converted arrays.

Another significant advantage of PyTorch is that users can freely replace any component in PyTorch that does not meet their project requirements or performance needs. These components are designed to be completely interchangeable, allowing users to adjust them based on their temporary needs.

Efficiently running deep learning algorithms from the Python interpreter is currently one of the biggest challenges in this field. However, PyTorch addresses this issue differently by carefully optimizing various aspects of deep learning execution while allowing users to leverage additional optimization strategies easily. The PyTorch classification flowchart is illustrated in Figure 7.

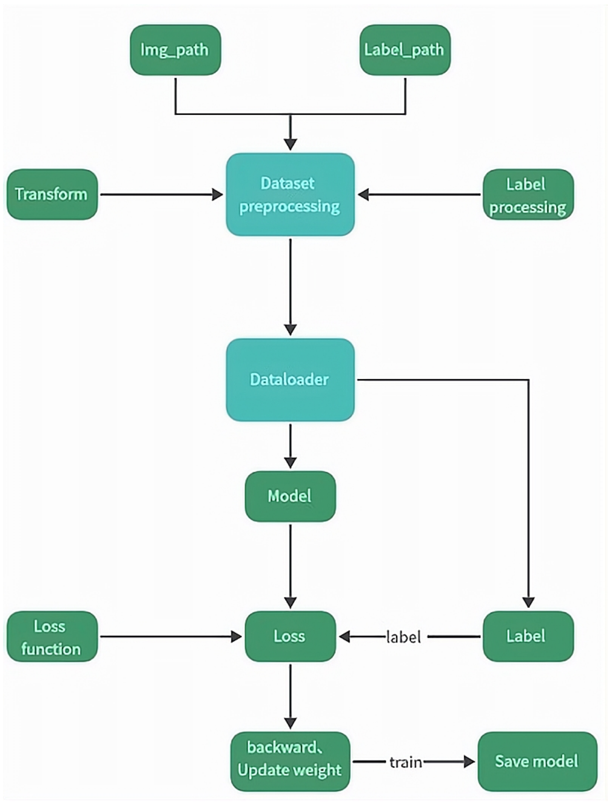


Figure 7. PyTorch classification flowchart

3.3.2. YOLOv5

YOLOv5 has achieved significant improvements in both detection accuracy and inference speed. It comes with a small weight file, approximately 90% smaller than YOLOv4, making it suitable for real-time detection on embedded devices. Compared to its predecessors, YOLOv5 is characterized by high detection accuracy, lightweight, fast detection times, and relatively mature technology. YOLOv5 includes four different models: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, each with distinct weight files. The differences among these architectures lie in their feature extraction modules and the convolutional kernels of the network. Another distinction is the size of the model and the number of model parameters, which vary for the four different architectures. We choose the YOLOv5s.pt as the weight file for training, and the training process flowchart for YOLOv5 is depicted in Figure 8.

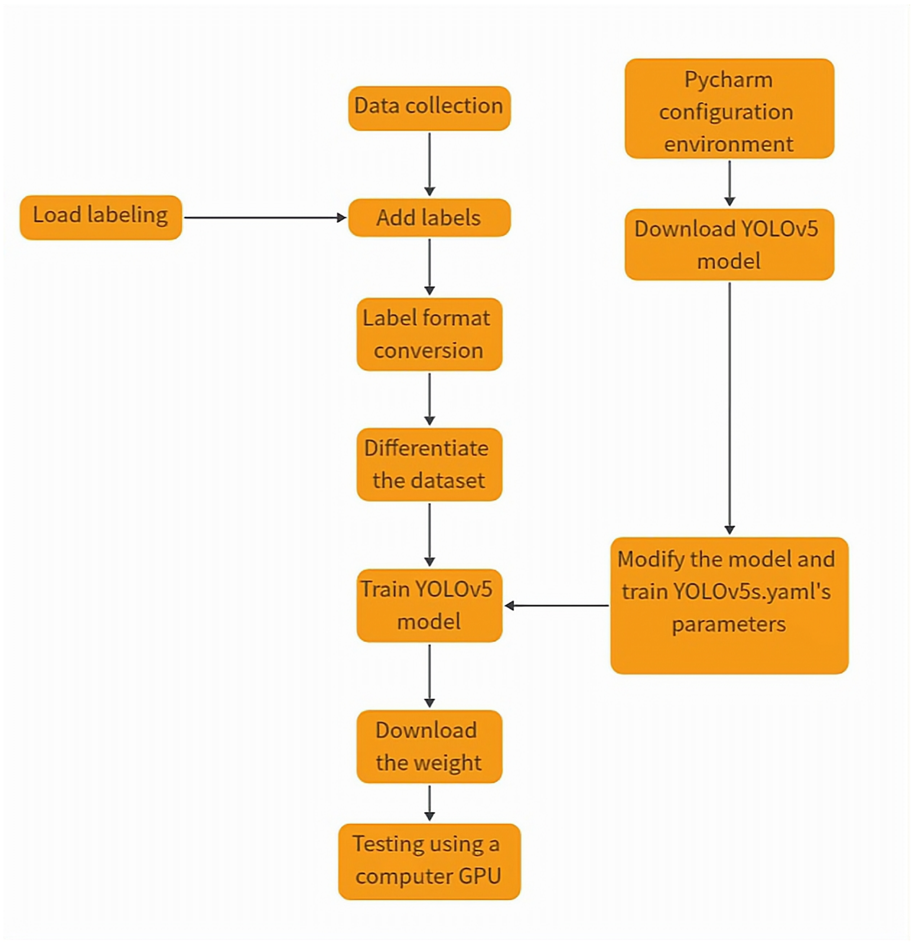

Data collection
Pycharm configuration environment
Load labeling
Add labels
Download YOLOv5 model
Label format conversion
Differentiate the dataset
Train YOLOv5 model
Modify the model and train YOLOv5s.yaml's parameters
Download the weight
Testing using a computer GPU

Figure 8. YOLOv5 classification flowchart

3.3.3. Tensorflow 2.0

TensorFlow is a numerical computing software library based on data flow graphs, providing interfaces and computational frameworks for implementing machine learning or deep learning algorithms. It combines flexibility and scalability, supporting various commonly used programming languages. TensorFlow ensures its efficiency and stability by utilizing CUDA, among other technologies. It allows mapping computation results to different hardware or operating systems, such as Windows, Linux, Android, iOS, and even large-scale GPU clusters. This significantly reduces development challenges. TensorFlow provides large-scale distributed training methods, enabling users to update model parameters using different devices, thus saving development costs. This flexibility allows users to implement model designs and train models on massive datasets quickly.

In recent years, TensorFlow has been widely applied in machine learning, deep learning, and other computational fields, including speech recognition, natural language processing, computer vision, robot control, information extraction, and more. In October 2019, Google released TensorFlow 2.0, and one of its major changes was the official integration and comprehensive support for Keras. Keras is a high-level neural network API known for its highly modular, minimalist, and extensible features. It provides clear and actionable error messages and supports Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). TensorFlow 2.0 uses the Sequential, compile, and fit methods of tf.keras to build, compile, and train models. The TensorFlow 2.0 training process flowchart is illustrated in Figure 9.

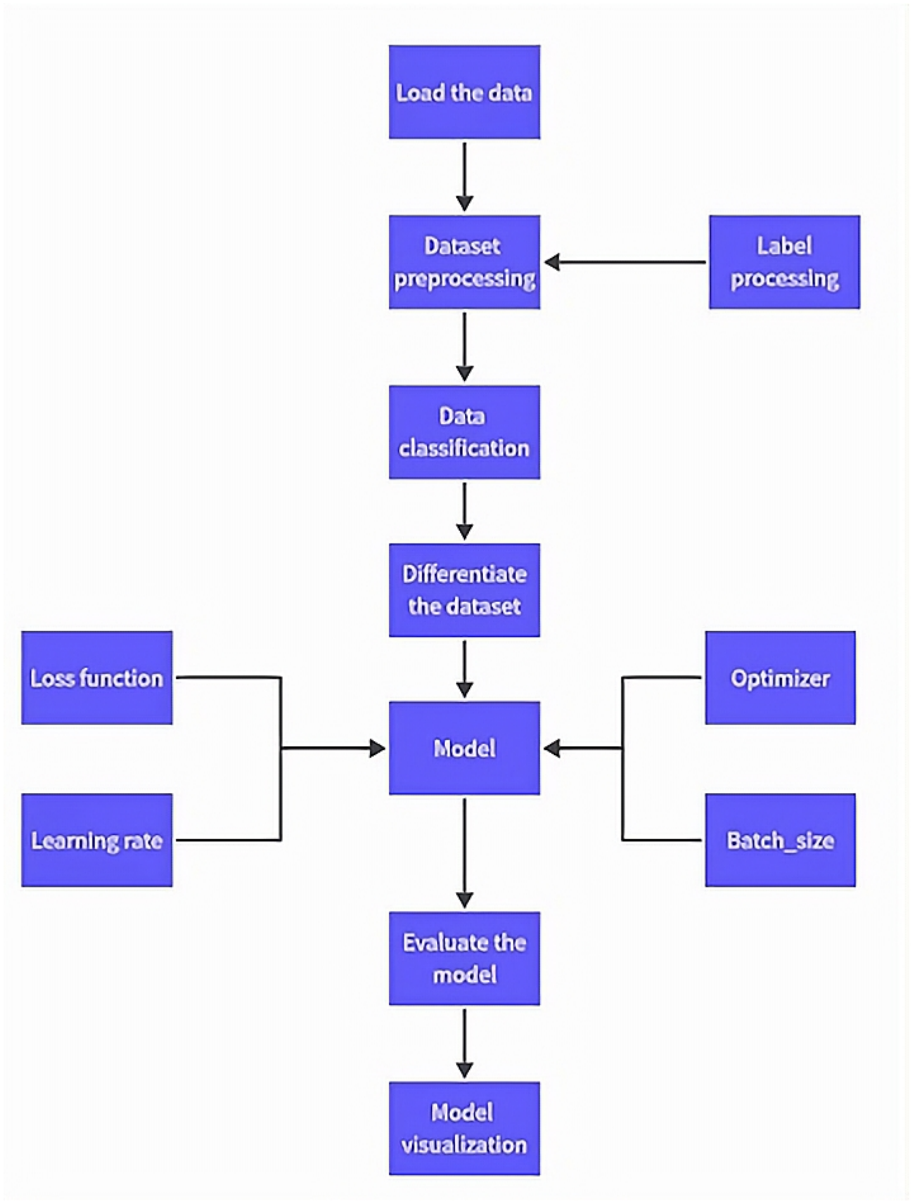


Figure 9. TF2.0 classification flowchart

3.3.4. Parameters settings

We set the input image size to 64 × 64 pixels for training classification models using these three deep learning frameworks. During the training process, we utilized a batch size of 4, selected the SGD optimizer to optimize our algorithm, and employed shuffling to prevent overfitting and ensure the model captures more accurate features. The training was conducted for a total of 50 epochs. The initial learning rate was set to1 * $10^{-}4$ and the momentum during training was set to 1 * $10^{-}3$. All the parameters obtained from the final trained model were recorded in Tensorboard, allowing us to monitor the model's changing trends in real-time.

3.4. Random Forest

Random Forest is a supervised learning algorithm employing an ensemble learning method consisting of numerous decision trees to generate a consensus output, representing the best answer to a given problem. Random Forest can be used for classification or regression and is a popular ensemble learning algorithm. It implements the Bagging (Bootstrap Aggregation) method in ensemble learning, serving as a homogeneous estimator composed of many decision trees. The individual decision trees in a Random Forest are not correlated. When performing classification with a Random Forest, each sample is evaluated and classified by every decision tree in the forest. Each decision tree produces a classification result, and the final result of the Random Forest is determined by the majority result (mode) among all decision trees.

4. Results and Discussion

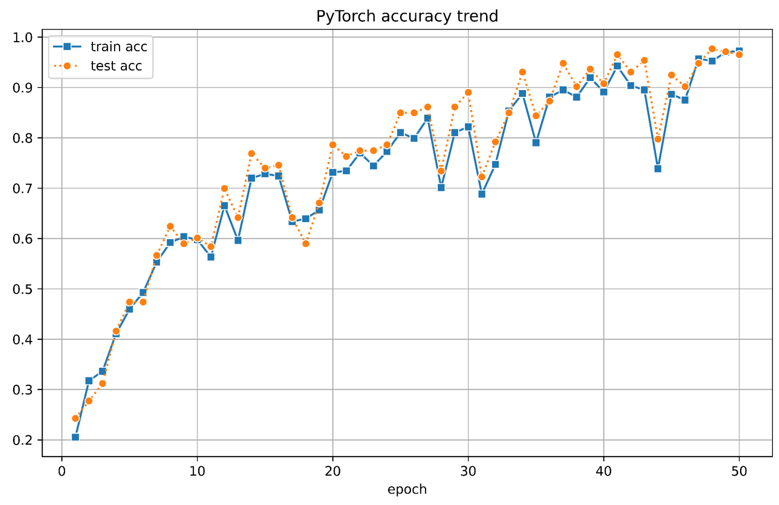
4.0.1. Framework Comparison

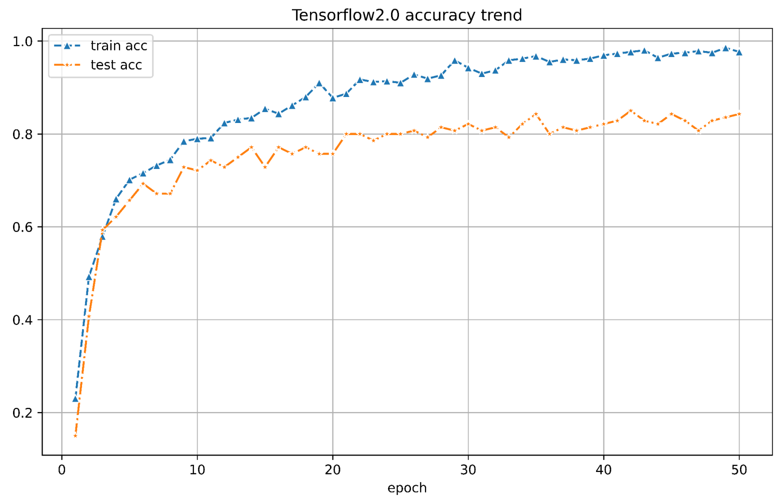
The primary objective of this research is to assess how various models affect tree species classification using PTC. Three prominent frameworks, PyTorch, Tensorflow 2.0, and YOLOv5, were selected for analysis. The results are summarized in Table 2 in Average Training Time and Average Classification Accuracy. We found PyTorch is more flexible and user-friendly than the other two methods. It provides an intuitive Python API, making it more convenient for users. While TensorFlow 2.0 has seen improvements in API design compared to version 1.0, it may still feel relatively complex in certain situations. YOLOv5, being specialized in object detection tasks, has its model and structure fixed towards specific objectives in object detection and may not be as flexible as PyTorch in image classification. PyTorch boasts strong community support and a wealth of third-party libraries, offering various pre-trained models and tools for rapidly developing image classification applications. Although TensorFlow 2.0 also has robust community support, its ecosystem is relatively complex compared to PyTorch. YOLOv5, on the other hand, has fewer model weights and options than the first two. In achieving the same goal, PyTorch often achieves the desired results with less effort, while TensorFlow 2.0 may require more work for similar objectives. YOLOv5, being less widely used in image classification and having a relatively smaller scale, may face limitations in support and issue resolution compared to the broader availability of PyTorch.

Table 2. The Average training time and average classification accuracy of different AI models

	Average Training time	Average Classification Accuracy
PyTorch	0h:44m:23s	0.9826
TensorFlow	1h:41m:53s	0.9200
YOLOv5	1h:17m:07s	0.9748

Under 50 epochs, YOLOv5 showed the most volatile performance but yielded a good final convergence. In contrast, TensorFlow 2.0 showed consistent stability but relatively lower accuracy. The results are illustrated in Figures 10 through 13. It was observed that the accuracy of all models exceeded 90% around 40 epochs, with models trained using the PyTorch framework achieving over 95%. While both PyTorch and Tensorflow 2.0 achieved training set accuracy of over 95%, the test set accuracy of Tensorflow2.0 was only around 80%, whereas PyTorch maintained an accuracy of over 95%, maintaining a good performance. For YOLOv5, it was noticed that when the classification threshold was set to 0.5, the training accuracy was generally higher than when the threshold was between 0.5 and 0.95. However, its highest classification accuracy only reached 92%, never exceeding 95%. Moreover, when all three models underwent 50 epochs, the training time for PyTorch was 45 minutes, while Tensorflow2.0 and YOLOv5 took 76 minutes and 102 minutes, respectively (Table 2). Overall, PyTorch demonstrated stronger stability, higher accuracy, and better efficiency in tree species classification.





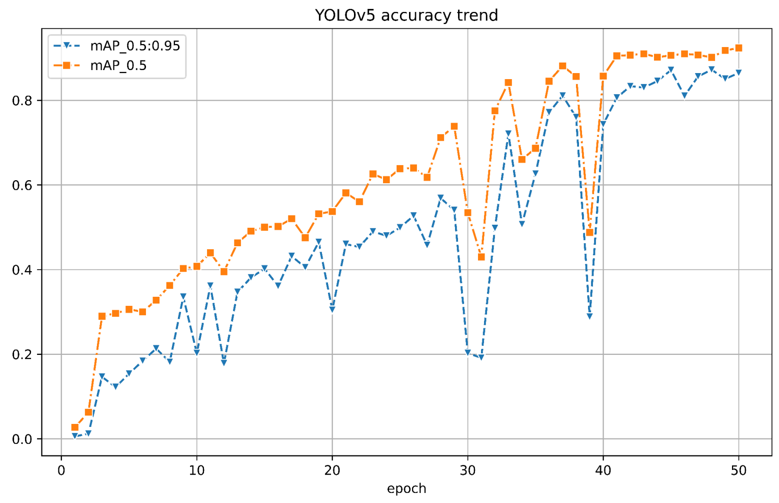


Figure 10. The accuracy of training and validation on the dataset for PyTorch, TensorFlow 2.0, and YOLOv5 is described in (a)-(c) respectively

4.1. Classification Accuracy Assessment

To further corroborate our results, we evaluated each model's overall classification accuracy and confusion matrices, as shown in Figure 14. Precision, recall, and specificity for each tree class are detailed in Table 3 to 5, providing quantitative metrics for the models' performance in tree species classification tasks. Through analyzing these results, we can gain a clearer and more comprehensive understanding of the performance of each model in tree species classification.

PyTorch achieved the highest overall classification accuracy at 98.26%, followed by Tensorflow2.0 at 84.29%, YOLOv5 at 75.89%, and the traditional Random Forest (RF) with an overall classification accuracy of 70.71%. As indicated in the table, in PyTorch, the precision, recall, and specificity for Aa, Lc, and Sp were all 1.0, while Mi maintained these three metrics above 0.95. The precision for Fm was slightly lower at 0.935, but the recall and specificity remained above 0.96. In Tensorflow2.0, Sp's performance was excellent, achieving 1.0 for all three metrics. Aa also maintained results above 0.9. Mi had a lower precision of 0.735, with recall and specificity above 0.92. The precision and recall for Lc ranged between 0.8 and 0.85, but specificity reached 0.965. Although the precision for Fm in Tensorflow2.0 was 0.79, the recall was only 0.556, and the specificity was 0.965. YOLOv5 and RF had less favourable classification results compared to PyTorch and Tensorflow2.0. Despite generally lower performance, YOLOv5 achieved a recall of 1.0 for Sp. RF attained a precision and specificity of 1.0 for Sp, indicating that our dataset performed best in identifying Sp regardless of the classification method used.

In summary, our research results suggest that due to the influence of factors such as the blurred edges and interweaving crowns of canopy images and poor texture effects in the original data, PyTorch's classification performance is superior to Tensorflow2.0, YOLOv5, and RF. This demonstrates that PyTorch has significant potential for multi-classification tasks using PTC from high-resolution remote sensing images. However, when evaluating the efficiency of a classification model, there are still many evaluation metrics to consider, along with the need to integrate various features of the dataset and the specific requirements of the classification task. Further research and refinement of these models' performance using different datasets in various environments are necessary.

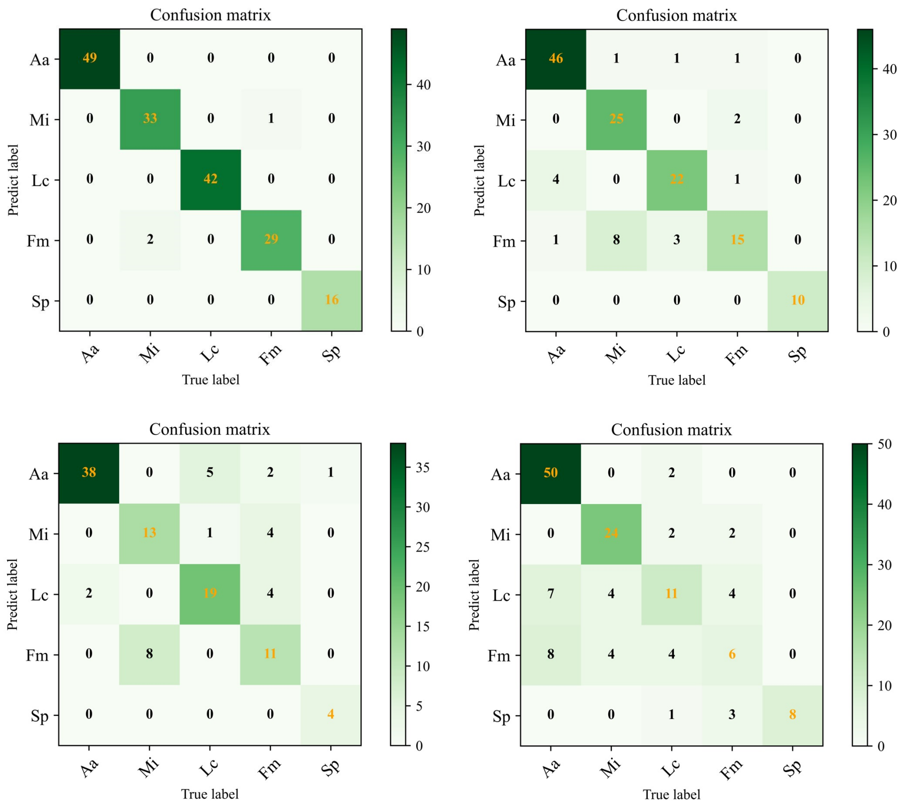


Figure 11. The confusion matrices for training on the dataset in PyTorch, TensorFlow 2.0, YOLOv5, and RF are presented in (a)-(d) respectively.

Table 3. Precision (PyTorch, TF2.0, YOLOv5 and RF)

Species	PyTorch	TF2.0	YOLOv5	RF
Archontophoenix alexandrae (Aa)	1.000	0.902	0.950	0.769
Mango indica (Mi)	0.971	0.735	0.620	0.750
Livistona chinensis (Lc)	1.000	0.846	0.760	0.579
Ficus microcarpa (Fm)	0.935	0.790	0.524	0.400
Sago palm (Sp)	1.000	1.000	0.800	1.000

Table 4. Recall (PyTorch, TF2.0, YOLOv5 and RF)

Species	PyTorch	TF2.0	YOLOv5	RF
Archontophoenix alexandrae (Aa)	1.000	0.938	0.826	0.962
Mango indica (Mi)	0.943	0.926	0.722	0.857
Livistona chinensis (Lc)	1.000	0.815	0.760	0.423
Ficus microcarpa (Fm)	0.967	0.556	0.579	0.273
Sago palm (Sp)	1.000	1.000	1.000	0.727

4.2. PTC Azimuth and Elevation Angle Impact Study

To explore the influence of varying azimuth and elevation angles on tree species classification outcomes using PTC, we selected three sets of angles: -120°, 75°; 90°, 75°; and 120°, 75°, as illustrated in Figure 12. PTCs were generated using these different angles and inputted into our PyTorch-based model for classification. The resultant classification outcomes are shown in Figure 13.

Table 5. Specificity (PyTorch, TF2.0, YOLOv5 and RF)

Species	PyTorch	TF2.0	YOLOv5	RF
Archontophoenix alexandrae (Aa)	1.000	0.945	0.970	0.828
Mango indica (Mi)	0.993	0.920	0.915	0.928
Livistona chinensis (Lc)	1.000	0.965	0.931	0.929
Ficus microcarpa (Fm)	0.986	0.965	0.893	0.923
Sago palm (Sp)	1.000	1.000	0.991	1.000

The graph illustrates that across all angle configurations, the final accuracy of both the training and test datasets consistently exceeds 95%. This suggests that changes in azimuth and elevation angles have minimal impact on the overall classification accuracy. This resilience is logical given PyTorch's application in training on human images, which does not necessitate specific viewing angles.

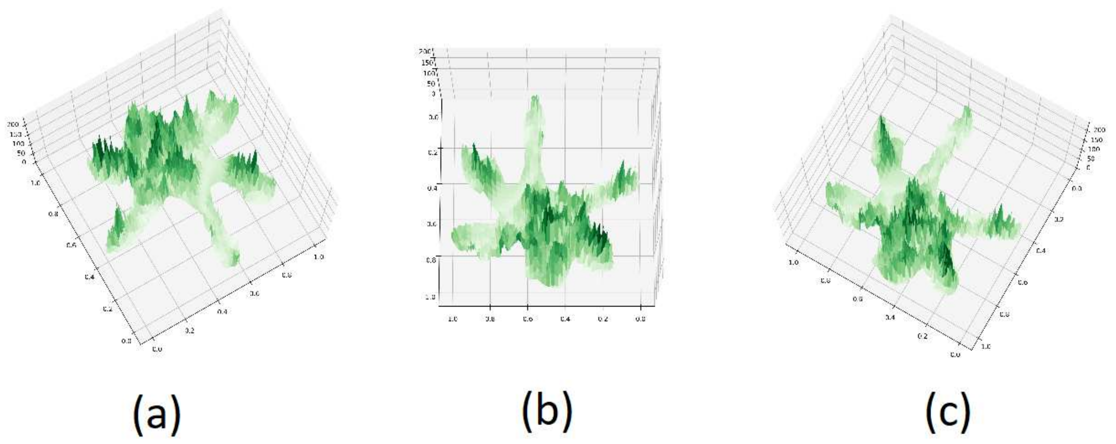


Figure 12. PTC of Aa with different azimuth and elevation:(a) -120, 75 (b) 90, 75 (c) 120, 75

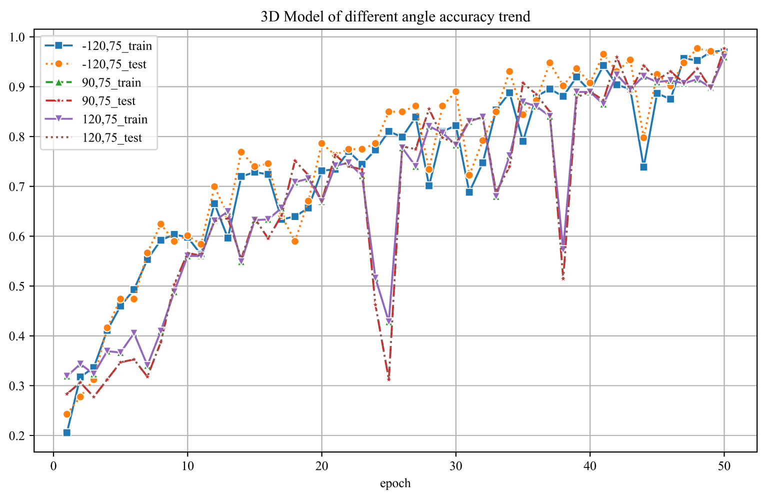


Figure 13. The PTC classification accuracy by different azimuth and elevation angles

4.3. Different Spatial Resolution PTC Impact Analysis

We also examined the influence of spatial resolution on PTC. Given that PTC derives height information from grayscale values and partially obscures tree crown details due to viewing angles, it inherently masks out many spatial intricacies, suggesting a natural resilience to changes in spatial resolution. However, we were interested in determining the scale at which classification accuracy would be affected.

To investigate this, we downsampled the original RGB images into lower-resolution versions, which reduces the resolution to a factor of 3, 5 and 10 of the original resolution. We extracted the green band again, cropped out patches of individual trees for each species, and established a resampled dataset, which is illustrated in Figure 14 (reduced by a factor of 10 from 0.03 m to 0.3 m). Different spatial resolution PTC were created after that. All datasets maintain accuracy above 90%, while the highest resolution, 0.03 m, archives a classification accuracy of over 95%. Remarkably, we observed no significant decrease in accuracy until the scale was reduced by over ten times. This indicates that as the resolution of the original data decreases, the features displayed by PTC are, to some extent, unfamiliar to the classifier in our PyTorch framework, leading to a slight decrease in accuracy.

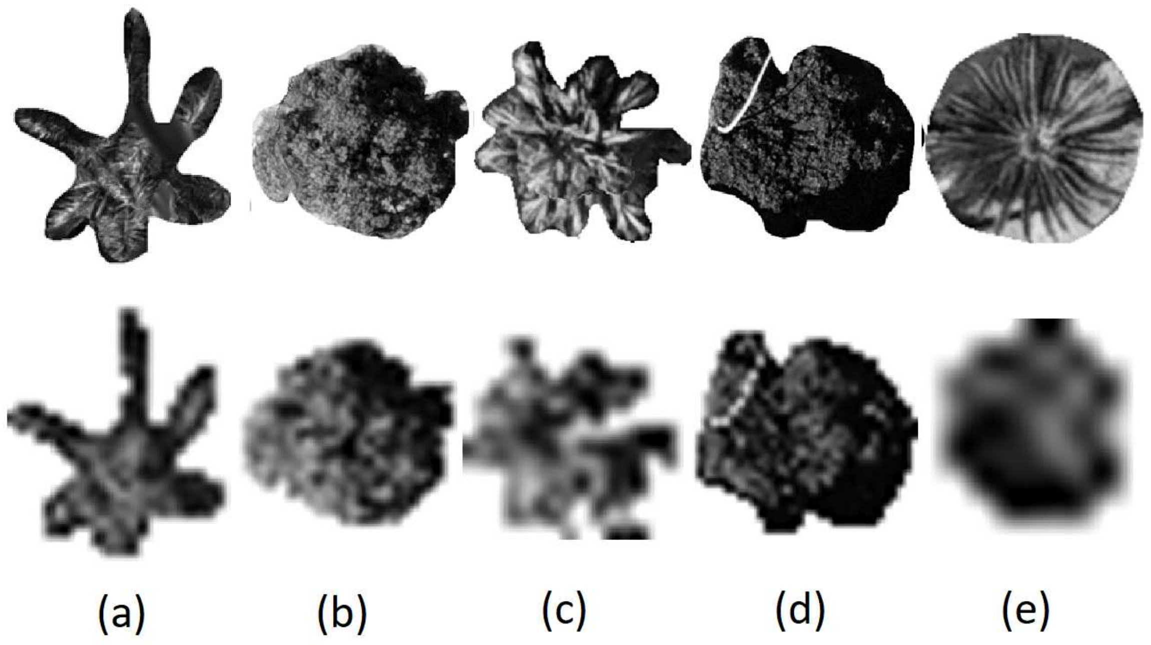


Figure 14. Resampled images of (a) Aa (b) Mi (c) Lc (d) Fm) (e) Sp

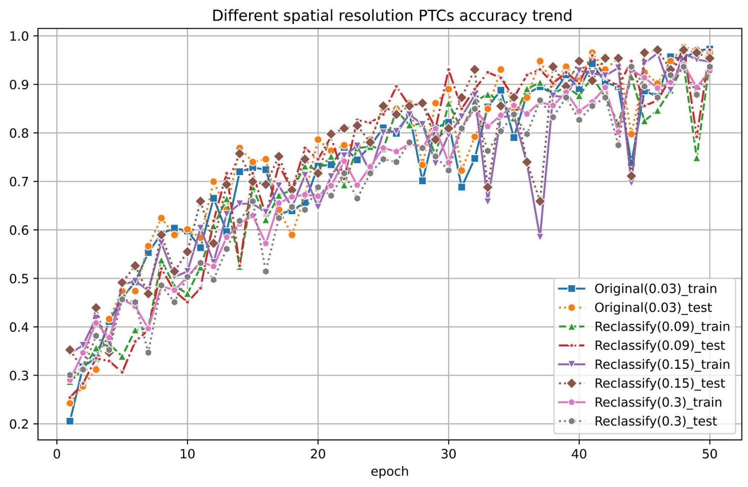


Figure 15. The classification results by different spatial resolution PTCs

4.4. Compare the Original RGB Image

Additionally, we compared the PTC with the original nadir view images using PyTorch classification. To keep the model's performance trained in PyTorch consistent on the PTC and original images, we also extracted the green band of the original 2D nadir view images as the input dataset. The accuracy of the original nadir view dataset during training reached between 80% and 85% for both the training and test sets, which is a bit worse than 97% from PTC using PyTorch.

We further explored the resistance of the resolution change for the original images. We did the same resample for the original nadir images. All of these datasets were then input into PyTorch for training, and the accuracy trends on the training and test sets are shown in Figure 16.

In contrast, due to reduced resolution, the accuracy of the reclassified original images dropped to between 70% and 80% for both the training and test sets. It is much more significant compared to the PTC, which remains approximately 90% even at a factor of 10.

Therefore, we conclude the PTC is much more resistant to reducing spatial resolution. It implicates more flexible application areas and robust classification results.

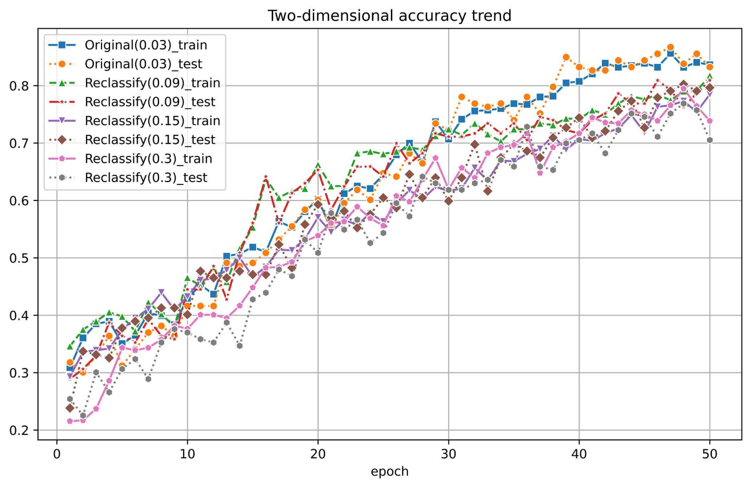


Figure 16. The graphical representation depicting the variation in accuracy for the two-dimensional dataset

4.5. Other Findings

It is worth mentioning that the accuracy for the tree species Aa is the highest in each classification method. This indicates that the classification model more easily recognizes the PTC, which is favourable to the species that have more spatial features. This could be attributed to its robust trunk and sparse, broad leaves, resulting in higher pixel distinguishability than other tree species. On the other hand, the accuracy for the Fm is relatively low, possibly due to its large crown, complex crown structure, and the intertwining growth of each tree. This leads to unclear boundaries of the crown layer, making it more challenging for the model to accurately recognize its features, thus affecting the classification of the banyan tree. In the future, we aim to improve our classification model algorithm further to achieve comprehensive and higher classification accuracy.

Eva Lindberg et al. utilized dense airborne laser scanning (ALS) data for crown delineation, extracting the spatial distribution of ALS data based on an elliptical crown model and conducting classification. Their method improved crown recognition, achieving a classification result of only 71% for tree species such as pine and spruce in cross-validation. In contrast, in this paper, by simply outputting the green band of RGB images and creating PTC, we achieved a classification accuracy of over 95% with simpler data processing and less workload. This further highlights the sensitivity of deep learning classification algorithms to the recognition of tree crown three-dimensional models.

5. Conclusions

In this study, we utilize a deep learning-based methodology to evaluate tree species classification within the Cangshan campus of Fujian Normal University, leveraging high-resolution aerial imagery acquired through low-altitude UAV flights. We have attained promising outcomes by employing individual tree PTC derived from these high spatial resolution aerial images. Our approach involves integrating PTC images into the PyTorch classification framework, resulting in classification accuracies consistently surpassing 95

We are excited to present the first application of PTC in image classification, yielding a 95% accuracy compared to utilizing nadir view images directly with PyTorch (87%) using PyTorch-based classification. Furthermore, among various AI-based classification methodologies, PyTorch has demonstrated superior robustness, accuracy, and efficiency compared to TF2.0 and YOLOv5.

Future research will focus on exploring the correlation between physical tree crowns and PTCs by incorporating LiDAR data. Preliminary findings indicate encouraging results. If successful, this endeavour will directly link 2D nadir images to 3D tree structures, facilitating the integration of additional parameters—such as diameter at breast height (dbh)—into classification models, a feat previously unattainable with solely 2D image data.

Funding: We would like to thank Fujian Province Forestry Science and Technology Project (2022FKJ03) for the financial support.

Data Availability Statement: All data and Python source codes are available upon request

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
GLM	Generalized Linear Models
MLC	Maximum Likelihood Classification
PTC	Pesedo Tree Crown
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle
ITS	Individual tree species

References

1. Pause, M., Schweitzer, C., Rosenthal, M., Keuck, V., Bumberger, J., Dietrich, P., Heurich, M., Jung, A., and Lausch, A. In Situ/Remote Sensing Integration to Assess Forest Health - A Review. *Remote Sensing* **2016**, *8*, 471, 10.3390/rs8060471.
2. Lausch, A., Borg, E., Bumberger, J., Dietrich, P., Heurish, M., Huth, A., Jung, A., Klenke, R., Knapp, S., Mollenhauer, H., Paasche, H., Paulheim, H., Pause, M., Schweitzer, C., Schmulius, C., Settele, J., Skidmore, A. K., Wegmann, M., Zacharias, S., Kirsten, T., and Schaepman, M. E., Understanding Forest Health with Remote Sensing, Part III: Requirements for a Scalable Multi-Source Forest Health Monitoring Network Based on Data Science Approaches. *Remote Sensing* **2018**, *10*, 1120, 10.3390/rs10071120.
3. Trisasongko, B. H., and Paull, D. A review of remote sensing applications in tropical forestry with a particular emphasis in the plantation sector. *Geocarto International* **2018**, , , https://doi.org/10.1080/10106049.2018.1516245 .
4. Gyamfi-Ampadu, E., and Gebreslasie, M. Two Decades Progress on the Application of Remote Sensing for Monitoring Tropical and Sub-Tropical Natural Forests: A Review. *Forests* **2021**, *12*, 739, https://doi.org/10.3390/f12060739 .
5. Plesoiamu, A., Stupariu, M., Sandric, I., Patru-Stupariu, I., and Dragut, L. Individual Tree-Crown Detection and Species Classification in Very High-Resolution Remote Sensing Imagery Using a Deep Learning Ensemble Model. *Remote Sensing* **2020**, *12*, 2426, 10.3390/rs12152426 .

6. Wagner, F. H., Perreira, M. P., Sanchez, A., Hirye, M. C.M., Zortea, M., Gloor, E., Phillips, O. L., Filho, C., Shimabukuro, Y. E., and Aragao, L. E.O.C. Individual tree crown delineation in a highly diverse tropical forest using very high resolution satellite images. *ISPRS Journal of Photogrammetry and Remote Sensing* **2018**, *145*, 362-377, doi.org/10.1016/j.isprsjprs.2018.09.013 .
7. Ballanti, L., Blesius, L., Hines, E., and Kruse, B. Tree Species Classification Using Hyperspectral Imagery: A Comparison of Two Classifiers. *Remote Sensing* **2016**, *8*, 445, 10.3390/rs8060445 .
8. Marrs, J., and Ni-Meister, W. Machine Learning Techniques for Tree Species Classification Using Co-Registered LiDAR and Hyperspectral Data. *Remote Sensing* **2019**, *11*, 819, 10.3390/rs11070819 .
9. Zhao, D., PAng, Y., Liu, L., and Li, Z. Individual Tree Classification Using Airborne LiDAR and Hyperspectral Data in a Natural Mixed Forest of Northeast China. *Forests* **2020**, *11*, 303, 10.3390/f11030303 .
10. Liu, H., Su, X., Zhang, C., and An, H. Landscape tree species recognition using RedEdge-MX: Suitability analysis of two different texture extraction forms under MLC and RF supervision. *Open Geosciences* **2022**, *14*, 985-994, doi.org/10.1515/geo-2022-0416.
11. Slavik, M., Kuzelka, K., Modlinder, R., and Surovy, P. Spatial Analysis of Dense LiDAR Point Clouds for Tree Species Group Classification Using Individual Tree Metrics. *Forests* **2023**, *14*, 1581, doi.org/10.3390/f14081581.
12. Rochdi, N., Yang, X., Staenz, K., Patterson, S., and Purdy, B., Mapping Tree Species in a Boreal Forest Area using RapidEye and LiDAR Data. In Proceedings of Earth Resources and Environmental Remote Sensing 2014 SPIE; doi.org/10.3390/f14081581.
13. Adelabu, S., Mutanga, O., Adam, E., and Cho, M.A. Exploiting machine learning algorithms for tree species classification in a semiarid woodland using RapidEye image. *Journal of Applied Remote Sensing* **2013**, *17*, , 10.1117/1.JRS.7.073480.
14. Freeman, E. A., Moisen, G. G., Frescino, T. S. Evaluating effectiveness of down-sampling for stratified designs and unbalanced prevalence in Random Forest models of tree species distributions in Nevada. *Ecological Modelling* **2012**, *233*, 1-10 , 10.1016/j.ecolmodel.2012.03.007.
15. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-Based Learning Applied to Document Recognition. In Proceedings of IEEE; **1998**.
16. Krizhevsky, A., Sutskever, I., and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM* **2017**, *60*, 84-90, doi.org/10.1145/3065386.
17. Li, M., Zhou, G., and Li, Z., Fast recognition system for Tree images based on dual-task Gabor convolutional neural network. *Multimedia Tools and Applications* **2022**, *81*, 28607-28631 , doi.org/10.1007/s11042-022-12963-4.
18. Lei, Z., Li, H., Zhao, J., Jing, L., Tang, Y., Wang, H.J. Individual Tree Species Classification Based on a Hierarchical Convolutional Neural Network and Multitemporal Google Earth Images *Remote Sensing* **2023**, *14*, 5124 , doi.org/10.3390/rs14205124.
19. Qin, Y., Chi, M., Liu, X., Zhang, Y., Zeng, Y., Zhao, Z., and Hinton, G. E., Classification of high resolution urban remote sensing images using deep networks by integration of social media photos. In Proceedings of the IGARSS 2018 (IEEE International Geoscience and Remote Sensing Symposium). **2018**, 7243-7446 .
20. Egli, S., and Hopke, M., CNN-Based Tree Species Classification Using High Resolution RGB Image Data from Automated UAV Observations *Remote Sensing* **2020**, *12*, 3892 , doi.org/10.3390/rs12233892.
21. Lee, E., Baek, W., and Jung, H., Mapping Tree Species Using CNN from Bi-Seasonal High-Resolution Drone Optic and LiDAR Data *Remote Sensing* **2023**, *15*, 2140 , doi.org/10.3390/rs15082140.
22. Li, H., Hu, B., Li, Q., and Jing, L., CNN-based tree species classification using airborne lidar data and high-resolution satellite image. In Proceedings of the IGARSS 2020 (IEEE International Geoscience and Remote Sensing Symposium). **2020**, 2679-2682.
23. Liang, J., Li, P., Zhao, H., Han, L., and Qu, M. Forest species classification of UAV hyperspectral image using deep learning. In Proceedings of the 2020 Chinese Automation Congress (CAC). **2020**, 7126-7130.
24. Nezami, S., Khoramshahi, E., Nevalainen, O., Pölönen, I., and Honkavaara, E. Tree species classification of drone hyperspectral and RGB imagery with deep learning convolutional neural networks *Remote Sensing* **2020**, *12*, 1070 , doi.org/10.3390/rs12071070.
25. Shi, Y., Ma, D., Lv, J., and Li, J. ACTL: Asymmetric Convolutional Transfer Learning for Tree Species Identification Based on Deep Neural Network *IEEE Access* **2021**, *9*, 13643-13654 , 10.1109/ACCESS.2021.3051015.

26. Li, Y., Chai, G., Wang, Y., Lei, L., and Zhang, X., Ace R-CNN: An attention complementary and edge detection-based instance segmentation algorithm for individual tree species identification using UAV RGB images and LiDAR data. *Remote Sensing* **2022**, *14*, 3035 , 10.3390/rs14133035.
27. Chen, X., Jiang, K., Zhu, Y., Wang, X., and Yun, T., Individual tree crown segmentation directly from UAV-borne LiDAR data using the PointNet of deep learning *Forest* **2021**, *12*, 131 , 10.3390/f12020131.
28. Fourier, R.A., Edwards, G., and Eldridge, N.R. A catalogue of potential spatial discriminators for high spatial resolution digital images of individual crowns. *Canadian Journal of Remote Sensing* **1995**, *3*, 285-298.
29. Zhang, K., and Hu, B., Individual Urban Tree Species Classification Using Very High Spatial Resolution Airborne Multi-Spectral Imagery Using Longitudinal Profiles *Remote Sensing* **2012**, *4(6)*, 1741-1757 , 10.3390/rs4061741.
30. Balkenhol, L., and Zhang, K., Identifying Individual Tree Species Structure with High-Resolusion Hyperspectral Imagery Using a Linear Interpretation of the Spectral Signature, Proceedings of the 38th Canadian Symposium on Remote Sensing. **2018**, Montreal, QC.
31. Miao, S., Zhang, K., and Liu, J., An AI-based Tree Species Classification Using a 3D Tree Crown Model Derived From UAV Data, Proceedings of the 44th Canadian Symposium on Remote Sensing, **2023**, Yellowknife, NWT.