A Point-Based Fully Convolutional Neural Network for Airborne LiDAR Ground Point Filtering in Forested Environments

Shichao Jin, Yanjun Su, Xiaoqian Zhao, Tianyu Hu, and Qinghua Guo

***Abstract*—Airborne laser scanning (ALS) data is one of the most commonly used data for terrain products generation. Filtering ground points is a prerequisite step for ALS data processing. Traditional filtering methods mainly use handcrafted features or predefined classification rules with preprocessing/post-processing operations to filter ground points iteratively, which is empirical and cumbersome. Deep learning provides a new approach to solve classification and segmentation problems because of its ability to self-learn features, which has been favored in many fields, particularly remote sensing. In this article, we proposed a point-based fully convolutional neural network (PFCN) which directly consumed points with only geometric information and extracted both point-wise and tile-wise features to classify each point. The network was trained with 37449157 points from 14 sites and evaluated on 6 sites in various forested environments. Additionally, the method was compared with five widely used filtering methods and one of the best point-based deep learning methods (PointNet++). Results showed that the PFCN achieved the best results in terms of mean omission error (T1 = 1.10%), total error (Te = 1.73%), and Kappa coefficient (93.88%), but ranked second for the root mean square error of the digital Terrain model caused by the worst commission error. Additionally, our method was on par with or even better than PointNet++ in accuracy. Moreover, the method consumes one-third of the computational resource and one-seventh of the training time. We believe that PFCN is a simple and flexible method that can be widely applied for ground point filtering.**

***Index Terms*—Digital terrain model (DTM), deep learning, fully convolutional neural network (FCN), ground filtering, light detection and ranging (LiDAR).**

Manuscript received January 30, 2020; revised June 7, 2020 and June 29, 2020; accepted July 7, 2020. Date of publication July 10, 2020; date of current version July 22, 2020. This work was supported in part by the National Key R&D Program of China under Grant 2016YFC0500202 and Grant 2017YFC0503905 and in part by the National Science Foundation of China under Grant 31971575, Grant 41871332, and Grant 41901358. *(Corresponding author: Qinghua Guo.)*

Shichao Jin is with the Plant Phenomics Research Center, Nanjing Agricultural University, Nanjing 210095, China, with the State Key Laboratory of Vegetation and Environmental Change, Institute of Botany, Chinese Academy of Sciences, Beijing 100093, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: jinshichao1993@gmail.com).

Yanjun Su, Xiaoqian Zhao, Tianyu Hu, and Qinghua Guo are with the State Key Laboratory of Vegetation and Environmental Change, Institute of Botany, Chinese Academy of Sciences, Beijing 100093, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China. (e-mail: ysu@ibcas.ac.cn; zhaoxiaoqian@ibcas.ac.cn; tianyuhu@ibcas.ac.cn; guo.qinghua@gmail.com).

Digital Object Identifier 10.1109/JSTARS.2020.3008477

I. Introduction

GENERATING a digital terrain model (DTM) or removing the effect of terrain is a prerequisite for many ecological studies in forested environments [1]–[3]. Light detection and ranging (LiDAR), which is an active remote sensing technology, can measure the distance from the sensor to the target by recording the time between the laser emission and the reception of the reflected light [4]. Because of its higher penetration ability compared with optical images [5] and radar interferometry [6], [7], LiDAR has offered new perspectives on terrain products generation [8]–[10] and their related applications [11]–[14]. Airborne laser scanning (ALS) is highly accurate and efficient, providing useful data for various large scale applications [15]–[17]. However, ground point filtering, as one of the prerequisites for ALS data preprocessing, is the key to generating accurate DTMs, which consumes nearly 80% of the workload of ALS data processing [18]. Meanwhile, given nonground objects with different shapes and sizes, in addition to the terrain surface with various slopes and discontinuities, ALS data filtering can be difficult and troublesome.

Traditional ground point filtering methods can be grouped into three types: Slope-based, surface-based, and segmentation-based methods [19]. Slope-based methods [20]–[23] filter ground points based on the assumption that if the height difference between two points is greater than a defined threshold, then they should be classified differently. Slope-based methods are efficient and simple, but they are not robust when terrain points and object points are mixed unequally, particularly in areas with large terrain fluctuations [1]. Surface-based methods filter ground points by defining a surface that approximates bare earth within a buffer zone, which defines the 3-D space where ground points are expected to reside [24]. Depending on the surface generation method, surface-based methods can be further divided into two subcategories: Morphology-based and interpolation-based methods. Morphology-based methods [25]–[29] approximate the terrain surface using morphological operations (e.g., opening and closing), which are easily implemented, but their performance relies on the setting of a window size [30]. Interpolation-based methods [19], [31]–[34] adapt hierarchical or iterative ideas to densify filtered results, which can manage discontinuous and dense scenes, but a priori knowledge of the terrain and objects is typically required to set optimal parameters in the interpolation process. Segmentation-based methods [35], [36] first segment points into patches and then group these patches into clusters (ground or nonground) based on a set of

predefined rules. Segmentation-based methods are better suited to urban [1] than forested environments, where many patches may be segmented [35].

Overall, these methods have alleviated the task of ALS data filtering to a certain extent, and have proved that the use of LiDAR attributes (e.g., echo width and intensity) is beneficial [37]. However, these filtering methods have some common shortcomings that need to be addressed: they are based on a set of handcrafted rules or statistically derived thresholds, which can result in errors in complex scenes, such as over forested environments; they mostly use multi-scale window sizes or hierarchical iterations to achieve optimized results, which is computationally intensive and time-consuming; and some methods sacrifice the number of ground points to minimize commission errors, although preserving a large volume of ground points is the prerequisite for ensuring the accuracy of the DTM, particularly for applications that are concerned with microtopography [31], [38].

The convolutional neural network (CNN) is a well-known class of deep learning methods that is commonly used in computer vision-related fields [39]. In the field of remote sensing [40], [41], deep learning is mainly used to solve image-based problems (e.g., detection, classification, and segmentation) [40]–[42], but a few studies have demonstrated the advantage of deep learning in processing LiDAR points [18], [41], [43]–[56]. There are a few practical challenges in using CNN for filtering LiDAR points. First, convolutional operations used in structured data (e.g., 1-D sequence sentences/voice, 2-D grid images, and 3-D voxel) cannot be applied directly to irregular points acquired by LiDAR. Second, unordered points require a CNN to be invariant to the data feeding order, which has $N!$ types of permutations with N points. Third, interactions and discontinuities among points require a CNN with the ability to extract hierarchical features from very local to global scales. Finally, transformation invariance requires the result of a CNN to remain unchanged after affine transformations (e.g., rotation and shift).

Currently, there are four types of methods that address the challenges of using deep learning with points: view/surface-based methods, voxel/tree-based methods, point-based methods, and graph-based methods. View/surface-based methods [44] render points into collections of images, and use well-engineered 2-D CNNs to perform classification and retrieval tasks, but they are nontrivial in terms of point classification and segmentation because of the loss of some inherent structure of the 3-D points in the transformation stage. Voxel-based methods [45], [46] transform points into regular 3-D voxel grids, which enables 3-D convolution and feature extraction but causes unnecessarily sparse volumes and computational costs, thereby limiting their application at a large scale. These deficiencies have been partly solved by some tree-based methods [47]–[49], but these are limited by high storage costs in large-scale applications [47]. Point-based methods [50], [51], [57], and [58] perform highly efficient 1-D convolutional operations. They directly consume irregular and unordered points as input and output the classification of the object or segmentation of each point. Graph-based methods [52], [53] combine superpoint embedding using efficient CNNs (e.g., PointNet) and contextual segmentation using edge-conditioned convolution [59] and a gated recurrent unit [60] to capture both fine spatial details and long-range contextual relationships. Although some graph-based methods perform well in point semantic segmentation at a large scale, they introduce unnecessary computation, complexity, and uncertainties in the unsupervised partition of superpoint generation with human-introduced features (e.g., linearity, planarity, scattering, and verticality) [60], [61]. Therefore, from an engineering application perspective, point-based methods are more promising because of their simple, efficient, robust, and expandable characteristics.

The application of point-based methods in ALS point filtering is still rare. Some studies have demonstrated the feasibility of view-based methods in ALS data filtering. For example, Hu and Yuan [18] viewed the classification of each point as a binary image classification problem and achieved satisfactory results. However, view-based methods may lack local information because point rendering is conducted based on the height difference of all points within a fixed and predefined unit. To capture more detailed information, Yang *et al.* [62] improved the rendering method by choosing a unique center coordinate of each unit through eigenvalue calculation. These two view-based methods are both time-consuming and computationally intensive because of the rendering of each point into an image by calculating context information for each point, and were empirical because of the defined rendering rules and unit size. To address this issue, some researchers have attempted to use voxel-based and point-based methods to perform semantic segmentation by exploiting the sparsity often inherent to 3-D data. For example, Schmohl and Sörgel [54] proposed an encoder-decoder architecture with sparse submanifold convolutional networks for efficient ALS point cloud filtering, but it may still have issues of limited resolution and sampling extent due to the voxelization representation at a large scale. Yousefhussien *et al*. [55] presented a multiscale fully convolutional neural network (FCN) modified from PointNet for ALS point semantic segmentation, which consumed terrain-normalized points with spectral information (if available) in urban and rural areas. Although the modified method labeled ALS points with promising results, it operated on terrain-normalized data. A critical problem that we need to solve is how to filter ground points from raw data that are heavily affected by terrain conditions, such as those found in complex and discontinuous forested environments.

In this study, we aim to propose a point-based fully CNN (PFCN) for filtering ground points from ALS data in forested environments. The main challenges we aim to solve are as follows.

1) In forested environment, can we design a network that can directly consume ALS data with low density, high noise, complex terrain variation, and occlusion?
2) In forested environment, can we successfully train a deep learning network by manually preparing massive dataset without free available benchmark dataset?
3) Can we adopt deep learning tricks (e.g., joint loss, multi-task learning and residual learning [63]–[66]) to improve the performance of deep learning network?

To provide a clear illustration of the method, the rest of the article is organized as follows. Section II specifies the architecture of the network and loss function. Section III describes the

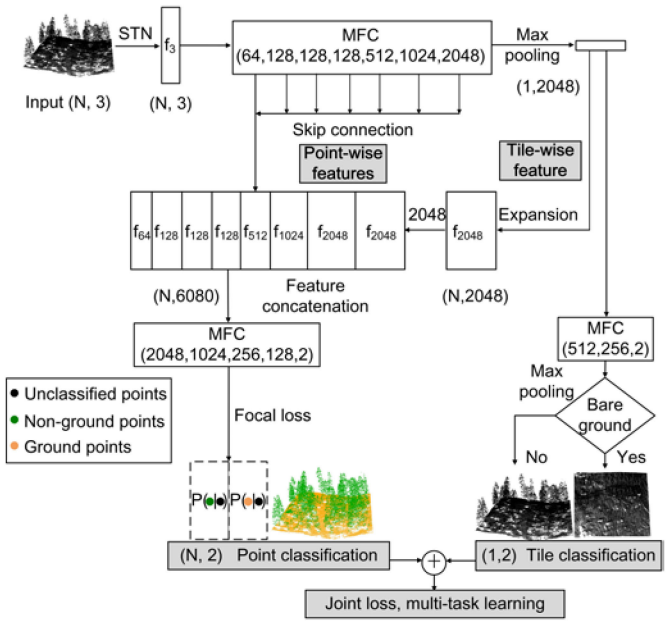


Fig. 1. Architecture of the PFCN. The input layer is the normalized and unclassified point cloud in black. The hidden layers contain an STN and a set of MFCs. Each MFC consists of 1-D convolution operations, followed by 1-D batch normalization and rectified linear unit activation (ReLu). The MFCs extract both point-wise and tile-wise features for a multitask learning, including a point classification task and a tile classification task.

experimental analysis, including study area and data collection, data preparation, network training and testing, accuracy assessment, design of comparison experiments with similar methods, and analysis of tile size/shape influence. Section IV presents the results of the training/validation loss, point classification results, comparison results, and tile size/shape influence analysis results. Although it is interesting to make systematic comparisons among more various traditional methods and deep learning methods that are not included in this article, it may exceed the scope of this article. Finally, Section V presents the discussion of the experiment results and an analysis of the influential factors. Section VI ends with a short conclusion.

II. Methodology

The PFCN is a point-based deep learning method that takes advantage of FCN [67], residual learning [66], multitask joint learning [63], [64], and focal loss (FL) [68], and is trained "end-to-end" to extract hierarchical features. As shown in Fig. 1, the PFCN consists of three parts: input layer, hidden layers, and output layers. The input layer is a point cloud. The hidden layers are composed of a spatial transformer network (STN) [69] and many multilayer fully convolutional (MFC) operations [70], which are designed to extract both point-wise and tile-wise features robustly against unordered data. Point-wise features contain detailed information about each point. Tile-wise features represent global features derived from all points in the tile, which are different from neighborhood features derived from sampling points. The output layers are composed of two tasks through a joint loss, which is trained in a multitask learning way. One main task is the point classification (i.e., classifying each point into a ground or nonground class), which utilizes the concatenated point-wise and tile-wise features. The other associated task is tile classification (i.e., classifying whether the input data/tile is bare ground or not) by using only tile-wise feature.

A. PFCN Architecture

The input layer data is an $N \times M$ matrix, where N is the number of points and M is the number of attributes of each point. Because the *x, y, z* coordinates are the inherent attributes of any 3-D point cloud, the matrix size of the PFCN is $N \times 3$, unless otherwise specified. The raw LiDAR data do not need to be preprocessed to extract some handcrafted features, like with traditional methods. The initial data are sent into the PFCN with a normalization operation using (1), which can accelerate model convergence and enhance model robustness [43], [71]

$$[X, Y, Z] = \frac{[X, Y, Z] - \mathrm{Min}([X, Y, Z])}{\mathrm{Max}([X, Y, Z]) - \mathrm{Min}([X, Y, Z])} \tag{1}$$

where X, Y, and Z are the vectors of the point coordinates in the x, y, and z directions, respectively.

The normalized data are then sent into the hidden layers, which consist of an STN layer and three MFC layers. The STN is a data-dependent network that can align points in a canonical space before feature extraction begins. This can make the network invariant to data transformations, such as shift and rotation [51], [69]. In this article, the STN has multiple layers, and each layer uses a different number of filters that are composed of 1-D fully convolutional operations with 1-D batch normalization and activation

$$\begin{aligned} N_{1024} &= f_{1024}(f_{128}(f_{64}(N_3))) \\ N_9 &= f_9(f_{256}(f_{512}(N_{1024}))) \\ M_{\mathrm{stn}} &= \mathrm{Max}(N_9) + I_3 \end{aligned} \tag{2}$$

where N_x denotes N points with x attributes; $f_x(N_x)$ denotes the N_x data convoluted using x filters, and each 1-D convolution is accompanied by 1-D batch normalization and rectified linear unit activation operations (ReLu); Max is the maximum function that operates in the point dimension; M_{stn} is the learned matrix for the spatial transformation, and I_X is the x-order unit matrix.

The STN outputs a spatial transformation matrix, which is used to multiply the raw input data (matrix) before sending it into the MFC layers. Each MFC layer consists of numerous filters, and each filter performs a 1-D convolution in the feature dimension with 1-D batch normalization and activation. In Fig. 1, the number of filters at each MFC stage is noted between parentheses. The first MFC has seven layers of size 64, 128, 128, 128, 512, 1024, and 2048, and extracts point-wise features of various dimensions. The 2048-dimensional feature is max pooled in the point dimension to obtain a global tile-wise feature vector. On the one hand, the tile-wise feature vector is expanded into an $N \times 2048$ matrix, which means that each point has a global feature size of 1×2048. By concatenating all point-wise features with the expanded global tile-wise features, each point ends up having a 6080-dimensional feature (see Fig. 1). These features are further processed using the second MFC with layer sizes of 2048, 1024, 256, 128, and 2, to do point classification. The last layer of the second MFC outputs an $N \times 2$ matrix,

which represents the probability of each point to be classified as either ground or nonground. On the other hand, to enhance the point classification task, the network trains another task inspired by the multitask learning [63], [64], which is classifying the total points/tile into bare ground or not. In the tile classification task, the max-pooled 2048-dimensional feature is processed by the last MFC. The last MFC has layer sizes of 512, 256, and 2, and outputs a 1×2 vector using the softmax function, which represents two types of probabilities: that the tile is bare ground or not bare ground. However, the tile classification is just an associate task to the training phase and is not used in the output result. The tile classification procedure produces the tile classification loss, which is added to the point classification loss to make a multitask training. The loss functions for tile classification and point classification are described later.

B. Loss Function

The loss function for the PFCN is the sum of the tile classification loss and the point classification loss. For the tile classification, the loss function is the cross entropy (CE), which has been widely used for classification problems because it measures the difference between the prediction and ground truth from the point of view of the probability distribution instead of a simple distance [72]. The tile classification loss (tile_loss) is defined as

$$pt_{\text{tile}} = \begin{cases} p_{\text{tile}} & \text{if} t_{\text{tile}} = 1 \\ 1 - p_{\text{tile}} & \text{otherwise} \end{cases}$$

$$\text{tile\_loss} = CE(p_{\text{tile}}, t_{\text{tile}}) = -\log(pt_{\text{tile}}) \tag{3}$$

where p_{tile} is the output probalility of the tile classification task and t_{tile} is the groud truth label of the tile classification.

For the point classification, we use the FL to eliminate the influence of data distribution imbalance [68], which is a common scenario in very sparse and dense forest scenes. The FL function improves the CE by decreasing the weight of the easy examples and thus focusing on training hard negatives by adding a modulating factor $(1 - pt_{\text{point}})^{\gamma}$ to the cross-entropy loss [68]. The point classification loss (point_loss) is defined as

$$pt_{\text{point}} = \begin{cases} p_{\text{point}} & \text{if} t_{\text{point}} = 1 \\ 1 - p_{\text{point}} & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{CE}(p_{\text{point}}, t_{\text{point}}) &= -\log(pt_{\text{point}}) \\ \text{point\_loss} &= \text{FL}(p_{\text{point}}, t_{\text{point}}) \\ &= (1 - pt_{\text{point}})^{\gamma} \text{CE}(p_{\text{point}}, t_{\text{point}}) \end{aligned} \tag{4}$$

where p_{point} is the output probability of the point classification task, t_{point} is the ground truth label of point classification, and γ is a given parameter, which is set to 0.2 in this article by referring to Lin *et al*. [68].

Finally, the total loss of the multitask network (Loss) is defined by adding the tile classification loss and the point classification loss. Because there are always some gaps in the forest and the penetration ability of ALS is strong, the number of ground points usually occupies a certain proportion of the total points. However, the number of vegetation points may be sparse, which means that a data imbalance of low vegetation ratio is more common. In this case, to further avoid overfitting the point classification task (i.e., all points are predicted to belong to one class) even when FL is used, we define a self-weighted weight for each task in the total loss. The self-weighted total loss can balance the loss in point classification where vegetation is less dense by giving more weights to the simple tile classification loss. The equation is defined as

$$w_0 = \begin{cases} 0.1 & \text{if} \frac{n_1}{(n_0+n_1)} < 0.1 \\ 0.9 & \text{if} \frac{n_1}{(n_0+n_1)} > 0.9 \\ \frac{n_1}{(n_0+n_1)} & \text{elsewise} \end{cases}$$

$$w_1 = 1 - w_0$$

$$\text{Loss} = w_0 \times \text{tile\_loss} + w1 \times \text{point\_loss} \tag{5}$$

where n_1 is the number of ground points, n_0 is the number of nonground points. w_0 and w_1 are the fractions of ground points and nonground points, which are both constrained into the range of 0.1 to 0.9 to avoid any one of them being too small to train.

III. Experimental Analysis

A. Study Area and Data Collection

In this article, four study areas (i.e., Providence, San Joaquin Range, Courtwright Road, and Wolverton and Tokopah from west to east) were selected in the Southern Sierra Nevada Mountains, CA, USA. These areas have large changes in topography and vegetation. Elevation ranges from nearly 0 to 4000 m above sea level. The vegetation type is mixed conifer, and the dominant species, in order of abundance, are white fir (*Abies concolor*), ponderosa pine (*Pinus ponderosa*), and incense cedar (*Calocedrus decurrens*). Additional associated species are black oak (*Quercus kelloggii*) and canyon live oak (*Quercus chrysolepis*). These oak trees are secondary forest species after fire, and there are some shrub-like small oak trees distributed under the canopy.

LiDAR data for these areas were acquired in August 2010 using an Optech GEMINI Airborne Laser Terrain Mapper mounted on a twin-engine Piper PA-31 Chieftain. The pulsing rate and scanning frequency were 100 kHz and 50 Hz, respectively. The scanning angle was $\pm 14°$ and the single swath width was 233.26 m with over 50% swath overlap. The system recorded up to 4 echoes per pulse. The average point density was approximately 10.27 pts/m^2. Additionally, to quantify the vertical accuracy of the LiDAR beam, the data provider (i.e., the National Center for Airborne Laser Mapping) collected 243 checkpoints on roads using a global positioning system mounted on a vehicle. The average vertical accuracy, which was derived by calculating the root-mean-square error (RMSE) of the elevation differences between the LiDAR shot and the nearest checkpoints, was 0.024 m and was homogenous among all sites.

Twenty sites with an area of 500×500 m were chosen from the four study areas (upper right corner in Fig. 2). Sites 1 and 2 were located in Courtwright Road; sites 3–8 were located in San Joaquin; site 9 was located in Providence, and sites 10–20 were located in Wolvert and Tokopah. Statistical information for the vegetation and terrain factors (see Table I) was calculated from a DTM, digital surface model (DSM), canopy height model (CHM), and individual tree segmentation results. For each site, ground points were generated using methods in Section III-B. The classified ground points were used to generate the DTM

TABLE I
STATISTICAL INFORMATION OF THE VEGETATION AND TERRAIN OF THE 20 SELECTED SITES

Site	Canopy cover, %				Tree height, m				Elevation, m				Slope, °				Mean point density, pts/m^2
	Min[1]	Max[2]	Mean	SD[3]	Min[1]	Max[2]	Mean	SD[3]	Min[1]	Max[2]	Mean	SD[3]	Min[1]	Max[2]	Mean	SD[3]	
1*	0	88	30	19	3.8	25.1	10.3	4.8	2590	2728	2671	31	0.0	65.5	17.1	7.0	10.4
2*	13	80	51	12	5.3	21.5	13.1	2.9	2693	2724	2703	5	0.0	53.7	6.0	3.7	12.0
3	0	100	71	22	4.5	60.8	25.9	8.9	1456	1584	1524	29	0.0	59.4	16.8	8.1	9.8
4	50	100	90	9	8.3	36.5	20.6	4.4	1482	1697	1562	46	0.2	65.6	21.9	8.4	14.1
5*	51	100	89	9	7.8	41.4	22.3	5.8	1478	1690	1577	41	0.1	60.4	21.2	8.8	14.0
6	0	100	72	23	3.8	51.3	24.9	8.6	1823	1965	1901	28	0.0	79.0	15.8	8.5	12.0
7	8	100	68	18	4.3	33.1	16.9	4.7	1898	1978	1949	19	0.0	74.6	13.2	7.6	11.7
8	2	94	54	21	4.0	27.7	13.7	5.3	1908	1993	1960	19	0.0	71.8	12.4	7.1	11.3
9	0	86	24	18	3.8	13.6	6.2	1.8	415	442	426	6	0.0	52.0	7.1	4.4	11.1
10	0	94	59	22	3.9	39.2	20.6	7.4	2250	2465	2358	54	0.2	72.7	25.9	8.1	11.0
11	13	92	55	18	4.7	29.0	15.4	5.6	2672	3014	2829	85	1.6	77.7	31.4	6.5	11.8
12	0	91	17	17	3.9	25.3	5.3	1.9	2800	3213	3041	106	0.3	88.3	36.1	14.0	8.5
13	0	61	1	5	6.8	34.0	19.8	6.1	2905	3110	2972	57	0.0	87.1	28.2	18.3	8.0
14*	3	97	57	20	4.3	39.7	16.7	6.9	2090	2468	2251	90	0.1	80.6	33.3	9.9	12.1
15	0	93	29	21	3.9	30.8	9.7	6.4	2176	2727	2446	145	0.3	88.1	41.5	14.2	9.6
16*	0	70	18	15	3.9	15.4	5.8	2.2	2252	2788	2509	130	0.8	84.5	41.0	12.3	9.2
17	0	68	20	14	4.0	16.1	6.3	2.3	2682	2993	2849	87	0.2	85.6	29.2	12.7	12.3
18	0	27	0	1	4.8	16.2	9.6	3.0	3020	3317	3168	60	0.0	87.6	33.5	12.3	9.6
19*	0	83	21	19	3.9	17.5	6.7	3.1	2944	3186	3059	58	0.1	83.0	26.2	10.5	8.3
20	0	0	0	0	0.0	0.0	0.0	0.0	3315	3384	3347	18	0.0	45.8	9.2	4.8	8.5

[a] "Min" is the minimum value of canopy cover, tree height, elevation, and slope.
[b] "Max" is the maximum value of canopy cover, tree height, elevation, and slope.
[c] "SD" is the standard deviation value of canopy cover, tree height, elevation, and slope.
* "*" indicates that a site was chosen as a testing site; otherwise, it was chosen as a training site.

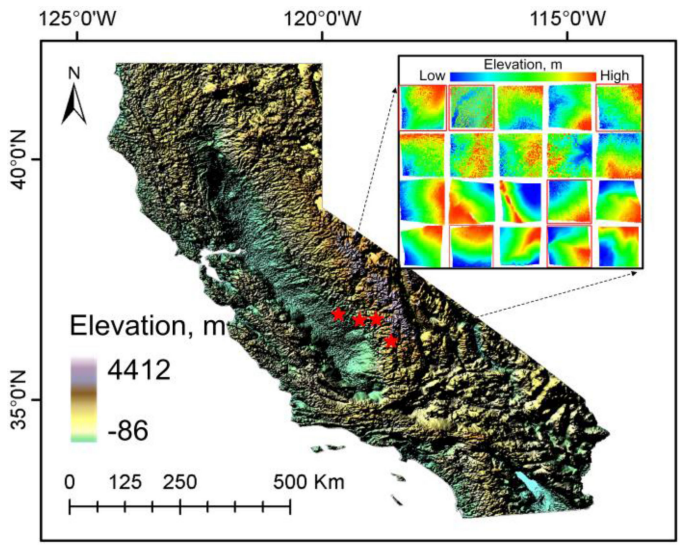


Fig. 2. Four study areas indicated by red stars in the Southern Sierra Nevada, CA, USA. Twenty sites were chosen across these areas, as shown, colorized by elevation, in the black box in the upper right corner. The sites with a red bounding box were used for testing while the others were used for training.

using the most widely used ordinary kriging method [38], [73] in the ESRI ArcGIS software with default parameter settings. Of which, the semivariogram used the *stable* model, whose parameters are automatically fitted using weight least squares methods. The minimum/maximum number of neighboring points were set as *auto*, the neighbor type was set as *standard* without smooth, the sector type was set as *4 sectors with 45° offset*, and the angle/major-semiaxis/minor-semiaxis were copied from the variogram parameters. The DSM was generated using the same method, but with the first return points. The CHM was derived by subtracting the DTM from the DSM, and the spatial resolutions of the DTM, DSM, and CHM were all set to 0.5 m. A 0.5 m resolution was chosen because we found that the RMSE of LiDAR derived-DTM decreased quickly when the spatial resolution varied from 10 to 0.5 m, and stayed relatively stable after 0.5 m using the similar LiDAR dataset (collected with the same equipment with similar point density) in the Sierra Nevada mountains [38]. The canopy cover was derived using a canopy-based method [74], calculating the ratio of pixels with a CHM value higher than 2 m in each patch of 30 × 30 m. Finally, we calculated the minimum (Min), maximum (Max), mean, and standard deviation (SD) of all canopy cover values for each site.

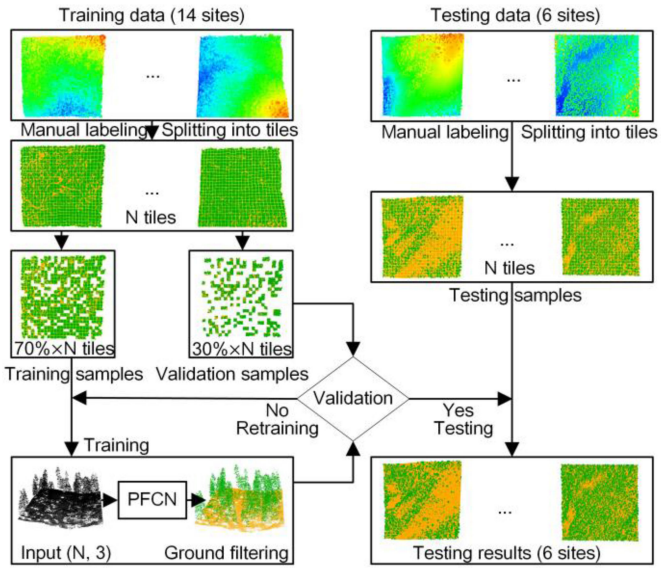


Fig. 3. Flowchart of data preparation, network training, validation, and testing.

Statistical information for the elevation was calculated using the DTM directly. Statistical information for the slope was calculated using the slope raster generated from the DTM raster. Tree height was calculated from the individual segmentation result from LiDAR data using Green Valley International LiDAR360 software [75].

The statistical information showed that the vegetation and terrain conditions in these sites were complex. The mean canopy cover of these sites ranged from 0% to 90% with an SD range of 0%–23%. Meanwhile, the mean tree height of these sites ranged from 0to 25.9 m, with an SD range of 0–8.9 m. The large range of the canopy cover and tree height is representative of most forest scenes. Additionally, the mean elevation and slope range of these sites were 426–3347 m and 6.0°–41.5°, respectively, which covers very flat and steep terrain in various mountain types. Moreover, the mean point density ranged from 8.0 pts/m^2 to 14.1 pts/m^2.

B. Data Preparation (Training, Validation, and Testing)

Well-labeled training data are the prerequisite for training a satisfactory model. In this article, 14 sites (70% of all sites) were selected for training (see Fig. 3), which covered various vegetation and terrain conditions, and with canopy cover and slope ranging from 0% to 90% and 7.1° to 41.5°, respectively. The unclassified points were filtered to obtain preliminary results using the automatic method in the TerraScan software. The preliminary results of each site were checked manually to eliminate incorrectly filtered points and increase the number of ground points as much as possible. Manual checking was performed by visualizing the cross section of the point cloud with the LiDAR360 software, which can easily identify mistakes and reclassify incorrect points [76], [77]. The standard and detailed procedures can be found at our previous study [78]. Information about the final labeled data and the ratio of manually corrected points is given in Table II.

TABLE II
INFORMATION ABOUT THE CLASSIFIED GROUND AND NONGROUND POINTS OF ALL SITES OBTAINED BY VISUALLY CHECKING AND REVISING THE DATA CLASSIFIED BY THE AUTOMATIC FILTERING RESULTS USING TERRASCAN AND LIDAR360 SOFTWARE

Site	Non-ground points	Ground points	All points	Ratio of ground	Manually Corrected ratio,%
1*	801184	1790837	2592021	0.69	14.44
2*	1316392	1679836	2996228	0.56	15.53
3	1640373	837296	2477669	0.34	29.08
4	2622421	912013	3534434	0.26	27.81
5*	2640707	861903	3502610	0.25	25.80
6	2079881	945601	3025482	0.31	10.99
7	1771128	1146426	2917554	0.39	10.20
8	1413681	1418843	2832524	0.50	13.18
9	697704	2079773	2777477	0.75	16.91
10	1580568	1164838	2745406	0.42	14.83
11	1447138	1503428	2950566	0.51	13.71
12	66710	2061090	2127800	0.97	25.44
13	35671	2011533	2047204	0.98	0.12
14*	1515462	1504245	3019707	0.50	24.50
15	521078	1873376	2394454	0.78	33.61
16*	193757	2096175	2289932	0.92	43.03
17	343791	2730967	3074758	0.89	77.79
18	2816	2420110	2422926	1.00	0
19*	350419	1719166	2069585	0.83	21.89
20	0	2120903	2120903	1.00	0

* "*" indicates that the site was chosen as a testing site; otherwise, it was chosen as a training site.

Because each site was 500 × 500 m, hardware limitations made it difficult to input such a large scene as a single training sample. We therefore increased the training samples by tiling each classified training datum (site) into 62 20 × 20 m tiles. Each tile was labeled with two types of labels for the two training tasks (i.e., tile classification and point classification). For the tile classification, the tile was labeled 0 if the tile was bare ground (i.e., all points were ground points); and 1 otherwise. For the point classification, each point of the tile was labeled as a ground point (0) or nonground point (1). The labeled data were normalized to create a training sample using (1). Finally, 12500 tiles were labeled as ground/nonground samples, of which 8750 samples were labeled from 14 training sites and 3750 samples were labeled from 6 testing sites. Moreover, 70% (6125) of the training samples (8750) were selected for training and the remaining 30% (2625) were reserved for validation to select the best model in the training process. The 3750 testing samples were used for accuracy assessment.

The best model saved in the training stage was used for testing. Points in each tile of a site were predicted separately and then merged to obtain the final result.

C. Network Training and Testing

The PFCN was trained "end-to-end" using strong GPU acceleration in the PyTorch framework [79]. The 6125 training samples were fed into the network using a batch size of 1 because the number of points in each training sample was not

TABLE III
ACCURACY ASSESSMENT METHOD FOR POINT CLASSIFICATION

		Target	
		Non-ground (0)	Ground(1)
Prediction	Non-ground (0)	TN (True negative)	FN (False negative)
	Ground (1)	FP (False positive)	TP (True positive)
Evaluation metrics, where S=(TN+FN+FP+TP)	T1 = FN/(FN +TP)		
	T2 = FP/(TN+FP)		
	Te= (FN+FP)/S		
	P0 = (TN+TP)/S		
	$Pc = ((TN+FN)\times(TN+FP)+(FN+TP)\times(FP+TP))/S^2$		
	Kp = (P0-Pc)/(1-Pc)		

the same. After each epoch, the loss for tile classification and point classification was calculated using cross-entropy and FL, respectively. The two losses were added to obtain the total loss, which was optimized using the Adam method through back propagation with a time-dependent learning rate [80]. The initial learning rate was 0.0001, which was halved every 20 epochs when the epoch was less than 100. To monitor the best model, we validated the model using validation samples after each epoch. In the validation stage, 2625 samples were predicted to obtain the total loss and overall accuracy between the prediction and target. If the total validation loss decreased and the overall accuracy increased over the last epoch, then the model of this epoch was saved. The initial total loss and overall accuracy were set to 1000%, and 0%, respectively. The network was trained until the validation loss did not decrease and the overall accuracy did not increase over 50 consecutive epochs. Finally, the last saved model was considered the best model.

D. Accuracy Assessment

The point classification accuracy was evaluated at the point level using cross matrices of type I error (T1), type II error (T2), total error (Te), and Kappa coefficient (Kp) (see Table III) [24], [81], [82]. T1 error indicates the omission of target ground points (i.e., the proportion of ground points misclassified as nonground points). T2 error indicates the commission of nonground points (i.e., the proportion of nonground points misclassified as ground points). Te is the total proportion of misclassified points. Kp determines the point classification accuracy. Low T1, T2, Te, and high Kp are expected for a good result.

Additionally, the RMSE of the DTM was evaluated at different resolutions: 0.5, 1, 5, and 10 m. The DTM of both the prediction and ground truth was generated using the most widely used Kriging method [38]. The ground truth points were prepared using methods in Section III-B. At each resolution, the RMSE between the ground truth DTM and predicted DTM was calculated. Meanwhile, by averaging the RMSE of all sites, the mean RMSE of different sites at each resolution was calculated. Moreover, by averaging the RMSE of all resolutions, the average RMSE was calculated.

E. Comparison with Traditional Methods and PointNet++

To show the performance difference between the PFCN and traditional ALS filtering methods, five widely used traditional methods were selected for quantitative comparison, which included a slope-based method [20], a morphology-based method [27], and three interpolation-based methods [31], [83], [84]. Traditional methods were chosen instead of the latest methods for the following reasons:

1) traditional methods are stable and widely used;
2) they share key ideas with the latest algorithms [19], [33], [85], [86]; and
3) their performances are on par with or better than the latest methods [87].

The slope-based method (hereafter SAGA) is based on a height difference assumption and is implemented in the open-source SAGA software [20]. The morphology-based method (hereafter SMRF) uses a simple morphological filter with a linearly increasing window and simple slope threshold to classify ALS data, which is suited for DTM generation because of the lower ground point omission error [27]. The first interpolation-based method is based on adaptive TIN models (hereafter TerraScan), which have been implemented as a package in the commercial TerraSolid software [84]. The second interpolation-based method is a multi-scale curvature algorithm designed for forested environments (hereafter MCC), which retains a high ratio of ground points and low commission errors to derive a highly accurate ground surface [83]. The last interpolation method (hereafter Fusion) is also designed for wooded areas, and is implemented in the open-source Fusion software [31], [88]. The parameter setting of each method was determined by exhausting all possible parameter combinations, and we used the parameter setting with the optimized filtering results in each study site (the lowest Te) for comparison. The detailed information of parameter settings of Fusion, MCC, SAGA, and TerraScan can be found at our previous study [78]. The SMRF method has five parameters: cell size, slope, maximum window radius, elevation threshold, and scaling factor for ground identification. To find the optimal parameter setting for comparison, the cell size was fixed at 1 m according to [27]. Slope was changed from 5% to 100% with an increment of 5%, and with an increment of 1% during a fine-tune optimization. Maximum window radius was varied from 1 to 20 m with an increment of 1 m. Elevation threshold was varied from 0 to 1.0 m with an increment of 0.1 m. Elevation scaling factor was varied from 0 to 2.5 with an increments of 0.05.

In addition, we analyzed whether PFCN is as accurate as state-of-the-art deep learning-based filtering methods and whether it has additional strengths. A well-known point-based deep learning method, PointNet++, was selected for comparison, because it captures both local and global features and has been shown to have state-of-the-art performance in point semantic segmentation [50]. PointNet++ was also trained using the same training dataset used in PFCN and monitored by the validation dataset until the validation loss did not decrease and the overall accuracy did not increase over 50 consecutive epochs. The hyper-parameters (e.g., search radius and number of sampling point) used in PointNet++ were set as default as in [50].

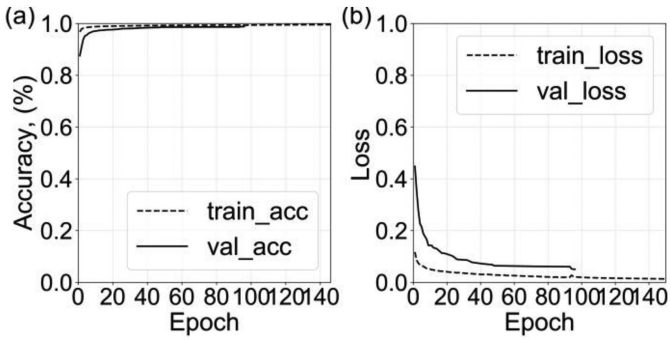


Fig. 4. Overall accuracy and loss of training and validation during the training process.

F. Influence Analysis of Input Tile Size and Tile Shape

Splitting a large scene into small tiles for training is a necessary step because of computational limitations. Splitting is also a simple approach to perform data augmentation, which can enhance the robustness of the model [89]. Moreover, the global feature of the tile that has been expanded for each point can be considered as local information used in traditional methods to improve filter performance [24]. However, the tile size and shape may affect model performance because previous studies have indicated that features are related to target and input data sizes [1], [24].

To analyze the influence of tile size, we compared the performance of the PFCN with training samples of different tile sizes: 5×5 m, 10×10 m, 15×15 m, 20×20 m, 25×25 m, and 30×30 m. Sizes smaller than 5×5 m were not included because a size that is too small cannot provide sufficient information; sizes that are larger than 30×30 m were also excluded because of computational limitations.

In addition to the tile size, the tile shape (i.e., width and length) may also affect the performance of neural networks. Therefore, we analyzed the performance of the PFCN with training samples of different shapes but same area: 20×20 m, 10×40 m, 5×80 m, 2×200 m, and 1×400 m.

IV. Result

A. Training/Validation Loss Decreasing

The PFCN was trained to converge with a total number of training epochs of 146 and training time of approximately 43 h on a Windows 10 server with Intel(R) Xeon(R) CPU E5-2640 v4 @2.40GHz, 256 GB RAM, and an NVIDIA Tesla P100-PCIE-12GB GPU. The trends for the training loss, training accuracy, validation loss, and validation accuracy are shown in Fig. 4.

B. Point Classification Results

The visual results of the whole point classification are shown in Fig. 5. There are no obvious visual differences in the ground point filtering results when compared with the ground truth. The difference between the ground truth and prediction was calculated and is shown in the right column in Fig. 5. The matched points are in gray and the misclassified points are in red. As can be seen, the misclassified areas appear mainly in Figs. 5(d) and (e). Most of the misclassified points are discretely distributed, apart from some misclassified points that appear in a continuous area in Fig. 5(e). Moreover, the misclassified points mostly appear at the junction of terrain and vegetation, as can be seen from the difference (profile) figures in the right column in Fig. 5. The quantitative results are described in Section IV-C.

C. Comparison With Traditional Methods and PointNet++

To quantitatively evaluate the performance of the PFCN, evaluation metrics (i.e., Kp, T1, T2, and Te) were calculated and compared with traditional filtering methods. The results showed that the PFCN had the highest mean Kp value (93.88%) over all sites, followed by SMRF, MCC, TerraScan, Fusion, and SAGA [see Fig. 6(b)]. Meanwhile, the mean SD of Kp of PFCN in all sites was the lowest (6.03%), followed by SMRF, MCC, TerraScan, Fusion, and SAGA. Additionally, most of the methods performed relatively weakly in site 16 and site 5 [see Fig. 6(a)], but the PFCN was always the best in each site.

In addition to the Kp accuracy, we also evaluated the performance of the PFCN and traditional methods using the T1, T2, and Te metrics (see Fig. 7). The results showed that the PFCN had the lowest mean T1 error and highest mean T2 error. However, the mean Te of the PFCN was the lowest among all methods, followed by SMRF, MCC, TerraScan, Fusion, and SAGA, which is consistent with the results of the Kp metrics.

Moreover, the mean RMSE of the DTM products showed that the RMSE of each method was almost unchanged or slightly increased as the resolution became coarser: from 0.5 to 10 m [see Fig. 8(a)]. Additionally, the average value of the mean RMSE of the DTM at different resolutions of each method is shown in Fig. 8(b), which indicates that the PFCN ranked second, followed by SMRF, MCC, FUSION, and SAGA. The best RMSE result of the DTM was TerraScan, which had a 0.05 m lower RMSE than the PFCN.

Besides the comparisons with traditional methods, we found that the Kp and Te values of the PFCN and PointNet++ methods were almost the same (see Fig. 9). Meanwhile, we found that the Mean RMSE of the DTM of the PFCN method was slightly better than that of the PointNet++ method. Moreover, the training (including validation stage) time and GPU resources of each epoch of the PFCN and PointNet++ methods were 0.295 and 2.142 h, and approximately 3 GB and 9 GB, respectively. Additionally, for each sample with a size of 20 m $\times$ 20 m, the mean testing time of PFCN and PointNet++ methods were 0.033 s and 0.899 s, respectively.

D. Influence of the Input Tile Size and Tile Shape

The mean values of T1, T2, Te, and Kp, and Mean RMSE of the DTM show the specific influence of the tile size in Fig. 10. The mean Te and Kp, and Mean RMSE of the DTM of different tile sizes were very similar, of which the 20×20 m tile size had the lowest Te and highest Kp, and a 0.25 m RMSE that was very close to the minimum. Meanwhile, the 20×20 m tile

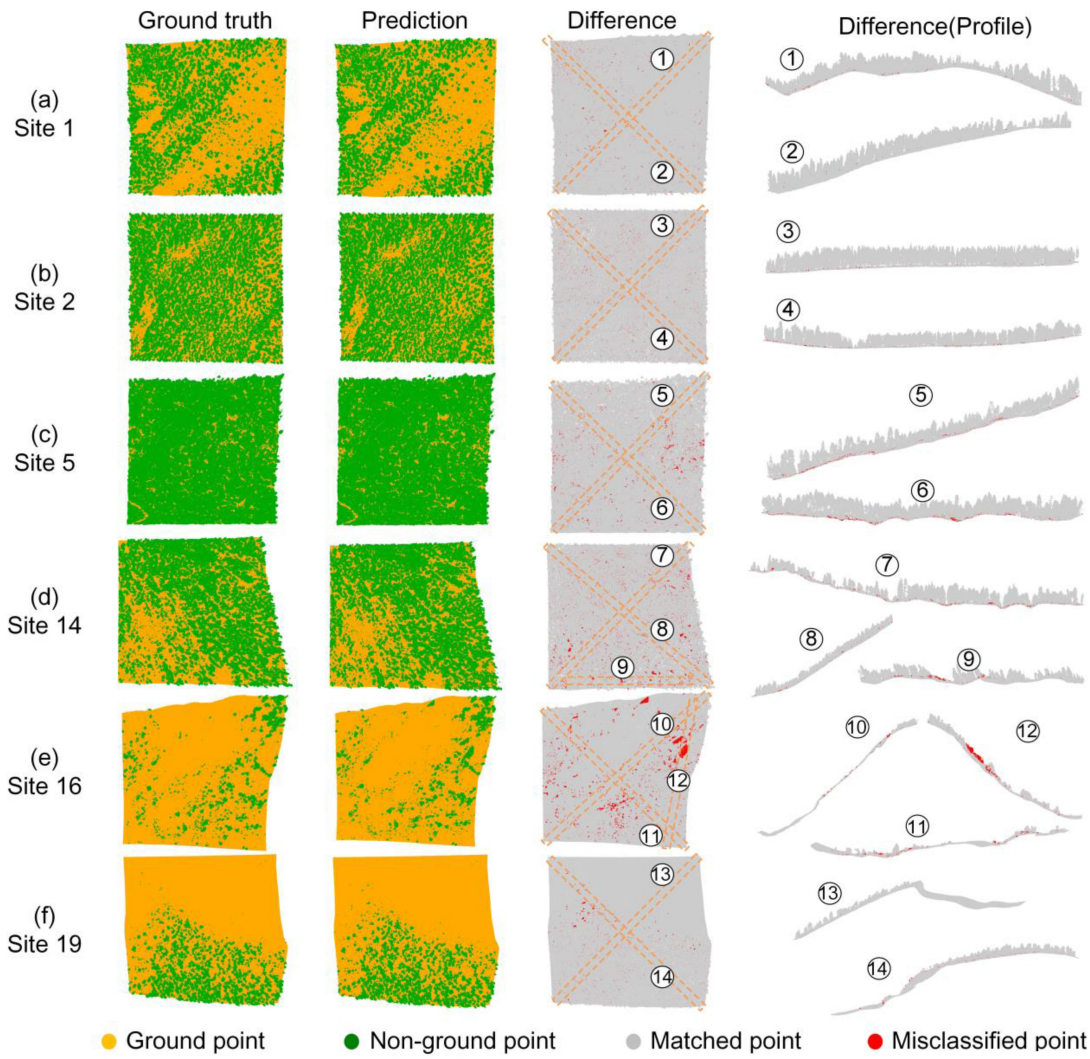


Fig. 5. Point classification result of the PFCN. Subfigures, from left to right in each row, are the ground truth, prediction, difference between the ground truth and prediction, and difference in 20 m interval profile between the ground truth and prediction. Circled numbers indicate the location of the difference profile. (a)–(f) Six testing sites.

size had the lowest T1 with a middle T2 value. However, the influence of tile size in all accuracy metrics was insignificant at the 95% confidence level using Tukey's multiple comparison method [90].

V. Discussion

A. Comparison with Traditional Methods and PointNet++

The comparison of the point filtering methods showed that the PFCN performed best according to all the metrics except T2, followed by surface-based methods (i.e., SMRF, MCC, TerraScan, and Fusion) and sloped-based methods (i.e., SAGA). The main reason may be that the PFCN captured both point-wise and tile-wise features, which were learned automatically from millions of points that covered various complexities. Additionally, the PFCN adopts many ideas from other current state-of-the-art methods, such as using FL to avoid data imbalance [68], using residual learning to improve the robustness of the model [66], and using multitask learning to improve performance [63]. The reason that surface-based methods were better than slope-based methods was that they captured more context information [24]. Among these surface-based methods, the SMRF method is based on morphology operations, which are effective at minimizing the T1 error on highly varied terrain [27] Meanwhile, the morphology-based methods that operate on the surface also proved to be less sensitive to slope, vegetation conditions, point density, and other factors [27], [87], which may explain why SMRF was better than the other three surface-based methods using the interpolation surface. However, the interpolation-based methods (i.e., MCC, TerraScan, and Fusion) proved to be better than slope-based methods [78]. MCC has been specifically developed for forested environments, and is more robust to changes in terrain and canopy cover because it balances T1 and T2 errors [83]. By contrast, TerraScan tended to flatten hilltops and resulted in a higher T1 error [78], which may be why MCC performed better than TerrasScan. Meanwhile, MCC outperformed Fusion [87] and TerraScan outperformed

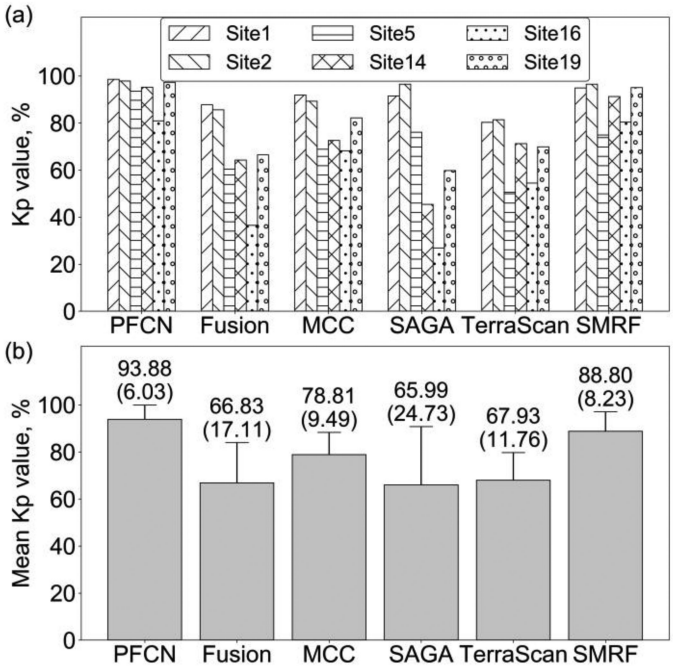


Fig. 6. Comparison of Kp of PFCN and traditional filtering methods, including Fusion, MCC, SAGA, TerraScan, and SMRF. The value above each bar represents the mean value, and the value in parentheses represents the SD.

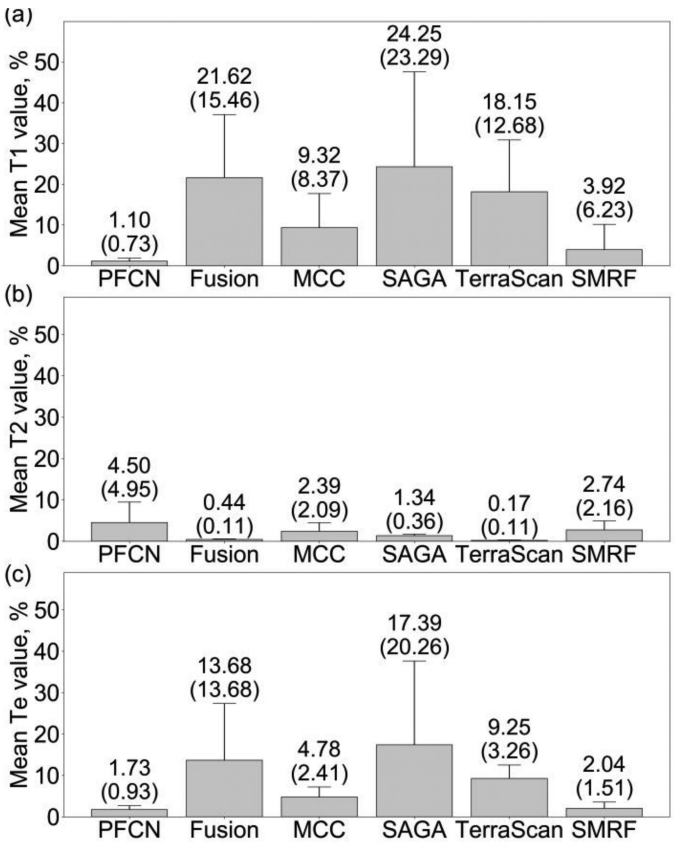


Fig. 7. Comparison of the error metrics (i.e., T1, T2, and Te) of the PFCN and traditional filtering methods, including Fusion, MCC, SAGA, TerraScan, and SMRF. The value above each bar represents the mean value, and the value in parentheses represents the SD.

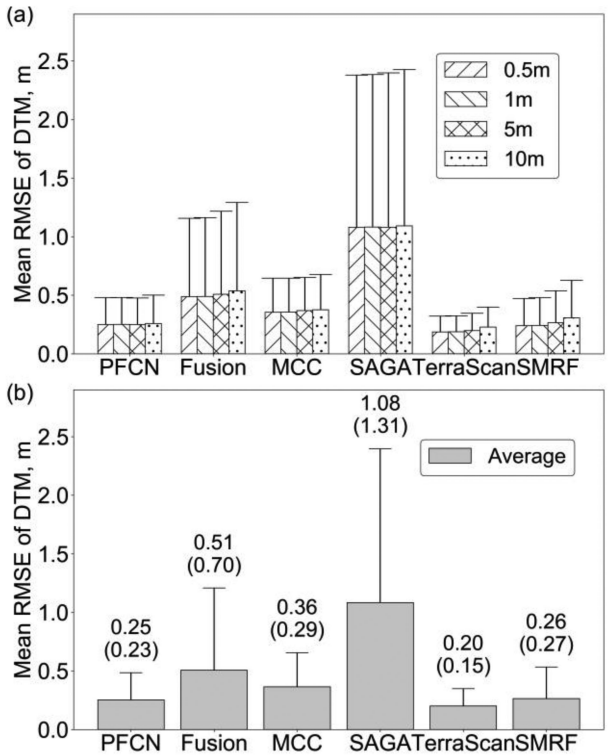


Fig. 8. Comparison of mean RMSE of DTM of the PFCN and traditional filtering methods, including Fusion, MCC, SAGA, Glidar, IPTD, PTDF, and SMRF. The value above each bar represents the mean value, and the value in parentheses represents the SD.

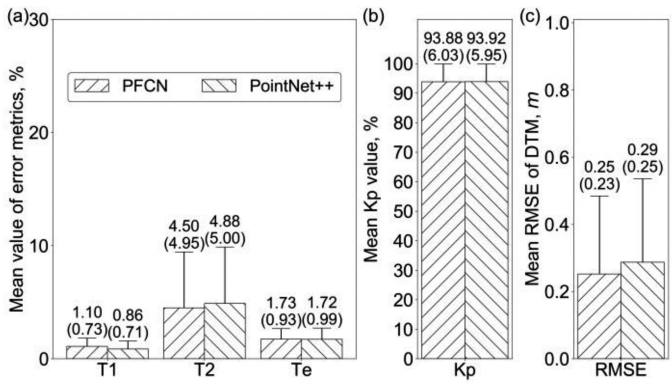


Fig. 9. Comparison of the PFCN and PointNet++ methods using (a) T1, T2, and Te, (b) Kp, and (c) mean RMSE of the DTM metrics at 0.5, 1, 5, and 10 m resolutions. The value above each bar represents the mean value, and the value in parentheses represents the SD.

FUSION, particularly when the canopy cover was high [78]. Additionally, these surface-based methods outperformed the slope-based method adopted by SAGA, which is also consistent with previous studies [78], [87], [91].

The lowest Kp value was for sites 5 and 16 for almost all methods. This may be related to the site conditions: site 5 had a large mean canopy cover of 89% and site 16 had a large mean slope of 41.0°. This result is supported by previous studies in forested environments, in which the canopy and slope conditions were two important factors that affected filtering accuracy, and a

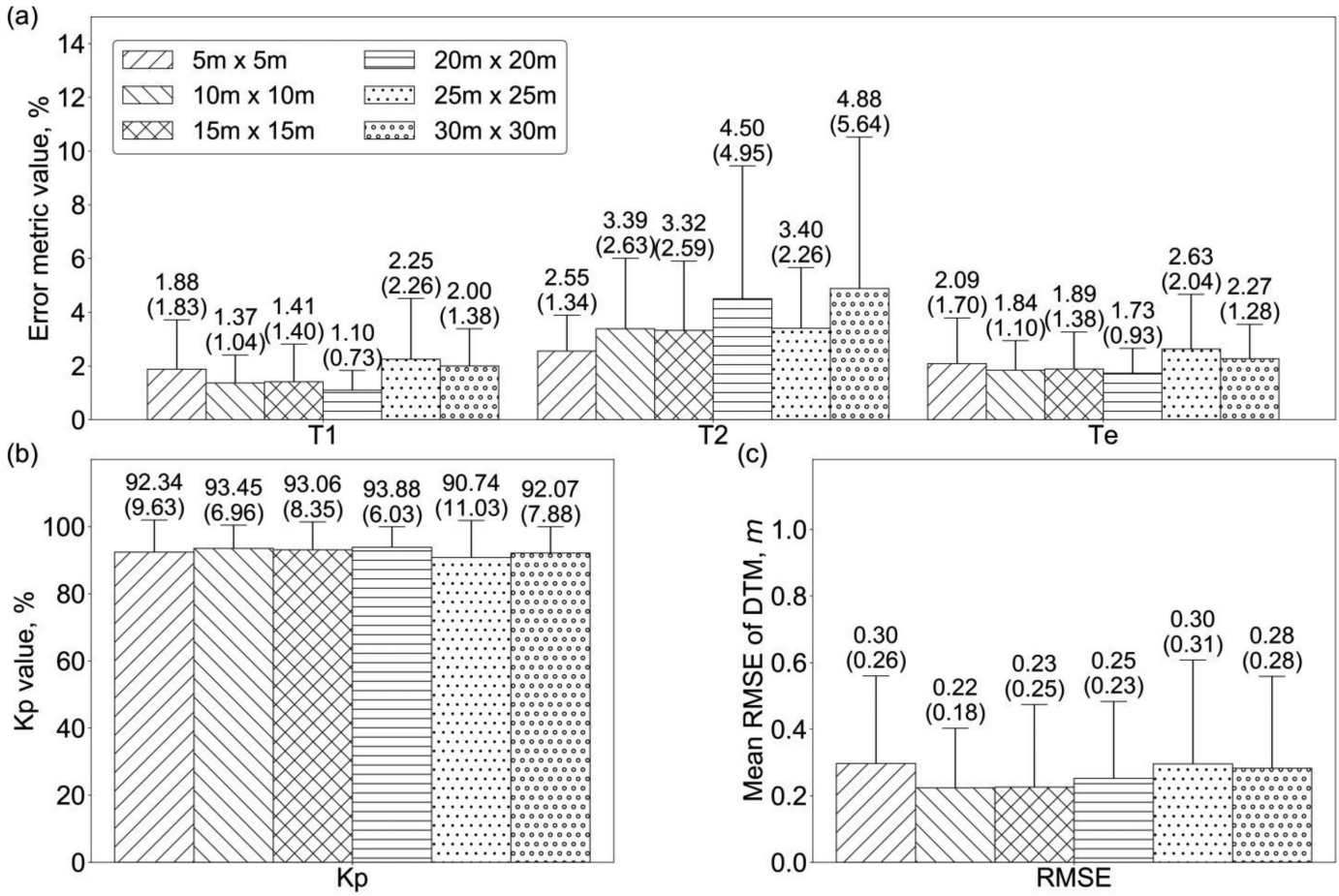


Fig. 10. Influence of tile size of the same shape indicated by (a) T1, T2, and Te, (b) Kp, and (c) mean RMSE of the DTM generated at four kinds of resolutions (i.e., 0.5, 1, 5, and 10 m) at each tile shape. The value above each bar represents the mean value, and the value in parentheses represents the SD. There are no significant differences between the different tile sizes in all accuracy metrics at the 95% confidence level using Tukey's multiple comparison method. The mean values of T1, T2, Te, and Kp, and mean RMSE of the DTM illustrate the influence of tile shape in Fig. 11. The mean T1, Te, and Kp, and mean RMSE of the DTM of square size 20 × 20 m were the best, while the T2 value of square size 20 × 20 m was middle. The Kp value decreased when the tile shape narrowed, except for the 5 × 80 m shape. This exception also existed in the error metric values (T1, T2, and Te), but there was no exception from the point of view of the Mean RMSE of the DTM.

larger canopy cover and slope value resulted in larger errors [1], [78]. Meanwhile, we found that the slope-based method, SAGA, was more heavily affected by slope, which is also consistent with previous studies [78]. This may explain why the Kp of site 4 (slope = 33.3) for the SAGA prediction was also low. Although the PFCN method was also affected by slope and vegetation factors, the Kp value was 93.56%, even under 89% canopy cover, and the Kp value was greater than 80%, even in the steep hill area with a mean slope of 41.0°. According to [82], a Kp value above 0.75 generally reflects excellent agreement, which indicates that the PFCN was in excellent agreement with the ground truth.

Besides the Kp value, the PFCN performed best in terms of Te. The main reason is that the PFCN was optimized by both the cross-entropy and FL to obtain the maximum overall accuracy, which tended to classify all possible ground points to minimize both the T1 and T2 errors. By contrast, the traditional methods had a strategy to select a set of ground points by iteratively densifying ground points based on some critical points, such as the lowest points in a series of local neighbors [84]. Therefore, the traditional methods had a higher omission error (T1) but lower commission error (T2) compared with the PFCN. The higher T2 problem was also mentioned for previous deep learning-based methods [92]. Although the T1 error can be easily interpolated, whereas the T2 error requires additional filters, a high T2 error is preferred over a high T1 error because it is easier to amend when manual checking is unavoidable in practice [81]. Meanwhile, higher T1 errors result in gaps in the landscape, and it is difficult and costly to determine whether a gap is caused by T1 errors or the removal of objects [24], [78]. The gap issue should be solved by reducing the Type I error especially for the ALS data because it has relatively low point density, and generating high-resolution DEM (e.g., 0.5–1m) needs high ground point density [38]. In this article, the T1 of the PFCN was the lowest. Meanwhile, because all T2 errors (0.17%–4.50%) were much smaller than T1 errors (1.10%–24.25%), the Te of the PFCN was still the best.

Additionally, a higher T2 error may result in a higher RMSE of the DTM, which is the reason why the mean-RMSE of the DTM of the PFCN ranked second among all methods. However, they

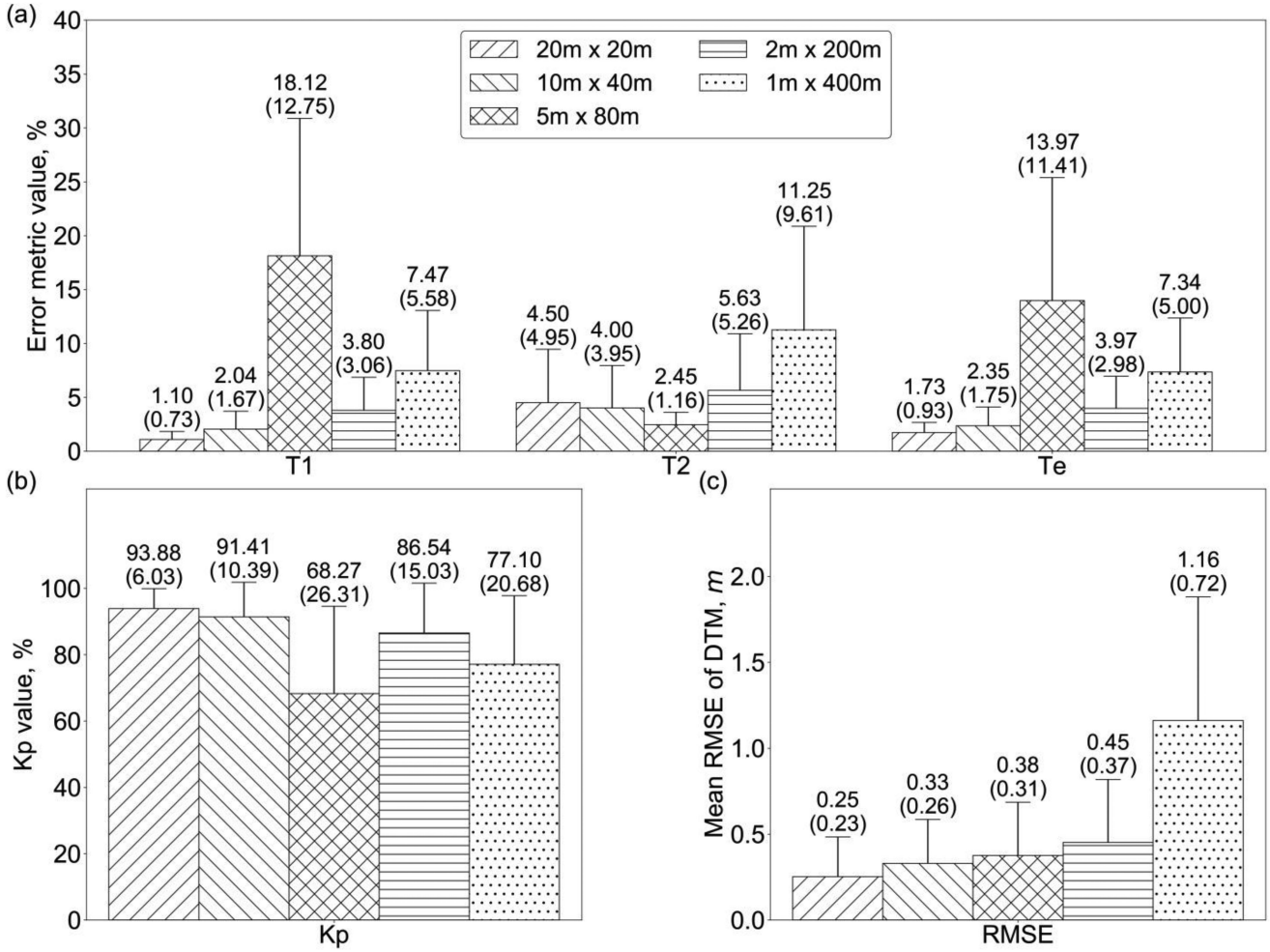


Fig. 11. Influence of tile shape with the same area indicated by (a) T1, T2, and Te, (b) Kp, and (c) mean RMSE of the DTM generated at four kinds of resolutions (i.e., 0.5, 1, 5, and 10 m) at each tile shape. The value above each bar represents the mean value, and the value in parentheses represents the SD. There was no significant difference between the different tile shapes in all accuracy metrics except T1 at the confidence level of 95% using Tukey's multiple comparison method. The significance of the differences in tile shape influence on these metrics is shown using letters a and b. If two groups do not have the same letter, they have significant difference

were comparable to the best results generated by the TerraScan method with a 0.05 m difference considering the average vertical accuracy of the ALS data. In practice, most ALS data have a resolution of approximately 0.15 m vertically and 0.3 m horizontally under the best conditions [93], [94]. For the DTMs, the T1 and T2 errors caused the same absolute elevation errors [24], [78]. In the case of comparable accuracy, it is necessary to ensure overall filtering accuracy (Te and Kp) to obtain as many ground points as possible. The PFCN was the method that achieved this purpose, which has also been reflected in recent studies [27]. In addition, similar to traditional methods, the PFCN was able to work with dataset of different point densities, terrain types, and vegetation conditions (Appendix Fig. 14).

From the comparison of PFCN with PointNet++, we found that the PFCN was on par in terms of Te and Kp, and slightly better in terms of the mean RMSE of the DTM product. More importantly, the PFCN provided much more efficient ALS filtering in terms of both time and resource consumption. These strengths may benefit from the point-wise and tile-wise loss function as well as the multitask network architecture, which is discussed in Section V-B.

B. Network Architecture

To increase the performance and robustness of PFCN, we designed a concatenated feature that combines both point-wise and tile-wise features. In the point classification task (Appendix Table IV), the only point-wise feature works well, but the only tile-wise feature does not work. However, combining the tile-wise feature with the point-wise feature can improve the mean overall accuracy (i.e., Kp) and decrease the mean total error (i.e., Te) of point classifications. The reason why the combined loss decreases the accuracy of Mean RMSE of DTM at sites 14 and 16 may be that their slopes are both too steep to learn a good tile-wise feature by PFCN.

Comparing the slight improvement of multiwise feature, the improvement of multitask training is more considerable. (Appendix Tables V and VI). It improves the tile classification

accuracy at 5 of the 6 testing sites as well as the mean tile classification accuracy (Appendix Table V) when compared with only using the tile classification task. Meanwhile, multitask improves the point classification accuracy at all testing sites in terms of Te and Kp, and decreases the RMSE of DTM by 8 cm on average (Appendix Table VI) when compared with only using the point classification task. The finding that multitask deep learning contributes to the performance of each task is consistent with previous studies [64], [95], [96].

The reason why multitask learning does not bring a tremendous improvement to the only point classification task (Appendix Table VI) may be that the multitask loss is designed to solve the misclassification of low vegetation cover in steep areas, as in the example in Fig. 12. In this case, the improvement of Kp and Te is not obvious, because the proportion of corrected points is small. However, the improvement is obvious in DTM, in which it greatly corrects the local overestimation problem (see Fig. 12). This is why the improvement of the RMSE of the DTM is greater than Te and Kp (Appendix Table VI). Through further analysis of the improved samples, we find that the improvements are mainly concentrated on areas with a small proportion of vegetation points and areas with large slopes (see Fig. 13). This further proves the validity of the proposed loss function. Because only a few of all test samples in each site belong to the abovementioned case, the overall improvement for each site is not very obvious.

C. Influence of the Input Tile Size and Tile Shape

From the analysis of tile size influence in Fig. 10, we found that 20 × 20 m is an optimal tile size in terms of the lowest Te and highest Kp, but 10 × 10 m may be a better choice for a highly accurate DTM. Additionally, although the performances of the PFCN with different tile size input were similar, there was a slight tendency for the RMSE and T2 to increase with the increase of the tile size. A possible reason for this is that the global feature of the tile contributes less to finer classification, whereas the global feature of small tiles may better represent the local neighbor information about each point. Moreover, a tile size that is too small (less than 5 × 5 m) may result in an increase of the RMSE and Te because of the lack of sufficient local information when comparing the result of 5 × 5 m with that of 10 × 10 m.

Additionally, a potential law of tile shape influence appears to indicate that as the tile shape becomes narrower, the accuracy decreases (see Fig. 11). It is difficult to analyze the reason for the exception appeared at 5 × 80 m shape using deep learning methods, but we can see that the Mean RMSE of the DTM increased gradually when tile shape narrowed from 20 × 20 m to 1 × 400 m. This means that a narrower area is not conducive to accurate point classifications. A narrower area may contain only a very small amount of vegetation point data (see Fig. 12), which makes the tile classification more difficult (see the tile classification accuracy of site 16 in Appendix Table V), and further affects the result of multitask PFCN.

D. Contributions and Future Work

The PFCN is a point-based method based on the PointNet backbone [51], and integrates a series of state-of-the-art deep

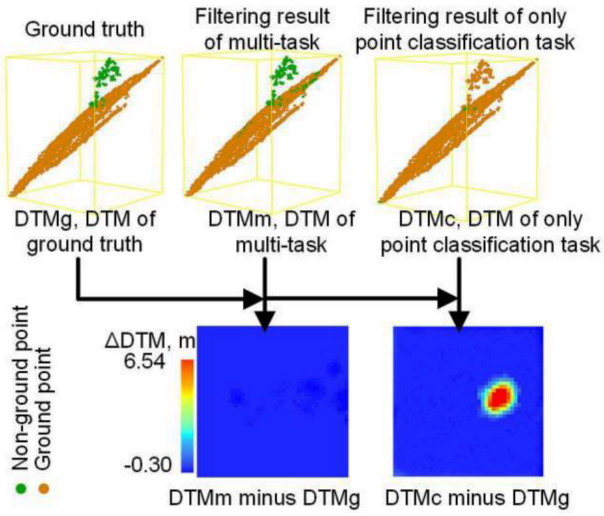


Fig. 12. Example when multitask network usually predicts better results than the only point classification task network. In such cases the percentage of vegetation points is very low and the slope is very steep.

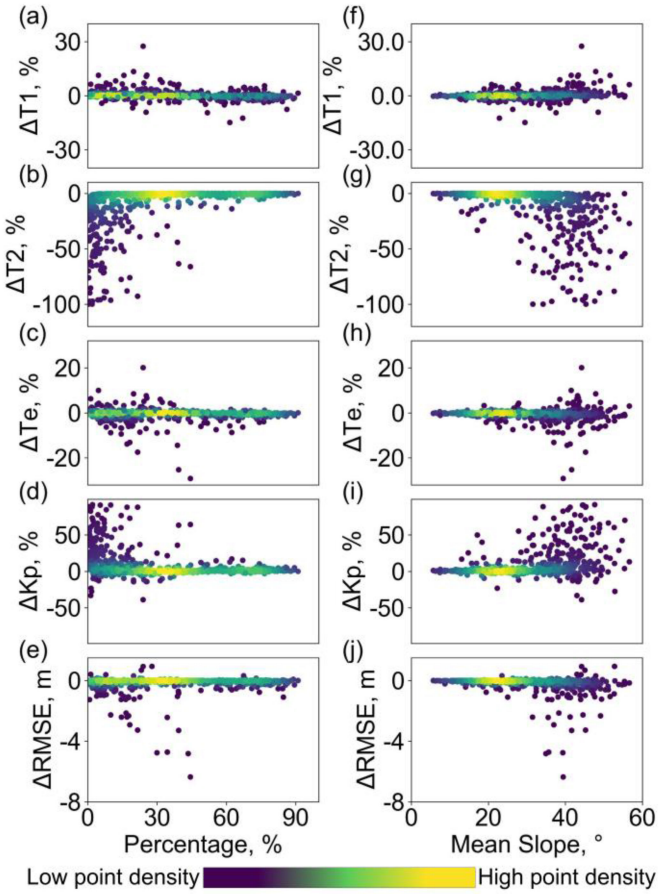


Fig. 13. Improvement relationships, in terms of T1, T2, Te, Kp, and mean RMSE of DTM, with percentage of vegetation point and mean slope of each sample between using multitask training and only point classification task training. ΔT1 is derived using T1 of multitask PFCN minus T1 of only point classification task, and the calculations of ΔT2, ΔTe, ΔKp, and ΔRMSE are similar.

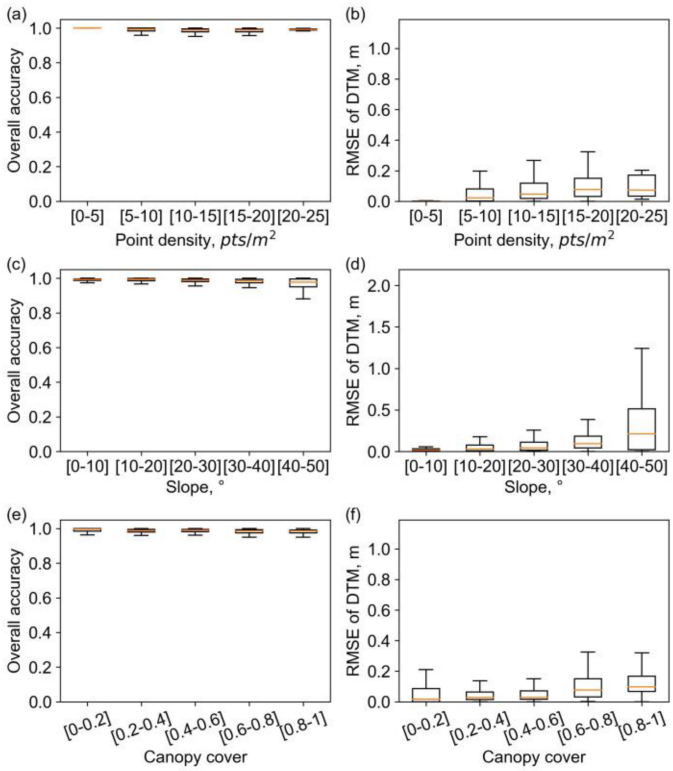


Fig. 14. Overall accuracy and RMSE of DTM (at 0.5 m resolution) of PFCN under different point density, terrain type, and vegetation type using testing samples with an area of 20 × 20 m.

learning technologies, such as FCN [67], residual network (ResNet) [66], and multitask learning [63]. The major contributions to ALS filtering include as follows.

1) A combination of point-wise and tile-wise features was designed, which can lead to better performances than using only point-wise or only tile-wise features.
2) Multitask (point classification and tile classification) training with a joint loss (the sum of FL and cross-entropy loss) was adopted, which can achieve better filtering results than using only point classification task.
3) The PFCN method used FL to enhance the classification ability under different imbalanced data distribution scenarios [68].

In addition, the model was trained end-to-end to generate the filtered ground points directly, which is very flexible for use with only raw point information or points with more attributes/features (e.g., intensity and echo width information). The PFCN method can be extended to other areas of discrete data processing problems, such as multiple class classification, by easily setting the number of output classes.

Some limitations of this method still exist, which will be addressed in future studies. First, the T2 filtering error was relatively low, but it was the highest compared to the other tested methods. This error could be reduced further by designing new loss functions or a more advanced network to understand the semantic relationships among local and global points [97], [98]. If the T2 error was lower, then we would expect the RMSE of the DTM to also be lower, accordingly. Second, the ground truth was generated through manual checking of the automatic filtering result of TerraScan, which may potentially have reduced the T1 of TerraScan and improved the accuracy of its DEM. That may be a possible reason why the RMSE of the DTM in the method was not the best. However, this approach is currently the most widely used method for producing benchmark data for point filtering. In this article, we have carefully and laboriously checked the filtering result in the 3-D profile to make the dataset more accurate. The manually corrected ratio is given in Table II. Third, the points used in this study contained only geometric information, and we did not analyze whether the PFCN accuracy would improve if points were provided with additional features. In general, adding more features, such as spectral information and normal information about the points would increase the network's performance, as shown in [55]. Fourth, in this article, each site was split into smaller tiles before being sent into the network, but this can be avoided in the future with the hardware development, particularly GPU. If a large scene (e.g., 500 m × 500 m) can be consumed as a single sample, we may not need to split the scene into tiles for training. However, we may need to adjust our network to add layers for local information extraction with the aid of some local feature extraction methods [55], [99]. Besides, high efficient network structures might be also needed to further reduce computation resource and time [100]. Fifth, because massive training data is needed to train a deep learning network, we acknowledge the generalization of the method is worse than heuristic methods. The needed minimum data size is important for training a deep neural network, but it has not been analyzed because it is complicatedly determined by both the network structure and data distribution. We recommend that the method of using validation set to monitor model training to explore the minimum sample size. Besides, carrying out small sample learning and transfer learning is an important work to improve the transferability of deep learning methods [101]. Finally, we have found that the accuracy of our method is reduced when vegetation coverage is small and slope is large. A quantitative analysis of the influence of these factors may be needed in the future, which can give more guidance about setting scanning patterns and flight heights.

VI. Conclusion

In this article, we proposed a PFCN for ALS ground point filtering in forested environments. The method operated on the point cloud directly and was trained in a multitask learning way using 37449157 points from 14 sites with various vegetation and terrain characteristics, and evaluated on six equally diverse sites (16470083 points). Additionally, the method was compared with traditional filtering methods. The PFCN showed the best average performance in terms of T1 (1.10%), Te (1.73%), and Kp (93.88%), and ranked second in terms of the RMSE of the DTM, but had the highest T2 (4.5%). Additionally, the filtering method was on par with or even better than one of the state-of-the-art point-based deep learning methods (PointNet++). Overall, the proposed PFCN is an end-to-end method that consumes simple geometric information and directly outputs the filtered result. We believe that this multitask method can be widely applied in ALS ground point filtering tasks.

APPENDIX

TABLE IV
ACCURACY OF THE POINT CLASSIFICATION TASK WHEN USING ONLY POINT-WISE, ONLY TILE-WISE, AND POINT-WISE AND TILE-WISE FEATURES. THE BOLD FONTS INDICATE THE ACCURACY WAS HIGHER WHEN USING POINT-WISE AND TILE-WISE FEATURES THAN USING ONLY POINT-WISE FEATURE OR ONLY TILE-WISE FEATURE

Sites	Only point-wise feature			Only tile-wise feature			Point-wise and tile-wise features		
	Te, %	Kp, %	Mean RMSE of DTM, m	Te, %	Kp, %	Mean RMSE of DTM, m	Te, %	Kp, %	Mean RMSE of DTM, m
Site1	0.69	98.39	0.04	30.91	0.00	6.28	0.69	98.39	0.04
Site2	1.09	97.80	0.03	43.93	0.00	6.89	**1.05**	**97.88**	**0.03**
Site5	2.59	93.19	0.28	75.39	0.00	16.33	**2.54**	**93.31**	**0.20**
Site14	2.86	94.27	0.29	50.19	0.00	11.13	3.00	94.00	0.73
Site16	3.78	77.45	0.51	8.46	0.00	2.52	**3.41**	**78.52**	0.78
Site19	0.87	96.90	0.21	16.93	0.00	3.33	**0.83**	**97.03**	**0.22**
Mean	1.98	93.00	0.23	37.64	0.00	7.75	**1.92**	**93.19**	0.33

TABLE V
TILE CLASSIFICATION ACCURACY USING ONLY TILE CLASSIFICATION LOSS AND MULTITASK LOSS. THE BOLD FONTS INDICATE THE ACCURACY WAS HIGHER WHEN USING MULTITASK LOSS THAN USING ONLY TILE CLASSIFICATION LOSS

Sites	Only tile classification loss	Multi-task loss
	Classification accuracy, %	Classification accuracy, %
Site1	99.04	**99.20**
Site2	100	**100**
Site5	100	**100**
Site14	99.36	99.04
Site16	81.92	**84.32**
Site19	92.96	**93.28**
Mean	95.55	**95.97**

TABLE VI
POINT CLASSIFICATION ACCURACY USING ONLY POINT CLASSIFICATION LOSS AND MULTITASK LOSS. THE BOLD FONTS INDICATE THE ACCURACY WAS HIGHER WHEN USING MULTITASK LOSS THAN USING ONLY POINT CLASSIFICATION LOSS

Sites	Only point classification loss			Multi-task loss		
	Te, %	Kp, %	Mean RMSE of DTM, m	Te, %	Kp, %	Mean RMSE of DTM, m
Site1	0.69	98.39	0.04	**0.66**	**98.45**	0.08
Site2	1.05	97.88	0.03	**1.01**	**97.95**	**0.03**
Site5	2.54	93.31	0.20	**2.49**	**93.48**	**0.19**
Site14	3.00	94.00	0.73	**2.63**	**94.74**	**0.31**
Site16	3.41	78.52	0.78	**2.60**	**83.07**	**0.66**
Site19	0.83	97.03	0.22	**0.79**	**97.15**	**0.21**
Mean	1.92	93.19	0.33	**1.70**	**94.14**	**0.25**

REFERENCES

[1] X. Meng, N. Currit, and K. Zhao, "Ground filtering algorithms for airborne LiDAR data: A review of critical issues," *Remote Sens.*, vol. 2, no. 3, pp. 833–860, 2010.

[2] W. T. Tinkham *et al.*, "A comparison of two open source LiDAR surface classification algorithms," *Remote Sens.*, vol. 3, no. 3, pp. 638–649, 2011.

[3] A. Khosravipour *et al.*, "Generating pit-free canopy height models from airborne LiDAR," *Photogrammetric Eng. Remote Sens.*, vol. 80, no. 9, pp. 863–872, 2014.

[4] J. Shan and C. K. Toth, *Topographic Laser Ranging and Scanning: Principles and Processing*. Boca Raton, FL, USA: CRC press, 2018, pp. 4–8.

[5] P. Gong, Y. Sheng, and G. Biging, "3D model-based tree measurement from high-resolution aerial imagery," *Photogrammetric Eng. Remote Sens.*, vol. 68, no. 11, pp. 1203–1212, 2002.

[6] D. Hoekman, and C. Varekamp, "High resolution single-pass interferometric radar observation of tropical forest trees," in *Proc. 4th Int. Workshop Radar Polarimetry*, 1998, pp. 517–525.

[7] K. A. Razak *et al.*, "Airborne laser scanning of forested landslides characterization: Terrain model quality and visualization," *Geomorphology*, vol. 126, no. 1–2, pp. 186–200, 2011.

[8] X. Liu, "Airborne LiDAR for DEM generation: some critical issues," *Prog. Phys. Geog.*, vol. 32, no. 1, pp. 31–49, 2008.

[9] J. Brasington, D. Vericat, and I. Rychkov, "Modeling river bed morphology, roughness, and surface sedimentology using high resolution terrestrial laser scanning," *Water Resour. Res.*, vol. 48, no. 11, pp. 11519–11536, 2012.

[10] J. Telling *et al.*, "Review of earth science research using terrestrial laser scanning," *Earth-Sci. Rev.*, vol. 169, pp. 35–68, 2017.

[11] R. O. Dubayah, and J. B. Drake, "LiDAR remote sensing for forestry," *J. Forest*, vol. 98, no. 6, pp. 44–46, 2000.

[12] J. C. Suárez *et al.*, "Use of airborne LiDAR and aerial photography in the estimation of individual tree heights in forestry," *Comput. Geosci.*, vol. 31, no. 2, pp. 253–262, 2005.

[13] S. Jin *et al.*, "The transferability of random forest in canopy height estimation from multi-source remote sensing data," *Remote Sens.*, vol. 10, no. 8, pp. 1183–1203, 2018.

[14] X. Guo *et al.*, "Regional mapping of vegetation structure for biodiversity monitoring using airborne LiDAR data," *Ecol. Inform.*, vol. 38, pp. 50–61, 2017.

[15] T. Hu *et al.*, "A simple and integrated approach for fire severity assessment using bi-temporal airborne LiDAR data," *Int. J. Appl. Earth Observ.*, vol. 78, pp. 25–38, 2019.

[16] Y. Su *et al.*, "SRTM Dem correction in vegetated mountain areas through the integration of spaceborne LiDAR, airborne LiDAR, and optical imagery," *Remote Sens.*, vol. 7, no. 9, pp. 11202–11225, 2015.

[17] S. Tao *et al.*, "Global patterns and determinants of forest canopy height," *Ecology*, vol. 97, no. 12, pp. 3265–3270, 2016.

[18] X. Hu and Y. Yuan, "Deep-Learning-Based classification for DTM Extraction from ALS point cloud," *Remote Sens.*, vol. 8, no. 9, pp. 730–745, 2016.

[19] X. Zhao *et al.*, "Improved progressive TIN densification filtering algorithm for airborne LiDAR data in forested areas," *ISPRS J. Photogrammetric*, vol. 117, pp. 7–91, 2016.

[20] G. Vosselman, "Slope based filtering of laser altimetry data," *Int. Arch. Photogrammetry Remote Sens.*, vol. 33, , 2000, Art. no. 935942.

[21] G. Sithole and G. Vosselman, "Filtering of laser altimetry data using a slope adaptive filter," *Int. Arch. Photogrammetry Remote Sens. Spatial Inf. Sci.*, vol. 34, pp. 203–210, 2001.

[22] J. Susaki, "Adaptive slope filtering of airborne LiDAR data in urban areas for digital terrain model (DTM) generation," *Remote Sens.*, vol. 4, no. 6, pp. 1804–1819, 2012.

[23] X. Meng *et al.*, "A multi-directional ground filtering algorithm for airborne LiDAR," *ISPRS J. Photogrammetry*, vol. 64, no. 1, pp. 117–124, 2009.

[24] G. Sithole and G. Vosselman, "Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds," *ISPRS J. Photogrammetry*, vol. 59, no. 1–2, pp. 85–101, 2004.

[25] Q. Chen *et al.*, "Filtering airborne laser scanning data with morphological methods," *Photogrammetric Eng. Remote Sens.*, vol. 73, no. 2, pp. 175–185, 2007.

[26] K. Zhang *et al.*, "A progressive morphological filter for removing nonground measurements from airborne LiDAR data," *IEEE Trans. Geosci. Remote*, vol. 41, no. 4, pp. 872–882, Apr. 2003.

[27] T. J. Pingel, K. C. Clarke, and W. A. McBride, "An improved simple morphological filter for the terrain classification of airborne LiDAR data," *ISPRS J. Photogrammetry*, vol. 77, pp. 21–30, 2013.

[28] Y. Li *et al.*, "An Improved top-hat filter with sloped brim for extracting ground points from airborne LiDAR point clouds," *Remote Sens.*, vol. 6, no. 12, pp. 12885–12908, 2014.

[29] Z. Hui *et al.*, "An improved morphological algorithm for filtering airborne LiDAR point cloud based on multi-level Kriging interpolation," *Remote Sens.*, vol. 8, no. 1, pp. 35–50, 2016.

[30] H. Arefi, and M. Hahn, "A morphological reconstruction algorithm for separating off-terrain points from terrain points in laser scanning data," *Int. Arch. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 36, no. 3/W19, pp. 120–125, 2005.

[31] K. Kraus and N. Pfeifer, "Determination of terrain models in wooded areas with airborne laser scanner data," *ISPRS J. Photogrammetry*, vol. 53, no. 4, pp. 193–203, 1998.

[32] C. Chen *et al.*, "A fast and robust interpolation filter for airborne LiDAR point clouds," *PloS One*, vol. 12, no. 5, 2017, Art no. e0176954.

[33] D. Mongus and B. Žalik, "Parameter-free ground filtering of LiDAR data for automatic DTM generation," *ISPRS J. Photogrammetry*, vol. 67, pp. 1–12, 2012.

[34] X. Shi *et al.*, "A parameter-free progressive TIN densification filtering algorithm for LiDAR point clouds," *Int. J. Remote Sens.*, vol. 39, no. 20, pp. 6969–6982, 2018.

[35] X. Lin and J. Zhang, "Segmentation-based filtering of airborne LiDAR point clouds by progressive densification of terrain segments," *Remote Sens.*, vol. 6, no. 2, pp. 1294–1326, 2014.

[36] G. Sithole and G. Vosselman, "Filtering of airborne laser scanner data based on segmented point clouds," *Int. Arch. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 36, pp. 66–71, 2005.

[37] B. Yunfei *et al.*, "Classification of LiDAR point cloud and generation of DTM from LiDAR height and intensity data in forested area," *Int. Arch. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 37, no. 7, pp. 313–318, 2008.

[38] Q. Guo *et al.*, "Effects of topographic variability and LiDAR sampling density on several dem interpolation methods," *Photogrammetric Eng. Remote Sens.*, vol. 76, no. 6, pp. 701–712, 2010.

[39] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[40] L. Ma *et al.*, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS J. Photogrammetry*, vol. 152, pp. 166–177, 2019.

[41] J. E. Ball, D. T. Anderson, and C. S. Chan, "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community," *J. Appl. Remote Sens.*, vol. 11, no. 4, 2017, Art. no. 042609.

[42] X. X. Zhu *et al.*, "Deep Learning in Remote Sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

[43] S. Jin *et al.*, "Deep Learning: Individual maize segmentation from terrestrial LiDAR data using Faster R-CNN and regional growth algorithms," *Front Plant Sci.*, vol. 9, pp. 866–875, 2018.

[44] H. Su *et al.*, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.

[45] D. Maturana and S. Scherer, "Voxnet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 922–928.

[46] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. 4th Int. Conf. 3D Vis.*, 2016, pp. 565–571.

[47] P.-S. Wang *et al.*, "O-CNN: Octree-based convolutional neural networks for 3D Shape analysis," *ACM T Grap.*, vol. 36, no. 4, pp. 1–11, 2017.

[48] G. Riegler, A. O. Ulusoys, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 2017, pp. 3577–3586.

[49] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 863–872.

[50] C. R. Qi *et al.*, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Adv. Neural Inf. Process. Systs.*, 2017, pp. 5099–5108.

[51] C. R. Qi *et al.*, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 2017, pp. 652–660.

[52] L. Landrieu, and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 2018, pp. 4558–4567.

[53] Y. Wang *et al.*, "Dynamic graph CNN for learning on point clouds," *ACM T Grap.*, vol. 38, no. 5, pp. 1–12, 2019.

[54] S. Schmohl and U. Sörgel, "Submanifold sparse convolutional networks for semantic segmentation of large-scale ALS point clouds," *ISPRS Ann. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 4, pp. 1–8, 2019.

[55] M. Yousefhussien *et al.*, "A multi-scale fully convolutional network for semantic labeling of 3D point clouds," *ISPRS J. Photogrammetry*, vol. 143, pp. 191–204, 2018.

[56] D. Griffiths and J. Boehm, "A review on deep learning techniques for 3D sensed data classification," *Remote Sens.*, vol. 11, no. 12, pp. 1499–1526, 2019.

[57] A. Boulch, "Generalizing discrete convolutions for unstructured point clouds," in *3DOR*, 2019, pp. 71–78.

[58] L. Winiwarter *et al.*, "Classification of ALS point clouds using end-to-end deep learning," *PFG–J. Photogrammetry, Remote Sens. Geoinformation Sci.*, vol. 87, no. 3, pp. 75–90, 2019.

[59] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3693–3702.

[60] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[61] N. Chehata, L. Guo, and C. Mallet, "Airborne LiDAR feature selection for urban classification using random forests," *Int. Arch. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 38, pp. 207–212, 2009.

[62] Z. Yang *et al.*, "A convolutional neural network-based 3D semantic labeling method for ALS point clouds," *Remote Sens.*, vol. 9, no. 9, pp. 936–952, 2017.

[63] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017, *arXiv:1706.05098*.

[64] X. Wang *et al.*, "Associatively segmenting instances and semantics in point clouds," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 2019, pp. 4096–4105.

[65] K. Xu *et al.*, "Residual blocks PointNet: A novel faster PointNet framework for segmentation and estimated pose," in *Proc. 5th IEEE Int. Conf. Cloud Comput. Intell. Syst.*, 2019, pp. 446–450.

[66] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[67] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[68] T.-Y. Lin *et al.*, "Focal loss for dense object detection," 2017, *arXiv:1708.02002*.

[69] M. Jaderberg, K. Simonyan, and A. Zisserman, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

[70] D. W. Ruck *et al.*, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans. Neural Netw.*, vol. 1, no. 4, pp. 296–298, Dec. 1990.

[71] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[72] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: A theoretical and experimental comparison, " in *Proc. 14th Annu. Conf. Int. Speech Commun. Assoc.*, 2013, pp. 1756–1760.

[73] M. L. Clark, D. B. Clark, and D. A. Roberts, "Small-footprint LiDAR estimation of sub-canopy elevation and tree height in a tropical rain forest landscape," *Remote Sens. Environ.*, vol. 91, no. 1, pp. 68–89, 2004.

[74] R. M. Lucas *et al.*, "Empirical relationships between AIRSAR backscatter and LiDAR-derived forest biomass," *Remote Sens. Environ.*, vol. 100, no. 3, pp. 407–425, 2006.
[75] W. Li *et al.*, "A new method for segmenting individual trees from the LiDAR point cloud," *Photogrammetic Eng. Remote Sens.*, vol. 78, no. 1, pp. 75–84, 2012.
[76] M. E. Hodgson, and P. Bresnahan, "Accuracy of airborne LiDAR-derived elevation," *Photogrammetric Eng. Remote Sens,*, vol. 70, no. 3, pp. 331–339, 2004.
[77] C. Waldhauser *et al.*, "Automated classification of airborne laser scanning point clouds," in *Solving Computationally Expensive Engineering Problems*, New York, NY, USA: Springer, 2014, pp. 269–292.
[78] X. Zhao *et al.*, "A Comparison of LiDAR filtering algorithms in vegetated mountain areas," *Can. J. Remote Sens.*, vol. 44, pp. 1–12, 2018.
[79] A. Paszke *et al.*, "PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration,"2017. [Online]. Avialable: https: //github. com/pytorch/pytorch
[80] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[81] G. Sithole and G. Vosselman, "Comparison of filtering algorithms," in *Proc. ISPRS Work. Group III/3 Workshop*, 2003, pp. 71–78.
[82] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psycholog.Meas.*, vol. 20, no. 1, pp. 37–46, 1960.
[83] J. S. Evans and A. T. Hudak, "A multiscale curvature algorithm for classifying discrete return LiDAR in forested environments," *IEEE Trans. Geosci. Remote*, vol. 45, no. 4, pp. 1029–1038, Apr. 2007.
[84] P. Axelsson, "DEM generation from laser scanner data using adaptive TIN models," *Int. Arch. Photogrammetry Remote Sens.*, vol. 33, no. 4, pp. 110–117, 2000.
[85] J. Zhang and X. Lin, "Filtering airborne LiDAR data by embedding smoothness-constrained segmentation in progressive TIN densification," *ISPRS J. Photogrammetry*, vol. 81, pp. 44–59, 2013.
[86] Y. Li *et al.*, "A gradient-constrained morphological filtering algorithm for airborne LiDAR," *Opt Laser Technol.*, vol. 54, pp. 288–296, 2013.
[87] A. L. Montealegre, M. T. Lamelas, and J. d. l. Riva, "A comparison of open-source LiDAR filtering algorithms in a mediterranean forest environment," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 8, pp. 4072–4085, Aug. 2015.
[88] R. J. McGaughey, "FUSION/LDV: Software for LiDAR data analysis and visualization," US Dept. Agr., Forest Service, Pac. Northwest Res. Station, Seattle, WA, USA, 2009.
[89] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in *Proc. 16th Int. Soc. Music Inf. Retrieval Conf.*, 2015, pp. 121–126.
[90] H. Keselman and J. C. Rogan, "The Tukey multiple comparison test: 1953–1976," *Psycholog. Bull.*, vol. 84, no. 5, p. 1–1050, 1977.
[91] K. Zhang and D. Whitman, "Comparison of three algorithms for filtering airborne LiDAR data," *Photogrammetric Eng. Remote Sens.*, vol. 71, no. 3, pp. 313–324, 2005.
[92] A. Rizaldy *et al.*, "Fully convolutional networks for ground classification from LiDAR point clouds," *ISPRS Ann. Photogrammetry, Remote Sens. Spatial Inf. Sci.*, vol. 4, no. 2, pp. 231–238, 2018.
[93] L. Cheng *et al.*, "Semi-Automatic registration of airborne and terrestrial laser scanning data using building corner matching with boundaries as reliability check," *Remote Sens.*, vol. 5, no. 12, pp. 6260–6283, 2013.
[94] B. Pradhan, *Laser Scanning Applications in Landslide Assessment*. Berlin, Germany: Springer, 2017, pp. 87–88.
[95] Q.-H. Pham *et al.*, "JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 2019, pp. 8827–8836.
[96] S. Jin *et al.*, "Separating the structural components of maize for field phenotyping using terrestrial lidar data and deep convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 2, pp. 2644–2658, Apr. 2020.
[97] F. Liu *et al.*, "3DCNN-DQN-RNN: A deep reinforcement learning framework for semantic parsing of large-scale 3D point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5678–5687.
[98] Z. Liang, M. Yang, and C. Wang, "3D Graph embedding learning with a structure-aware loss function for point cloud semantic instance segmentation," 2019, *arXiv:1902.05247*.
[99] C.-H. Lin *et al.*, "Eigen-feature analysis of weighted covariance matrices for LiDAR point cloud classification," *ISPRS J. Photogrammetry Remote Sens.*, vol. 94, pp. 70–79, 2014.
[100] Z. Liu *et al.*, "Point-Voxel CNN for Efficient 3D Deep Learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 963–973.
[101] Z. Cai, H. Ma, and L. Zhang, "Model transfer-based filtering for airborne LiDAR data with emphasis on active learning optimization," *Remote Sens. Letters*, vol. 9, no. 2, pp. 111–120, 2018.

Shichao Jin received the B.S. degree in forestry from Huazhong Agricultural University, Wuhan, China, in 2016, and the Ph.D. degree in ecology from Institute of Botany, Chinese Academy of Sciences, Beijing, China, in 2020.

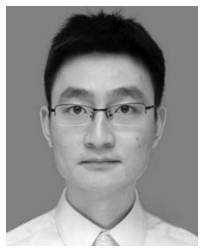
He is currently an Associate Professor with Plant Phenomics Research Center, Nanjing Agricultural University, Nanjing, China. His research focuses on using deep learning and LiDAR technology to solve phenotyping related challenges.

Yanjun Su received the B.E. degree in surveying and mapping engineering from China University of Geosciences, Beijing, China, in 2009, the M.S. degree in geographic information science from the Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing, China, in 2012, and the Ph.D. degree in environmental systems from the University of California at Merced, Merced, CA, USA, in 2017.

He is currently an Associate Professor with the Institute of Botany, Chinese Academy of Sciences, Beijing, China. His research interests include applying geographic information science and remote sensing to understand the influence of anthropogenic activities and global climate change on terrestrial ecosystems, with a particular emphasis on the terrestrial carbon cycle, terrestrial biodiversity, energy balance and land-use/land-cover change.

Xiaoqian Zhao received the B.S. degree in geographic information System from Lanzhou Jiaotong University, Lanzhou, China, in 2009, the M.S. degree in cartography and geographical information system from Wuhan University, Wuhan, China, in 2011, and the Ph.D. degree in ecology from Institute of Botany, Chinese Academy of Sciences, Beijing, China, in 2017.

She is currently an Engineer with the Institute of Botany, Chinese Academy of Sciences, Beijing, China. Her research interests include biodiversity informatics and biodiversity studies based on remote sensing.



Tianyu Hu received the B.S. degree in ecology from China Agriculture University, Beijing, China, in 2008, and the Ph.D. degree with the Institute of Botany, Chinese Academy of Sciences, Beijing, China, in 2014.

He is currently an Assistant Professor with the Institute of Botany, Chinese Academy of Sciences, Beijing, China. His research interests include using LiDAR technology and dynamic vegetation model to understand forest ecosystem, especially in forest structure, function and biodiversity.

Qinghua Guo received the B.S. degree in environmental science and the M.S. degree in remote sensing and geographic information system (GIS) from Peking University, Beijing, China, in 1996 and 1999, respectively, and the Ph.D. degree in environmental science from the University of California, Berkeley, CA, USA, in 2005.

He is currently a Professor with the Institute of Botany, Chinese Academy of Sciences, Beijing, China. He is also an Adjunct Professor and a member of the founding faculty with the School of Engineering, University of California at Merced, Merced, CA, USA. His research interests include GIS and remote sensing algorithm development and their environmental applications, such as object-based image analysis, geographic one-class data analysis, and LiDAR data processing.