remote sensing

MDPI

Article

Deep-Learning-Based Classification for DTM Extraction from ALS Point Cloud

Xiangyun Hu [1,2] and Yi Yuan [1,*]

[1] School of Remote Sensing and Information Engineering, 129 Luoyu Road, Wuhan University, Wuhan 430079, China; huxy@whu.edu.cn

[2] Collaborative Innovation Center of Geospatial Technology, Wuhan University, Wuhan 430079, China

* Correspondence: yuan_yi@whu.edu.cn; Tel.: +86-138-8605-6791

Academic Editors: Jie Shan, Juha Hyyppä, Lars T. Waser, Xiaofeng Li and Prasad S. Thenkabail
Received: 20 May 2016; Accepted: 29 August 2016; Published: 5 September 2016

Abstract: Airborne laser scanning (ALS) point cloud data are suitable for digital terrain model (DTM) extraction given its high accuracy in elevation. Existing filtering algorithms that eliminate non-ground points mostly depend on terrain feature assumptions or representations; these assumptions result in errors when the scene is complex. This paper proposes a new method for ground point extraction based on deep learning using deep convolutional neural networks (CNN). For every point with spatial context, the neighboring points within a window are extracted and transformed into an image. Then, the classification of a point can be treated as the classification of an image; the point-to-image transformation is carefully crafted by considering the height information in the neighborhood area. After being trained on approximately 17 million labeled ALS points, the deep CNN model can learn how a human operator recognizes a point as a ground point or not. The model performs better than typical existing algorithms in terms of error rate, indicating the significant potential of deep-learning-based methods in feature extraction from a point cloud.

Keywords: deep learning; convolutional neural network (CNN); digital terrain model (DTM); ALS; ground point classification

1. Introduction

In recent decades, airborne laser scanning (ALS) has become more important in the process of digital terrain model (DTM) production [1]. ALS can provide a description of a surface on a terrain with high accuracy and density. However, ALS also records the information of non-terrain objects, such as buildings and trees. Thus, the ALS filtration is important in the processing of ALS data. Given the various non-ground objects on the surface and the lack of topology among the points, filtering of the ALS point cloud can be difficult and troublesome. In fact, point filtering often occupies approximately 80% of the workload of ALS data processing in DTM production. The algorithms of ALS filtration can be divided into three categories based on their characteristics, as follows:

(1) Slope-based methods. The kernel foundation of these methods considers that two adjacent points are likely to belong to different categories if they have a mutation in height [2,3]. Slope-based methods are fast and easy to implement. Their shortcoming is their dependency on different thresholds in different terrains.

(2) Mathematical morphology-based methods. These methods are composed of a series of 3D morphological operations on the ALS points. The results of morphological methods heavily rely on the filter window size. Small windows can only filter small non-ground objects, such as telegraph poles or small cars. By contrast, large windows often filter several ground points and make the results of filtration smooth. Zhang [4] proposed progressive morphological filters,

which can filter large non-ground objects with ground points preserved by varying the filter window size, to overcome this problem.

(3) Progressive triangular irregular network (TIN)-based method. Axelsson [5] proposed the iterative TIN; this network has been used in some business software. The TIN selects the coarse lowest points as ground points and builds a triangulated surface from them. Then, the TIN adds new points to the triangular surface under many constrains for slope and distance. However, the method is easily affected by negative outliers; these outliers draw the triangular surface downward.

(4) Surface-based methods. These methods maintain a surface model of the ground based on the interpolation of ground points [6–9]. However, these methods are sensitive to input parameters and negative outliers.

Other recent algorithms try to use optimization to obtain accurate classification. For instance, semi-global filtering (SGF) [10] employs a novel energy function balanced by adaptive ground saliency to adapt to steep slopes, discontinuous terrains, and complex objects. Then, the SGF uses semi-global optimization to determine labels by minimizing the energy.

Although the existing methods have done well in ALS filtration, they still need much human labor to generate DTM based on the filtration results. We want to make full use of the existing ALS and responding DTM by learning a deep neural network from a big amount of the existing data. Neural networks has been used in pattern recognition and classification for a long time [11,12]. The deep convolutional neural networks (CNN) [13] are inspired by biological vision systems; these networks have recently shown their ability to extract high-level representations through compositions of low-level features [14]. In the present study, we propose a new filtering algorithm based on deep CNN. First, training samples are obtained from many labeled points. Each image is generated from the point and its neighboring points; the image can be a positive or negative training sample depending on the label. Second, a deep CNN model is trained using the labeled data. Images generated from points are treated as input of the deep CNN model. Then, the input will be processed by several components being comprised of a convolution layer, a batch Normalization layer, an activation layer and a pooling layer after some components. At last, the results of the last pooling layer will be connected to subsequent three fully-connected layers, the last fully-connected layer will produce the probability for the input to be a ground point or a non-ground point. Detailed construction of deep CNN model can be seen in Section 2.2. The deep CNN model can learn the important feature of the input automatically from the huge training data, which usually work better than hand-craft features. Finally, each point is mapped to an image to classify a raw ALS point cloud; this image is classified as an image belonging to a ground point or is not used by the trained CNN model.

This paper is organized as follows: Section 2 describes the proposed method. Section 3 presents the ALS filtration results and analysis. Section 3.3 compares the proposed method with other methods. Section 4 concludes this study and identifies several aspects for improvement.

2. Methods

The workflow of our approach for filtering is shown in Figure 1. ALS filtering means to find and delete all non-ground points from ALS data. We treat filtering as a binary classification problem to classify all the points of ALS data as ground points or non-ground points. The major steps include the calculation of context information for each point from the neighboring points in a window, the transformation of the information of the window into an image, and the training and classification based on the images using the CNN model. Training sample points are selected from a large number of point clouds with different terrain complexities. A deep CNN model is trained from the labeled images.

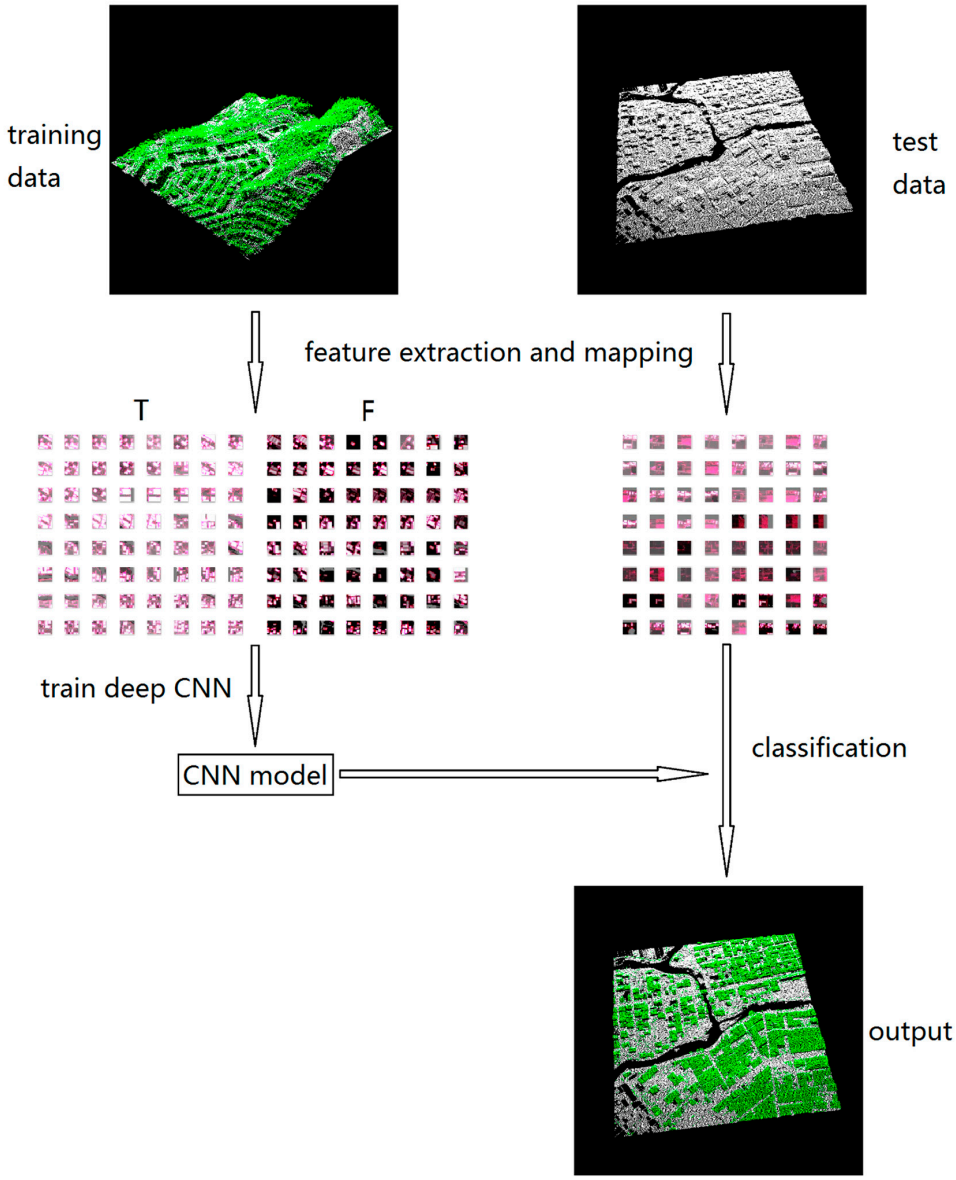


Figure 1. Workflow of the proposed approach. "T" means the samples of ground points and "F" means the samples of non-ground points.

2.1. Information Extraction and Image Generation

For each ALS point (P_i), its surrounding points within its "square window" are divided into many cells. The "square window" means a square in (x, y) spatial coordinates. It is a two dimensional window. In the method, the size of "square window" is 96 m × 96 m, which is divided into 128 × 128 cells. Each "square window" extracted for a point can be transferred to a 128 × 128 image by mapping each cell to a pixel with red, blue, and green colors. For each cell, the maximum (Z_{max}), minimum (Z_{min}), and mean (Z_{mean}) of the height among all points within the cell are obtained. Then, the difference values between Z_{max}, Z_{min}, and Z_{mean} and the height (Z_i) of point (P_i) are transferred to three integers within 0 to 255 following Equations (1) and (2); these integers would be the red, green, and blue values of the corresponding pixel in the image transformed from the cells, as follows:

$$\begin{aligned} F_{red} &= \lfloor 255 * Sigmoid(Z_{max} - Z_i) - 0.5 \rfloor \\ F_{green} &= \lfloor 255 * Sigmoid(Z_{min} - Z_i) - 0.5 \rfloor \\ F_{blue} &= \lfloor 255 * Sigmoid(Z_{mean} - Z_i) - 0.5 \rfloor \end{aligned} \quad (1)$$

The sigmoid function is expressed in Equation (2), as follows:

$$Sigmoid(x) = (1 + e^{-x})^{-1} \quad (2)$$

An example for the point-to-image transformation is shown in Figure 2.

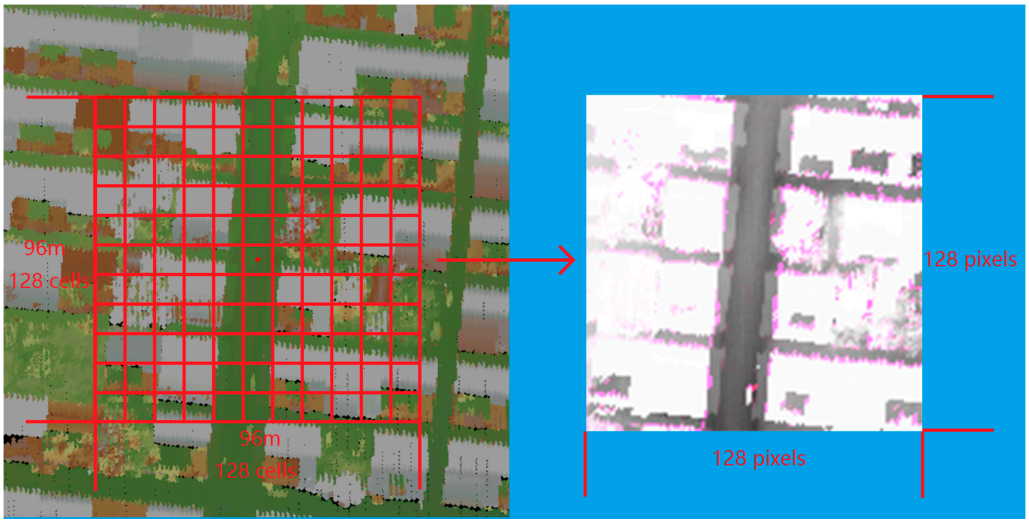


Figure 2. Point-to-image transformation (source: own study in the "FugroViewer").

2.2. Convolutional Neural Network

CNNs have been the focus of considerable attention for a wide range of vision-related [15–18], audio-related [19], or language-related [20] tasks. The existing best-performing models [21–23] on ImageNet ILSVRC have all been based on deep CNNs since 2012. CNNs are designed to process data that come in the form of multiple arrays, such as 1D arrays for signals like language and 2D arrays for images or audio spectrograms. CNNs have four key ideas, namely, local connections, shared weights, pooling, and use of many layers [24]. More detailed description of CNN can be found in [25].

The architecture of the CNN model used in our approach is shown in Figure 3.

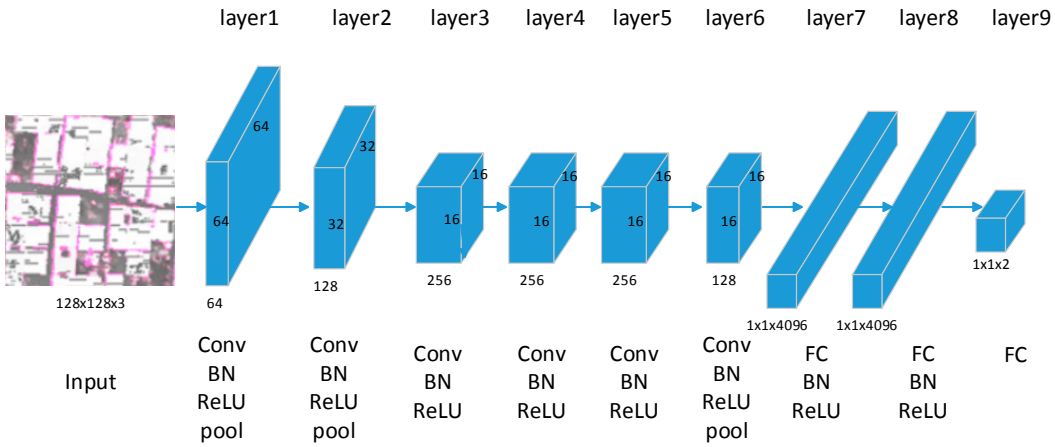


Figure 3. The architecture of the proposed deep CNN.

Deep CNN model is comprised of 6 kinds of layers, the size of layers can be defined as $width \times height \times depth$ in which $width \times height$ describes the spatial size and $depth$ refers to the number of channels of its feature maps. The detailed explanation of the layers in Figure 3 can be seen below:

(1) An input layer is denoted as $Input$ here. The input layer contains the input data for the network. The input of the deep CNN model is a three-channel (red, green, blue) 128 × 128 image generated from an ALS points.

(2) A convolution layer is denoted as *Conv* here. The convolution layer is the core building block of a convolutional network that performs most of the computational heavy lifting. Convolutional layers convolve the input image or feature maps with a learnable linear filter, which have a small receptive field (local connections) but extend through the full depth of the input volume. The output feature maps represent the responses of each filter on the input image or feature maps. Each filter is replicated across the entire visual field and the replicated unit share the same weights and bias (shared weights), which allows for features to be detected regardless of their position in the visual field. As a result, the network learns filters that activate when they see a specific type of feature at some spatial position in the input. In our model, all of the convolution layers use the same sized 3 × 3 convolution kernel.

(3) A batch normalization layer is denoted as BN here. Given that the deep CNN often has a large number of parameters, taking care to prevent overfitting is necessary, particularly when the number of training samples is relatively small. Batch normalization [26] normalizes the data in each mini-batch, rather than merely performing normalization once at the beginning, using the following equation:

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}\gamma + \beta \tag{3}$$

The input of BN is normalized to zero mean and unit variance and then linearly transformed. During training, μ and σ^2 are the mean and variance of the input mini-batch. During testing, μ and σ^2 are the average statistics calculated from the training data. γ and β are learned parameters which scale and shift the normalized value. ε is a constant added to the mini-batch variance for numerical stability.

Batch normalization can significantly reduce overfitting, allow higher learning rates and accelerate the training for deep network.

(1) A rectified linear units layer is denoted as $ReLU$ here. Activation layers are neuron layers that apply nonlinear activations on input neurons. They increase the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Rectified linear units (ReLU) proposed by Nair and Hinton in 2010 [27] is the most popular activation function. ReLU can be trained faster than typical smoother nonlinear functions and allows the training of a deep supervised network without unsupervised pretraining. The function of ReLU can be demonstrated as $f(x) = max(0, x)$.

(2) A pooling layer is denoted as *Pooling* here. Pooling layers are nonlinear downsampling layers that achieve maximum or average values in each sub-region of input image or feature maps. The intuition is that once a feature has been found, its exact location is not as important as its rough location relative to other features. Pooling layers increase the robustness of translation and reduce the number of network parameters.

(3) A fully-connected layer is denoted as FC here. After several convolutional and max pooling layers, high-level reasoning in the neural network is performed via fully-connected layers. A fully-connected layer takes all neurons in the previous layer and connects it to every single neuron it has. Fully-connected layers are not spatially located anymore, thereby making them suitable for classification rather than location or semantic segmentation.

A BN and a $ReLU$ are applied after every *conv* layer and the first two FC layers. Thus, layers 1 to 6 are composed of $Conv \rightarrow BN \rightarrow ReLU$ and layers 7 and 8 are composed of $FC \rightarrow BN \rightarrow ReLU$. Pooling layers are applied after layers 1, 2, and 6. The output of the last FC layer is fed to a 2-way softmax, which produces a distribution over the 2 class labels. Our network maximizes the multinomial logistic regression objective.

To train the model of the deep CNN, over 150 million parameters need to be learned. Two measures are taken to avoid overfitting: huge amount of training data and batch normalization layers which are proven to be effective.

3. Experimental Analysis

3.1. Experimental Data

A total of 17,280,000 labeled points are sampled evenly from 900 airborne ALS datasets in south China to be used to train a general model tested in variety types of terrains to evaluate the proposed approach. Each dataset has an area size of 500 m by 500 m and an average density of 4 points/m^2. Moreover, 40 scenes outside the training areas and the International Society for Photogrammetry and Remote Sensing (ISPRS) benchmark datasets provided by the ISPRS Commission III/WG2 [28] are classified to validate the trained CNN model. The 40 scenes have the same area size as the training data and approximately 40 million points. All training and testing ground truths are produced by a procedure of DTM production, including automatic filtering by TerraScan software and post manual editing. Examples of the training dataset and feature maps of the training samples are shown in Figures 4 and 5, respectively.

It is easy to see from Figure 5 that as the ground points usually being lower than their surrounding points while non-ground points more probable being higher than their surrounding points, most of the F_{red}, F_{green} and F_{blue} calculated from surrounding cells of ground points by Equation (1) are much bigger than non-ground points, which causes that the feature images of ground points are much brighter than non-ground points.

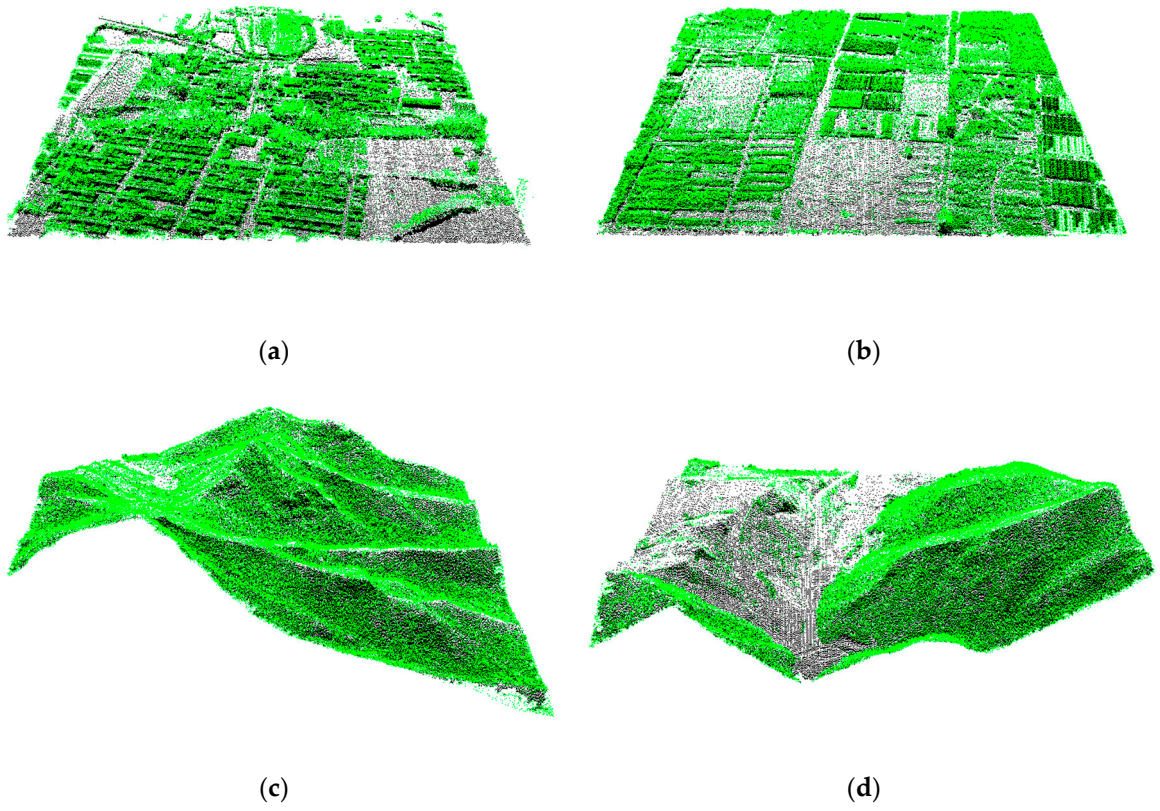


Figure 4. Four examples of the training ALS point clouds with different terrain features: (**a**,**b**) flat terrain with buildings and farmland; (**c**,**d**): mountainous terrain. White denotes the ground points, and green denotes the non-ground points.

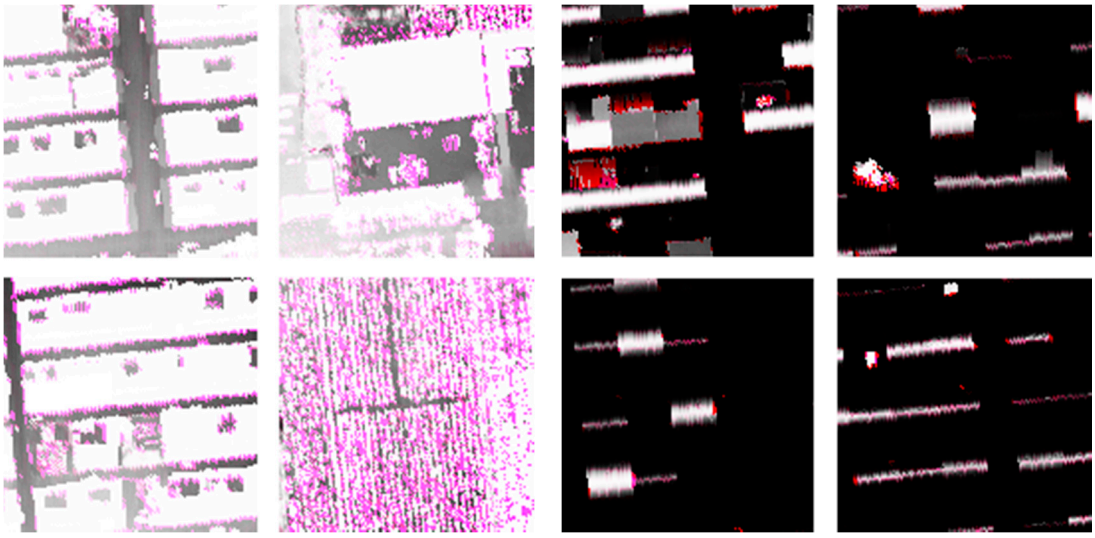


Figure 5. Training samples of the feature images corresponding to: (**a**) ground points; and (**b**) non-ground points.

3.2. Training

Batch gradient descent with a batch size of 256 examples, momentum of 0.9, and weight decay of 0.0005 to estimate the CNN parameters is used for the training. To find a local minimum of a function, gradient descent takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. In batch gradient descent, the gradient is approximately estimated by the mini-batch in each iteration.

The loss function of the CNN model can be calculated as:

$$L = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)} = j\} log \frac{e^{w_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{w_l^T x^{(i)}}}\right], \tag{4}$$

where m is the size of batch 256, and k is the number of classes (in here k = 2 because there are 2 classes, ground points and non-ground points), w is the parameters of the model, x is the output of the upper layer and for the first hidden layer, and x is the input layer. $y^{(i)}$ is the label of training sample i. The value of $1\left\{y^{(i)} = j\right\}$ equals 1 while $y^{(i)} = j$ and 0 otherwise.

The update rule for weight w was:

$$\begin{aligned} v_{t+1} &:= 0.9 \cdot v_t - 0.0005 \cdot \varepsilon \cdot W_t - \varepsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{W_t} \right\rangle_{D_t} \\ W_{t+1} &:= W_t + v_{t+1} \end{aligned} \tag{5}$$

where t is the iteration index, mini-batch D_t is the m training samples which will be used to estimate the gradient in this iteration, v is the momentum variable, ε is the learning rate, and $\left\langle \frac{\partial L}{\partial w}\Big|_{W_t} \right\rangle_{D_t}$ is the average over the t th batch D_t of the derivative of the objective loss function with respect to W, evaluated at W_t [21].

The training of the CNN model takes approximately three weeks on a PC with Intel i7-4790 CPU, 32 GB RAM, and a NIVIDIA GTX TitanX GPU.

3.3. Results and Comparison with Other Filtering Algorithms

We compare the deep CNN model with the popular commercial software TerraSolid TerraScan, Mongus's parameter-free ground filtering algorithm in 2012 [1], SGF [10], Axelsson's algorithm, and Mongus's connected operators-based algorithm in 2014 [29] on the ISPRS benchmark dataset. TerraScan uses the TIN-based filtering method; this software produces a significantly low average total error when a set of tunable parameters of the data is processed using the algorithm. The classification that CNN used for this test over two datasets is the one trained by the 900 airborne ALS datasets in south China to challenge the versatility of the CNN.

The filtering accuracy is measured based on the Type I error, which is the percentage of rejected bare ground points; Type II error, which is the percentage of accepted non-ground points; and total error, which is the overall probability of points being incorrectly classified. The results are shown in Table 1 and Figures 6–8. The classification using deep CNN model takes approximately 200 s on a test ALS dataset with a million points using a computer with an i7-4790 CPU and a TitanX GPU.

We also compare deep CNN model with TerraScan on 40 cases from the test ALS data with various terrain complexities in the aspects of both error rates and root mean square error of DTM. The comparison of error rates is shown in Figure 7 and comparison of root mean square error (RMSE) between the generated DTM with the ground truths is shown in Figure 8.

Table 1. Comparison of deep CNN model and other methods on the ISPRS dataset.

	Type I Error (%)	Type II Error (%)	Total Error (%)
TerraScan	11.05	4.52	7.61
Mongus 2012	3.49	9.39	5.62
SGF	5.25	4.46	4.85
Axelsson	5.55	7.46	4.82
Mongus 2014	2.68	12.79	4.41
Deep CNN	0.67	2.262	1.22

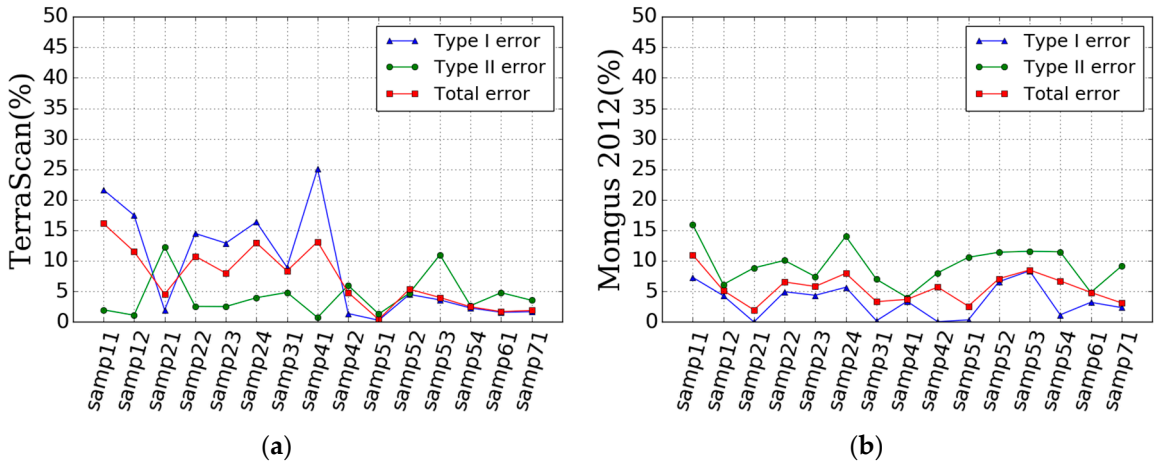


Figure 6. *Cont.*

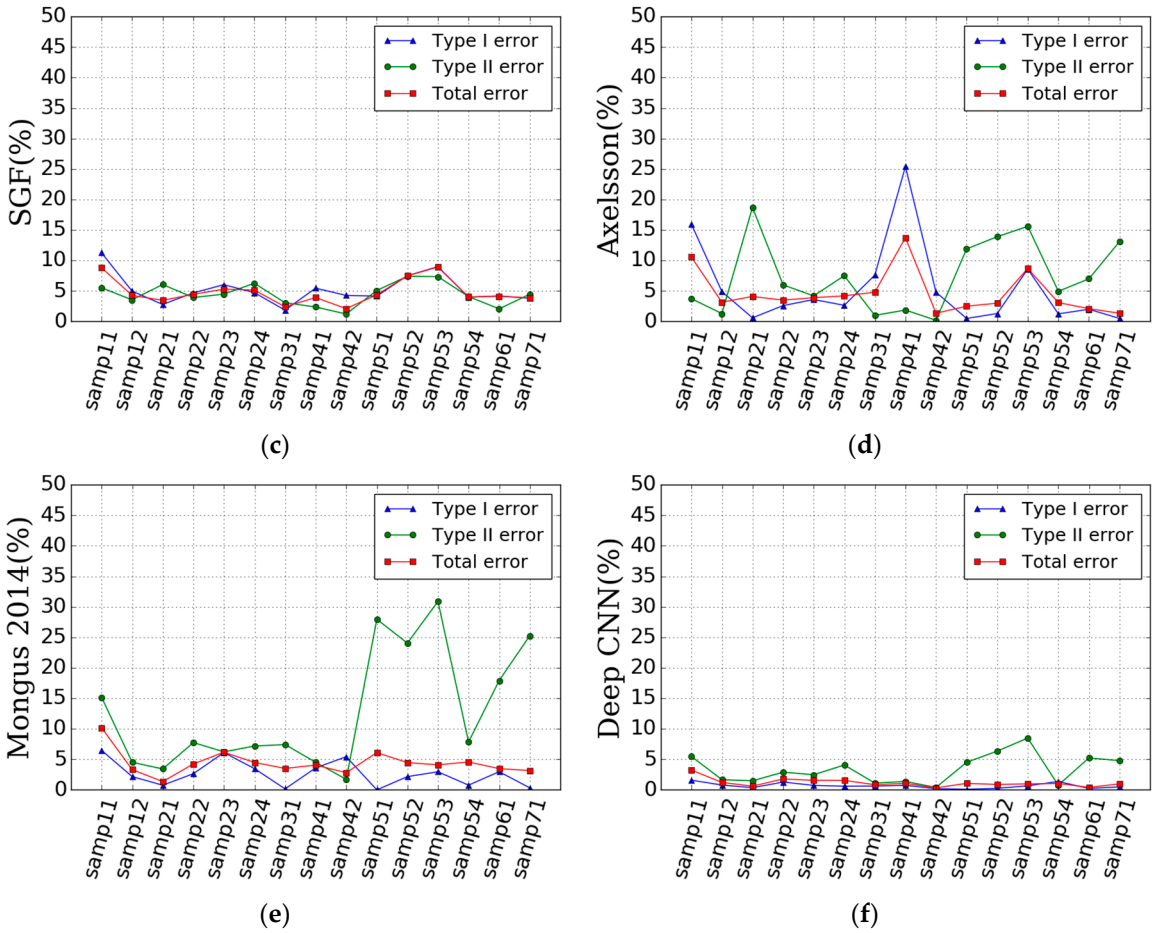


Figure 6. Detailed comparison with other methods and the proposed algorithm across 15 samples in the ISPRS dataset: (**a**) error rates of Terrasan, (**b**) error rates of Mongus 2012, (**c**) error rates of SGF, (**d**) error rates of Axelsson, (**e**) error rates of Mongus 2014, (**f**) error rates of Deep CNN.

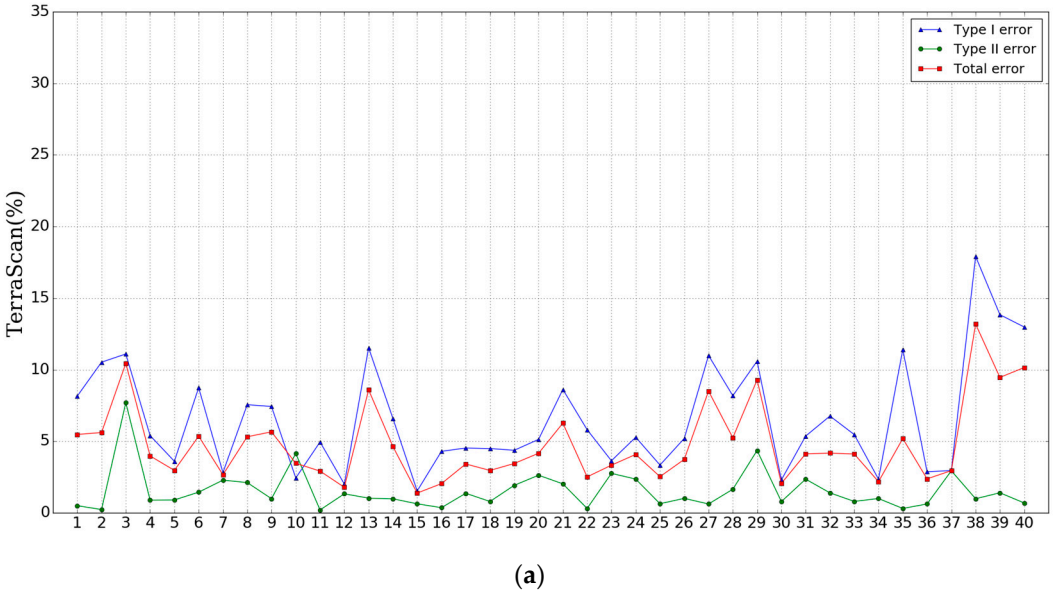
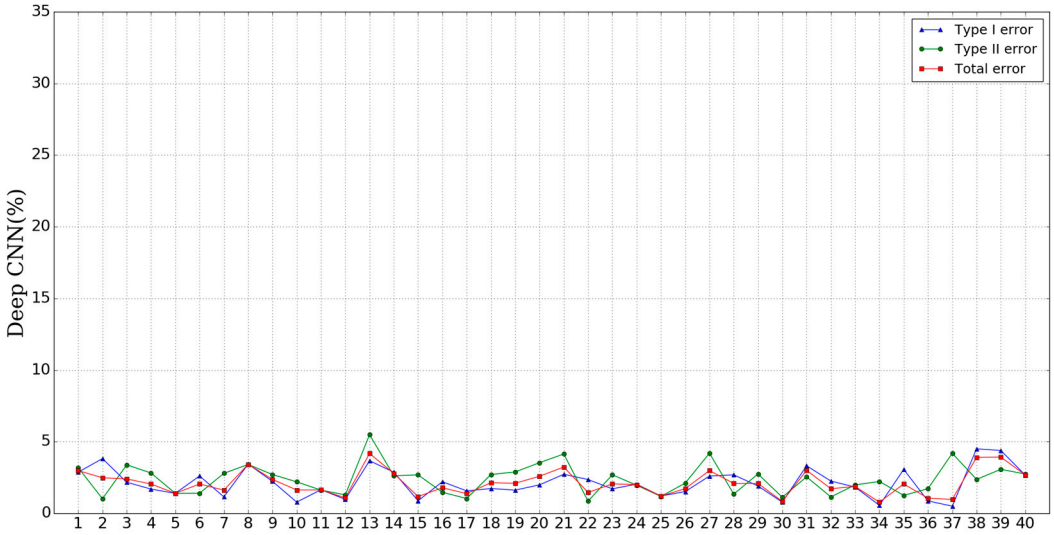


Figure 7. *Cont.*



(**b**)

Figure 7. Error rate of TerraScan (**a**) and Deep CNN (**b**) in 40 test cases.

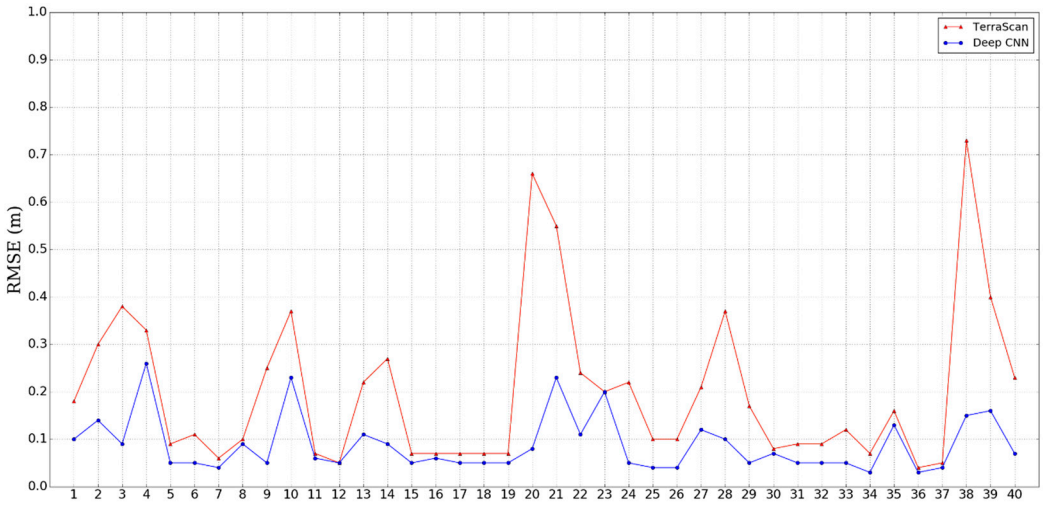


Figure 8. Comparison of root mean square error (RMSE) between the generated DTM with the ground truths.

The comparison of total error over 40 various complex terrains can be seen in Table 2 below and the detailed comparison of several examples with different terrains are shown in Figures 9–12.

Table 2. Comparison of total error over 40 various complex terrains between TerraScan and deep CNN model.

Error	TerraScan	Deep CNN
type I	10.5%	3.6%
type II	1.4%	2.2%
total	6.3%	2.9%

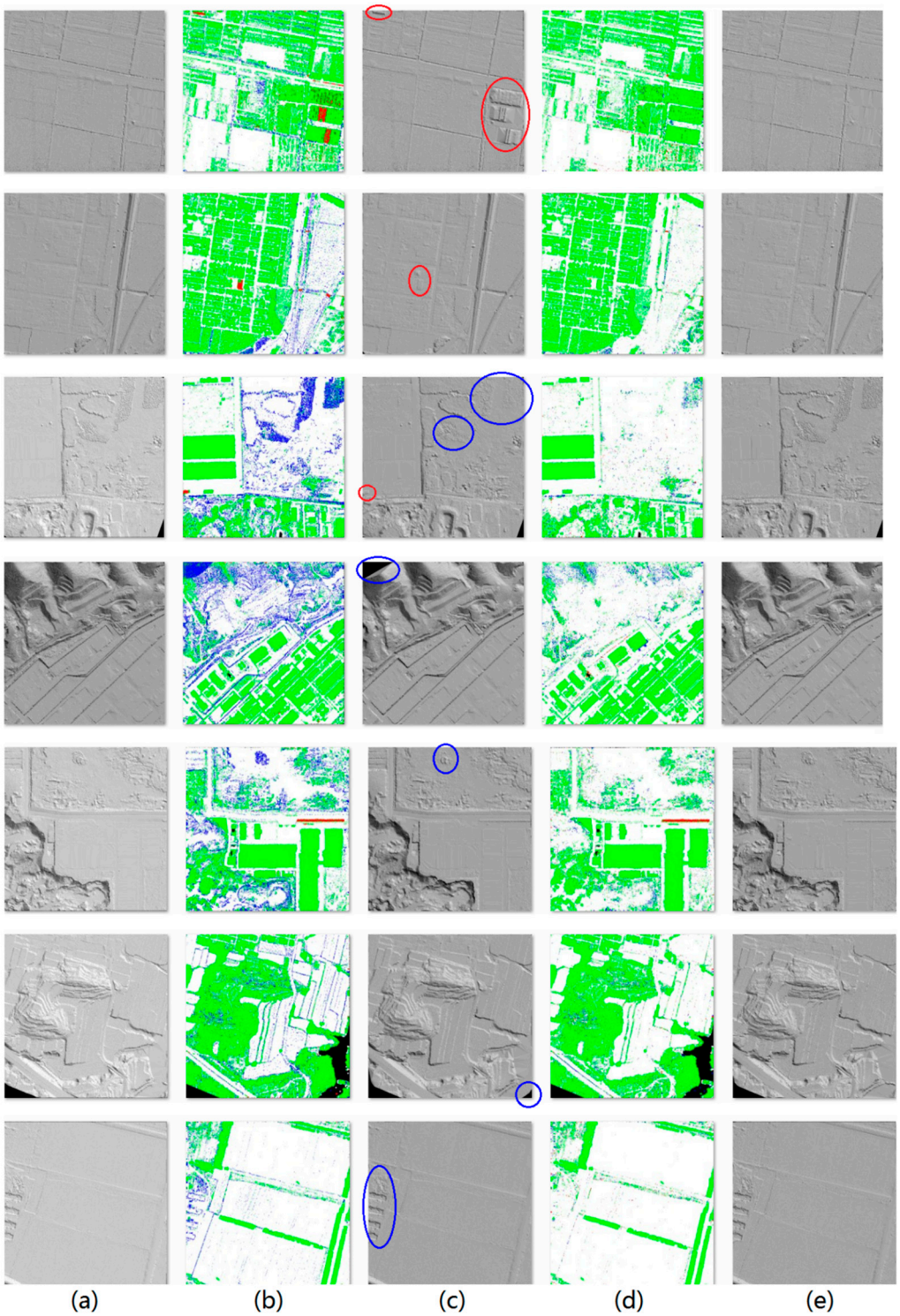


Figure 9. Comparison of the proposed method and TerraScan on the detailed difference of the DTM. Column (**a**) is the ground truth TIN-rendered gray image of the test data. Columns (b,d) are the results of filtration by TerraScan and Deep CNN, respectively; the white points denote correctly classified ground points, the green points denote correctly classified non-ground points, the red points denote accepted non-ground points, and the blue points denote rejected ground points. Columns (**c**) and (**e**) are TIN-rendered DTM extracted from the results of filtration by TerraScan and deep CNN model, respectively. In Column (**c**), blue ellipses denote type I error and red ellipses denote type II error.

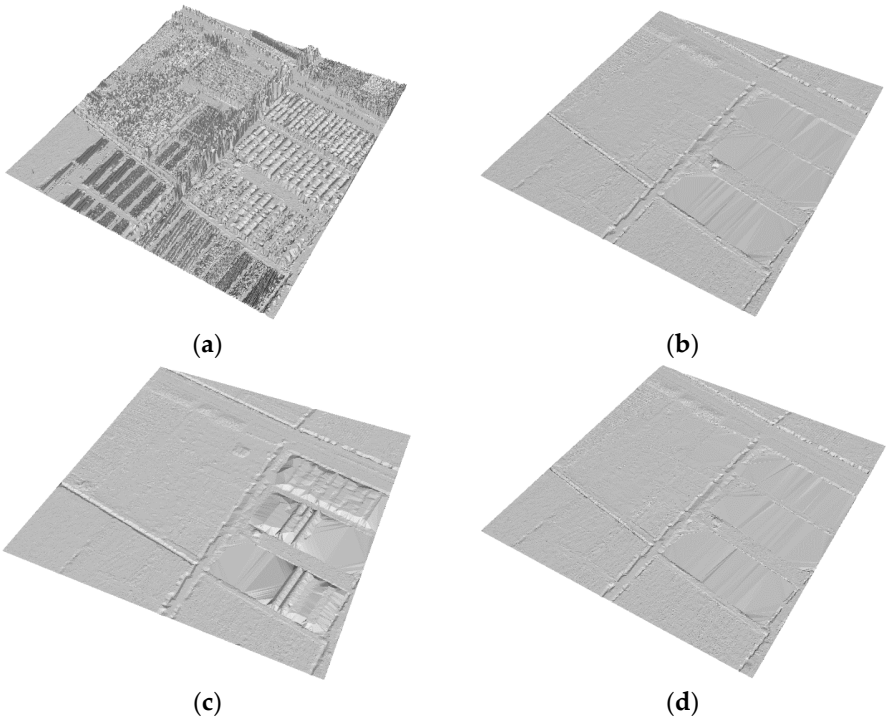


Figure 10. Comparison of the proposed method and TerraScan on the details of the plain area: (**a**) raw ALS data; (**b**) ground truth; (**c**) result of TerraScan; and (**d**) result of deep CNN model.

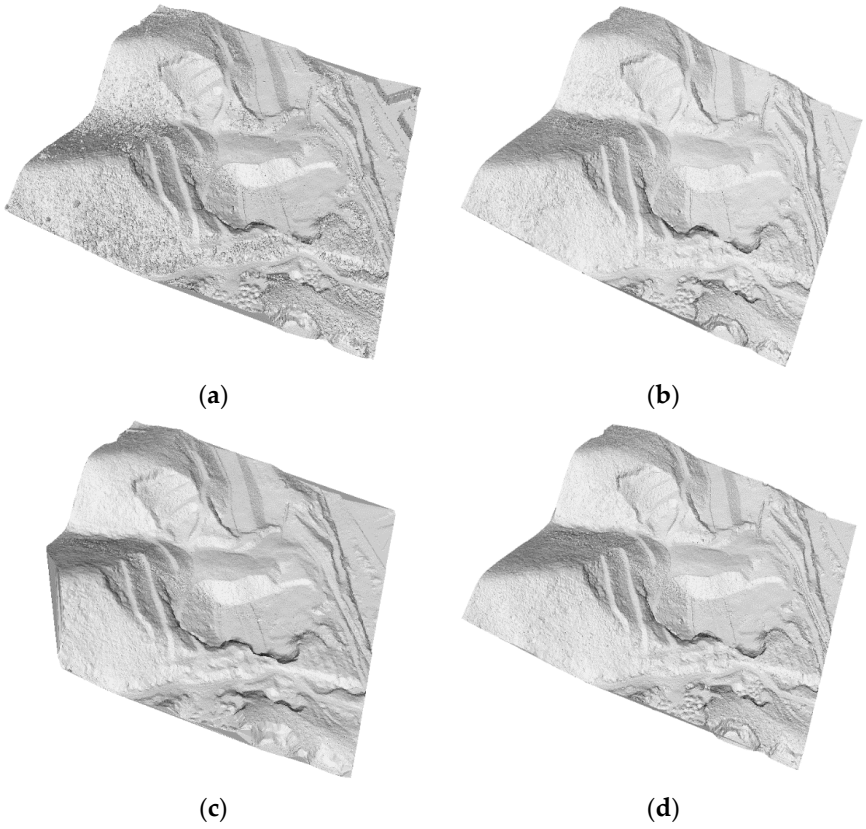


Figure 11. Comparison of the proposed method and TerraScan on the details of the mountain area: (**a**) raw ALS data; (**b**) ground truth; (**c**) result of TerraScan; and (**d**) result of deep CNN model.

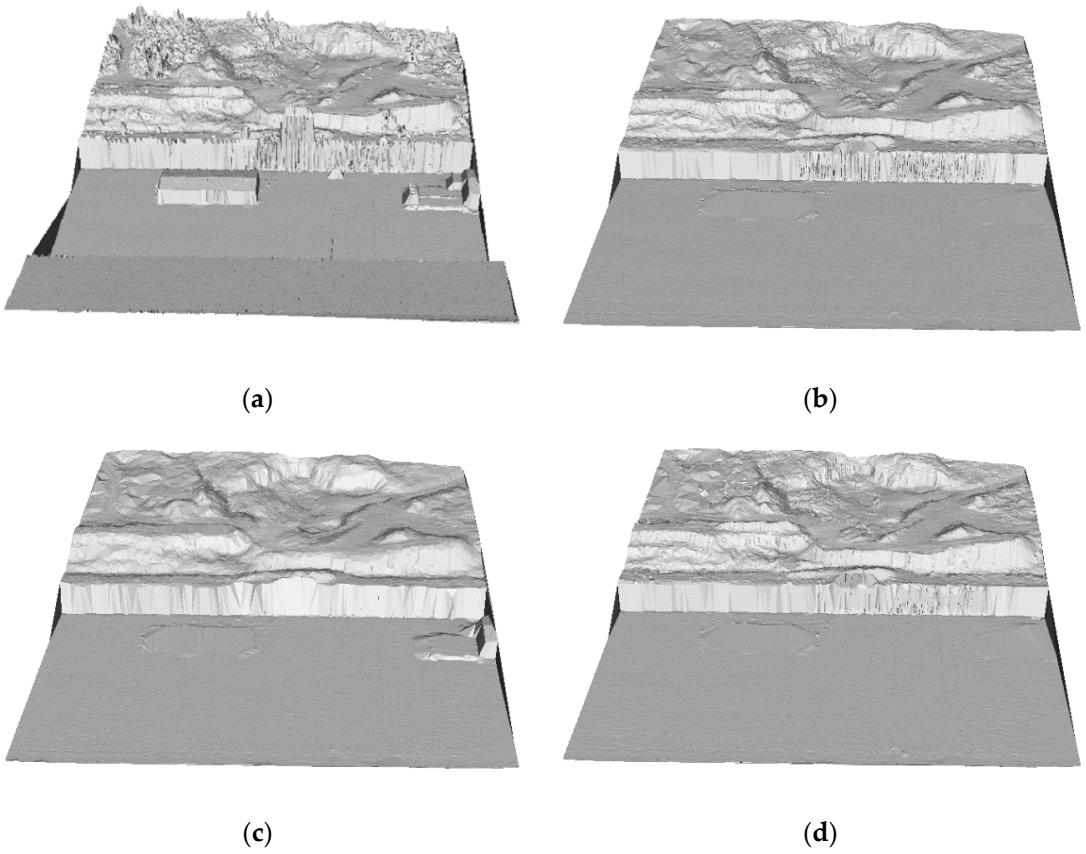


Figure 12. Comparison of the proposed method and TerraScan on the details of the complex area: (**a**) raw ALS data; (**b**) ground truth; (**c**) result of TerraScan; and (**d**) result of deep CNN model.

The tests strongly indicate that the CNN model produces significantly low Type I error; this result indicates only a slightly tedious manual post-editing for DTM production because removing the non-ground points (Type II error) is usually easier than finding the incorrectly rejected ground points. The proposed CNN-based classification can generate high-quality DTM, particularly to retain subtle, micro, and steep terrains; existing handcrafted algorithms may produce more Type I error.

Figures 10 and 12 show that deep CNN model does well in some big scale non-ground situations which are hard for TerraScan such as buildings and farmlands. Figure 11 shows that deep CNN model conserve the terrain feature well even in the mountains which often has little ground points caused by the shield of trees. The ground hit density in mountain area of Figure 11 is about 1–2 ground points per square meter while the ground hit density in plain area in the same data is about 4–6 ground points per square meter.

However, there are still some cases that deep CNN model cannot deal with very well, as shown in Figure 13.

The CNN model generates some wrong results (type II error) as shown in Figure 13. This is mainly because these wrong non-ground points belong to very low (down to 10 cm) man-made structures, which are too close to the ground. Modifying points-image transformation to represent shape and structure information may improve the accuracy in the similar cases.

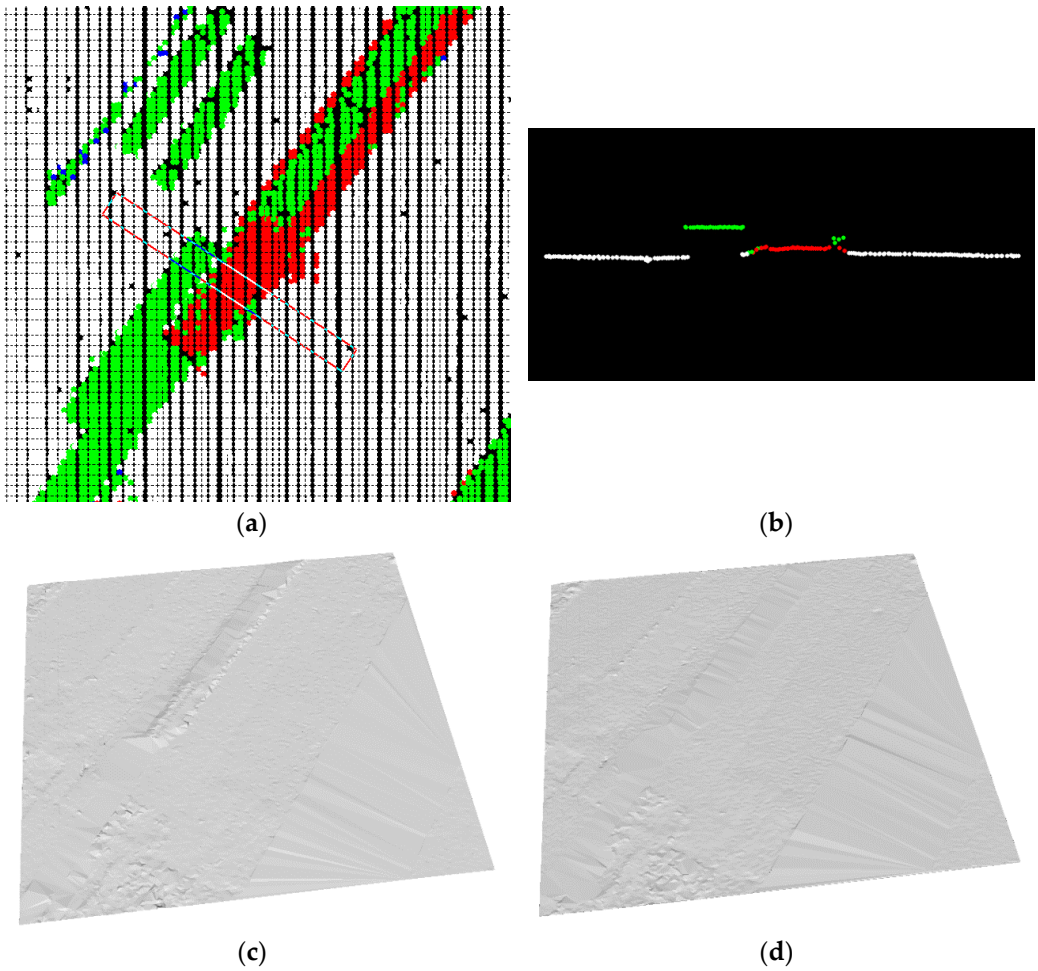


Figure 13. (**a**) Area that deep CNN model accept many wrong ground points, where the white points denote correctly classified ground points, the green points denote correctly classified non-ground points, the red points denote accepted non-ground points, and the blue points denote rejected ground points; (**b**) the profile of that area; (**c**) DTM of that area by deep CNN model; and (**d**) ground truth DTM of that area. The root mean square error (RMSE) between the DTM by deep CNN model with the ground truth DTM in this section shown below is 0.1 m.

4. Conclusions

To the best of our knowledge, this study is the first one that reports using deep CNN-based classification for DTM extraction from ALS data. Relative elevation differences between each point and its surrounding points are extracted and transformed into an image representing the point feature. Then, the deep CNN model is used to train and classify the images. Each point to be processed can be classified as a ground or non-ground point by the trained deep CNN. A total of 40 ALS point clouds with 40 million points and the ISPRS benchmark dataset with various scene complexities and terrain types are tested using one CNN trained by 17 million labeled points. The results show the high accuracy of the proposed method. The developed method provides a general framework for ALS point cloud classification.

However, the drawback of deep-learning-based methods is that they usually require large labeled data and powerful computational resources. Future work should focus on better point-image transformation and making more compact classification through the deep CNN model to improve the training and classification. Future work should also perform tests on larger datasets.

Acknowledgments: This study was partially supported by the research funding from Guangdong Province (2013B090400008) and Guangzhou City (201508020054) of China. The authors thank Guangzhou Jiantong Surveying, Mapping, and Geographic Information Technology Ltd. for providing the data used in this research project.

Author Contributions: Xiangyun Hu originally proposed using deep CNN for the classification-based filtering and transforming the point context to an image; he also guided the algorithm design and revised the paper. Yi Yuan conducted the detailed algorithm design, implementation, and experimental study. He also wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mongus, D.; Žalik, B. Parameter-free ground filtering of LiDAR data for automatic DTM generation. *ISPRS J. Photogramm. Remote Sens.* **2012**, *67*, 1–12. [CrossRef]
2. Vosselman, G. Slope based filtering of laser altimetry data. *Int. Arch. Photogram. Remote Sens.* **2000**, *33*, 935–942.
3. Brzank, A.; Heipke, C. Classification of Lidar Data into water and land points in coastal areas. *Int. Arch. Photogram. Remote Sens. Spat. Inform. Sci.* **2006**, *36*, 197–202.
4. Zhang, K.; Chen, S.-C.; Whitman, D.; Shyu, M.-L.; Yan, J.; Zhang, C. A progressive morphological filter for removing non-ground measurements from airborne LiDAR data. *IEEE Trans. Geosci. Remote Sens.* **2003**, *41*, 872–882. [CrossRef]
5. Axelsson, P. DEM generation from laser scanner data using adaptive TIN models. *Proc. Int. Arch. Photogramm. Remote Sens.* **2000**, *33*, 110–117.
6. Kraus, K.; Pfeifer, N. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS J. Photogramm. Remote Sens.* **1998**, *53*, 193–203. [CrossRef]
7. Kamiński, W. M-Estimation in the ALS cloud point filtration used for DTM creation. In *GIS FOR GEOSCIENTISTS*; Hrvatski Informatički Zbor-GIS Forum: Zagreb, Croatia; University of Silesia: Katowice, Poland, 2012; p. 50.
8. Błaszczak-Bąk, W.; Janowski, A.; Kamiński, W.; Rapiński, J. Application of the Msplit method for filtering airborne laser scanning datasets to estimate digital terrain models. *Int. J. Remote Sens.* **2015**, *36*, 2421–2437. [CrossRef]
9. Błaszczak-Bąk, W.; Janowski, A.; Kamiński, W.; Rapiński, J. ALS Data Filtration with Fuzzy Logic. *J. Indian Soc. Remote Sens.* **2011**, *39*, 591–597.
10. Hu, X.; Ye, L.; Pang, S.; Shan, J. Semi-Global Filtering of Airborne LiDAR Data for Fast Extraction of Digital Terrain Models. *Remote Sens.* **2015**, *7*, 10996–11015. [CrossRef]
11. Kubik, T.; Paluszynski, W.; Netzel, P. *Classification of Raster Images Using Neural Networks and Statistical Classification Methods*; University of Wroclaw: Wroclaw, Poland, 2008.
12. Meng, L. Application of neural network in cartographic pattern recognition. In Proceedings of the 16th International Cartographic Conference, Cologne, Germnay, 3–9 May 1993; Volume 1, pp. 192–202.
13. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
14. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 24–27 June 2014; pp. 580–587.
15. Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; Bregler, C. Efficient object localization using convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 648–656.
16. Taigman, Y.; Yang, M.; Ranzato, M.; Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014.
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *arXiv Preprint*, 2015. arXiv:1512.03385.

18. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; Lecun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
19. Waibel, A.; Hanazawa, T.; Hinton, G.E.; Shikano, K.; Lang, K. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust. Speech Signal Process.* **1989**, *37*, 328–339. [CrossRef]
20. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
21. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems 2012, Lake Tahoe, NV, USA, 3–8 December 2012.
22. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014.
23. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations, Banff, Canada, 16 April 2014.
24. LeCun, Y.; Yoshua, B.; Geoffrey, H. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
25. Ian, G.; Yoshua, B.; Aaron, C. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
26. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015.
27. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010.
28. WG III/2: Point Cloud Processing. Available online: http://www.commission3.isprs.org/wg2/ (accessed on 1 September 2016).
29. Mongus, D.; Zalik, B. Computationally efficient method for the generation of a digital terrain model from airborne LiDAR data using connected operators. *IEEE J. Sel. Top. Appl. Remote Sens.* **2014**, *7*, 340–351. [CrossRef]