ISPRS Journal of Photogrammetry and Remote Sensing 166 (2020) 128–139

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

DANCE-NET: Density-aware convolution networks with context encoding for airborne LiDAR point cloud classification

Xiang Li[a,b,c,d], Lingjing Wang[a,b,c,d], Mingyang Wang[a,b,c], Congcong Wen[b], Yi Fang[a,b,c,d,*]

[a] NYU Multimedia and Visual Computing Lab, NYU Tandon, United States
[b] NYU Multimedia and Visual Computing Lab, NYU Abu Dhabi, United Arab Emirates
[c] Tandon School of Engineering, New York University, New York, United States
[d] Department of Electrical and Computer Engineering, NYU Abu Dhabi, United Arab Emirates

ARTICLE INFO

Keywords:
Airborne LiDAR
Point cloud classification
Density-aware convolution
Context encoding

ABSTRACT

Airborne LiDAR point cloud classification has been a long-standing problem in photogrammetry and remote sensing. Early efforts either combine hand-crafted feature engineering with machine learning-based classification models or leverage the power of conventional convolutional neural networks (CNNs) on projected feature images. Recent proposed deep learning-based methods tend to develop new convolution operators which can be directly applied on raw point clouds for representative point feature learning. Although these methods have achieved satisfying performance for the classification of airborne LiDAR point clouds, they cannot adequately recognize fine-grained local structures due to the uneven density distribution of 3D point clouds. In this paper, to address this challenging issue, we introduce a density-aware convolution module which uses the point-wise density to reweight the learnable weights of convolution kernels. The proposed convolution module can approximate continuous convolution on unevenly distributed 3D point sets. Based on this convolution module, we further develop a multi-scale CNN model with downsampling and upsampling blocks to perform per-point semantic labeling. In addition, to regularize the global semantic context, we implement a context encoding module to predict a global context encoding and formulated a context encoding regularizer to enforce the predicted context encoding to be aligned with the ground truth one. The overall network can be trained in an end-to-end fashion and directly produces the desired classification results in one network forward pass. Experiments on the ISPRS 3D Labeling Dataset and 2019 Data Fusion Contest Dataset demonstrate the effectiveness and superiority of the proposed method for airborne LiDAR point cloud classification.

1. Introduction

With the rapid development of 3D sensor technology, 3D point cloud data has become more and more accessible through innovations in Light Detection and Ranging (LiDAR), Synthetic Aperture Radar (SAR), dense stereo- or multiview-photogrammetry in both remote sensing and computer vision fields. Among all these techniques, Airborne light detection and ranging (LiDAR) provides reliable 3D spatial information of real-world scenes and plays an crucial role in many applications such as forest monitoring (Mongus and Žalik, 2013; Solberg et al., 2009), powerline detection (Ene et al., 2017; Kim and Sohn, 2011), 3D building reconstruction (Kada and McKinley, 2009; Yang et al., 2017a), etc. Despite the prosperity of 3D point cloud data, automatic classification of 3D point clouds remain a challenging problem due to the unordered and irregular nature of raw point clouds.

Point cloud classification is also known as point cloud semantic segmentation in the computer vision field. To approach this problem, early researches mostly focus on the hand-crafted design of geometric features that represents the local structure of each point, such as normal, curvature, roughness. Some machine learning-based classification models, Support Vector Machine (Colgan et al., 2012; García et al., 2011) and Random Forest (Niemeyer et al., 2013, 2012) are then used to conduct per-point labeling. Other researches attempt to enhance the performance by consolidating contextual information to enforce label smoothness (Munoz et al., 2009; Shapovalov et al., 2010; Niemeyer et al., 2014; Weinmann et al., 2015b; Niemeyer et al., 2012). Nevertheless, these methods still require hand-crafted feature engineering and the generalization abilities of these models have limited performance when dealing with large-scale wild scenes.

Recent years, with the prevalence of deep learning methods in remote sensing field, remarkable performance has been achieved in various applications include scene classification, object detection, change

* Corresponding author at: NYU Multimedia and Visual Computing Lab, NYU Abu Dhabi, United Arab Emirates.
E-mail address: yfang@nyu.edu (Y. Fang).

https://doi.org/10.1016/j.isprsjprs.2020.05.023
Received 14 October 2019; Received in revised form 26 May 2020; Accepted 27 May 2020
Available online 12 June 2020

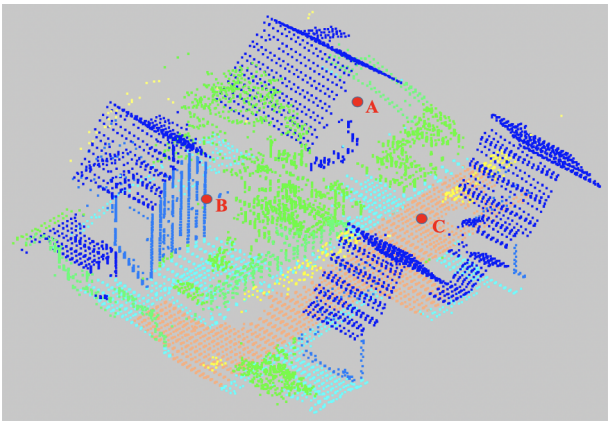

Fig. 1. Illustration of uneven distribution of 3D point clouds. Point 'A', 'B', and 'C' have different density values.

detection, hyperspectral image classification, etc (Hu et al., 2015a,b; Maggiori et al., 2016; Cheng et al., 2016; Zhan et al., 2017; Li et al., 2018b). Following the trend, researchers have been shifting their attention towards deep learning-based methods to approach the problem of semantic labeling of 3D point clouds (Qi et al., 2017a; Li et al., 2018c; Yang et al., 2017b; Yousefhussien et al., 2018; Wang et al., 2018). For example, to make use of the great power of convolutional neural networks (CNNs), some researchers (Yang et al., 2017b; Zhao et al., 2018) propose to project raw point clouds into 2D images and then leverage conventional CNNs for airborne point cloud classification. These methods usually require hand-crafted features to enhance the 2D feature image representations and classification performance is limited due to the information loss during 3D to 2D transformation. More recent works (Yousefhussien et al., 2018; Wang et al., 2018; Wen et al., 2020) also try the direct application of convolution operators on raw point clouds and learn high-level point features with deep neural networks. Although these methods have achieved state-of-the-art performance on several point cloud classification benchmarks, they do not adequately recognize fine-grained local structures due to the uneven density distribution of the point cloud data. Fig. 1 gives an example of unevenly distributed 3D point clouds.

In this paper, we propose a novel density-aware convolution module to extract more powerful geometric features from unstructured raw point clouds. Our key innovation is to force the convolution module to be aware of the local density distribution when learning its kernel weights. To achieve the goal of point cloud classification, we further develop a multi-scale CNN model with downsampling and upsampling blocks to enable multi-scale feature learning and per-point label prediction. Moreover, considering the imbalanced class distribution of an outdoor scene, a context encoding module is integrated into our network to regularize the global semantic context. The overall network can be trained in an end-to-end fashion and produce the classification results in one forward pass. Our main contributions are listed as follows:

1. This paper introduces a novel density-aware convolution module that directly applies convolution on irregular point clouds to learn representative point features. The inversed density after the nonlinear mapping is used to reweight the convolutional kernel weigh at each point.
2. Based on the proposed density-aware convolution module, we develop a multi-scale CNN model with downsampling and upsampling blocks to achieve per-point semantic labeling of 3D point clouds.
3. We introduce a context encoding module to regularize the global semantic context and experimentally demonstrated its effectiveness.
4. We achieve new state-of-the-art performance on the ISPRS 3D labeling benchmark dataset and 2019 Data Fusion Contest dataset.

The remainder of this paper is organized as follows. In Section 2, we summarize both traditional and recent proposed methods for airborne LiDAR point cloud classification. The proposed density-aware convolutional module, context encoding module, and the overall architecture are presented in Section 3. In Section 4, we carry out extensive experiments to show the performance of our method. We further discuss the effectiveness of the proposed density-aware convolution module and context encoding module in Section 5. Finally, the paper is concluded in Section 6.

2. Related work

Point cloud segmentation methods can be generally divided into two main categories: traditional machine-learning-based methods and deep learning-based methods.

2.1. Classical machine learning-based methods

Traditional machine learning-based methods start by the design of hard-crafted point features and then employ simple point-wise discriminative models to labeling the input point sets. These hand-crafted features are commonly generated from the local neighborhood of each point and characterize the local geometric features of that point, e.g. normal, curvature, planarity (Lin et al., 2014). After that, conventional machine learning-based classification algorithms, such as support vector machine (SVM) (Zhang et al., 2013; Mallet et al., 2011), random forests (RF) (Chehata et al., 2009; Kim and Sohn, 2011), and AdaBoost (Lodha et al., 2007), are leveraged to build supervised models and achieve point cloud classification. Nevertheless, these methods usually perform classification for each point independently by only taking its own local features as inputs and miss to model the spatial correlations among neighboring points. Therefore, these methods may suffer from classification noise and label inconsistency (Weinmann et al., 2015a).

To address this issue, recent researches have developed several contextual-based classification approaches to improve the smoothness of the classification results. For example, Niemeyer et al. develop a contextual classification model based on Conditional Random Field (CRF). Their experiment results demonstrate an improvement of 2% on the overall accuracy by integrating contextual features (Niemeyer et al., 2014). Niemeyer et al. further enhances contextual information by incorporating a two-layer Conditional Random Field (CRF). The first layer takes the original point clouds as inputs to generate point segments and the second layer processes the generated segments with a larger spatial scale (Niemeyer et al., 2016). In García-Gutiérrez et al. (2015), the authors propose a contextual classification method based on a Support Vector Machine (SVM) and an evolutionary majority voting technique. Their method demonstrates superior performance than the vanilla SVM model for land cover classification. Munoz et al. propose to incorporate the Associative Markov Network (AMN) to enable better high-order contextual interactions for the classification of 3D LiDAR point clouds classification (Munoz et al., 2009). In Shapovalov et al. (2010), the authors develop a Non-Associative Markov Network model by using dynamic pairwise potentials for a pair of different class labels, instead of the constant ones used in AMN model.

However, the above non-learning based methods need to manually extract point-wise local and context features in advance. This involves unwanted pre-processing time and makes these models sensitive to the quality of feature engineering. Moreover, the classification performance of these methods may degrade when processing point clouds in complex scenes (Zhao et al., 2018).

2.2. Deep learning-based methods

In contrast to the above machine learning-based methods, deep learning-based methods can automatically extract high-level features from a large amount of input data without feature engineering through

a hierarchical deep neural network. In general, these methods can be grouped into two main categories: feature image-based methods and point cloud-based methods.

2.2.1. Feature image-based methods

Convolutional neural networks (CNN) have achieved some extremely promising results in numerous vision-related tasks, such as scene classification, object detection, semantic segmentation, etc. However, conventional CNN models cannot be directly applied to the 3D point clouds which are unordered and irregular in nature. To address this issue, early researchers try to transform 3D point clouds into more tracTable 2D images followed by 2D CNN models to perform point clouds classification (Yang et al., 2017b, 2018; Zhao et al., 2018). For example, Su et al. (2015) proposes to first generate multiple 2D rendered images of 3D shapes and then a conventional 2D CNN is used to extract high-level features from each view. A view-pooling (aggregation) layer is further proposed to fuse information from multiple views and conduct point-wise classification. A similar method is proposed in Yang et al. (2017b) which generates projected 2D feature images that characterize both global and local geometric features of input point clouds.

For airborne LiDAR point cloud, Yang et al. develop a projective CNN-based method for the semantic labeling of airborne LiDAR point clouds. Their method starts by projecting 3D point clouds into multiple feature images that characterize the geometric features (e.g., intensity, planarity) of each point at different resolutions on the ground plane. Then, a 2D CNN is developed to learn representative features from multiple 2D projected feature images and then conduct classification of each projected point (Yang et al., 2018). Similarly, the authors in Zhao et al. (2018) propose to firstly generate a group of multi-scale contextual feature images, and then design a CNN model to generate classification results. Nevertheless, these methods involve the handcrafted process of feature images engineering from raw point clouds and their classification performance suffers from the information loss during projection transformation.

2.2.2. Point cloud-based methods

Pioneering work PointNet (Qi et al., 2017a) starts the trend of direct application of deep neural networks on the irregular point clouds. PointNet formulates the network architecture by a stack of multi-layer-perceptron (MLP) for hierarchical per-point feature learning and a global pooling function is adopted to generate the global feature vector for object classification. Their experiments on various 3D tasks demonstrate the powerful abilities of PointNet for point signature learning. PointNet++ (Qi et al., 2017b) further boosts the performance by introducing downsampling and upsampling modules to enable multi-scale feature learning. A hierarchical neural network is developed by applying a unit PointNet recursively on each grouped local region.

In light of the success of PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b), recent 3D point cloud classification and segmentation methods mostly build their methods upon PointNet-like architectures. For example, SO-Net (Li et al., 2018a) explores the Self-organizing map (SOM) to model the spatial distribution of point cloud and conducts hierarchical feature extraction on individual points and SOM nodes. PointSIFT (Jiang et al., 2018) develops a new convolution module which encodes neighbor information under different orientations and scales using a SIFT like operator. KC-Net (Shen et al., 2018) introduces a kernel correlation-based convolution module to capture the local geometric structures of raw point clouds. PointCNN (Li et al., 2018c) proposes a new general framework for point feature learning by transforming the input point sets into latent and potentially canonical order thus enables leveraging spatially-local correlation.

For airborne LiDAR point clouds, the authors in Yousefhussien et al. (2018) propose a fully convolutional network which takes as input both raw coordinates of input point clouds and three additional spectral features extracted from geo-referenced images of the same location for point-wise classification. Similarly, Wang et al. develop a multi-scale deep neural network to enable more powerful feature learning and further leads to a better point cloud classification performance. Their method starts by extracting per-point features using a shared MLP network. Then, a downsampling block is leveraged to aggregate per-point features into the cluster-based features. Finally, another MLP network followed by the Softmax classifier is used to conduct per-point classification. In Wen et al. (2020), the authors propose a direction-constrained convolution operator for point feature extraction and design a multi-scale fully convolutional network for point cloud classification. The authors in Arief et al. (2019) develop an Atrous XCRF module to enhance the original PointCNN model (Li et al., 2018c) and achieves promising performance for airborne LiDAR point cloud classification.

Although these point cloud-based methods have achieved outstanding performance for airborne LiDAR point cloud classification, they cannot model the density variations of input point clouds. To address this issue, recent researches propose different solutions to deal with varying densities in point clouds. For example, the authors in Hua et al. (2018) introduce a density normalization strategy which divides the convolutional kernel weights by the number of point in each kernel bin. Instead of normalizing density in the equation of the convolution, the authors in Thomas et al. (2019) develop a spatially deformable point convolution named Kernel Point Convolution (KPConv) which normalizes the density by point subsampling. The method most similar to this paper is Monto Carlo Convolution (MCCNN) proposed in Hermosilla et al. (2018). In Hermosilla et al. (2018), the authors propose an efficient and effective method to learn convolutions for non-uniformly sampled point clouds from a Monte Carlo perspective (Hermosilla et al., 2018). They use the local density of each point to reweight the convolutional kernel weights. Unlike Monte Carlo Convolution, in our proposed density-aware convolution, we feed the inversed density of each point to another MLP network for further enhancement. Moreover, a context encoding module regularizer is designed to regularize the global semantic context. We detail our method in the next section.

3. Method

In this section, we introduce the proposed Density-Aware Networks with Context Encoding (DANCE-NET) for airborne LiDAR point cloud classification. Our model introduces a generalized convolution operation that can directly process point clouds without any form of projections. In Section 3.1, we introduce the traditional formulation of point convolution. We introduce our proposed density-aware convolution operation for point feature learning in Section 3.2. In Section 3.3, we describe how we incorporate kernel density estimation to re-balance learned weight functions in point convolution. In Section 3.4, we propose a novel auxiliary task of predicting contexts using global features to further regularize our model. The overall network architecture is illustrated in Section 3.5.

3.1. Formulation of point convolution

A point cloud of size N can represented by a set $S = \{p_i, ..., p_n\}$, where each point $p_i = (x, y, z, f)$ contains the 3D coordinates as well as its feature such as color, surface normal, etc. In contrast to images pixels which are located in fixed grids, the points of a 3D point cloud are scattered in $\mathbb{R}^3$, which can take an arbitrary continuous value. Therefore, there does not exist a fixed kernel pattern that can be applied to each point for feature extraction, which makes conventional 2D convolution inapplicable for an irregular 3D point set.

To generalize convolution operations to irregular 3D point sets, researches have proposed various convolution operation on 3D point cloud. The commonly used form can be expressed as,

$$FConv(p_i) = \sum_{p_j \in R_i} \mathcal{K}\left(p_i, p_j\right)\mathcal{F}(p_i) \tag{1}$$

where, for each $p_i \in S$ we find all its neighborhood points p_j in the local region R_i centered at p_i. $\mathcal{F}(p_i)$ represents the features at point p_i and $\mathcal{K}(p_i, p_j)$ measures the correlation between point p_i and its neighbor point p_j. For example, KC-Net (Shen et al., 2018) use Gaussian kernel to account for function $\mathcal{K}(p_i, p_j)$. In Wang et al. (2018), the authors use MLP to implement the kernel function $\mathcal{K}(p_i, p_j)$ which enables learning of a parametric continuous convolution (Wang et al., 2018).

3.2. Density-aware convolution

In this paper, we aim to enhance this convolution operation from the perspective of the density distribution of 3D point clouds. Our key observation is that the irregular structure of 3D points gives rise to over-sampling and under-sampling of certain regions and resulting in disparate densities across the convolution kernel.

An analogy of this disparity in 2D images will be receiving multiple values for one pixel. This creates a bias when we update the kernel, putting excessive weights on the over-sampled regions and vice versa. In digitized images, pixels are placed evenly on a 2D grid where distances between neighboring pixels are constant; every cell in the grid has exactly one value and there are no empty cells. This results in uniform density across the grid and unfortunately, does not apply to irregular 3D point clouds.

To counter this imbalance, in Hermosilla et al. (2018), Wu et al. (2019) the authors design a new convolution operation that applies the estimated densities in local regions to re-scale the kernel functions. The convolution function at point p_i is defined as:

$$DConv(p_i) = \sum_{p_j \in \mathcal{N}_i} \frac{1}{\mathcal{D}(p_j|p_i)}\mathcal{K}\left(p_i, p_j\right)\mathcal{F}(p_i) \tag{2}$$

where denotes the neighborhood of point p_i. $\mathcal{D}(p_j|p_i)$ represents the probability density function at point p_j given center point p_i. $\mathcal{F}(p_i)$ and $\mathcal{K}(p_i, p_j)$ have the same meanings as in Eq. 1. Usually, a multiple layer perceptron (MLP) network is used to implement the kernel function.

In this paper, we claim that directly use the inversed density to reweight the convolutional kernel is not sufficient to make full use of the density information. We, therefore, introduce the following convolution function,

$$DConv(p_i) = \sum_{k=1}^{K} \frac{1}{\mathcal{D}(p_j)}\mathcal{K}\left(p_i, p_j\right)\mathcal{F}(p_i), \tag{3}$$

where $\mathcal{D}(p_j)$ denotes the density at point p_i and K denotes the number of neighbor points. If less than K points are within a certain radius of the center point p_i, we fill the neighborhood with the center point itself to ensure a fixed size of the neighborhood at each point location. Fig. 2 gives an illustration of the proposed density-aware convolution module. Unlike Monto Carlo Convolution which uses a dynamic size neighborhood, our method uses a fixed size of the neighborhood to enable efficient matrix multiplication.

3.3. Adaptive density estimation

To compute the inverse density function $\mathcal{D}$, we think of the local region around any $p_j \in S$ as a ball with radius R and we apply the Parzen-Rosenbatt window method (Parzen, 1962) to fit density distributions along each of the axes inside the ball.

$$\mathcal{D}(p_j)_{KDE} = \frac{1}{|\mathcal{N}_j|h}\sum_{p_k \in \mathcal{N}_j} G\left(\frac{p_j - p_k}{h}\right), \tag{4}$$

where $\mathcal{N}_j$ denotes the neighborhood of point p_j and $|\mathcal{N}_j|$ stands for the neighborhood size, h represents a smoothing factor.

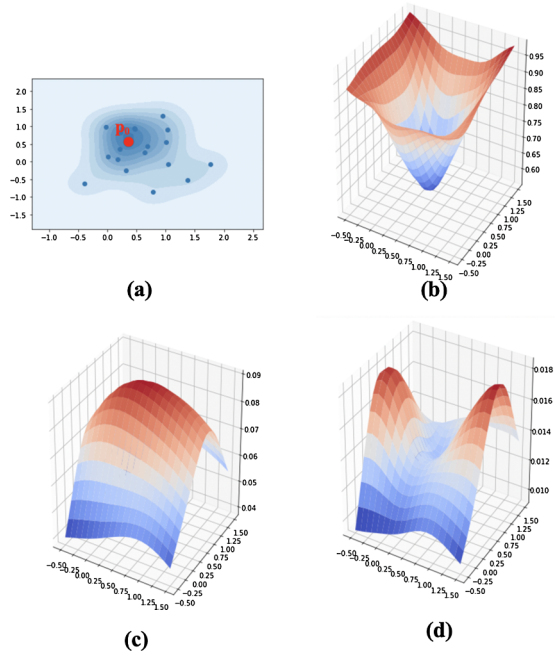

Fig. 2. Illustration of the proposed density-aware convolution module. (a) Shows the sampled point set in the local region centered at p_0; our convolution module uses the relative distance between p_0 and other points to predict (b), the continuous inverse density function; Then we multiply the weight function (c) learned from the training data with the inverse density function (b), and produce (d) which approximates the weight function for the same scene but uniformly sampled.

To encourage the convolution module to adaptively decide how to use the density estimations, we then pass these density estimations to an MLP network for non-linear transformations which provides more refined approximations to the continuous density estimations. The weights of the MLP network here are shared across all points in order to encourage the permutation invariance. The inverse density at point p_j is thus formulated as,

$$\mathcal{D}(p_j) = \frac{1}{MLP(\mathcal{D}(p_j)_{KDE})} \tag{5}$$

3.4. Context encoding module

A common problem in point cloud classification is that different object classes have very different numbers of training samples (Batista et al., 2004). This could happen by non-uniform sampling where homogeneous clusters are over-sampled, or simply by the natural distribution of these classes. Regardless of the cause, this discrepancy in the number of training samples for each class introduces an issue where the classes with small numbers of samples get under-fitted and are likely to be misclassified as the most frequent class in the process of maximizing accuracy.

In this section, we propose an auxiliary task of predicting the context of the input point cloud to help guide the segmentation task. Given a patch of point set S as a scene, we define the context vector as an indicator for the presence of different classes in this specific scene. Given a cluster of points $B_i = \{p_1, ..., p_N | p_j = (x, y, z, c) \in S\}$, where c is the categorical label for each point, we define the ground truth global context vector C^{gt} as in Eq. (6)

$$C_i^{gt} = \begin{cases} 1 & \text{if } \exists\ p_j\ s.t.\ \ c_j = i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $|C^{gt}| = k$ denotes the total number of classes, C_i^{gt} is the i-th

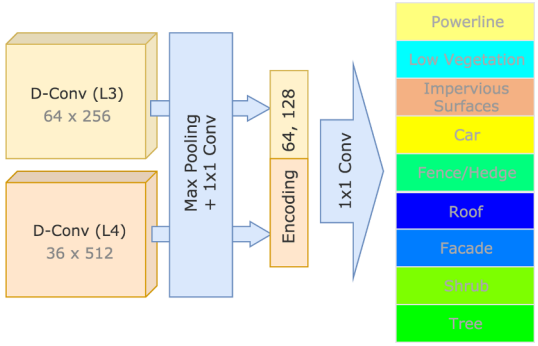

Fig. 3. Illustration of the proposed context encoding module. 'L3' and 'L4' denotes the feature maps after 3 and 4 downsampling layers respectively.

element of C. The resulting C^{gt} represents the set of all possible contexts with the cardinality $|S_{scene}| = 2^k$.

With this formulation of context, an under-represented class, if present in the scene, will take the same value of 1 in the context vector C as with other more frequently sampled classes. Therefore, predicting the context vector becomes the task of predicting the statistical property on the existence of different classes in the scene. This, in turn, constrains the main classification predictions by giving more weights to the under-sampled classes and pushing the model to evaluate with a global view of the input set.

To generate the context vector prediction, we incorporate multi-scale global feature learning where the network leverages features learned from multiple point densities and merges them using nonlinear transform into a compact encoding. This encoding is then fed to a multi-layer perceptron (MLP) to predict a context vector C, which indicates the probability of the existence of each category. Fig. 3 illustrates the mechanism of our proposed context encoding module (see Fig. 4).

We use binary cross entropy loss between the predicted context vectors C_i and the ground truth ones C_i^{gt} to regularize the context information, calculated as,

$$\mathcal{L}_{ctx} = \sum_{i=1}^{|C|} C_i^{gt} * log(C_i) \tag{7}$$

We then put this context regularization term to our final loss function with, calculated as,

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda * \mathcal{L}_{ctx} \tag{8}$$

$$\mathcal{L}_{cls} = \sum_{j=1}^{N} \sum_{c=1}^{|C|} \left[y_{jc} log p_{jc} + \left(1 - y_{jc}\right) log \left(1 - p_{jc}\right) \right] \tag{9}$$

where $\mathcal{L}_{cls}$ and $\mathcal{L}$ denote the classification loss and the final loss function respectively. λ denotes a hyper-parameter to balance the classification loss and context encoding loss. By default, we set λ to 1 in our experiments.

3.5. Network architecture

The design of our DANCE-NET follows the widely used encoder-decoder architecture where the input point clouds are firstly down-sampled multiple times to learn hierarchical feature embeddings. Then, the learned embeddings are up-sampled back to the original point sets for the semantic point classification. A context encoding module is integrated between the down-sampling and up-sampling path for the context encoding and regularization.

3.5.1. Downsampling and upsampling blocks

Following the success of neural networks that learn high-level

Fig. 4. Overview of our proposed DANCE-Net method. Our model starts with an encoder network to extract high-level contextual features using the newly proposed density-aware convolution layers. Then a context encoding module is adopted to learn the global context encoding, which was further fed into a regularization module to encourage the predicted context encoding to be aligned with the ground truth one. Finally, a decoder network with successive density-aware convolution module and upsampling block were used to generate per-point classification results.

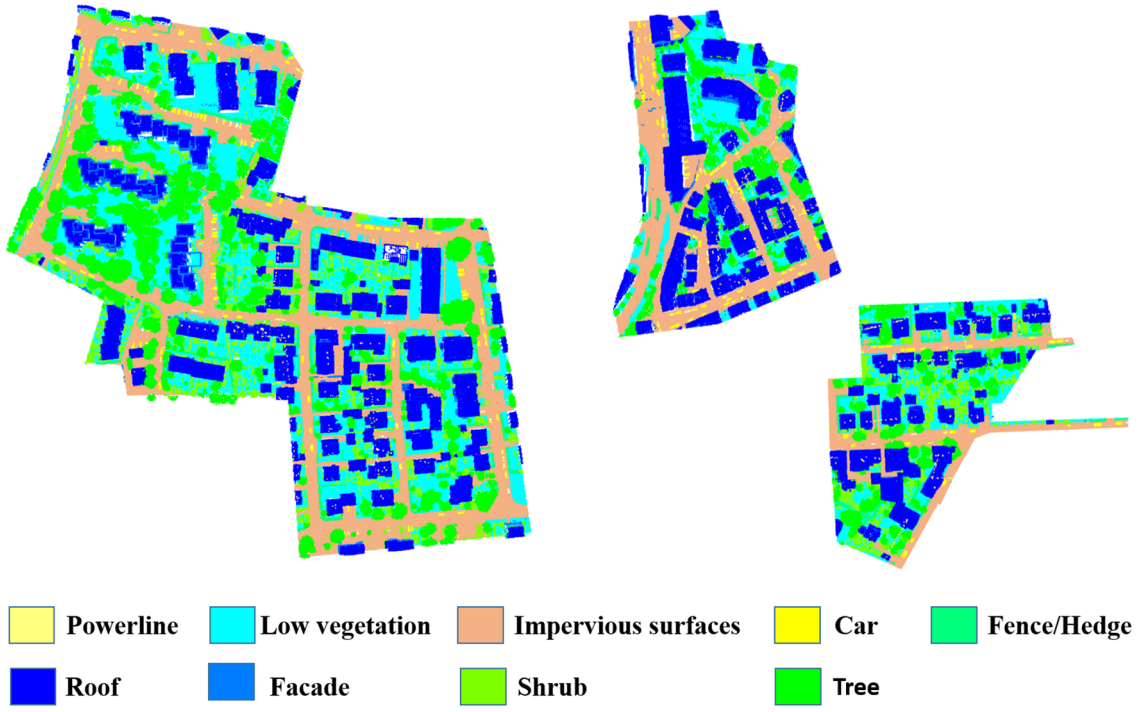

Fig. 5. ISPRS 3D labeling dataset. The left scene is used as the training set, and the right two scenes are used as the test set. Here we plot the ground truth labels for each point.

features from multiple scales, we build our model with down-sampling blocks, producing point feature sets that are successively sparser and more complex, and up-sampling blocks that interpolate the sparse set back to the original points. Our downsampling and upsampling blocks are designed following (Qi et al., 2017b). We present a short explanation for the two modules below.

The down-sampling block takes a point set $S = \{p_1, ..., p_n | p_i = (x, y, z, f)\}$ where x, y, z are coordinates of the point and f is the feature vector, and outputs a new set $\hat{S} \subseteq S$ that contains a reduced number of points. In other words, the size of the output for the i-th down-sampling block $|S_i|$ is less than to $|S_{i-1}|$. To pick the candidates of this new set S_i, we use farthest point sampling which selects the set of points that each of the selected points has the largest distance possible from the rest of the set. After selecting the candidates, we group the points in their respective local regions and feed them to our convolution module, obtaining density-weighted embeddings for each candidate. We define local regions with a ball query with a hyper-parameter on the search radius.

The up-sampling block takes two point sets S_i and S_{i-1} which corresponds to the output of the i-th and $(i-1)$-th down-sampling block. It generates a point set $\bar{S} \supseteq S_i$ that contains the same points like in S_{i-1}, but has different features. In the up-sampling process, we first take both sets and compute the distance between the known points S_i and the points S_{i-1} that we want to interpolate to. We normalize the inverse of these distances to the range of $(0, 1)$ and use it to weight the interpolated point features. Instead of directly predicting a continuous inverse density function for interpolation, we adopt this straightforward re-scaling method to reduce the number of parameters and still ensure that the embeddings are reweighted by some measure of density. Finally, we concatenate the feature vectors in S_{i-1} with those of the interpolated set $\bar{S}_i$ for 1×1 convolution and extract high-level features of the desired dimension.

3.5.2. Overall architecture

To present our network in more detail, the down-sampling stage consists of four down-sampling blocks reducing the size of input point sets to 1024, 256, 64, and 36 sequentially, each having the channel size 64, 128, 256 and 512 respectively. The point features outputted by the third and fourth blocks are passed to the context encoding module where each of the block's output is max-pooled and condensed by 1×1 convolution to feature maps with 1/4 of the original channel size. We then concatenate these encodings, feed the result to an MLP, and compute the binary cross-entropy loss (BCE) with the ground truth context. In the up-sampling stage, we use four up-sampling blocks to raise the size of each point set to 64, 256, 1024, and to the original size. To preserve low-level information, we incorporate the features learned in the down-sampling stage and use skip-connections to link them to up-sampling blocks of the same output size. For example, the first up-sampling block will first interpolate the 36-dimensional point features of the fourth down-sampling block to 64-dimensional and then concatenate them with the 64-dimensional point features learned by the third down-sampling block. This incorporation of low-level features has seen various successes in preceding works for 2D segmentation, including performance improvement and a faster convergence (Ronneberger et al., 2015; Badrinarayanan et al., 2017).

4. Experiments and results

In this section, we conduct experiments to demonstrate the effectiveness of the proposed DANCE-NET model for airborne LiDAR point cloud labeling. We introduce the experimental dataset and data prepossessing in Section 4.1 and Section 4.2 respectively. Evaluation metrics are given in 4.3. In Section 4.4, we present the experimental setups. The classification results are presented in Section 4.5.

4.1. Experimental dataset

We conduct our experiments on the International Society for Photogrammetry and Remote Sensing (ISPRS) 3D labeling dataset (Niemeyer et al., 2014) and 2019 Data Fusion Contest Dataset (Bosch et al., 2019; Le Saux et al., 2019). The ISPRS 3D labeling dataset contains airborne LiDAR point clouds covering three isolated areas of Vaihingen city. Each point in the dataset is labeled in nine categories, including powerline, low vegetation (low_veg), impervious surface

(imp_surf), car, fence/hedge (fen/hed), roof, facade, shrub, and tree.

Following the benchmark settings of the ISPRS 3D labeling contest, the whole dataset is split into two parts. The first scene (Fig. 5 left) is used as the training set and the remaining two scenes (Fig. 5 right) are used as the test set. For each scene, a simple ASCII file with XYZ, reflectance, return count information, and point labels were provided in an ASCII file. We also generate height above ground feature using LasTools (Hug et al., 2004) and use it as additional inputs for our method.

The 2019 Data Fusion Contest Dataset is collected by the IEEE Geoscience and Remote Sensing Society (GRSS) and covers approximately 100 km^2 over Jacksonville, Florida, Omaha, Nebraska, and the United States. All points in this dataset are annotated in 5 semantic categories, including ground, high vegetation, buildings, water, and bridge deck. Each point contains five attributes, including x-y-z coordinates, intensity, and return number. In our experiments, we use x,y,z coordinates as well as the intensity as the model inputs. Results on 2019 Data Fusion Contest Dataset are shown in Section 5.4

4.2. Data prepossessing

Original ISPRS 3D labeling dataset covers a large area with each scene contains more than 100 K points. Such a large point set cannot be directly fed into our model for network training due to the limited GPU memory. To facilitate training, we divide the training scene into small patches and only use a small batch of patches for model training at each training step, see Fig. 6 for illustration. More specifically, we divided the training and test scenes into regular blocks in the horizontal direction, each with a size of 30 m $*$ 30 m. Note that each block contains a different number of points. In the training phase, we randomly choose a bunch of blocks and sample a fixed number of points (e.g., 8192 points) from each block to formulate a batch of input point sets. To further improve the robustness impede over-fitting, we randomly drop a certain number of points at each block during model training. In the default setting, the dropout ratio is set to 12.5% in our experiments. Another thing worth mentioning is that, due to the irregular boundary of each scene, some edge blocks may contain a small number of points and thus cannot be effectively classified due to the limited geometric information. We thus merge those edge blocks with their surrounding blocks with more points.

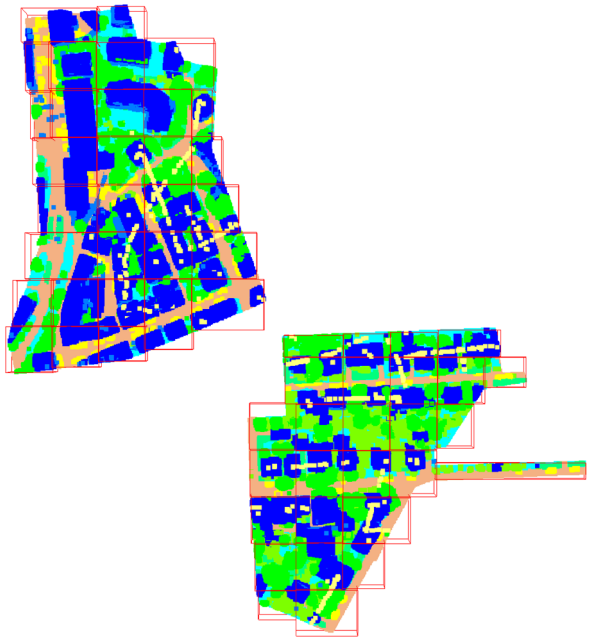

Fig. 6. The spatial distribution of the test blocks. And small blocks broken on the edge were merged into surrounding large blocks.

As the inference stage, all points in each block are directly fed into our DANCE-Net model for point labeling. Here we do not need point sampling for each block because our model by nature is a fully convolutional network and can arbitrary size of input point sets. We can easily merge the label predictions from each test block to generate the final prediction results.

Further investigating the ISPRS 3D labeling dataset, one can find out that there is quite a different number of points for each object category. Directly training on this unbalanced dataset may cause the issue where the classes with small numbers of points get under-fitted and are likely to be misclassified by deep neural networks. To address this issue, we introduce the context encoding module in Section 3.4 to force context prediction to be aligned with the ground truth. Here, we introduce a more simple strategy to achieve the same goal. In order to encourage our model to focus more on those categories with fewer points, we incorporate a category-specific weight factor for each class to reweight the classification loss. The weight of each category is computed by the inverse of the logarithm of the ratio of that category, formulated as Eq. 10.

$$W_i = \frac{1}{\ln\left(\alpha + \frac{N_i}{\sum_{i=1}^{|C|} N_i}\right)} \tag{10}$$

where W_i denotes the balance weight of the ith category, N_i represents the number of points in that category, $|C|$ denotes the number of categories, α is a hyper-parameter for class balance. We set α to 1.2 according to experimental evaluation. By integrating the category-specific weights, we re-formulate Eq. 9 as:

$$\mathcal{L}_{cls} = \sum_{j=1}^{N} w_j * \sum_{c=1}^{|C|} \left[y_{jc} log p_{jc} + \left(1 - y_{jc}\right) log\left(1 - p_{jc}\right) \right] \tag{11}$$

where N indicates the number of sampled points at each training step, p_{jc} and y_{jc} represent predicted probability and ground truth label of the jth point on the cth category, and w_j indicates the balance weight for the jth sampling point and is calculated as $w_j := W_k | y_{jk} = 1$.

4.3. Evaluation metrics

The performance of our model is evaluated by two commonly used metrics, i.e., overall accuracy and F1 score. Overall accuracy (OA) measures the classification performance for all categories as a whole, which is calculated as the ratio of correctly classified points over the total test sets. While F1 score deal with each category separately and takes into account both the precision and recall values. F1 score is more suitable for classification performance evaluating in the case of uneven class distribution. The calculation of overall accuracy and F1 score are formulated as follows:

$$OA = \frac{tp + tn}{tp + tn + fp + fn} \tag{12}$$

$$precision = \frac{tp}{tp + fp} \tag{13}$$

$$recall = \frac{tp}{tp + fn} \tag{14}$$

$$F1 = 2 \times \frac{precision}{precision + recall} \tag{15}$$

where tp (true positive)/tn (true negative) represents the number of the positive/negative points that are correctly classified by the classifier and fp (false positive)/fn (false negative) denote the negative/positive points that are incorrectly classified by the classifier.

4.4. Experimental setup

Our DANCE-NET method is implemented using the Tensorflow framework. In the training phase, the batch size is set to 6. Adam optimizer with initial learning of 0.01 is used to train our network, and we divide the learning rate by 2 every 3000 steps. It takes about 10 h to train our DANCE-Net model for 1000 epochs until convergence on a single TESLA K80 GPU. Our code is available at https://github.com/lx709/DANCE-NET.

In the downsampling and upsampling blocks, we set a base neighborhood size of 2 m for the input point clouds. In the downsampling stage, with the decrease of the point cloud sizes, we multiply the neighborhood size by 2 after each downsampling block. Similarly, in the up-sampling stage, with the increase of the size of the point cloud, we divided the neighborhood size by two after each upsampling block. The bandwidth for KDE follows the same rules but has its base value set as 1 m. The radius for KDE estimation (R) is set to 2 m.

4.5. Classification results

We first investigate the performance of our DANCE-Net model under different hyper-parameter configurations of searching radius R and λ. Experimental results demonstrate that our DANCE-NET model gets quite stable performance with different settings and reaches the best performance when R and λ are set to 2 m and 1 respectively. We use this configuration as our default setting. We visualize the classification results with the best hyper-parameter configurations in Fig. 7. As shown in Fig. 7, the proposed DANCE-NET model gets a satisfying classification result where most of the points in the test scenes are correctly classified.

Table 1 lists the performance of our model, where we report precision, recall and F1 score of each category as well as the classification confusion matrix. As can be seen in Table 1, our DANCE-Net model achieves an F1 score higher than 70% on 5 out of 9 categories and obtains quite satisfying performance on impervious surface and roof categories. Moreover, our DANCE-Net model gets an acceptable performance on the facade category as indicated by the F1 score. While the classification performance on shrub and fence/hedge categories are relatively lower. As indicated by Table 1, many fence/hedge points are misclassified as shrub. This is probably because the fence/hedge category has fewer points and shows similar structural characteristics with the shrub category, which causes our model to be not fully trained and thus hinders it from differentiating these two categories. Though powerline and car categories also have a small number of points in the dataset, they present quite different structures from other categories and thereby acquired higher classification performance. For example, one can easily identify a powerline in the point cloud by checking the characteristics: the object is on the ground, the object is almost perpendicular to the ground, the object is tall, the object is very thin (with very small variations in x and y directions). These characteristics make it easy for a model to identify those powerline points.

5. Discussion

In this section, we firstly demonstrate the effectiveness of our newly proposed density-aware point convolution module and context encoding loss module in Section 5.1 and 5.2. Then, in Section 5.3 we compare the performance of our model with other best-performing methods. We show further experimental results on 2019 Data Fusion Contest Dataset in Section 5.4.

5.1. Effect of density encoding

To investigate the effect of our newly proposed density-aware point convolution module and context encoding module, we develop three models: (a) the model without density encoding module and context encoding module, we mark this model as ‘baseline’, (b) our model with density encoding only, marked as ‘ours (with density encoding)’, (c) the model with both density encoding and context encoding, marked as ‘ours (final)’. The classification performances of these models are listed in Table 2. Qualitative comparison are illustrated in Fig. 8.

As shown in Table 2, with a density-aware encoding convolution module, our DANCE-Net model gets an improvement of 0.8% and 2.1% on overall accuracy and average F1 score respectively. Comparing Fig. 8 (b) and Fig. 8(c), one can see that our model successfully corrects some misclassified roof points by introducing the density-aware encoding convolution module.

5.2. Effect of context encoding module

As shown in Table 2, with the proposed context encoding module, our model gets an extra improvement of 0.6% on overall accuracy and 1.0% on average F1 score. Comparing Fig. 8 (c) and Fig. 8(d), one can find out that our model successfully corrects a large number of misclassified roof points (211 corrected points vs. 2761 roof points) and several impervious points (824 corrected points vs. 7791 impervious surface points).

5.3. Comparison with other methods

We compare the performance of our DANCE-NET model with the top three models on the IRPRS 3d labeling benchmark, including RIT_1 (Yousefhussien et al., 2018), NANJ2 (Zhao et al., 2018) and WhuY4 (Yang et al., 2018). Table 3 lists the classification performance of our DANCE-NET model and all comparing methods. As can be seen in Table 3, our DANCE-Net model achieves superior performance than all

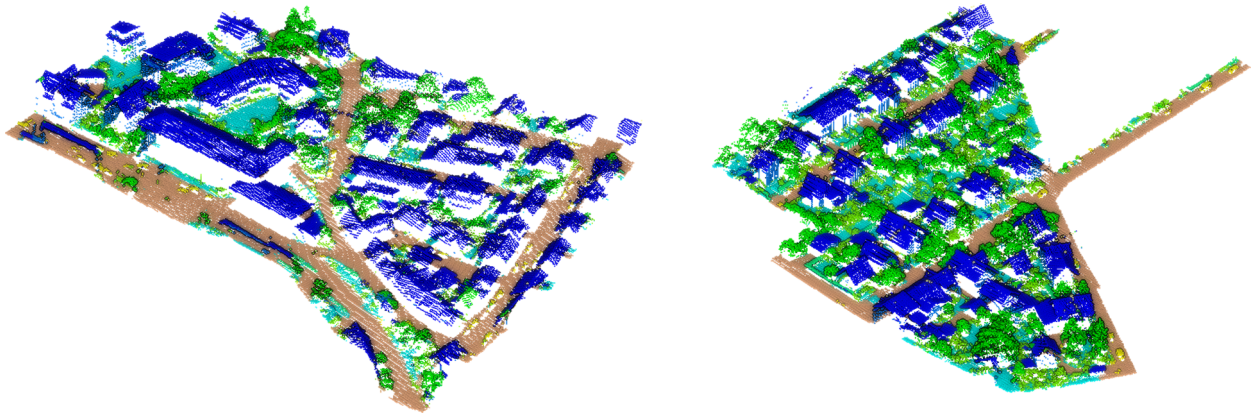

Fig. 7. The classification results of our DANCE-NET model on ISPRS test set. Results are shown in front view with eye dome lighting.

Table 1
The classification confusion matrix of our DANCE-NET model. Precision, recall, and F1 score of each class are also reported. Note that numbers in the confusion matrix are normalized along each column.

Categories	powerline	low_veg	imp_surf	car	fence_hedge	roof	facade	shrub	tree
powerline	**0.668**	0.000	0.000	0.000	0.000	0.001	0.001	0.000	0.000
low_veg	0.002	**0.771**	0.051	0.042	0.075	0.014	0.053	0.121	0.015
imp_surf	0.000	0.091	**0.946**	0.008	0.023	0.001	0.009	0.006	0.002
car	0.000	0.001	0.001	**0.745**	0.02	0.000	0.007	0.009	0.001
fence_hedge	0.000	0.006	0.000	0.013	**0.283**	0.000	0.003	0.022	0.002
roof	0.228	0.009	0.001	0.026	0.047	**0.937**	0.178	0.055	0.026
facade	0.015	0.003	0.000	0.010	0.017	0.010	**0.535**	0.019	0.011
shrub	0.008	0.095	0.001	0.156	0.399	0.011	0.148	**0.580**	0.109
tree	0.078	0.024	0.000	0.000	0.136	0.026	0.067	0.188	**0.834**
Precision	0.668	0.771	0.946	0.745	0.283	0.937	0.535	0.580	0.834
Recall	0.700	0.866	0.910	0.802	0.606	0.942	0.690	0.398	0.795
F1 score	0.684	0.816	0.928	0.772	0.386	0.939	0.602	0.472	0.814

Table 2
Ablation analysis of our DANCE-NET model.

Method	OA	Average F1
baseline	0.825	0.681
Ours (with density-encoding)	0.833	0.702
Ours (final)	0.839	0.712

comparing methods as indicated by the average F1 score, which surpasses the best performed WhuY4 (Yang et al., 2018) model by 2.0%. Also, our DANCE-Net model obtains an overall accuracy of 83.9%, which is comparable to the best performed NANJ2 (Zhao et al., 2018) model with an overall accuracy of 85.2%. Note that the NANJ2 method uses additional corresponding RGB features and hand-crafted features (e.g., roughness) as input for point cloud classification, while our model only uses raw XYZ coordinates as inputs. More specifically, the proposed model achieves remarkable higher performance on the powerline, impervious surface, car, and facade category.

We further investigate the performance of our model by comparing the qualitative results with the top three models reported in the benchmark. Fig. 9 shows the classification results generated by all comparing methods and the red circle indicates where the comparing methods get incorrect predictions and our model gets correct predictions. For example, by comparing Fig. 9(b), (c), (d) and (e), one can find out that RIT_1 misclassified some low vegetation points as tree and

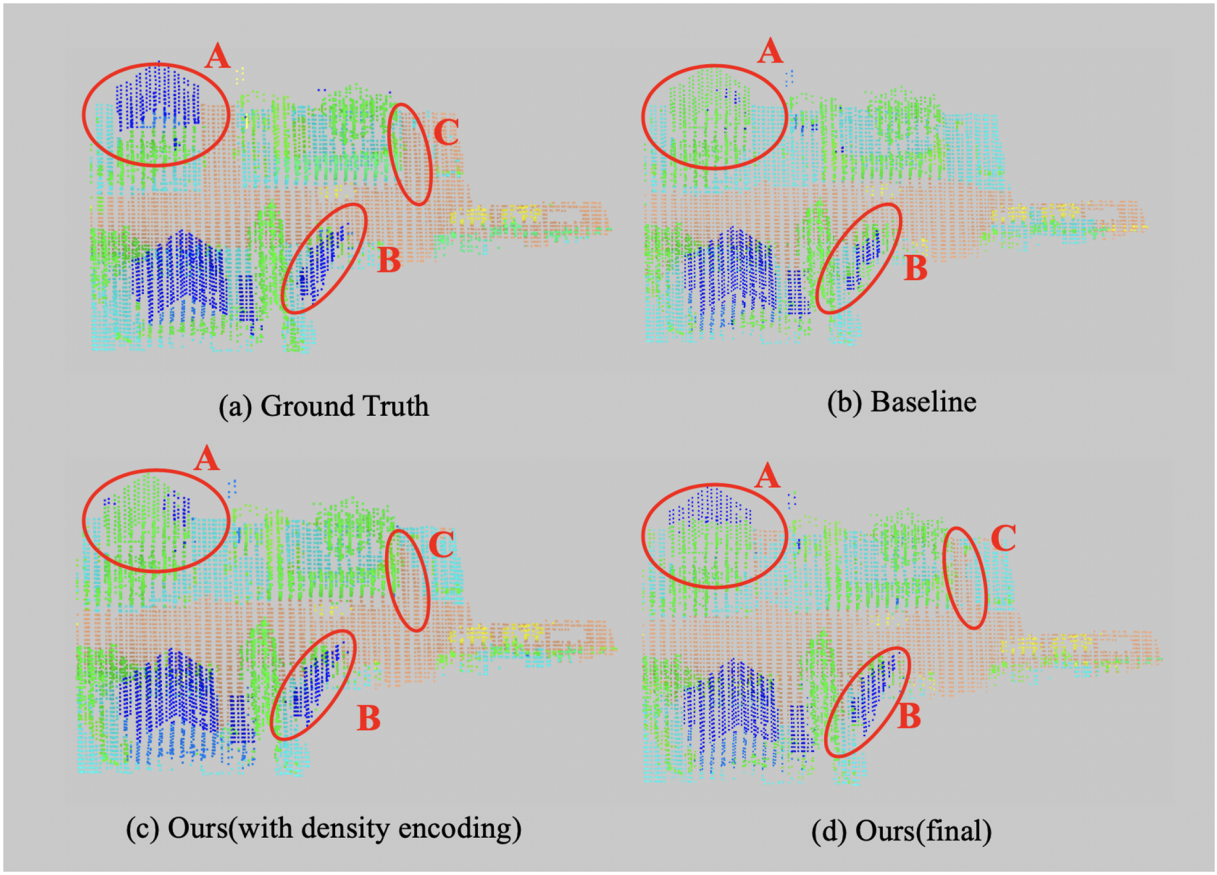

Fig. 8. The classification results with different model configurations. (a) Ground truth labels, (b) baseline model, (c) our model with density-encoding and (d) our model (final). The red circled part highlight the differences between different models. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Classification performance of our model and the top three models on the ISPRS benchmark. We report the per-category F1 score in the first 9 columns as well as the overall accuracy (OA) and average F1 score (Average F1) in the last two columns. The boldface text indicates the best performance.

Models	powerline	low_veg	imp_surf	car	fence_hedge	roof	facade	shrub	tree	OA	Average F1
RIT_1 (Yousefhussien et al., 2018)	0.375	0.779	0.915	0.734	0.180	0.940	0.493	0.459	0.825	0.816	0.633
NANJ2 (Zhao et al., 2018)	0.620	**0.888**	0.912	0.667	0.407	0.936	0.426	**0.559**	0.826	**0.852**	0.693
WhuY4 (Yang et al., 2018)	0.425	0.827	0.914	0.747	**0.537**	**0.943**	0.531	0.479	0.828	0.849	0.692
Ours	**0.684**	0.816	**0.928**	**0.772**	0.386	0.939	**0.602**	0.472	0.814	0.839	**0.712**

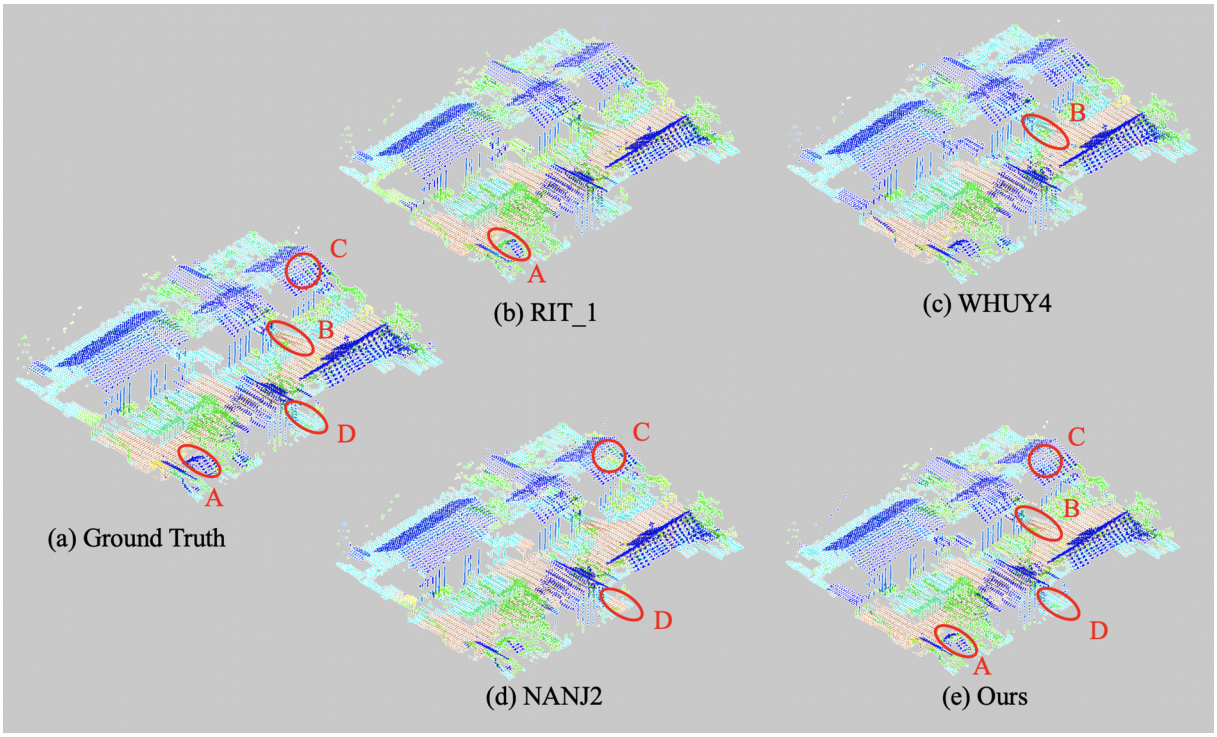

Fig. 9. Qualitative comparison of the classification results in a selected complicated scene. Red circle indicates those areas where our model performance better than the comparing methods.

Table 4
Performance comparison between our method and other prevalent point cloud-based models. We report the F1 score for each category in the first 9 columns as well as the overall accuracy (OA) and average F1 score (Average F1) in the last two columns. The boldface text indicates the best performance.

Method	powerline	low_veg	imp_surf	car	fence_hedge	roof	facade	shrub	tree	OA	Average F1
PointNet++ (Qi et al., 2017b)	0.579	0.796	0.906	0.661	0.315	0.916	0.543	0.416	0.770	0.812	0.656
PointSIFT (Jiang et al., 2018)	0.557	0.807	0.909	0.778	0.305	0.925	0.569	0.444	0.796	0.822	0.677
PointCNN (Li et al., 2018c)	0.615	**0.827**	0.918	0.758	0.359	0.927	0.578	0.491	0.781	0.833	0.695
PointCNN + A-XCRF (Arief et al., 2019)	0.630	0.826	0.919	0.749	**0.399**	**0.945**	0.593	**0.508**	**0.827**	**0.850**	0.711
KPConv (Thomas et al., 2019)	0.631	0.823	0.914	0.725	0.252	0.944	0.603	0.449	0.812	0.837	0.684
D-FCN (Wen et al., 2020)	**0.704**	0.802	0.914	**0.781**	0.370	0.930	**0.605**	0.460	0.794	0.822	0.707
Ours	0.684	0.816	**0.928**	0.772	0.386	0.939	0.602	0.472	0.814	0.839	**0.712**

Table 5
Performance of our model and all comparing methods on the 2019 Data Fusion Contest Dataset. We report the F1 score for each category in the first 5 columns as well as the overall accuracy (OA) and average F1 score (Average F1) in the last two columns. The boldface text indicates the best performance.

Method	Ground	High Vegetation	Building	Water	Bridge Deck	OA	Average F1
PointNet++ (Qi et al., 2017b)	0.983	0.958	0.797	0.044	0.073	0.927	0.571
PointSIFT (Jiang et al., 2018)	0.986	0.970	0.855	0.464	0.604	0.940	0.776
PointCNN (Li et al., 2018c)	0.987	0.972	0.849	0.441	0.653	0.938	0.780
KPConv (Thomas et al., 2019)	0.984	0.942	0.874	0.430	0.775	0.945	0.801
D-FCN (Wen et al., 2020)	**0.991**	**0.981**	**0.899**	0.450	0.730	0.956	0.810
Ours	**0.991**	0.939	0.870	**0.583**	**0.839**	**0.968**	**0.844**

WHUY4 method misclassified some tree points as roof (area 'D'), while NANJ2 method misclassified some roof points as tree (area 'E'). Our got correct predictions for all those challenging areas (including 'A', 'B', 'C', 'D', 'E').

Moreover, we compare the performance our DANCE-Net model with several recent proposed point cloud-based models in both remote sensing and computer vision fields, including PointNet++ (Qi et al., 2017b), PointSIFT (Jiang et al., 2018), PointCNN (Li et al., 2018c), KPConv (Thomas et al., 2019), PointCNN with A-XCRF model (Arief et al., 2019) and D-FCN (Wen et al., 2020). Classification results of all comparing methods are listed in Table 4. From Table 4 one can see that the proposed DANCE-NET model achieves better performance than all comparing models as indicated by the average F1 score. Note that PointCNN with A-XCRF model gets a quite close F1 score, but it utilizes a further post-processing step to refine the classification results, while our model does not involve any post-processing techniques. Moreover, from Table 4 one can see that our model achieves better performance than the vanilla PointCNN model.

One may also note that our DANCE-Net model falls behind the state-of-the-art method on the overall accuracy. This is mainly because our model used a weighted loss to rebalance the learning process for each category, see Section 4.2. Another reason also we use average F1 score to determine the model hyper-parameters rather than referring to overall accuracy. Considering that the ISPRS 3D labeling dataset contains a very different number of points for each category, focusing on overall accuracy may lead to those minority categories being ignored, we, therefore, mainly focus on the F1 score in this paper.

5.4. Results on 2019 data fusion contest dataset

In this section, we carry out further experiments on the 2019 Data Fusion Contest Dataset to show the abilities of our DANCE-NET model for dealing with large scale datasets. Following previous researches (Wen et al., 2020), 100 regions are used as the training set and the remaining 10 regions as the test set. Similar to the settings in Section 4.4, in the training phase, we randomly select a group of $128\,m * 128\,m * 210\,m$ cuboid block and sample 14000 points from each cuboid as model inputs. In the test phase, we divide each test region into $128\,m * 128\,m$ blocks in the horizontal direction and feed all points in each block to our DANCE-Net model for classification. In our experiment, the radius for KDE estimation (R) is set to 3 m.

We compare the performance of DANCE-Net model with several prevalent point cloud-based classification methods, including PointNet++ (Qi et al., 2017b), PointSIFT (Jiang et al., 2018), PointCNN (Li et al., 2018c), (Hermosilla et al., 2018) MCCNN, KPConv (Thomas et al., 2019) and D-FCN (Wen et al., 2020). The classification performance of DANCE-NET model and all comparing methods are listed in Table 5. As shown in Table 5, our DANCE-NET model achieves superior performance than all comparing methods on both overall accuracy and average F1 score. More specifically, our model achieves significant better performance on 'water' and 'bridge deck' categories.

6. Conclusions

In this paper, we introduce a novel density encoding-aware network with a context encoding module to achieve the classification of 3D point clouds. To better model the spatial distribution of unevenly distributed 3D point sets, a density-aware convolution module is designed by re-weighting the weights of convolution kernels using the inversed density of each point. Based on the proposed convolution module, we further build a multi-scale CNN model to achieve point cloud semantic labeling. Moreover, to deal with the issue of imbalanced class distribution of outdoor point clouds, we develop a context encoding module to encourage the predicted context to be aligned with the ground truth context information. Experimental results on the ISPRS 3D labeling dataset and 2019 Data Fusion Contest Dataset demonstrate the effectiveness of our proposed density-aware convolution module and context encoding module. The proposed method achieves new state-of-the-art performance on both datasets.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank for NYUAD Institute (AD131) for providing partial financial support for this work. The Vaihingen data set was provided by the German Society for Photogrammetry, Remote Sensing, and Geoinformation (DGPF) [Cramer, 2010]: http://www.ifp.uni-stuttgart.de/dgpf/DKEP-Allg.html. The authors would like to thank the Johns Hopkins University Applied Physics Laboratory and IARPA for providing the data used in this study, and the IEEE GRSS Image Analysis and Data Fusion Technical Committee for organizing the Data Fusion Contest (http://www.grss-ieee.org/community/technical-committees/data-fusion).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.isprsjprs.2020.05.023.

References

Arief, H.A., Indahl, U.G., Strand, G.-H., Tveite, H., 2019. Addressing overfitting on pointcloud classification using atrous xcrf. arXiv preprint arXiv:1902.03088.
Badrinarayanan, V., Kendall, A., Cipolla, R., 2017. Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39 (12), 2481–2495.
Batista, G.E.A.P.A., Prati, R.C., Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor. Newsletter 6 (1), 20.
Bosch, M., Foster, K., Christie, G., Wang, S., Hager, G.D., Brown, M., 2019. Semantic stereo for incidental satellite images. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, pp. 1524–1532.
Chehata, N., Guo, L., Mallet, C., 2009. Airborne lidar feature selection for urban classification using random forests. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. 38 (Part 3), W8.
Cheng, G., Zhou, P., Han, J., 2016. Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images. IEEE Trans. Geosci. Remote Sens. 54 (12), 7405–7415.
Colgan, M.S., Baldeck, C.A., Féret, J.-B., Asner, G.P., 2012. Mapping savanna tree species at ecosystem scales using support vector machine classification and brdf correction on airborne hyperspectral and lidar data. Remote Sens. 4 (11), 3462–3480.
Ene, L.T., Næsset, E., Gobakken, T., Bollandsås, O.M., Mauya, E.W., Zahabu, E., 2017. Large-scale estimation of change in aboveground biomass in miombo woodlands using airborne laser scanning and national forest inventory data. Remote Sens. Environ. 188, 106–117.
García-Gutiérrez, J., Mateos-García, D., Garcia, M., Riquelme-Santos, J.C., 2015. An evolutionary-weighted majority voting and support vector machines applied to contextual classification of lidar and imagery data fusion. Neurocomputing 163, 17–24.
García, M., Riaño, D., Chuvieco, E., Salas, J., Danson, F.M., 2011. Multispectral and lidar data fusion for fuel type mapping using support vector machine and decision rules. Remote Sens. Environ. 115 (6), 1369–1379.
Hermosilla, P., Ritschel, T., Vázquez, P.-P., Vinacua, À., Ropinski, T., 2018. Monte carlo convolution for learning on non-uniformly sampled point clouds. In: SIGGRAPH Asia 2018 Technical Papers. ACM, p. 235.
Hu, F., Xia, G.-S., Hu, J., Zhang, L., 2015a. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. Remote Sens. 7 (11), 14680–14707.
Hu, W., Huang, Y., Wei, L., Zhang, F., Li, H., 2015b. Deep convolutional neural networks for hyperspectral image classification. J. Sens.
Hua, B.-S., Tran, M.-K., Yeung, S.-K., 2018. Pointwise convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 984–993.
Hug, C., Krzystek, P., Fuchs, W., 2004. Advanced lidar data processing with lastools. In: XXth ISPRS Congress, pp. 12–23.
Jiang, M., Wu, Y., Lu, C., 2018. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. arXiv preprint arXiv:1807.00652.

Kada, M., McKinley, L., 2009. 3d building reconstruction from lidar based on a cell decomposition approach. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. 38 (Part 3), W4.

Kim, H., Sohn, G., 2011. Random forests based multiple classifier system for power-line scene classification. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. 38 (5), W12.

Le Saux, B., Yokoya, N., Haensch, R., Brown, M., 2019. 2019 ieee grss data fusion contest: Large-scale semantic 3d reconstruction [technical committees]. IEEE Geosci. Remote Sens. Mag. 7 (4), 33–36.

Li, J., Chen, B.M., Lee, G.H., 2018a. So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9397–9406.

Li, X., Yao, X., Fang, Y., 2018b. Building-a-nets: Robust building extraction from high-resolution remote sensing images with adversarial networks. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. 99, 1–8.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018c. Pointcnn: Convolution on x-transformed points. In: Advances in Neural Information Processing Systems, pp. 820–830.

Lin, C.-H., Chen, J.-Y., Su, P.-L., Chen, C.-H., 2014. Eigen-feature analysis of weighted covariance matrices for lidar point cloud classification. ISPRS J. Photogramm. Remote Sens. 94, 70–79.

Lodha, S.K., Fitzpatrick, D.M., Helmbold, D.P., 2007. Aerial lidar data classification using adaboost. In: Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007). IEEE, pp. 435–442.

Maggiori, E., Tarabalka, Y., Charpiat, G., Alliez, P., 2016. Convolutional neural networks for large-scale remote-sensing image classification. IEEE Trans. Geosci. Remote Sens. 55 (2), 645–657.

Mallet, C., Bretar, F., Roux, M., Soergel, U., Heipke, C., 2011. Relevance assessment of full-waveform lidar data for urban area classification. ISPRS J. Photogramm. Remote Sens. 66 (6), S71–S84.

Mongus, D., Žalik, B., 2013. Computationally efficient method for the generation of a digital terrain model from airborne lidar data using connected operators. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. 7 (1), 340–351.

Munoz, D., Bagnell, J.A., Vandapel, N., Hebert, M., 2009. Contextual classification with functional max-margin markov networks. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, pp. 975–982.

Niemeyer, J., Rottensteiner, F., Soergel, U., 2012. Conditional random fields for lidar point cloud classification in complex urban areas. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. 3, 263–268.

Niemeyer, J., Rottensteiner, F., Soergel, U., 2013. Classification of urban lidar data using conditional random field and random forests. In: Joint Urban Remote Sensing Event 2013. IEEE, pp. 139–142.

Niemeyer, J., Rottensteiner, F., Soergel, U., 2014. Contextual classification of lidar data and building object detection in urban areas. ISPRS J. Photogramm. Remote Sens. 87, 152–165.

Niemeyer, J., Rottensteiner, F., Sörgel, U., Heipke, C., 2016. Hierarchical higher order crf for the classification of airborne lidar point clouds in urban areas. Int. Arch. Photogramm. Remote Sens. Spatial Inform. Sci.-ISPRS Arch. 41, 655–662.

Parzen, E., 1962. On estimation of a probability density function and mode. Ann. Math. Stat. 33 (3), 1065–1076.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660.

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems, pp. 5099–5108.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. Springer, pp. 234–241.

Shapovalov, R., Velizhev, E., Barinova, O., 2010. Nonassociative markov networks for 3d point cloud classification. the. In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVIII, Part 3A, Citeseer.

Shen, Y., Feng, C., Yang, Y., Tian, D., 2018. Mining point cloud local structures by kernel correlation and graph pooling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4548–4557.

Solberg, S., Brunner, A., Hanssen, K.H., Lange, H., Næsset, E., Rautiainen, M., Stenberg, P., 2009. Mapping lai in a norway spruce forest using airborne laser scanning. Remote Sens. Environ. 113 (11), 2317–2327.

Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 945–953.

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L.J., 2019. Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 6411–6420.

Wang, S., Suo, S., Ma, W.-C., Pokrovsky, A., Urtasun, R., 2018. Deep parametric continuous convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2589–2597.

Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015a. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS J. Photogramm. Remote Sens. 105, 286–304.

Weinmann, M., Schmidt, A., Mallet, C., Hinz, S., Rottensteiner, F., Jutzi, B., 2015b. Contextual classification of point cloud data by exploiting individual 3d neigbourhoods. ISPRS Ann. Photogramm. Remote Sens. Spatial Inform. Sci. II-3 W4(2), 271–278.

Wen, C., Yang, L., Li, X., Peng, L., Chi, T., 2020. Directionally constrained fully convolutional neural network for airborne lidar point cloud classification. ISPRS J. Photogramm. Remote Sens. 162, 50–62.

Wu, W., Qi, Z., Fuxin, L., 2019. Pointconv: Deep convolutional networks on 3d point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9621–9630.

Yang, B., Huang, R., Li, J., Tian, M., Dai, W., Zhong, R., 2017a. Automated reconstruction of building lods from airborne lidar point clouds using an improved morphological scale space. Remote Sens. 9 (1), 14.

Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., Huang, W., 2017b. A convolutional neural network-based 3d semantic labeling method for als point clouds. Remote Sens. 9 (9), 936.

Yang, Z., Tan, B., Pei, H., Jiang, W., 2018. Segmentation and multi-scale convolutional neural network-based classification of airborne laser scanner data. Sensors 18 (10), 3347.

Yousefhussien, M., Kelbe, D.J., Ientilucci, E.J., Salvaggio, C., 2018. A multi-scale fully convolutional network for semantic labeling of 3d point clouds. ISPRS J. Photogramm. Remote Sens. 143, 191–204.

Zhan, Y., Fu, K., Yan, M., Sun, X., Wang, H., Qiu, X., 2017. Change detection based on deep siamese convolutional network for optical aerial images. IEEE Geosci. Remote Sens. Lett. 14 (10), 1845–1849.

Zhang, J., Lin, X., Ning, X., 2013. Svm-based classification of segmented airborne lidar point clouds in urban areas. Remote Sens. 5 (8), 3749–3775.

Zhao, R., Pang, M., Wang, J., 2018. Classifying airborne lidar point clouds via deep features learned by a multi-scale convolutional neural network. Int. J. Geogr. Inf. Sci. 32 (5), 960–979.