Rapid Classification of Large Aerial LiDAR Datasets

Fauzy Othman [1,2], Phil Bartie [1], Dongdong Chen [1]

[1] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
[2] PETRONAS, Malaysia
*Correspondence: fobo2000@hw.ac.uk / fauzyomar@petronas.com

Keywords: Classification, semantic segmentation, point clouds, DALES, MPVCNN, Look Twice

Abstract

Point cloud data from aerial LiDAR scan ("ALS") are used for object detection and classification of energy industry facilities and assets. It is advantageous to be able to carry out point cloud classifications in near real time on secure hardware at the survey location and to be able to rapidly train the model on custom object classes. Such requirements create the need for efficient deep learning architectures which produce accurate predictions with low computational cost and time. This research presents a solution using Modified Point Voxel CNN ("MPVCNN") which consists of feature-level fusion between voxel and point features for local feature extraction. In doing so, this architecture circumvents indexing operations and GPU memory limitations. The MPVCNN developed in this research was trialled using dense DALES datasets. Additionally, Aerial LiDAR scan datasets typically suffer from a class imbalance for rare objects and those which are physically small or thin-shaped, relative to other object classes. This research explores how a second classification pass can be used to improve the initial classification prediction for such imbalanced object classes, by using predicted class labels as a criterion to group points which are semantically homogeneous in computing geometric features. This paper demonstrates that the MPVCNN architecture is capable of high accuracy (>0.9 F1-score and OA) classifications, with short training times (approximately 1 hour), on dense ALS datasets using standard hardware (e.g. 8GB GPU).

1. Introduction

This paper develops and evaluates an efficient deep learning methodology for 3D point cloud classification. The intended application is for aerial LiDAR based asset surveillance campaigns within the energy industry, with a primary goal to detect intrusions or anomalies around critical infrastructures such as pipelines and powerlines. There are reported instances of theft from oil pipelines across several countries (Smith, 2022), where valuable oil products were illegally diverted from the main pipeline network causing significant economic loss and creating potential hazards (Ambituuni et al., 2015). In addition, critical assets require monitoring for signs of ground movement, primarily due to landslides, subsidence, erosion, or flooding (UKOPA, 2019). Such infrastructures typically traverse large swaths of land, which makes monitoring for safety and security a challenging task for asset owners and operators. Due to the large data size, the survey data is usually post-processed upon completion of the surveillance mission, resulting in a delay before any appropriate actions can be taken. From a safety and security point of view, the delay could be more detrimental than accuracy (or resolution quality) of the captured data. Thus, techniques towards rapid 3D point cloud classification are crucial for this application, such that objects could be identified during the surveillance campaign, instead of days or weeks after the event.

With the advent of high-resolution airborne LiDAR scanners, large-scale dense public datasets are becoming more common. This study used the relatively new Dayton Annotated LiDAR Earth Scan ("DALES") dataset (Varney et al., 2020). Aerial LiDAR scan point cloud coordinate data can be supplemented with handcrafted geometric properties as additional information. These geometric features (e.g. Verticality, Roughness, Planarity, Linearity etc.) represent the shape, size or orientation of the local point cloud distribution and are derived from the eigenvalue decomposition of a group of points (Weinmann et al., 2013).

As a classification tool, adequately high prediction accuracy i.e. F1-score of above 0.9 on important classes is a reasonable performance target; to enable near real-time inferencing during field work surveillance campaigns and rapid re-training. Therefore, the main goal of this research is to achieve low training times of less than 1 hour and inferencing in under 1 second for a typical tile size of 500m x 200m (to account for the width of energy pipeline right-of-way), operating on a laptop equipped with a GPU e.g. RTX3070 8GB. In this pursuit, we introduce a Modified Point Voxel CNN ("MPVCNN") architecture to address the requirements for a rapid deep learning classification tool for aerial asset surveillance.

A common challenge on aerial point cloud classification is to obtain high accuracy on class imbalanced and relatively small objects such as 'car' and 'truck'. A case in point is the low F1-score for 'car' object class as reported by the authors of TONIC architecture (Özdemir et al., 2021); in the range of 0.666 – 0.682. Thus, the second research goal is towards development of methodology to detect changes and identify objects from aerial surveillance on infrastructures, thus it is in our interest to detect signs of human activity via proxies such as vehicle. Therefore, the secondary focus of this paper is to develop a methodology to improve classification accuracy on objects that are commonly impacted from class imbalance, specifically for 'car'. A higher F1-score figure (e.g. above 0.90) would be desirable for reliable detection for this object class.

In calculating geometric features, either a fixed radius or k-nearest neighbours are used to group neighbourhood points around a given centroid point. It is also likely that the grouped points are composed from multiple object classes (e.g. 'car', 'vegetation', 'ground') which are commonly found near to each other. In this situation, the conventional computation of geometric features has the tendency to aggregate points from a mixture of object classes. A better representation is to re-calculate the geometric features for an individual object in a

discretize manner i.e. points from 'car' are segregated from 'vegetation'. Intuitively, points from different object classes are unlikely to share similar geometric shapes and features. Conversely, points that belong to the same class but from different instances (e.g. 'car A' and 'car B'.) would tend to have similar geometric shapes and features. This research studies the use of class information as a filter to group points which are semantically homogeneous i.e. belonging to the same class for geometric features computation. The re-calculation of geometric features is performed during the second train-test pass of the chosen deep learning architecture. As a proof of concept, we use MPVCNN architecture for its low training durations and high accuracy in the development of this second processing pass technique aptly referred as "Look Twice".

The contributions of this paper are:
(i) MPVCNN architecture that balances accuracy and F1-score with low training times and computational resources, suited for rapid processing in the field.
(ii) Look Twice approach: a novel second processing pass workflow, designed to maximize F1-scores for class imbalanced objects within ALS dataset i.e. 'car' and 'powerline', achieving the highest F1-score values for the DALES dataset to date

2. Related Work

Our Modified Point Voxel CNN ("MPVCNN") is a new improvement for aerial point cloud datasets, developed from the original indoor segmentation task point-voxel deep learning architecture named PVCNN (Liu et al., 2019). From the onset, PVCNN was built with goals to achieve low training times and low computational demand. It has been shown to have superior accuracy, intersection-over-union and latency against other benchmark models for indoor scene segmentation using the S3DIS dataset (Armeni et al., 2016). The main limiting factor of 3D point-voxel architecture is the size of the voxel grids, specifically the cubic growth of the number of voxels will increase GPU memory used. The authors of PVCNN did not include the usage of other input features such as geometric features, apart from coordinate data and LiDAR derived features.

Other direct point-based architectures, primarily PointNet++ (Qi et al., 2017) or its derivatives such as Modified PointNet++ (Y. Chen et al., 2021) and GADH-Net (Li et al., 2020) are known for their longer train times, due to inherent point sampling and grouping operations. A PointNet++-like architecture involves selecting centroid points based on a farthest point sampling technique, then grouping of neighbourhood points around the centroid point that resides within a sphere search query. The subsequent step is known as feature abstraction layer, whereby multiple convolution steps are then applied on the features (coordinates or other data representation) contained within the centroid and its group of neighbourhood points. Structurally, PointNet++ is a form of U-Net architecture, pioneered by Ronneberger et al. (2015), in which the network learns by sampling centroids at various receptive scales to encode information at different physical sizes. Modified PointNet++ and GADH-Net were developed using Vaihingen 3D dataset (Rottensteiner et al., 2012) - a sparse publicly released LiDAR dataset. The main difference between the two mentioned architectures is that GADH-Net included geometric features as its input features. However, for a dataset with large point density, in order for any PointNet++ derivative architecture to gain benefit from a large amount of data, it would require more centroid points to be sampled and grouping of neighbourhood points, based on distance calculation between all sampled points and keeping a register of the point indexes, which thus increases its computational cost. The authors of Modified Pointnet++ reported an average F1-score of 0.712 for the Vaihingen 3D dataset with an Overall Accuracy of 0.832, and training time of 2 hours on a 32GB Nvidia Tesla V100. Whereas authors of GADH-Net achieved an average F1-score of 0.717, Overall Accuracy of 0.850, with training time of 7 hours on 2x12GB Nvidia Titan Xp.

TONIC (Özdemir et al., 2021) is an architecture which uses voxelization step to encode coordinate and LiDAR features (e.g. intensity). This is followed by k-nearest neighbour selection; to group neighbourhood points around a centroid point. Subsequently 5 geometric features (Linearity, Sphericity, Omnivariance, Planarity, Verticality) and 2 height features (height above ground, height change) are calculated and included as its input features. The selection criteria on geometric features were not discussed by the author, which we believe is an imperative step and included in our study as section 4.3.3. In TONIC, the input features (point coordinates, derived geometric and LiDAR sensor data) are encoded as an image map in a 2D matrix (row: number of points x column: features). The row of the matrix represented by number of points is sorted based on the coordinate values, thus providing some order to the data points. Given the data structure, 2D CNN operations can be applied, similar to image classification. TONIC is a relevant benchmark to our research, as the architecture was developed for low training times and designed to work with dense ALS datasets, using the DALES dataset for evaluation. When the TONIC model was trained and tested on DALES tiles it reported high F1-scores (> 0.9) for 'ground', 'vegetation', 'powerline' and 'building' classes, but noticeably lower for 'car' (0.666) and 'truck' (0.000), which are objects of interest for aerial surveillance. To tackle processing challenges with large ALS datasets, the TONIC authors downsampled the data to 5.6% of the original density, effectively reducing from estimated 15 million points available in the train tile to less than 1 million points for a tile with the size of 500m x 500m (equivalent to point density of 3 points / m^2), an approach that while practical, may risks the loss of valuable information.

In recent times, attention-based architectures have emerged for point cloud data such as DAPnet (L. Chen et al., 2021).The authors reported the state-of-the-art performance on Vaihingen dataset with Average F1-score of 0.823 and Overall Accuracy of 0.907. Structurally, DAPnet uses point sampling and grouping methods similar to the approach by PointNet++. Where it differs from PointNet++ is in the feature abstraction layer; extracting point level and group level features using self-attention method (instead of multiple Multi Layer Perceptrons in PointNet++ inspired architectures). Nonetheless, the authors did not report the training duration and GPU size used in the study.

Superpoint Transformer by Robert et al. (2023) was a recent development using a transformer-based architecture applied on cluster of point clouds; partitioned based on geometrically homogeneous groupings known as 'superpoints' or reference points, in contrast to farthest point sampling technique commonly used in PointNet++ type of architectures. The authors of Superpoint Transformer also included handcrafted features to define the relationship between superpoints such as the relative positions of centroid points, position of paired points in each superpoint, principal directions, ratio between the superpoints' length, volume, surface, and point count. These features were computed once during preprocessing step. In this design, superpoints are partition as hierarchical graphs ('parent' – 'children' points relation) at multiple scales. Then a transformer-based self-attention module (consists of key, query, value

vectors) is applied at each partition layer to propagate information between neighbouring superpoints. Using the DALES dataset, the authors reported mIOU of 0.796. However, no OA nor F1-scores per class objects were reported, and details of which tiles were used for training and testing were not disclosed. In terms of time, the pre-processing step (superpoint graphs, handcrafted features) took 148min (48 cores CPU) for DALES dataset but the corresponding train duration was not reported. On the S3DIS dataset, the preprocessing time was 12 minutes plus training time of around 3 hours (on A40 DPU with 512Gb RAM). As such, we could infer that this methodology would consume higher than the targeted 1 hour train time.

On the second processing pass workflow, to the best of our knowledge, no other prior study has explored the approach of introducing a second pass to take the predicted class labels as a filter in a further point cloud classification processing pipeline.

3. Methodology

3.1 Dataset

Dayton Annotated LiDAR Earth Scan ("DALES") dataset by Varney et al. (2020) is a relatively recent large-scale ALS made accessible to public, with nearly a half-billion points spanning an area of 10 km^2. This data was captured using a Riegl Q1560 LiDAR scanner over the city of Surrey, Canada, from an altitude of 1300m with four overlapping passes to increase point density. To put in context, in comparison to Vaihingen 3D dataset, DALES dataset contains 400x greater number of points and 7x more point density. This dataset is divided into 29 train and 11 test files, covering urban, suburban, rural, and commercial areas. On average, each file contains 12 million points, with tile size of 500m x 500m, with a point density of approximately 48 points/m^2.

The dataset consists of XYZ coordinates, LiDAR return intensity and was manually classified into 8 classes: 'ground' (road, grass covered surface), 'vegetation' (including tree, shrubs, hedges), 'car', 'truck', 'powerline', 'poles', 'fence' (residential and highway barriers), 'building' (residential, commercial, warehouse). Major classes in this dataset are: 'ground', 'vegetation', and 'building'. The DALES dataset covers rural areas, hence dense vegetation areas are included. Class imbalance is also prevalent in the DALES dataset as depicted in Table 1.

		Training	%	Testing	%
1	Ground	4,974,996	43.3	6,201,250	53.3
2	Vegetation	2,888,008	25.1	2,908,888	25.0
3	Car	66,146	0.6	133,464	1.2
4	Truck	63,669	0.6	11,523	0.1
5	Powerline	55,534	0.5	31,358	0.3
6	Fence	11,732	0.1	107,560	0.9
7	Pole	12,114	0.1	8,566	0.1
8	Building	3,424,274	29.7	2,237,107	19.1
	Total	**11,496,473**	**100**	**11,639,716**	**100**

Table 1 - Number of points per class within DALES train tile 5095-54440 & test tile 5135-54435 (500m x 500m)

For batch training, the tile data was divided into 25m x 25m blocks with overlap of 12.5m - to eliminate edge effects and to increase the number of data points for training. Data downsampling was implemented to avoid excessively large data to be processed during model training and substantial computing resources that would be needed. In this study, train and test data were reduced to 25% of the original data points provided in the DALES dataset. To circumvent the stochastic effects from different points being sampled in each train run, a sample size of 8192 points was used per data grid block. This figure is above the average number of points within a 25m x 25m grid block, thus consuming 100% of the available data points as samples. A lower sample size (e.g. 4096 points) would randomly include different points in each training run, leading towards varying F1-scores on particularly sensitive imbalanced object classes.

3.2 Input Features

Input data used from DALES dataset are X,Y,Z coordinates and LiDAR intensity (I). To account for terrain elevation impact, Z coordinates are converted into height-above-ground ("H") using third party software (QGIS). In which, points classed as 'ground' were used to interpolate ground elevation in raster format in QGIS. Thereafter, height-above-ground values are calculated by subtracting Z coordinates from the interpolated ground level raster. Subsequently the input features X, Y, H and I values are normalized between 0 and 1 using min-max scaling.

Just as variance indicates the degree of spread within a group of data points (in this case XYZ coordinates), covariance indicates the orientation in the distribution of the data. In the case of point cloud data, the covariance matrix is derived from the distribution of neighbouring point clouds around a reference point. As defined by Weinmann et al. (2013) from the covariance matrix and its consequent eigenvalues (λ), several geometric parameters can be derived, with example included as Equation 1. The features are Anisotropy (A_λ) - a measure of oriented or non-oriented, Eigenentropy (E_λ) - a measure of order or disorder, Linearity (L_λ), Omnivariance (O_λ) – a measure of spread in the group of points, Planarity (P_λ), Sphericity (S_λ). Roughness (R_λ) is another representation of geometric feature described by Glira et al. (2015), which corresponds to the standard deviation of the selected points from the estimated plane. The largest eigenvalue corresponds to the covariance matrix and indicates the magnitude and direction of the largest spread within a given data. In a 3-coordinate system, there are 3 corresponding eigenvalues (λ_1, λ_2, λ_3) with λ_1 being the largest and λ_3 the smallest. As a good practice for network training, the input features are normalized from 0 to 1. Thus, each of the three eigenvalues are normalized by dividing it with the sum of the eigenvalues, for instance $e_1 = \lambda_1 / (\lambda_1 + \lambda_2 + \lambda_3)$, such that the sum of the normalized eigenvalues ($e_1 + e_2 + e_3$) would be equal to 1.

$$P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1} \qquad R_\lambda = \sqrt{\lambda_3} \tag{1}$$

In addition to eigenvalues-derived geometric features, other parameters such as local point cloud density (D) and surface normal along the z-axis or verticality (V) are also evaluated as features. Local point cloud density is a measure of number of neighbouring points within a search sphere with a defined radius (in this study, we chosen r=3 metres). Eigenvalues and consequent geometric features are also dependent on the number of points selected here nearest neighbour searches of k=5 and k=30 were explored to study the impact of various search scales.

As reported by Weinmann et al. (2013), geometric features can be redundant or irrelevant to the classification task. Although one could expect a deep learning network to learn and reduce information contributed from insignificant features, but in

practice, this may not be the case. As such, feature relevance and feature selection are included in our approach. Analysis of variance ("ANOVA") was used to determine the degree of variability between different geometric features, as a ratio to the degree of variability within the feature itself. In some literatures, ANOVA is referred as F-test.

In our approach, calculation of geometric features was performed as a pre-processing step, prior to sampling of data points through the voxelization step. In this way, the geometric feature calculation has access to the complete dataset without being limited by grouping points that reside only in the same voxel cell. The data points are grouped according to a defined search radius (r=3 m) and then the first k=30 nearest neighbour points were selected.

3.3 Deep Learning Architecture

A key design in MPVCNN and PVCNN is the fusion of features using two encoding branches; a voxel-based branch which considers the effect of local neighbourhood point features, and a Multi Layer Perceptron ("MLP") for individual point features. In the voxel branch, local neighbourhood point cloud $\{p_k, f_k\}$ where $p_k = (x_k, y_k, z_k)$ is the normalized coordinates and features f_k of k^{th} input are transformed into a voxel grid system (u,v,w). These operations are performed for two voxel sizes (64^3 and 32^3) to account for different receptive scales. Subsequently, an average pool operation is applied on the voxelized features $V_{u,v,w,c}$ (e.g. coordinates, intensity, geometric features, feature maps); by averaging the sum of the features divided by the number of points in the voxel, as described in Equation 2.

$$V_{u,v,w,c} = \frac{1}{N_{u,v,w}} \sum_{k=1}^{n} \mathbb{I}[\, floor(x_k \times r) = u,\ floor(y_k \times r) = v,\ floor(z_k \times r) = w\,] \times f_{k,c} \quad (2)$$

Where r represents the voxel resolution, $\mathbb{I}[.]$ denotes the binary indicator if p_k occupies a voxel grid (u,v,w), f_k,c represents the feature c^{th} channel which correspond to point p_k, N is the number of points within the voxel grid.

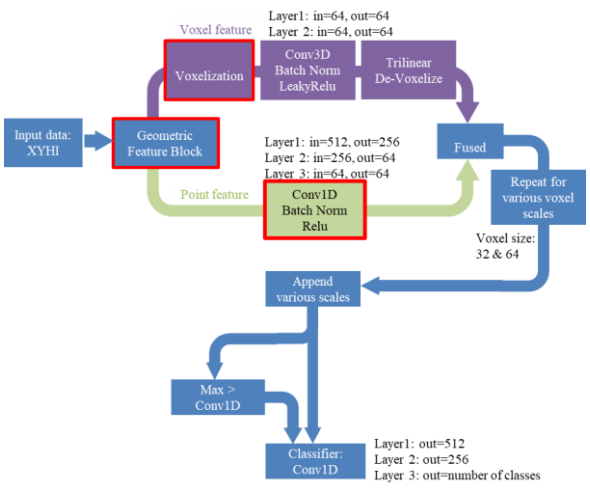


Figure 1 – MPVCNN network architecture

Upon this step, the network executes two layers of 3D Convolution operations on the voxel grid (also with batch normalization & nonlinear activation function on each layer) producing output features tensor in dimension of batch size, output channel size, number of sampled points. The 'voxel feature' branch in Figure 1 however represents the neighbourhood information in a coarse granularity. For finer granularity at individual point level, the 'point feature' branch in the same Figure 1 extracts information of the (individual) point.

In the devoxelization step, the network re-maps voxel domain features back to the point cloud cartesian grid domain. To avoid excessive loss of information if all the points in the same voxel grid were to have the same value, a trilinear interpolation method is used. More importantly, trilinear interpolation is a differentiable function, thus enabling the voxel branch to be trainable.

Referring to Figure 1 modification to the original PVCNN architecture is marked by the boxes outlined in 'red'. One of the two improvements introduced in our MPVCNN is the geometric feature block as an input feature, in addition to coordinate data. Thus, providing more information and dimensionality towards ALS type data. The other improvement introduced in MPVCNN is within the 'point feature' branch - for finer granularity at individual point level, we utilised 3 sequences of MLPs in the form of 1D Convolution operation with kernel size of 1 i.e. 1x1 filter, and setting its bias as zero. The network begins with an MLP layer with large input channel at the beginning and gradually reducing the input channel towards the third MLP layer.

Both branches, from high-resolution individual point information and low-resolution voxel-based neighbourhood information, are concatenated for more complete information for per-point classification operations. This fused feature map is passed through a max pooling operator to down-sample and select only the maximum value among the various feature maps. The max pooled feature map is appended to the fused feature map and subsequently fed into two fully connected layers in the classification step. The final output is class prediction for each point in the point cloud sampled set.

3.4 Look Twice Workflow

For the second processing pass, the geometric features were calculated from class-filtered group of points e.g. calculated from nearby points that share the same class label as the index points. For the second pass training dataset, the provided ground truth labels were used to search and group points belonging to the same class labels. At this stage, the model will be trained and learn the 'sanitized' geometric features associated with an ideal-case objects, where all the points in an object are from the same classes, i.e. 'car' verticality feature being calculated only from points belonging to the 'car' class. This is not a problem for a training dataset where class labels are provided, but would be a missing piece of information in an unclassified test dataset, of which obtaining the class label is the intended purpose.

Therefore, for the second pass test dataset, class label information needs to be generated from the result of the test dataset evaluated from a prior deep learning model. This first processing pass is executed in a typical train-test sequence of deep learning framework, without applying class-based filter, to generate class labels prediction in the test dataset. Albeit, at this stage, the predicted label in the test dataset is not 100% accurate, containing prediction errors (false negative and false positive predictions) of the first processing pass. Even so, we hypothesize that despite the erroneous predicted labels from the first pass, the grouped points are still semantically similar without inclusion from points belonging to other classes, and will also be evaluated on

geometric features derived from the same class filtered points. The overall workflow of our second processing method is depicted in Figure 2. Both models (from first pass and second pass) once trained, are ready to process a new unseen test data.

The proposed 'Look Twice' method can be summarized as follows:

1) Run the first processing pass - execution of train-test deep learning classifier, without applying a class filter. Generate predicted class labels in the test dataset.
2) Run the second processing pass (training) - for the training dataset use ground truth class labels and apply a class-based filter to group neighbouring points. The same (or different, user choice) deep learning architecture is then trained with the class filter being applied.
3) Run the second processing evaluation step in the second processing pass, the test dataset makes use of the predicted class labels from the first processing pass; as criterion in neighbourhood points grouping or sampling for the test dataset. Ultimately an evaluation run from this second processing pass generates the updated and final class label predictions.

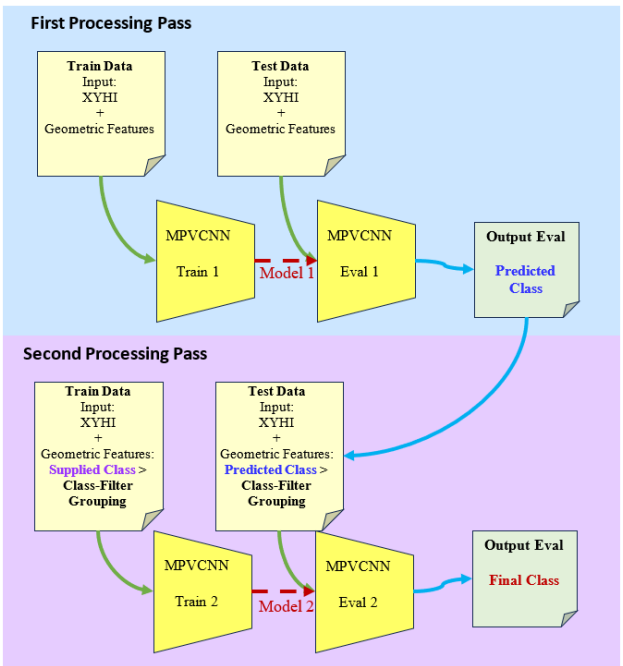


Figure 2 - Workflow first & second processing pass for train & test runs

4. Results and Discussion

4.1 Features Selection

To determine which geometric features to include as model inputs, ANOVA scores (Table 2) were used to rank features based on their sensitivity to the class labels. Subsequently, Pearson correlation was applied to eliminate geometric features which are highly correlated and therefore deemed as duplicates. , High degrees of correlation were found between Roughness, Omnivariance, Sphericity and Anisotropy, thus, the latter three features are excluded to avoid noise in the data whilst Roughness was retained as a representative input feature. Using these tools to aid feature selection, Verticality and Roughness were selected as the top two features. The next two features selected were Eigenentropy and Local Density as they both have similar ANOVA scores. Thus, the selected features used in this study are: Verticality, Roughness, Eigenentropy, Local Density and Planarity.

Feature	ANOVA score	Feature	ANOVA score
Z	213731	Intensity	45629
Verticality	166115	Eigenentropy	34719
Roughness	159234	Local Density	32448
Omnivariance	150087	X	16474
Sphericity	84142	Y	3647
Anisotropy	84140	Planarity	2220

Table 2 - ANOVA scores for feature selection

4.2 MPVCNN

The DALES dataset is relatively recent, therefore only a few deep learning networks have been developed using this dataset as reference. For benchmarking purposes, we present the IoU results as published by the authors of TONIC architecture (Özdemir et al., 2021) and the authors of DALES dataset (Varney et al., 2020) in Table 3. This table also includes the IoU results from MPVCNN trained on tile 5185_54485 (500m x 500m). For test, we selected tiles 5135_54430 + 5135_54435 (combined size of 500m x 1000m). The authors of TONIC and DALES did not explicitly indicate the tiles used for training and testing in their reporting, thus no meaningful comparison on the same train and test tile sets could be made in Table 3. Nonetheless, IoU figures are indicators on the trends among various methods. From the results, two MPVCNN models attained high OA and IoU scores for 'ground', 'vegetation', 'powerline', 'building' object classes at above 0.9, close to KPConv (Thomas et al., 2019) figures.

Model	ground	vegetation	car	truck	powerline	fence	pole	building	OA
KPConv *	0.971	0.941	0.853	0.419	0.955	0.635	0.750	0.966	0.978
PointNet++ *	0.941	0.912	0.754	0.303	0.799	0.462	0.400	0.891	0.957
TONIC (2DCNN)	0.926	0.863	0.499	0.000	0.823	0.360	0.306	0.837	0.938
MPVCNN VREDP k=5	0.959	0.930	0.681	0.130	0.279	0.502	0.015	0.950	0.970
MPVCNN VR k=30	0.954	0.915	0.647	0.085	0.917	0.430	0.031	0.924	0.967

*sourced from Varney et al. (2020)

Table 3 - IoU benchmark vs MPVCNN

In terms of F1-score benchmark (Table 4), we could draw comparison with TONIC architecture of which the authors have reported its figures. TONIC also highlighted its fast train time (0.5 hours on Vaihingen dataset, on 11GB Nvidia RTX 2080Ti GPU) which share similar ethos as our study. Again, without the tile information and data density on the experiments conducted using TONIC, we are unable to conclusively state which architecture is superior in terms of performance. Nonetheless, MPVCNN recorded higher OA, F1-scores for 'ground', 'vegetation' and 'building' on two separate test tiles. The author

of DALES did not include in the published paper the performance in F1-score for KPConv and PointNet++, hence excluded in Table 4.

Duration for model training for two tile sizes of 130,000m^2 and 250,000m^2 were 50 and 95 minutes respectively, each was set at 40 epochs. Additional preprocessing of geometric features (2 features e.g. VR) took approximately 10 and 20 minutes respectively for the mentioned tile sizes, when performed on 8GB RAM GPU RTX3070 machine.

Model	ground	vegetation	car	truck	powerline	fence	pole	building	OA
TONIC (2DCNN)	0.962	0.927	0.666	0.000	0.903	0.530	**0.468**	0.911	0.938
MPVCNN VREDP k=5	**0.982**	**0.965**	**0.810**	**0.235**	0.436	**0.669**	0.032	**0.975**	**0.970**
MPVCNN VR k=30	0.981	0.959	0.788	0.167	**0.963**	0.606	0.059	0.962	0.967

Table 4 - F1-score benchmark vs MPVCNN

4.3 Look Twice

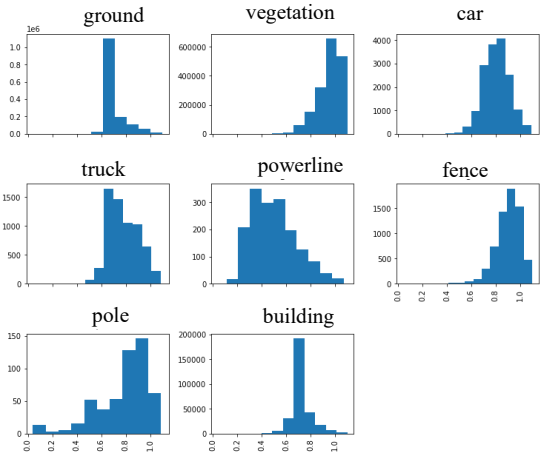
The results of the model train-test with a second processing pass is summarized in Table 5. In the same table, we included the results from (first pass) MPVCNN model VR k=30 as a baseline comparison with the test accuracies and F1-scores from second pass models. On class imbalanced objects such as 'car', model VR k=30 has produced a range of gains in F1-scores by 8% and 12 % when tested on two different tiles, corresponding to actual F1-score of 0.852 and 0.814 respectively. This is the highest figure within the benchmark data reported for DALES dataset.

MPVCNN Model	ground	vegetation	car	truck	powerline	fence	pole	building	OA
VR k=30 1st pass	0.981	0.959	0.788	0.167	0.963	0.606	0.059	0.962	0.967
VR k=30 2nd pass	0.983	0.967	**0.852**	0.227	0.900	0.620	0.071	0.975	0.973
VREDP k=30 2nd pass	0.983	0.967	**0.814**	0.280	0.988	0.554	0.096	0.975	0.973

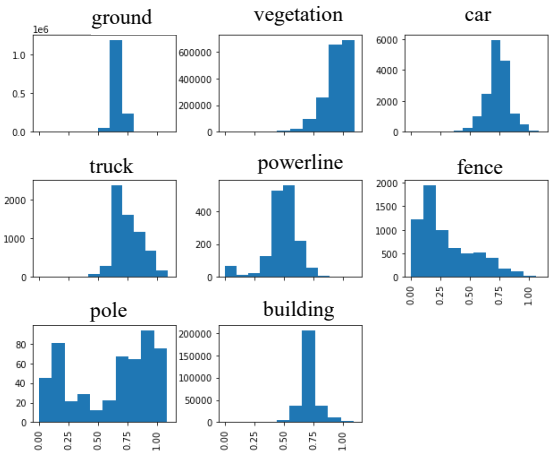
Table 5 - F1-score and OA between 1st and 2nd pass models

Referring to Figure 3, with class filter grouping, the Eigenentropy for 'powerline' exhibited distinctly reduced overlap with 'vegetation'. This could be the probable explanation on the further gain observed in F1-score from the second pass model VREDP k=30 in Table 5 on 'powerline', above the first pass value.

To aide visualizing the effects of class filter grouping, we include the verticality map of the train tile without class-filter i.e. first pass (Figure 4(b)) and with class-filtered grouping i.e. second pass (Figure 4(c)). With class-filter grouping, 'car' objects have noticeably narrow band of verticality values, indicated by the colour range, unlike the wide band observed without class-filtered. As reference, ground truth class label for the train tile is included as Figure 4(a). On the test tiles prediction results; Figure 5(b) depicts without class-filter and Figure 5(c) depicts with class-filter being applied. In Figure 5, specifically on the four purple-coloured arrows indicating 'car' objects, the prediction from the first pass model contains mixture of two class labels: 'car' and 'vegetation' (Figure 5(b)). Upon the second pass, with class-filtered geometric feature, the updated model has corrected some of the previously erroneous 'vegetation' points to 'car' (Figure 5(c)), which explains the increase in F1-score in Table 5 for the second pass models.



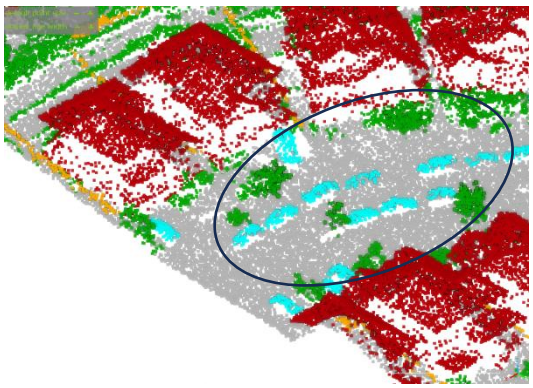
(a) without class filter



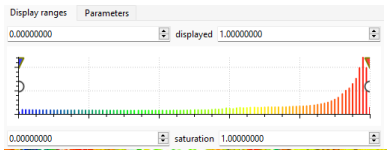
(b) with class filter

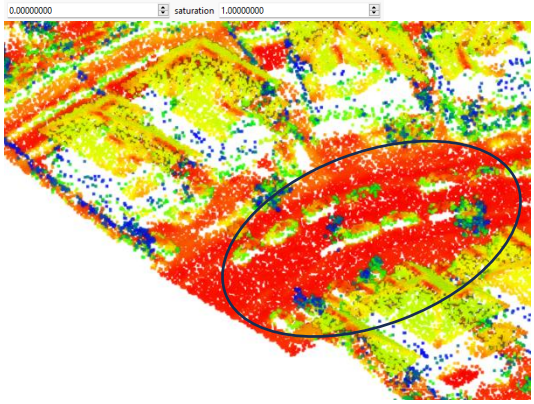
Figure 3 - Geometric features 'Eigenentropy' distribution plot per class for DALES train tile 5185_54485 (k=30)

Upsides from this model as a second processing pass were also observed on the Overall Accuracy ("OA") and slight improvement in F1-scores for large objects & major classes ('ground', 'vegetation', 'building'). As anticipated, the impact of this methodology would be pronounced on areas with presence of two or more classes and imbalanced classes. On large objects, there may not be substantial information gain from the second pass method, further away from the object boundary. Also, corrections made at the boundary edges are relatively small in comparison to the substantially large point counts associated with big objects such as 'ground' and 'vegetation' to affect F1-scores.

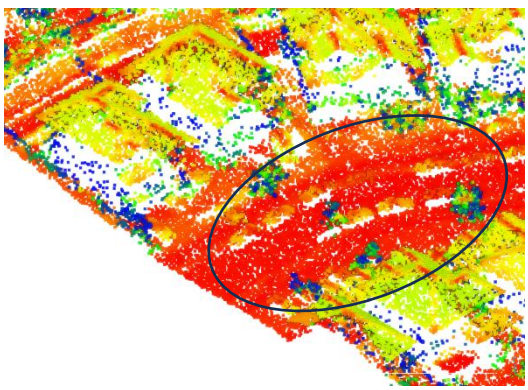


(a) ground truth class labels



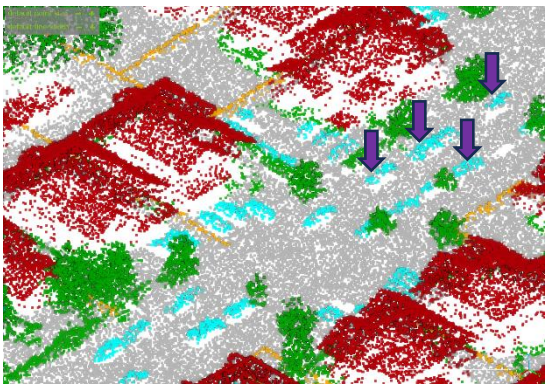


(b) verticality map without class-filter

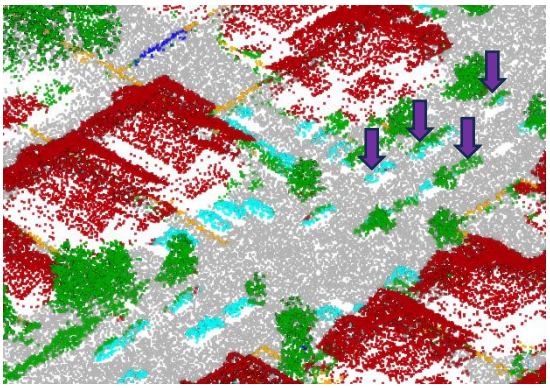


(c) verticality map with class-filter

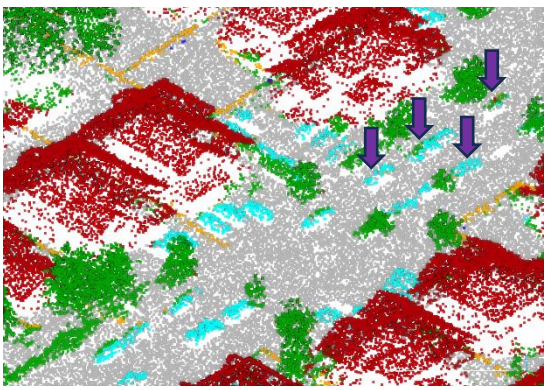
Figure 4 - Train tile 5185_54445



(a) ground truth class label



(b) without class-filter predicted class labels



(c) with class-filter predicted class label

Figure 5 – Test tile 5135_54430 + 5135_54435

5. Conclusions

We have demonstrated MPVCNN architecture as an efficient end-to-end deep learning architecture with high accuracy (>0.9 F1-score and OA), low demand in GPU memory (e.g. 8GB) and with low train times (approximately 1 hour) on dense ALS datasets. In practical use, these qualities are beneficial when processing new datasets from different geographies or adding new object classes, which may require persistent updated training. In terms of overall accuracy and F1-score performance in Table 4, our MPVCNN model is potentially superior to other benchmark such as the TONIC framework, but without the information of the benchmark train-test tiles used by other literatures, no definitive performance comparison can be drawn.

The class-filter grouping technique in the second train-test deep learning processing pass using MPVCNN is an effective tool towards extracting higher F1-scores on low frequency data points and small physical size objects such as 'car' and on thin objects such as 'powerline'. For both classes, 'car' and 'powerline', we recorded the highest F1-scores for DALES dataset at 0.852 and 0.988 respectively, with F1-score increment in the range of 8% to 12% (Table 5). Using class-filtered grouping, geometric features were determined based on a single object class, with more distinctive profiles than geometric features computed from an aerial dataset without class-filter grouping. Which also supports the benefit of this methodology in correcting prediction via learning from an ideal definition of the geometric features, especially on the less frequent classes.

In addition to the increased F1-score on two targeted class imbalanced objects ('car', 'powerline'), this method was also observed to produce minor positive gain in F1-scores for bulk objects with higher representation of data points such as 'ground', 'vegetation' and 'building'. Understandably, this method is not intended nor anticipated to provide substantial improvement on large objects and major classes, as the corrective predictions are expected to be made at boundary areas with the presence of multiple object classes.

On other class imbalanced objects such as 'truck' and 'pole', there are lesser gains from this method. We believe this could be attributed to overlapping geometric features between 'truck' and 'building', 'pole' and 'vegetation'. Moreover, the instances and number of points for 'truck' and 'pole' are relatively low, making training for such objects remain a challenge. As such, other features or descriptors are needed beyond what is included in this study to adequately differentiate such objects.

Acknowledgements

The authors acknowledge the contribution by PETRONAS for sponsoring this research, as part of a PhD project.

References

Ambituuni, A., Hopkins, P., Amezaga, J., Werner, D., & Wood, J. (2015). Risk assessment of a petroleum product pipeline in Nigeria: the realities of managig problems of theft/sabotage. *WIT Transactions of The Built Environment, 151*, 49-60.

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., & Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces. Proceedings of the IEEE conference on computer vision and pattern recognition,

Chen, L., Chen, W., Xu, Z., Huang, H., Wang, S., Zhu, Q., & Li, H. (2021). DAPnet: A double self-attention convolutional network for point cloud semantic labeling. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14*, 9680-9691.

Chen, Y., Liu, G., Xu, Y., Pan, P., & Xing, Y. (2021). PointNet++ network architecture with individual point level and global features on centroid for ALS point cloud classification. *Remote Sensing, 13*(3), 472.

Li, W., Wang, F.-D., & Xia, G.-S. (2020). A geometry-attentional network for ALS point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing, 164*, 26-40.

Liu, Z., Tang, H., Lin, Y., & Han, S. (2019). Point-voxel cnn for efficient 3d deep learning. *Advances in neural information processing systems, 32*.

Özdemir, E., Remondino, F., & Golkar, A. (2021). An efficient and general framework for aerial point cloud classification in urban scenarios. *Remote Sensing, 13*(10), 1985.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems, 30*.

Robert, D., Raguet, H., & Landrieu, L. (2023). Efficient 3D semantic segmentation with superpoint transformer. Proceedings of the IEEE/CVF International Conference on Computer Vision,

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18,

Rottensteiner, F., Sohn, G., Jung, J., Gerke, M., Baillard, C., Benitez, S., & Breitkopf, U. (2012). The ISPRS benchmark on urban object classification and 3D building reconstruction. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences; I-3, 1*(1), 293-298.

Smith, H. (2022). Progress and challenges in pipeline theft detection. *Pipeline Technology Conference 2022, Berlin*

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., & Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. Proceedings of the IEEE/CVF international conference on computer vision,

UKOPA. (2019). Good Practice Guide - Managing pipelines subject to ground movement [Industry Guideline]. (UKOPA/GP/020 Edition 1).

Varney, N., Asari, V. K., & Graehling, Q. (2020). DALES: A large-scale aerial LiDAR data set for semantic segmentation. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops,

Weinmann, M., Jutzi, B., & Mallet, C. (2013). Feature relevance assessment for the semantic interpretation of 3D point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2*, 313-318.