Vehicle Classification from Profile Measures

Marco Patanè and Andrea Fusiello
DPIA - University of Udine
Viale delle Scienze 208, 33100 Udine, IT

***Abstract*—This paper proposes two novel convolutional neural networks for 3D object classification, tailored to process point clouds that are composed of planar slices (profiles). In particular, the application that we are targeting is the classification of vehicles by scanning them along planes perpendicular to the driving direction, within the context of Electronic Toll Collection. Depending on sensors configurations, the distance between slices can be measured or not, thus resulting in two types of point clouds, namely metric and non-metric. In the latter case, two coordinates are indeed metric but the third one is merely a temporal index. Our networks, named SliceNets, extract metric information from the spatial coordinates and neighborhood information from the third one (either metric or temporal), thus being able to handle both types of point clouds. Experiments on two datasets collected in the field show the effectiveness of our networks in comparison with state-of-the-art ones.**

I. Introduction

In this paper we address a specific application of 3D object recognition, namely the classification of vehicles that drive under a portal equipped with laser scanner(s), with the goal of implementing an Electronic Toll Collection (ETC) system.

If the portal is equipped with a single laser, a sequence of profiles (or slices) is returned, but the distance between them is unknown, because it depends on the vehicle speed. If the driving direction is identified with the y-axis, the y coordinate is only an index representing the temporal relationship between slices, with no metric meaning, as opposed to x and z. Only if the vehicle speed can be measured, e.g. by placing an additional laser on the ground in the proximity of the portal, the y coordinate is metric.

In summary, the input data is a point cloud composed by slices that lie on planes parallel to the x-z planes. The situation is similar to aerial laser scanning (slices are called *swaths* in this context), where the swaths provide 2D coordinates, and the relationship between them is given by IMU and GNSS mounted on the aircraft.

Until recent years, state-of-the-art for 3D classification problems have followed a traditional pipeline, based on the extraction and aggregation of hand-engineered features. Usually, these features are sophisticated shape descriptors, such as Fisher Vectors [1] or Hough Transforms and 3D SURF [2], which are then consumed by standard SVMs for solving the classification task.

Nowadays feature-based approaches have been largely superseded by Deep Learning [3], where both features and classifier are learned from the data end-to-end. In particular, in image classification, Convolutional Neural Networks (CNNs) are the state-of-the-art [4], [5], [6]. While a lot of effort have been done in applying deep learning techniques to 2D data, deep learning on 3D data is less explored.

Depending on the type of input which is considered, three different strategies are adopted for solving 3D object classification. The first input [7], [8], [9] consist of multiple 2D views of the 3D object, which are exploited using classical two-dimensional CNNs. The second [10], [11], [12] entails the voxelization of the object, which is then fed to volumetric CNNs, that generalize the classical CNNs to three dimensions. The third input [13], [14], [15] is point clouds, which due to its structure cannot be analyzed by standard convolutions. Indeed, Qi et al. proposed two NNs: PointNet [13], a CNN where each point is independently processed by *point-wise* convolutions and then all the features are summarized through a max pooling, and PointNet++ [14], an extension on PointNet, where PointNet is applied recursively in an hierarchical fashion to nested partitioning of the input set. Another application of point-wise convolutions is Depthwise Separable Convolutions[16], [17], which are mainly employed for compression purposes. Klokov et al. [15] proposed instead a network based on kd-tree in order to process point clouds.

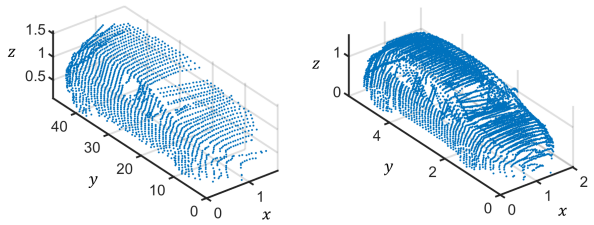


Fig. 1: Sample point clouds obtained in the two configurations analyzed. On the left, the single (lateral) laser configuration. On the right, the three lasers configuration.

In this paper we describe a network architectures tailored to the specific application we are targeting. We analyze two possible laser configurations, as shown in Fig. 1. The first is set up by one lateral laser, while the second is set up by two lateral lasers and a frontal one. We call these configurations *non-metric* and *metric* respectively. Let us remark again that only in the metric scenario the distances between slices are measured, while in the non-metric there is no way to recover them since both the vehicle length and velocity are unknown.

While for the metric case there are solutions in the literature, notably PointNet and VoxNet, the non-metric case calls for a new CNN.

In order to handle both type of point clouds we develop a novel family of CNNs, dubbed *SliceNets*, based on PointNet and VoxNet. These networks are able to extract metric information from x and z coordinates through a PointNet-like module and neighborhood information from y indices employing standard convolutions (as VoxNet).

We did not consider PointNet++ [14] because its complexity compared to PointNet is too high ($7\times$ slower in a forward pass, as Qi et al. [14] showed.)

The paper is organized as follows. In Section II VoxNet and PointNet are reviewed. Section III describes our CNNs, namely SliceNet and SliceNet-GAP. In Section IV we show the results on both non-metric and metric point clouds of our networks compared against VoxNet and PointNet. Finally, the conclusions are drawn in Section V.

II. Background

Before presenting SliceNet, we review the two CNNs from which it inherits some features, namely VoxNet and PointNet. Furthermore in Section IV, we use VoxNet and PointNet as references to test the performance of SliceNet.

A. VoxNet

The natural way to exploit CNNs with 3-dimensional (voxel) data is to process them with 3-dimensional convolutional layers. The network which exploits this strategy is VoxNet, originally described by [11]. This approach is also suitable for classifying point clouds, provided they are voxelized.

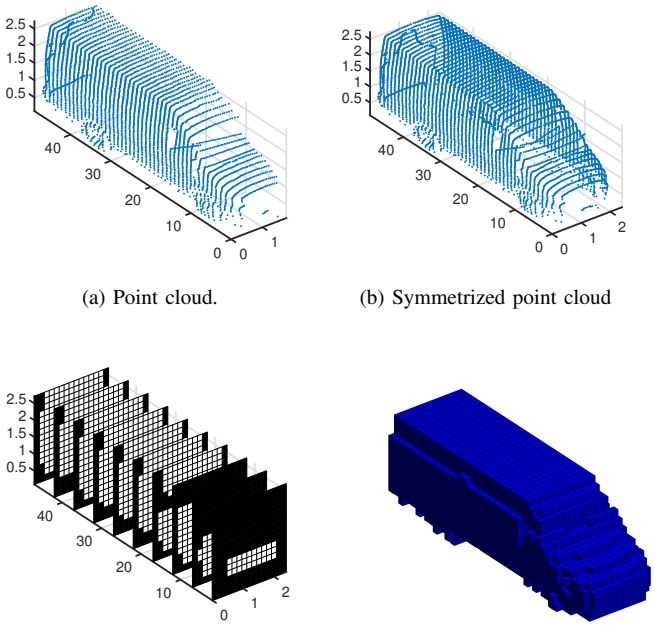
Indeed each point cloud is converted to a discretized volume, which in our application has size $16\times48\times16$. We choose such a low resolution in order to maintain a computation cost similar to CNNs based on 2D images. Furthermore, we privilege the y-axis in terms of resolution simply because the main dimension of vehicles and in particular of semi-trucks and buses is their length.

The voxelization proceeds as follows: first of all (if needed) the point cloud is symmetrized, then each xz slice is transformed to a 16×16 binary image. Finally, 48 slices are generated by interpolation from the original ones. The above procedure is depicted in Fig. 2.

Symmetrization is performed only in the case of single lateral laser. Since in this case the width of the vehicle cannot be established from the point cloud, we fixed it a-priori as a value in the middle between cars and trucks.

The slice binarization is performed defining a 16×16 grid from the xz limits extracted by the point cloud itself, and then associating 1 if the pixel is inside the polygon defined by the points in that slice, 0 otherwise. Note that during the binarization we lose the ratio between x and z dimensions, thus wasting some metric information. However CNNs can still learn rich features from the shape of vehicles, in much the same way it does with images, that are always non-metric due to perspective projection.

The last step, the interpolation along y-axis, is needed since each point cloud has a different number of slices depending on its length and its velocity during the acquisition. Once point clouds are transformed to voxels, we can feed them to VoxNet to obtain the vehicle classification.



(a) Point cloud. (b) Symmetrized point cloud

(c) 16×16 slices. (d) Voxelization.

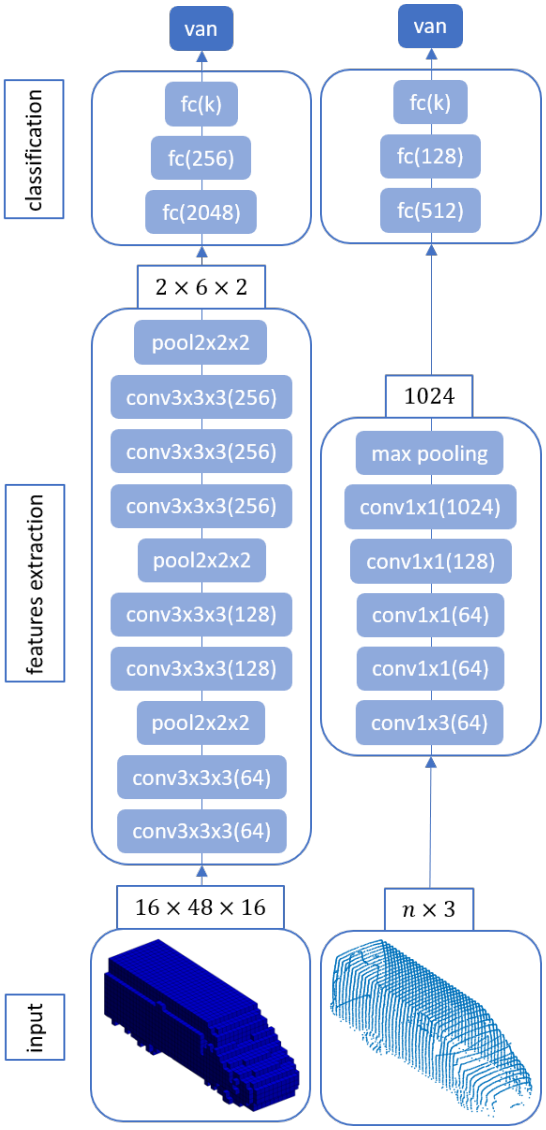
Fig. 2: Voxelization. The point cloud (2a) is symmetrized (2b), and then each xz slice is binarized obtaining a 16×16 image (2c). Finally, the voxelization is computed via interpolation (2d).

Our implementation of VoxNet is depicted in Fig. 3a. The input of size $16 \times 48 \times 16$ is fed into the network, which processes it with a sequence of convolutional and pooling layers in order to perform feature extraction. Then, the extracted features are fed to a set of final fully-connected layers, which solve the classification task by computing the final output of the network. Batch normalization [18] and dropout [19] are employed for controlling over-fitting.

B. PointNet

Solving vehicle classification using VoxNet comes with two drawbacks, both due to the process of voxelization. First of all, the complexity of the network is particularly high, and this forces to set a low resolution for the voxels at the cost of losing fine details of point clouds. The second drawback is that, in order to exploit the whole input volume for the representation of each point cloud, the coordinates are stretched, thereby losing all metric information, namely width, height and length, when available.

PointNet [13] is a specialized architecture that handles both problems. This is possible since the input of this network is given by the point cloud itself without any intermediate transformation. This network is designed for consuming sets of unordered 3D points with variable cardinality. More precisely,



(a) VoxNet. (b) PointNet.

Fig. 3: On the left VoxNet (3a), which is fed with $16 \times 48 \times 16$ voxels. The voxel is analyzed by convolution with filter size 3 and pooling with filter and stride sizes equal to 2. Then the classification is performed by three fully connected layers using the features extracted from the voxel. On the right PointNet (3b), which is fed with $n \times 3$ point clouds. Each point is independently analyzed by the network via point-wise convolutions. Then the features extracted from points are summarized in a 1024-dimensional vector via max pooling. The classification is performed by three fully connected layers using this feature vector. In both networks batch normalization and ReLu activation are used for all layers except the last one. Furthermore, dropout is used for all fully connected layers except the last one. "conv", "pool" and "fc" stand for convolutional, pooling and fully connected layers respectively. Numbers in bracket are layer sizes and the other numbers represent filter sizes.

given an $n \times 3$ array, which represents a point cloud, each point (1×3) is independently processed by the network and then a symmetric function summarizes the information over all the n points.

Each point is processed first by 1×3 convolutions and then by 1×1 convolutions. The latter are also called *point-wise convolutions*, due to the fact that the point are independently analyzed by the network. Then, max pooling with mask $n \times 1$ is performed, and finally the resulting array of features is processed by fully connected layers.

The structure of PointNet is simple, as shown in Fig. 3b, but effective, as demonstrated by the experiments conducted in [13]. Note that the complexity of PointNet is intrinsically low, since the point cloud representation is more efficient than the corresponding voxelization, since the latter wastes voxels just to represent empty space. For instance, a $16 \times 48 \times 16$ voxelized volume occupies twice the space of a 2048×3 point cloud, nevertheless it is a fairly coarse representation.

PointNet solves our problem in the metric case, but it is not able to directly solve the non-metric one, for the third coordinate is not commensurate with the other two. Indeed, PointNet processes all 3 coordinates together implicitly assuming that they have the same nature, but this assumption is not valid in the non-metric scenario. Keeping this in mind, we tried anyway to use PointNet with y coordinates normalized in $[0, 1]$, surprisingly obtaining acceptable classification results (see Section IV).

III. Proposed Solution

As we discussed in the previous section, PointNet is an effective network for processing point clouds, for it achieves a low complexity without loss of information from input discretization. However this network was thought to consume metric point clouds only. In order to process non-metric point clouds without the necessity of voxelization and its resulting disadvantages, we developed *SliceNets*, a family of networks built on both PointNet and VoxNet.

The basic idea behind our novel architectures is to exploit the mixed nature of our vehicle point clouds. Indeed they are unordered set of points with respect to x and z coordinates, which can be processed by a PointNet-like network, and a sequence of slices with respect to y axis, that can be fruitfully processed by standard (*i.e.* not point-wise) convolutional layers.

In the following we describe two types of SliceNets, namely *SliceNet* and *SliceNet-GAP*. They share the same structure, except for a layer (the GAP layer), which allow SliceNet-GAP to be agnostic about the dimension of the input.

A. SliceNet

Following the intuition described above, the $n \times 3$ point cloud can be decomposed in s sets of two-dimensional points of cardinality n_i, where s is the number of slices and n_i is the number of points in slice $i \in \{1, \dots, s\}$. Thus, if we fix the number of points per slice to $m = \max_i n_i$, we can define a tensor of dimension $s \times m \times 2$ as input for our network.

The first part of SliceNet operates on $\{1 \times m \times 2\}_{i=1}^{s}$ sets of points, where each point is processed by point-wise convolutions ($1 \times 1 \times 2$ first, and $1 \times 1 \times 1$ convolutions after), and then max pooling with mask $1 \times m \times 1$ is applied to each set. The second part of SliceNet operates on the resulting $s \times 1 \times 1$ slices, processing them via $3 \times 1 \times 1$ convolutions and $2 \times 1 \times 1$ pooling with stride 2. Finally, as usual, the feature array is processed by fully connected layers in order to obtain the classification. The complete network architecture is summarized in Fig. 4a.

In practice, to obtain the desired input tensor each set of n_i points is augmented to $m = \max_i n_i$ simply adding points from the i-th set, since the max pooling operation is insensitive to repetition of features extracted from the same points. Furthermore, all the point clouds must have the same number of slices in order to be processed by the classification block. Thus we generated s slices via interpolation over the original ones.

B. SliceNet-GAP

The main limitation of SliceNet is given by the fact that s (number of slices) must be the same for the point clouds of all vehicles, which can be solved by interpolation at first instance. If instead one wants to avoid this operation, a variation of SliceNet, named *SliceNet-GAP*, can be employed, which is able to process point clouds with variable number of slices s.

SliceNet-GAP employs global average pooling (GAP), which was proposed by Lin et al. [20] as a regularization technique. In our case GAP is used to remove the dependencies by s summing out the spatial information of slices. More precisely the GAP layer is applied before the classification block since the fully connected layer need a fixed input dimension. Apart from this extra layer, SliceNet and SliceNet-GAP architectures are identical. The complete network architecture is summarized in Fig. 4b.

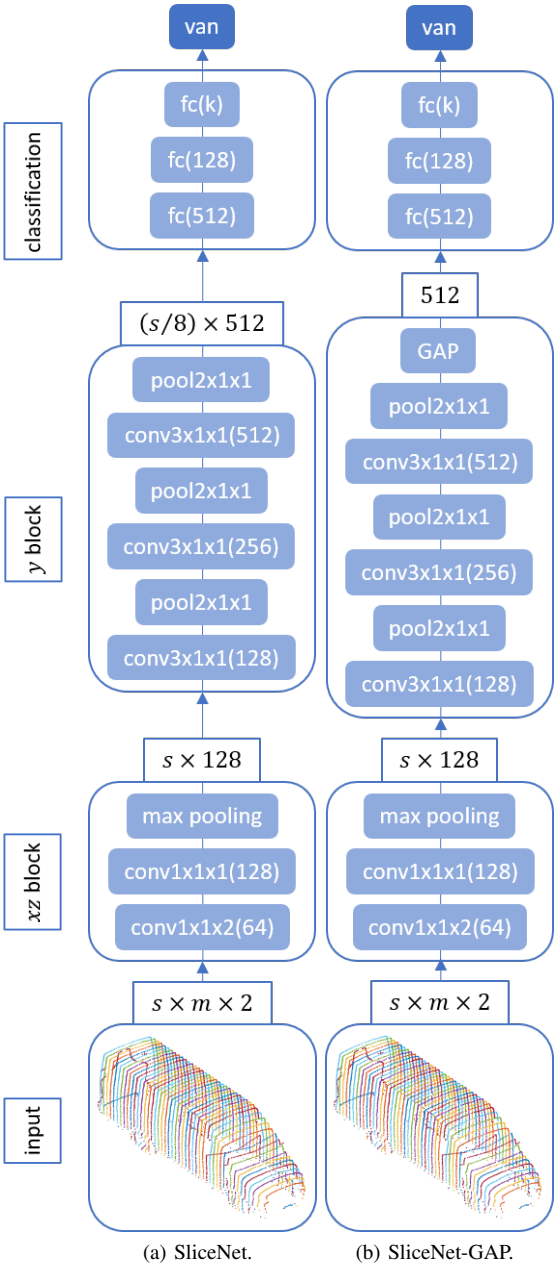
IV. Experiments

In this Section we present our classification results on both non-metric and metric datasets of vehicle point clouds. We discuss the results of our networks, *i.e.* SliceNet and SliceNet-GAP, along with comparisons with PointNet and VoxNet. Furthermore, we also provide an analysis on time and space complexity of the networks. For all the experiments we fixed $n = 2048$ as number of points for PointNet, while $s = 50$ and $m = 60$ as number of slices and per slice number of points respectively for SliceNets.

A. Datasets

We evaluate the networks through two datasets of vehicle point clouds. The main difference between the datasets is in their structural configuration: one of them is a non-metric dataset, the other is a metric one. Furthermore, they differs in the number of samples and classes.

In particular, the non-metric dataset is made of about 6900 samples and 18 classes, namely *pedestrian*, *motorbike*, *car*, *van*, *van open*, *bus*, *coach*, *truck closed*, *truck open*, *truck*



(a) SliceNet. (b) SliceNet-GAP.

Fig. 4: SliceNets. Both networks are fed with point clouds arranged as $s \times m \times 2$ tensors. Each slice of two-dimensional points is processed via a PointNet module, and then convolutions are applied over slices. The output is obtained applying three fully connected layers. The main difference between (4a) and (4b) is given by the GAP layer, which allow SliceNet-GAP to handle point clouds with different number of slices s. In both networks batch normalization and ReLu activation are used for all layers except the last one. Furthermore, dropout is used for all fully connected layers except the last one. "conv", "pool" and "fc" stand for convolutional, pooling and fully connected layers respectively. Numbers in bracket are layer sizes and the other numbers represent filter sizes.

tanker, *truck tip*, *truck cement*, *truck road tractor*, *semi-truck closed*, *semi-truck open*, *semi-truck tanker*, *semi-truck tip*, and *semi-truck car*. While the metric dataset is composed by about 2500 samples and 8 classes, namely *car*, *car-van*, *van*, *truck*, *truck road tractor*, *semi-truck closed*, *semi-truck open*, and *semi-truck tanker*. Some samples of non-metric dataset and metric dataset are shown in Fig. 9 and Fig. 8 respectively.

The non-metric dataset is more various since the samples have been collected on an highway, while the metric samples have been acquired on a harbor entrance, and for instance, almost no buses come through it. Furthermore, in order to augment the number of classes we divided the larger classes, namely *van*, *truck* and *semi-truck*, in sub-classes which differ in shape enough to distinguish them.

B. Evaluation

Since the datasets are quite small, in particular the metric one, we used *k-fold cross-validation* to evaluate the performances of our networks.. They are randomly partitioned in k equal sized subsets (we chose $k = 6$ in our experiments) and k networks are trained independently choosing each time a partition as test set and its complementary as training set. Then the k results are averaged to produce a single figure. Results are summarized in Tab. I and II.

TABLE I: Non-metric dataset results. Classification accuracy of the four networks obtained by k-fold cross-validation. "average" and "total" stand respectively for average per class accuracy and overall accuracy.

network	accuracy	
	average	total
VoxNet	91.2(±2.5)	96.6(±0.5)
PointNet	91.4(±1.4)	96.7(±0.4)
SliceNet	**93.2**(±1.8)	**97.2**(±0.3)
SliceNet-GAP	92.8(±1.6)	97.1(±0.5)

TABLE II: Metric dataset results. Classification accuracy of the four networks obtained by k-fold cross-validation. "average" and "total" stand respectively for average per class accuracy and overall accuracy.

network	accuracy	
	average	total
VoxNet	97.9(±0.6)	99.3(±0.2)
PointNet	98.2(±0.8)	**99.4**(±0.2)
SliceNet	**98.5**(±1.1)	**99.4**(±0.3)
SliceNet-GAP	98.4(±0.9)	99.2(±0.4)

As expected SliceNets are the best performing networks in the non-metric dataset (Tab. I), indeed they are designed specifically for this type of data. In particular, per class average accuracy of SliceNet is 1.8% higher than PointNet and 2.0% higher than VoxNet. Also SliceNets total accuracies are higher compared to the other CNNs. Furthermore, Slicenet works better with classes with less samples, obtaining that this network is able to learn richer features for all the classes, even the small ones.

In the metric dataset (Tab. II) SliceNet achieves the best accuracy together with PointNet, thus validating the effectiveness of our networks also for metric point clouds. Our hypothesis is that SliceNets compensate for the ignorance of the metric value of the y coordinate with the information brought by rich shape features obtained via convolution, which, on the contrary, is not available to PointNet.

SliceNet-GAP and VoxNet obtained high (total) accuracies, however their average accuracies per class are marginally worse compared to SliceNet.

Both tables show that SliceNet-GAP achieves slightly worse accuracies compared to SliceNet: the GAP layer trades some accuracy for the capability of processing point clouds with variable s (the number of slices).

In the following experiments we fixed a splitting of the training-test with proportion 80%-20%. Fig. 5 and Fig. 7 show the confusion matrices obtained by SliceNet on non-metric and metric datasets respectively. In particular, Fig. 5 shows that *van/truck* (*closed*) and *bus/coach* sometimes are misclassified, presumably due to fact that they have similar shapes and dimensions. Indeed, in Fig. 6, we can see how, even for a human, it is difficult to distinguish *van/truck* (*closed*) and *bus/coach*.

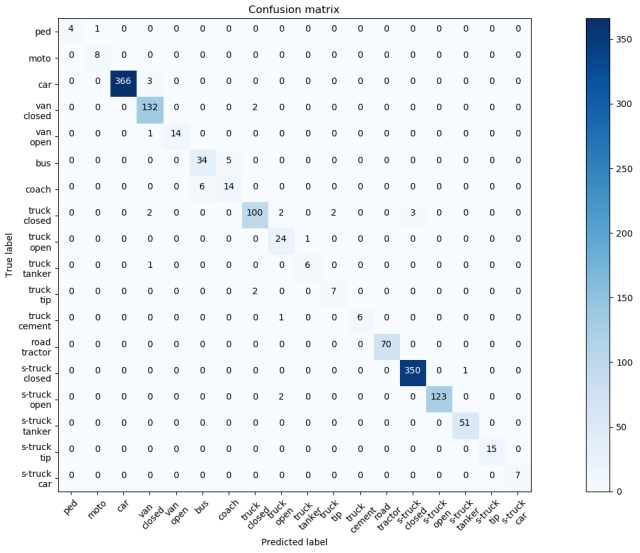
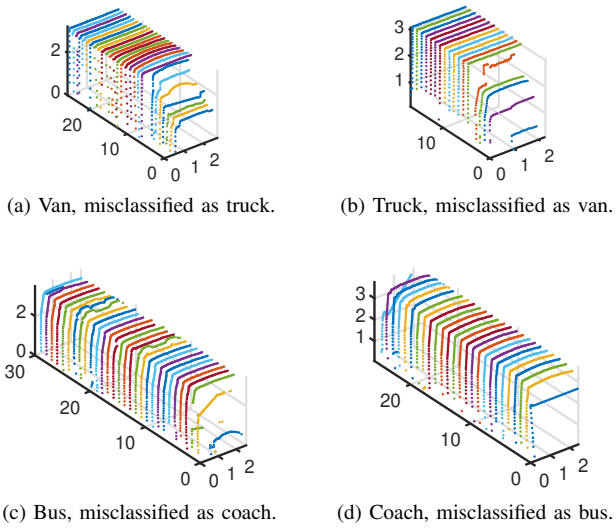


Fig. 5: Confusion matrix on non-metric dataset.

Fig. 7 shows instead almost perfect results, probably thanks to classes that are more separable and conversely have less inter-class variance.

C. Complexity

Here we compare SliceNets against PointNet and VoxNet in terms of time and space complexity. The number of parameters represents the space complexity of the neural networks, while the FLOPs (*i.e.* floating-point operations) per sample the time



(a) Van, misclassified as truck.

(b) Truck, misclassified as van.

(c) Bus, misclassified as coach.

(d) Coach, misclassified as bus.

Fig. 6: Some non-metric samples misclassified by SliceNet.

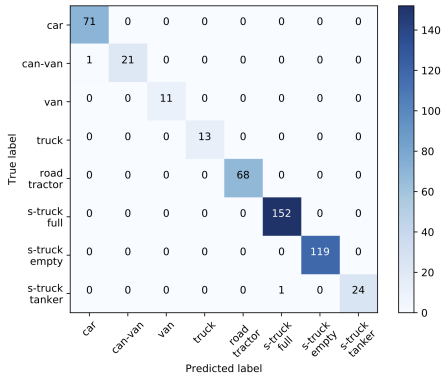


Fig. 7: Confusion matrix on metric dataset.

complexity, where we consider multiply-adds operations only. Results are summarized in Tab. III.

TABLE III: Time and space complexity. Number of parameters and floating-point operations (*i.e.* multiply-adds operations) of one sample for each network. The "M" stands for million.

network	parameters	FLOPs/sample
VoxNet	22.0M	3249M
PointNet	0.7M	287M
SliceNet	1.7M	34M
SliceNet-GAP	0.9M	33M

As expected, VoxNet is the most complex network in terms of both time and space. In particular Voxnet is $11\times$ less efficient having $31\times$ more parameters compared to PointNet. While SliceNets have more or less the same number of parameters of PointNet, being more efficient than the competitors in term of FLOPs/sample, namely $8\times$ and $95\times$ more efficient than PointNet and VoxNet respectively.

These results confirm that SliceNets are very efficient, as well as accurate, showing their suitability for real-time applications.

D. Training Details

For all four networks, the weights are initialized as described by Glorot et al. [21]. Regularization is provided by dropout [19] and batch normalization [18]. In particular, dropout is performed with rate 0.7 on all fully connected layers except the last one. Conversely, batch normalization is performed on all layers except the last one with initial decay rate 0.5, then exponentially increased to 0.99 during training. We choose Adam optimizer [22] with momentum 0.9 and initial learning rate 0.0005, then exponentially decreased to 0.00001. Fixing the batch size to 128, training takes 1-2 hours to converge with TensorFlow [23] on a Tesla K40c GPU.

V. Conclusion

This paper described SliceNets, a family of two convolutional neural networks for 3D point-cloud classification. They are specifically tailored to process point clouds obtained by scanning vehicles along planes perpendicular to the driving direction. The resulting point clouds can be metric or non-metric, depending on sensors configurations; in the latter case the slice coordinate is merely a temporal index.

SliceNets are able to extract metric information from the spatial coordinates and neighborhood information from the third one (either metric or temporal), thus being able to handle both types of point clouds.

Experiments on metric and non-metric datasets showed that SliceNets compares favourably with the competitors on both accuracy and complexity. In particular, as expected, SliceNet performs better than others on non-metric data, since it has designed for that task, but it can also deal with metric configuration with performances on par with the state-of-the-art.

At last, SliceNet was ported on an embedded system for real-time ETC.

Acknowledgements

This work has been carried out in 2017/18 within the SPATA project, funded by Friuli-Venezia Giulia region, POR FESR 2014-2020 program. Data have been provided by Comark srl as a partner of the project. Authors are grateful to A. Abramo for kindly granting access to the Tesla GPU.

References

[1] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.

[2] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3d surf for robust three dimensional classification," *Computer vision–ECCV 2010*, pp. 589–602, 2010.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

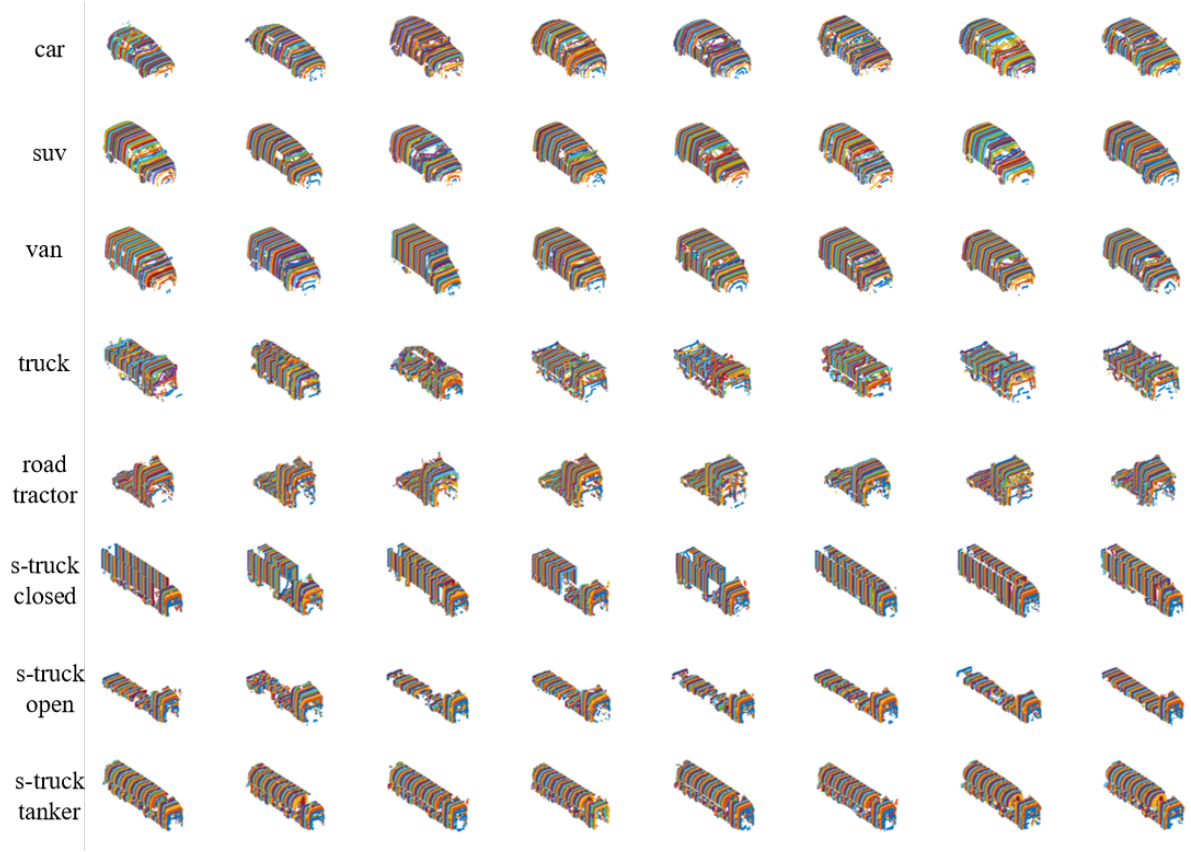


Fig. 8: Metric dataset. Each column represents seven samples of a class. Colors underline the slice structure of point clouds.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[8] C. Wang, M. Pelillo, and K. Siddiqi, "Dominant set clustering and pooling for multi-view 3d object recognition." in *Proceedings of British Machine Vision Conference (BMVC)*, 2017.

[9] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[10] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[11] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.

[12] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," in *3D Deep Learning Workshop*, 2016, pp. 1–9.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5099–5108.

[15] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 863–872.

[16] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[19] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[20] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[21] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

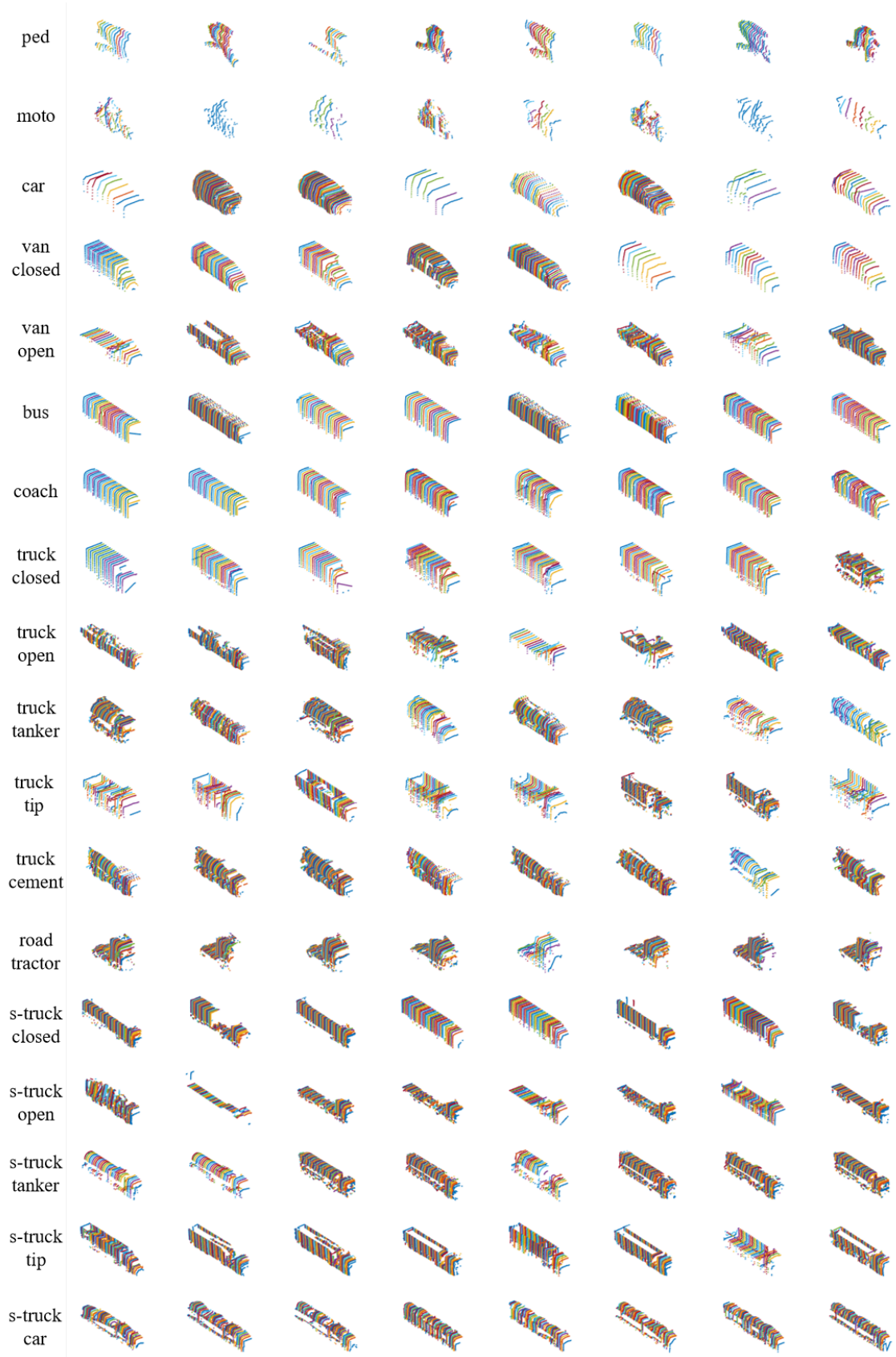


Fig. 9: Non-metric dataset. Each column represents seven samples of a class. Colors underline the slice structure of point clouds.