PointNet with Spin Images

Jakub Střelský[1][0000−0002−5096−863X]

Charles University, Faculty of Mathematics and Physics, Department of Software and Computer Science Education, Prague, Czech Republic

Abstract. Machine learning on 3D point clouds is challenging due to the absence of natural ordering of the points. PointNet is a neural network architecture capable of processing such unordered point sets directly, which has achieved promising results on classification and segmentation tasks. We explore methods of utilizing point neighborhood features within PointNet and their impact on classification performance. We propose neural models that operate on point clouds accompanied by point features. The results of our experiments suggest that traditional spin image representations of point neighborhoods can improve classification effectiveness of PointNet on datasets comprised of objects that are not aligned into canonical orientation. Furthermore, we introduce a feature-based alternative to spatial transformer, which is a sub-network of PointNet responsible for aligning misaligned objects into canonical orientation. Additional experiments demonstrate that the alternative might be competitive with spatial transformer on challenging datasets.

1 Introduction

Machine analysis of 3D geometrical data is becoming an important area of research because of the increasing demand from applications such as autonomous driving. Thanks to the advances in the development of depth sensors, large amounts of such data are publicly available, which makes development and employment of data-oriented algorithms more accessible.

Convolutional neural networks (CNNs) have established state-of-the-art results in computer vision tasks such as image classification, but their application on tasks involving 3D data remains a problem. CNNs rely on regular grid representations that are very memory demanding and computationally expensive to process in 3D. CNNs were already utilized on voxel data, but even with optimization like hierarchical octrees, this solution is limited to grids of resolution 256^3 and will probably be very difficult to scale to finer resolutions.

Point cloud representation is appealing alternative to voxel representation for several reasons. The data sparsity is naturally reflected in point cloud representation which is typically much more concise compared to voxel representation. There is no trade-off between precision and memory demands like in the case of voxel representation and point cloud can capture an arbitrary level of detail. Point clouds are also close to raw measurements of sensors as LiDAR or RGBD cameras. Automatic machine analysis of point cloud representation is, however,

challenging, mainly because the points of a point cloud have no ordering so any permutation of the points represents the same point cloud.

PointNet [8] is a neural network architecture designed to process point cloud representations directly. It obtains hidden representation of each input by independently processing each point by a Multi Layer Perceptron (MLP). Those representations are then aggregated by maximum pooling to obtain a permutation invariant representation. PointNet provided a considerable boost in computational efficiency of 3D object classification while keeping up in the terms of classification performance with other state-of-the-art approaches. Furthermore, the model is also straightforwardly applicable to other useful tasks involving 3D data such as the task of point cloud segmentation. PointNet effectively samples the 3D domain via so called point functions. But, unlike e.g. voxelization, it works in an efficient and data dependent way. Unfortunately, when used on objects appearing in an arbitrary orientation, the effectiveness of sampling the 3D domain seems limited, as the number of locations in which the points can be located is greatly increased.

The authors utilize spatial transformer network [2] in order to deal with this issue, but aligning point clouds to canonical orientation is a difficult task, which would itself require recognition of object classes in some cases. Furthermore, the spatial transformer itself relies on PointNet within PointNet, so the alignment capabilities of spatial transformer share the limitations of PointNet.

It seems intuitive that additional local information extracted from point neighborhoods could be beneficial for classification, especially in the case when point clouds are not aligned to canonical orientation. A successor of PointNet called PointNet++ [9] was introduced in order to add capabilities of utilizing the local neighborhood features into PointNet by applying a small PointNet on point neighborhoods and repeating the process on gradually higher-dimensional point clouds.

In this paper, we follow the direction of PointNet++ towards adding local point features into PointNet. We focus on the tasks in which the input objects are not aligned into canonical position. We develop models based on rotation invariant point features and PointNet. Several experiments were conducted in order to compare our models with the PointNet baselines. Our model manifests comparable classification performance on datasets with manually aligned objects and noticeably better performance on datasets in which objects are oriented arbitrarily. We also propose a simple feature-based heuristic for point cloud alignment in the form of a neural network layer, and we empirically show that our heuristic can be more effective than spatial transformer in certain cases.

2 Related work

There are several ways of applying machine learning to 3D point clouds that are currently actively researched. One common way is to transform the point cloud representation to voxel grid representation, which can be processed by 3D CNNs [7], [12], [10]. Scaling these methods to classification of complex objects which

require fine level of detail to be distinguished is, nevertheless, difficult due to the inherent trade-off between managable memory demands and admissible loss of information.

Sequences of images obtained by rendering the point cloud representation from different view-points is another grid-based representation which can be processed naturally by 2D CNNs [4], [11], [10]. These approaches have established state-of-the-art results on classification benchmarks. Limitation of these methods are difficulty of their extension for different tasks like the point cloud segmentation. The point cloud representations are also in principle capable of capturing more complex data than surfaces and such data would be difficult to render into images without potentially loosing important information.

Point clouds can also be processed directly by several recent models. PointNet [8] applies a neural network on every input coordinate of the point cloud independently and extracts a permutation invariant representation by applying global pooling. Spatial transformer is applied on the input coordinates to deal with variance of input orientations and is also applied on the hidden representations. From the reported results, it is, however, not clear how the model would perform on datasets with objects of highly varying pose. PointNet++ [9] utilizes small PointNet networks on point neighborhoods across several scales in order to introduce local point features to the original architecture. Such local features are powerful, since they are learned from the point cloud data directly, but they not invariant under rotations, which might cause a decrease of classification performance on unaligned data. Kd-Net [5] allows convolution-like processing of point clouds by building a balanced kd-tree and then following a bottom-up traversal of the tree, while applying learned affine transformation and non-linearity on features contained in child nodes in each parent node. Kd-net is also not invariant under rotations and could also potentially benefit from rotation invariant local features.

3 Methods

Our work extends PointNet [8] by using local point features in a way that is similar to PointNet++ [9]. We focus on features which are rotation invariant, and we investigate if such features have a positive impact on classification on unaligned data. In this section, we describe feature extraction techniques and our method for aligning point clouds. Section 5 describes the exact models derived from methods of this section.

3.1 Spin Images

There is a plethora of descriptors invariant under rigid transformations which could be incorporated into PointNet. In this work, we opt for spin images [3] primarily because the representation of spin images can be straightforwardly processed by empirically succesfull CNNs. We leave investigation of other descriptors for future work. We briefly summarize the spin images technique here.

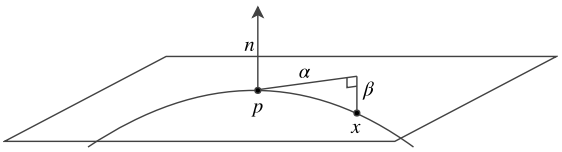

Fig. 1. Spin image coordinates of point $\boldsymbol{x}$ when computing the representation for point $\boldsymbol{p}$ with associated normal vector $\boldsymbol{n}$.

In order to extract a spin image of a neighborhood around a point $\boldsymbol{p} \in \mathbb{R}^3$, knowledge of a normal vector $\boldsymbol{n} \in \mathbb{R}^3$ associated with $\boldsymbol{p}$ is required. For classification of point clouds that represent surfaces, this is not a very restrictive assumption, since normal vectors can be estimated from eigenvalue decomposition of local covariance matrices [1].

Given input points of a neighborhood around $\boldsymbol{p}$, every input point $\boldsymbol{x}$ is projected to the new coordinates (α, β) indicated in Figure 1 and accumulated into a two-dimensional histogram. If the points carry additional information in the form of a vector such as the color, the vector can also be accumulated into bins for example by addition. Spin images have appealing properties. Their descriptiveness is easily adjusted by changing the histogram resolution. They can also be made local and global point cloud descriptors by changing the size of the point neighborhood.

3.2 Spin Coordinates

Spin image coordinate transformation also provides a straightforward way to make PointNet++ features invariant under rotation, simply by using the transformation on local point clouds before they are processed by local PointNets of PointNet++. This is essentially equivalent to forcing the point functions learned by PointNet to be axially symmetrical around the local normal vector. We will refer to these features as the spin coordinates.

3.3 Orientation Alignment Layer

For a given object, in the form of point cloud and corresponding point features, if we were able to select points which are accompanied by distinctive features, then objects of the same class could be approximately aligned in a coordinate system that would be based on these points. Based on this idea, we designed a simple heuristic algorithm, which we call orientation alignment layer (Algorithm 1). The algorithm is also easily extensible to the problem of pose alignment, but we will only consider alignment for simplicity (see Section 6.1 for clarification of pose and orientation). Algorithm 1 rotates an input point cloud so that points with selected features would be positioned in a direction of canonically chosen orthogonal vectors.

Algorithm 1 Orientation Alignment Layer(X)

Input: $X = (\boldsymbol{x_i})_{i=1}^n$ where $\boldsymbol{x_i} \in \mathbb{R}^{3+d}$ ▷ sequence of coordinates and features
Output: sequence of n rotated points from X
  Let $(\boldsymbol{c_i})_{i=1}^n, \boldsymbol{c_i} = (x_{i,1}, x_{i,2}, x_{i,3}), \boldsymbol{x_i} \in X$ ▷ sequence of coordinates
  Let $(\boldsymbol{f_i})_{i=1}^n, \boldsymbol{f_i} = (x_{i,4}, x_{i,5}, ...x_{i,3+d}), \boldsymbol{x_i} \in X$ ▷ sequence of features
  $i, j \leftarrow$ Feature Selection Heuristic$((\boldsymbol{f_i})_{i=1}^n, 2)$ Algorithm 2
  Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$ be two orthogonal unit vectors chosen canonically
  $\boldsymbol{R_1} \leftarrow$ the rotation matrix such that $\boldsymbol{R_1} \frac{\boldsymbol{c_i}}{\|\boldsymbol{c_i}\|} = \boldsymbol{x}$
  $\boldsymbol{v} \leftarrow \boldsymbol{R_1 c_j}$
  $\boldsymbol{v} \leftarrow \boldsymbol{v} - (\boldsymbol{v} \cdot \boldsymbol{x})\boldsymbol{x}$
  $\boldsymbol{R_2} \leftarrow$ the rotation matrix such that $\boldsymbol{R_2} \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} = \boldsymbol{y}$
  return $(\boldsymbol{R_2 R_1 c_i}, \boldsymbol{f_i})_{i=1}^n$

Features that are common within a class but uncommon within an individual point cloud could be good candidates for the selection. Selection of features that are frequent within a class provides consistent orientation of objects within the same class. Furthermore, selection of features that are unique within a point cloud provides robustness in case of presence of multiple good candidates in the point cloud. These rather abstract qualities are, however, not straightforward to define and compute quantitatively.

We have chosen simple heuristic approach to the feature selection described by Algorithm 2. We do not have satisfactory justification of the heuristic, but it seems intuitive that selecting the features with maximal entries could provide at least somewhat consistent selection. Besides, when the heuristic is applied within hierarchical PointNet which apply max pooling of local features, we assume that the maximal features are likely to be important for classification.

Algorithm 2 Feature Selection Heuristic(F, k)

Input: $F = (\boldsymbol{f_i})_{i=1}^n$ where $\boldsymbol{f_i} \in \mathbb{R}^d$ ▷ sequence of features
  k = number of features to be selected
Output: k integer indices of selected feature vectors
  $F' \leftarrow (f'_i)_{i=1}^n, f'_i = \max \boldsymbol{f_i}$ ▷ maximum entries of features
  return indices of k largest elements of F'

4 Datasets

Our experiments were based on datasets which are described in this section.

ModelNet

The Princeton ModelNet dataset [12] has two variants: ModelNet10 which contains 4899 objects of 10 categories and ModelNet40 which contains 12311 objects

of 40 categories. We use point clouds consisting of 1024 points extracted from the original CAD models by [8]. In the case of ModelNet10, the individual objects are manually aligned (each object has the identical pose). The objects are centered and scaled so that each object fits into the unit ball. We use the original train/test splits consisting of 3991/908 objects from ModelNet10 and 9843/2468 objects from ModelNet40. We further split the train partitions for the purpose of validation.

Augmented ModelNet10

We prepared a challenging modification of the ModelNet10 dataset by replacing each original object with two modified copies. Every object is subject to random rotation of angle up to π. The objects are translated by a vector of random direction and of random length from uniform distribution on $[0, 0.25]$. Additionally, up to 3 cubes of random size and orientation are inserted into each point cloud, so that they never intersect with the original objects. The inserted cubes were represented by 50 points, and their maximum size was $0.5 \times 0.5 \times 0.5$.

SHREC17

A subset of the ShapeNet dataset consisting of 51,162 triangle meshes of objects. We use the provided 70%/10%/20% training/validation/test split for the experiments. There are two variants of the SHREC17: normal and perturbed. Here, we use the perturbed dataset where the objects are subjected to random rotations. Point clouds are sampled from the provided triangle meshes by sampling the triangles with probability proportional to their area, and then sampling the triangle surfaces uniformly so that the obtained point clouds are consistent with ModelNet point clouds. We use 1024 sampled points for the classification and an additional feature engineering as required. For the methods that require normal vectors, we calculate the normal vectors from the meshes rather than from the sampled point clouds. It should be noted that there are both inward-pointing and outer-pointing normal vectors in every mesh, which most likely hinders the performance of some of the methods relying on the normal vectors to some extent.

5 Models

In this section we provide a description for every model that will be evaluated in the next section. The model architectures were selected so that their sizes would be roughly comparable in terms of the number of learnable parameters. We did not fine-tune hyperparameters in this work as we were mainly interested in observing major differences of models, and we did not intent to achieve the best performance.

1 PointNet: A small PointNet model. The shared MLP part of the model before the maximum pooling is formed of fully connected layers with sizes 64, 64, 64, and 256 neurons. The MLP part of the model after maximum pooling consists of dropout with probability 0.2 and two fully connected layers with 512 and 128 neurons.

2 PointNetST: The same model as the previous PointNet model, but a spatial transformer parametrized by linear or affine transformations is additionally inserted as the first layer for appropriate tasks, that is: the linear transformer for the tasks where the input objects are possibly rotated but not translated and the affine transformer for the rest. The spatial transformer itself is a PointNet consisting of layers with 32, 32, and 128 neurons before the maximum function and then a single layer of 128 neurons.

3 PointNetSTL: Re-implementation of the original PointNet [8] with 2 differences: we only use the first spatial transformer and we do not utilize batch-normalization layers.

4 Spin Images: This model makes predictions based on spin images only and does not utilize the coordinates of the features. Spin images are of size 32×32. Thirty-two spin images of radius 1 are utilized. The 32 points are selected by farthest sampling algorithm. The model consists of 3D convolutional layers with 32, 64 and 128 filters of size $1 \times 3 \times 3$ followed by $1 \times 2 \times 2$, $1 \times 2 \times 2$ and $32 \times 1 \times 1$ maximum pooling, respectively, followed by dropout with probability 0.2 and fully connected layers with 512 and 256 neurons.

5 Hierarchical Spin Images: This model utilizes both the representations obtained from the spin images and the coordinates. Spin images are of size 16×8. Thirty-two spin images of radius 0.6 are utilized. Spin images are processed by the same 3D CNN from previous model, then the representation is concatenated with point coordinates and fed into PointNet (Model 1).

6 Hierarchical PointNet: Thirty-two point neighborhoods, each consisting of 32 nearest points, are utilized. Each neighborhood is processed by a small PointNet consisting of layers of 32, 32, and 32 neurons followed by max pooling and 64 neurons. The extracted embedding is concatenated with point coordinates and fed into PointNet (Model 1).

7 Hierarchical PointNet Spin Coordinates: The same model as the Hierarchical PointNet, but the local coordinates are first transformed using the spin image coordinate transformation.

8 Hierarchical PointNet Orientation Alignment: This is the same as the previous model, except that orientation alignment layer (see Algorithm 1) is additionally inserted after the concatenation of embeddings and coordinates.

6 Experiments

In this section, we describe the experiments that were carried out in order to empirically compare suggested models and features. In order to evaluate performance of the models, we measured classification accuracy on official test sets given in the respective benchmark tasks. We further split the original training

sets into two parts for training and validation. We evaluate performance of models after each epoch of training on validation partition of data. A model with the best performance on validation data across all epochs is taken as a result of the training. The categories are not balanced in terms of their frequencies, so the data are split in stratified manner meaning that frequencies of the categories is the same in training and validation parts. We use the Adam optimization algorithm with parameters ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.99$) and batch size 128. Training strategy is adjusted to take imbalanced categories into account by filling batches in a way such that categories are uniformly distributed in each batch. We apply L2-regularization of network weights with $\lambda = 0.0001$.

6.1 Robustness to Rotations

Let us informally define notions about object orientation in order to clarify descriptions of experiments from this section. We assume that every object has a unique **reference pose** which is given by semantics. Reference pose is described by a canonical coordinate system. **Pose** of an observed object is then the coordinate system (taken w.r.t. the canonical system) in which the object is in its reference pose. When we refer to **orientation** of an object, we mean the pose of the object without translation element, i.e. the coordinate systems are zero centered. We will also use the notion of **orientation vector**, by which we mean a vector parallel to one canonically chosen axis of the orientation coordinate system.

With the following experiment, we tested robustness of PointNet against rotations of point cloud objects. We augmented the ModelNet10 dataset by rotating the objects from dataset randomly. Two rotated samples of each object were placed into the augmented dataset instead of each original object. We then performed 10-fold cross-validation on the augmented dataset to evaluate classification accuracy. The models PointNet and PointNetST (see Model 1 and 2) were subject to the experiment.

The rotation matrices used for rotating the objects were sampled in such a way, so that orientation vectors of all objects were distributed uniformly on a cap of the unit sphere with the apex at the original orientation vectors. The tested maximal angles of the rotations were $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3\pi}{4}$, and π.

Table 1. Mean accuracy with standard deviation on rotated ModelNet10 with increasing maximal angle of object rotations.

	Maximum angle				
Model	0	0.25π	0.5π	0.75π	π
PointNet	0.90 ± 0.01	0.87 ± 0.02	0.83 ± 0.02	0.79 ± 0.02	0.79 ± 0.02
PointNetST	0.91 ± 0.01	0.88 ± 0.02	0.85 ± 0.02	0.78 ± 0.03	0.77 ± 0.03

Table 1 reveals that PointNet without spatial transformer is quite robust versus rotations. Higher accuracy could be achieved with more augmentation

and further regularization techniques. Nevertheless, the decrease of accuracy is noticeable. Spatial transformer clearly helps for the rotations of small angles, but does not seem to help for the rotations of large angles, where the accuracy is nearly the same for the PointNet and PointNetST models.

6.2 Entropy of Orientation Distributions

Spatial transformer was designed to decrease variance of orientations of the point cloud objects present in the data. It seems highly probable that increase of accuracy is correlated with the decrease of variance of orientations, but if we wanted to compare other mechanisms for decreasing orientation variance, it might be better not to rely solely on accuracy which might be also affected by other factors. With access to the original orientation of each object, we can directly measure how the variance of the orientations is affected by transformations produced by spatial transformer or other techniques.

We have only considered orientation vectors in this experiment for simplicity, even though an orientation vector $\boldsymbol{v}$ is not sufficient to fully describe orientation of an object in 3D since the rotation component around $\boldsymbol{v}$ is left unspecified. The unit orientation vectors of the objects can be viewed as samples from a distribution X on the two-dimensional unit sphere $\boldsymbol{S}$ embedded in $\mathbb{R}^3$, which we will refer to as the **orientation distribution**. The differential entropy $H(X)$ of the distribution X with a probability density function f whose support is $\boldsymbol{S}$ defined as:

$$H(X) = -\int_{\boldsymbol{x}\in\boldsymbol{S}} f(\boldsymbol{x}) \log\left(f(\boldsymbol{x})\right) d\boldsymbol{x} \tag{1}$$

is a measure (not in the mathematical sense) of uncertainty of the distribution. The lower the entropy of orientation distribution within a dataset is, the more aligned the dataset is. By comparing entropy of the orientation distribution of the augmented input data and the data transformed by the spatial transformer, we can observe whether the spatial transformer performs alignment of the objects or not. We do not have access to the probability density function directly for computation of the entropy, but we can estimate the entropy from samples. We chose the Kozachenko-Leonenko entropy estimator [6] for the purpose because it relies on pairwise distances of the samples, which can be computed trivially, whereas other approaches to the problem, e.g. that rely on density estimation, are not so straightforwardly applicable on spherical distributions.

Let $X = (\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n})$, $\boldsymbol{x_i} \in \mathbb{R}^d$ be the samples from the distribution subject to the entropy estimation. Let $(d_i)_{i=1}^n$ be the distances of the samples $\boldsymbol{x_i}$ to their k-th nearest neighbors, then the Kozachenko-Leonenko estimate can be written as:

$$\widehat{H}(X) = \psi(n) - \psi(k) + \log(c) + \frac{d}{n}\sum_{i=1}^{n} \log(d_i) \tag{2}$$

where ψ is the digamma function, and c is the volume of the unit ball dependent on the norm used to calculate the distances. In the case of X being distribution

on the unit sphere, the distance is defined by the angle between samples, and the $c = 2\pi(1 - \cos 1)$ is the surface area of the spherical cap with the unit angle between the apex and the edge.

We repeated the experiment from Section 6.1 and we estimated the differential entropy of the orientation distributions of the data before and after application of the transformations generated by the spatial transformer. Since spatial transformer is not restricted to produce only orthogonal transformations, we normalized the transformed orientation vectors in order to obtain spherical distribution.

Table 2. Results of the entropy estimation experiment with increasing maximal angle of rotation. Analytical differential entropy $H(X)$ of uniform distributions of the spherical caps is depicted as a reference. $\widehat{H}(X)$ is the estimated entropy of input orientation distribution. $\widehat{H}'(X)$ is the estimated entropy of orientation distribution of the objects transformed by the spatial transformer or by the orientation alignment layer. The last column is simply the difference of 3rd and 4th columns. The values were measured on validation splits of 10-fold cross validation and the reported results are averages.

Spatial transformer				
Angle	$H(X)$	$\widehat{H}(X)$	$\widehat{H}'(X)$	difference
0.25π	0.61	0.54	-0.45	0.99
0.5π	1.84	1.77	1.55	0.22
0.75π	2.37	2.27	2.18	0.09
π	2.53	2.42	2.36	0.06
Orientation alignment layer				
π	2.53	2.42	1.68	0.74

Results of the entropy experiment are given in Table 2. We can see that there is a relation between accuracy and differential entropy of orientation distribution by comparing the results with the experiment from previous section summarized by Table 1, where spatial transformer was most helpful in the cases of maximum angle up to $\frac{\pi}{2}$, which was also the case in this experiment. The experiment suggests that the spatial transformer probably helps in certain cases, but it is likely not a universal remedy for the problem of pose alignment. Careful adjustment of the spatial transformer hyper-parameters might be needed in order to enjoy its benefits. Orientation alignment layer performed better than spatial transformer on fully uniform rotations with more significant entropy reduction. Disadvantage of orientation alignment layer is that it performs in a way that is independent on the input orientation distribution entropy by the nature of Algorithm 1, so the entropy after transformation is the same for all tests.

6.3 Benchmarks

On datasets which are mostly aligned (ModelNet datasets), the PointNet models 1–3 perform well and additional local features wre not helpful for classification.

Fig. 2. left: orientation distribution of inputs in rotated ModelNet10 (two different view angles), **right**: orientation distribution of objects after orientation alignment layer

Table 3. Test classification accuracy of presented methods on the ModelNet10, ModelNet40, SHREC17, and Augmented ModelNet10 datasets. Number of parameters slightly differ for each variant because of the size of output layer. The reported number of parameters is taken from the ModelNet10 variant. Micro indicates the accuracy averaged over the test set. Macro is the average of class accuracy averages.

	ModelNet10		ModelNet40		SHREC17		AModelnet10		
Model	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	# params
1 PN	0.90	0.90	**0.85**	**0.82**	0.42	0.30	0.38	0.37	291k
2 PNST	**0.91**	**0.91**	**0.85**	0.81	0.42	0.30	0.38	0.37	314k
3 PNSTL	**0.91**	**0.91**	**0.85**	0.81	0.42	0.31	0.22	0.23	1.6M
4 SI	0.83	0.82	0.71	0.67	**0.67**	**0.51**	**0.79**	**0.78**	438k
5 HSI	0.88	0.88	0.84	0.81	0.63	0.50	0.75	0.74	403k
6 HPN	0.89	0.89	**0.85**	**0.82**	0.44	0.34	0.53	0.52	293k
7 HPNS	0.89	0.90	0.84	0.80	0.46	0.38	0.61	0.61	293k
8 HPNOA	0.73	0.71	0.66	0.62	0.58	0.45	0.62	0.60	293k

On the other two datasets, which are perturbed by rotations and additionally translations in the case of Augmented ModelNet10, the Model 4 was superior to others probably because of its invariance under rigid transformations. The Model 5 seems stable in the sense that it is never substantially worse than the best model in each task, so it seems that a combination of rotation invariant features with absolute point coordinates is a promising direction.

The performance of the Model 8, which utilizes orientation alignment layer, was inferior in most cases, because the orientation alignment layer is harmful when the data are well aligned. However, we see that in the SHREC17 task, the presence of orientation alignment is beneficial compared to Models 6 and 7, which indicates that reduction of orientation distribution entropy was achieved.

7 Conclusion

We have empirically demonstrated that PointNet can benefit from point neighborhood features on classification tasks where objects represented by point clouds may appear in arbitrary orientation. Spin images seem to be promising candidates of point neighborhood features. Experiments also suggest that the spatial transformer technique, employed by PointNet in order to deal with the problem of object orientation alignment, may be difficult to utilize properly depending

on the data. We have also proposed a simple experiment to measure quality of alignment achieved by spatial transformer interpretatively on the tasks where orientation of objects is known in advance. We have also introduced a simple heuristic algorithm as an alternative to spatial transformer, which we call orientation alignment layer. Further experiments suggest that orientation normalization layer might be able to achieve better quality of orientation alignment than spatial transformer on difficult data.

In the future work, we would like to design better feature selection method for our orientation alignment layer in order to make it more robust. We believe that spatial transformer could also benefit from local point features, and we would like to investigate the idea. It would also be possible to combine spatial transformer and orientation alignment layer into single model. Finally, we intent to compare spin images with other rotation invariant features.

References

1. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. SIGGRAPH Comput. Graph. 26 (1992)
2. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In: Advances in Neural Information Processing Systems 28, pp. 2017–2025 (2015)
3. Johnson, A.E., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI). pp. 433–449 (1999)
4. Kanezaki, A., Matsushita, Y., Nishida, Y.: Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
5. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: 2017 IEEE International Conference on Computer Vision (ICCV) (2018)
6. Kozachenko, L.F., Leonenko, N.N.: Sample estimate of the entropy of a random vector. Probl. Peredachi Inf. 23, 9–16 (1987)
7. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2015)
8. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2017)
9. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Neural Information Processing Systems (NIPS) (2017)
10. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.: Volumetric and multi-view cnns for object classification on 3d data. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2016)
11. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.G.: Multi-view convolutional neural networks for 3d shape recognition. In: Proc. ICCV (2015)
12. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2015)