Iterative Transformer Network for 3D Point Cloud

Wentao Yuan David Held Christoph Mertz Martial Hebert
The Robotics Institute
Carnegie Mellon University
{wyuan1, dheld, cmertz, mhebert}@cs.cmu.edu

Abstract

3D point cloud is an efficient and flexible representation of 3D structures. Recently, neural networks operating on point clouds have shown superior performance on tasks such as shape classification and part segmentation. However, performance on these tasks are evaluated using complete, aligned shapes, while real world 3D data are partial and unaligned. A key challenge in learning from unaligned point cloud data is how to attain invariance or equivariance with respect to geometric transformations. To address this challenge, we propose a novel transformer network that operates on 3D point clouds, named Iterative Transformer Network (IT-Net). Different from existing transformer networks, IT-Net predicts a 3D rigid transformation using an iterative refinement scheme inspired by classical image and point cloud alignment algorithms. We demonstrate that models using IT-Net achieves superior performance over baselines on the classification and segmentation of partial, unaligned 3D shapes. Further, we provide an analysis on the efficacy of the iterative refinement scheme on estimating accurate object poses from partial observations.

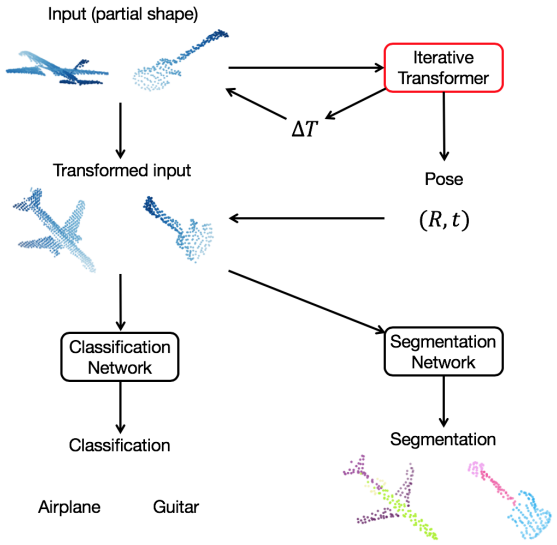


Figure 1: Iterative Transformer Network (IT-Net) predicts rigid transformations from point clouds in an iterative fashion. It can be used independently as a pose estimator or jointly with classification and segmentation networks.

1. Introduction

3D point cloud is the raw output of most 3D sensors and multiview stereo pipelines [6] and a widely used representation for 3D structures in applications such as autonomous driving [7] and augmented reality [11]. Due to its efficiency and flexibility, there is a growing interest in using point clouds for high level tasks such as object recognition, skipping the need for meshing or other post-processing. These tasks require an understanding of the semantic concept represented by the points. On other modalities like images, deep neural networks [8, 12] have proven to be a powerful model for extracting semantic information from raw sensor data, and have gradually replaced hand-crafted features. A similar trend is happening on point clouds. With the introduction of deep learning architectures like PointNet [16], it is possible to train powerful feature extractors that outperform traditional geometric descriptors on tasks such as shape classification and object part segmentation.

However, existing benchmark datasets [23, 25] that are used to evaluate performance on these tasks make two simplifying assumptions: first, the point clouds are sampled from complete shapes; second, the shapes are aligned in a canonical coordinate system[1] (see Figure 2). These assumptions are rarely met in real world scenarios. First, due to occlusions and sensor limitations, real world 3D scans usually contain holes and missing regions. Second, point clouds are often obtained in the sensors coordinates, which do not align with the canonical coordinates of the object model. In other words, real 3D point cloud data are *partial* and *unaligned*.

[1] In ModelNet [23], shapes are allowed to have rotations, but only along the vertical axis.

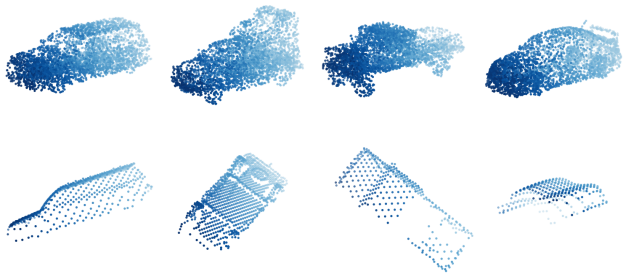
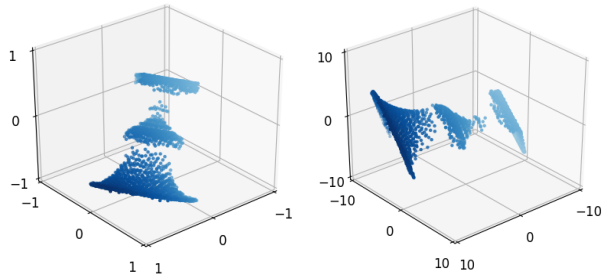


Figure 2: Complete point clouds in canonical frame (top row) versus partial point clouds in our dataset (bottom row). Note how the canonical frame defines a correspondence across different models in the same category.



(a) Input (b) Transformed input

Figure 3: T-Net [16] scales and distorts the input. Note the different scales on the plots.

In this work, we tackle the problem of learning from partial, unaligned point cloud data. To this end, we build a dataset consisting of partial point clouds generated from virtual scans of CAD models in ModelNet and ShapeNet. Our dataset contains challenging inputs with arbitrary 3D rotation, translation and realistic self-occlusion patterns.

A key challenge in learning from such data is how to learn features that are invariant or equivariant with respect to geometric transformations. For tasks like classification, we want the output to remain the same if the input is transformed. This is called invariance. For tasks like pose estimation, we want the output to vary according to the transformation applied on the input. This is called equivariance.

This property can be achieved via a transformer network [9], which predicts a transformation that is applied to the input before feature extraction. This allows explicit geometric manipulation of data within networks, so the networks can learn to align the inputs into a canonical space that makes subsequent tasks easier. T-Net [16] is a transformer network based on PointNet that operates on 3D point clouds. However, T-Net outputs an unconstrained affine transformation. This can introduce undesirable shearing and scaling which causes the object to lose its shape (see Figure 3). Moreover, T-Net is evaluated on inputs with 2D rotations only.

Therefore, we propose a novel transformer network on 3D point clouds, named Iterative Transformer Network (IT-Net). It has two major differences from T-Net. First, it outputs a rigid transformation instead of an affine transformation. This preserves the shape of the inputs and leads to better performance on subsequent tasks such as shape classification and part segmentation. Further, this allows us to use the outputs of IT-Net directly as estimates of object poses. Second, instead of predicting the transformation in a single step, IT-Net takes advantage of an iterative scheme which decomposes a large transformation into smaller ones that are easier to predict. We note that similar ideas of iterative pose refinement have been employed by classical image and point cloud alignment algorithms [3, 15].

We evaluate the performance of IT-Net on 3 different tasks – pose estimation, shape classification and object part segmentation (Figure 1) – using inputs with 3D rotation and translation from our partial point cloud dataset. Our experiments demonstrate that models trained with IT-Net can learn transformation-invariant or equivariant features from challenging partial inputs. Further, IT-Net can be readily integrated into various state-of-the-art networks and improve their performance on different tasks.

The key contributions of our work are as follows:

- We propose a new transformer network on 3D point clouds that uses iterative refinement to predict rigid transformations;
- We show that our transformer can be used independently as a pose estimator or trained jointly with classification and segmentation networks and increases performance over baselines in both sets of tasks;
- We introduce a benchmark dataset for shape classification and object part segmentation consisting of partial, unaligned point clouds.

2. Related Work

Feature Learning on Point Clouds Traditional point cloud features [1, 19, 21] rely on statistical properties of points such as local curvatures. They do not encode semantic information and it is non-trivial to find the combination of features that is optimal for specific tasks.

PointNet [16, 17] proposes a way to extract semantic and task-specific features from point clouds using a deep neural network. The key idea of PointNet is to use a symmetric function (e.g. max-pooling) to aggregate pointwise features so that the global feature is invariant to permutations of the points. A drawback of PointNet is that it does not account for local interactions among points. Thus, several extensions [13, 22] which augment the input with information from local neighborhoods of points have been proposed.

Most datasets [23, 25] used to evaluate feature learning on point clouds consist of complete point clouds. A few works [16, 26] have investigated feature learning from partial point clouds. However, they all assume that the point clouds are aligned in a canonical coordinate system. In this work, we show how to remove this assumption using a transformer network.

Spatial Transformer Network Spatial Transformer Network (STN) [9] is a network module that performs explicit geometric transformations on the input data. STN can be thought of as a geometry predictor which models the complicated non-linear relationship between the appearance of the image and geometric transformations. It can be trained jointly with classification networks and has the benefit of introducing invariance to geometric transformations.

Inverse Compositional Spatial Transformer Network (IC-STN) [14] is an extension of STN that makes use of an iterative alignment scheme analogous to the Lucas-Kanade algorithm [15]. It demonstrates that geometric transformations can be predicted from images more accurately in an iterative fashion.

3. Iterative Transformer Network

Iterative Transformer Network (IT-Net) takes a 3D point cloud and produces a transformation that can be used directly as a pose estimate or applied to the input before feature extraction for subsequent tasks. Alternatively, we can think of its output as the transformation between the input's coordinates and the canonical coordinates of the semantic concept represented by input, which can be defined either explicitly as in Figure 2 or implicitly as any coordinate system that makes subsequent tasks (e.g. classification) easier.

IT-Net has two key features that differentiate it from existing transformer networks on 3D point clouds: first, it predicts a 3D rigid transformation; second, the final output is composed of multiple transformations produced in an iterative fashion. We will introduce these features one by one followed by additional implementation details.

3.1. Rigid Transformation Prediction

A 3D rigid transformation T is an element of the Special Euclidean group $SE(3)$. It consists of a rotation R and translation $\mathbf{t}$ where R is a 3×3 matrix satisfying $RR^T = I, \det(R) = 1$ and $\mathbf{t}$ is a 3×1 vector. Due to the constraints on R, it is inconvenient to represent the rotation as a 3×3 matrix during optimization. Thus, many classical [3] as well as modern deep learning methods [10, 24] parametrize 3D rotations with unit quaternions. This parametrization allows us to map an arbitrary 4D vector to a valid rotation.

Similar to PointNet [16] used for classification, a single iteration IT-Net consists of a shared multi-layer perceptron for each point and a max-pooling aggregation function followed by fully connected layers, but instead of predicting class scores, it outputs 7 numbers – the first 4 are normalized into a unit quaternion $\mathbf{q}$ and the last 3 are treated as a 3D translation vector $\mathbf{t}$. Then, $\mathbf{q}$ and $\mathbf{t}$ are assembled into a 4×4 matrix $T = \begin{bmatrix} R(\mathbf{q}) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$ where $R(\mathbf{q})$ is the rotation matrix corresponding to $\mathbf{q}$. Representing the output transformation as 4×4 matrices turns the composition of two rigid transformations into a matrix multiplication, which is useful for the multi-iteration IT-Net introduced in 3.2.

In contrast to the affine transformation produced by T-Net, the rigid transformation predicted by IT-Net can be directly interpreted as a 6D pose, making it possible to use IT-Net independently for pose estimation. More importantly, rigid transformations preserve scales and angles. As a result, the appearance of a point cloud will not vary drastically if it is transformed by the output of IT-Net. This makes it possible to apply the same network iteratively (see Figure 4) to obtain a more accurate estimation of the transformation.

We note that it is possible to add a regularization term $\|AA^T - I\|$ that forces an affine matrix A to be orthogonal in order to achieve similar effects of predicting a rigid transformation[2]. We explore this possibility (named T-Net reg) in our experiments and show that the results are not as good as the network that directly predicts a rigid transformation.

3.2. Iterative Alignment

The idea of using an iterative scheme for predicting geometric transformations goes back to the classical Lucas-Kanade (LK) algorithm [15] for estimating dense alignment between images. The key insight of LK is that the complex non-linear mapping from image appearance to geometric transformations can be estimated iteratively using simple linear predictors. Specifically, at each iteration, a warp transformation Δp is predicted with a linear function that takes a source and a target image as inputs. Then, the source image is warped by Δp and the process is repeated. The final transformation is a composition of Δp at each step. Later, [2] shows that the parameters used to predict Δp can remain constant across iterations while achieving the same effect as non-constant predictors.

The same idea is employed in the Iterative Closest Point (ICP) algorithm [3] for the alignment of 3D point clouds. At each iteration of ICP, a corresponding set is identified and a rigid transformation ΔT is produced to align the corresponding points. Then, the source point cloud is transformed by ΔT and the process is repeated. Again, the final output is a composition of ΔT at each step. The effectiveness of ICP shows that the iterative refinement framework applies not only to images, but also to 3D point clouds.

[2] In [16], this regularization is added to the feature transformation, but not to the input transformation.

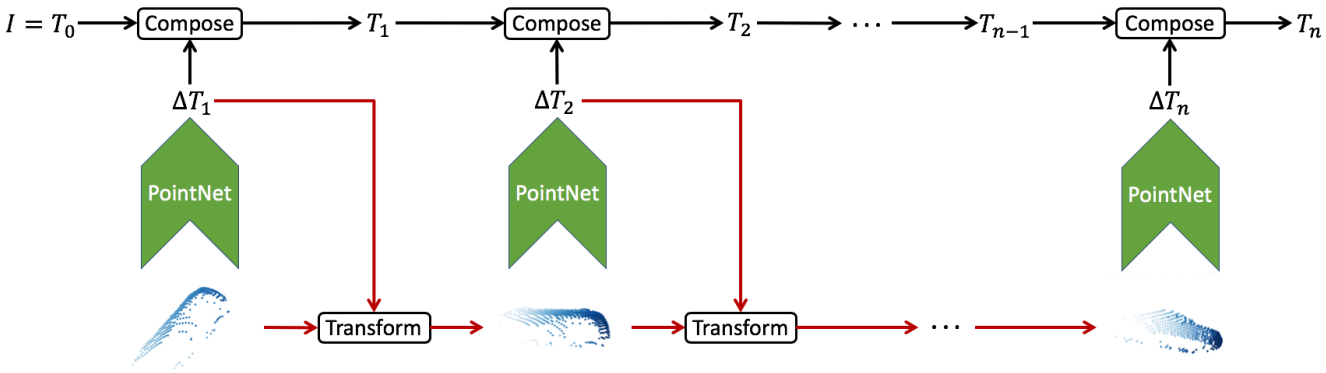


Figure 4: Illustration of the iterative scheme employed by IT-Net. At each iteration, the output of the network is used to transform the input for the next iteration. The network parameters (green arrows) are shared across iterations. The final output is a composition of the transformations predicted at each iteration. Arrows colored in red indicate places where the gradient flow is stopped to decorrelate the inputs at different iterations.

The multi-iteration IT-Net (Figure 4) can be viewed as an instantiation of this iterative framework. Specifically, the prediction of the transformation T is unfolded into multiple iterations. At the i-th iteration, an update transformation ΔT_i is predicted using the network introduced in 3.1. Then, the input is transformed by ΔT_i and the process is repeated. The final output after n iterations is a composition of the transformations predicted at each iteration, which can be written as a simple matrix product $T_n = \prod_{i=n}^{1} \Delta T_i$. Following [2], we use a fixed predictor (i.e. share the network's parameters) across iterations.

The iterative refinement scheme can be interpreted as a form of gradient descent. In LK, each update is a gradient step that brings the source closer to the target. In ICP, the update is no longer a gradient step, but it has similar effects of bringing the source closer to the target. Consequently, the magnitudes of the update transformations diminish as the algorithms converge, just like the gradient approaches 0 near a local optimum. In the case of IT-Net, the input is a single point cloud instead of a pair of images or point clouds and the transformation is predicted by a neural network. Despite these differences, we observe that the transformations predicted by IT-Net also have diminishing magnitudes (Figure 5). This behavior shows that in a sense, IT-Net is learning the "gradient transformation" that will bring the input shape closer to its appearance in the canonical frame.

3.3. Implementation Details

In addition to the key ingredients above, there are a couple of details that are important for the training of IT-Net.

First, we initialize the network to predict the identity transformation, i.e. $\mathbf{q} = [1\ 0\ 0\ 0]$, $\mathbf{t} = [0\ 0\ 0]$. In this way, the default behavior of each iteration is to preserve the transformation predicted by previous iterations, similar to residual networks [8]. This initialization is especially important when the number of iterations becomes large.

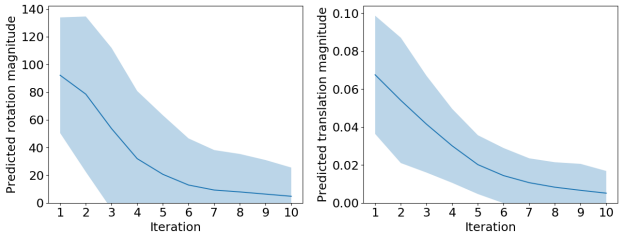


Figure 5: Magnitude of rotation (left) and translation (right) predicted by IT-Net decreases with more iterations.

Second, we stop the gradients flowing through the input transformations, indicated by red edges in Figure 4. This removes the dependence of the input at each iteration from the predictions of previous iterations. The reason behind this design choice can be perceived in terms of the analogy between IT-Net and gradient descent. If we interpret the prediction of IT-Net at each iteration as the gradient direction at a point, then it should not depend on the path that leads to the point.

4. Experiments

In this section, we evaluate the efficacy of IT-Net on various tasks. First, we demonstrate its use as a plug-in module that improves the performance of state-of-the-art classification and segmentation networks. Next, we show that it can also be used independently for pose estimation.

4.1. Partial 3D Shape Classification

In this section, we evaluate IT-Net on the classification of partial, unaligned shapes and show that it can improve the performance of classification networks by introducing invariance to geometric transformations.

Data As noted in Section 1, existing benchmark datasets for 3D shape classification such as ModelNet40 [23] fail to capture the incomplete and unaligned nature of real world 3D data. To test the performance of classifiers under a more realistic setting, we build the partial ModelNet40 dataset. To the best of our knowledge, this is the first controlled dataset for partial 3D shape classification.

The dataset consists of 81,212 point clouds from 40 categories, split into 78,744 for training and 2,468 for testing. Each point cloud is generated by fusing a sequence of depth scans of models in ModelNet40 into a point cloud. Figure 6 includes some qualitative examples. It can be seen that the point clouds are in a variety of poses with realistic self-occlusion patterns. Please refer to the supplementary for more details on the data generation.

Model The network used for the partial shape classification task consists of two parts – the transformer and the classifier. The transformer takes a point cloud and produces a transformation T. The classifier takes the point cloud transformed by T and outputs a score for each class. The entire network is trained with cross-entropy loss on the class scores and no explicit supervision is applied on T.

We compare classifiers trained with three different transformers, IT-Net, T-Net and regularized T-Net (T-Net reg) against the baseline classifier that does not use a transformer. All three transformers have the same architecture except for the last layer. Specifically, the architecture consists of three parts. The first part is a multi-layer perceptron that is applied on each point independently. It takes the $N \times 3$ coordinate matrix and produces a $N \times 1024$ feature matrix. The second part is a max-pooling function which aggregates the features in to a 1×1024 vector. The third part is another multi-layer perceptron which outputs the transformation parameters. For IT-Net, the last layer outputs 7 numbers for rotation (quaternion) and translation. For T-Net and T-Net reg, the last layer outputs 9 numbers to form a 3×3 affine transformation matrix A. For T-Net reg, a regularization term $\|AA^T - I\|$ is added to the loss with weight 0.001. Batch normalization is applied to all layers except the last layer. Further, we apply the iterative scheme described in Section 3.2 to each transformer for 2 iterations.

For the classifier, we use two state-of-the-art networks, PointNet [16] and Dynamic Graph CNN (DGCNN) [22]. The model architectures are identical to the ones used in their ModelNet40 experiments.

Training The networks are trained for 50 epochs with batch size 32. We use the Adam optimizer with an initial learning rate of 0.001, decayed by 0.7 every 6250 steps. The initial decay rate for batch normalization is 0.5 and gradually increased to 0.99. We clip the gradient norm to 20.

Classifier	PointNet						
Transformer	None	T-Net		T-Net reg		IT-Net (ours)	
# Iterations	0	1	2	1	2	1	2
Accuracy (overall)	59.97	66.04	35.13	65.84	67.06	68.72	**69.94**
Accuracy (class avg)	53.07	60.57	30.57	60.38	62.19	63.42	**65.05**
Classifier	DGCNN						
Transformer	None	T-Net		T-Net reg		IT-Net (ours)	
# Iterations	0	1	2	1	2	1	2
Accuracy (overall)	65.60	70.38	16.61	71.15	72.69	72.57	**74.15**
Accuracy (class avg)	58.34	62.66	13.73	65.49	66.40	65.16	**68.34**

Table 1: Classification accuracy on partial ModelNet40.

Results The overall accuracy and the average of accuracy per class are reported in Table 1. For both classifiers, IT-Net trained with 2 iterations brings the largest amount of improvement in accuracy over baselines that do not use any transformer network. This is evidence that the advantage of IT-Net over other transformer networks is agnostic to the architecture of the classifier.

Figure 6 shows some examples of how the inputs are transformed by IT-Net. It can be seen that without explicit supervision, IT-Net learns to transform the inputs into a canonical frame similar to the manually defined one shown in Figure 2. Inputs in various initial poses and even different categories are aligned in this canonical frame. This removes the variations introduced by geometric transformations of the input and simplifies the classification problem. Moreover, unlike T-Net, IT-Net does not introduce any scaling or shearing and preserves the shape of the input.

We note that in this experiment, we don't need to iterate as many times as in the case of pose estimation shown later in 4.3. The reason is that the classifiers themselves should be robust against small transformations to the input. Thus, instead of getting precise alignments, the transformer's job is to roughly align the inputs so that they can be recognized by the classifiers. Nevertheless, there is still a critical difference between using and not using the iterative scheme, as shown by the difference between IT-Net trained with 2 iterations and 1 iteration.

We also note that the performance of T-Net drops significantly if we try to apply the same iterative scheme on its output. This can be explained by the fact that the output of T-Net is on a very different scale than the original input (see Figure 6). As a result, the network sees vastly different inputs from different iterations. This will cause the training to diverge. This issue is resolved with the regularized T-Net. However, its performance is still worse than IT-Net.

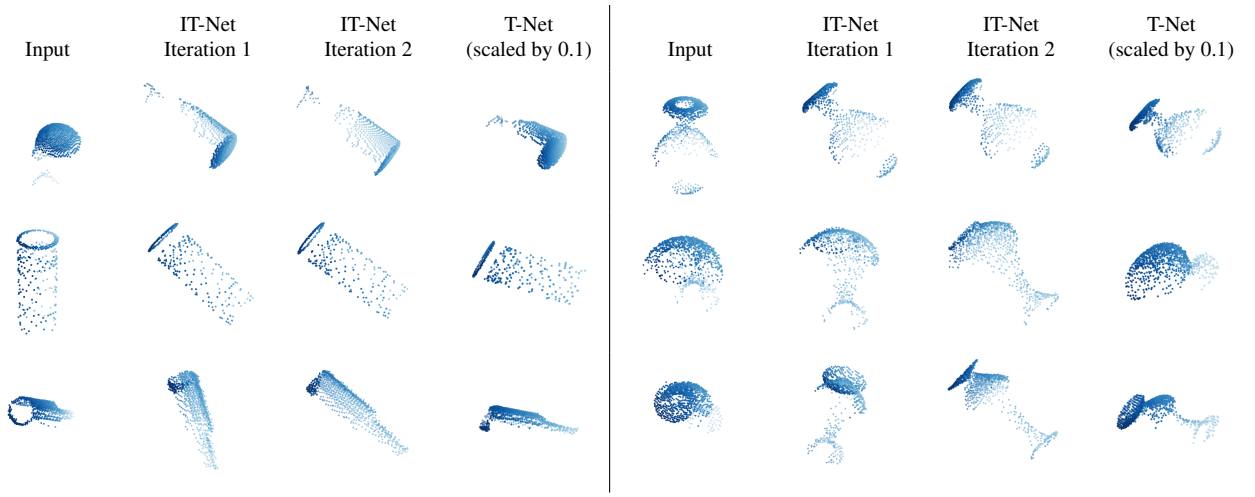


Figure 6: Inputs transformed by IT-Net and T-Net trained jointly with DGCNN. Note how the input pose converges with more iterations and the similarity of final poses across different categories (columns 3, 7). T-Net's outputs are on a much different scale (10 times bigger) than the original inputs (columns 4, 8).

	mean	table	chair	air plane	lamp	car	guitar	laptop	knife	pistol	motor cycle	mug	skate board	bag	ear phone	rocket	cap
# shapes		5271	3758	2690	1547	898	787	451	392	283	202	184	152	76	68	66	55
# parts		3	4	4	4	4	3	2	2	3	6	2	3	2	3	3	2
None	76.9	78.8	82.6	77.3	71.3	52.3	90.1	76.8	80.0	70.1	40.4	**86.1**	67.6	71.0	66.7	53.1	76.9
T-Net	77.1	79.2	82.5	78.0	70.1	55.7	89.1	73.1	81.5	73.0	39.1	81.1	69.1	**74.1**	71.1	51.4	74.6
IT-Net-1	78.2	79.9	84.3	78.2	**72.9**	54.9	91.0	**78.7**	78.1	71.8	**44.6**	84.8	66.6	71.2	**72.7**	**55.0**	77.9
IT-Net-2	**79.1**	**80.2**	**84.7**	**79.9**	72.1	**62.6**	**91.1**	76.4	**82.8**	**76.9**	44.0	84.4	**71.8**	68.1	66.8	54.2	**80.4**

Table 2: Part segmentation results on partial shapes from ShapeNet. The number appending IT-Net indicates the number of iterations. The base segmentation model is DGCNN [22]. The metric is mIoU(%) on points. The mean is calculated as the average of per-category mIoUs weighted by the number of shapes. We order the categories by number of shapes since the performance is more unstable for categories with fewer shapes.

4.2. Partial Object Part Segmentation

To show that the invariance to geometric transformations learnt by IT-Net is not specific to the classification task, we demonstrate IT-Net's capability to improve the performance of part segmentation networks.

Data Similar to ModelNet40, the ShapeNet part dataset [25] commonly used for the evaluation of object part segmentation also consists of complete and aligned point clouds. Thus, similar to partial ModelNet40, we build a more realistic version of the ShapeNet part dataset with partial and unaligned point clouds to evaluate the performance of part segmentation models. Since the ShapeNet part dataset does not come with meshed models, we use an approximate rendering procedure that mimics an orthographic depth camera to create realistic-looking partial point clouds. Please refer to the supplementary for more details.

The dataset contains 16,881 shapes from 16 categories, annotated with 50 parts in total and 2-6 parts per category. We use the same training and testing split provided in [25].

Model The network used for part segmentation simply replaces the classifier in the joint transformer-classifier model from 4.1 with a segmentation network. We use DGCNN [22] as the baseline segmentation network and compare the performance gain of adding T-Net and IT-Net. The architecture of the transformers are identical to the ones in 4.1. Following [16, 22], we treat part segmentation as a per-point classification problem and train the network with per-point cross entropy loss. Similar to 4.1, no explicit supervision is applied on the transformations.

Training The networks are trained for 200 epochs with batch size 32. Other hyperparameters are the same as the ones used in partial shape classification.

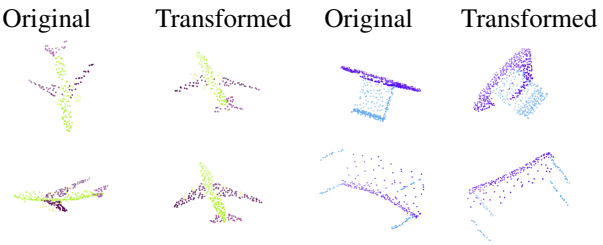


Figure 7: Inputs transformed by IT-Net trained with DGCNN on part segmentation. The colors indicate predictions of the segmentation network.

Results We use the mean Intersection-over-Union (mIoU) on points as the evaluation metric following [16, 22]. The results are summarized in Table 2. Again, IT-Net with 2 iterations leads to the largest amount of improvement in mean mIoU and mIoU for most categories. As in the case of classification, IT-Net removes variations in the inputs caused by geometric transformations (see Figure 7). This demonstrates the potential of IT-Net as a plug-in for any task that requires invariance to geometric transformations without task-specific adjustments to the model architecture.

4.3. Pose Estimation

We have shown that IT-Net outperforms existing transformers on the tasks of partial shape classification and partial object part segmentation. Next, we perform an analysis to try to better understand the efficacy of the iterative refinement scheme on predicting accurate geometric transformations. To this end, we choose the task of estimating the canonical pose of an object from a partial observation.

Specifically, given a partial point cloud, we use IT-Net to estimate the transformation that aligns it to a canonical frame defined across all models in the same category, as illustrated in Figure 2. Unlike most existing works on pose estimation, we do not assume knowledge of the object's model and we train a single network that generalizes to different objects in the same category.

Data Our dataset for pose estimation consists of 12,000 partial point clouds from 2,400 car models in ShapeNet [5]. The point clouds are generated by back-projecting depth scans of models from random viewpoints into the camera's coordinates. Each point cloud is labeled with the transformation that aligns it to the model's coordinates. The data are split into training, validation and testing with a 10:1:1 ratio. Note that the test set and the training set are created using different car models.

Model We use the same transformer architecture as in 4.1 and 4.2, but here an explicit loss is applied on the transformation. We compare IT-Nets trained with up to 6 iterations using the scheme described in 3.2.

Loss For the loss function, we use a variant of PLoss proposed in [24]. It measures the average distance between the same set of points under the estimated pose and the ground truth pose. Compared to the L2 loss used in earlier works [10], this loss has the advantage of automatically handling the tradeoff between small rotations and small translations. The loss can be written as

$$L((R,\mathbf{t}),(\widetilde{R},\tilde{\mathbf{t}})) = \frac{1}{|X|}\sum_{\mathbf{x}\in X}\|(R\mathbf{x}+\mathbf{t})-(\widetilde{R}\mathbf{x}+\tilde{\mathbf{t}})\|_2^2, \quad (1)$$

where $R,\mathbf{t}$ are the ground truth pose and $\widetilde{R},\tilde{\mathbf{t}}$ are the estimated pose. In the original formulation, X is the set of 3D model points. Since we do not assume knowledge of the model, we let X be the set of input points instead.

Training The networks are trained for 20000 steps with batch size 100. We use Adam optimizer with an initial learning rate of 0.001, decayed by 0.7 every 2000 steps. The initial decay rate for batch normalization is 0.5 and gradually increased to 0.99.

Results The results are summarized in Figure 8. From Figure 8b, we can observe that for all but the single iteration network, the pose accuracy keeps increasing even if we apply the network for more iterations than it is trained for. This is another evidence that IT-Net is learning the "gradient transformation" noted in Section 3.2. Moreover, this property allows us to use IT-Net as an any-time pose estimator. In particular, we can train IT-Net for a fixed number of iterations. During inference, we can keep applying the trained IT-Net to obtain increasingly accurate pose estimates until the time budget runs out.

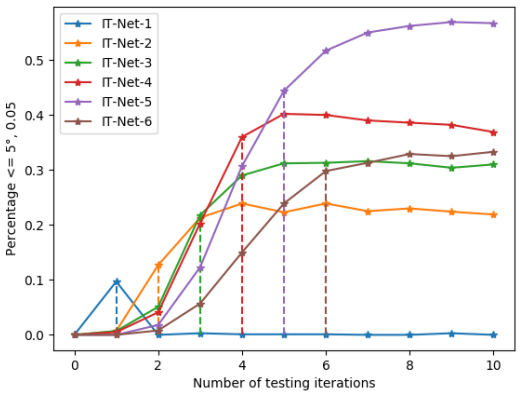
We note that another way to interpret the iterative scheme is to treat it as a form of Dataset Aggregation (DAGGER) method [18] used in imitation learning. From Figure 8c, we can see that there is almost no example with rotation error less than 15 degrees in the initial inputs (iteration 0), but there are many such examples in the inputs to later iterations. In some sense, the network is generating the data it needs to learn more accurate pose estimates. It is even more convenient than DAGGER since there is no need to explicitly add examples to the dataset as the aggregation happens automatically with the training.

5. Conclusion

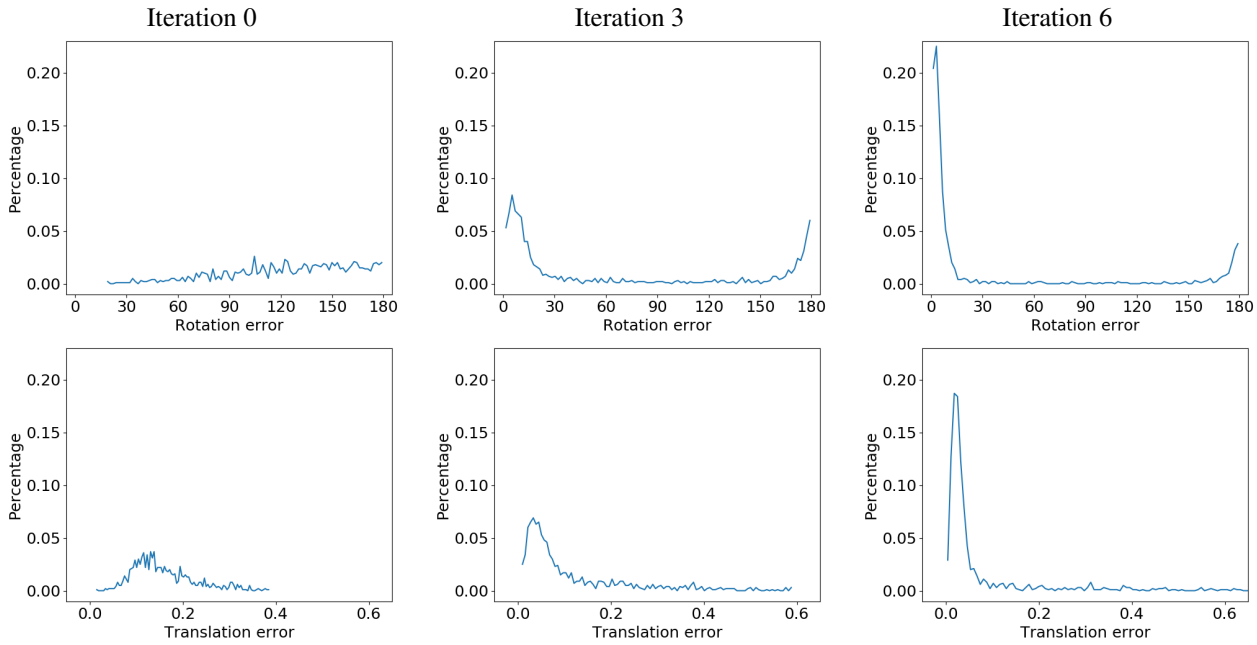
In this work, we propose a new transformer network on 3D point clouds. IT-Net predicts a rigid transformation by iteratively outputting update transformations that are used to transform the input for the next iteration. The advantage of using iterative transformers in various tasks shows that the classical idea of iterative pose refinement still applies in the context of deep learning.

Model	$\leq 5°, 0.05$	$\leq 10°, 0.1$	$\leq 20°, 0.2$
IT-Net-1	9.7	36.9	67.5
IT-Net-2	12.7	41.7	62.6
IT-Net-3	21.8	47.7	58.9
IT-Net-4	36.0	61.1	69.5
IT-Net-5	44.4	67.6	75.4
IT-Net-6	29.8	62.6	74.8

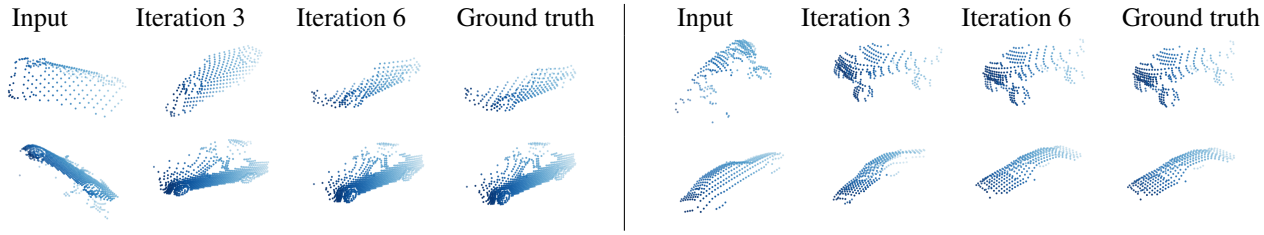
(a) Pose accuracy (%) under different error thresholds



(b) Pose accuracy with more iterations applied during testing



(c) Distribution of rotation and translation errors at different iterations



(d) Qualitative examples

Figure 8: Results on ShapeNet car pose estimation. **(a)** Pose accuracy (percentage of examples below an error threshold) of IT-Nets trained with different number of iterations. The number following the model names indicates the number of iterations used in training. **(b)** Pose accuracy against the number of iterations applied in testing. The dotted line corresponds to the number of iterations used in training. Note how the accuracy keeps improving even when more iterations are applied than the networks are trained for. **(c)** The distribution of rotation and translation error of all test instances at different iterations. Note how the error distribution is much spikier in later iterations. The peak at 180 degrees for rotation error is caused by symmetries in the car models. **(d)** Qualitative examples.

Our transformer can be easily integrated with existing deep learning architectures for shape classification and segmentation, and improve the performance on these tasks with partial, unaligned inputs by introducing invariance to geometric transformations. This opens up many avenues for future research on using neural networks to extract semantic information from real world point cloud data.

Acknowledgements

This project is supported by Carnegie Mellon University's Mobility21 National University Transportation Center, which is sponsored by the US Department of Transportation. We would like to thank Brian Okorn and Chen-Hsuan Lin for their helpful comments and suggestions.

References

[1] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1626–1633. IEEE, 2011. 2

[2] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004. 3, 4

[3] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 2, 3

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2018. 10

[5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 7, 10

[6] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010. 1

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012. 1

[8] 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*, pages 770–778, 2016. 1, 4

[9] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 2, 3

[10] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 3, 7

[11] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007. 1

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[13] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn. *arXiv preprint arXiv:1801.07791*, 2018. 2

[14] C.-H. Lin and S. Lucey. Inverse compositional spatial transformer networks. *arXiv preprint arXiv:1612.03897*, 2016. 3

[15] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981. 2, 3

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 1, 2, 3, 5, 6, 7, 10, 11

[17] 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*, pages 5099–5108, 2017. 2

[18] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. 7

[19] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217, 2009. 2

[20] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351*, 2016. 10

[21] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009. 2

[22] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 2, 5, 6, 7, 10

[23] 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*, pages 1912–1920, 2015. 1, 3, 5

[24] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 3, 7

[25] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas, et al. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):210, 2016. 1, 3, 6, 10, 11

[26] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018. 3

Suppelementary

A. Overview

In this document we provide technical details, ablation studies and visualizations in support of our paper. Here is a summary of the contents:

In Section B, we describe the generation of partial point clouds in our dataset. Section C contains additional results on part segmentation and Section D includes more architecture details. Section E provides ablation studies on the network design. Finally, we show visualizations of pose clusters learned by IT-Net in Section F.

B. Data Generation

In this section, we cover details of the generation of partial, unaligned point clouds used in our experiments.

Each partial point cloud is generated by fusing a sequence of depth scans in the initial camera's coordinates. The initial camera's orientation is random and the distance between the camera center and the object center is varied. Subsequent camera positions are generated by rotating the camera around the object center for up to 30 degrees. Compared to the datasets used in [16, 22], which consist of point clouds uniformly sampled from mesh models, our dataset contains much more challenging inputs with various poses, densities and levels of incompleteness (see Figure 9). More statistics and detail parameters are summarized in Table 3.

Task	Classification	Segmentation	Pose estimation
Model source	ModelNet40 [20]	ShapeNet part [25]	ShapeNet [5]
# classes	40	16	1
# train	78,744	12,137	12,000
# test	2,468	1,870	2,000
# scans	1-4	1	1
Scan size	64×64	50×50	128×128
Focal length	57	∞	64
Camera distance to object center	2-4	∞	1-2

Table 3: Statistics and parameters of our partial point cloud dataset for classification, segmentation and pose estimation.

We use Blender [4] to render depth scans of mesh models in ModelNet and ShapeNet, but the ShapeNet part dataset only has point clouds, so we use an approximate rendering procedure that mimics an orthographic depth camera. Specifically, we randomly rotate the complete point cloud and project the points onto a virtual image with pixel size 0.02 in the xy-plane. For each pixel of the virtual image, we keep the point with the smallest z value and discard all other points that project onto the same pixel as they are considered as occluded by the selected point.

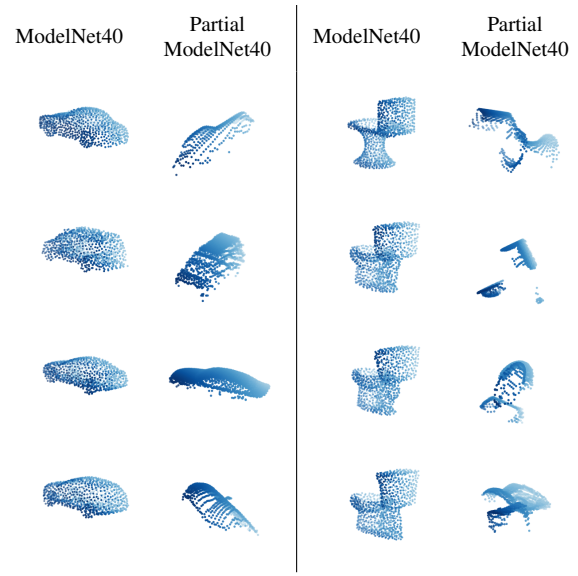


Figure 9: Comparison between ModelNet40 used in [16, 22] and partial ModelNet40 used in our experiments.

We translate the partial point clouds so that their centroids are at the origin. This reduces the range of translation and makes training more stable.

C. Additional Part Segmentation Results

Table 4 shows part segmentation results using PointNet [16] as the base segmentation model. The results are similar to those using DGCNN [22] as the base segmentation model. This is evidence that the advantage of IT-Net is agnostic to the architecture of the segmentation network.

D. Architecture Details

Figure 10 shows the detailed architecture of the transformer networks used in all the experiments. For the classification and segmentation networks, we use publicly available implementations of PointNet [16] and DGCNN [22]. The detailed architectures can be found in Section C of the supplementary for [16] and Section 5.1 and 5.4 of [22].

E. Ablation Studies

In this section, we provide ablation studies for the design decisions mentioned in Section 3.3 of our paper. In particular, we evaluate the effect of initializing the network's prediction with the identity transformation and stopping the gradient during input transformations.

Table 5 and Table 6 show the performance of ablated models on classification and pose estimation respectively.

	mean	table	chair	air plane	lamp	car	guitar	laptop	knife	pistol	motor cycle	mug	skate board	bag	ear phone	rocket	cap
# shapes		5271	3758	2690	1547	898	787	451	392	283	202	184	152	76	68	66	55
# parts		3	4	4	4	4	3	2	2	3	6	2	3	2	3	3	2
None	67.9	71.6	75.2	68.8	56.9	48.2	82.4	58.0	68.5	61.7	39.0	65.6	49.6	41.9	43.5	28.1	**50.9**
T-Net	71.1	73.7	77.5	73.6	60.2	53.0	85.8	**63.2**	73.6	65.4	48.5	70.3	57.7	15.9	41.8	41.7	48.5
IT-Net-1	72.3	74.5	**78.7**	75.9	60.6	**57.7**	85.1	58.3	**78.6**	67.9	**51.5**	70.3	61.6	31.6	**53.9**	35.2	45.3
IT-Net-2	**72.6**	**75.1**	78.3	**76.3**	**62.1**	56.3	**86.8**	58.9	74.5	**68.6**	46.4	**70.6**	**65.9**	**43.5**	51.6	**42.6**	45.9

Table 4: Part segmentation results on partial shapes from ShapeNet part dataset [25]. The number appending IT-Net indicates the number of iterations. The base segmentation model is PointNet [16]. The metric is mIoU(%) on points. The mean is calculated as the average of per-category mIoUs weighted by the number of shapes.

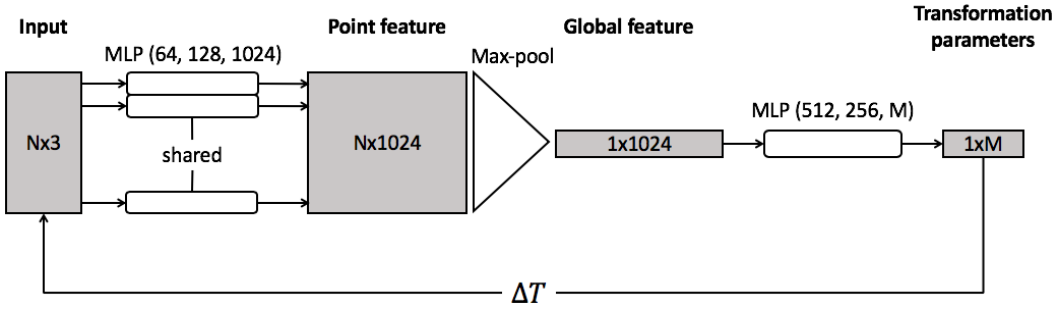


Figure 10: Detailed transformer architecture. Numbers in the parenthesis indicate the number of neurons in each MLP layer. The output dimension M is 7 for IT-Net and 9 for T-Net and T-Net reg.

Classifier	PointNet					
Transformer	IT-Net (no init)		IT-Net (no stop)		IT-Net	
# Iterations	1	2	1	2	1	2
Accuracy (overall)	44.21	3.65	66.73	66.65	68.72	**69.94**
Accuracy (class avg)	36.68	2.55	61.05	61.38	63.42	**65.05**

Table 5: Comparison of ablated models on classification accuracy on partial ModelNet40. No init means the output is *not* initialized as the identity transformation. No stop means the gradient is *not* stopped during input transformations.

As expected, the performance degrades especially for IT-Net trained with larger number of iterations, which indicates that both identity initialization and gradient stopping are crucial for the iterative refinement scheme to achieve desired behavior.

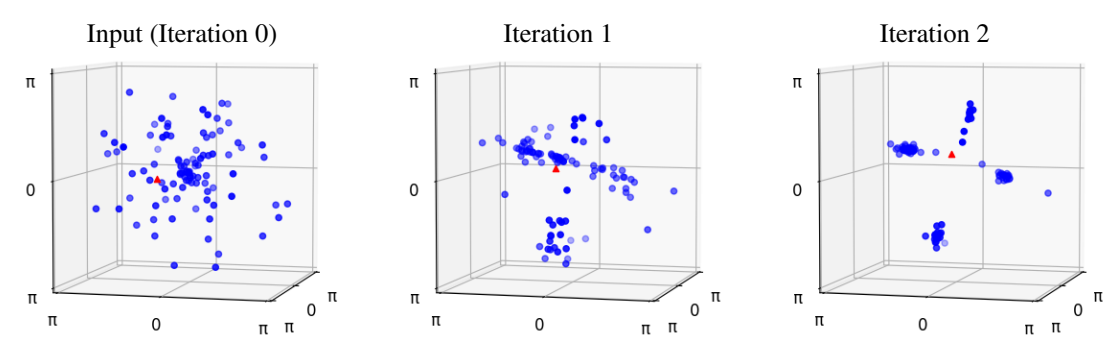
F. Pose Cluster Visualizations

In this section, we visualize the transformations learned by IT-Net. Specifically, we take the IT-Net trained with DGCNN on the classification task and calculate the difference between the canonical orientation of the input shape and the orientation of the transformed input at different iterations. Then, we convert the orientation difference into axis-angle representation, which is a 3D vector, and plot these vectors for all test examples in a particular category.

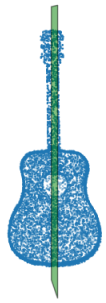
Model	Ablation		
	No init	No stop	-
IT-Net-1	17.1	36.0	36.9
IT-Net-2	0.5	5.0	41.7
IT-Net-3	18.4	0.0	47.7
IT-Net-4	7.3	0.0	61.1
IT-Net-5	6.2	4.1	67.6

Table 6: Comparison of ablated models on pose accuracy with error threshold $10^\circ, 0.1$. No init means the output is *not* initialized as the identity transformation. No stop means the gradient is *not* stopped during input transformations. The number appending IT-Net is the number of iterations.

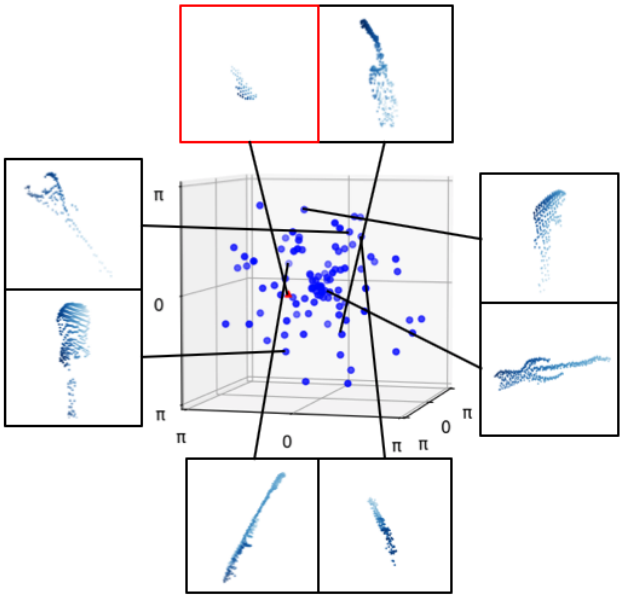
The visualizations for the guitar and bottle category are shown in Figure 11 and Figure 12 respectively. We observe that although the object poses are uniformly distributed initially, clusters of poses emerge after applying the transformations predicted by IT-Net (Figure 11a, 12a). This is evidence that IT-Net discover a canonical space to align the inputs with no explicit supervision. Interestingly, there are usually more than one cluster and the shapes of the clusters are related to the symmetries of the object (Figure 11b, 12b). Further, we note that sometimes even objects across different categories are aligned after being transformed by IT-Net (Figure 11d, 12d). Nevertheless, the pose clusters are less apparent for certain categories where the shapes are nearly spherical, e.g. flower pots.



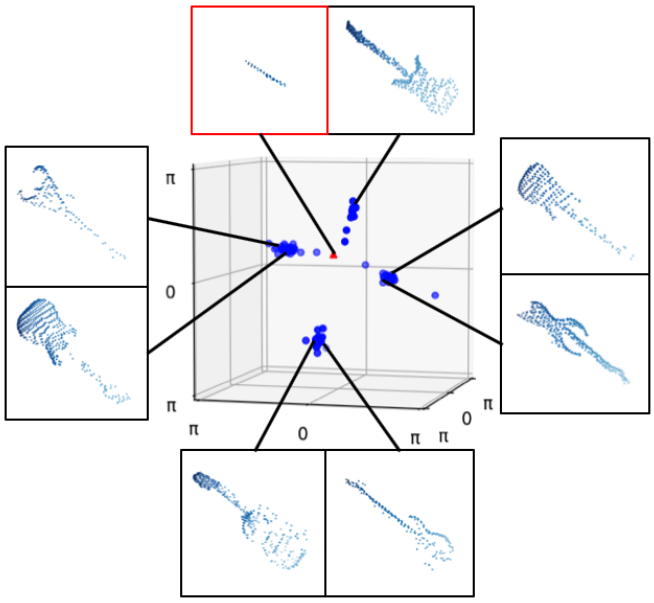
(a) Distribution of orientations at different iterations.



(b) Reflection symmetry of guitars.

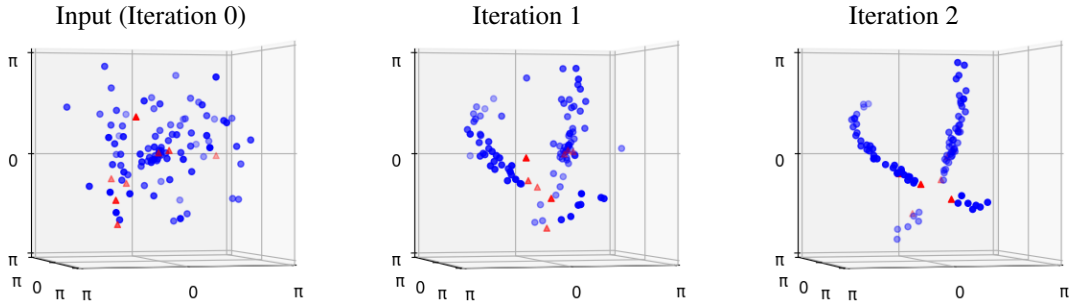


(c) Examples of original inputs (Iteration 0).

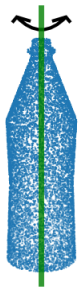


(d) Examples of transformed inputs (Iteration 2).

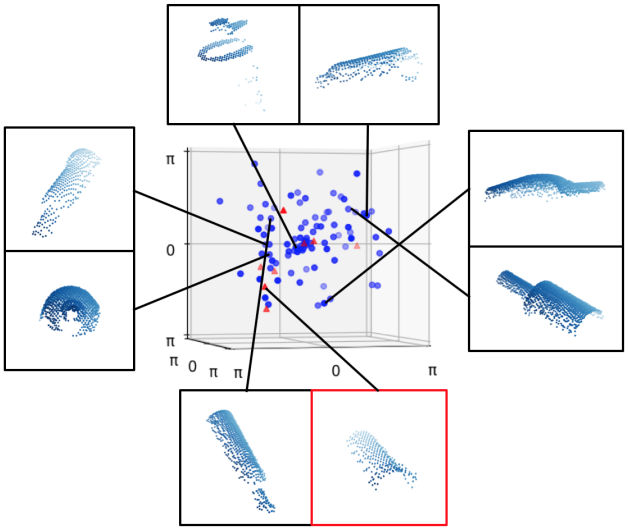
Figure 11: Pose cluster visualization for guitars. **(a)** Distribution of axis-angle representation of orientations of all test examples at different iterations. Note how clusters emerge from uniformly distributed poses. Correctly classified examples are shown in blue and incorrectly classified examples are shown in red. **(b)** The reflection symmetry present in most guitars. **(c)** Examples of original inputs at iteration 0. The object orientations are uniformly distributed. **(d)** Examples of transformed inputs at iteration 2. Note that these are the inputs received by the classifier. The object orientations are grouped into 4 clusters, but visually there seems to be only 2 major orientations due to the reflection symmetry shown in **(b)**. A failure case is shown in the red box. Due to the heavy occlusion the model's failure is expected.



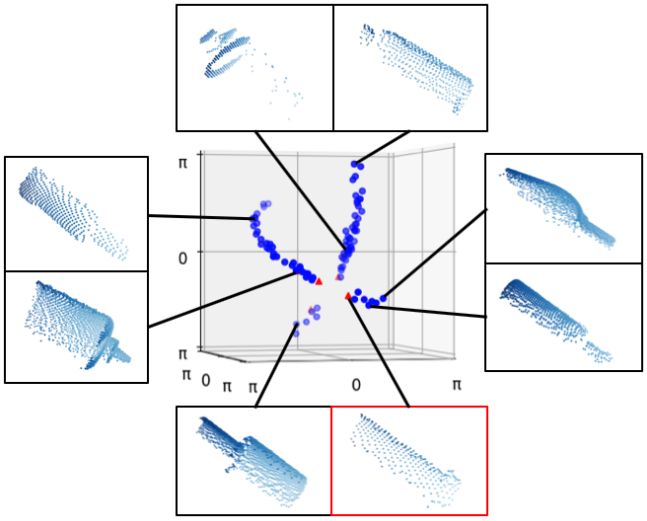
(a) Distribution of orientations at different iterations.



(b) Rotational symmetry of bottles.



(c) Examples of original inputs (Iteration 0).



(d) Examples of transformed inputs (Iteration 2).

Figure 12: Pose cluster visualization for bottles. **(a)** Distribution of axis-angle representation of orientations of all test examples at different iterations. Note how clusters emerge from uniformly distributed poses. Correctly classified examples are shown in blue and incorrectly classified examples are shown in red. **(b)** The rotational symmetry present in most bottles. **(c)** Examples of original inputs at iteration 0. The object orientations are uniformly distributed. **(d)** Examples of transformed inputs at iteration 2. Note that these are the inputs received by the classifier. The object orientations after transformation are grouped into 2 clusters. The clusters have semicircle shapes since any orientation in these semicircles are in fact indistinguishable due to the rotational symmetry shown in **(b)**. A failure case is shown in the red box. In this case the model misclassifies the bottle as a vase.