PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Charles R. Qi* Hao Su* Kaichun Mo Leonidas J. Guibas
Stanford University

Abstract

Point cloud is an important type of geometric data structure. Due to its irregular format, most researchers transform such data to regular 3D voxel grids or collections of images. This, however, renders data unnecessarily voluminous and causes issues. In this paper, we design a novel type of neural network that directly consumes point clouds, which well respects the permutation invariance of points in the input. Our network, named PointNet, provides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing. Though simple, PointNet is highly efficient and effective. Empirically, it shows strong performance on par or even better than state of the art. Theoretically, we provide analysis towards understanding of what the network has learnt and why the network is robust with respect to input perturbation and corruption.

1. Introduction

In this paper we explore deep learning architectures capable of reasoning about 3D geometric data such as point clouds or meshes. Typical convolutional architectures require highly regular input data formats, like those of image grids or 3D voxels, in order to perform weight sharing and other kernel optimizations. Since point clouds or meshes are not in a regular format, most researchers typically transform such data to regular 3D voxel grids or collections of images (e.g, views) before feeding them to a deep net architecture. This data representation transformation, however, renders the resulting data unnecessarily voluminous — while also introducing quantization artifacts that can obscure natural invariances of the data.

For this reason we focus on a different input representation for 3D geometry using simply point clouds – and name our resulting deep nets *PointNets*. Point clouds are simple and unified structures that avoid the combinatorial irregularities and complexities of meshes, and thus are easier to learn from. The PointNet, however, still has to respect the fact that a point cloud is just a set of points and therefore invariant to permutations of its members, necessitating certain symmetrizations in the net computation. Further invariances to rigid motions also need to be considered.

Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

Our PointNet is a unified architecture that directly takes point clouds as input and outputs either class labels for the entire input or per point segment/part labels for each point of the input. The basic architecture of our network is surprisingly simple as in the initial stages each point is processed identically and independently. In the basic setting each point is represented by just its three coordinates (x, y, z). Additional dimensions may be added by computing normals and other local or global features.

Key to our approach is the use of a single symmetric function, max pooling. Effectively the network learns a set of optimization functions/criteria that select interesting or informative points of the point cloud and encode the reason for their selection. The final fully connected layers of the network aggregate these learnt optimal values into the global descriptor for the entire shape as mentioned above (shape classification) or are used to predict per point labels (shape segmentation).

Our input format is easy to apply rigid or affine transformations to, as each point transforms independently. Thus we can add a data-dependent spatial transformer network that attempts to canonicalize the data before the PointNet processes them, so as to further improve the results.

* indicates equal contributions.

PointNet Paper

Nail Ibrahimli

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Charles R. Qi* Hao Su* Kaichun Mo Leonidas J. Guibas
Stanford University

Abstract

Point cloud is an important type of geometric data structure. Due to its irregular format, most researchers transform such data to regular 3D voxel grids or collections of images. This, however, renders data unnecessarily voluminous and causes issues. In this paper, we design a novel type of neural network that directly consumes point clouds, which well respects the permutation invariance of points in the input. Our network, named PointNet, provides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing. Though simple, PointNet is highly efficient and effective. Empirically, it shows strong performance on par or even better than state of the art. Theoretically, we provide analysis towards understanding of what the network has learnt and why the network is robust with respect to input perturbation and corruption.

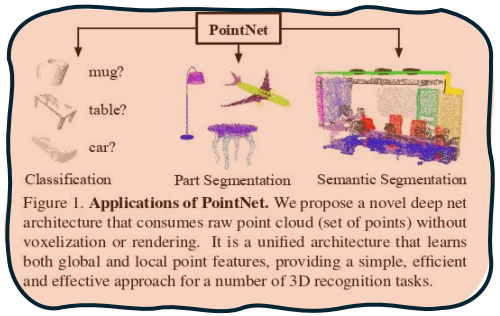


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

1. Introduction

In this paper we explore deep learning architectures capable of reasoning about 3D geometric data such as point clouds or meshes. Typical convolutional architectures require highly regular input data formats, like those of image grids or 3D voxels, in order to perform weight sharing and other kernel optimizations. Since point clouds or meshes are not in a regular format, most researchers typically transform such data to regular 3D voxel grids or collections of images (e.g, views) before feeding them to a deep net architecture. This data representation transformation, however, renders the resulting data unnecessarily voluminous — while also introducing quantization artifacts that can obscure natural invariances of the data.

For this reason we focus on a different input representation for 3D geometry using simply point clouds – and name our resulting deep nets *PointNets*. Point clouds are simple and unified structures that avoid the combinatorial irregularities and complexities of meshes, and thus are easier to learn from. The PointNet, however, still has to respect the fact that a point cloud is just a set of points and therefore invariant to permutations of its members, necessitating certain symmetrizations in the net computation. Further invariances to rigid motions also need to be considered.

Our PointNet is a unified architecture that directly takes point clouds as input and outputs either class labels for the entire input or per point segment/part labels for each point of the input. The basic architecture of our network is surprisingly simple as in the initial stages each point is processed identically and independently. In the basic setting each point is represented by just its three coordinates (x, y, z). Additional dimensions may be added by computing normals and other local or global features.

Key to our approach is the use of a single symmetric function, max pooling. Effectively the network learns a set of optimization functions/criteria that select interesting or informative points of the point cloud and encode the reason for their selection. The final fully connected layers of the network aggregate these learnt optimal values into the global descriptor for the entire shape as mentioned above (shape classification) or are used to predict per point labels (shape segmentation).

Our input format is easy to apply rigid or affine transformations to, as each point transforms independently. Thus we can add a data-dependent spatial transformer network that attempts to canonicalize the data before the PointNet processes them, so as to further improve the results.

* indicates equal contributions.

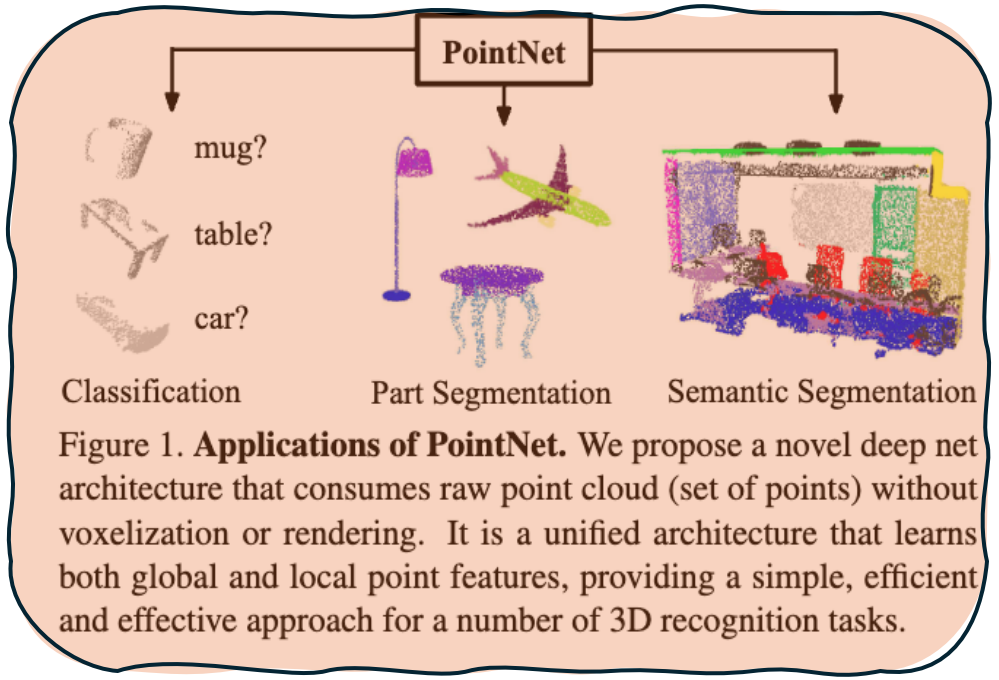


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

We provide both a theoretical analysis and an experimental evaluation of our approach. We show that our network can approximate any set function that is continuous. More interestingly, it turns out that our network learns to summarize an input point cloud by a sparse set of key points, which roughly corresponds to the skeleton of objects according to visualization. The theoretical analysis provides an understanding why our PointNet is highly robust to small perturbation of input points as well as to corruption through point insertion (outliers) or deletion (missing data).

On a number of benchmark datasets ranging from shape classification, part segmentation to scene segmentation, we experimentally compare our PointNet with state-of-the-art approaches based upon multi-view and volumetric representations. Under a unified architecture, not only is our PointNet much faster in speed, but it also exhibits strong performance on par or even better than state of the art.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

The problem of processing unordered sets by neural nets is a very general and fundamental problem – we expect that our ideas can be transferred to other domains as well.

2. Related Work

Point Cloud Features Most existing features for point cloud are handcrafted towards specific tasks. Point features often encode certain statistical properties of points and are designed to be invariant to certain transformations, which are typically classified as intrinsic [2, 24, 3] or extrinsic [20, 19, 14, 10, 5]. They can also be categorized as local features and global features. For a specific task, it is not trivial to find the optimal feature combination.

Deep Learning on 3D Data 3D data has multiple popular representations, leading to various approaches for learning. *Volumetric CNNs:* [28, 17, 18] are the pioneers applying 3D convolutional neural networks on voxelized shapes. However, volumetric representation is constrained by its resolution due to data sparsity and computation cost of 3D convolution. FPNN [13] and Vote3D [26] proposed special methods to deal with the sparsity problem; however, their operations are still on sparse volumes, it's challenging for them to process very large point clouds. *Multiview CNNs:* [23, 18] have tried to render 3D point cloud or shapes into 2D images and then apply 2D conv nets to classify them. With well engineered image CNNs, this line of methods have achieved dominating performance on shape classification and retrieval tasks [21]. However, it's nontrivial to extend them to scene understanding or other 3D tasks such as point classification and shape completion. *Spectral CNNs:* Some latest works [4, 16] use spectral CNNs on meshes. However, these methods are currently constrained on manifold meshes such as organic objects and it's not obvious how to extend them to non-isometric shapes such as furniture. *Feature-based DNNs:* [6, 8] firstly convert the 3D data into a vector, by extracting traditional shape features and then use a fully connected net to classify the shape. We think they are constrained by the representation power of the features extracted.

Deep Learning on Unordered Sets From a data structure point of view, a point cloud is an unordered set of vectors. While most works in deep learning focus on regular input representations like sequences (in speech and language processing), images and volumes (video or 3D data), not much work has been done in deep learning on point sets.

One recent work from Oriol Vinyals et al [25] looks into this problem. They use a read-process-write network with attention mechanism to consume unordered input sets and show that their network has the ability to sort numbers. However, since their work focuses on generic sets and NLP applications, there lacks the role of geometry in the sets.

3. Problem Statement

We design a deep learning framework that directly consumes unordered point sets as inputs. A point cloud is represented as a set of 3D points $\{P_i | \ i = 1, ..., n\}$, where each point P_i is a vector of its (x, y, z) coordinate plus extra feature channels such as color, normal etc. For simplicity and clarity, unless otherwise noted, we only use the (x, y, z) coordinate as our point's channels.

For the object classification task, the input point cloud is either directly sampled from a shape or pre-segmented from a scene point cloud. Our proposed deep network outputs k scores for all the k candidate classes. For semantic segmentation, the input can be a single object for part region segmentation, or a sub-volume from a 3D scene for object region segmentation. Our model will output $n \times m$ scores for each of the n points and each of the m semantic subcategories.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

We provide both a theoretical analysis and an experimental evaluation of our approach. We show that our network can approximate any set function that is continuous. More interestingly, it turns out that our network learns to summarize an input point cloud by a sparse set of key points, which roughly corresponds to the skeleton of objects according to visualization. The theoretical analysis provides an understanding why our PointNet is highly robust to small perturbation of input points as well as to corruption through point insertion (outliers) or deletion (missing data).

On a number of benchmark datasets ranging from shape classification, part segmentation to scene segmentation, we experimentally compare our PointNet with state-of-the-art approaches based upon multi-view and volumetric representations. Under a unified architecture, not only is our PointNet much faster in speed, but it also exhibits strong performance on par or even better than state of the art.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

The problem of processing unordered sets by neural nets is a very general and fundamental problem – we expect that our ideas can be transferred to other domains as well.

2. Related Work

Point Cloud Features Most existing features for point cloud are handcrafted towards specific tasks. Point features often encode certain statistical properties of points and are designed to be invariant to certain transformations, which are typically classified as intrinsic [2, 24, 3] or extrinsic [20, 19, 14, 10, 5]. They can also be categorized as local features and global features. For a specific task, it is not trivial to find the optimal feature combination.

Deep Learning on 3D Data 3D data has multiple popular representations, leading to various approaches for learning. *Volumetric CNNs:* [28, 17, 18] are the pioneers applying 3D convolutional neural networks on voxelized shapes. However, volumetric representation is constrained by its resolution due to data sparsity and computation cost of 3D convolution. FPNN [13] and Vote3D [26] proposed special methods to deal with the sparsity problem; however, their operations are still on sparse volumes, it's challenging for them to process very large point clouds. *Multiview CNNs:* [23, 18] have tried to render 3D point cloud or shapes into 2D images and then apply 2D conv nets to classify them. With well engineered image CNNs, this line of methods have achieved dominating performance on shape classification and retrieval tasks [21]. However, it's nontrivial to extend them to scene understanding or other 3D tasks such as point classification and shape completion. *Spectral CNNs:* Some latest works [4, 16] use spectral CNNs on meshes. However, these methods are currently constrained on manifold meshes such as organic objects and it's not obvious how to extend them to non-isometric shapes such as furniture. *Feature-based DNNs:* [6, 8] firstly convert the 3D data into a vector, by extracting traditional shape features and then use a fully connected net to classify the shape. We think they are constrained by the representation power of the features extracted.

Deep Learning on Unordered Sets From a data structure point of view, a point cloud is an unordered set of vectors. While most works in deep learning focus on regular input representations like sequences (in speech and language processing), images and volumes (video or 3D data), not much work has been done in deep learning on point sets.

One recent work from Oriol Vinyals et al [25] looks into this problem. They use a read-process-write network with attention mechanism to consume unordered input sets and show that their network has the ability to sort numbers. However, since their work focuses on generic sets and NLP applications, there lacks the role of geometry in the sets.

3. Problem Statement

We design a deep learning framework that directly consumes unordered point sets as inputs. A point cloud is represented as a set of 3D points $\{P_i | \ i = 1, ..., n\}$, where each point P_i is a vector of its (x, y, z) coordinate plus extra feature channels such as color, normal etc. For simplicity and clarity, unless otherwise noted, we only use the (x, y, z) coordinate as our point's channels.

For the object classification task, the input point cloud is either directly sampled from a shape or pre-segmented from a scene point cloud. Our proposed deep network outputs k scores for all the k candidate classes. For semantic segmentation, the input can be a single object for part region segmentation, or a sub-volume from a 3D scene for object region segmentation. Our model will output $n \times m$ scores for each of the n points and each of the m semantic subcategories.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

We provide both a theoretical analysis and an experimental evaluation of our approach. We show that our network can approximate any set function that is continuous. More interestingly, it turns out that our network learns to summarize an input point cloud by a sparse set of key points, which roughly corresponds to the skeleton of objects according to visualization. The theoretical analysis provides an understanding why our PointNet is highly robust to small perturbation of input points as well as to corruption through point insertion (outliers) or deletion (missing data).

On a number of benchmark datasets ranging from shape classification, part segmentation to scene segmentation, we experimentally compare our PointNet with state-of-the-art approaches based upon multi-view and volumetric representations. Under a unified architecture, not only is our PointNet much faster in speed, but it also exhibits strong performance on par or even better than state of the art.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

The problem of processing unordered sets by neural nets is a very general and fundamental problem – we expect that our ideas can be transferred to other domains as well.

2. Related Work

Point Cloud Features Most existing features for point cloud are handcrafted towards specific tasks. Point features often encode certain statistical properties of points and are designed to be invariant to certain transformations, which are typically classified as intrinsic [2, 24, 3] or extrinsic [20, 19, 14, 10, 5]. They can also be categorized as local features and global features. For a specific task, it is not trivial to find the optimal feature combination.

Deep Learning on 3D Data 3D data has multiple popular representations, leading to various approaches for learning. *Volumetric CNNs:* [28, 17, 18] are the pioneers applying 3D convolutional neural networks on voxelized shapes. However, volumetric representation is constrained by its resolution due to data sparsity and computation cost of 3D convolution. FPNN [13] and Vote3D [26] proposed special methods to deal with the sparsity problem; however, their operations are still on sparse volumes, it's challenging for them to process very large point clouds. *Multiview CNNs:* [23, 18] have tried to render 3D point cloud or shapes into 2D images and then apply 2D conv nets to classify them. With well engineered image CNNs, this line of methods have achieved dominating performance on shape classification and retrieval tasks [21]. However, it's nontrivial to extend them to scene understanding or other 3D tasks such as point classification and shape completion. *Spectral CNNs:* Some latest works [4, 16] use spectral CNNs on meshes. However, these methods are currently constrained on manifold meshes such as organic objects and it's not obvious how to extend them to non-isometric shapes such as furniture. *Feature-based DNNs:* [6, 8] firstly convert the 3D data into a vector, by extracting traditional shape features and then use a fully connected net to classify the shape. We think they are constrained by the representation power of the features extracted.

Deep Learning on Unordered Sets From a data structure point of view, a point cloud is an unordered set of vectors. While most works in deep learning focus on regular input representations like sequences (in speech and language processing), images and volumes (video or 3D data), not much work has been done in deep learning on point sets.

One recent work from Oriol Vinyals et al [25] looks into this problem. They use a read-process-write network with attention mechanism to consume unordered input sets and show that their network has the ability to sort numbers. However, since their work focuses on generic sets and NLP applications, there lacks the role of geometry in the sets.

3. Problem Statement

We design a deep learning framework that directly consumes unordered point sets as inputs. A point cloud is represented as a set of 3D points $\{P_i | \ i = 1, ..., n\}$, where each point P_i is a vector of its (x, y, z) coordinate plus extra feature channels such as color, normal etc. For simplicity and clarity, unless otherwise noted, we only use the (x, y, z) coordinate as our point's channels.

For the object classification task, the input point cloud is either directly sampled from a shape or pre-segmented from a scene point cloud. Our proposed deep network outputs k scores for all the k candidate classes. For semantic segmentation, the input can be a single object for part region segmentation, or a sub-volume from a 3D scene for object region segmentation. Our model will output $n \times m$ scores for each of the n points and each of the m semantic sub-categories.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

Permutation Invariance:

$$f(x_1, x_2, x_3, \dots, x_n) = y$$
$$f(x_{\pi 1}, x_{\pi 2}, x_{\pi 3}, \dots, x_{\pi n}) = y$$

Permutation Equivariance:

$$f(x_1, x_2, x_3, \dots, x_n) = (y_1, y_2, y_3, \dots, y_n)$$
$$f(x_{\pi 1}, x_{\pi 2}, x_{\pi 3}, \dots, x_{\pi n}) = (y_{\pi 1}, y_{\pi 2}, y_{\pi 3}, \dots, y_{\pi n})$$

We provide both a theoretical analysis and an experimental evaluation of our approach. We show that our network can approximate any set function that is continuous. More interestingly, it turns out that our network learns to summarize an input point cloud by a sparse set of key points, which roughly corresponds to the skeleton of objects according to visualization. The theoretical analysis provides an understanding why our PointNet is highly robust to small perturbation of input points as well as to corruption through point insertion (outliers) or deletion (missing data).

On a number of benchmark datasets ranging from shape classification, part segmentation to scene segmentation, we experimentally compare our PointNet with state-of-the-art approaches based upon multi-view and volumetric representations. Under a unified architecture, not only is our PointNet much faster in speed, but it also exhibits strong performance on par or even better than state of the art.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

The problem of processing unordered sets by neural nets is a very general and fundamental problem – we expect that our ideas can be transferred to other domains as well.

2. Related Work

Point Cloud Features Most existing features for point cloud are handcrafted towards specific tasks. Point features often encode certain statistical properties of points and are designed to be invariant to certain transformations, which are typically classified as intrinsic [2, 24, 3] or extrinsic [20, 19, 14, 10, 5]. They can also be categorized as local features and global features. For a specific task, it is not trivial to find the optimal feature combination.

Deep Learning on 3D Data 3D data has multiple popular representations, leading to various approaches for learning. *Volumetric CNNs:* [28, 17, 18] are the pioneers applying 3D convolutional neural networks on voxelized shapes. However, volumetric representation is constrained by its resolution due to data sparsity and computation cost of 3D convolution. FPNN [13] and Vote3D [26] proposed special methods to deal with the sparsity problem; however, their operations are still on sparse volumes, it's challenging for them to process very large point clouds. *Multiview CNNs:* [23, 18] have tried to render 3D point cloud or shapes into 2D images and then apply 2D conv nets to classify them. With well engineered image CNNs, this line of methods have achieved dominating performance on shape classification and retrieval tasks [21]. However, it's nontrivial to extend them to scene understanding or other 3D tasks such as point classification and shape completion. *Spectral CNNs:* Some latest works [4, 16] use spectral CNNs on meshes. However, these methods are currently constrained on manifold meshes such as organic objects and it's not obvious how to extend them to non-isometric shapes such as furniture. *Feature-based DNNs:* [6, 8] firstly convert the 3D data into a vector, by extracting traditional shape features and then use a fully connected net to classify the shape. We think they are constrained by the representation power of the features extracted.

Deep Learning on Unordered Sets From a data structure point of view, a point cloud is an unordered set of vectors. While most works in deep learning focus on regular input representations like sequences (in speech and language processing), images and volumes (video or 3D data), not much work has been done in deep learning on point sets.

One recent work from Oriol Vinyals et al [25] looks into this problem. They use a read-process-write network with attention mechanism to consume unordered input sets and show that their network has the ability to sort numbers. However, since their work focuses on generic sets and NLP applications, there lacks the role of geometry in the sets.

3. Problem Statement

We design a deep learning framework that directly consumes unordered point sets as inputs. A point cloud is represented as a set of 3D points $\{P_i | \ i = 1, ..., n\}$, where each point P_i is a vector of its (x, y, z) coordinate plus extra feature channels such as color, normal etc. For simplicity and clarity, unless otherwise noted, we only use the (x, y, z) coordinate as our point's channels.

For the object classification task, the input point cloud is either directly sampled from a shape or pre-segmented from a scene point cloud. Our proposed deep network outputs k scores for all the k candidate classes. For semantic segmentation, the input can be a single object for part region segmentation, or a sub-volume from a 3D scene for object region segmentation. Our model will output $n \times m$ scores for each of the n points and each of the m semantic sub-categories.

The key contributions of our work are as follows:

- We design a novel deep net architecture suitable for consuming unordered point sets in 3D;
- We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks;
- We provide thorough empirical and theoretical analysis on the stability and efficiency of our method;
- We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance.

3. Problem Statement

We design a deep learning framework that directly consumes unordered point sets as inputs. A point cloud is represented as a set of 3D points $\{P_i | \ i = 1, ..., n\}$, where each point P_i is a vector of its (x, y, z) coordinate plus extra feature channels such as color, normal etc. For simplicity and clarity, unless otherwise noted, we only use the (x, y, z) coordinate as our point's channels.

For the object classification task, the input point cloud is either directly sampled from a shape or pre-segmented from a scene point cloud. Our proposed deep network outputs k scores for all the k candidate classes. For semantic segmentation, the input can be a single object for part region segmentation, or a sub-volume from a 3D scene for object region segmentation. Our model will output $n \times m$ scores for each of the n points and each of the m semantic sub-categories.

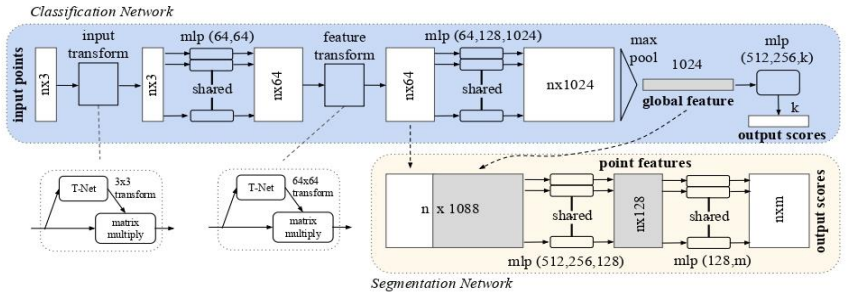


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

4. Deep Learning on Point Sets

The architecture of our network (Sec 4.2) is inspired by the properties of point sets in $\mathbb{R}^n$ (Sec 4.1).

4.1. Properties of Point Sets in $\mathbb{R}^n$

Our input is a subset of points from an Euclidean space. It has three main properties:

- Unordered. Unlike pixel arrays in images or voxel arrays in volumetric grids, point cloud is a set of points without specific order. In other words, a network that consumes N 3D point sets needs to be invariant to $N!$ permutations of the input set in data feeding order.
- Interaction among points. The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures.
- Invariance under transformations. As a geometric object, the learned representation of the point set should be invariant to certain transformations. For example, rotating and translating points all together should not modify the global point cloud category nor the segmentation of the points.

4.2. PointNet Architecture

Our full network architecture is visualized in Fig 2, where the classification network and the segmentation network share a great portion of structures. Please read the caption of Fig 2 for the pipeline.

Our network has three key modules: the max pooling layer as a symmetric function to aggregate information from all the points, a local and global information combination structure, and two joint alignment networks that align both input points and point features.

We will discuss our reason behind these design choices in separate paragraphs below.

Symmetry Function for Unordered Input In order to make a model invariant to input permutation, three strategies exist: 1) sort input into a canonical order; 2) treat the input as a sequence to train an RNN, but augment the training data by all kinds of permutations; 3) use a simple symmetric function to aggregate the information from each point. Here, a symmetric function takes n vectors as input and outputs a new vector that is invariant to the input order. For example, $+$ and $*$ operators are symmetric binary functions.

While sorting sounds like a simple solution, in high dimensional space there in fact does not exist an ordering that is stable w.r.t. point perturbations in the general sense. This can be easily shown by contradiction. If such an ordering strategy exists, it defines a bijection map between a high-dimensional space and a $1d$ real line. It is not hard to see, to require an ordering to be stable w.r.t point perturbations is equivalent to requiring that this map preserves spatial proximity as the dimension reduces, a task that cannot be achieved in the general case. Therefore, sorting does not fully resolve the ordering issue, and it's hard for a network to learn a consistent mapping from input to output as the ordering issue persists. As shown in experiments (Fig 5), we find that applying a MLP directly on the sorted point set performs poorly, though slightly better than directly processing an unsorted input.

The idea to use RNN considers the point set as a sequential signal and hopes that by training the RNN

- Unordered. Unlike pixel arrays in images or voxel arrays in volumetric grids, point cloud is a set of points without specific order. In other words, a network that consumes N 3D point sets needs to be invariant to $N!$ permutations of the input set in data feeding order.
- Interaction among points. The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures.
- Invariance under transformations. As a geometric object, the learned representation of the point set should be invariant to certain transformations. For example, rotating and translating points all together should not modify the global point cloud category nor the segmentation of the points.

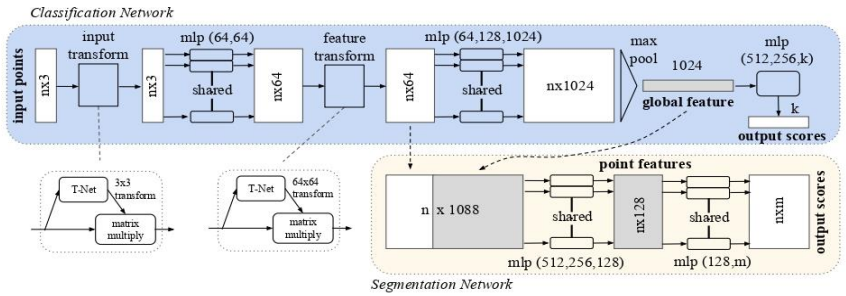


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

4. Deep Learning on Point Sets

The architecture of our network (Sec 4.2) is inspired by the properties of point sets in $\mathbb{R}^n$ (Sec 4.1).

4.1. Properties of Point Sets in $\mathbb{R}^n$

Our input is a subset of points from an Euclidean space. It has three main properties:

- Unordered. Unlike pixel arrays in images or voxel arrays in volumetric grids, point cloud is a set of points without specific order. In other words, a network that consumes N 3D point sets needs to be invariant to $N!$ permutations of the input set in data feeding order.
- Interaction among points. The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures.
- Invariance under transformations. As a geometric object, the learned representation of the point set should be invariant to certain transformations. For example, rotating and translating points all together should not modify the global point cloud category nor the segmentation of the points.

4.2. PointNet Architecture

Our full network architecture is visualized in Fig 2, where the classification network and the segmentation network share a great portion of structures. Please read the caption of Fig 2 for the pipeline.

Our network has three key modules: the max pooling layer as a symmetric function to aggregate information from all the points, a local and global information combination structure, and two joint alignment networks that align both input points and point features.

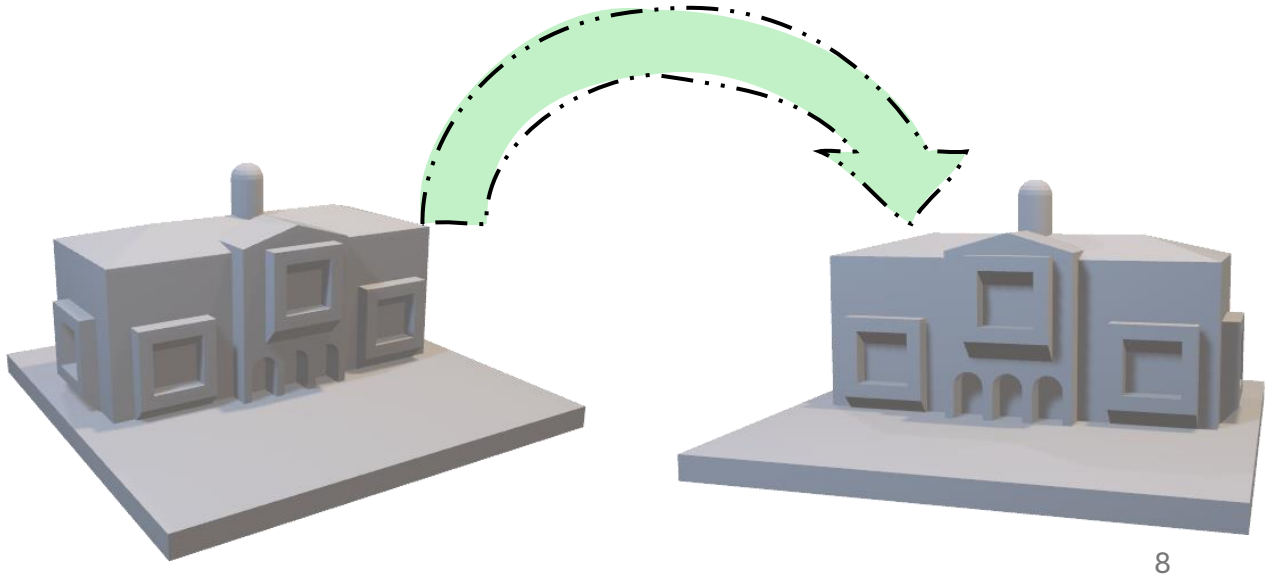
We will discuss our reason behind these design choices in separate paragraphs below.

Symmetry Function for Unordered Input In order to make a model invariant to input permutation, three strategies exist: 1) sort input into a canonical order; 2) treat the input as a sequence to train an RNN, but augment the training data by all kinds of permutations; 3) use a simple symmetric function to aggregate the information from each point. Here, a symmetric function takes n vectors as input and outputs a new vector that is invariant to the input order. For example, $+$ and $*$ operators are symmetric binary functions.

While sorting sounds like a simple solution, in high dimensional space there in fact does not exist an ordering that is stable w.r.t. point perturbations in the general sense. This can be easily shown by contradiction. If such an ordering strategy exists, it defines a bijection map between a high-dimensional space and a $1d$ real line. It is not hard to see, to require an ordering to be stable w.r.t point perturbations is equivalent to requiring that this map preserves spatial proximity as the dimension reduces, a task that cannot be achieved in the general case. Therefore, sorting does not fully resolve the ordering issue, and it's hard for a network to learn a consistent mapping from input to output as the ordering issue persists. As shown in experiments (Fig 5), we find that applying a MLP directly on the sorted point set performs poorly, though slightly better than directly processing an unsorted input.

The idea to use RNN considers the point set as a sequential signal and hopes that by training the RNN

- Unordered. Unlike pixel arrays in images or voxel arrays in volumetric grids, point cloud is a set of points without specific order. In other words, a network that consumes N 3D point sets needs to be invariant to $N!$ permutations of the input set in data feeding order.
- Interaction among points. The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures.
- Invariance under transformations. As a geometric object, the learned representation of the point set should be invariant to certain transformations. For example, rotating and translating points all together should not modify the global point cloud category nor the segmentation of the points.



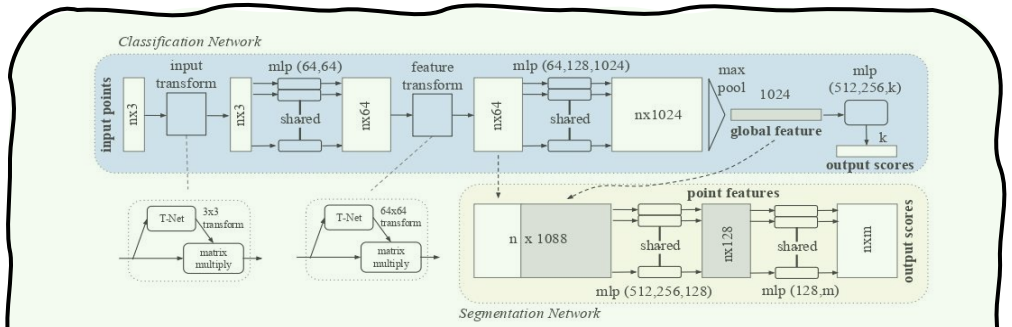


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

4. Deep Learning on Point Sets

The architecture of our network (Sec 4.2) is inspired by the properties of point sets in $\mathbb{R}^n$ (Sec 4.1).

4.1. Properties of Point Sets in $\mathbb{R}^n$

Our input is a subset of points from an Euclidean space. It has three main properties:

- Unordered. Unlike pixel arrays in images or voxel arrays in volumetric grids, point cloud is a set of points without specific order. In other words, a network that consumes N 3D point sets needs to be invariant to $N!$ permutations of the input set in data feeding order.
- Interaction among points. The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures.
- Invariance under transformations. As a geometric object, the learned representation of the point set should be invariant to certain transformations. For example, rotating and translating points all together should not modify the global point cloud category nor the segmentation of the points.

4.2. PointNet Architecture

Our full network architecture is visualized in Fig 2, where the classification network and the segmentation network share a great portion of structures. Please read the caption of Fig 2 for the pipeline.

Our network has three key modules: the max pooling layer as a symmetric function to aggregate information from all the points, a local and global information combination structure, and two joint alignment networks that align both input points and point features.

We will discuss our reason behind these design choices in separate paragraphs below.

Symmetry Function for Unordered Input In order to make a model invariant to input permutation, three strategies exist: 1) sort input into a canonical order; 2) treat the input as a sequence to train an RNN, but augment the training data by all kinds of permutations; 3) use a simple symmetric function to aggregate the information from each point. Here, a symmetric function takes n vectors as input and outputs a new vector that is invariant to the input order. For example, $+$ and $*$ operators are symmetric binary functions.

While sorting sounds like a simple solution, in high dimensional space there in fact does not exist an ordering that is stable w.r.t. point perturbations in the general sense. This can be easily shown by contradiction. If such an ordering strategy exists, it defines a bijection map between a high-dimensional space and a $1d$ real line. It is not hard to see, to require an ordering to be stable w.r.t point perturbations is equivalent to requiring that this map preserves spatial proximity as the dimension reduces, a task that cannot be achieved in the general case. Therefore, sorting does not fully resolve the ordering issue, and it's hard for a network to learn a consistent mapping from input to output as the ordering issue persists. As shown in experiments (Fig 5), we find that applying a MLP directly on the sorted point set performs poorly, though slightly better than directly processing an unsorted input.

The idea to use RNN considers the point set as a sequential signal and hopes that by training the RNN

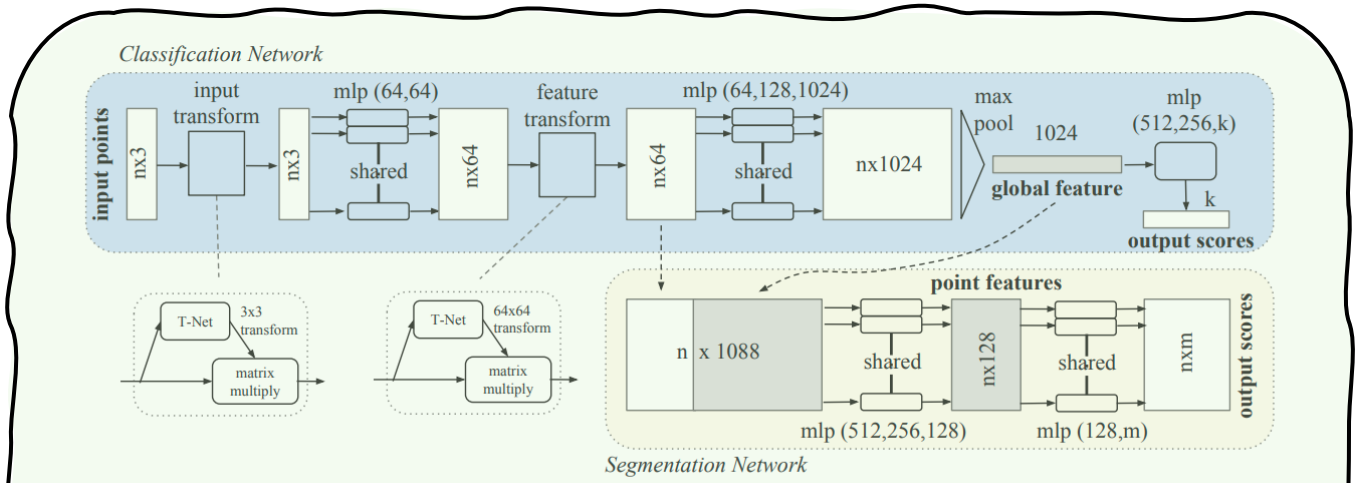


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

with randomly permuted sequences, the RNN will become invariant to input order. However in "OrderMatters" [25] the authors have shown that order does matter and cannot be totally omitted. While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets. Empirically, we have also shown that model based on RNN does not perform as well as our proposed method (Fig 5).

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

While our key module seems simple, it has interesting properties (see Sec 5.3) and can achieve strong performace (see Sec 5.1) in a few different applications. Due to the simplicity of our module, we are also able to provide theoretical analysis as in Sec 4.3.

Local and Global Information Aggregation The output from the above section forms a vector $[f_1, \ldots, f_K]$, which is a global signature of the input set. We can easily train a SVM or multi-layer perceptron classifier on the shape global features for classification. However, point segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner.

Our solution can be seen in Fig 2 (*Segmentation Network*). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

With this modification our network is able to predict per point quantities that rely on both local geometry and global semantics. For example we can accurately predict per-point normals (fig in supplementary), validating that the network is able to summarize information from the point's local neighborhood. In experiment session, we also show that our model can achieve state-of-the-art performance on shape part segmentation and scene segmentation.

Joint Alignment Network The semantic labeling of a point cloud has to be invariant if the point cloud undergoes certain geometric transformations, such as rigid transformation. We therefore expect that the learnt representation by our point set is invariant to these transformations.

A natural solution is to align all input set to a canonical space before feature extraction. Jaderberg et al. [9] introduces the idea of spatial transformer to align 2D images through sampling and interpolation, achieved by a specifically tailored layer implemented on GPU.

Our input form of point clouds allows us to achieve this goal in a much simpler way compared with [9]. We do not need to invent any new layers and no alias is introduced as in the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about the T-net are in the supplementary.

This idea can be further extended to the alignment of feature space, as well. We can insert another alignment network on point features and predict a feature transformation matrix to align features from different input point clouds. However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a mini-network. An orthogonal transformation will not lose information in the input, thus is desired. We find that by adding the regularization term, the optimization becomes more stable and our model achieves better performance.

4.3. Theoretical Analysis

Universal approximation We first show the universal approximation ability of our neural network to continuous set functions. By the continuity of set functions, intuitively, a small perturbation to the input point set should not greatly change the function values, such as classification or segmentation scores.

Formally, let $\mathcal{X} = \{S : S \subseteq [0,1]^m \text{ and } |S| = n\}$, $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function on $\mathcal{X}$ w.r.t to Hausdorff distance $d_H(\cdot,\cdot)$, i.e., $\forall \epsilon > 0, \exists \delta > 0$, for any $S, S' \in \mathcal{X}$, if $d_H(S, S') < \delta$, then $|f(S) - f(S')| < \epsilon$. Our theorem says that f can be arbitrarily approximated by our network given enough neurons at the max pooling layer, i.e., K in (1) is sufficiently large.

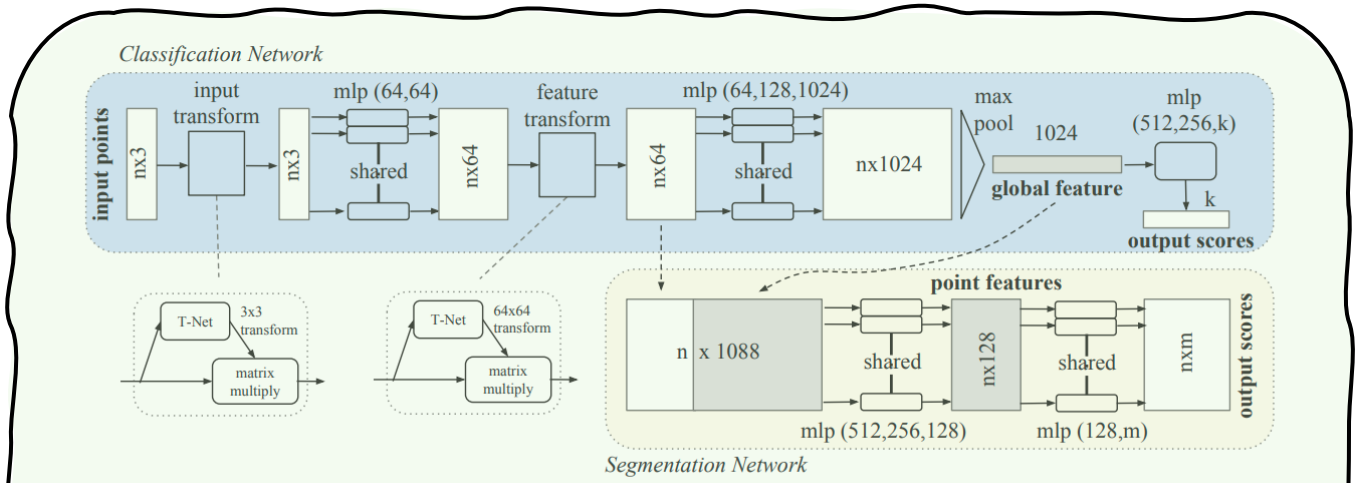


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

with randomly permuted sequences, the RNN will become invariant to input order. However in "OrderMatters" [25] the authors have shown that order does matter and cannot be totally omitted. While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets. Empirically, we have also shown that model based on RNN does not perform as well as our proposed method (Fig 5).

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

While our key module seems simple, it has interesting properties (see Sec 5.3) and can achieve strong performace (see Sec 5.1) in a few different applications. Due to the simplicity of our module, we are also able to provide theoretical analysis as in Sec 4.3.

Local and Global Information Aggregation The output from the above section forms a vector $[f_1, \ldots, f_K]$, which is a global signature of the input set. We can easily train a SVM or multi-layer perceptron classifier on the shape global features for classification. However, point segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner.

Our solution can be seen in Fig 2 (*Segmentation Network*). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

With this modification our network is able to predict per point quantities that rely on both local geometry and global semantics. For example we can accurately predict per-point normals (fig in supplementary), validating that the network is able to summarize information from the point's local neighborhood. In experiment session, we also show that our model can achieve state-of-the-art performance on shape part segmentation and scene segmentation.

Joint Alignment Network The semantic labeling of a point cloud has to be invariant if the point cloud undergoes certain geometric transformations, such as rigid transformation. We therefore expect that the learnt representation by our point set is invariant to these transformations.

A natural solution is to align all input set to a canonical space before feature extraction. Jaderberg et al. [9] introduces the idea of spatial transformer to align 2D images through sampling and interpolation, achieved by a specifically tailored layer implemented on GPU.

Our input form of point clouds allows us to achieve this goal in a much simpler way compared with [9]. We do not need to invent any new layers and no alias is introduced as in the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about the T-net are in the supplementary.

This idea can be further extended to the alignment of feature space, as well. We can insert another alignment network on point features and predict a feature transformation matrix to align features from different input point clouds. However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a mini-network. An orthogonal transformation will not lose information in the input, thus is desired. We find that by adding the regularization term, the optimization becomes more stable and our model achieves better performance.

4.3. Theoretical Analysis

Universal approximation We first show the universal approximation ability of our neural network to continuous set functions. By the continuity of set functions, intuitively, a small perturbation to the input point set should not greatly change the function values, such as classification or segmentation scores.

Formally, let $\mathcal{X} = \{S : S \subseteq [0,1]^m \text{ and } |S| = n\}$, $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function on $\mathcal{X}$ w.r.t to Hausdorff distance $d_H(\cdot,\cdot)$, i.e., $\forall \epsilon > 0, \exists \delta > 0$, for any $S, S' \in \mathcal{X}$, if $d_H(S, S') < \delta$, then $|f(S) - f(S')| < \epsilon$. Our theorem says that f can be arbitrarily approximated by our network given enough neurons at the max pooling layer, i.e., K in (1) is sufficiently large.

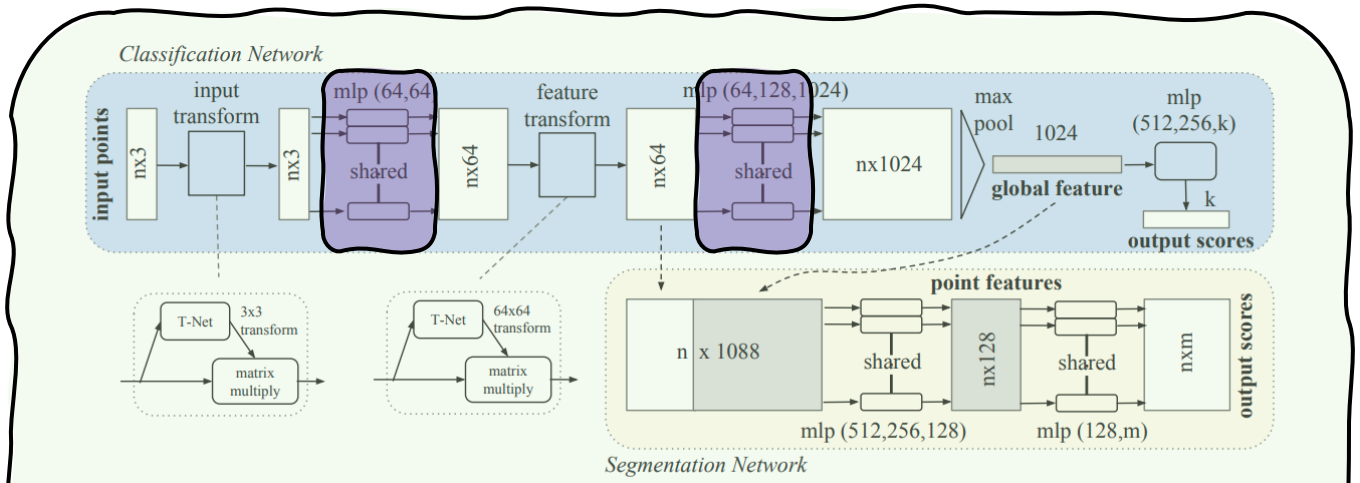


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

with randomly permuted sequences, the RNN will become invariant to input order. However in "OrderMatters" [25] the authors have shown that order does matter and cannot be totally omitted. While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets. Empirically, we have also shown that model based on RNN does not perform as well as our proposed method (Fig 5).

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

While our key module seems simple, it has interesting properties (see Sec 5.3) and can achieve strong performace (see Sec 5.1) in a few different applications. Due to the simplicity of our module, we are also able to provide theoretical analysis as in Sec 4.3.

Local and Global Information Aggregation The output from the above section forms a vector $[f_1, \ldots, f_K]$, which is a global signature of the input set. We can easily train a SVM or multi-layer perceptron classifier on the shape global features for classification. However, point segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner.

Our solution can be seen in Fig 2 (*Segmentation Network*). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

With this modification our network is able to predict per point quantities that rely on both local geometry and global semantics. For example we can accurately predict per-point normals (fig in supplementary), validating that the network is able to summarize information from the point's local neighborhood. In experiment session, we also show that our model can achieve state-of-the-art performance on shape part segmentation and scene segmentation.

Joint Alignment Network The semantic labeling of a point cloud has to be invariant if the point cloud undergoes certain geometric transformations, such as rigid transformation. We therefore expect that the learnt representation by our point set is invariant to these transformations.

A natural solution is to align all input set to a canonical space before feature extraction. Jaderberg et al. [9] introduces the idea of spatial transformer to align 2D images through sampling and interpolation, achieved by a specifically tailored layer implemented on GPU.

Our input form of point clouds allows us to achieve this goal in a much simpler way compared with [9]. We do not need to invent any new layers and no alias is introduced as in the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about the T-net are in the supplementary.

This idea can be further extended to the alignment of feature space, as well. We can insert another alignment network on point features and predict a feature transformation matrix to align features from different input point clouds. However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a mini-network. An orthogonal transformation will not lose information in the input, thus is desired. We find that by adding the regularization term, the optimization becomes more stable and our model achieves better performance.

4.3. Theoretical Analysis

Universal approximation We first show the universal approximation ability of our neural network to continuous set functions. By the continuity of set functions, intuitively, a small perturbation to the input point set should not greatly change the function values, such as classification or segmentation scores.

Formally, let $\mathcal{X} = \{S : S \subseteq [0,1]^m \text{ and } |S| = n\}$, $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function on $\mathcal{X}$ w.r.t to Hausdorff distance $d_H(\cdot,\cdot)$, i.e., $\forall \epsilon > 0, \exists \delta > 0$, for any $S, S' \in \mathcal{X}$, if $d_H(S, S') < \delta$, then $|f(S) - f(S')| < \epsilon$. Our theorem says that f can be arbitrarily approximated by our network given enough neurons at the max pooling layer, i.e., K in (1) is sufficiently large.

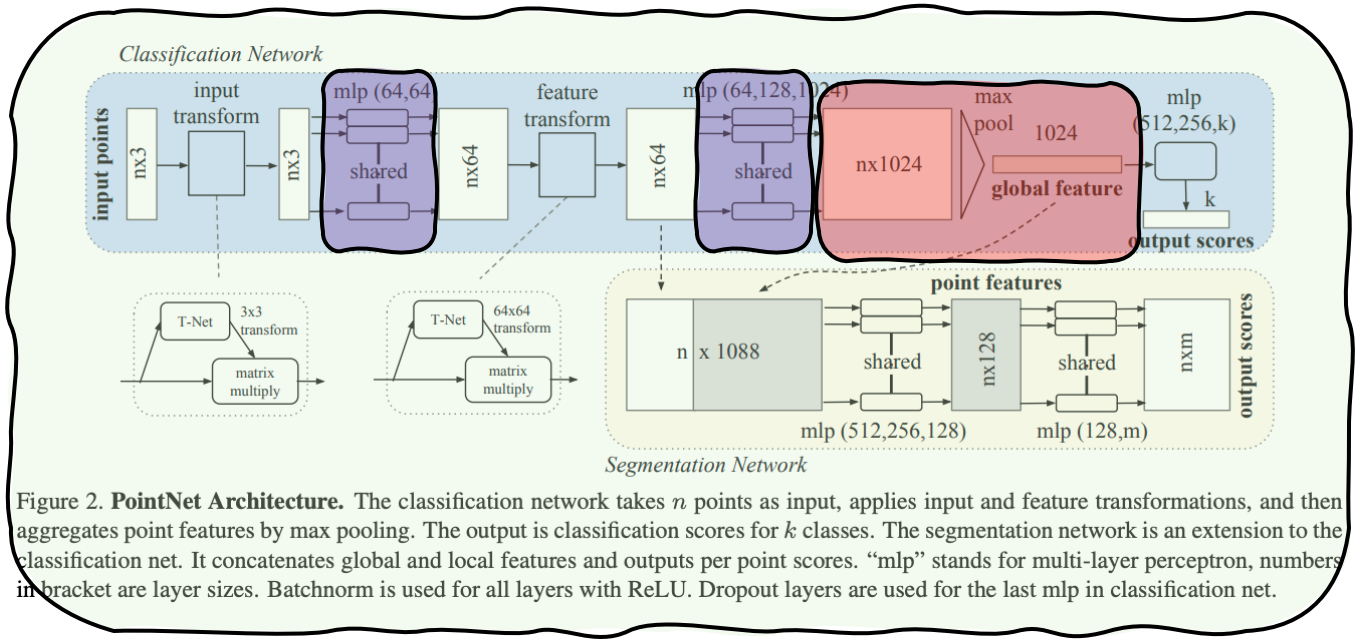


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

with randomly permuted sequences, the RNN will become invariant to input order. However in "OrderMatters" [25] the authors have shown that order does matter and cannot be totally omitted. While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets. Empirically, we have also shown that model based on RNN does not perform as well as our proposed method (Fig 5).

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

While our key module seems simple, it has interesting properties (see Sec 5.3) and can achieve strong performace (see Sec 5.1) in a few different applications. Due to the simplicity of our module, we are also able to provide theoretical analysis as in Sec 4.3.

Local and Global Information Aggregation The output from the above section forms a vector $[f_1, \ldots, f_K]$, which is a global signature of the input set. We can easily train a SVM or multi-layer perceptron classifier on the shape global features for classification. However, point segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner.

Our solution can be seen in Fig 2 (*Segmentation Network*). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

With this modification our network is able to predict per point quantities that rely on both local geometry and global semantics. For example we can accurately predict per-point normals (fig in supplementary), validating that the network is able to summarize information from the point's local neighborhood. In experiment session, we also show that our model can achieve state-of-the-art performance on shape part segmentation and scene segmentation.

Joint Alignment Network The semantic labeling of a point cloud has to be invariant if the point cloud undergoes certain geometric transformations, such as rigid transformation. We therefore expect that the learnt representation by our point set is invariant to these transformations.

A natural solution is to align all input set to a canonical space before feature extraction. Jaderberg et al. [9] introduces the idea of spatial transformer to align 2D images through sampling and interpolation, achieved by a specifically tailored layer implemented on GPU.

Our input form of point clouds allows us to achieve this goal in a much simpler way compared with [9]. We do not need to invent any new layers and no alias is introduced as in the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about the T-net are in the supplementary.

This idea can be further extended to the alignment of feature space, as well. We can insert another alignment network on point features and predict a feature transformation matrix to align features from different input point clouds. However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a mini-network. An orthogonal transformation will not lose information in the input, thus is desired. We find that by adding the regularization term, the optimization becomes more stable and our model achieves better performance.

4.3. Theoretical Analysis

Universal approximation We first show the universal approximation ability of our neural network to continuous set functions. By the continuity of set functions, intuitively, a small perturbation to the input point set should not greatly change the function values, such as classification or segmentation scores.

Formally, let $\mathcal{X} = \{S : S \subseteq [0,1]^m \text{ and } |S| = n\}$, $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function on $\mathcal{X}$ w.r.t to Hausdorff distance $d_H(\cdot,\cdot)$, i.e., $\forall \epsilon > 0, \exists \delta > 0$, for any $S, S' \in \mathcal{X}$, if $d_H(S, S') < \delta$, then $|f(S) - f(S')| < \epsilon$. Our theorem says that f can be arbitrarily approximated by our network given enough neurons at the max pooling layer, i.e., K in (1) is sufficiently large.

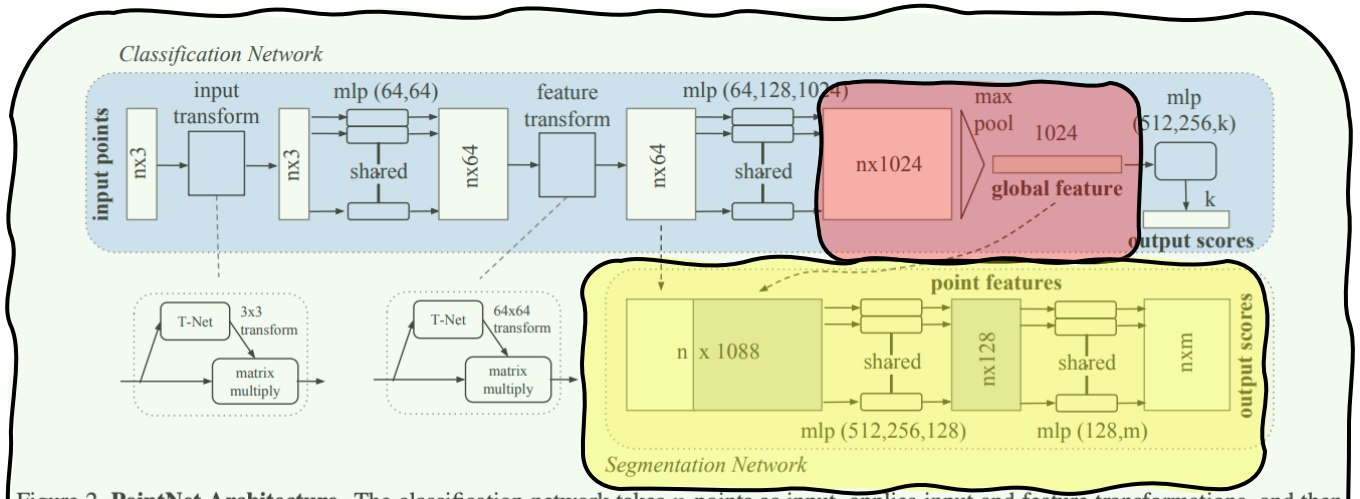


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner.

Our solution can be seen in Fig 2 (*Segmentation Network*). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

with randomly permuted sequences, the RNN will become invariant to input order. However in "OrderMatters" [25] the authors have shown that order does matter and cannot be totally omitted. While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets. Empirically, we have also shown that model based on RNN does not perform as well as our proposed method (Fig 5).

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

While our key module seems simple, it has interesting properties (see Sec 5.3) and can achieve strong performace (see Sec 5.1) in a few different applications. Due to the simplicity of our module, we are also able to provide theoretical analysis as in Sec 4.3.

Local and Global Information Aggregation The output from the above section forms a vector $[f_1, \ldots, f_K]$, which is a global signature of the input set. We can easily train a SVM or multi-layer perceptron classifier on the shape global features for classification. However, point segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner.

Our solution can be seen in Fig 2 (*Segmentation Network*). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

With this modification our network is able to predict per point quantities that rely on both local geometry and global semantics. For example we can accurately predict per-point normals (fig in supplementary), validating that the network is able to summarize information from the point's local neighborhood. In experiment session, we also show that our model can achieve state-of-the-art performance on shape part segmentation and scene segmentation.

Joint Alignment Network The semantic labeling of a point cloud has to be invariant if the point cloud undergoes certain geometric transformations, such as rigid transformation. We therefore expect that the learnt representation by our point set is invariant to these transformations.

A natural solution is to align all input set to a canonical space before feature extraction. Jaderberg et al. [9] introduces the idea of spatial transformer to align 2D images through sampling and interpolation, achieved by a specifically tailored layer implemented on GPU.

Our input form of point clouds allows us to achieve this goal in a much simpler way compared with [9]. We do not need to invent any new layers and no alias is introduced as in the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about the T-net are in the supplementary.

This idea can be further extended to the alignment of feature space, as well. We can insert another alignment network on point features and predict a feature transformation matrix to align features from different input point clouds. However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a mini-network. An orthogonal transformation will not lose information in the input, thus is desired. We find that by adding the regularization term, the optimization becomes more stable and our model achieves better performance.

4.3. Theoretical Analysis

Universal approximation We first show the universal approximation ability of our neural network to continuous set functions. By the continuity of set functions, intuitively, a small perturbation to the input point set should not greatly change the function values, such as classification or segmentation scores.

Formally, let $\mathcal{X} = \{S : S \subseteq [0,1]^m \text{ and } |S| = n\}$, $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function on $\mathcal{X}$ w.r.t to Hausdorff distance $d_H(\cdot,\cdot)$, i.e., $\forall \epsilon > 0, \exists \delta > 0$, for any $S, S' \in \mathcal{X}$, if $d_H(S, S') < \delta$, then $|f(S) - f(S')| < \epsilon$. Our theorem says that f can be arbitrarily approximated by our network given enough neurons at the max pooling layer, i.e., K in (1) is sufficiently large.

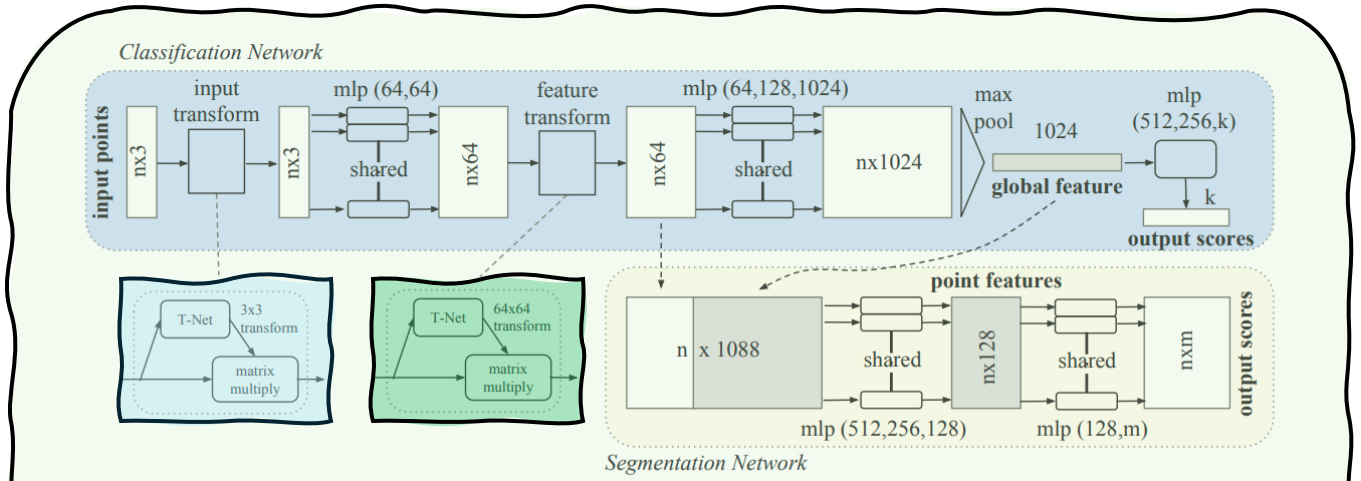


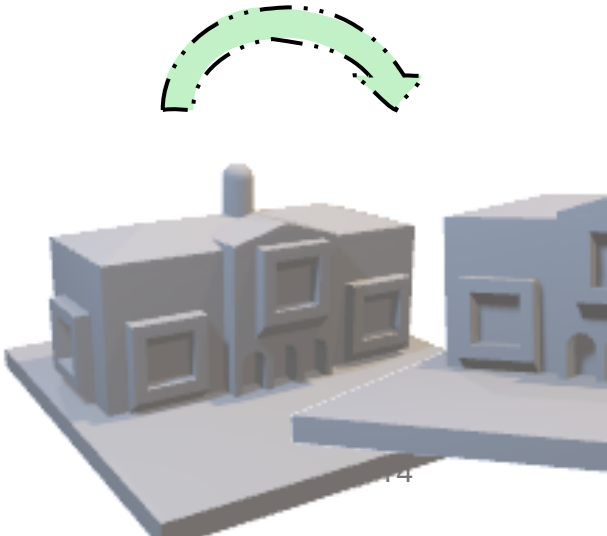
Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about

However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a



with randomly permuted sequences, the RNN will become invariant to input order. However in "OrderMatters" [25] the authors have shown that order does matter and cannot be totally omitted. While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets. Empirically, we have also shown that model based on RNN does not perform as well as our proposed method (Fig 5).

Our idea is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(\{x_1, \ldots, x_n\}) \approx g(h(x_1), \ldots, h(x_n)), \qquad (1)$$

where $f : 2^{\mathbb{R}^N} \to \mathbb{R}$, $h : \mathbb{R}^N \to \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \to \mathbb{R}$ is a symmetric function.

Empirically, our basic module is very simple: we approximate h by a multi-layer perceptron network and g by a composition of a single variable function and a max pooling function. This is found to work well by experiments. Through a collection of h, we can learn a number of f's to capture different properties of the set.

While our key module seems simple, it has interesting properties (see Sec 5.3) and can achieve strong performace (see Sec 5.1) in a few different applications. Due to the simplicity of our module, we are also able to provide theoretical analysis as in Sec 4.3.

Local and Global Information Aggregation The output from the above section forms a vector $[f_1, \ldots, f_K]$, which is a global signature of the input set. We can easily train a SVM or multi-layer perceptron classifier on the shape global features for classification. However, point segmentation requires a combination of local and global knowledge. We can achieve this by a simple yet highly effective manner.

Our solution can be seen in Fig 2 (*Segmentation Network*). After computing the global point cloud feature vector, we feed it back to per point features by concatenating the global feature with each of the point features. Then we extract new per point features based on the combined point features - this time the per point feature is aware of both the local and global information.

With this modification our network is able to predict per point quantities that rely on both local geometry and global semantics. For example we can accurately predict per-point normals (fig in supplementary), validating that the network is able to summarize information from the point's local neighborhood. In experiment session, we also show that our model can achieve state-of-the-art performance on shape part segmentation and scene segmentation.

Joint Alignment Network The semantic labeling of a point cloud has to be invariant if the point cloud undergoes certain geometric transformations, such as rigid transformation. We therefore expect that the learnt representation by our point set is invariant to these transformations.

A natural solution is to align all input set to a canonical space before feature extraction. Jaderberg et al. [9] introduces the idea of spatial transformer to align 2D images through sampling and interpolation, achieved by a specifically tailored layer implemented on GPU.

Our input form of point clouds allows us to achieve this goal in a much simpler way compared with [9]. We do not need to invent any new layers and no alias is introduced as in the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about the T-net are in the supplementary.

This idea can be further extended to the alignment of feature space, as well. We can insert another alignment network on point features and predict a feature transformation matrix to align features from different input point clouds. However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a mini-network. An orthogonal transformation will not lose information in the input, thus is desired. We find that by adding the regularization term, the optimization becomes more stable and our model achieves better performance.

4.3. Theoretical Analysis

Universal approximation We first show the universal approximation ability of our neural network to continuous set functions. By the continuity of set functions, intuitively, a small perturbation to the input point set should not greatly change the function values, such as classification or segmentation scores.

Formally, let $\mathcal{X} = \{S : S \subseteq [0,1]^m \text{ and } |S| = n\}$, $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function on $\mathcal{X}$ w.r.t to Hausdorff distance $d_H(\cdot,\cdot)$, i.e., $\forall \epsilon > 0, \exists \delta > 0$, for any $S, S' \in \mathcal{X}$, if $d_H(S, S') < \delta$, then $|f(S) - f(S')| < \epsilon$. Our theorem says that f can be arbitrarily approximated by our network given enough neurons at the max pooling layer, i.e., K in (1) is sufficiently large.

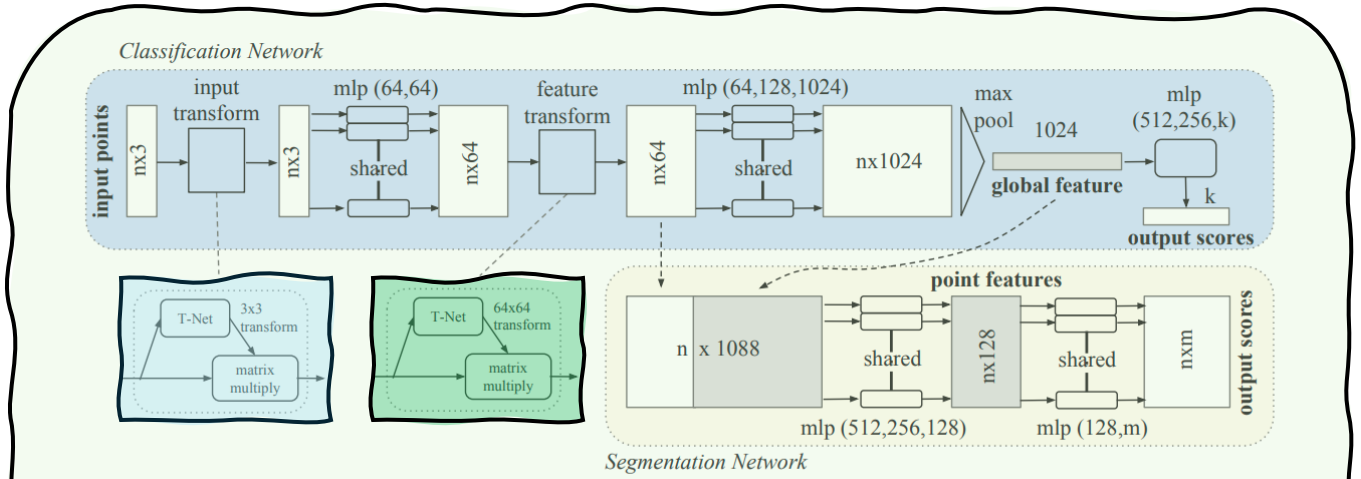


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

the image case. We predict an affine transformation matrix by a mini-network (T-net in Fig 2) and directly apply this transformation to the coordinates of input points. The mini-network itself resembles the big network and is composed by basic modules of point independent feature extraction, max pooling and fully connected layers. More details about

However, transformation matrix in the feature space has much higher dimension than the spatial transform matrix, which greatly increases the difficulty of optimization. We therefore add a regularization term to our softmax training loss. We constrain the feature transformation matrix to be close to orthogonal matrix:

$$L_{reg} = \|I - AA^T\|_F^2, \qquad (2)$$

where A is the feature alignment matrix predicted by a

```python
def pointnetloss(outputs, labels, m3x3, m64x64, alpha = 0.0001):
    criterion = torch.nn.NLLLoss()
    bs = outputs.size(0)
    id3x3 = torch.eye(3, requires_grad=True).repeat(bs, 1, 1)
    id64x64 = torch.eye(64, requires_grad=True).repeat(bs, 1, 1)
    if outputs.is_cuda:
        id3x3 = id3x3.cuda()
        id64x64 = id64x64.cuda()
    diff3x3 = id3x3 - torch.bmm(m3x3, m3x3.transpose(1, 2))
    diff64x64 = id64x64 - torch.bmm(m64x64, m64x64.transpose(1, 2))
    return criterion(outputs, labels) + alpha * (torch.norm(diff3x3) + torch.norm(diff64x64)) / float(bs)
```

Figure 4. **Qualitative results for semantic segmentation.** Top row is input point cloud with color. Bottom row is output semantic segmentation result (on points) displayed in the same camera viewpoint as input.

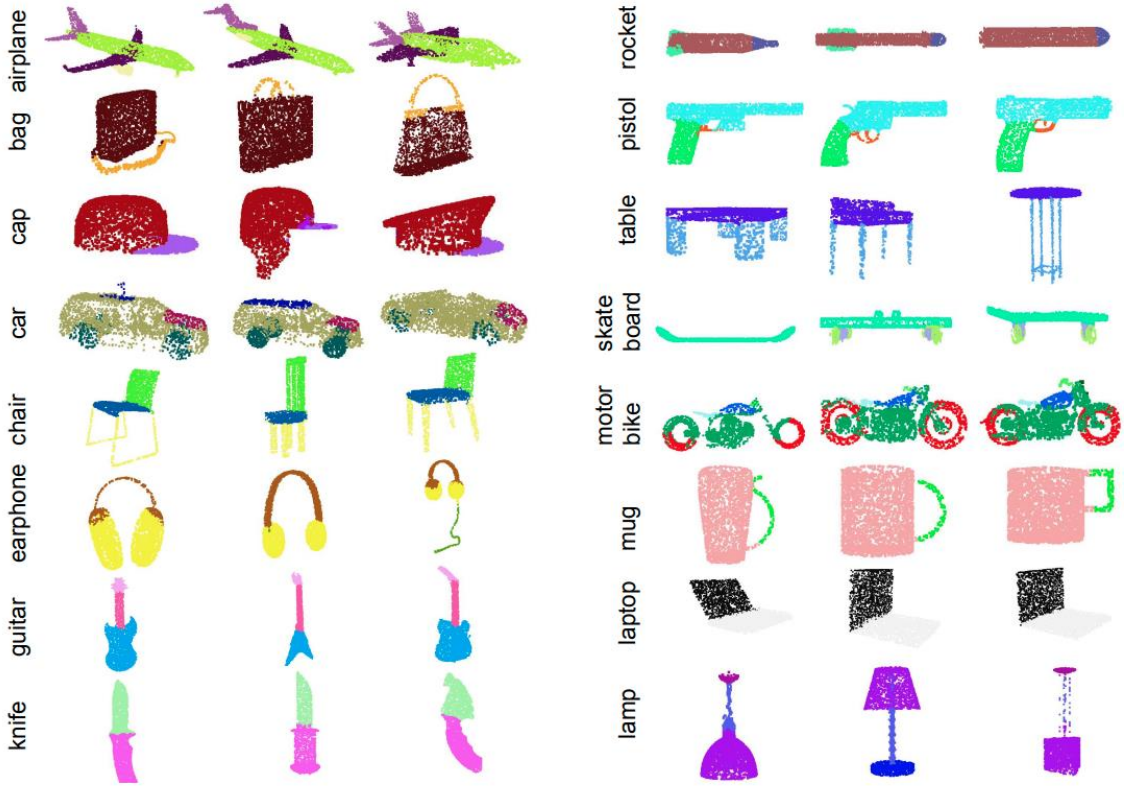


Figure 21. **PointNet segmentation results on complete CAD models.**

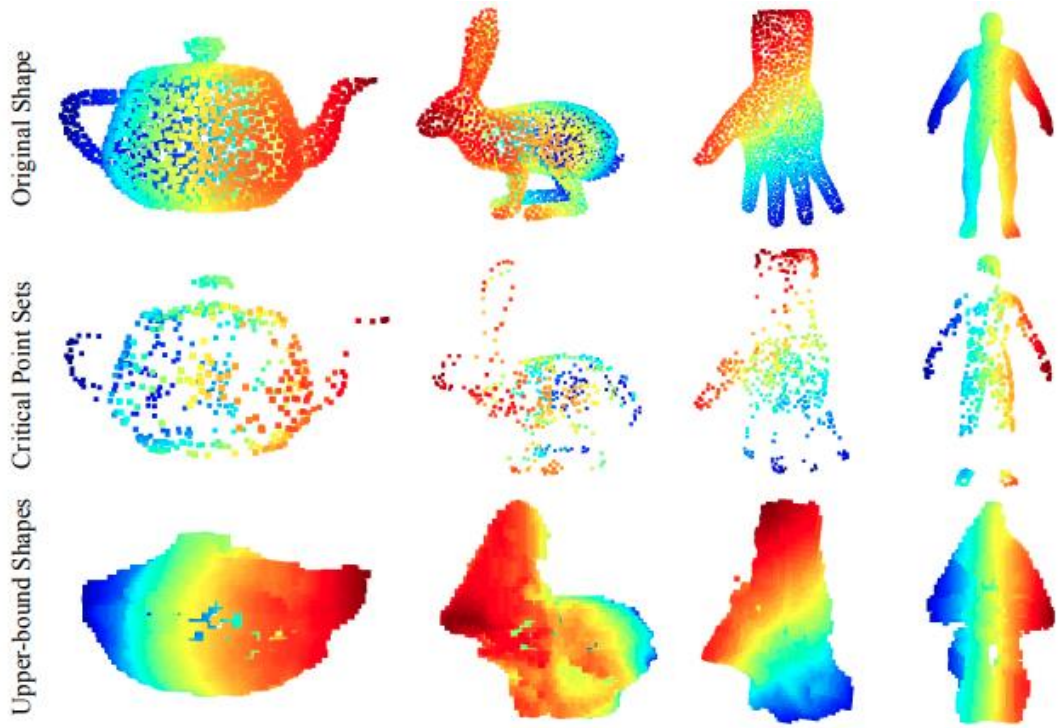


Figure 18. **The critical point sets and the upper-bound shapes for unseen objects.** We visualize the *critical point sets* and the *upper-bound shapes* for teapot, bunny, hand and human body, which are not in the ModelNet or ShapeNet shape repository to test the generalizability of the learnt per-point functions of our PointNet on other unseen objects. The images are color-coded to reflect the depth information.

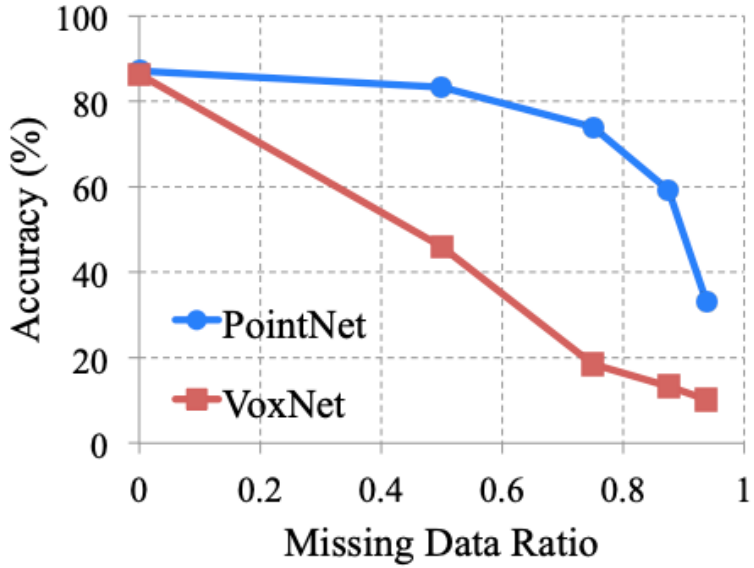


Figure 8. **PointNet v.s. VoxNet [17] on incomplete input data.** Metric is overall classification accuracy on ModelNet40 test set. Note that VoxNet is using 12 viewpoints averaging while PointNet is using only one view of the point cloud. Evidently PointNet presents much stronger robustness to missing points.

References:

1. PointNet paper: https://arxiv.org/pdf/1612.00593.pdf
2. Code snippet at slide 15: https://towardsdatascience.com/deep-learning-on-point-clouds-implementing-pointnet-in-google-colab-1fd65cd3a263