Convolutional point Transformer

Chaitanya Kaul[1](✉), Joshua Mitton[1], Hang Dai[2], and Roderick Murray-Smith[1]

[1] School of Computing Science, University of Glasgow, G12 8RZ
[2] Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
chaitanya.kaul@glasgow.ac.uk

Abstract. We present CpT: Convolutional point Transformer – a novel neural network layer for dealing with the unstructured nature of 3D point cloud data. CpT is an improvement over existing MLP and convolution layers for point cloud processing, as well as existing 3D point cloud processing transformer layers. It achieves this feat due to its effectiveness in creating a novel and robust attention-based point set embedding through a convolutional projection layer crafted for processing dynamically local point set neighbourhoods. The resultant point set embedding is robust to the permutations of the input points. Our novel layer builds over local neighbourhoods of points obtained via a dynamic graph computation at each layer of the network's structure. It is fully differentiable and can be stacked just like convolutional layers to learn intrinsic properties of the points. Further, we propose a novel Adaptive Global Feature layer that learns to aggregate features from different representations into a better global representation of the point cloud. We evaluate our models on standard benchmark ModelNet40 classification and ShapeNet part segmentation datasets to show that our layer can serve as an effective addition for various point cloud processing tasks while effortlessly integrating into existing point cloud processing architectures to provide significant performance boosts.

1 Introduction

3D data takes many forms. Meshes, voxels, point clouds, multi view 2D images, RGB-D are all forms of 3D data representations. Amongst these, the simplest raw form that 3D data can exist in is a discretized representation of a continuous surface. This can be visualized as a set of points (in $\mathbb{R}^3$) sampled from a continuous surface. Adding point connectivity information to such 3D point cloud representations creates representations known as 3D meshes. Deep learning progress on processing 3D data was initially slow primarily due to the fact that early deep learning required a structured input data representation as a prerequisite. Thus, raw sensor data was first converted into grid-like representations such as voxels, multi view images, or data from RGB-D sensors was used to interpret geometric information. However, such data is computationally expensive to process, and

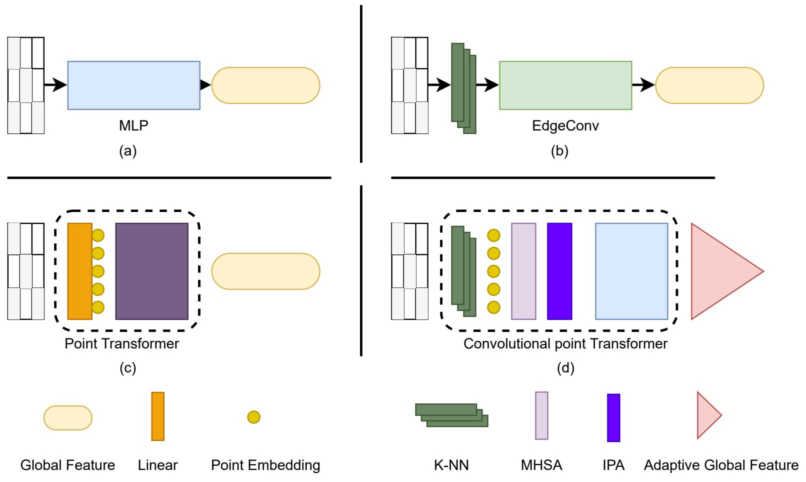


Fig. 1: Overview of the structure of (a) PointNet, (b) DGCNN, (c) Point Transformer, and, (d) Convolutional point Transformer illustrating the differences between these point based convolution layers. Global Feature denotes max or average pooling operations, MHSA denotes multi-head self attention, IPA denotes InterPoint Attention.

in many cases, it is not the true representation of the 3D structure that may be required for solving the task. Modern applications of point clouds require a large amount of data processing. This makes searching for salient features in their representations a tedious task. Processing 3D meshes (along with added 3D point cloud information), requires dealing with their own sets of complexities and combinatorial irregularities, but such structures are generally efficient to process due to knowledge of point set connectivity. This motivation has sparked interest in processing 3D points directly at a point level instead of converting the points to intermediate representations. Lack of any general structure in the arrangement of points serves as a challenge in the ability to process them. This is due to the fact that point clouds are essentially set representations of a continuous surface in a 3D space. The seminal works on processing points proposed approaches to deal with this lack of order by constructing set based operations to ingest 3D points directly. They then created a symmetric mapping of such set based representations in a high dimensional space [22, 42]. This representation was further processed by symmetry preserving operations to create a feature representation of the point cloud, before passing them through multi layer perceptrons for solving the task. Representations created on a per-point basis are generally not robust as there is no concept of locality in the data representations used to create them. Various methods have been introduced in literature to add the notion of locality to point set processing [23, 16, 38, 33, 10, 11].

Recently proposed transformer architectures have found great success in language tasks due to their ability to handle long term dependencies well [31]. These models have been successfully applied to various computer vision applications [2, 3] and are already even replacing the highly successful CNNs as the de facto approach in many applications [35, 19]. Even though various domains of vision research have adopted the transformer architecture, their applications to 3D point cloud processing are still very limited [44, 4]. This leaves a gap in the current research on processing points with transformer based structures.

The basic intuition of our work lies in three simple points. First, transformers have been shown to be intrinsically invariant to the permutations in the input data [41] making them ideal for set processing tasks. Second, 3D points processed by most existing deep learning methods exploit local features to improve performance. However, these techniques treat points at a local scale to keep them invariant to input permutations leading to neglecting geometric relationships among point representations, and the inability of said models to capture global concepts well. Third, existing methods rely on one symmetric aggregation function (eg. max pooling) to aggregate the features that point operators (such as MLPs and 1D Convolutions) learn. This function is likely to leave out important feature information about the point set. To address these points, we propose *CpT: Convolutional point Transformer*. CpT differs from existing point cloud processing methods due to the following reasons:

i) CpT uses a K-nearest neighbour graph at every layer of the model. Such a dynamic graph computation [33] requires a method to handle the input data, in order to create a data embedding that can be fed into a transformer layer. Towards this end, we propose a novel Point Embedding Module that first constructs a dynamic point cloud graph at every stage of the network, and creates a point embedding to feed into a transformer layer for its processing.
ii) Transformers employ multi-head self attention to create contextual embeddings. Such an attention mechanism works to enhance the learning of features in the data. However, it has been shown that adding sample wise attention to data can help improve the performance of a transformer model even further [27, 8, 24, 6]. To this end, we propose to add an InterPoint Attention Module (Figure 2 (c)) to the Transformer which learns to enhance the output by learning to relate each point in the input to every other point. This helps capture better geometric relationships between the points and aids in better learning of the local and global concepts in the data. The resultant transformer block, i.e., the CpT layer (shown in Figure 2 (a)) is a combination of multi-head self attention operation, followed by an InterPoint Attention module. The Q, K, V attention projections in this block are convolutional in nature to facilitate learning spatial context.
iii) We propose a novel Global Feature Aggregation layer (Figure 2 (b)) that learns to aggregate features from multiple symmetric global representations of the points through a spatial attention mechanism. Our main novelty here lies in the adaptive nature of this layer's information aggregation as it scales

the attention output through multiplication with a vector β and learns to weight the attention aggregated output over the training of the model.

We evaluate CpT on 3-point cloud processing tasks – classification of ModelNet40 CAD models and estimating their normals, and segmentation on the ShapeNet part dataset. We perform extensive experiments and multiple studies and ablations to show the robustness of our proposed model. Our results show that CpT can serve as an accurate and effective backbone for various point cloud processing tasks.

2 Related Literature

Processing Point Clouds using Deep Learning. The lack of a grid-like structure to points makes applying convolutions directly on them a tedious task. Due to this reason, all previous works in processing 3D points using deep learning required the points to be first converted to a structured representation like voxels, depth maps, multiple 2D views of an object, etc, and then process the resultant representation with conventional deep models [29, 45, 21]. This process has massive computational overheads which result from first converting the data to the grid-like representation and then training large deep learning models on it. The first methods that proposed to treat points as set embeddings in a 3D space were PointNet [22] and Deep Sets [42]. They took the approaches of creating data embeddings that preserve point set symmetry using permutation invariance and permutation equivariance respectively. This drastically reduced the computational overhead as the models used to train the data worked directly on the raw data points, and the models used to train on such data were not very large scale, and yet accurate. These models however, suffered from the drawback of only looking globally at the points. This meant that the higher dimension embeddings created by such methods were not robust to occlusions as the representations only took the particular input point into consideration while creating its representation. This drawback was tackled in works where local neighbourhoods of points were taken into consideration to compute per point representations. PointNet++ [23] used farthest point sampling to estimate the locality of points first, before processing them via weight-shared multilayer perceptrons (MLPs). ECC Nets [26] proposed the Edge Conv operation over a local neighbourhoods of points. Dynamic Graph CNNs [33] created local neighbourhoods by computing a graphical representation of the points before every shared-weighted MLP layer to create a sense of locality in the points. Parameterized versions of convolution operations for points have also been proposed such as Spider CNN [38], and PointCNN [16]. Previous work with kernel density functions to weight point neighbourhoods [36] treated convolutional kernels as non linear functions of the 3D points containing both weight and density estimations. The notion of looking both locally and globally at points was proposed in SAWNet [10]. Other notable approaches that process 3D points as a graph either densely connect local neighbourhoods [43], operate on a superpoint graph to create contextual relationships

[13], or use spectral graph convolutions to process the points [32].

Attention Mechanisms in Vision. Various attention mechanisms exist in the deep learning literature, with one of the first works introducing them for natural language understanding being [20]. The first attention mechanisms applied to vision were based on self attention via the squeeze-and-excite operation of SE Nets [7]. This attention mechanism provided channel-wise context for feature maps and was widely successful in increasing the accuracy over non attention based methods on ImageNet. It was extended in [1] to non-local networks, and in [25] to fully convolutional networks for medical imaging applications. One of the first works that combined channel-wise self attention with spatial attention for images [34, 9] first created attention maps across the entire feature map, followed by a feature map re-calibration step to only propagate the most important features in the networks forward. FatNet [11] successfully applied this concept to point clouds by first applying spatial attention over feature groups of local regions of points, followed by feature weighting using a squeeze-and-excite operation.

Transformers in 2D and 3D Vision. A concurrent line of upcoming work applies the dot product attention to input data for its efficient processing. The Vision Transformer (ViT) [3] was the first real application of the Transformer to vision. It employed a Transformer encoder to extract features from image patches of size 16×16. An embedding layer converted the patches into a transformer-friendly representation and added positional embedding to them. This general structure formed the basis of initial transformer based research in vision, but required large amounts of data to train. DETR [2] was the first detection model created by processing the features of a Convolutional Neural Network by a transformer encoder, before adding a bipartite loss to the end to deal with a set based output. The Convolutional Vision Transformer (CvT) [35] improved over the ViT by combining convolutions with Transformers via a novel convolutional token embedding and a convolutional projection layer. The recently proposed Compact Convolutional Transformer (CCT) [5] improve over the ViT and CvT models by showing that with the right size and tokenization, transformers can perform head-to-head with state of the art CNN models on small scale datasets. The Swin Transformer [19] proposed a hierarchical transformer network whose representations are computed with shifted windows.

The application of the transformer to point cloud processing is limited. Existing works include the Point Transformer [44] which applies a single head of self attention on the points to create permutation invariant representations of local patches obtained via K-NN and furthest point sampling, and the Point Cloud Transformer [4] which applies a novel offset attention block to 3D point clouds. Over work extends over the existing literature to create a novel transformer block that performs feature wise as well as pointwise attended features of the input for its accurate and effective processing.

3 Convolutional Point Transformer

Notation. We denote a set of N-points with a D-dimensional embedding in the lth CpT layer as, $\mathbf{X}^l = \{\mathbf{x}_1^l, \dots, \mathbf{x}_N^l\}$, where $\mathbf{x}_i^l \in \mathbb{R}^D$. For K nearest neighbours of point i ($i \in \{1 \dots N\}$), we define the set of all nearest neighbours for that point to be $\varDelta\mathbf{X}_i^l = \{\mathbf{x}_{j_1}^l - \mathbf{x}_i^l, \dots, \mathbf{x}_{j_K}^l - \mathbf{x}_i^l\}$, where $j_k : j_k \in \{1, \dots, N\} \wedge (j_k \neq i)$.

The Convolutional point Transformer (CpT) layer is shown in Figure 2. Our main contributions are the Point Embedding Module (Section 3.1), the Inter-Point Attention Module with a convolutional attention projection (Section 3.2) and the Adaptive Global Feature module (Section 3.3). We use CpT in 3 different newtwork architectures - PointNet [22], DGCNN [33], Point Transformer [44]. We do this by replacing the point convolution layers in PointNet, the Edge-Conv layer in DGCNN and the attention layer in Point Transoformer with CpT. The information flow in all these architectures is similar. When an input point cloud of size $N \times 3$ is passed through the architecture, a graph of the points is computed via finding its K-Nearest Neighbours based on Euclidean distance. This representation is then passed through a point embedding layer that maps the input data into a representation implicitly inclusive of the nearest neighbours of the points. This is done via a 2D convolution operation whose degree of overlap across points can be controlled through the length of the stride. A multi-head attention operation is then applied to this embedded representation which is followed InterPoint Attention. The multi-head attention can be seen as learning relevant features of a points embedding as a function of its K nearest neighbours. Such an attention mechanism learns to attend to the features of the points rather than the points themselves (column-wise matrix attention), i.e. for a set of points in a batch, it learns to weight individual feature transformations. The InterPoint Attention on the other hand can be interpreted as learning the relationships between different the points themselves, within a batch (a row-wise matrix attention operating per point embedding, rather than per individual feature of the points). Each attention block is followed by a residual addition and a combination of LayerNormalization and 2 1D Convolutions. These operations form one CpT layer. We update the graph following these operations and pass it into the next CpT layer. The output of the final CpT layer in all network architectures is fed through our Adaptive Global Feature module. It progressively learns to attend to features of max pooled and average pooled global representations of the features during training, to create a richer final feature representation.

3.1 Point Embedding Module

The point embedding module takes a k-NN graph as an input. The general structure of the input to this module is denoted by $(B, f, N, \varDelta\mathbf{X}_i^l) \in \mathbb{R}^{(B \times f \times N \times \varDelta\mathbf{X}_i^l)}$, where B and f are the batch size and input features respectively. This layer is essentially a mapping function F_θ that maps the input into an embedding E_θ. This operation is denoted by,

$$\mathbf{y} = \mathbf{F}_\theta(I(B, f, N, \varDelta\mathbf{X}_i^l)), \mathbf{y} \in \mathbb{R}^{(B \times f \times \mathbf{E}_\theta)}.$$

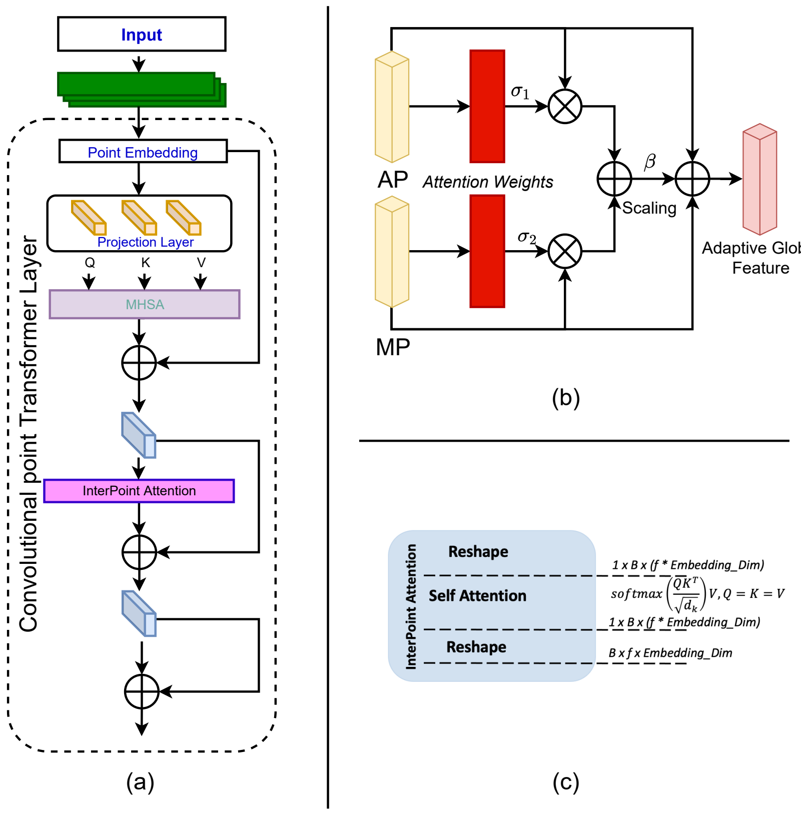


Fig. 2: The CpT Transformer Block with Dot Product Attention and InterPoint Attention are shown in (a). (c) shows the inner workings of the InterPoint Attention module. Flow of information is sequential from top to bottom. (b) shows the Adaptive Global Feature module. AP is average pooling and MP is max pooling. the Flow of information is from left to right.

There are many choices available for the mapping function $\mathbf{F}_\theta$. We use a convolution operation with a fixed size padding and stride. This allows us to train this module in a end-to-end setting along with the rest of the network as it is fully differentiable and can be plugged anywhere in the architecture.

3.2 Convolutional point Transformer Layer

Our Convolutional point Transformer layer uses a combination of multi-head self attention and InterPoint attention to propagate the most important, salient features of the input through the network. The CpT layer leverages spatial context and moves beyond a fully connected projection, by using a convolution layer to sample the attention matrices. Instead of creating a more complicated network design, we leverage the ability of convolutions to learn relevant feature sets for 3D points. Hence, we replace the original fully connected layers in the attention block with depthwise convolution layers which forms our convolutional projection to obtain the attention matrices. The general structure of the CpT layer is shown in Figure 2 (a). The functionality of the rest of the transformer block is similar to ViT [3] where normalization and feedforward layers are added after every attention block and residual mappings are used to enhance the feature learning. The notable difference from ViT is that the feedforward layers are replaced with 1D convolution operations.

Formally, the projection operation of the CpT layer is denoted by the following operation,

$$z_i^{q/k/v} = Convolution(z_i, p, s),$$

where $z^{q/k/v}$ is the input for the $Q/K/V$ attention matrices for the i-th layer and z_i is the input to the convolution block. The convolution operation is implemented as a depthwise separable convolution operation with tuples of kernel size p and stride s. We now formalize the flow of data through the entire CpT block for a batch size B as,

$$out_i^a = \gamma(MHSA(z_i^{q/k/v})) + z_i^{q/k/v},$$

where out_i^a is the output of the multi-head attention block, $MHSA$ is the multi-head attention given by $MHSA(\cdot) = softmax(\frac{QK^T}{\sqrt{d}}V)$. $\sqrt{d}$ is used to scale the attention weights to avoid gradient instabilities. γ is the layer normalization operation. The addition denotes a residual connection. This output is further processed as in a vanilla transformer in the following way,

$$out_i^b = \gamma(FF(out_i^a)) + out_i^a.$$

FF here denotes the feedforward 1D convolution layers. The InterPoint attention is then computed on this output as,

$$out_i^c = \gamma(IPA(out_i^b)_{i=1}^B) + out_i^b,$$

which is then processed in a similar manner by layer normalization and feedforward layers as,

$$out_i^d = \gamma(FF(out_i^b) + out_i^b.$$

Following this step, the graph is recomputed and the process restarts for the $l + 1$-th layer.

3.3 Adaptive Global Feature

The Adaptive Global feature layer (Figure 2 (b)) takes the output of the last CpT layer in a network and learns to create a robust attention-based global representation from it. The output of the final CpT layer are passed through max pooling and average pooling operations. Then attention weights σ_1 and σ_2 are learnt to weight these global representations before combining them via an aggregation function. The output of this aggregation function is further scaled via a vector β. β is initialized as a vector of all zeros, and the same shape as the pooled feature outputs. It gradually learns to assign weight to the attended features as the training progresses. Let the pooling operations, P be defined as the set, $P = \{AP, MP\}$. The output of the Adaptive Global Feature layer is denoted by,

$$AGF = \beta(\sum_{i=1}^{2} \sigma_i \cdot P) + AP + MP$$

4 Experiments and Results

We evaluate our model on two different datasets for point cloud classification, part segmentation and surface normal estimation tasks. For classification and surface normal estimation, we use the benchmark ModelNet40 dataset [37] and for object part segmentation, we use the ShapeNet Part dataset [40].

4.1 Implementation Details

Unless stated otherwise, all our models are trained in PyTorch on a batch size of 32 for 250 epochs. SGD with an initial learning rate of 0.1 and momentum 0.9 is used. We use a cosine annealing based learning rate scheduler. The momentum for batch normalization is 0.9. Batch Normalization decay is not used. Dropout, wherever used, is used with a rate of 0.5. Custom learning rate schedules are used for the segmentation tasks after initial experimentation. For the classification and 3D indoor scene segmentation tasks, we compute dynamic graphs using 20 nearest neighbours, while for the part segmentation task, we use 40 nearest neighbours. We use NVIDIA A6000 GPUs for our experiments.

4.2 Classification with ModelNet40

The ModelNet40 dataset [37] contains meshes of 3D CAD models. A total of 12,311 models are available belonging to 40 categories, split into a training-test set of 9,843-2,468 respectively. We use the official splits provided for all our experiments and datasets to keep a fair comparison. In terms of data pre-processing, we follow the same steps as [22]. We uniformly sample 1024 points from the mesh surface and rescale the point cloud to fit a unit sphere. Data augmentation is used during the training process. We perturb the points with random jitter and scalings during the augmentation process.

It can be seen from the results that CpT outperforms existing classification methods, including the Point Transformer [44]. CpT, when replacing the point convolution in PointNet and the EdgeConv in DGCNN, can be seen to provide significant performance boosts. In the Static Graph approach, the K-NN graph

Method	Class Accuracy (%)	Instance Accuracy (%)
3D ShapeNets [40]	77.3	84.7
VoxNet [21]	83.0	85.9
PointNet [22]	86.0	89.2
PointNet++ [23]	-	90.7
SpiderCNN [38]	-	90.5
PointWeb [43]	89.4	92.3
PointCNN [16]	88.1	92.2
Point2Sequence [17]	90.4	92.6
ECC [26]	83.2	87.4
DGCNN [33]	90.2	92.2
FatNet [11]	90.6	93.2
KPConv [30]	-	92.9
SetTransformer [14]	-	90.4
PCT [4]	-	93.2
PointTransformer [44]	90.6	93.7
CpT (PointNet*)	88.1	90.9
CpT (Static Graph)	90.3	92.1
CpT (DGCNN*)	**90.9**	**93.9**

Table 1: Classification results on the ModelNet40 dataset. CpT (PointNet*) and CpT (DGCNN*) denote the backbone architecture the CpT layer was placed in. All trained architectures use our Adaptive Global Feature module.

is only computed once (in the first CpT layer) and the same graph (and it's features) is used by the subsequent layers in the network. Here, even when we do not recompute the graph at every layer, CpT outperforms Dynamic Edge Conditioned Filters (ECC) [26] and performs at par with DGCNN. Dynamic graph computations before every layer help CpT surpass the accuracy of all existing graph and non graph based approaches, including outperforming existing transformer based approaches [4, 44].

4.3 Segmentation results with ShapeNet Part

The ShapeNet Part dataset [40] contains 16,881 3D shapes from 16 object categories. A total of 50 object parts are available to segment. We sample 2048 points from each shape and follow the official training-testing splits for our experiments. Our results on the dataset are summarized in Table 2, while the visualizations produced by our model are shown in Figure 3. We train two CpT models for this task, with a DGCNN backbone, and a Point Transformer backbone. We note that CpT (DGCNN*) outperforms DGCNN by 0.9% on the IoU metric while CpT (PointTransformer*) out performs the point transformer by 0.2%.

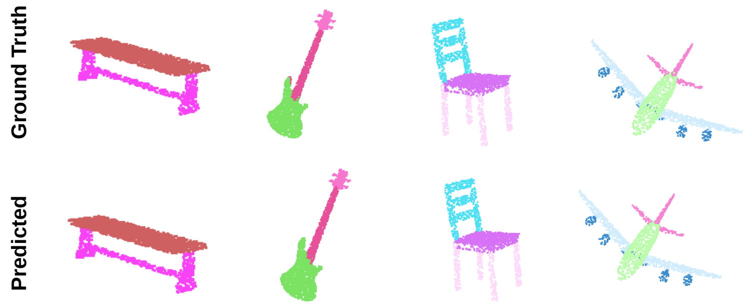


Fig. 3: Visualizing the segmentation results from the ShapeNet Part Dataset. Ground truth images are in the top row and their corresponding segmentation maps in the bottom row..

Method	IoU
Kd-Net [12]	82.3
SO-Net [15]	84.6
PointNet++ [23]	85.1
SpiderCNN [38]	85.3
SPLATNet [28]	85.4
PointCNN [16]	86.1
PointNet [22]	83.7
DGCNN [33]	85.1
FatNet [11]	85.5
PointASNL [39]	86.1
RSCNN [18]	86.2
KPConv [30]	86.4
PCT [4]	86.4
PointTransformer [44]	86.6
CpT (DGCNN*)	85.9
CpT (PointTransformer*)	**86.8**

Table 2: Results on the ShapeNet Part Segmentation dataset. CpT (DGCNN*) and CpT (PointTransformer*) denote the backbone architecture the CpT layer was placed in.

4.4 Normal Estimation

We estimate normals of the ModelNet40 dataset where each point cloud has a corresponding normal label. Normals are crucial in understand the shape and underlying geometry of 3D objects. We train 2 backbone architectures for this task – CpT (PointNet*) and CpT (DGCNN*). Our results are summarized in table 3. It can be seen that, when replacing the point convolution in PointNet, and the EdgeConv in DGCNN with our CpT layer, we get significant performance boosts over the baseline models.

Method	Error
PointNet [22]	0.47
CpT (PointNet*)	**0.31**
DGCNN [33]	0.29
CpT (DGCNN*)	**0.15**

Table 3: Results on the normal estimation task. Results on the PointNet and DGCNN backbones are reported. The reported error is the average cosine distance error. Lower is better.

5 Ablation Study

In this section, we detail experiments of our extensive studies to shed light at the inner workings of the CpT layer. We conducted a series of ablation studies to highlight how the different building blocks of CpT come together. The DGCNN backbone is used for these experiments.

Static v/s Dynamic Graphs. We trained two CpT models for these experiments. The K-NN graph computation is only done before the first CpT Layer in the Static CpT model. The results for this experiment are shown in Table 1. Computing the graph before each transformer layer helps boost CpT's instance accuracy by 1.8%.

Global Representations vs Graph Representations. We compared dynamic graph computation with feeding point clouds directly into the CpT layer. This equates to removing the K nearest neighbour step from the layer in Figure 2 (a). As the points were directly fed into CpT, we also replaced the Point Embedding Module with a direct parameterized relational embedding This relation was learnt using a convolution operation. The rest of the layer structure remained unchanged. The resultant model trained on global representations performed well. The results are summarized in Table 4. PointNet [22] is added to the table for reference as it also takes a global point cloud representation as an input. The CpT with dynamic graph computation outperforms both methods, but it is interesting to note that the CpT model that works directly on the entire point cloud manages a higher class accuracy than PointNet.

Model	Class Accuracy (%)
PointNet [22]	86.2
CpT (No locality)	87.6
CpT (Dynamic Graph)	90.6

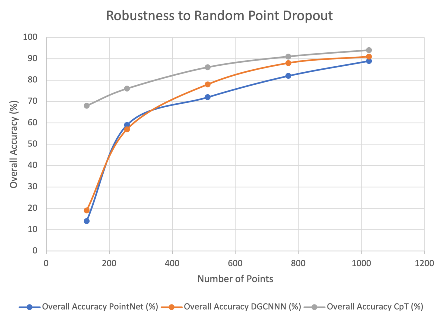
Table 4: Comparison of different input representations.

Number of Nearest Neighbours for dynamic graph computation. We also experiment with the number of nearest neighbours used to construct the graph. CpT layers implemented with 20 nearest neighbour graphs performs the best. This is due to the fact that for large distances beyond a particular threshold, the euclidean distance starts to fail to approximate the geodesic distance. This leads to capturing points that may not lie in the true neighbourhood of the point while estimating its local representation.

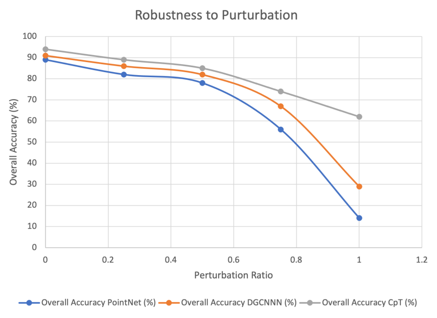
k	Class Accuracy (%)
10	89.6
20	90.6
30	90.3
40	90.4

Table 5: CpT with different number of nearest neighbours.

CpT Robustness. We test the robustness of CpT towards classifying sparse datasets by sampling point clouds at various resolutions from the ModelNet40 dataset and evaluating CpT's performance on them. We sample 1024, 768, 512, 256 and 128 points for each CAD model in the dataset, keeping the input point clouds' resolution small. Figure 4 (a) shows the robustness of CpT compared to PointNet and DGCNN. CpT is not tied to observing input data through receptive fields like CNNs in order to construct its feature space. Even at very small resolutions of the point cloud, the multi-head attention and InterPoint attention learns to model long range dependencies in the input points well to create a local and global understanding of the shape. This leads to a high performance even when there are a small sample of points available.



(a) Effect of randomly dropping out points on performance of CpT.



(b) Effect of adding perturbations to points on the performance of CpT.

Fig. 4: Comparing the robustness of CpT to PointNet [22] and DGCNN [33]

To observe the effect of perturbations on CpT's performance, we perturb each point in the point clouds independently using Gaussian noise. Our results (summarized in Figure 4 (b)) show that our network still manages to remain robust to the added noise even during high amounts of perturbations. The X-axis of the graph in Figure 4 (b)) shows the amount of standard deviation of the Gaussian noise which is linearly increased to perturb the points by a larger amount.

6 Conclusions and Future Work

In this paper, we proposed CpT: Convolutional point Transformer. We showed how transformers can be effectively used to process 3D points with the help of dynamic graph computations at each intermediate network layer. The main contributions of our work include, first, a Point Embedding Module capable of taking a dynamic graph as an input and transforming it into a transformer friendly data representation. Second, the InterPoint Attention Module which uses self attention to facilitate cross talk between the points in an arbitrary batch. Through this work, we have shown for the first time that different self attention methods can be efficiently used inside a single transformer layer for 3D point cloud processing ([44] only uses one form of self attention while [4] only uses an offset attention operator). We also proposed an Adaptive Global Feature module that compliments CpT by learning to attend to the most important features inside different global representations of point cloud features. CpT outperforms most existing convolutional and transformer based approaches for point cloud processing on a variety of benchmark tasks. To improve performance, future directions of this research lie in learning to sample points uniformly along the manifold of the 3D point cloud to preserve its local shape. This can lead to learning better local representations of the data and in turn, creating models with improved accuracy. Our results already show that CpT is capable of taking local context and processing it effectively with global information present in the points. We have shown how CpT can be easily integrated into various existing architectures that are used for processing 3D points. We believe that CpT can serve as an effective backbone for future point cloud processing tasks and be extended to various applications.

Acknowledgements Chaitanya Kaul and Roderick Murray-Smith acknowledge funding from the QuantIC project funded by the EPSRC Quantum Technology Programme (grant EP/MO1326X/1) and the iCAIRD project, funded by Innovate UK (project number 104690). Joshua Mitton is supported by a University of Glasgow Lord Kelvin Adam Smith Studentship. Roderick Murray-Smith acknowledges funding support from EPSRC grant EP/R018634/1, Closed-loop Data Science.

References

1. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: GCNet: Non-local networks meet squeeze-excitation networks and beyond. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pp. 1971–1980 (2019)
2. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16×16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020)
4. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: Point cloud transformer. Computational Visual Media **7**(2), 187–199 (2021)
5. Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., Shi, H.: Escaping the Big Data paradigm with Compact Transformers. arXiv preprint arXiv:2104.05704 (2021)
6. Ho, J., Kalchbrenner, N., Weissenborn, D., Salimans, T.: Axial attention in multi-dimensional transformers. arXiv preprint arXiv:1912.12180 (2019)
7. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
8. Iida, H., Thai, D., Manjunatha, V., Iyyer, M.: Tabbie: Pretrained representations of tabular data. arXiv preprint arXiv:2105.02584 (2021)
9. Kaul, C., Manandhar, S., Pears, N.: FocusNet: An attention-based fully convolutional network for medical image segmentation. In: 2019 IEEE 16th international symposium on biomedical imaging (ISBI 2019). pp. 455–458. IEEE (2019)
10. Kaul, C., Pears, N., Manandhar, S.: SAWNet: A spatially aware deep neural network for 3D point cloud processing. arXiv preprint arXiv:1905.07650 (2019)
11. Kaul, C., Pears, N., Manandhar, S.: FatNet: A feature-attentive network for 3D point cloud processing. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 7211–7218. IEEE (2021)
12. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: Proceedings of the IEEE international conference on computer vision. pp. 863–872 (2017)
13. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4558–4567 (2018). https://doi.org/10.1109/CVPR.2018.00479
14. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set Transformer: A framework for attention-based permutation-invariant neural networks. In: International Conference on Machine Learning. pp. 3744–3753. PMLR (2019)
15. Li, J., Chen, B.M., Lee, G.H.: SO-Net: Self-Organizing Network for Point Cloud Analysis. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9397–9406. IEEE (2018)
16. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on $\mathcal{X}$-transformed points. Advances in neural information processing systems **31**, 820–830 (2018)
17. Liu, X., Han, Z., Liu, Y.S., Zwicker, M.: Point2sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 8778–8785 (2019)

18. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8895–8904 (2019)
19. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021)
20. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1412–1421 (2015)
21. Maturana, D., Scherer, S.: VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 922–928 (2015). https://doi.org/10.1109/IROS.2015.7353481
22. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
23. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems **30** (2017)
24. Rao, R.M., Liu, J., Verkuil, R., Meier, J., Canny, J., Abbeel, P., Sercu, T., Rives, A.: MSA Transformer. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 8844–8856. PMLR (18–24 Jul 2021), https://proceedings.mlr.press/v139/rao21a.html
25. Roy, A.G., Navab, N., Wachinger, C.: Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks. In: International conference on medical image computing and computer-assisted intervention. pp. 421–429. Springer (2018)
26. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3693–3702 (2017)
27. Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C.B., Goldstein, T.: Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. arXiv preprint arXiv:2106.01342 (2021)
28. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2530–2539 (2018)
29. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: Proceedings of the IEEE international conference on computer vision. pp. 945–953 (2015)
30. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: KPConv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6411–6420 (2019)
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
32. Wang, C., Samari, B., Siddiqi, K.: Local spectral graph convolution for point set feature learning. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. pp. 56–71. Springer International Publishing, Cham (2018)

33. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Transactions On Graphics (ToG) **38**(5), 1–12 (2019)
34. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: CBAM: Convolutional Block Attention Module. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018)
35. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: CvT: Introducing convolutions to vision transformers. arXiv preprint arXiv:2103.15808 (2021)
36. Wu, W., Qi, Z., Fuxin, L.: PointConv: Deep convolutional networks on 3D point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9621–9630 (2019)
37. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
38. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 87–102 (2018)
39. Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S.: PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5589–5598 (2020)
40. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A Scalable Active Framework for Region Annotation in 3D Shape Collections. ACM Trans. Graph. **35**(6) (Nov 2016), https://doi.org/10.1145/2980179.2980238
41. Yun, C., Bhojanapalli, S., Rawat, A.S., Reddi, S., Kumar, S.: Are transformers universal approximators of sequence-to-sequence functions? In: International Conference on Learning Representations (2019)
42. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. Advances in Neural Information Processing Systems **30** (2017)
43. Zhao, H., Jiang, L., Fu, C.W., Jia, J.: PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5560–5568 (2019). https://doi.org/10.1109/CVPR.2019.00571
44. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021)
45. Zhou, Y., Tuzel, O.: VoxelNet: End-to-end learning for point cloud based 3D object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4490–4499 (2018)