Adaptive Wavelet Transformer Network for 3D Shape Representation Learning

Anonymous authors
Paper under double-blind review

Abstract

We present a novel method for 3D shape representation learning using multi-scale wavelet decomposition. Distinct from previous works that either decompose 3D shapes into complimentary components at a single scale, or naively adopt up-/down-sampling to build hierarchies and treat all points or local regions equally, we decompose 3D shapes into sub-bands components at multiple scales and all scales form a decomposition tree in a principled manner rooted in *multi-resolution wavelet analysis*. Specifically, we propose Adaptive Wavelet Transformer Network (AWT-Net) that firstly generates approximation or detail wavelet coefficients per point, classifying each point into high or low sub-bands components, using lifting scheme at multiple scales recursively and hierarchically. Then, AWT-Net exploits Transformers that regard the features from different but complementary components as two holistic representations, and fuse them with the original shape features with different attentions. The wavelet coefficients can be learned without direct supervision on coefficients, and AWT-Net is fully differentiable and can be learned in an end-to-end fashion. Extensive experiments demonstrate that AWT-Net achieves state-of-the-art or competitive performance on 3D shape classification and segmentation benchmarks.

1 Introduction

Analysis of 3D point cloud is a crucial topic in computer vision and graphics, and has wide applications in robotics Chen et al. (2019), autonomous driving Qi et al. (2018), and visual SLAM Hitchcox & Forbes (2020), *etc*. To better understand the 3D shapes, effective point cloud analysis techniques and methods are in great demand. With the thriving of deep learning, numerous well-designed neural networks are applied to extract expressive semantics of 3D point clouds with layered operations, in contrast to low-level hand-crafted shape descriptors. The pioneering work PointNet Qi et al. (2017a) learns a spatial encoding of each point using multi-layer perceptrons and aggregates all point features to a global shape representation. The follow-on work PointNet++ Qi et al. (2017b) adopts a hierarchical encoder-decoder structure to process local regions, which down-samples point clouds in layers and gradually interpolates them back to the original resolution. The following works learn 3D shape representation from different perspectives: Xu et al. (2018); Thomas et al. (2019) extend regular grid convolution to irregular 3D point clouds, Li et al. (2018); Wu et al. (2019) design local operations on point patches, Wang et al. (2019); Shi & Rajkumar (2020) treat point clouds as graphs and apply graph convolution networks, to name a few.

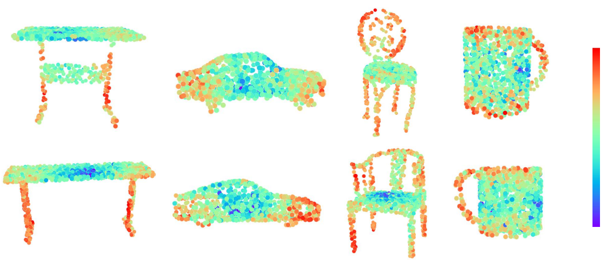

Figure 1: The detail component of table, car, chair, and cup from the test set of ModelNet40 Wu et al. (2015). Colors represent the absolute values of the detail coefficients per point. The larger values (redder colors) indicate, the higher probability that the points locate on high-frequency parts, *e.g.*, edges or non-flat areas. Such geometric representations are learned implicitly in the classification task.

An interesting yet challenging problem is how to decompose 3D shapes into interpretable representations and exploit them for better shape analysis. For instance, some attempts Wang et al. (2019); Lan et al. (2019) capture local geometric relations among center points and their neighbors. However, these works treat all points or local regions equally, making it hard to extract features of the most relevant and crucial geometric parts, *e.g.*, areas containing flat planes or areas containing edges and junction boundaries. Another challenging problem is how to analyze 3D shapes at multiple scales/resolutions in an explainable way. Though some works Qi et al. (2017b); Jiang et al. (2019) extract point cloud features hierarchically, these works build the hierarchy in spatial/semantic spaces based on pairwise point distance, which is less interpretable *w.r.t.* shape geometry.

In this work, we propose a principled method for 3D shape representation learning rooted in wavelet transform. Theoretical properties of wavelets in signal processing, such as time-frequency localization and multi-resolution analysis, have been well studied, making it appropriate to fuse into deep neural networks with interpretability. Inspired by that, we propose to perform multi-resolution analysis within neural networks through lifting scheme Sweldens (1998) to achieve a data-driven wavelet transform. The multi-resolution analysis generates decomposed visual representations at each scale, which contributes to the interpretability of the network. The generated approximation or detail coefficients per point at each scale, corresponding to high or low frequency components, capture essential geometric interest for down-streaming tasks.

Specifically, we first adapt wavelet transform that analyzes 1D temporal or 2D image signals to 3D shapes at different semantic levels and decompose the original point cloud into the approximation/coarse and detail components, as explained in Section 4.1 and 4.2. Next, in Section 4.3, we utilize Transfomers Vaswani et al. (2017); Choromanski et al. (2021) to pay different attention to features from approximation and detailed variation components, and then fuse them with the original point features, respectively. As shown in Figure 1, the approximation and detail components have distinct but complementary effects on reflecting the geometry of 3D shapes. Equipped with wavelet transform and Transformers, we propose *Adaptive Wavelet Transformer Network* (AWT-Net) that captures and refines the holistic and complementary geometry of 3D shapes to supplement neighboring local information. AWT-Net is trained using the loss functions proposed in Section 4.4. Experimental results on challenging benchmarks demonstrate that our AWT-Net achieves state-of-the-art or competitive performance on 3D shape classification and segmentation tasks. Comprehensive analysis and visualizations verify that our model effectively extracts the complementary geometric features from two decomposed components at multiple scales. For example, as shown in Figure 1, we observe that our model can learn meaningfully samplings of point clouds in different categories and can detect key regions consistently within a semantic class, *e.g.*, it separates chair legs and backs from chair seat, and separates cup handle from the body.

2 Related Work

Geometric Point Cloud Models. PointNet Qi et al. (2017a) and DeepSet Zaheer et al. (2017) are pioneering neural network models that directly process point cloud to learn a spatial encoding of each point and then aggregate all individual point features to a holistic representation. Point-Net++ Qi et al. (2017b) partitions point cloud into overlapping local regions by the distance metric of spatial and/or semantic spaces and extracts local features capturing fine geometric structures from neighbors hierarchically. DGCNN Wang et al. (2019) reconstructs the k-NN graph using nearest neighbors in feature space to capture local geometric structure while maintaining permutation invariance. Geo-CNN Lan et al. (2019) defines a convolution operation that aggregates edge features to capture the local geometric relations. Xu et al. (2020) aggregates points from local neighbors with similar semantics in both Euclidean and Eigenvalue space. RS-CNN Liu et al. (2019c) extends regular CNN to encode geometric relation in a local point set by learning the topology. DensePoint Liu et al. (2019b) recognizes the implicit geometry by extracting contextual shape semantics. Our model learns point cloud geometry from a novel perspective, *i.e.*, multi-resolution wavelet analysis which is extensively applied in signal processing but seldom explored for 3D shapes.

Wavelet in Vision. Theoretical properties of multi-resolution analysis using wavelets have been well studied, making such approaches more interpretable than CNNs. Several prior works incorporate wavelet representations into CNNs. Oyallon et al. (2017) proposed a hybrid network that replaces the first layers of ResNet with a wavelet scattering network. This modified ResNet resulted in comparable performance to the original ResNet but has a smaller number of trainable parameters. The wavelet pooling layer proposed in Williams & Li (2018) subsamples image features based on second-level wavelet decomposition. Wavelet-SRNet Huang et al. (2017) reconstructs super-resolution images from multi-scale wavelet coefficients estimated from low-resolution images under coefficient supervision. Similar scheme is adapted in Liu et al. (2020) for image demoiréing and Rong et al. (2020) for burst denoising. In Ramamonjisoa et al. (2021), they decompose a single image using Haar Transform and then reconstruct depth images hierarchically through inverse discrete wavelet transform. All of the mentioned works use pre-defined Haar wavelet which is less flexible for complex 3D shapes. Rustamov & Guibas (2013) adopts lifting scheme but still restricts to linear transformations of Haar coefficients for graph signals. DAWN-Net Rodriguez et al. (2020) uses lifting scheme to implicitly define wavelet transform functions for image recognition. Applying lifting scheme to learn 3D shape representations is more challenging as amounts of shape points are arranged irregularly, and as far as we know, we are among the first to tackle this problem.

Transformer in Vision. The seminal work to adapt Transformer Vaswani et al. (2017) to vision tasks commences with Vision Transformer (ViT) Dosovitskiy et al. (2020) and Detection Transformer Carion et al. (2020). Since then, various applications of Transformer in vision are proposed, including: semantic segmentation Wang et al. (2021); Zhang et al. (2020), generative modeling Chen et al. (2020); Hudson & Zitnick (2021), object detection Zhu et al. (2021); Dai et al. (2021), video analysis Gabeur et al. (2020); Zeng et al. (2020), to name a few. PCT Guo et al. (2021) applies Transformer in 3D point cloud classification and segmentation. Concurrent works include two types of Point Transfomer proposed by Zhao et al. (2021) and Engel et al. (2021). GDA-Net is built on graph spectral theory and uses a 2-order polynomial approximation (Laplacian) as the filter and applies it to a graph to disentangle the graph into high and low frequency components. Thus, the decomposition process in is deterministic and static. PoinTr Yu et al. (2021) converts a point cloud to a sequence of point proxies and employs Transformer for point cloud completion. PPT-Net Hui et al. (2021) adopts Transformer for large-scale place recognition. MViT Fan et al. (2021) and Swin Liu et al. (2021) connect multi-scale feature hierarchies with transformer models by applying different Transformers for features with distinct spatial resolutions. Since multi-scale discrete wavelet transform decomposes point clouds at each subsequent scales, it is intrinsically essential to combine with Transformer in a similar way as in MViT and Swin. Due to quadratic computational complexity of self-attention, Fast Autoregressive Transformers Katharopoulos et al. (2020) is among the first to address the quadratic complexity by using the associativity property of matrix products. Some recent works Kitaev et al. (2020); Dai et al. (2020); Tay et al. (2020); Choromanski et al. (2021) propose efficient Transformers for long sequences. These models can be also adapted for vision tasks and the one proposed in Choromanski et al. (2021) is explored in our work.

3 Background

We briefly review wavelet transform and lifting scheme as the building blocks of our model. We refer the reader to Sweldens (1996); Claypoole et al. (2003); Rodriguez et al. (2020) for more information.

Wavelet Transform/Decomposition. Let us consider a continuous scalar function of a single variable $f(x)$. Wavelet transform is to find a basis generated by a mother wavelet, denoted by $\psi(x)$, to represent $f(x)$ with a few parameters. The basic rule for generating this basis is to translate and dilate $\psi(x)$, *i.e.*, $\psi(2^j x - i)$, where j and i are integers. The basis is localized in both time (controlled by i) and scale/frequency (controlled by j) domains. The original function $f(x)$ is decomposed as a linear combination of the basis:

$$f(x) = \sum_{j,i} a_{j,i} \psi(2^j x - i) \ , \tag{1}$$

where $a_{j,i}$ are the wavelet coefficients of $f(x)$ and can be computed as the inner product of $\psi(2^j x - i)$ with $f(x)$. For a discretized signal $x[n]$ ($n \in Z$), we sample points from $\psi(2^j x - i)$ to form filter banks to decompose $x[n]$. This is classical or first-generation wavelet Santosa (2011). Some commonly used mother wavelets include Haar wavelet Haar (1910), Daubechies wavelet, Biorthogonal wavelet Daubechies (1992), to name a few. Wavelet transform provides a nice framework for multi-resolution analysis of functions, *i.e.*, the wavelet transform generates coarser and coarser versions of the same function while keeps track of the details missed going from fine to coarse. However, the first-generation wavelet mainly works well for infinite or periodic signals,

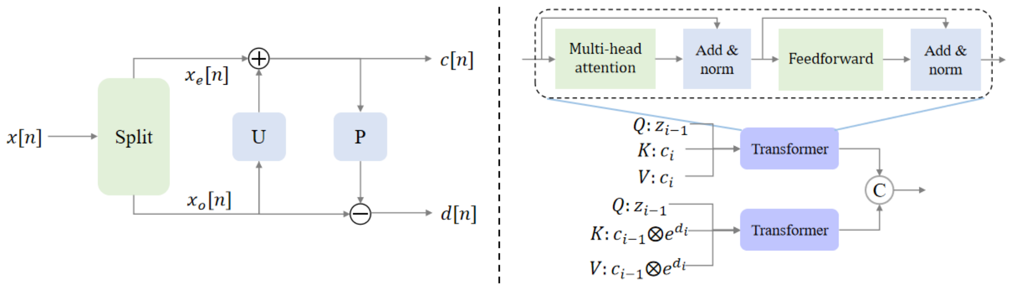

Figure 2: (Left) Lifting stages: split, update (U) and predict (P). (Right) The architecture of Transformers in AWT-Net.

which are rare in many applications. It is still unclear how to design a proper basis for any arbitrary signals sampled irregularly from a bounded domain. One solution is to give up designing a mother wavelet manually. Instead, the decomposition is entirely based on an approach called *lifting scheme*.

Lifting Scheme. Lifting scheme, *a.k.a.*, second-generation wavelets Sweldens (1998); Daubechies & Sweldens (1998), is a simple yet effective approach to define wavelets sharing the same properties as the first-generation wavelets. The lifting scheme takes as input a signal x and generates as outputs the approximation c and the detail d sub-bands of the wavelet transform. The lifting scheme consists of three stages Claypoole et al. (2003) as on the left panel of Figure 2.

- *Split.* This stage splits the input signal into two non-overlapping partitions, *e.g.*, the input signal x is divided into even x_e and odd x_o components, which are defined as $x_e[n] = x[2n]$ and $x_o[n] = x[2n+1]$.
- *Update (U).* This stage separates the signal in frequency domain, generating the approximation c that has the same running average as the original input signal. Let $x_o^{L_U}[n] = x_o[n-L_U], x_o[n-L_U+1], \cdots, x_o[n+L_U-1], x_o[n+L_U]$ denote the sequence of $2L_U+1$ neighboring odd polyphase samples around $x_e[n]$. The even polyphase samples are updated through an update operator $U(\cdot)$ on $x_o^{L_U}[n]$, resulting the approximation c as:
$$c[n] = x_e[n] + U(x_o^{L_U}[n]) \ . \tag{2}$$
- *Predict (P).* As the partitions x_e and x_o are usually closely correlated, given one of them, it is feasible to build a predictor $P(\cdot)$ for the other, by tracking the difference (or detail) d among them. As the even component x_e corresponds to the approximation $c[n]$, it is possible to define $P(\cdot)$ as a function of $c[n]$. Let $c^{L_P}[n] = c[n-L_P], c[n-L_P+1], \cdots, c[n+L_P-1], c[n+L_P]$ denote a sequence of $2L_P+1$ approximation coefficients. The odd polyphase samples are predicted as $P(c^{L_P}[n])$. The resulting prediction residual, or detail d, are computed as:
$$d[n] = x_o[n] - P(c^{L_P}[n]) \ . \tag{3}$$

Note that the update and predict operators U and P implicitly and effectively define the wavelet for decomposition.

4 Adaptive Wavelet Transformer Network

Overview. The update and predict operators in Eq. 2 and Eq. 3 are linear, which are incapable of learning geometry of 3D shapes. In this section, we propose a data-driven *adaptive lifting scheme* that introduces non-linearity into wavelet. Based on this scheme, we develop an Adaptive Wavelet Transformer Network, dubbed as *AWT-Net*, for 3D shape representation learning. The AWT-Net generates shape representations hierarchically, and the decomposition tree is shown on the left panel of Figure 3. Here we show three scales where each scale corresponds to a separate lifting process as illustrated on the left panel of Figure 2. We omit the split, update and predict blocks for conciseness. At each scale, the approximation c_i serves the coarse version of the input signal while the detail d_i maintains the detailed difference between c_i and the input signal. The approximation c_i is further decomposed by another lifting process at the next scale. The AWT-Net is built based on this decomposition tree but replaces the linear operators U and P with non-linear neural networks. Once trained and fixed, the update U and predict P operators can be regarded as two sub-band basis/filters to subdivide the point cloud into high-frequency and low-frequency components, in analogy to the basis in Eq. 1. Its unrolled version is shown on the right panel of Figure 3.

Each module in AWT-Net is described as follows.

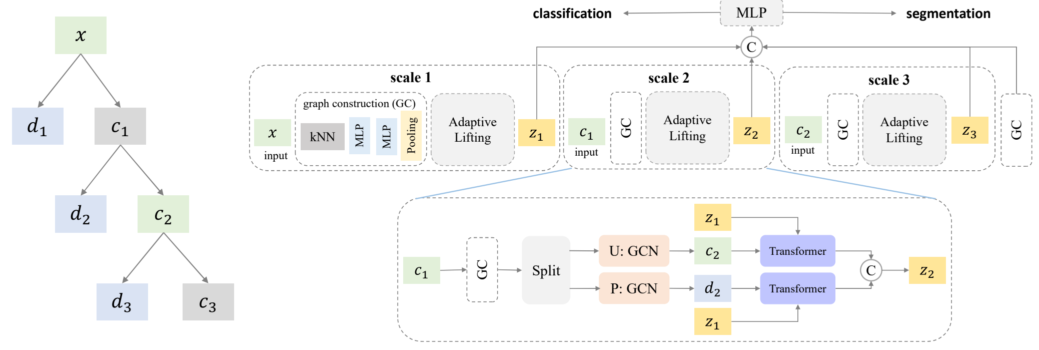

Figure 3: (Left) A multi-resolution wavelet decomposition tree (of three scales) with lifting scheme. (Right) An unrolled version of AWT-Net corresponding to the decomposition tree. Here we omit update and predict operators for simplicity.

4.1 Even-Odd Split

Graph Construction. Assume a point cloud containing N points associated with features $x_i \in \mathbb{R}^d, i \in \{1, 2, \cdots, N\}$. Features can be 3D coordinates, normals and/or high-dimensional features. We construct a k-NN graph $\mathcal{G} = (\mathcal{V}, \tilde{\mathcal{A}}^u, \tilde{\mathcal{A}}^w)$, where each point x_i corresponds to a node $v_i \in \mathcal{V}$, $\tilde{\mathcal{A}}^u$ and $\tilde{\mathcal{A}}^w \in \mathbb{R}^{N\times N}$ are unweighted and weighted adjacency matrices encoding point similarity in feature space. The unweighted and weighted edges connecting two nodes v_i and v_j are defined as:

$$\tilde{\mathcal{A}}^u_{ij}\ \ [,\tilde{\mathcal{A}}^w_{ij}] = \begin{cases} 1\ \ [,\kappa(\|x_i - x_j\|_2)], & x_j \in \mathcal{N}(x_i) \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where $\kappa(\cdot)$ is an non-negative function, *e.g.*, Gaussian function, to ensure that $\tilde{\mathcal{A}}^w$ is a diagonally dominant matrix and $\mathcal{N}$ represents neighborhood. Following Xu et al. (2021), we normalize all edge weights in each row in $\mathcal{A}^w$ to remove the impacts of varying neighborhood sizes and feature scales.

Graph Split. We partition all nodes in a graph into two sets, containing *even* and *odd* nodes, respectively. Since lifting scheme uses information from odd (*resp*. even) nodes to update (*resp*. predict) even (*resp*. odd) nodes as in Eq. 2 (*resp*. Eq. 3), having all neighboring nodes with same parity means that local information cannot be used. Following Narang & Ortega (2009), we define the number of conflicts as the percentage of neighboring nodes that have the same parity after partition. The optimal splitting minimizes the number of conflicts. Theoretically, it is an NP-hard bipartite subgraph problem, but a heuristic solution can be found. We adapt the conservative fixed probability colorer (CFP) algorithm in Fitzpatrick & Meertens (2001), which is based on iteratively greedy local heuristics: at each iteration, a few randomly chosen nodes are activated, and each activated node counts the number of conflicting edges (*i.e.*, connecting neighboring nodes with the same parity) and changes its parity based on the conflict. The algorithm is summarized in Algorithm 1.

4.2 Adaptive Lifting Scheme

Based on nodes' parity after the even-odd split, we rearrange nodes $\mathcal{V}$ to gather odd and even nodes, and also rearrange the weighted adjacency matrix $\mathcal{A}^w$ accordingly:

$$\mathcal{V} = \begin{pmatrix} \mathcal{V}_{odd} \\ \mathcal{V}_{even} \end{pmatrix}, \quad \mathcal{A}^w = \begin{pmatrix} \mathcal{F}_{N_o\times N_o} & \mathcal{J}_{N_o\times N_e} \\ \mathcal{K}_{N_e\times N_o} & \mathcal{L}_{N_e\times N_e} \end{pmatrix},$$

where $\mathcal{V}_{odd}$ and $\mathcal{V}_{even}$ represent the sets of odd and even nodes. The sub-matrix $\mathcal{F}$ is the adjacency matrix of a subgraph containing odd nodes only. The submatrix $\mathcal{J}$ contains edges that do not have conflicts, *i.e.*, edges connecting nodes with the opposite parity. Similar definitions for $\mathcal{K}$ and $\mathcal{L}$.

Algorithm 1: Even-Odd Node Split

```
Require: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}^u)$ with unweighted adjacency, maximum number of iterations $M$
1  Randomly assign initial parity (e.g., 1 for odd, 0 for even) to each node
2  for $m$ = $1{:}M$ do
3      Activate each node with a probability sampled from a uniform distribution independently
4      for each activated node do
5          Choose a parity that minimizes its conflict with neighboring nodes
       /* Keep the number of odd and even nodes roughly the same */
6      $N_o, N_e \leftarrow$ number of nodes with parity1, parity0
7      if $N_o > N_e$ then
8          Randomly flip $\lfloor \frac{N_o - N_e}{2} \rfloor$ odd nodes' parity
9      else
10         Randomly flip $\lfloor \frac{N_e - N_o}{2} \rfloor$ even nodes' parity
```

Unlike the linear lifting scheme introduced in Section 3, we propose data-dependent non-linear update and predict operators, implemented as neural networks whose parameters are adaptively optimized to capture complex shape geometry. Specifically, we apply two Graph Convolutional Networks (GCNs) proposed in Kipf & Welling (2017) as the update and predict operators, separately. The graph convolution allows us to compute the response of a node based on its neighbors decided by the adjacency matrix. Performing graph convolution is equal to performing message gather within a neighborhood around each center node, sharing similarities with the definitions of $x_o^{L_u}[n]$ and $c^{L_p}[n]$ in Eq. 2 and Eq. 3. The non-linear update and predict stages at i-th scale are defined as:

$$c_i = \mathcal{V}_{even} + \text{GCN}(\mathcal{V}_{odd}, \mathcal{F}) \ , \tag{5}$$

$$d_i = \mathcal{V}_{odd} - \text{GCN}(c_i, \mathcal{L}) \ , \tag{6}$$

where c_i and d_i represent the approximation and detail components at i-th scale. We repeat this process at each scale recursively and form a hierarchical decomposition tree, capturing shape geometry at multi-scales. The maximum number of scales can be decided analytically as $\log_2 N$ and the coarsest scale contains only a single point.

4.3 Transformer for Cross Attention

The approximation and detail components play different but complementary roles in representing 3D shape geometry, and thus should not be treated equally. To utilize different variation components gained from lifting scheme, we adopt Transformers Vaswani et al. (2017); Choromanski et al. (2021) to integrate different components with the original input shape features. The architecture of Transformers is shown on the right panel of Figure 2, and multi-head attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^\top)V \ , \tag{7}$$

where Q, K, V denote query, key and value, respectively.

For approximation c_i as coarse component at i-th scale, we set query, key and value to:

$$Q = z_{i-1}, K = c_i, V = c_i \ , \tag{8}$$

where the definition of z_i is shown in Figure 3. Note that for the first scale (*i.e.*, $i = 1$), we set $Q = x$ as input. For detail d_i, as most of the values are close to zeros (explained in Subsection 4.4), directly feed them into Transformer yields sub-optimal results. Utilizing a nice property of wavelets, namely time-frequency localization[1], we regard d_i as a multiplier to strengthen or weaken the context features. Formally, the query, key and value for detail d_i at i-th scale are set as:

$$Q = z_{i-1}, K = c_{i-1} \otimes \exp(d_i), V = c_{i-1} \otimes \exp(d_i) \ , \tag{9}$$

where $\otimes$ represents element-wise multiplication. For the first scale, we set both z_{i-1} and c_{i-1} to x. In the part segmentation task, since each point cloud contains 2048 points, to efficiently

[1]In analogy wave signals expanding time domain, for 3D shapes which expand in the spatial domain, this property can be rephrased as spatial-frequency localization.

compute multi-head attention, we also test Performers Choromanski et al. (2021) in addition to the vanilla Transformers Vaswani et al. (2017). Briefly, Performers represent the (attention) softmax kernel in terms of the Gaussian kernel, and approximate the latter by sampling from the Fourier Transform Rahimi et al. (2007).

4.4 Loss Function

End-to-end training is performed using the cross-entropy loss function, in combination with regularization terms, to enforce a wavelet decomposition structure during training. The loss function takes the form of Eq. 10, where K in the first term denotes the number of classes, y_i and p_i are the ground-truth and the predicted probability for belonging to class i, respectively.

$$\text{Loss} = -\sum_{i=1}^{K} y_i \log(p_i) + \lambda_1 \sum_{i=1}^{S} \text{Smooth}_{L1}(\mathcal{A}^2 c_{i-1} - c_i) + \lambda_2 \sum_{i=1}^{S} \text{Smooth}_{L1}(d_i) \ . \tag{10}$$

In lifting scheme, a perfect output of Eq. 2, *i.e.*, the series $c[n]$, would be a coarse approximation of $x[n]$ at half the resolution. In the second term, we enforce each lifted even node has the same local average as its corresponding node in the original graph. Due to the half resolution of c_i compared with the input c_{i-1}, we raise the adjacent matrix $\mathcal{A}$ to the power of two to capture the node features up to second-order neighbors, and minimize the difference between the input graph features and the coarser version across all the decomposition scales. The rationale for the third term is: if the underlying signal $x[n]$ is locally smooth, the prediction residuals $d[n]$ should be small. Thus, the third term promotes the minimization of the magnitude of detail coefficients $d[n]$ in Eq. 3, indicating smaller energy is contained in $d[n]$.

5 Experiments and Results

We evaluate our model on 3D shape classification and part segmentation tasks on different datasets.

5.1 Implementation details

The number of the constructed neighborhoods at each stage is 64. The maximum number of iterations M used in the split Algorithm 1 is set to 10. The update $U(\cdot)$ and $P(\cdot)$ operators are implemented as two GCNs Kipf & Welling (2017) and the internal structure of each GCN is: Conv-BN-ReLU-Dropout-Conv-BN. We use a two-heads attention in Transformer (for both the vanilla one and Performer). We implement our model in PyTorch Paszke et al. (2019). We use the SGD optimizer with momentum and weight decay set to 0.9 and 0.0001, respectively. For both 3D shape classification and 3D object part segmentation tasks, we train for 350 epochs. The initial learning rate is set to 0.1 and is dropped by 5x every 40 epochs and clipped at $0.9e-5$. The values of λ_1 and λ_2 are set to 0.1 for all tasks.

5.2 Object Classification

We evaluate AWT-Net on ModelNet40 Wu et al. (2015) which contains 9843 training shapes and 2468 test shapes across 40 categories. Points are uniformly sampled from the mesh models by Qi et al. (2017a). Following Wang et al. (2019); Xu et al. (2021), we translate shapes and then randomly shuffle points in the training set. Similar to Qi et al. (2017a;b), we perform several voting tests with random scaling and average the predictions during testing. The quantitative comparisons of the overall classification accuracy with the state-of-the-art methods are listed in Table 1. '+V' denotes the vanilla Transformer Vaswani et al. (2017) and '+P' denotes Performer Choromanski et al. (2021), a variant of the vanilla Transformer. AWT-Net with two scales achieves better or comparable performance with other methods using the same number of points as the input, as shown in Table 1.

We also evaluate AWT-Net on ScanObjectNN by Uy et al. (2019) to examine the robustness to deformed and noisy objects scanned in the real world. We test on the OBJ_ONLY and noisy OBJ_BG splits, and the results are summarized in Table 2, where our model gets the comparable accuracy and performance drop from OBJ_ONLY to OBJ_BG.

Method (time order)	Points	Accuracy
PointNet Qi et al. (2017a)	1K	89.2
PointNet++ Qi et al. (2017b)	1K	90.7
SCN Xie et al. (2018)	1K	90.0
KCNet Shen et al. (2018)	1K	91.0
PointCNN Li et al. (2018)	1K	92.2
Point2Sequence Liu et al. (2019a)	1K	92.6
DGCNN Wang et al. (2019)	1K	92.9
InterpCNN Mao et al. (2019)	1K	93.0
RS-CNN Liu et al. (2019c)	1K	93.6
3D-GCN Lin et al. (2020b)	1K	92.1
FPConv Lin et al. (2020a)	1K	92.5
GSNet Xu et al. (2020)	1K	92.9
PointTransformer Zhao et al. (2021)	1K	93.7
PCT Guo et al. (2021)	1K	93.2
GDANet Xu et al. (2021)	1K	93.8
AWT-Net +V (ours)	1K	93.5
AWT-Net +P (ours)	**1K**	**93.9**

Table 1: Classification accuracy (%) on ModelNet40 dataset.

Method	OBJ_ONLY	OBJ_BG	Acc. drop
PointNet Qi et al. (2017a)	79.2	73.3	↓ 5.9
PointNet++ Qi et al. (2017b)	84.3	82.3	↓ 2.0
SpiderCNN Xu et al. (2018)	79.5	77.1	↓ 5.4
PointCNN Li et al. (2018)	87.9	85.8	↓ 2.1
DGCNN Wang et al. (2019)	86.2	82.8	↓ 3.4
GDANet Xu et al. (2021)	**88.5**	**87.0**	↓ 1.5
AWT-Net +V (ours)	**88.2**	86.4	↓ 1.8
AWT-Net +P (ours)	**88.5**	86.8	↓ 1.7

Table 2: Classification accuracy (%) *w.r.t.* noise on ScanObjectNN dataset.

5.3 Part Segmentation

We evaluate AWT-Net with three scales for shape part segmentation on ShapeNet Part Yi et al. (2016) dataset, which contains 16881 shapes in 16 categories and is labeled in 50 parts where each shape has $2\tilde{5}$ parts. The same voting test in classification is adopted. The instance average, the class average, and each class mean Inter-over-Union (mIoU) are summarized in Table 3. AWT-Net +P achieves the best performance with class mIoU of 85.0% and instance mIoU of 86.6%. It is worth noticing that AWT-Net performs better on objects with obvious junction boundaries such as bag, knife, laptop, and skateboard. Some qualitative segmentation results are visualized in Figure 4.

Method (time order)	class mIOU	instance mIOU	aero	bag	cap	car	chair	ear phone	guitar	knife	lamp	lap top	motor	mug	pistol	rocket	skate board	table
PointNet Qi et al. (2017a)	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++ Qi et al. (2017b)	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
SyncCNN Yi et al. (2017)	82.0	84.7	81.6	81.7	81.9	75.2	90.2	74.9	**93.0**	86.1	84.7	95.6	66.7	92.7	81.6	60.6	82.9	82.1
SCN Xie et al. (2018)	81.8	84.6	83.8	80.8	83.5	79.3	90.5	69.8	91.7	86.5	82.9	96.0	69.2	93.8	82.5	62.9	74.4	80.8
KCNet Shen et al. (2018)	82.2	84.7	82.8	81.5	86.4	77.6	90.3	76.8	91.0	87.0	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
SpiderCNN Xu et al. (2018)	82.4	85.3	83.5	81.0	87.2	77.5	90.7	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.8
Point2Sequence Liu et al. (2019a)	-	85.2	82.6	81.8	87.5	77.3	90.8	77.1	91.1	86.9	83.9	95.7	70.8	94.6	79.3	58.1	75.2	82.8
DGCNN Wang et al. (2019)	82.3	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
RS-CNN Liu et al. (2019c)	84.0	86.2	83.5	84.8	88.8	79.6	91.2	81.1	91.6	88.4	86.0	96.0	73.7	94.1	83.4	60.5	77.7	83.6
3D-GCN Lin et al. (2020b)	82.1	85.1	83.1	84.0	86.6	77.5	90.3	74.1	90.0	86.4	83.8	95.6	66.8	94.8	81.3	59.6	75.7	82.8
GSNet Xu et al. (2020)	83.5	85.3	82.9	84.3	88.6	78.4	89.7	78.3	91.7	86.7	81.2	95.6	72.8	94.7	83.1	62.3	81.5	83.8
PCT Guo et al. (2021)	-	86.4	**85.0**	82.4	89.0	**81.2**	**91.9**	71.5	91.3	88.1	**86.3**	95.8	64.6	95.8	83.6	62.2	77.6	83.7
GDANet Xu et al. (2021)	**85.0**	86.5	84.2	88.0	**90.6**	80.2	90.7	**82.0**	91.9	88.5	82.7	96.1	**75.8**	95.7	83.9	62.9	83.1	**84.4**
PointTransformer Zhao et al. (2021)	83.7	**86.6**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AWT-Net +V (ours)	84.3	85.9	84.3	87.2	88.0	80.3	90.3	81.4	91.6	87.8	82.3	96.0	70.8	95.5	**84.6**	60.1	83.2	83.8
AWT-Net +P (ours)	**85.0**	**86.6**	83.9	**88.4**	88.5	79.2	90.6	81.3	92.5	**88.7**	82.1	**96.3**	73.9	**95.9**	83.5	**63.0**	**84.1**	84.2

Table 3: Segmentation results (%) on ShapeNet Part dataset.

6 Ablation Study

All ablative experiments are conducted using Performer ('+P') and no voting during testing (unless specified).

Module Analysis. We ablate each module in AWT-Net, and the results are listed in Table 4. When the Transformer takes as input the original point cloud and both approximation and detail components, the network achieves the best accuracy with 93.4%. AWT-Net also surpasses the architecture of only using k-NN with 1.1% improvement. Our method ultimately obtains an accuracy of 93.9% with voting tests. This experiment supports our hypothesis that the decomposed approximation and detail components capture different but complementary geometric features to get complete shape representation.

No.	k-NN	self	approximation	detail	voting	accuracy
1						91.4
2	✓					92.3
3		✓				89.8
4	✓	✓				92.4
5			✓	✓		92.9
6	✓		✓			92.5
7	✓			✓		92.7
8	✓		✓	✓		**93.4**
9	✓		✓	✓	✓	**93.9**

Table 4: Results of ablating each module in AWT-Net on ModelNet40. 'k-NN' indicates k-NN aggregation using MLPs, 'self' means self-attention is used in Transformer. 'approximation' and 'detail' denote the input point cloud is fused with features of approximation and detail components through Transformer, 'voting' is the voting strategy during testing.

Method	ModelNet40	OBJ_ONLY	OBJ_BG
random	89.6	82.9	83.8
FPS	92.0	85.3	84.2
app.-det.	93.4	88.0	86.3

Table 5: Classification results of using different point split schemes in our decomposition tree.

Method	Accuracy
Pre-computed	91.8
Dynamic	93.4

Table 6: Impact of dynamic strategy.

Split Scheme. We replace the split of approximation and detail components with random and farthest point sampling (FPS). The results are listed in Table 5, where decomposing point clouds into approximation and detail components gets the highest accuracy. Empirically, our wavelet decomposition strategy localizes points carried with informative geometry instead of noisy points, which improves the robustness against noise.

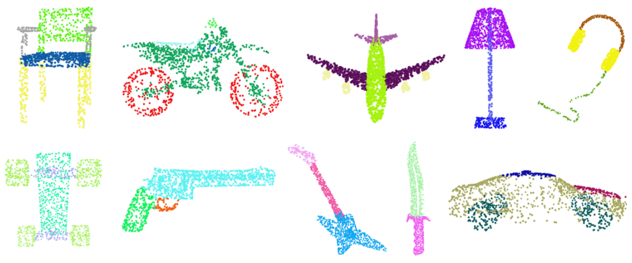
Figure 4: Qualitative results of part semantic segmentation.

Adjacency Matrix Computation. The result of dynamic adjacency matrix calculation in Subsection 4.1 is shown in Table 6. We pre-compute the adjacency matrix on 3D coordinates to split point clouds, which gets the accuracy of 91.8%. Nevertheless, by dynamically calculating the adjacency matrix in high dimensional semantic space at different scales, our network gains 1.6% improvement. Figure 1 shows that our decomposition strategy successfully decomposes points into flat areas and areas containing edges and junction boundaries. Note that we do not provide any supervision *w.r.t.* shape geometry.

Decomposition Levels. We investigate the impact of the number of decomposition levels (*i.e.*, resolution scales), and the results are summarized in Table 7. AWT-Net performs best when the decomposition level is set to 2. Too few scales cannot capture enough geometric details, while more scales introduce redundant information hurting the performance.

Robustness Analysis. First, the robustness of our network on sampling density is tested by using sparser points as the input to ATW-Net trained with 1024 points. Figure **??** lists the results. Although sparser points bring difficulties, ATW-Net still performs consistently across all density settings.

Moreover, Table 8 summarizes the results of rotation robustness, where AWT-Net outperforms other compared methods, especially with 2.4% improvement than the second-best at (s/s).

Last, our model is evaluated on ScanObjectNN Uy et al. (2019) that consists of noisy and deformed scanned objects with non-uniform surface density. Table 2 summarizes AWT-Net gains the comparable accuracy drop from OBJ_ONLY to OBJ_BG, proving that ATW-Net is robust against noise.

7 Conclusion

We propose AWT-Net for 3D shape representation learning. Rooted in wavelet transform, AWT-Net dynamically decomposes point clouds into approximation and detail components at multiple resolution scales, which respectively denotes the low and high sub-bands of point clouds recursively. Another core component is Transformer, which explores the relations between the original complete points and coarse and detail components to provide complementary geometric information for learning shape representations. Extensive experiments show that our method achieves state-of-the-art or comparable performances and is robust to noise.

REFERENCES

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, pp. 213–229. Springer, 2020.

Jingdao Chen, Yong Kwon Cho, and Zsolt Kira. Multi-view incremental segmentation of 3-d point clouds for mobile robots. *IEEE Robotics and Automation Letters*, 4(2):1240–1246, 2019.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pp. 1691–1703. PMLR, 2020.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Ua6zuk0WRH`.

Roger L Claypoole, Geoffrey M Davis, Wim Sweldens, and Richard G Baraniuk. Nonlinear wavelet transforms for image coding via lifting. *IEEE Transactions on Image Processing*, 12(12):1449–1459, 2003.

Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1601–1610, 2021.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems*, 2020.

Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

Ingrid Daubechies and Wim Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, 4(3):247–269, 1998.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *IEEE Access*, 9: 134826–134840, 2021.

Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021.

Stephen Fitzpatrick and Lambert Meertens. An experimental assessment of a stochastic, anytime, decentralized, soft colourer for sparse graphs. In *International Symposium on Stochastic Algorithms*, pp. 49–64. Springer, 2001.

Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. Multi-modal transformer for video retrieval. In *Proceedings of the European Conference on Computer Vision*, pp. 214–229. Springer, 2020.

Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3): 331–371, 1910.

Thomas Hitchcox and James Richard Forbes. A point cloud registration pipeline using gaussian process regression for bathymetric slam. In *2020 IEEE International Conference on Intelligent Robots and Systems*, pp. 4615–4622. IEEE, 2020.

Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1689–1697, 2017.

Drew A Hudson and C Lawrence Zitnick. Generative adversarial transformers. *arXiv preprint arXiv:2103.01209*, 2021.

Le Hui, Hang Yang, Mingmei Cheng, Jin Xie, and Jian Yang. Pyramid point cloud transformer for large-scale place recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6098–6107, 2021.

Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 10433–10441, 2019.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkgNKkHtvB`.

Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 998–1008, 2019.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems*, 31:820–830, 2018.

Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302, 2020a.

Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1800–1809, 2020b.

Lin Liu, Jianzhuang Liu, Shanxin Yuan, Gregory Slabaugh, Aleš Leonardis, Wengang Zhou, and Qi Tian. Wavelet-based dual-branch network for image demoiréing. In *Proceedings of the European Conference on Computer Vision*, pp. 86–102. Springer, 2020.

Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. 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*, volume 33, pp. 8778–8785, 2019a.

Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5239–5248, 2019b.

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8895–8904, 2019c.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1578–1587, 2019.

Sunil K Narang and Antonio Ortega. Lifting based wavelet transforms on graphs. In *Proceedings of Asia-Pacific Signal and Information Processing Association*, pp. 441–444. Asia-Pacific Signal and Information Processing Association, 2009.

Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5618–5627, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32: 8026–8037, 2019.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017a.

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 918–927, 2018.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017b.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 3, pp. 5. Citeseer, 2007.

Michael Ramamonjisoa, Michael Firman, Jamie Watson, Vincent Lepetit, and Daniyar Turmukhambetov. Single image depth prediction with wavelet decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11089–11098, 2021.

Maria Ximena Bastidas Rodriguez, Adrien Gruson, Luisa Polania, Shin Fujieda, Flavio Prieto, Kohei Takayama, and Toshiya Hachisuka. Deep adaptive wavelet network. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 3111–3119, 2020.

Xuejian Rong, Denis Demandolx, Kevin Matzen, Priyam Chatterjee, and Yingli Tian. Burst denoising via temporally shifted wavelet transforms. In *Proceedings of the European Conference on Computer Vision*, pp. 240–256. Springer, 2020.

Raif Rustamov and Leonidas J Guibas. Wavelets on graphs via deep learning. *Advances in Neural Information Processing Systems*, 26:998–1006, 2013.

Fadil Santosa. Second generation wavelets: theory and application. *University of Minnesota, Institute for Mathematics and its Applications*, 6, 2011.

Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4548–4557, 2018.

Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1711–1719, 2020.

Wim Sweldens. Wavelets and the lifting scheme: A 5 minute tour. *Journal of Applied Mathematics and Mechanics*, 76(2):41–44, 1996.

Wim Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, 1998.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pp. 9438–9447. PMLR, 2020.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6411–6420, 2019.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1588–1597, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5463–5474, 2021.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics*, 38(5):1–12, 2019.

Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *International Conference on Learning Representations*, 2018.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920, 2015.

Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4606–4615, 2018.

Mingye Xu, Zhipeng Zhou, and Yu Qiao. Geometry sharing network for 3d point cloud classification and segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 12500–12507, 2020.

Mutian Xu, Junhao Zhang, Zhipeng Zhou, Mingye Xu, Xiaojuan Qi, and Yu Qiao. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3056–3064, 2021.

Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision*, pp. 87–102, 2018.

Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6):1–12, 2016.

Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2282–2290, 2017.

Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 12498–12507, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in Neural Information Processing Systems*, 30, 2017.

Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *Proceedings of the European Conference on Computer Vision*, pp. 528–543. Springer, 2020.

Dong Zhang, Hanwang Zhang, Jinhui Tang, Meng Wang, Xiansheng Hua, and Qianru Sun. Feature pyramid transformer. In *Proceedings of the European Conference on Computer Vision*, pp. 323–339. Springer, 2020.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021.

A Appendix

Number of scales	1	2	3	4
Overall	92.5	93.4	93.1	91.3
Radio	90.0	83.5	75.0	75.0
Wardrobe	75.0	71.0	68.0	65.0
Stool	75.0	78.0	85.0	85.2
Person	85.0	88.5	95.0	100.0

Table 7: Effect of decomposition levels on ModelNet40. Note that the results are achieved without voting during inference.

Method	z/z	s/s
PointNet Qi et al. (2017a)	81.6	66.3
PointNet++ Qi et al. (2017b)	90.1	87.8
SpiderCNN Xu et al. (2018)	83.5	69.6
DGCNN Wang et al. (2019)	90.4	82.6
GDANet Xu et al. (2021)	91.2	90.5
AWT-Net (ours)	91.4	90.2

Table 8: Accuracy comparisons of rotation on ModelNet40. z/z: both training and test samples are augmented by random rotation along the z axis; s/s: both training and test samples are augmented by random rotation along (x,y,z) three axis.

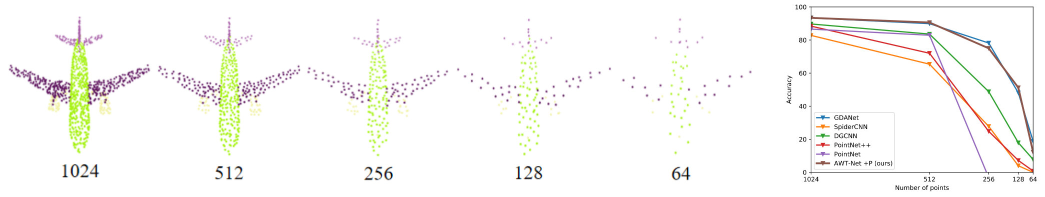

Figure 5: Density robustness test. (Left). Point cloud with random point dropout. (Right). Test results on ModelNet40 of using sparser points as the input to a model trained with 1024 points.

A.1 Execution time and model parameters

Model	**Number of Params**	**Time**
GDA-Net Xu et al. (2021)	936,616	1.057
AWT-Net (scales 1)	965,864 / 966,120	0.851 / 0.843
AWT-Net (scales 2)	**1,170,216 / 1,170,728**	**1.200 / 1.191**
AWT-Net (scales 3)	1,382,760 / 1,383,528	1.670 / 1.657
AWT-Net (scales 4)	1,603,496 / 1,604,520	2.036 / 2.024

Table 9: The number of parameters and computational overhead (time) comparison. For our model, we list the statistics of both using the vanilla Transformer and Performer as (left / right). The computational overhead is measured in second for a single forward and backward pass. All results are average over ten trails.

In this subsection, we compare the number of parameters and computational overhead *w.r.t.* GDA-Net Xu et al. (2021) on ModelNet40 and the results are listed in Table 9. We measure four scales. *Note that GDA-Net consists of two building blocks, which is fairly comparable to "scale 2" in our model.* In this case, though our model contains around 25% more parameters, the execution time is only 13% (+P) and 15% (+V) slower than GDA-Net, which shows the practicability of our model. We also notice that Performer is faster than the vanilla Transformer, which is consistent with our motivation.

A.2 Randomness in even-odd node split

Number of scales	**1**	**2**	**3**	**4**
Even-odd split	92.5	93.4	93.1	91.3
Random split	91.7	92.9	92.6	90.9

Table 10: Classification results of using even-odd split and random split in Algorithm 1.

No.	Configuration
1	Ablate U, P and MLPs in GC.
2	Ablate U, P.
3	Ablate U, P, and MLPs in GC; set K, Q, V in Transformer to z.
4	Ablate U, P; set K, Q, V in Transformer to z.
5	Ablate MLPs in GC.
6	Ablate U.
7	Ablate P.
8	Use the full model without voting.
9	Use the full model with voting.

Table 11: A description of configurations in module analysis shown in 4. Refer to Figure 3 for the notations.

In this subsection, we compare the effect of using even-odd split and random split in Algorithm 1 and the results are listed in Table 11. Using random split performers slightly worse than even-odd split. With even-odd split, we can separate any two connected nodes into two different sets to the best extent, thus ensuring each local area contains both even and odd nodes. However, with random split, some local areas might contain only one type of nodes, thus failing to be decomposed properly.

A.3 Configurations of Module Analysis

In this subsection, we provide a description of the configurations of module analysis in Table 4.