Computer Aided Geometric Design 91 (2021) 102038

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Multi-resolution 3D CNN for learning multi-scale spatial features in CAD models

Sambit Ghadai, Xian Yeow Lee, Aditya Balu, Soumik Sarkar, Adarsh Krishnamurthy *

Iowa State University, Ames, IA, USA

ARTICLE INFO

Article history:
Available online 27 September 2021

Keywords:
Object recognition
Multi-resolution CNN
Multi-level voxel representation
Interpretability
Deep learning

ABSTRACT

Learning multi-scale spatial features from 3D spatial geometric representations of objects such as point clouds, 3D CAD models, surfaces, and RGB-D data can potentially improve object recognition accuracy. Current deep learning approaches learn such features using structured data representations such as volume occupancy grids (voxels) and octrees or unstructured representations such as graphs and point clouds. Structured representations are generally restricted by their inherent limitations on the resolution, such as the voxel grid dimensions or the maximum octree depth. At the same time, it is challenging to learn directly from unstructured representations of 3D data due to non-uniformity among the samples. A hierarchical approach that maintains the structure at a larger scale while still accounting for the details at a smaller scale in specific spatial locations can provide an optimal solution for learning from 3D data. In this paper, we propose a multi-level learning approach to capture large-scale features at a coarse level (for example, using a coarse voxelization) while simultaneously capturing flexible sparse information of the small-scale features at a fine level (for example, a local fine-level voxel grid) at different spatial locations. To demonstrate the utility of the proposed multi-resolution learning, we use a multi-level voxel representation of CAD models to perform object recognition. The multi-level voxel representation consists of a coarse voxel grid containing volumetric information of the 3D objects and multiple fine-level voxel grids corresponding to each voxel in the coarse grid containing a portion of the object boundary. In addition, we develop an interpretability-based feedback approach to transfer saliency information from one level of features to another in our hierarchical end-to-end learning framework. Finally, we demonstrate the performance of our multi-resolution learning algorithm for object recognition. We outperform several previously published benchmarks for object recognition while using significantly less memory during training.

1. Introduction

A three-dimensional object comprises of different multi-scale features inherent to its geometry and its overall shape. Considerable efforts have been made in shape detection & searching (Iyer et al., 2005; Schnabel et al., 2007; Tangelder and

* Corresponding author.
E-mail addresses: sambitg@iastate.edu (S. Ghadai), xylee@iastate.edu (X.Y. Lee), baditya@iastate.edu (A. Balu), soumiks@iastate.edu (S. Sarkar), adarsh@iastate.edu (A. Krishnamurthy).

https://doi.org/10.1016/j.cagd.2021.102038
0167-8396/

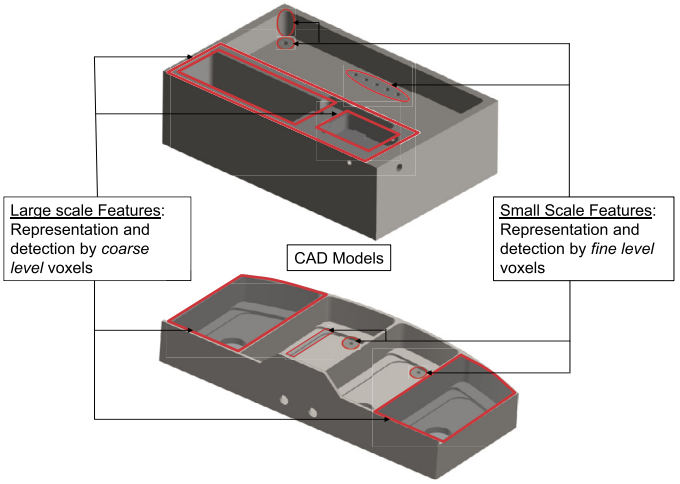

Fig. 1. An illustrative example showing the need for multi scale feature detection in two mechanical parts (CAD models). The CAD models have a multitude of features, each on a separate spatial scale, which create challenges in specific feature detection.

Veltkamp, 2004; Veltkamp, 2001) and feature detection (Elinson et al., 1997; Lowe, 2001; Tombari et al., 2011) from 3D objects, and CAD models for various applications such as design, manufacturing, and analysis (Cardone et al., 2003; Jayanti et al., 2006; Joshi and Chang, 1988; Kyprianou, 1980; Ramesh et al., 2001). A major research area in computer vision is representing 3D spatial data and extracting meaningful information or features using deep neural networks. Previous works have proposed learning from 3D data by projecting the 3D information to 2D or 2.5D (depth inclusion) images, thereby extending image recognition principles to 3D object recognition (Song and Xiao, 2016; Su et al., 2015; Wu et al., 2015). Researchers have also demonstrated object recognition from multiple 2D views of the 3D object (Kanezaki et al., 2018; Li et al., 2018; Qi et al., 2016; Sfikas et al., 2017). Though this approach is effective in many applications, including 3D reconstruction, some spatial relationships among the features may be lost, and this makes it infeasible for certain problems such as graphics rendering (Tatarchenko et al., 2017), point cloud labeling (Qi et al., 2017), design and manufacturing (Ghadai et al., 2018), etc. However, a major limitation of directly using 3D data with deep neural networks is the high memory requirement. The presence of abundant information in spatial data coupled with large data requirements for efficient training of deep learning algorithms renders this task impractical for high-resolution 3D data. On the other hand, since the resolution of data representation directly influences the memory requirement, reducing memory usage results in low fidelity feature extraction and detection. Hence, new efficient and scalable feature learning techniques are required to enable better detection, classification, and feature analysis of CAD models.

Effective utilization of high-resolution 3D CAD data for object classification using deep learning requires developing new data representations and novel deep learning architectures. The learning algorithm needs to preserve spatial localization while learning hierarchical features from data. Therefore, convolutional neural networks (CNNs), which have been proven to be effective for 3D spatial data (Ioannidou et al., 2017; Maturana and Scherer, 2015) are the natural candidates. However, training CNNs using uniform data representations (such as voxelized representation) becomes inefficient when the spatial features exist on different scales since the uniform data representation cannot effectively accommodate this non-uniformity (Abdul-Rahman and Pilouk, 2007). This concept is illustrated in Fig. 1, where the CAD models have features such as holes, slots, and pockets pertaining to different spatial scales. Thus, there is a need to represent CAD data hierarchically and build an efficient deep learning architecture to learn features from each level of the hierarchy. In addition, the data representation needs to be both memory-efficient and preserve the spatial relationship in the data. Similar to the learning paradigm of CNNs, where the first few layers learn fine-level features and final layers learn coarse-level features, hierarchical learning can enable extraction of key features from a fine level of the hierarchy and then merge the understanding of the small scale aspects with the coarse level features. Such an approach will also exploit the sparse nature of the spatial data better. Extending this idea explicitly, we use a multi-level voxel representation of CAD models representing 3D models in a more memory-efficient manner.

One of the primary requirements for the efficient functioning of a deep neural network, especially CNNs, is a large representative labeled dataset that facilitates object classification. To demonstrate the main concept of our paper, we use the open-source ModelNet dataset (Wu et al., 2015), consisting of an extensive collection of 3D models, to perform object classification tasks based on the features of each class of models in the dataset. This follows along with the concept that 3D objects can be segregated based on their feature-based similarity (Kriegel et al., 2003), as well as geometry-based similarity (Keim, 1999; Seidl and Kriegel, 1998).

Table 1
Comparisons between different spatial deep learning approaches. Our approach (MRCNN) retains some of the advantages OctNet of such as memory efficiency while still having a flexible data structure. MRCNN also enables spatial multi-scale learning from multi-resolution data.

Method	Data Representation	Data Structure	Memory Efficient	Spatial Multi-scale Learning
VoxNet	Voxels	Structured	×	×
PointNet++	Point cloud	Unstructured	×	×
OctNet	Octree	Structured	✓	×
MRCNN	Multi-level voxels	Structured-flexible	✓	✓

In this paper, we present a novel approach to enable hierarchical learning of features at different scales and thus perform object classification from a flexible multi-level voxel representation of 3D CAD data using deep learning. We achieve this by adopting the multi-level voxelization framework developed by Young and Krishnamurthy (2018), which generates a multi-level voxel grid structure (see Table 1 for comparisons between different spatial deep learning approaches). The 3D object is represented using a binary occupancy grid at two levels with independent user-defined resolutions at each level. This is to ensure that CAD model features at different levels are effectively captured in the data representation. Based on the scale of features, the size and number of voxels are generically selected. We developed a multi-level CNN that efficiently learns from this multi-level data representation by extending the same idea to the deep learning algorithm. This paper is an extension of our previous CVPR workshop paper (Ghadai et al., 2019). However, in this paper, we have devised an interpretability-based feedback approach to enable the flow of salient information from the fine level to the coarse level. The specific contributions of this paper include:

1. A framework for learning from a hierarchical multi-scale representation of 3D CAD data, where there are two levels of information, a coarse level and a fine level. The coarse level description of features in the spatial data can give an overall understanding of the object, while some important finer level features could improve the learning by augmenting the coarse level description.
2. A new method of connecting the multiple learning levels via salient information flow between them. We develop the connection between the two levels of the network using feedback from the coarse level to the fine level provided by interpretability mechanisms such as GradCAM (Gradient of the Class Activation Map) regarding important spatial regions.
3. Demonstration of the methods mentioned above on a multi-level voxel representation used for object classification of ModelNet10 and ModelNet40 datasets. We achieve similar accuracy of results as previously reported while using considerably lower memory during training.

The paper is arranged as follows. In Section 2, we discuss a few relevant works in the field of learning from 3D data using different data representations. In Section 3, we describe the multi-resolution representation of 3D data using multi-level voxelization, which we generate using a GPU accelerated voxelization algorithm. We explain the Multi-Resolution Convolutional Neural Network (MRCNN) architecture that effectively learns the features from the multi-level voxel representation in Section 4. Finally, in Section 5, we present the results from evaluating the MRCNN on multi-level voxel data to classify objects in the ModelNet10 dataset and also explain the effectiveness of MRCNN to learn from 3D data with reduced memory usage.

2. Related work

Learning from spatial data has been an active research topic, and researchers have developed several approaches to address this challenge. Most of the approaches can be categorized into two main learning methodologies. The first category of approaches is based on learning from unstructured spatial data such as point clouds (Charles et al., 2017; Klokov and Lempitsky, 2017; Qi et al., 2017; Roveri et al., 2018), meshes (Boscaini et al., 2016; Fey et al., 2018; Masci et al., 2015) and graphs (Bronstein et al., 2017; Kovnatsky et al., 2013; Wang et al., 2018a). The second class of methods use structured spatial data such as voxel grids (Ma et al., 2018; Maturana and Scherer, 2015; Wu et al., 2016, 2015; Xu and Todorovic, 2016), octrees (Riegler et al., 2017; Tatarchenko et al., 2017; Wang et al., 2017), RGB-D images (Couprie et al., 2013; Liu et al., 2019; Socher et al., 2012), etc. Choy et al. (2019) developed a 4D CNN using Minkowski operations for object recognition from 3D-videos and LIDAR scans. A 3D attention-based recurrent neural network has been developed by Liu et al. (2019) to target the best view of an object for object recognition. Various other data representations for geometric deep-learning are being explored recently (Xiao et al., 2020). Using general deep learning methods (such as CNNs) to learn from unstructured spatial data is challenging since many operations require a structured input. Our work is focused on a sub-class of structured methods that use volumetric representation (voxels, octrees) to learn from spatial data while also being flexible in terms of the data structure. Our multi-level voxel data structure makes use of user-defined voxel resolution at each level, making it more flexible than the octree data structure (where each voxel is divided exactly into 2^3 sub-voxels). This allows us to achieve very high effective resolutions using only two levels while retaining better memory efficiency.

2.1. Multi-level voxel learning

Learning from 3D voxel data was initially explored by Wu et al. (2015) (3DShapeNet) using convolutional deep belief networks (CDBNs) and later by Maturana and Scherer (2015) (VoxNet) using 3D-CNN. It is challenging to achieve a good classification accuracy on the ModelNet10 and ModelNet40 datasets using these approaches. Most approaches that use voxel data use a maximum of 32^3 resolution; recently, with the increase in GPU memory, using a resolution of 64^3 is possible. However, increasing the resolution beyond that is not practical with current systems for training deep networks using dense voxel grids for large datasets.

Increasing the effective voxel resolution beyond 64^3 requires new specialized data structures. Tatarchenko et al. (2017) and Riegler et al. (2017) developed OctNets, which leverages an octree-based voxel representation of the data. In a tree structure, each voxel is represented as a node, and each of those nodes is connected to exactly eight subdivided voxels or octants. However, traversing the octree structure requires recursive algorithms. OctNets solved this problem by making use of shallow octree data stored using a regular grid. This allowed them to directly index the data in the octree without recursively parsing the tree and achieve an effective resolution of 256^3. In this work, we use a different strategy of having multiple voxelization levels with arbitrary resolutions at each level. With this approach, we can also achieve a very high resolution similar to those achieved using OctNets. For example, we can achieve an effective resolution of 256^3 by having a coarse resolution of 64^3 and a fine resolution of 4^3.

2.2. Feedback from interpretable attention maps

Convolutional Neural Networks (CNNs) (He et al., 2016; Huang et al., 2017; Krizhevsky et al., 2017; Lecun et al., 1998) are the preferred vision detectors when it comes to deep learning-based computer vision such as image classification, segmentation, and detection. However, CNNs are considered to be black boxes since they do not provide insights into the workings of the feature extractors and the hidden layers. To overcome this setback, recent research has been dedicated to unlocking this black box nature of CNNs for more explainable/interpretable insights (Zhang et al., 2018b). Most of the interpretability methods (Olah et al., 2017; Ribeiro et al., 2016; Selvaraju et al., 2017; Zeiler and Fergus, 2014; Zhang et al., 2018a) are based on understanding a pre-trained network; although, a few works present disentangled/interpretable representations of neural networks that produce self-explanatory graphs. Using these disentangled representations, researchers have developed middle-to-end learning models that use weak to strong supervisory inputs from interpretable models to enhance learning. These supervisory inputs are in the form of feedback cues, such as active question-answering and object annotations, to enable hierarchical object understanding (Zhang et al., 2017), one-shot/multi-shot learning, etc. In this paper, we show the use of such an interpretability mechanism to augment 3D-CNNs with feedback based on attention maps. According to the multi-level data representation we use in our work, this feedback mechanism is used to bridge fine-level learning to coarse-level learning.

3. Multi-level voxelization

In this section, we briefly describe the GPU-accelerated algorithm (Young and Krishnamurthy, 2018) we used to generate a multi-level voxelization of a boundary representation (B-rep) CAD model. A solid model, $\mathcal{S}$ is often represented using the B-rep surfaces $d\mathcal{S}$. Whereas, the multi-level voxelization is a binary occupancy grid having two major components, namely, *coarse-level voxelization* G_c and *fine-level voxelization* G_f. Note that, while our approach can be abstracted for multiple levels of features, here in this paper, we consider only two levels of representation. Our experiments show that these two levels are sufficient for representing the fine features in the geometry.

Multi-resolution voxel representations of B-rep CAD models are shown in Fig. 2 along with corresponding coarse and dense voxel grids. At each level, the voxel occupancy is represented using a binary value of 0/1 (i.e. $G = \{0, 1\}^d$, where d refers to the total number of voxels in the grid) that defines whether the object occupies a voxel or not. To create the coarse-level voxelization, a standard grid of voxels using a user-defined resolution is constructed in the region occupied by the object (denoted by the axis-aligned bounding box (AABB)). A *triangle-box* intersection test is used in the next step to identify the boundary voxels by checking the intersection of each triangle of the B-rep model with every voxel. Identifying the boundary voxels enables a further division of the coarse-level grid into a fine resolution only at the boundary of the object without cubically increasing the voxel count. Once the boundary voxels are identified, they are further sub-divided to construct the fine-level voxel grid only at the boundary locations. The same *triangle-box* intersection test is then used to identify the fine-level voxels that intersect with the triangles of the B-rep model. The complete voxelization framework is GPU-accelerated, where each *triangle-box* intersection test is computed in parallel at individual levels. Using this method on the GPU, a high-resolution (effective resolution of 128^3) multi-level voxelization of a low-mid polygon count model can be generated in about 180 ms.

A prefix sum address array is created using *exclusive prefix sum* that maps the memory location of each coarse-level boundary voxel to its corresponding fine-level voxel grid as shown in Fig. 3. We make use of a flat data structure to store the multi-level voxelization. This data structure consists of unrolling the 3D voxels first along the x-direction, followed by y- and z-direction, sequentially at the coarse level. We explain the details of the data structure for a 2D example of a two-level voxelization using Fig. 3. As shown in the figure, the Coarse level voxelization consists of a 3×3 voxel grid unrolled

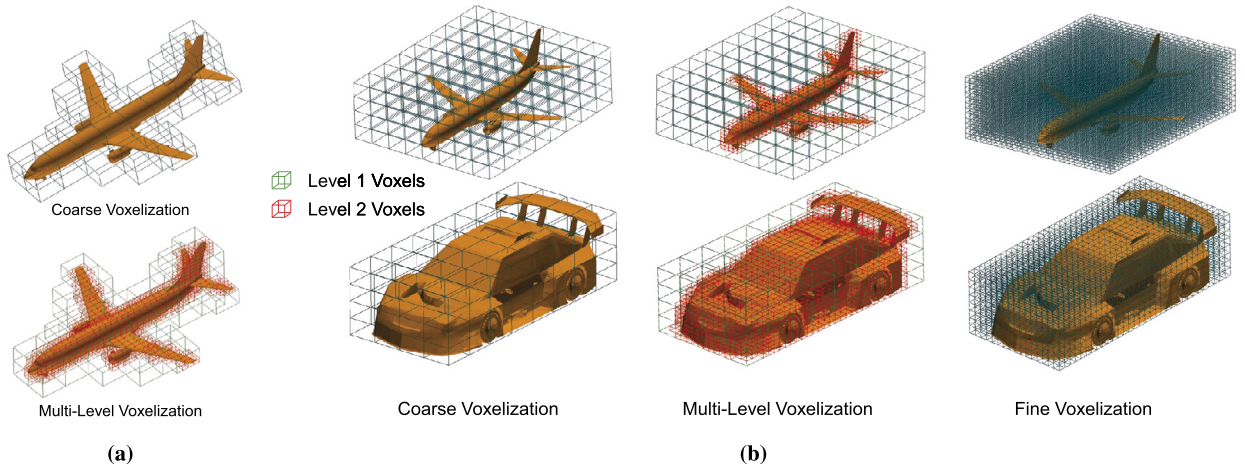

Fig. 2. Multi-level voxelization of B-rep CAD models. The fine level voxelization is performed only near the boundaries of the coarse level voxelization. The final resolution is equivalent to having dense level voxels throughout the model. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

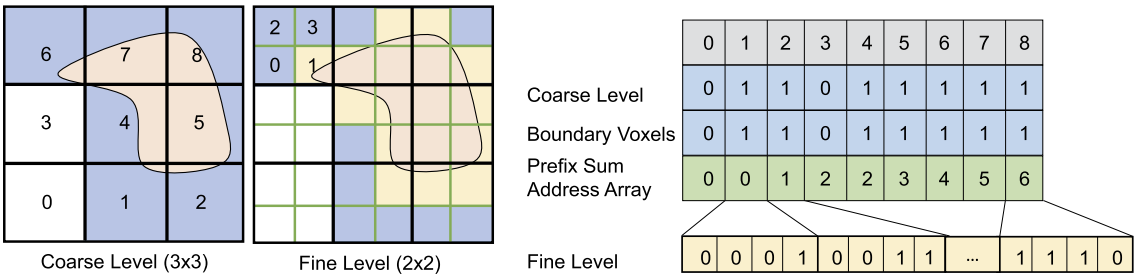

Fig. 3. Data structure for storing the multi-level voxelization.

along the x-direction first and then along the y-direction. Out of the total 9 voxels, 7 are occupied and are identified using the value "1" in the Coarse Level voxelization. All 7 voxels intersect with the boundary of the object; these voxels are further subdivided in the fine-level voxelization and are identified by "1" using the Boundary Voxels array. To efficiently store the information only from the boundary voxel, we use the Prefix Sum Address Array. The exclusive prefix sum array is created using the *scan* algorithm (Blelloch, 1989) on the Boundary Voxels array. It consists of the sum of the values of the elements until the current element, which gives a unique index location for each boundary voxel. In the present example, each boundary voxel is further divided into a 2×2 grid of fine level voxels. The 4 values of the fine level array for each boundary voxel are then stored in a single flat array that consists of a total of 28 values.

Representing a 3D model using a multi-level voxel grid structure exploits the sparse nature of the data. When spatial data is represented using voxel grids, the voxel count increases as $\mathcal{O}(n^3)$ with an increase in resolution of the grid and in turn reduces the total voxel space occupied by the object. To prove this, we represent the coarse-level voxel grid by G_c and its corresponding fine-level voxel grids by $G_f = \{G_f^1, G_f^2, \ldots, G_f^{\phi_b}\}$ with resolutions of $n_c \times n_c \times n_c$ and $n_f \times n_f \times n_f$, respectively, where the number of boundary voxels are ϕ_b. Comparing it with a dense voxel grid of resolution $n_d \times n_d \times n_d$ where $n_d = n_c \times n_f$, we see that, total voxels in multi-level data structure are $\phi_b \times n_f^3$. As the number of boundary voxels can never be more than the total voxel count, $1 \leq \phi_b \leq n_c^3$. Define

$$G_c \rightarrow n_c \times n_c \times n_c$$
$$G_f \rightarrow n_f \times n_f \times n_f$$
$$n_d = n_c \times n_f$$

From the above,

$$1 \leq \#\text{Boundary Voxels}, \phi_b \leq \text{Total Voxels}, n_c^3$$
$$\therefore \phi_b \times n_f^3 \leq n_c^3 \times n_f^3 = n_d^3$$
$$\Rightarrow \text{Voxel count}_{\text{multi}} \leq \text{Voxel count}_{\text{dense}}.$$

The number of boundary voxels for a fixed-genus closed object scales with $\mathcal{O}(n^2)$ with the resolution n. In practical cases, this scaling is much smaller than the volumetric scaling of voxels, leading to a compression in the data structure without losing any accuracy in the representation. In addition, the spatial locations of the locally dense fine level voxels can also potentially aid in learning the shape of the object at the coarse level.

4. Multi-resolution CNN

In the previous section, we showed that representing 3D spatial data of a geometry, $\mathcal{S}$, using a flexible multi-level data structure exploits the sparse nature of the data and is memory efficient. The multi-level voxel data stores the volume occupancy information in two hierarchical levels, the coarse voxel grid G_c and a set of voxel grids, G_f. In this section, we explain the multi-resolution hierarchical learning algorithm that enables end-to-end learning of individual features from both the coarse and fine voxel grids. Further, we investigate an interpretability-based feedback mechanism for adaptively extracting voxel zones that bridge the learning from the fine to the coarse level.

The multi-resolution convolutional neural network (MRCNN) consists of multiple 3DCNN networks, with each network dealing with features from the corresponding voxel grid. For example, with two levels of resolutions, each level has a 3DCNN network that learns the features commonly present in that resolution. One of these 3D CNNs, named as *Coarse-level CNN*, takes in the coarse level voxels G_c as input, and the other CNN called *Fine-level CNN* accepts the set of fine level voxels, G_f as input. Let θ_1 be the set of weights for the *Coarse-level CNN* and let $F(G_c, \theta_1)$ be the predicted output for a given coarse-level input G_c and θ_1. This provides a benchmark of the network's performance with some loss ϵ in the prediction. We then augment the performance of the network by adding *Fine-level CNN* with another set of weights, θ_2, which are used to learn features from G_f. These two neural networks are carefully combined to work together as a single unit in both the forward pass and backward pass of the algorithm so that it facilitates better learning from the multi-level data representation and makes it computationally comparable to performing sparse convolutions. The main challenge in this task from a geometric perspective is that all the operations we perform on the *Fine-level CNN* have to be integrated or appended to the operations performed in the *Coarse-level CNN*. Further, since the deep learning models are trained using the back-propagation algorithm, the operation we construct must be differentiably (Baydin et al., 2018; Wang et al., 2018b) defined. Therefore, we define the forward and backward pass in the next two sections to address this challenge.

4.1. Forward computation of MRCNN

Recalling the voxel data representation, a 3D object is represented as a grid of coarse voxels (say 8^3 resolution) with a binary voxel value of 0/1 at the boundary of the object. Each boundary voxel (with coarse voxel value of 1) is further subdivided into a fine voxel grid (say 4^3 resolution) with similar binary values 0/1.

The forward computation of MRCNN starts from the fine-level voxel grid by randomly sampling a subset of the total boundary voxels, Φ, of a 3D model. This subset of boundary voxels with individual fine voxel grid G_f of resolution ${n_f}^3$, are used as input to the *Fine-level CNN*. The *Fine-level CNN* network consists of blocks of *convolution - max pooling* layer pairs and *fully connected* layers connected in conjunction, each with a ReLU activation function associated with it. The *Fine-Level CNN* then outputs a single real numbered value $\eta_\phi \forall \phi \in \Phi, \eta_\phi \in \mathbb{R}$ for each boundary voxel, ϕ. This set of η_ϕ values are sent forward to be combined with the coarse-level voxel grid, G_c. The outputs of *Fine-level grid*, η_ϕ, replace the original coarse voxel grid values at the appropriate voxel positions, $G_c(\phi) = \eta_\phi$. This is performed with the help of the index array (obtained from the prefix sum array) that maps the position of each coarse-level boundary voxel with its respective fine-level voxel grid.

In the corresponding phase of the forward computation of MRCNN, the coarse-level voxel grid with selective embeddings from *Fine-level CNN*, G_c' (G_c, with specific values replaced with η_ϕ), is used as an input to the *Coarse-level CNN*. The architecture of *Coarse-level CNN* network comprises of different permutations of *convolution - max pooling* layer blocks. The end section of the network has multiple *fully connected* layers, and the output is the class prediction probability vector. The categorical cross-entropy loss function is used to compute the loss of the predicted classes with the true class labels. The forward run of the MRCNN network algorithm is depicted in Algorithm 1. Network hyperparameter details are provided in the online Appendix.

4.2. Backward computation of MRCNN

While the forward computation of MRCNN may be trivial, the back-propagation of the MRCNN can be tricky to implement correctly. The main challenge of the back-propagation computation is to link the two networks such that the gradients can be passed on from the coarse level network to the fine level network. Without this link, the weights of the fine-level network would not be updated accordingly.

For backpropagation, we begin with the final loss between the y_{pred} and y_{true} obtained as an output from the coarse level network using the categorical cross-entropy loss. We back-propagate this loss through the coarse level network using standard automatic differentiation packages. Thus, from the loss L, we can define the derivatives for weights in each layer of the coarse level (i.e., $\partial L/\partial \theta_1$). Finally, using the same backpropagation scheme, we can obtain the gradients of the loss L

Algorithm 1: MRCNN Forward & Backward Passes.

Forward Pass:
  forall *Boundary voxels,* ϕ_b **in parallel do**
    $\eta_b = Forward_{Fine-levelCNN}(\vartheta_{2_b})$;
    $\vartheta_1(\phi_b) = \eta_b$
  end
  $y_{pred} = Forward_{Coarse-levelCNN}(\vartheta_1)$
Backward Pass:
  $d\vartheta_1 = Backward_{Coarse-levelCNN}(\vartheta_1, dy_{pred})$
  forall *Boundary voxels,* ϕ_b **in parallel do**
    $d\eta_b = d\vartheta_1(\phi_b)$
    $d\vartheta_{2_b} = Backward_{Coarse-levelCNN}(\vartheta_{2_b}, d\eta_b)$;
  end

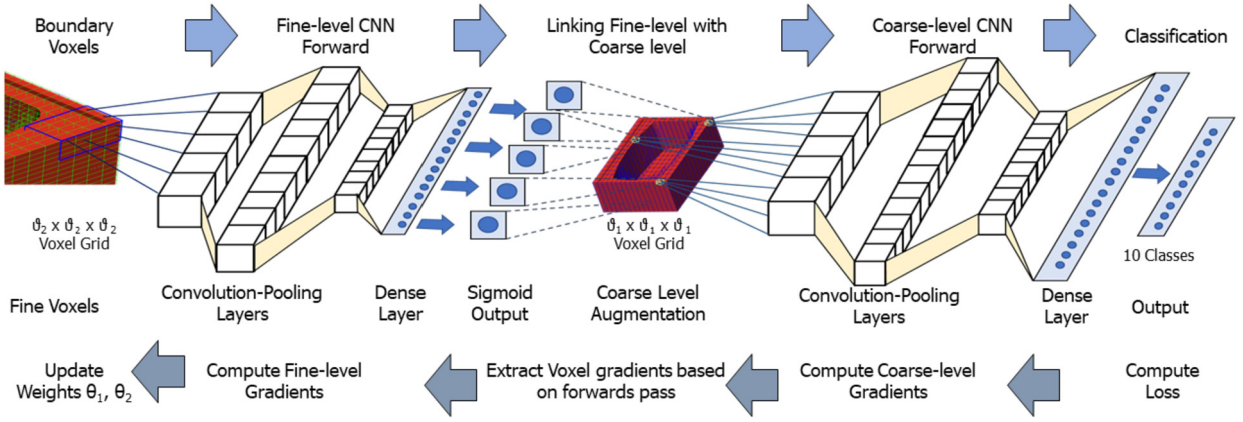

Fig. 4. Multi-Resolution Convolutional Neural Network (MRCNN). Our proposed network can learn from a hierarchical CAD data representation with a coarse level of information and information of boundary voxels which connects to the fine level voxels. For a forward pass (left to right) the information learned from selected fine level voxels using the L2 CNN is embedded in the coarse level input to L1 CNN and then the final prediction is obtained. The backward pass follows the reverse order of the forward pass (right to left).

with respect to the input G_c' (modified coarse level input with selective fine level embedding). Recall that we embedded the features from the fine voxel grid to G_c' using the variable η_ϕ.

$$\frac{\partial L}{\partial \eta_\phi} = \frac{\partial L}{\partial G_c'}(\phi) \tag{1}$$

This linking of derivative from G_c to η_ϕ is obtained using the index array obtained from the prefix sum. Thus, we get the derivative of the loss with respect to the output of the fine level network (η_ϕ) and use it to back-propagate through the network to obtain the gradients of the fine level network. This process is explained in Algorithm 1 and Equation (2) shows the back-propagation of the loss from the output level to the fine level network.

$$\frac{\partial L}{\partial \theta_2} = \sum_{\forall \phi \in \Phi} \frac{\partial L}{\partial \eta_\phi} \frac{\partial \eta_\phi}{\partial \theta_2} \tag{2}$$

Using Equation (1), we can now define the backpropagation process completely. It is also worth noting that since the same *Fine-level CNN* is shared among all the boundary voxels, the gradients of θ_2 for *Fine-level CNN* are computed for all boundary voxels only once. With that gradient, the network could be trained to update its weights θ_1 and θ_2 in such a way that the loss L, of the final prediction y_{pred}, is minimized. This approach facilitates the end-to-end learning of network parameters θ_1, θ_2 in a hierarchical order. The network parameters' update could be performed using an optimizer, such as SGD, Adam, Adadelta, etc. The complete operation of MRCNN is explained schematically in Fig. 4.

4.3. Adaptive salient-voxel-zone feedback via interpretable attention maps

In the forward computation of MRCNN as discussed in Section 4.1, we randomly sample Φ boundary voxels from coarse-level voxel grid as the input to the *Fine-level CNN*. We do so to reduce the computational time incurred in using all the boundary voxels. From our experiments, we found that using all the Φ voxels did not improve the learning capability of MRCNN substantially ($\approx 0.2\%$). However, we note the Φ voxels sampled randomly might not be the perfect candidates that carry important spatial information in the fine-level grid. Hence, we propose an adaptive *salient-voxel-zone* (SVZ) selection

Algorithm 2: Feedback$_{\text{MRCNN}}$.

$\theta_1 \leftarrow Pretrain_{Coarse-levelCNN}$
forall *Epochs* **do**
  $\gamma = 3DGradCAM(\vartheta_1(\Phi), \theta_1)$
  $y_{pred} \leftarrow Forward_{MRCNN}(\vartheta_2(\gamma(\phi_b)), \vartheta_1)$
  $L = Cross - entropy(y_{true}, y_{pred})$
  $\theta_1, \theta_2(Update) \leftarrow Backward_{MRCNN}(Loss)$

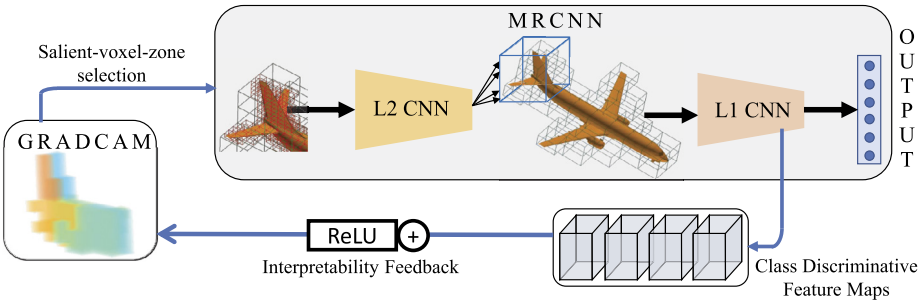

Fig. 5. Interpretability based feedback for Multi-Resolution CNN. The output is first evaluated with just the coarse voxels using L1 CNN. The interpretability of the L1 CNN provides feedback salient features to select voxel-zones (SVZs) to train MRCNN as explained in Algorithm 2.

technique that samples the coarse-level boundary voxels based on an interpretability algorithm. These Φ_{int} voxels, a fraction of the total Φ boundary voxels, are input to the *Fine-level CNN*; with the rest of the forward and backward computations of MRCNN similar to the algorithm discussed in Algorithm 1. Note that this approach is not to get better performance but to get more physically interpretable models. In this section, we explain the process of using interpretability as feedback to MRCNN for an intelligent sampling of boundary voxels from the coarse-level grid.

The interpretability mechanism adopted to detect important boundary voxels is an enhancement of GradCAM (Selvaraju et al., 2017) for 3D objects (3DGradCAM (Ghadai et al., 2018)). 3DGradCAM backpropagates the gradients of class-specific feature maps, retrieved from the CNN, with respect to the loss prediction to generate class activation maps (CAM) that estimate the importance of regions in the input 3D object. The feature maps are first obtained from the CNNs, and a class discriminative gradient is computed with respect to these feature maps. Spatially important feature maps are thus obtained for a particular class. Mapping the CAM to the input voxel grid allows us to extract regions, which we call *salient-voxel-zones*, from the coarse-voxel grid.

In the first step to extract SVZ, we use a pre-trained model of *Coarse-level CNN*, trained on coarse level voxel grid ϑ_1 to generate the CAMs, γ, using 3DGradCAM. To overlay the CAM to the input data, we interpolate it to match the coarse level voxel grid size. This acts as an attention map of the voxel grid to interpret the important voxels contributing to the classification. The relevant voxels interpreted from γ have higher activation than the non-relevant voxels. We exploit this factor to choose $\gamma(\Phi_b)$ voxels from the total boundary voxels Φ to be used as input to *Fine-level CNN*. Then, the forward computation of MRCNN proceeds, as discussed in Section 4.1, to classify the object using the network parameters θ_1, θ_2. After computing the classification loss L, the backward computation of MRCNN updates the network parameters θ_1, θ_2 by back-propagating L to the input layer of *Fine-level CNN*. This constitutes one training iteration (or epoch) of MRCNN. The next epoch again begins with selecting salient-voxel-zones using 3DGradCAM with the updated parameter θ_1 instead of the pre-trained parameters. The complete process of incorporating the interpretability-based feedback system is shown in Fig. 5 and explained in Algorithm 2.

5. Experimental results & discussion

In this section, we present the classification results of the proposed MRCNN framework on Princeton's ModelNet10 dataset (Wu et al., 2015) that contains 3D CAD models of 10 different categories. The 3D CAD models were voxelized using the voxelization scheme mentioned in Section 3, yielding a set of coarse voxel grid of 8^3 and another set of dense voxel grid of 32^3. In addition, we also voxelized two sets of multi-resolution data to test the efficacy of MRCNN; an 8^3 coarse voxel grid with a 4^3 fine voxel grid giving an effective resolution of 32^3 resolution and a 32^3 coarse voxel grid with a 4^3 fine voxel grid, resulting in an effective resolution of 128^3. We ran a set of experiments on the four different resolutions of the data and compared the classification performance between a *Coarse-Level CNN* applied on the coarse and dense resolution data and MRCNN applied on the multi-resolution data. For the coarse and dense resolutions, we applied the *Coarse-level CNN* on the data and evaluated the classification performance of the network. For the multi-resolution representation, we applied our proposed MRCNN by sampling 40% of SVZ, computed by 3DGradCAM, and allowed the fine resolution voxels of these coarse voxels to run on *Fine-level CNN*. We selectively embed these values in the coarse level boundary voxels and continue the forward pass as explained in Section 4.1. We find that sampling 40% of boundary voxels gives a good classification

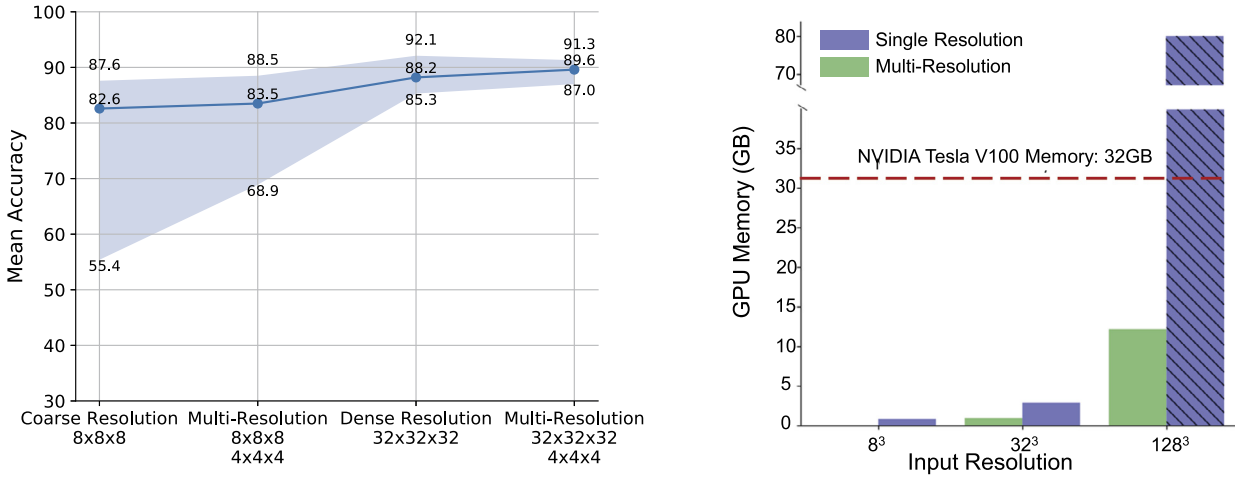

Fig. 6. (a) Mean classification performance with different input resolutions on ModelNet10 dataset. The coarse and dense resolutions are trained with a conventional 3DCNN, while the multi-resolution voxel grids are trained with MRCNN. (b) GPU memory usage of MRCNN training & equivalent CNN training on specified voxel grid resolutions. The red horizontal line shows the current prominent GPU capacity. Blue hatched bar depicts the anticipated memory usage while training a 128^3 dense voxel grid on CNN.

Table 2
Comparison of deep learning frameworks with voxel-based representation for ModelNet10 object recognition. MRCNN (our method), along with OctNet, can learn from a voxel grid resolution of 128^3, so we compare our work with OctNet for better clarification. We also show the performance of other voxel-based spatial learning methods. * represents value interpreted from plot.

Method	Data Representation	Accuracy %
MRCNN	**Multi-level voxels**	**91.3**
OctNet (Riegler et al., 2017)	Octree Voxels	91.0*
3D Shapenets (Wu et al., 2015)	Voxels	83.5
VoxNet (Maturana and Scherer, 2015)	Voxels	92.0
Beam Search (Xu and Todorovic, 2016)	Voxels	88.0
3DGAN (Wu et al., 2016)	Voxels	91.0
binVoxNetPlus (Ma et al., 2018)	Voxels	92.3
LightNet (Zhi et al., 2018)	Voxels	93.9

performance using MRCNN without excessively prolonging the training time. We ran multiple sets of experiments across all four resolutions with different hyperparameters to demonstrate that our proposed framework is agnostic to network architectures. These are described in the online Appendix.

We perform an ablation study to understand the performance of MRCNN compared to traditional CNN networks. Fig. 6a shows the mean test accuracy obtained from the experiments with the variance of the accuracy represented by the shaded regions. There is a clear trend that a denser resolution results in a minor improvement in classification accuracy. Starting from the coarse resolution of 8^3, the *Coarse-level CNN* can achieve a mean test accuracy of 81%. However, with a multi-resolution voxel grid of 32^3 effective resolution, MRCNN can obtain a better mean classification accuracy. Subsequently, a regular CNN applied on a dense voxel grid of 32^3 is able to achieve a slightly better classification accuracy than both. Due to the GPU's memory constraints, we are unable to demonstrate the performance of a *Coarse-level CNN* applied on a dense resolution data of 128^3. Nonetheless, using MRCNN, we are able to achieve the best classification performance using an effective resolution of 128^3 represented by a multi-resolution voxel grid with a base resolution of 32^3 and a fine resolution of 4^3. Another important factor to note is that the variance in the accuracy of the different network architectures is reduced with the increase in resolution. This reduction in variance could be due to the fine features being sufficiently represented in the higher resolution enabling better learning. We see that MRCNN has a variance value that is in between the range for the coarse resolution and the dense resolution CNNs. The lower variance also confirms that our method is robust to changes in the hyperparameters of the network and can perform equivalently better to a single level representation (although not to the same extent as the dense representation), with a much lower memory cost.

A comparison of our results with the performance of other spatial deep learning methods is also tabulated in Table 2. We compare our performance with OctNet due to the similarities in data representation (high-resolution voxel grid) and classification task that exploits spatial data sparsity. In addition to that, we compare MRCNN performance with other voxel-based methods employed on the ModelNet10 dataset. We observe that MRCNN (91.3%) outperforms some of the voxel-based

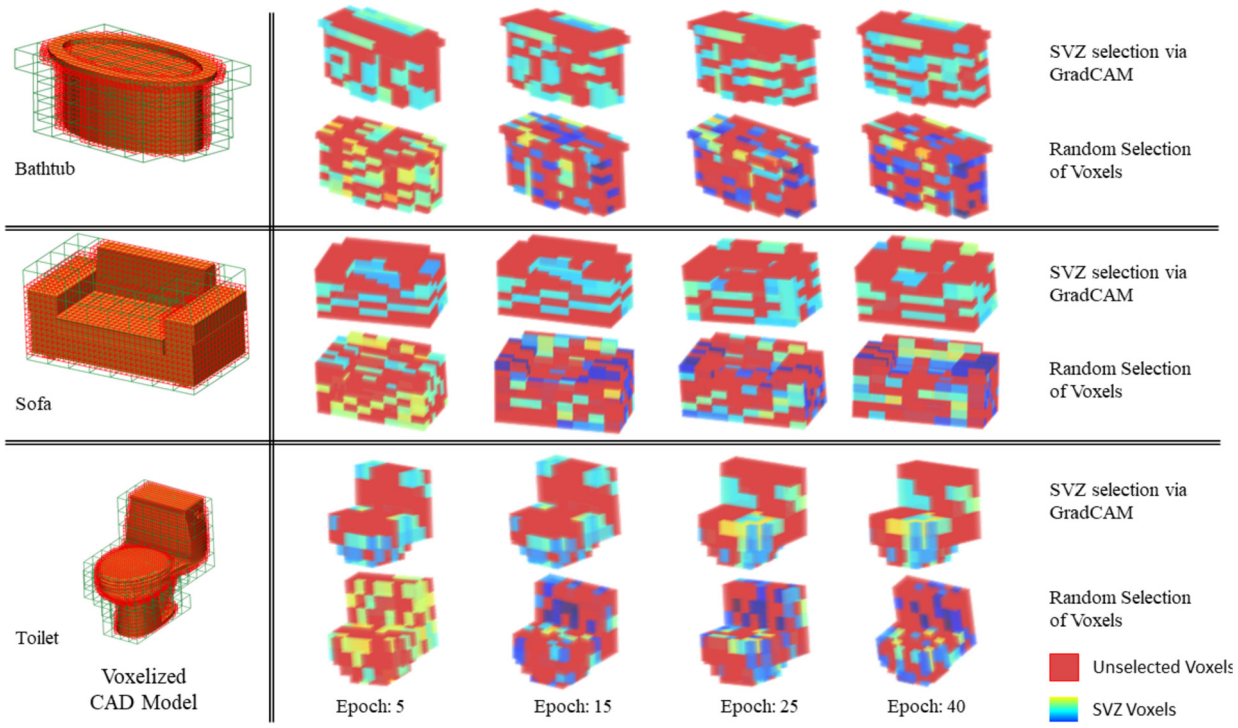

Fig. 7. Visualization of salient-voxel-zones from *Fine-level CNN* to *Coarse-level CNN* via random selection vs. GradCAM on ModelNet10 dataset using MRCNN. We can observe that the intensity of the embedding changes drastically between the 5^{th} and 15^{th} epoch, which demonstrates the learning of a region's importance. Hence, adaptive SVZ selection technique enables a more efficient sampling of important boundary voxels.

methods and is better at classification than OctNet (91.0%). To compare with VoxNet, our dense-level network with selectively chosen hyperparameters has better performance (92.1%) than VoxNet (92.0%) as seen from Fig. 6a, while consuming memory comparable to a coarser voxel grid. Additionally, OctNet defines a layer-based approach (i.e., defining a specific convolution operation to deal with each hierarchy) in deep learning parlance. However, we use a modular approach with two networks (one dealing with lower-level features and the other for coarser features). This modular approach helps in dealing with noise and different artifacts in the geometry robustly, unlike in the case of OctNet.

An interesting observation is that although the mean classification accuracy achieved on the coarse level voxel grid is relatively high, there is a great variance across the set of experiments performed as well, with certain network architectures having an accuracy of only 55%. Generally, we observed that network architectures that use average-pooling layers see a bigger increase in performance when used with MRCNN. Hence, the advantages of MRCNN are two-fold. First, we show that with a multi-level data structure, MRCNN can learn simultaneously from the two levels of data and perform better than a standard CNN trained on coarse resolution data. Second, MRCNN greatly reduces the effect of varying network architectures on the performance of the network, especially on the lower bound of the performance. This is advantageous as it allows greater flexibility when designing a deep network.

An additional advantage of the MRCNN framework is the GPU memory requirement during the training of the network. In Fig. 6b, we show a comparison between the memory requirements of the GPU for training on the four different resolutions of data. As shown in Fig. 6b, the memory requirements of the GPU scales exponentially with the test accuracy, resulting in an inability to train a dense-level network on 128^3 voxel resolution (shown as a blue hatched bar). To achieve a 6% (82.6% to 88.2%) increase in test accuracy by training on dense resolution data, an additional GPU memory of at least 300% is required when compared to training the MRCNN model. This highlights the effect of sparsity in a dense resolution voxel grid where the increase in classification performance scales non-linearly with data resolution.

Further, to study the effect of the different components of the multi-level network on classification accuracy, we perform an ablation study by comparing the performance of MRCNN with and without the SVZ selection technique. While we performed experiments on several network architectures, we repeatedly observed that SVZ performs marginally better than non-SVZ. For example, the performance of several architectures led to average test accuracy of 88.68% without the SVZ selection technique. In contrast, the performance of the same network architectures with the SVZ selection technique is 88.92%. While the improvement in accuracy is not significant, the SVZ approach leads to a more interpretable result by highlighting the regions of importance in the model. Fig. 7 shows a visual comparison of selecting boundary voxels via random sampling and using the SVZ selection technique. From the color intensity of the sampled boundary voxels, we can see that as the training progresses from the 5^{th} to the 15^{th} epoch, a regional pattern emerges from the embeddings

sampled through SVZ. This demonstrates that the network is learning the importance of different regions to classify an object correctly. Therefore, by applying the SVZ technique, the network has better control over the sampling of significant boundary voxels in comparison to random sampling.

MRCNN can also be applied to non-CAD model representations of 3D objects such as meshes and point clouds. A voxel grid can be generated from point clouds by sampling the points in the point cloud on an overlayed tight voxel grid. A multi-level voxel grid can also be constructed from the point cloud by sampling points inside each coarse voxel grid to create fine-level voxels. Object recognition on the point cloud can then be performed by applying MRCNN on this multi-level voxel grid.

6. Conclusions

In this paper, we explore a novel deep learning architecture, MRCNN, to learn from 3D data in a hierarchical manner using multi-resolution voxel-based data structures. We also demonstrated that a model interpretability mechanism could be leveraged to enable information flow between the two levels of MRCNN. Our object recognition results show that MRCNN performance is significantly better and robust than regular CNNs trained on coarse-resolution data while having similar memory requirements. MRCNN performs about as well as CNNs trained on dense data with equivalent resolution while keeping the memory requirements significantly lower. We can extend this inference to feature-rich CAD models of mechanical components given the availability of such a large dataset of models. Future works will include exploring efficacies of MRCNN on such a collection of CAD models and various object recognition datasets and other relevant engineering problems such as design, manufacturing, and analysis, where extraction of multi-scale features is critically important.

CRediT authorship contribution statement

Sambit Ghadai: Methodology, Software, Visualization, Writing – original draft. **Xian Yeow Lee:** Methodology, Software, Visualization, Writing – original draft. **Aditya Balu:** Conceptualization, Methodology, Software, Writing – original draft. **Soumik Sarkar:** Conceptualization, Supervision, Writing – review & editing. **Adarsh Krishnamurthy:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported in part by the National Science Foundation under Grant Number CMMI:1644441. We would also like to acknowledge NVIDIA for the donation of the GPUs used in this work.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cagd.2021.102038.

References

Abdul-Rahman, A., Pilouk, M., 2007. Spatial Data Modelling for 3D GIS. Springer Science & Business Media.

Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M., 2018. Automatic differentiation in machine learning: a survey. J. Mach. Learn. Res. 18.

Blelloch, G.E., 1989. Scans as primitive parallel operations. IEEE Trans. Comput. 38, 1526–1538.

Boscaini, D., Masci, J., Rodoià, E., Bronstein, M., 2016. Learning shape correspondence with anisotropic convolutional neural networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. USA, pp. 3197–3205.

Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P., 2017. Geometric deep learning: going beyond Euclidean data. IEEE Signal Process. Mag. 34, 18–42. https://doi.org/10.1109/MSP.2017.2693418.

Cardone, A., Gupta, S.K., Karnik, M., 2003. A survey of shape similarity assessment algorithms for product design and manufacturing applications. J. Comput. Inf. Sci. Eng. 3, 109–118.

Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J., 2017. PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 77–85.

Choy, C., Gwak, J., Savarese, S., 2019. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3075–3084.

Couprie, C., Farabet, C., Najman, L., LeCun, Y., 2013. Indoor semantic segmentation using depth information. In: International Conference on Learning Representations, pp. 1–8.

Elinson, A., Nau, D.S., Regli, W.C., 1997. Feature-based similarity assessment of solid models. In: Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications. ACM, pp. 297–310.

Fey, M., Lenssen, J.E., Weichert, F., Müller, H., 2018. SplineCNN: fast geometric deep learning with continuous B-spline kernels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 869–877.

Ghadai, S., Balu, A., Sarkar, S., Krishnamurthy, A., 2018. Learning localized features in 3D CAD models for manufacturability analysis of drilled holes. Comput. Aided Geom. Des. 62, 263–275. https://doi.org/10.1016/j.cagd.2018.03.024.

Ghadai, S., Yeow Lee, X., Balu, A., Sarkar, S., Krishnamurthy, A., 2019. Multi-level 3D CNN for learning multi-scale spatial features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 1–5.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.
Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2261–2269.
Ioannidou, A., Chatzilari, E., Nikolopoulos, S., Kompatsiaris, I., 2017. Deep learning advances in computer vision with 3D data: a survey. ACM Comput. Surv. 50, 20.
Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y., Ramani, K., 2005. Three-dimensional shape searching: state-of-the-art review and future trends. Comput. Aided Des. 37, 509–530. https://doi.org/10.1016/j.cad.2004.07.002.
Jayanti, S., Kalyanaraman, Y., Iyer, N., Ramani, K., 2006. Developing an engineering shape benchmark for CAD models. Comput. Aided Des. 38, 939–953.
Joshi, S., Chang, T.C., 1988. Graph-based heuristics for recognition of machined features from a 3D solid model. Comput. Aided Des. 20, 58–66.
Kanezaki, A., Matsushita, Y., Nishida, Y., 2018. RotationNet: joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5010–5019.
Keim, D.A., 1999. Efficient geometry-based similarity search of 3D spatial databases. SIGMOD Rec. 28, 419–430.
Klokov, R., Lempitsky, V., 2017. 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.
Kovnatsky, A., Bronstein, M.M., Bronstein, A.M., Glashoff, K., Kimmel, R., 2013. Coupled quasi-harmonic bases. Comput. Graph. Forum 32, 439–448. https://doi.org/10.1111/cgf.12064.
Kriegel, H.P., Brecheisen, S., Kröger, P., Pfeifle, M., Schubert, M., 2003. Using sets of feature vectors for similarity search on voxelized CAD objects. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. ACM, New York, NY, USA, pp. 587–598.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 84–90. https://doi.org/10.1145/3065386.
Kyprianou, L.K., 1980. Shape classification in computer-aided design. Ph.D. thesis. University of Cambridge.
Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2324. https://doi.org/10.1109/5.726791.
Li, J., Chen, B.M., Lee, G.H., 2018. So-net: self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9397–9406.
Liu, M., Shi, Y., Zheng, L., Xu, K., Huang, H., Manocha, D., 2019. Recurrent 3D attentional networks for end-to-end active object recognition. Comput. Vis. Media 5, 91–104.
Lowe, D.G., 2001. Local feature view clustering for 3D object recognition. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001. IEEE, p. 1.
Ma, C., Guo, Y., Lei, Y., An, W., 2018. Binary volumetric convolutional neural networks for 3D object recognition. IEEE Trans. Instrum. Meas., 1–11. https://doi.org/10.1109/TIM.2018.2840598.
Masci, J., Boscaini, D., Bronstein, M.M., Vandergheynst, P., 2015. Geodesic convolutional neural networks on Riemannian manifolds. In: 2015 IEEE International Conference on Computer Vision Workshop, pp. 832–840.
Maturana, D., Scherer, S., 2015. 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.
Olah, C., Mordvintsev, A., Schubert, L., 2017. Feature visualization. Distill. https://doi.org/10.23915/distill.00007.
Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J., 2016. Volumetric and multi-view CNNs for object classification on 3D data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5648–5656.
Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, pp. 5099–5108.
Ramesh, M., Yip-Hoi, D., Dutta, D., 2001. Feature based shape similarity measurement for retrieval of mechanical parts. J. Comput. Inf. Sci. Eng. 1, 245–256.
Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 1135–1144.
Riegler, G., Ulusoys, A.O., Geiger, A., 2017. Octnet: learning deep 3D representations at high resolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3577–3586.
Roveri, R., Rahmann, L., Oztireli, C., Gross, M., 2018. A network architecture for point cloud classification via automatic depth images generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4176–4184.
Schnabel, R., Wahl, R., Klein, R., 2007. Efficient ransac for point-cloud shape detection. In: Computer Graphics Forum. Wiley Online Library, pp. 214–226.
Seidl, T., Kriegel, H.P., 1998. Optimal multi-step k-nearest neighbor search. In: ACM Sigmod Record. ACM, pp. 154–165.
Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2017. Grad-cam: visual explanations from deep networks via gradient-based localization. In: 2017 IEEE International Conference on Computer Vision, pp. 618–626.
Sfikas, K., Pratikakis, I., Theoharis, T., 2017. Ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval. Comput. Graph.
Socher, R., Huval, B., Bhat, B., Manning, C.D., Ng, A.Y., 2012. Convolutional-recursive deep learning for 3D object classification. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, vol. 1. USA, pp. 656–664.
Song, S., Xiao, J., 2016. Deep sliding shapes for amodal 3D object detection in RGB-D images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 808–816.
Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-view convolutional neural networks for 3D shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 945–953.
Tangelder, J.W., Veltkamp, R.C., 2004. A survey of content based 3D shape retrieval methods. In: Shape Modeling Applications, 2004. Proceedings. IEEE, pp. 145–156.
Tatarchenko, M., Dosovitskiy, A., Brox, T., 2017. Octree generating networks: efficient convolutional architectures for high-resolution 3D outputs. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2088–2096.
Tombari, F., Salti, S., Di Stefano, L., 2011. A combined texture-shape descriptor for enhanced 3D feature matching. In: 18th IEEE International Conference on Image Processing (ICIP), 2011. IEEE, pp. 809–812.
Veltkamp, R.C., 2001. Shape matching: similarity measures and algorithms. In: SMI 2001 International Conference on Shape Modeling and Applications. IEEE, pp. 188–197.
Wang, C., Samari, B., Siddiqi, K., 2018a. Local spectral graph convolution for point set feature learning. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 52–66.
Wang, F., Decker, J., Wu, X., Essertel, G., Rompf, T., 2018b. Backpropagation with callbacks: foundations for efficient and expressive differentiable programming. Adv. Neural Inf. Process. Syst. 31, 10180–10191.

Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X., 2017. O-CNN: octree-based convolutional neural networks for 3D shape analysis. ACM Trans. Graph. 36.
Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B., 2016. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. USA, pp. 82–90.
Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3D ShapeNets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920.
Xiao, Y.P., Lai, Y.K., Zhang, F.L., Li, C., Gao, L., 2020. A survey on deep geometry learning: from a representation perspective. Comput. Vis. Media 6, 113–133.
Xu, X., Todorovic, S., 2016. Beam search for learning a deep convolutional neural network of 3D shapes. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 3506–3511.
Young, G., Krishnamurthy, A., 2018. GPU-accelerated generation and rendering of multi-level voxel representations of solid models. Comput. Graph. 75, 11–24.
Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.), Computer Vision – ECCV 2014. Springer International Publishing, Cham, pp. 818–833.
Zhang, Q., Cao, R., Wu, Y.N., Zhu, S., 2017. Mining object parts from CNNs via active question-answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3890–3899.
Zhang, Q., Wang, W., Zhu, S.C., 2018a. Examining CNN representations with respect to dataset bias. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4464–4473.
Zhang, Q.s., Zhu, S.c., 2018b. Visual interpretability for deep learning: a survey. Front. Inf. Technol. Electron. Eng. 19, 27–39. https://doi.org/10.1631/FITEE.1700808.
Zhi, S., Liu, Y., Li, X., Guo, Y., 2018. Towards real-time 3D object recognition: a lightweight volumetric CNN framework using multitask learning. Comput. Graph. 71, 199–207. https://doi.org/10.1016/j.cag.2017.10.007.