Computer-Aided Design 101 (2018) 12–22

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

FeatureNet: Machining feature recognition based on 3D Convolution Neural Network[☆]

Zhibo Zhang, Prakhar Jaiswal, Rahul Rai [*]

Manufacturing and Design (MAD) Lab, Department of Mechanical and Aerospace Engineering, University at Buffalo (UB)-SUNY, 318 Jarvis Hall, Buffalo, NY-14260, United States

ARTICLE INFO

Article history:
Received 21 August 2017
Accepted 21 March 2018

Keywords:
Machining feature recognition
Convolution neural network
Deep learning
Computer aided process planning (CAPP)

ABSTRACT

Automated machining feature recognition, a sub-discipline of solid modeling, has been an active research area for last three decades and is a critical component in digital manufacturing thread for detecting manufacturing information from computer aided design (CAD) models. In this paper, a novel framework using Deep 3D Convolutional Neural Networks (3D-CNNs) termed FeatureNet to learn machining features from CAD models of mechanical parts is presented. FeatureNet learns the distribution of complex manufacturing feature shapes across a large 3D model dataset and discovers distinguishing features that help in recognition process automatically. To train FeatureNet, a large-scale mechanical part datasets of 3D CAD models with labeled machining features is automatically constructed. The proposed framework can recognize manufacturing features from the low-level geometric data such as voxels with a very high accuracy. The developed framework can also recognize planar intersecting features in the 3D CAD models. Extensive numerical experiments show that FeatureNet enables significant improvements over the state-of-the-arts manufacturing feature detection techniques. The developed data-driven framework can easily be extended to identify a large variety of machining features leading to a sound foundation for real-time computer aided process planning (CAPP) systems.

1. Introduction

We are in the era of digital manufacturing and design. Globally, the methods by which products are designed and manufactured are advancing at a rapid pace just before our very eyes. Manufacturers are increasingly turning to digital manufacturing paradigm to meet the growing demands of increased product quality, greater product variability, shorter product lifecycle, and reduced cost. All things that are manufactured are conceptualized using processes such as sketching and brainstorming, and then converted into computer aided design (CAD, digital) models. CAD models are a great way to validate complex ideas. They capture form, fit, and function of a product and gives you confidence that a given idea is going to work. Once fully developed, CAD models and engineering drawings are handed over to manufacturing department, who take this information to make the physical product using computer aided machining (CAM). In this current paradigm of designing and making things, there is a disconnect between the digital and the physical environments, that is, most of the information created at digital CAD model is lost during the physical fabrication phase (CAM phase).

Computer aided process planning (CAPP) plays a key role in the digital manufacturing pipeline by acting as a nexus between CAD and CAM. CAPP is focused on generating a set of manufacturing operations to fabricate a given part specified by its CAD model. To reason about the fabrication instructions, CAPP has to interpret a given CAD model in terms of *features*. Features are semantically higher level geometric elements such as a hole, a slot, and a pocket and are different from pure geometric elements (such as ribs, pins, etc.) typically used in CAD systems. Features have application-context meaning, and specific set of features needs to be recognized from a CAD model depending on the intended manufacturing process that will be used to fabricate the part. *Machining feature* (interchangeably termed Manufacturing features) have received the most attention in the existing literature. This paper focuses on machining feature recognition that involves recognizing different features in the CAD model of a given part.

There exists significant research literature spanning last three decades in the domain of machining feature recognition (see Section 2 for a detailed discussion). Existing feature recognition techniques are beset with the following problematic issues: 1. Inability to learn and generalize, 2. Lack of tolerance to noise in the input CAD models, 3. Computationally intensive and inflexible, 4. Focused on specific types of CAD representation and thus not generalizable in cases where interoperability between different

[☆] This paper has been recommended for acceptance by William Regli.
[*] Corresponding author.
E-mail address: rahulrai@buffalo.edu (R. Rai).

https://doi.org/10.1016/j.cad.2018.03.006
0010-4485/© 2018 Elsevier Ltd. All rights reserved.

geometric representation is required, and 5. Limited in ability to handle feature variations.

In this paper, a novel application of Deep 3D Convolutional Neural Networks (3D-CNNs) in recognizing machining features from CAD models is outlined. To address the abovementioned issues, instead of relying on existing heuristic and hand-coded features based approach, a data-driven approach that learns *features of features* to identify the machining features from raw 3D data is the core focus of this paper.

The contribution of this paper is threefold: (1) a novel framework using Deep 3D-CNNs to learn machining features from a CAD model of mechanical parts is presented, (2) to train the developed 3D deep learning model, a large-scale machining feature dataset with 3D CAD models, termed FeatureNet database, is automatically constructed, and (3) the strength of our learned deep learning model at recognizing complex machining features in various 3D CAD model is demonstrated.

This paper is organized as follows: Section 2 reflects on some of the recent related works. Section 3 provides an overview of the overall methodology. Section 4 outlines the process for automatic creating of large-scale 3D model dataset. Section 5 provides details involving the architecture of the developed 3D CNNs. The effectiveness of the developed framework is demonstrated on various 3D CAD models in Section 6. Section 7 concludes with a summary of the framework, its effectiveness, and some possible future works.

2. Related work

2.1. Feature recognition

During the past three decades' researchers have proposed different types of feature recognition techniques to recognize machining feature including graph-based approach, neuron network based approach, volume decomposition based approach, cell based approach, hint based approach, rule based approach, and hybrid based approach [1]. All these methods have their pros and cons.

Graph based approach is often regarded as the most successful method. In this method, a B-Rep model of the part is translated into a graph, which has nodes and arcs to represent faces and edges [2]. The concept adjacency attributed graph (AAG), as first formal graph based approach, is usually used to represent B-Rep of part [3]. In AAG, nodes store surface information such as the type of face. And the edges between the adjacency faces can be represented as arcs in AAG. The attributes assigned to arcs include the convexity and concavity. The parts are then decomposed into subgraphs using following heuristic: a face whose incident edges are all convex does not form part of the feature and is deleted from the part graph. Ultimately, the subgraphs are used to match the existing feature graph to provide feature recognition results. This method is computationally expensive as subgraph matching is an NP-complete problem and graph matching is a quasipolynomial problem [4]. Furthermore, AAG is a non-unique structure for representing features. Graph based approach has been quite successful in recognizing isolated features, but it faces difficulty while dealing with overlapping features. A lot of effort has been put to tackle the feature interaction problem. Marefat and Kashyap [5] proposed a concept of a virtual link to restore missing link lost in interaction features. Gao and Shah [6] proposed extended-AAG (EAAG), a revised AAG containing more information, which can help to recognize interacting features. Ibrahim and McCormack [7] presented MAAG that generates multiple interpretations using hints instead of exact pattern matching to handle interacting features. Huang and Yip-Hoi [8] developed a feature relation graph to organize high-level features.

Artificial Neural Networks (ANNs) have been applied in the field of feature recognition since 1990s [9]. Prabhakar and Henderson [10] used ANN to recognize features from solid models. However, their method can just recognize a prime face and several secondary faces. Hwang [11] proposed an approach to use eight-element face score vector as input to ANN system. This method can recognize a large number of different features. Nezis and Vosniakos [12] generate a feedforward ANN to recognize planar and simple curve faces. Lankalapalli et al. [13] used a nine-element face score vector as an input of ART2 neuron network. Onwubolu [14] used the same input but proposed a different network, which is a multi-layer feed-forward back-propagation network. All of these methods can recognize only a limited number of features. Sunil and Pande [15] proposed a 12-node vector scheme to represent features. This approach can recognize a wide range of complex features variations in topology and geometry but fails to deal with interacting feature problem. To some degree, the ANN based approaches still use information extracted from B-Rep model and are based on AAG concepts. Existing feature recognition techniques suffer from the several problematic issues (also discussed in Section 1). Cell based methods are computationally expensive due to combinatorial explosion of number of ways of combining cells into meaningful features. Rule based methods suffer from scalability issues.

2.2. 2D convolution neuron network

In 1998, LeCun proposed LeNet [16] that led to the beginning of modern CNN. His five layers' network, including convolution layers and pooling layers, recognized large image dataset efficiently. Restricted by the performance of CPU and GPU, CNN fell into silence for ten years. After 2012, with the progressing of GPU performance, AlexNet was proposed by Krizhevsky et al. [17]. They used ReLU as activation function and added drop-out layer that greatly increased the accuracy and speed of CNN. In the next few years, deeper networks were proposed, such as VGG [18] and GoogleLeNet [19]. All these networks enhanced the performance of object classification. But they only work on single feature problems. Meanwhile, object detection problem became a prevalent topic with the surge in online database and repositories. Girshick et al. [20] proposed an R-CNN to detect and recognize a feature in images. Following this, fast R-CNN [21] and faster R-CNN [22] improved the structure making object detection problem more efficient and accurate [23]. Most of the 2D CNNs techniques work on images and are not suited to 3D geometric datasets.

2.3. 3D convolution neuron network

With the successful advancement of 2D CNN, some researchers have expanded CNN based framework to solve 3D geometric problems. Wu et al. [24] constructed ModelNet 3D dataset that has more than 150,000 3D models belonging to 660 categories. They proposed a five layer CNN to achieve good classification results. Maturana and Scherer [25] used volumetric occupancy grid to deal with point cloud data from LiDAR and RGBD cameras, and they created a three layers CNN as their classifier. Qi et al. [26] used point cloud data as their input to the CNN and proposed PointNet to perform classification and segmentation. Brock et al. [27] implemented auto-encoder to address challenges unique to voxel-based representation, and their network demonstrated a fifty percent improvement on ModelNet dataset. Hegde and Zadeh combined 3D CNN and 2D CNN [28]. They combined both representations and exploited them to learn new features that proved to be a better classifier than using either of the representations in isolation. Balu et al. [29] presented an application of 3D CNN in the manufacturing area. None of the existing work in the area of 3D CNN has focused on machining feature recognition, and to the best of our

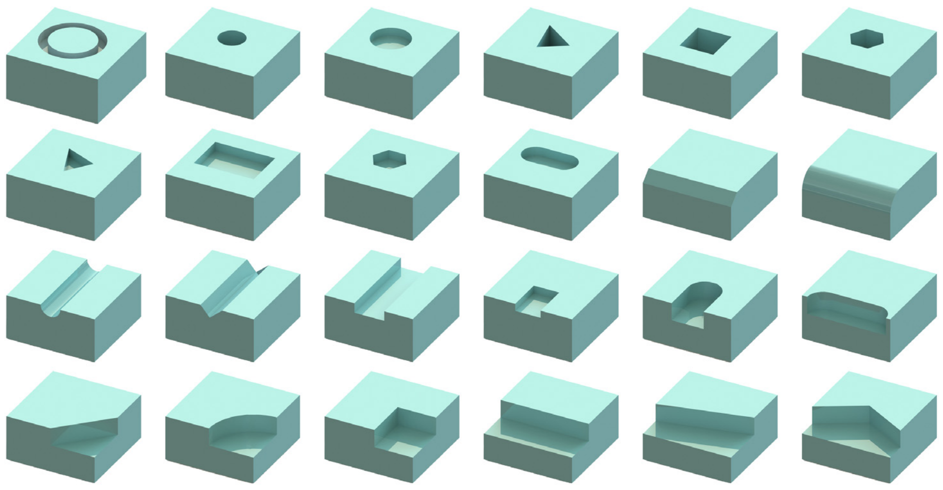


Fig. 1. A set of 24 machining features used in our experimentation (see Appendix for details).

knowledge, the outlined work is the first attempt to apply 3D CNN framework to study machining feature recognition problem.

3. Overview

Machining feature recognition is an important step in transforming CAD design into a machined component. Because of tremendous diversity in the types and shapes of machining features, automatic feature recognition is a hard problem. Several approaches, including rule-based, graph-based, and volume decomposition based, have been proposed in the past (see Section 2). However, most of these approaches fail in recognizing complex features and are limited to recognizing a small set of features only. We propose a deep learning approach that can handle a large set of complex feature recognition. The proposed deep learning approach uses multiple convolution layers to learn a robust and accurate model for recognizing 3D machining features. The performance of any machine learning system is highly dependent on the quality of the dataset used. Therefore, we generate a large dataset of different classes of features in different sizes and orientation. The proposed data-centric deep convolution neural network is trainable, i.e. using a different dataset, it would be possible to re-train the network to recognize a different set of features.

The synthetic machining feature dataset created consists of 24 unique and commonly occurring machining features. For each feature, 6000 different samples were generated. Randomly sampled sizes and all six orientations were used to generate the feature dataset. The variability in the dataset makes the trained model more robust and accurate. More details about the generation of the dataset are provided in Section 4. A Deep 3D Convolutional Neural Network (3D-CNN) was trained to recognize machining features using the generated dataset. The proposed 3D-CNN, termed *FeatureNet*, consists of eight layers — an input layer, four convolution layers, a pooling layer, a fully connected layer, and a final softmax classification layer. The network architecture is described in more detail in Section 5.

TensorFlow library [30] was used to implement and train the CNN network. About 34 million parameters of the network were learned using 144,000 3D models in the machining feature dataset. *The feature recognition accuracy obtained was* 96.7%. Detailed results are presented in Section 6. The proposed network only recognizes non-intersecting features that are the only feature on the CAD model. However, in general, CAD models contain several features that are often overlapping with each other as well. To recognize multiple features on a CAD model, we segment the features using watershed segmentation algorithm. Each segmented feature is then passed through the trained model for recognition. Several examples of multi-feature CAD models and the feature recognition results are also presented in Section 6.

4. Database creation

There is a large number of variations in the machining features that commonly occur in industrial products. Researchers and engineers have attempted in classifying features into groups and types. However, none of the existing classification schemes encompasses all the possible features due to the vastness of the variability in feature geometry and topology. In this paper, we have selected a set of 24 commonly occurring features to develop a feature recognition module. These features are shown in Fig. 1. Our method is independent of the selected set of features. The same approach could be used to train and classify a different set of features, provided that a sufficient number of training samples for each feature is used to learn the parameters of the 3D-CNN.

An accurate and robust machine learning algorithm for recognition and classification requires a large and encompassing training dataset. The dataset is usually acquired from single or multiple sources or is synthetically generated. To train our 3D-CNN network to classify the selected set of features, a synthetic dataset of machining features was generated. For each of the 24 features, 1000 models were generated using CAD modeling software in automatized fashion. A cubic raw stock of 10 cm side length was used to create all the models. The features were generated by randomly selecting feature specific parameter values in a predefined range, and removing the volume from the raw stock. For instance, to create models with blind hole features, four parameters were randomly sampled — R, C_x, C_y, and D, where R is the radius of the blind hole, (C_x, C_y) are the center coordinates, and D is the depth. Random vectors of these four parameter values were sampled from the ranges given in Table 1. Parameters of all the 24 machining features and their ranges are presented in Appendix. A random-valued vector for the parameters of a feature creates one instance of the feature. Hence, to generate 1000 models of the feature, the parameter ranges were uniformly sampled for obtaining 1000 random vectors for each feature.

It should be noted that the raw stock resided completely in the first octant (all positive axes) with one of its corners at the origin and three of its faces aligned with the principal planes — XY, YZ, and ZX. Note that the database examples were restricted to the first octant just for simplicity. The proposed approach is

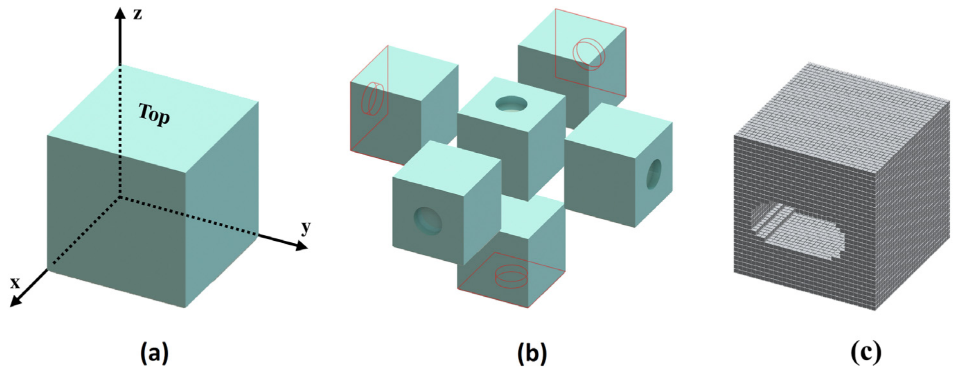


Fig. 2. (a) Cubic raw stock with side length of 10 cm, (b) Six rotations to get features on each of the six faces of the cube, (c) An example of voxelized model.

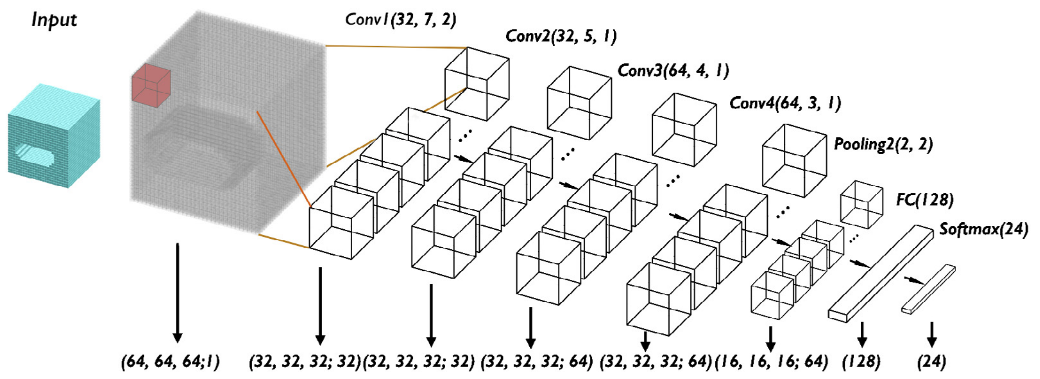


Fig. 3. The proposed architecture of the CNN network trained to recognize machining features on 3D CAD models.

Table 1
Parameters for defining a blind hole feature on a cubic raw stock with side length of 10 cm and the range of values used for each parameter.

Parameters	Ranges (in cm)
Radius (R)	[1.0, 4.5]
Center x-coordinate (C_x)	$[R + 0.5, 9.5 - R]$
Center y-coordinate (C_y)	$[R + 0.5, 9.5 - R]$
Depth (D)	[1.0, 9.0]

invariant to translation in 3D space and can recognize feature at any location. Also, the features and their parameters were defined concerning the *TOP* face of the raw stock that lies on $z = 10$ plane (see Fig. 2(a)). However, to make our method orientation-invariant, each model was rotated to generate six different models with the same feature appearing on each of the six faces of the raw stock cube (see Fig. 2(b)). Therefore, in total, the dataset consisted of 6000 models for each of the 24 selected features resulting in 144,000 models. The overall dataset generated for this study can be located at https://github.com/madlabub/Machining-feature-dataset.

The 3D shape representation used for model creation using CAD modeling software is boundary representation (B-rep). In B-rep, the 3D shape is represented by its boundary surface patches. However, this representation is not convenient for training and using in a CNN network. Therefore, the database models were converted from B-rep to spatial occupancy enumeration, also known as voxels. In this scheme, a 3D shape is represented as a binary valued 3D voxel grid, where 1 indicates that the voxel is inside the shape, and 0 indicates that it is outside. A 3D mesh voxelizer library called *binvox* [31] was used to obtain the voxelized models. *Binvox* incorporates the parity count method and the ray stabbing method for voxelization [32]. The voxel grid size used in our experiments was $64 \times 64 \times 64$. An example of voxelized model is depicted in (Fig. 2(c)).

A complimentary benefit of discretizing the input 3D models is attenuation of noise in the input models. Noise is a critical factor that affects the performance of existing feature recognition algorithms. Due to discretization of input models into voxels, small noise has little to no effect on the performance of our system. However, discretization itself introduces stair-stepping noise and can lead to incorrect classification in some cases. The discretization error can be mitigated by increasing the resolution.

5. FeatureNet: 3D convolution neural network

Convolution neural networks (CNN) have been very successful in image recognition, classification, and object detection. The success of the CNNs can be attributed to the publicly available large image repositories, such as ImageNet, and high performance computing systems, such as clusters and GPUs. Local spatial filters learned by the CNN architecture allow them to exploit the spatial structure of the discrete pixels in the images. They encode complex distinguishing features of images at multiple hierarchies to help with the recognition and classification tasks. In our case of 3D voxels representing CAD machining features, CNNs are a promising technique. They can utilize the spatial structures of the voxels to learn 3D local filters to encode simple spatial structures, like planes, edges, and corners. By stacking multiple convolution layers, more intricate and sophisticated structures can be encoded hierarchically. It would allow the network to recognize complex machining features on CAD models. The proposed CNN network, called *FeatureNet*, was used to recognize single features on a raw stock and consists of eight layers — one input layer, four convolution layers, one max-pooling, one fully connected layer, and one classification output layer (Fig. 3).

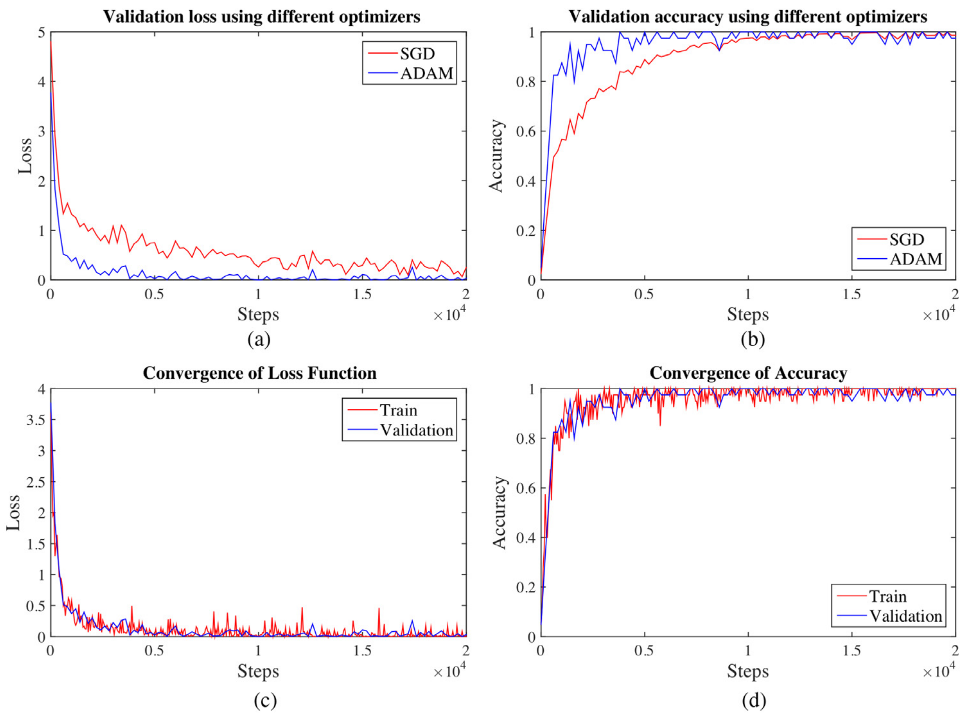


Fig. 4. (a) and (b) show the comparison of two optimizers — SGD and Adam for validation dataset loss and accuracy (SGDLR performed very similar to SGD and hence is omitted from the plot for clarity). (c) and (d) show the convergence of loss and accuracy for both training and validation dataset using Adam optimizer.

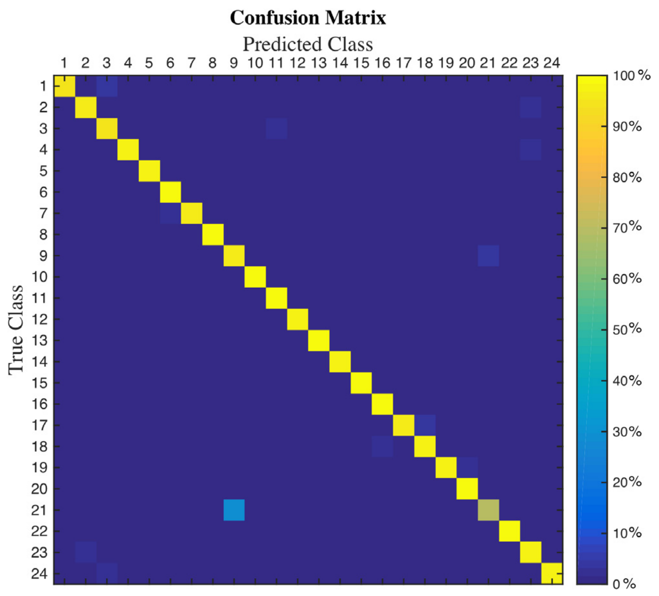


Fig. 5. Confusion matrix of 24 features in the test dataset classification using the learned model.

The input layer admits the 3D voxelized model of fixed size $P \times Q \times R$. In our experiments, $P = Q = R = 64$. The binary-valued voxel models were normalized before they were fed into the input layer. The normalization was performed by subtracting 0.5 from all voxel cells and multiplying by 2. This transforms the voxel values from $\{0, 1\}$ to $\{-1, 1\}$, respectively. This normalization process is termed zero-centering. Zero-centering has been shown to improve performance and result in faster convergence by eliminating bias in the data [33].

Following the input layer, there are four convolution layers. The parameters of the convolution layers are shape of each filter ($d \times d \times d$), number of feature maps (input f' and output f), and spatial strides (s). These layers accept four-dimensional input — three spatial dimensions and one feature map dimension. f feature maps are created by convolving the input with the filters applied at the spatial stride of s. The convolution results are then processed through an activation function called rectified linear units (ReLU) given by the following expression:

$$\mathrm{ReLU}(x) = \max(x, 0). \tag{1}$$

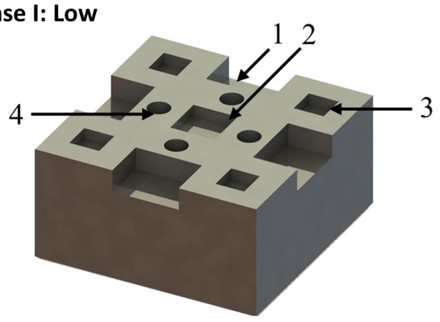
The f, d, and s parameter values of the four convolution layers used in our approach were set to (32, 7, 2), (32, 5, 1), (64, 4, 1), and (64, 3, 1). The f' parameter value for each layer is the number of output feature maps in the previous layer (input layer passes on only one feature map). The fourth convolution layer is followed by a pooling layer. Pooling layer down-samples the feature maps along the spatial dimension. The pooling layer used was max-pooling, i.e. they replace blocks of input with their maximum. The parameters of the pooling layers are the shape of the block ($q \times q \times q$) and the spatial stride (s). A value of 2 was used for both q and s in our approach. Therefore, each non-overlapping block of size $2 \times 2 \times 2$ was replaced by the maximum value in the corresponding block.

The penultimate layer, before the classification output layer, is a fully connected layer with m output neurons. A value of 128 was used for m in our approach. Each output neuron was obtained by applying ReLU on linear combination of all inputs and a bias term. The kth neuron output $u^{(k)}$ is given by the following expression:

$$u^{(k)} = \mathrm{ReLU}(\mathbf{w}^{(k)} \cdot \mathbf{x} + b^{(k)}) \tag{2}$$

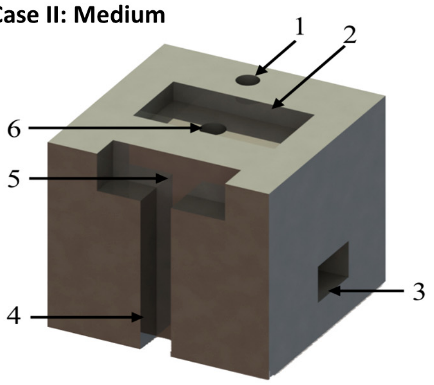
where, $\mathbf{x}$ is the vector of all outputs from the previous pooling layer, $\mathbf{w}^{(k)}$ is the weight vector for kth neuron, and $b^{(k)}$ is its bias term. Note that the ReLU$(\cdot)$ function was applied element-wise.

Case I: Low



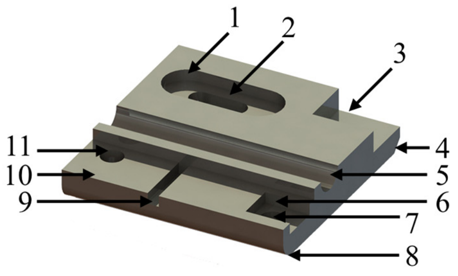
1	Rectangular blind slot
2	Rectangular pocket
3	Rectangular pocket
4	Blind hole

Case II: Medium



1	Through hole
2	Rectangular pocket
3	Rectangular passage
4	Rectangular through slot
5	Rectangular blind slot
6	Blind hole

Case III: High



1	Circular end pocket
2	Circular end pocket
3	Rectangular blind slot
4	Chamfer
5	Circular through slot
6	Rectangular blind slot
7	Through hole
8	Chamfer (Round)
9	Rectangular blind slot
10	Rectangular through step
11	Through hole

Fig. 6. Classification results for multi-feature 3D CAD models with varying complexity levels. All features on cases I and II, with low- and medium-level complexity, were recognized correctly. 10 out of 11 features were recognized correctly for high-level complexity in case III. The feature #8 (highlighted in yellow) was classified as "chamfer", instead of the correct "round" feature. This was due to the low voxel resolution of the input model.

The classification output layer consists of $n = 24$ outputs, one for each of the 24 classes. It was computed by applying softmax activation function on linear combinations of outputs from previous layers. The kth output $y^{(k)}$ is given by the following expression:

$$y^{(k)} = \text{softmax}(\mathbf{v})^{(k)} = \frac{\exp(v^{(k)})}{\sum_j \exp(v^{(j)})} \tag{3}$$

where, $v^{(k)}$ is the linear combination of all the penultimate layer output $(\mathbf{u})$ and a bias term. It is given by the expression $\mathbf{w}_o^{(k)} \cdot \mathbf{u} + b_o^{(k)}$. The softmax activation function normalizes and converts the linear combinations of the inputs $(\mathbf{u})$ into predicted probabilities for each class.

FeatureNet consists of about 34 million parameters. To learn these parameters, we experimented with three different optimizers — stochastic gradient descent, stochastic gradient descent with learning rate decay, and Adam optimizer [34]. The objective function minimized by the optimizers was cross-entropy between the predicted probabilities and the true labels. The cross-entropy cost function for a sample input is defined as:

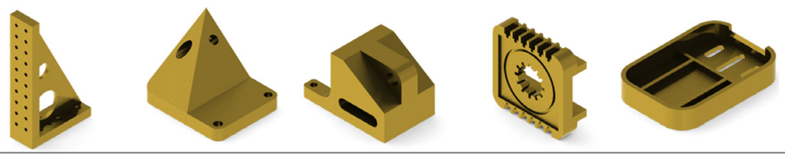
$$H(\mathbf{y}, \bar{\mathbf{y}}) = -\frac{1}{n} \sum_{k=1}^{n} \left(\bar{y}^{(k)} \log y^{(k)} + (1 - \bar{y}^{(k)}) \log(1 - y^{(k)}) \right) \tag{4}$$

where, $\mathbf{y}$ is the vector of softmax predicted probabilities from the final classification layer and $\bar{\mathbf{y}}$ is the one-hot encoded true class label vector.

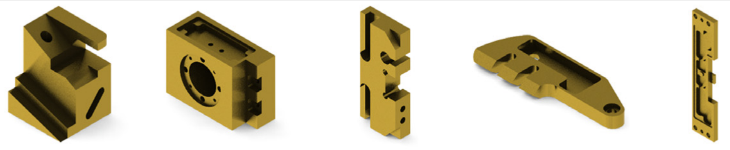
6. Results and discussion

6.1. Single feature recognition

The dataset of 144,000 models, each with a single feature, was split into three subsets — training, validation, and testing datasets. The split percentages were 70%, 15%, and 15%, respectively. TensorFlow library [30] was used for implementation of



True # features:	30	10	13	46	15
# segments identified:	30	12	9	46	15
# correctly classified:	30	10	9	46	15



True # features:	7	21	13	16	19
# segments identified:	8	19	13	12	19
# correctly classified:	7	18	13	12	19

Fig. 7. Feature recognition on ten complex and real industrial components using our approach. It can be noticed that our approach involving segmentation of overlapping feature followed to single feature classification using CNN is quite robust. The overall percentage of correctly classified features in these ten complex examples is **94.21%**.

Table 2
Effect of voxel resolution on computation time and classification accuracy.

Voxel resolution	Training time (min)	Classification time (ms)	Classification accuracy (%)
16 × 16 × 16	**7.5**	**1.27**	83.2
32 × 32 × 32	54	5.05	93.7
64 × 64 × 64	390	33.04	**97.4**

FeatureNet. It was trained using the training datasets and validated using validation dataset. The batch size used during training was set to 40. About 34 million parameters of the proposed network were learned. The experiment was independently performed for three different optimizers — stochastic gradient descent (SGD), stochastic gradient descent with learning rate decay (SGDLR), and Adam optimizer. The initial learning rate was set to 0.001 for all three optimizers. Results from SGD and Adam are compared in Figs. 4(a) and 4(b) for validation loss and validation accuracy, respectively. The SGDLR produced results very similar to SGD. Hence, it is not depicted in the figure for clarity. It can be observed that the Adam optimizer converged much more quickly than the SGD optimizer. Hence, we selected Adam optimizer for all further tests and experiments. The learned model produces a training and validation accuracy of 99.95% and 97.46%, respectively, in 20,000 steps. A graph showing the convergence of loss function and accuracy with steps is shown in Figs. 4(c) and 4(d), respectively. The graph presents results for both training and validation datasets. The test dataset accuracy obtained using the learned model was **96.70%**. A confusion matrix of the classification results among the 24 machining features is shown in Fig. 5. Notice that the confusion matrix is very close to diagonal matrix illustrating the effectiveness and accuracy of our learned model. The largest confusion occurs in feature #9 (rectangular blind slot) and feature #21 (vertical circular end blind slot), which are very similar in shape. The confusion can be mitigated by using a higher resolution voxelization.

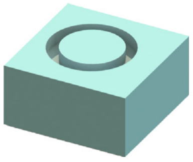
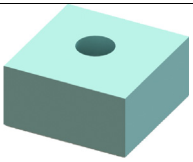
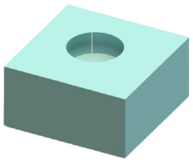
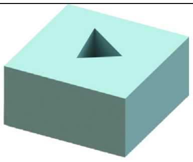
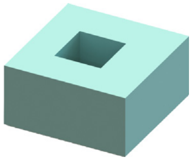
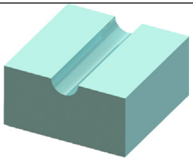
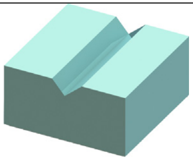
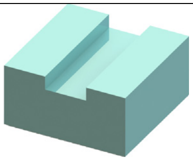
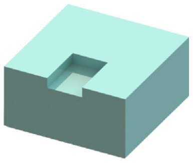
6.2. *Effects of voxel resolution*

To use the FeatureNet, the input 3D models are discretized into binary voxel models. The resolution of voxels is a key parameter that affects both computational efficiency and the classification accuracy of our system. Higher resolution voxels provide better accuracy compared to lower resolution voxels. However, they perform exponentially worse with respect to computation time. To demonstrate the extent of effect of voxel resolution on FeatureNet performance, experiments were conducted at three different resolutions — 16 × 16 × 16, 32 × 32 × 32, and 64 × 64 × 64. For this study, we selected a subset of our database with 7 machining features and 2,400 examples for each feature. Each example was voxelized at three resolutions to generate three separate datasets. The dataset for each resolution was divided into 75% training and 25% testing data. The network was trained at the three resolutions independently using Adam optimizer and tested using the corresponding test data. The time and accuracy results are shown in Table 2. The experiment was conducted on a PC with quad-core Intel i7 processor (2.60 GHz), Windows 10 operating system, 16GB RAM, and NVIDIA GeForce GTX 960M graphics card. The training time is the total amount of time taken by each network to converge and learn the optimal parameters for the network using the training dataset. The classification time is the average time taken by the trained network in classifying a single example model. It was computed by averaging over classification of 4200 example models. The classification accuracy is the percentage accuracy of the trained network over corresponding test dataset. As expected, the computation performance worsens exponentially with increasing voxel resolution, whereas the classification accuracy improves.

6.3. *Multi-feature recognition*

A typical CAD component usually has multiple machining features. Often, some of those features are overlapping. But the FeatureNet was trained to identify component with single feature only. Therefore, to enhance the utility of the proposed network and to recognize multiple overlapping features, it was augmented with a segmentation technique. First, the input CAD component is voxelized using the *binvox* library as discussed before. The disconnected features or disconnected subset of features are separated from each other in the voxelized model by using connected component labeling algorithm provided in the *scikit-image* package [35] for *Python*. Next, each disconnected subset of features is segmented into single individual features using watershed segmentation algorithm [36]. Finally, the single segmented features are passed as input into the trained CNN model for classification.

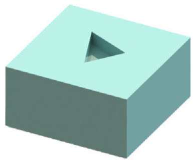
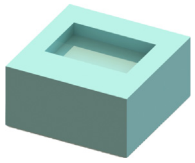
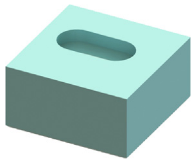
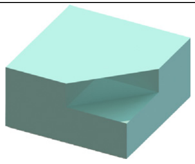
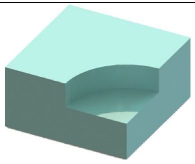
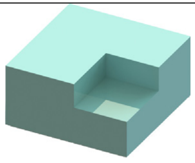
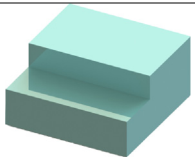
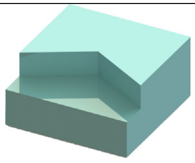
Table A.1
A list of 24 machining features used in our study. The parameters and parameter ranges used for generating samples of these features are also provided.

Class	Figure	Name	Parameters	Range (cm)
1		O-ring	R r C_x C_y D	$[1, 4.5]$ $[R/3, R - 0.2]$ $[R + 0.5, 9.5 - R]$ $[R + 0.5, 9.5 - R]$ $[1, 9]$
2		Through hole	R C_x C_y	$[1, 4.5]$ $[R + 0.5, 9.5 - R]$ $[R + 0.5, 9.5 - R]$
3		Blind hole	R C_x C_y D	$[1, 4.5]$ $[R + 0.5, 9.5 - R]$ $[R + 0.5, 9.5 - R]$ $[1, 9]$
4		Triangular passage	L X Y	$[1, 9]$ $[0.5, 9.5 - L]$ $[0.5, 9.5 - L]$
5		Rectangular passage	L W X Y	$[1, 9]$ $[1, 9]$ $[0.5, 9.5 - L]$ $[0.5, 9.5 - W]$
6		Circular through slot	R X	$[1, 4.5]$ $[R + 0.5, 9.5 - R]$
7		Triangular through slot	W X D	$[1, 9]$ $[0.5, 9.5 - W]$ $[1, 9]$
8		Rectangular through slot	W X D	$[1, 9]$ $[0.5, 9.5 - W]$ $[1, 9]$
9		Rectangular blind slot	W H D X	$[1, 9]$ $[1, 9]$ $[1, 9]$ $[0.5, 9.5 - W]$

(*continued on next page*)

As an initial experiment, three examples of CAD components with multiple features of varying complexity were tested, and the results are demonstrated in Fig. 6. It can be noticed that our approach performed well and correctly recognized all machining features on low- and medium-level complexity parts. For high-level complexity, 10 of the 11 machining features were recognized

Table A.1 (*continued*)

Class	Figure	Name	Parameters	Range (cm)
10		Triangular pocket	L X Y D	$[1, 9]$ $[0.5, 9.5 - L]$ $[0.5, 9.5 - L]$ $[1, 9]$
11		Rectangular pocket	L W X Y D	$[1, 9]$ $[1, 9]$ $[0.5, 9.5 - L]$ $[0.5, 9.5 - W]$ $[1, 9]$
12		Circular end pocket	W L X Y D	$[1, 8]$ $[1, 9 - W]$ $[W/2 + 0.5, 9.5 - W/2 - L]$ $[1, 9 - W]$ $[1, 9]$
13		Triangular blind step	X Y D	$[1, 9]$ $[1, 9]$ $[1, 9]$
14		Circular blind step	R D	$[1, 9]$ $[1, 9]$
15		Rectangular blind step	X Y D	$[1, 9]$ $[1, 9]$ $[1, 9]$
16		Rectangular through step	W H	$[1, 9]$ $[1, 9]$
17		2-sides through step	A B D	$[1, 8]$ $[A + 0.5, 9]$ $[1, 9]$

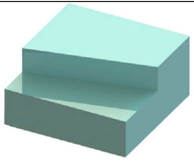
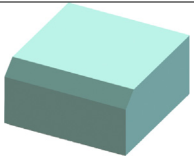
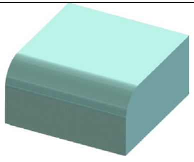
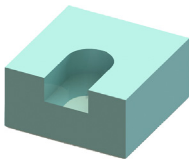
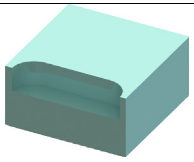
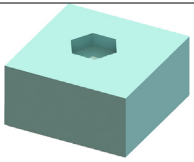
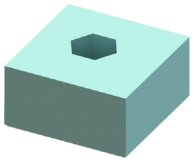
(*continued on next page*)

correctly. Furthermore, the incorrect recognition of a round feature as a chamfer feature is due to the low resolution of voxelization used. We also tested our system on ten highly complex and real industrial components. The results are presented in Fig. 7. 179 out of 190 features (94.21%) were correctly segmented and classified. In future, we plan to test and develop more sophisticated segmentation algorithms and train the network at higher resolution to further improve the accuracy and robustness of our approach.

7. Conclusion

To recognize machining features in 3D CAD models, this paper develops a 3D CNN based framework that operates on a 3D voxel grid. To train this 3D deep learning model, termed FeatureNet, a large-scale 3D CAD model dataset with labeled machining features is automatically constructed. Our 3D CNN model significantly outperforms existing approaches on a variety of machining feature recognition tasks, and it is also a promising approach for recognizing planar intersecting machining features. To the best of

Table A.1 (*continued*)

Class	Figure	Name	Parameters	Range (cm)
18		Slanted through step	A B D	$[1, 9]$ $[1, 9]$ $[1, 9]$
19		Chamfer	X Y	$[1, 9]$ $[1, 9]$
20		Round	R	$[1, 9]$
21		Vertical circular end blind slot	W H D X	$[1, 9]$ $[1, 9.5 - W/2]$ $[1, 9]$ $[0.5, 9.5 - W]$
22		Horizontal circular end blind slot	W L D	$[1, 4]$ $[1, 9 - 2W]$ $[1, 9]$
23		6-sides passage	R C_x C_y	$[1, 4.5]$ $[R + 0.5, 9.5 - R]$ $[R + 0.5, 9.5 - R]$
24		6-sides pocket	R C_x C_y D	$[1, 4.5]$ $[R + 0.5, 9.5 - R]$ $[R + 0.5, 9.5 - R]$ $[1, 9]$

our knowledge, this is the first application of deep learning for machining feature recognition application.

The same framework can be easily extended to learn a large variety of machining features from a variety of different manufacturing processes. In addition, the same framework can be used to identify non-manufacturable features in additively manufactured parts. Multiple trained 3D CNNs could be simultaneously combined to reason about manufacturability of a given 3D CAD model using different manufacturing processes. A promising direction of future work is to integrate the tasks of segmentation and classification. By restructuring the deep CNN architecture and training on large multi-feature dataset, both of these tasks can be performed simultaneously by the CNN.

Appendix. Machining feature set and their parameters

See Table A.1.

References

[1] Han J, Pratt M, Regli WC. Manufacturing feature recognition from solid models: a status report. IEEE Trans Robot Autom 2000;16(6):782–96.

[2] Verma AK, Rajotia S. A review of machining feature recognition methodologies. Int J Comput Integr Manuf 2010;23(4):353–68.

[3] Joshi S, Chang T-C. Graph-based heuristics for recognition of machined features from a 3D solid model. Comput Aided Des 1988;20(2):58–66.

[4] Babai L. Graph isomorphism in quasipolynomial time. In: Proceedings of the forty-eighth annual ACM symposium on theory of computing. ACM; 2016. p. 684–97.

[5] Marefat M, Kashyap RL. Automatic construction of process plans from solid model representations. IEEE Trans Syst Man Cybern 1992;22(5):1097–115.

[6] Gao S, Shah JJ. Automatic recognition of interacting machining features based on minimal condition subgraph. Comput Aided Des 1998;30(9):727–39.

[7] Ibrhim R, McCormack A. Process planning using adjacency-based feature extraction. Int J Adv Manuf Technol. 2002;20(11):817–23.

[8] Huang Z, Yip-Hoi D. High-level feature recognition using feature relationship graphs. Comput Aided Des 2002;34(8):561–82.

[9] Henderson M, Srinath G, Stage R, Walker K, Regli W. Boundary representation-based feature identification. Manuf Res Technol 1994;20: 15–15.

[10] Prabhakar S, Henderson MR. Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. Comput Aided Des 1992;24(7):381–93.

[11] Hwang J-L. Applying the perceptron to three-dimensional feature recognition [Ph.D. thesis], Tempe, AZ, USA: Arizona State University; 1992 UMI Order No. GAX92-10388.

[12] Nezis K, Vosniakos G. Recognizing 212d shape features using a neural network and heuristics. Comput Aided Des 1997;29(7):523–39.
[13] Lankalapalli K, Chatterjee S, Chang T. Feature recognition using ART2: a self-organizing neural network. J Intell Manuf 1997;8(3):203–14.
[14] Onwubolu GC. Manufacturing features recognition using backpropagation neural networks. J Intell Manuf 1999;10(3–4):289–99.
[15] Sunil V, Pande S. Automatic recognition of machining features using artificial neural networks. Int J Adv Manuf Technol 2009;41(9):932–47.
[16] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE 1998;86(11):2278–324.
[17] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012. p. 1097–105.
[18] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition; 2014. ArXiv preprint arXiv:1409.1556.
[19] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2015. p. 1–9.
[20] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2014. p. 580–7.
[21] Girshick R. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision; 2015; p. 1440–8.
[22] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. 2015. p. 91–9.
[23] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44.
[24] Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J. 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2015. p. 1912–1920.
[25] Maturana D, Scherer S. Voxnet: A 3d convolutional neural network for real-time object recognition. In: Intelligent robots and systems, 2015 IEEE/RSJ international conference on. IEEE; 2015. p. 922–8.
[26] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation; 2016. ArXiv preprint arXiv:1612.00593.
[27] Brock A, Lim T, Ritchie JM, Weston N. Generative and discriminative voxel modeling with convolutional neural networks; 2016. ArXiv preprint arXiv:1608.04236.
[28] Hegde V, Zadeh R. Fusionnet: 3d object classification using multiple data representations; 2016. ArXiv preprint arXiv:1607.05695.
[29] Balu A, Lore KG, Young G, Krishnamurthy A, Sarkar S. A deep 3d convolutional neural network based design for manufacturability framework; 2016. ArXiv preprint arXiv:1612.02141.
[30] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. Tensorflow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org; 2015. http://tensorflow.org/.
[31] Min P. Binvox: 3d mesh voxelizer. Retrieved July 24, 2017. http://www.patrickmin.com/binvox/.
[32] Nooruddin FS, Turk G. Simplification and repair of polygonal models using volumetric techniques. IEEE Trans Vis Comput Graphics 2003;9(2):191–205.
[33] Schraudolph NN. Centering neural network gradient factors. In: Neural networks: Tricks of the trade. Springer; 1998. p. 207–26.
[34] Kingma D, Ba J. Adam: a method for stochastic optimization; 2014. ArXiv preprint arXiv:1412.6980.
[35] Van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, Gouillart E, Yu T. scikit-image: image processing in Python. PeerJ 2014;2:e453.
[36] Neubert P, Protzel P. Compact watershed and preemptive SLIC: On improving trade-offs of superpixel segmentation algorithms. In: Pattern recognition, 2014 22nd international conference on. IEEE; 2014. p. 996–1001.