Computer Aided Geometric Design 62 (2018) 263–275

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Learning localized features in 3D CAD models for manufacturability analysis of drilled holes

Sambit Ghadai [1], Aditya Balu [1], Soumik Sarkar, Adarsh Krishnamurthy *

Department of Mechanical Engineering, Iowa State University, Ames, IA, USA

ARTICLE INFO

Article history:
Available online 23 March 2018

Keywords:
Localized Feature Detection
Design for Manufacturing
Machine Learning
3D Convolutional Neural Network
Voxelized Representations
GPU-Accelerated Algorithms

ABSTRACT

We present a novel feature identification framework to recognize *difficult-to-manufacture* drilled holes in a complex CAD geometry using deep learning. Deep learning algorithms have been successfully used in object recognition, video analytics, image segmentation, etc. Specifically, 3D Convolutional Neural Networks (3D-CNNs) have been used for object recognition from 3D voxel data based on the external shape of an object. On the other hand, manufacturability of a component depends on local features more than the external shape. Learning these local features from a boundary representation (B-Rep) CAD model is challenging due to lack of volumetric information. In this paper, we learn local features from a voxelized representation of a CAD model and classify its manufacturability. Further, to enable effective learning of localized features, we augment the voxel data with surface normals of the object boundary. We train a 3D-CNN with this augmented data to identify local features and classify the manufacturability. However, this classification does not provide information about the source of non-manufacturability in a complex component. Therefore, we have developed a 3D-CNN based gradient-weighted class activation mapping (3D-GradCAM) method that can provide visual explanations of the local geometric features of interest within an object. Using 3D-GradCAM, our framework can identify *difficult-to-manufacture* features, which allows a designer to modify the component based on its manufacturability and thus improve the design process. We extend this framework to identify *difficult-to-manufacture* features in a realistic CAD model with multiple drilled holes, which can ultimately enable development of a real-time manufacturability decision support system.

1. Introduction

Deep learning (DL) algorithms, designed to hierarchically learn multiple levels of abstractions from data, have been extensively used in computer vision (Sarkar et al., 2015; Lee et al., 2009; Lore et al., 2017). Specifically, 3D-Convolutional Neural Networks (3D-CNN) have been used extensively for object recognition, point cloud labeling, video analytics, human gesture recognition, object shape retrieval etc. (Wu et al., 2015; Maturana and Scherer, 2015a,b; Riegler et al., 2016; Huang and You, 2016; Zhu et al., 2016). Object recognition (Socher et al., 2012) has been one of the most challenging problems in the area of computer vision and pattern recognition. Early DL-based approaches used simple projection of the 3-dimensional object to a 2-dimensional representation such as depth images or multiple views to recognize an object. However, there is

* Corresponding author.
E-mail addresses: sambitg@iastate.edu (S. Ghadai), adarsh@iastate.edu (A. Krishnamurthy).
[1] S. Ghadai and A. Balu contributed equally to the paper.

https://doi.org/10.1016/j.cagd.2018.03.024
0167-8396/

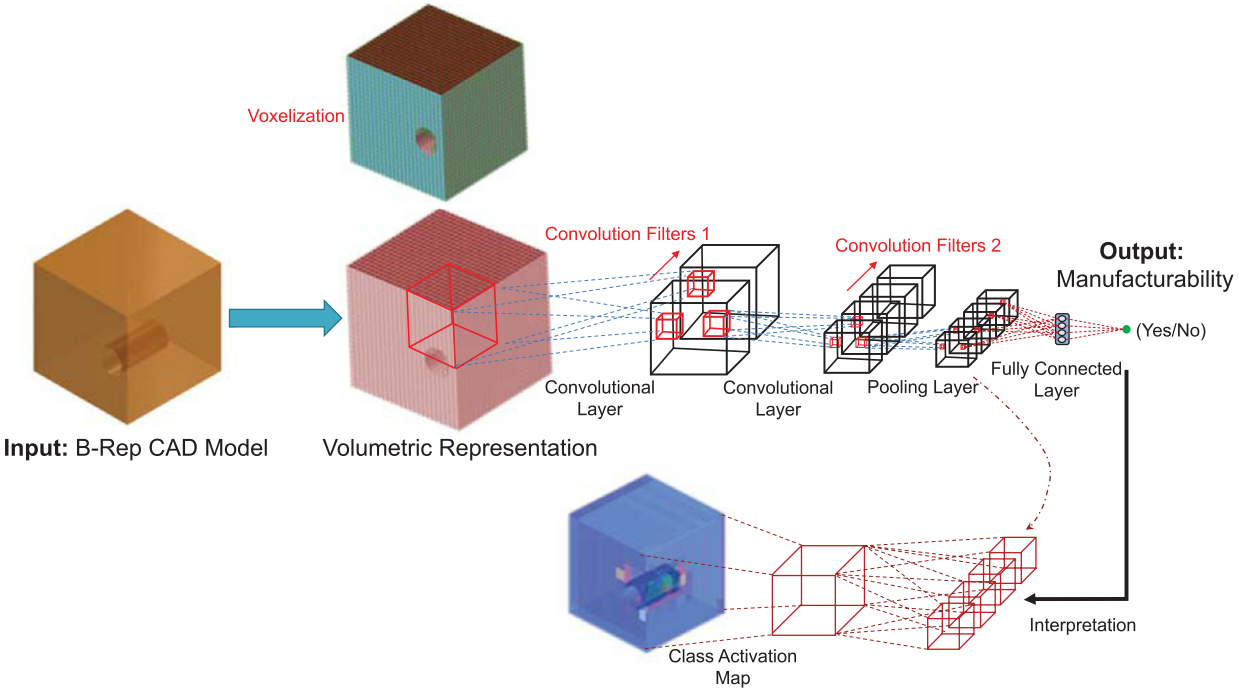


Fig. 1. Framework for deep-learning based design for manufacturability. The CAD model is converted to a voxel representation and is input to a 3D-CNN for manufacturability classification. The 3D-CNN output is analyzed using 3D-GradCAM to provide manufacturability feedback.

significant loss of geometric information while using a 2D representation of a 3D object. Therefore, it is difficult to learn about the local or internal features of the object using these projection-based techniques. In this paper, we make use of voxelized 3D representation of the object, augmented with surface normal information, to identify localized features in an object.

Voxelized models have been successfully used for object recognition (Wu et al., 2015; Maturana and Scherer, 2015a,b). Generally, point cloud data is converted to a volume occupancy grid or voxels to represent the model and identify the object. However, voxel-based occupancy grid representation does not inherently have information regarding the surfaces of the object without additional processing. It is also not robust enough to capture information about the location, size, or shape of a feature within an object. Providing additional information about the geometry increases the accuracy of object detection (Wang and Siddiqi, 2016). In this paper, we propose to use normal information of the surfaces of the object, in addition to the volume occupancy information, to efficiently learn the local features.

Learning local features in a geometry is different from object recognition. Object recognition is a classification problem, where the object is classified based on a collection of features identified in the object. In this work, we make use of a DL network to learn local feature descriptors without any specific input to describe the features. Based on the feature descriptors that are learned, we make use of a semi-supervised methodology to classify the part based on the local features of interest.

One of the applications of the aforementioned methodology explored in this paper, is to identify *difficult-to-manufacture* features in a CAD geometry and ultimately classify its manufacturability. A successful part or product needs to meet its specifications, while being feasible to manufacture. In general, manufacturability feedback is provided to the designer only after the design has been finalized. This leads to an iterative process, often leading to longer product development times and higher costs. There are different handcrafted design for manufacturability (DFM) rules that ensure manufacturability of a design. For this purpose, the hierarchical architecture of DL can be used to learn increasingly complex features by capturing *localized geometric features* and *feature-of-features*. Thus, a deep-learning-based design for manufacturing (DLDFM) tool (Fig. 1) can be used to learn the various DFM rules from different examples of manufacturable and non-manufacturable components without explicit handcrafting.

A primary concern while examining the manufacturability of CAD geometries using a DL based approach is the black-box nature of such deep networks. Interpretation of the decision making process in the form of visual explanations is essential for extracting the local features in an object that effectuates its non-manufacturability. Visual detection of local features further enables re-designing of the component to be manufacturable. Recent major work on interpreting DL output by Selvaraju et al. (2016) makes use of a 2D gradient-weighted class activation map for producing visual explanations of the CNN's decision making process for object recognition in images. In this work, we extend the GradCAM to 3D objects for interpretation of 3D-CNN's outputs and visualizing the regions that give rise to non-manufacturability conditions in the objects.

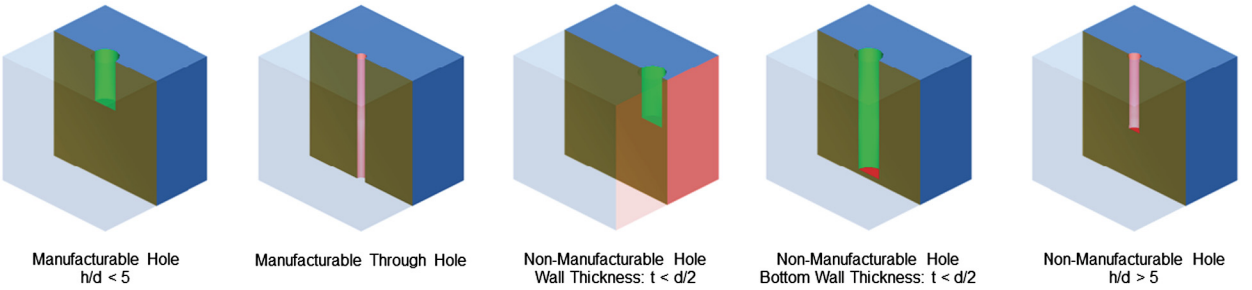


Fig. 2. Different DFM rules based hole examples to classify manufacturable and non-manufacturable geometries.

In this paper, we present a 3D Convolutional Neural Network (3D-CNN) based framework that will learn and identify localized geometric features from an expert database in a semi-supervised manner. Further, this framework is trained in the context of manufacturability with different CAD models classified as *manufacturable* and *non-manufacturable* based on traditional DFM rules. The main contributions of this paper include:

- GPU-accelerated methods for converting CAD models to volume representations (voxelization augmented with surface normals), which can be used to learn localized geometric features.
- Application of deep learning to manufacturability analysis of drilled holes.
- A novel voxel-wise 3D gradient-weighted feature localization based on the 3D-CNN framework to identify local features that are non-manufacturable.

This paper is arranged as follows. In Section 2, we discuss the design for manufacturability (DFM) rules used for generation of the datasets for training and testing the 3D-CNN. We explain about the volume representations that we use for 3D-CNN in Section 3. In Section 4, we discuss the details of 3D-CNN, including the network architecture and the hyper-parameters. In Section 5, we explain 3D gradient weighted class activation mapping for identifying the localized geometric features. Finally, in Section 6, we show the results of the deep learning based design for manufacturability (DLDFM) framework in classifying *manufacturable* and *non-manufacturable* CAD models and capability of the model to identify localized geometric features in a large realistic CAD model.

2. Manufacturability of drilled holes

Design for Manufacturing (DFM) rules for drilling have been traditionally developed based on the parameters of cylindrical geometry and geometry of the raw material. In this paper, we show an application of our approach to learn localized geometric features to identify non-manufacturability of drilled holes. Deciding the manufacturability of a part is framed as a binary classification problem. We apply 3D-CNN to learn the parameters to classify for manufacturability.

The important geometric parameters of a hole are the diameter, the depth, and position of the hole. However, there are certain additional geometric parameters that do not contribute to manufacturability analysis but might affect the machine learning framework. For example, the face of the stock on which the hole is to be drilled does not affect the manufacturability of the hole but need to be considered while training because the volumetric representation of the CAD model is not rotationally invariant.

In our DLDFM framework, the following DFM rules are used to classify the drilled hole as manufacturable as shown in Fig. 2:

1. **Depth-to-diameter ratio**: The depth-to-diameter ratio should be less than 5.0 for the machinability of the hole (Boothroyd et al., 2002; Bralla, 1999). It should be noted that this rule is generic and applicable for all materials.
2. **Through holes**: Since a through hole can be drilled from both directions, the depth-to-diameter ratio for a through hole should be less than 10.0 to be manufacturable.
3. **Holes close to the edges**: A manufacturable hole should be surrounded with material of thickness at least equal to the half the diameter of the hole.
4. **Thin sections in the depth direction of the hole**: A manufacturable hole should have material greater than half the diameter along the depth direction.

The preceding rules are used to generate the ground truth manufacturability data for the training set, which is then used to learn the manufacturable and non-manufacturable features by the DLDFM Framework. However, it should be noted that for a DLDFM framework, one need not explicitly mention the rule. Rather, for industrial applications, one can train the DLDFM framework using the industry relevant historical data available in the organization, which need not be strictly rule based. The historical data can also be based on experience during previous attempts to manufacture a part. Thus, this DLDFM framework is an attempt to generate a DFM framework based on few basic local features. For more complicated shapes, one can augment the training set and then re-train the DLDFM framework, which can then be used for analyzing complex

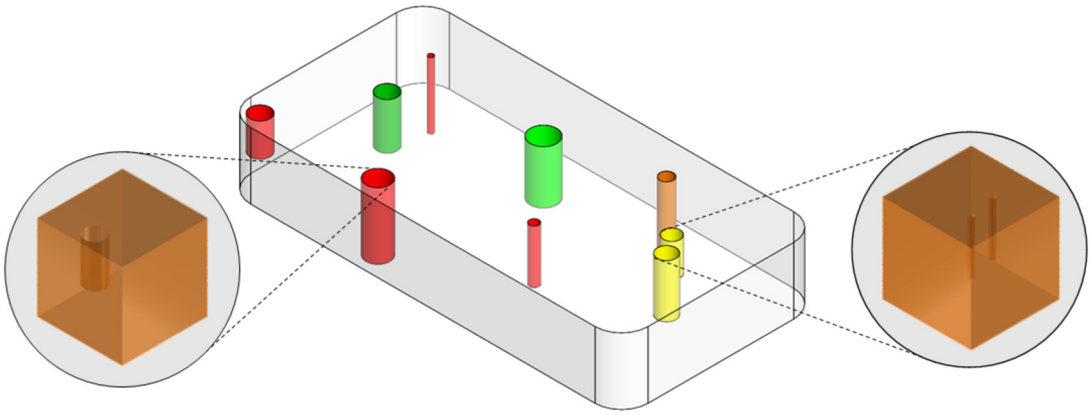


Fig. 3. Large scale CAD model to test DLDFM network. The holes in *red* are non-manufacturable and the *green* are manufacturable based on the DFM rules as discussed in Section 2. *Orange* colored hole is a marginal case of manufacturability. The two *yellow* holes are manufacturable individually but are non-manufacturable due to their proximity to each other. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

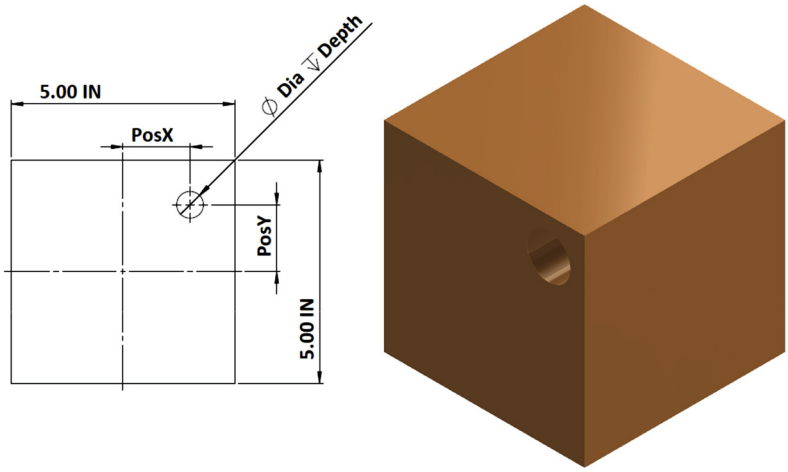


Fig. 4. A sample block with a drilled hole with its dimensions highlighted in the projected view.

parts for manufacturability. This eliminates explicit hand-crafting of rules, ultimately leading to a better manufacturability analysis.

2.1. Training data

Based on the DFM rules for drilling, different synthetic sample solid models are generated using a CAD modeling kernel. A large component can be considered a union of these synthetic blocks as shown in Fig. 3. We use ACIS (Spatial Corporation, 2014), a commercial CAD modeling kernel to create the solid models. Cuboids and Cylinders of different sizes that fit in a bounding box of 5.0 inches with different sizes of drilled holes were created (Fig. 4) as the training set. The diameter of the holes is randomly chosen from 0.1 in. to 1.0 in. with a discrete resolution of 0.1 in. Similarly, the depth of the hole is randomly chosen from 0.5 in. to 5.0 in. with an discrete resolution of 0.1 in. The holes are generated at different positions on the face of the cube by varying the value of $PosX$ and $PosY$ randomly with a discrete resolution of 0.1 in. (Fig. 4). In addition, the holes are generated randomly on any of the six faces. Repetition of the models is not allowed and thus, all the models generated are completely different from other. A total of 800,000 (approx.) possible models can be generated. Taking into advantage of the fact that 3D-CNN can learn the features and thus would not need much data about the variation of these models, we generate only a subset of the models. After the CAD models are generated using the solid modeling kernel, they are classified for manufacturability using the DFM rules for drilled holes.

After the B-Rep models are generated using the CAD modeling kernel, they are converted to volume representations. One of the framework design choices is to choose an appropriate voxel grid for the model. A fine voxelization of the model, while capturing all the features accurately, might be computationally expensive for training the 3D-CNN. Even a voxelization

resolution of $64 \times 64 \times 64$ pushes the limits of the GPU and CPU memory during the training, and hence, the parameters of the 3D-CNN have to be tuned for optimal performance. The complete data is split into training set and validation set.

To test the performance of the DLDFM network, a new set of CAD models were generated. The CAD geometry generated in this set is different from the CAD models used in training the DLDFM. The models generated in test-set do not have the same depth or diameter values as those in the models of training set. For the test data, the following parameters are varied to generate the samples:

- Diameter values from 1.1 to 1.5 are used for the representative test data. Diameter values from 0.1 to 1.0 inches were used for training.
- Position of the holes that are not used while training.

In addition to that we generate a few more models for testing that does not resemble training data. These models have geometries with the same primary hole parameters (depth, diameter, and position of the hole), but having additional or different external features. The details of the geometries in the non-representative data set is given below.

Multiple holes: The DLDFM has been trained to analyze the manufacturability of a single drilled hole. However, in a designed component, the features may not be independent; there can be multiple features, each of which may or may not be manufacturable. Moreover, it is possible that each of the features themselves are manufacturable, but due to their proximity or interaction with other features, the part may become non-manufacturable. Hence, we test the ability of the DLDFM framework to analyze the manufacturability of a part with two holes.

L-shaped blocks: All the models in the training set have an external cubical or cylindrical shape. Hence, to test the capability of DLDFM to capture the manufacturability of a hole irrespective of the external geometry, we use L-shaped Block and cylinders with holes. The rules established in Section 2 also apply to this geometry.

3. Volumetric representations for learning geometric features

Traditional CAD systems use boundary representations (B-Reps) to define and represent the CAD model (Krishnamurthy et al., 2009). In B-Reps, the geometry is defined using a set of faces that form the boundary of the solid object. B-Reps are ideally suited for displaying the CAD model by first tessellating the surfaces into triangles and using the GPU to render them. However, learning spatial features using B-Rep can be challenging, since the B-Rep does not contain any volumetric information.

3.1. Voxel representation of geometry

The use of voxelized shape representation allows a digital representation of the CAD model, where each voxel of the model can be represented using a binary digit corresponding to the voxel being inside or outside the model. Using this method, the entire model can be represented using a long string of binary digits that can be used as input for training the machine learning network.

In our framework, we convert the B-Rep CAD model to a volumetric occupancy grid of voxels. However, voxelizing a B-Rep CAD model is a compute intensive operation, since the center of each voxel has to be classified as belonging to either inside or outside the model. In addition, thousands of models need to be voxelized during training. Traditional CPU voxelization algorithms are too computationally slow for training the machine learning network in a reasonable time frame. Hence, we have developed methods for accelerated voxelization of CAD models using the graphics processing unit (GPU). These GPU methods are more than 100x faster than the existing state-of-the-art CPU-based methods and can create a voxelized representation of the CAD model with more than 1,000,000,000 voxels. Having a high resolution voxelization will enable us to capture small features in a complex CAD model.

To create the voxelized CAD model, we construct a grid of voxels in the region occupied by the object. We then make use of a rendering-based approach to classify the voxel centers as being inside or outside the object. A 2D example of the method is shown in Fig. 5; the method directly extends to 3D. The CAD model is rendered slice-by-slice by clipping it while rendering. Each pixel of this clipped model is then used to classify the voxel corresponding to the slice as being inside or outside the CAD model. This is performed by counting the number of fragments that were rendered in each pixel using the stencil buffer on the GPU. After the clipped model has been rendered, an odd value in the stencil buffer indicates that the voxel on the particular slice is inside the CAD model, and vice versa (Fig. 5). The process is then repeated by clipping the model with a plane that is offset by the voxel size. Once all the slices have been classified, we get the complete voxelized representation of the CAD model.

The time taken to perform the classification is the sum of the time taken to tessellate the model once and the total time taken to render each slice. As an example, the total time taken to voxelize the hole block is 0.133 seconds. These timings are obtained by running our voxelization algorithm on a Intel Xeon CPU with 2.4 GHz processor, 64 GB RAM, and an NVIDIA Titan X GPU.

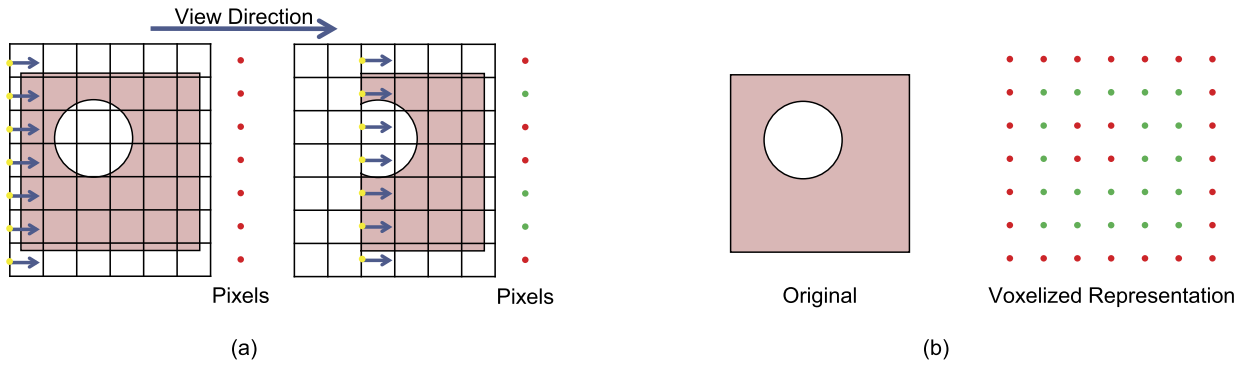


Fig. 5. Performing voxelization in 2D using GPU rendering. (a) A clipped CAD model is rendered slice-by-slice and the number of rendered pixels is counted. (b) The pixels that are rendered an odd number of times in each slice are inside the object (green).

3.2. Augmenting volume representation with surface normals

While a digital voxelized representation may be sufficient for a regular object recognition problem, it may not be enough to capture the detailed geometric information required for identifying local features of interest within the object. In particular, information about the boundary is lost in a voxel grid and the local region around a voxel grid need to be analyzed to identify a boundary voxel. To overcome this challenge in voxelized representations, we augment the voxel occupancy grid with the surface normals of the B-Rep geometry. First, we identify the boundary voxels of the B-Rep geometry. We consider each voxel as an axis-aligned bounding-box (AABB) using the center location and size of each voxel. We then find all the triangles of the B-Rep model that intersect with the AABBs. Finally, we average the surface normals of all triangles that intersect with each AABB. The $x, y,$ & z components of the surface normals are then embedded in the voxelization along with the occupancy grid.

4. 3D-CNN for learning localized geometric features

The voxel-based representation with the surface normals of the CAD model can be used to train a 3D-CNN that can identify local features. 3D-CNNs have mostly been used for complete 3D object recognition and object generation problems. However, to the best of the authors' knowledge, this is the first application of 3D-CNNs on identifying local features with applications in manufacturability analysis.

4.1. Network architecture and hyper-parameters

The input to the 3D-CNN is a voxelized CAD model. The input volumetric data is first padded with zeros before convolution is performed. Zero padding is necessary in this case to ensure that the information about the boundary of the CAD model is not lost while performing the convolution. The convolution layer with RELU activation is followed by a batch normalization layer and a Max. Pooling layer. The same sequence of Convolution, batch normalization and Max. Pooling is again used. A fully connected layer is used before the final output layer with sigmoid activation. The hyper-parameters of the 3D-CNN that needs to be tuned in order to ensure optimal learning. The specific hyper-parameters used in our framework are listed in Section 6.

The model parameters θ, comprised of weights $\mathbf{W}$ and biases, $\mathbf{b}$ are optimized by error back-propagation with binary cross-entropy as the loss function (Hinton and Salakhutdinov, 2006) using the ADADELTA optimizer (Zeiler, 2012). Specifically, the loss function ℓ to be minimized is:

$$\ell = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \tag{1}$$

where $y \in \{0, 1\}$ is the true class label and $\hat{y} \in \{0, 1\}$ is the class prediction.. For training the network, the CAD models that were generated based on the method illustrated in section 2 were used.

5. Interpretation of 3D-CNN output

The trained 3D-CNN network can be used to classify the manufacturability of any new geometry and can be treated as a black-box. However, interpretability and explainability of the output provided by the 3D-CNN is essential to understand the features learned by the model. In this paper, we attempt to visualize the input features that lead to a particular output and if possible modify it. A similar approach was used in object recognition in images by using class activation maps to obtain class specific feature maps (Selvaraju et al., 2016). The class specific feature maps are obtained by taking a class

discriminative gradient of the prediction with respect to the feature map to get the class activation. In this paper, we present the first application of gradient weighted class activation map (3D-GradCAM) for 3D object recognition.

In order to get the feature localization map using 3D-GradCAM, we need to compute the spatial importance of each feature map A_l in the last convolutional layer of the 3D-CNN, for a particular class, c (c can be either non-manufacturability or manufacturability, for the sake of generality) in the classification problem. This spatial importance for each feature map can be interpreted as weights for each feature map; it can be computed as the global average pooling of the gradients back from the specific class of interest as shown in Eq. (3).

The cumulative spatial activations that contribute to the class discriminative localization map, $L_{3DGradCAM}$, is computed using

$$L_{3DGradCAM} = ReLU\left(\sum_l \alpha_l \times A^l\right), \tag{2}$$

where α_l are the weights computed using

$$\alpha_l = \frac{1}{Z} \times \sum_i \sum_j \sum_k \frac{\partial y^c}{\partial A^l_{ijk}}. \tag{3}$$

We can compute the activations obtained for the input part using $L_{3DGradCAM}$ to analyze the source of output. The heat map of ($L_{3DGradCAM}$) is resampled using linear interpolation to match the input size, and then overlaid in 3D with the input to be able to spatially identify the source of non-manufacturability. This composite data is finally rendered using a volume renderer.

We make use of a GPU-based ray-marching approach to render this data. The rendering is parallelized on the GPU with each ray corresponding to the screen pixel being cast independently. The intersection of the ray with the bounding-cube of the volumetric data is computed, and then the 3D volumetric data is sampled at periodic intervals. The sum of all the sampled values along the ray is then computed, converted to RGB using a suitable color-bar, and rendered on the screen. Table 5 shows different volumetric renderings of the composite 3D-GradCAM data.

6. Results and discussion

The different CAD geometries generated as explained in Section 2.1 are classified to be manufacturable or non-manufacturable based on the rules discussed in Section 2. The B-Rep CAD geometries are converted to volumetric representation using the voxelization method explained in the Section 3. The grid size of $64 \times 64 \times 64$ is used for the volumetric representation in order to represent the geometry with sufficient resolution. We first trained the DLDFM network using only the voxelized representation of the CAD geometry. We also trained another network to use the surface normal information (i.e. x, y, z components) in addition to the voxelized representation of the CAD geometry. These are considered as *four* channels of the 3D-CNN input to train the second DLDFM network.

We generated 9531 CAD models in total for the training and validation set. Out of these, 75% of the models were used for training the 3D-CNN and the remaining 25% of the models were used for validation or fine-tuning the hyper-parameters of the 3D-CNN. A detailed description of the training process is provided in Section 4. The trained DLDFM network is then tested using the test set CAD models. The test set contains 675 geometries.

6.1. Voxelization timings

We recorded the time taken to generate the volumetric representation of the B-Rep CAD geometries at the voxel resolution used for the training. There is a slight dependence of the voxelization timings on the number of triangles in the CAD model, but it is predominantly dependent on the voxel grid resolution. The voxelization timings for the different CAD models shown in Table 5 is shown in Table 1. The table also includes the voxelization timings of the two large CAD models that are also used for testing the DLDFM.

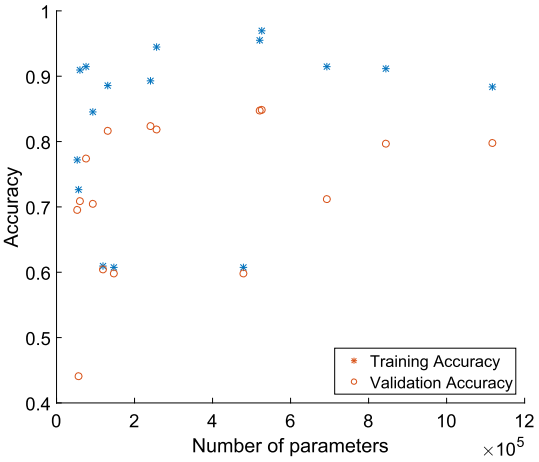
6.2. Tuning of the hyper-parameters

Finding the best hyper-parameters of the DLDFM is very important for the performance of the network. The hyper-parameters of DLDFM are fine-tuned to have least validation loss. Tweaking the hyper-parameters such as the number of layers, number of filters in each layer and filter sizes, the optimizer, loss function etc. is important to obtain the best possible results. Fig. 6a shows the variation of training and validation accuracy as a function of the number of parameters. The parameters with the best training and validation accuracy is then selected to be the final DLDFM network. Another important aspect of the training process is batch normalization, which allows for better training. Without batch normalization, the network saturates, and the training reaches a saddle point leading to low training accuracy as shown in Fig. 6b.

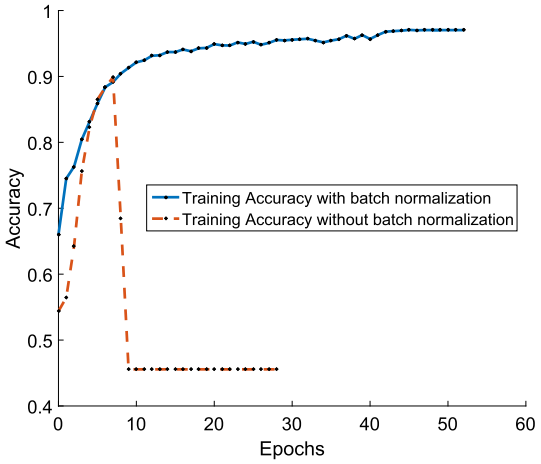
A batch size of 64 is selected while training the networks. The training was performed using Keras (Chollet, 2015) with a TensorFlow (Abadi et al., 2015) backend in Python environment. The training of DLDFM networks was performed

Table 1
Timings and voxel count of the voxelization of the CAD geometries shown in Table 5 and the large-scale model.

CAD Geometry	Triangle Count	Total Voxel Resolution	Inside Voxels	Time (s)
(a)	61688	262,144	106,240	0.177
(b)	59036	262,144	108,755	0.154
(c)	68694	262,144	110,592	0.179
(d)	63396	262,144	110,238	0.160
(e)	50544	262,144	72,778	0.150
(f)	51958	262,144	72,960	0.165
(g)	46582	262,144	88,293	0.151
(h)	68704	262,144	110,536	0.153
Part 1	57914	4,718,592	4,451,177	2.074
Part 2	96994	9,437,184	6,943,570	6.677



(a) Accuracy of DLDFM with different hyperparameters as a function of total number of parameters in each network.



(b) Comparison of the training performance with and without batch normalization for each layer.

Fig. 6. Hyper-parameter optimization of DLDFM.

Table 2
Optimized hyper-parameters with the least validation loss.

Common Training Parameters	DLDFM Network	Hyper-parameters
Batch Size: 64 Optimizer: Adadelta Loss Function: Cross-Entropy	Voxel	Convolutional Layer 1 (8 filters with size 8×8) Max Pooling Layer 1 (sub-sampling of 2×2) Convolutional Layer 2 (12 filters with size 4×4) Max Pooling Layer 2 (sub-sampling of 2×2) Convolutional Layer 3 (16 filters with size 2×2) Fully Connected Layer (128 neurons)
	Voxel + Normals	Convolutional Layer 1 (8 filters with size 8×8) Max Pooling Layer 1 (sub-sampling of 2×2) Convolutional Layer 2 (12 filters with size 4×4) Max Pooling Layer 2 (sub-sampling of 2×2) Convolutional Layer 3 (16 filters with size 2×2) Fully Connected Layer (128 neurons)

in a workstation with 128GB CPU RAM and a NVIDIA Titan X GPU with 12GB GPU RAM. The training is performed until the validation loss remains constant for at least 10 consecutive epochs. The final architecture of the DLDFM network is described in the Table 2. The trained DLDFM network is then tested using the test set CAD geometries, which contains 675 CAD models.

6.3. Results from test data set

After successful training, the DLDFM network was tested on a test set to benchmark its performance. Accuracy of DLDFM network on the test set using the two data representations is shown in Table 3 and Fig. 7. The test-set has completely

Table 3
Quantitative performance assessment of the DLDFM on test data sets.

Test Data Type	Model Description	True Positive	True Negative	False Positive	False Negative	Accuracy
675 models 408 Manufacturable	In–outs	391	90	17	176	0.7136
	In–outs + Surface Normals	334	201	74	65	**0.7938**

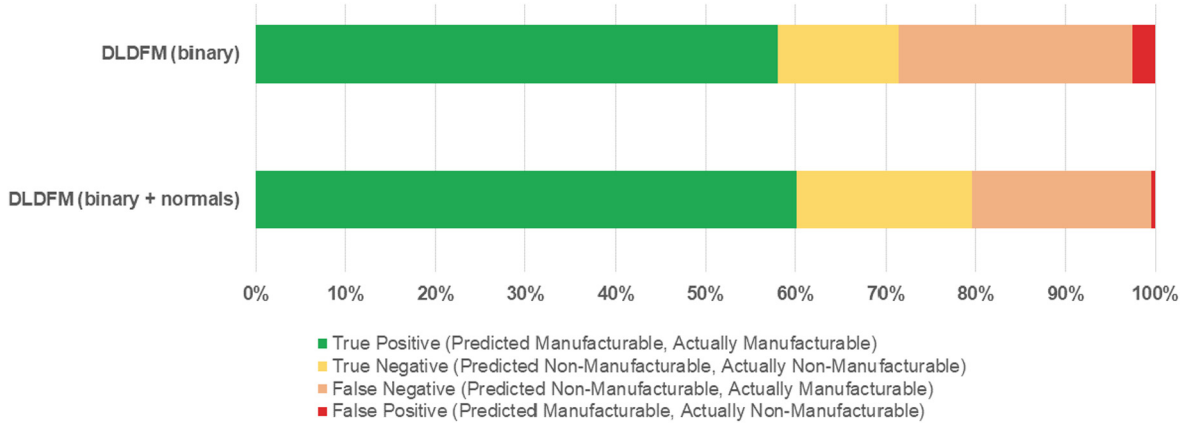


Fig. 7. Confusion matrix for the DLDFM performance on test data set. Using surface normals along with binary voxels increases overall prediction accuracy by $\approx 8\%$.

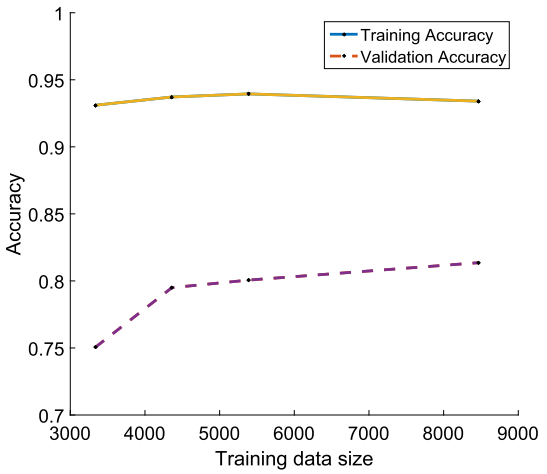


Fig. 8. Variation of accuracy with no. of training samples used for.

different geometries compared to the training set. Thus, the DLDFM is learning the *localized geometric features* that directly contribute to the manufacturability or the non-manufacturability of the CAD geometry.

Using the trained DLDFM network, it is possible to obtain the localization of the feature activating the decision of the DLDFM. The 3D-GradCAM renderings for various cases are shown in the Table 5. We have used 3D-GradCAM to visualize the results of various inputs such as manufacturable holes, non-manufacturable-holes, multiple holes in same face, and holes in multiple faces of the cube. 3D-GradCAM can localize the features that can cause the part to be non-manufacturable. For example, in Table 5, the second example shows a CAD model with a hole, which is non-manufacturable because it is too close to one of side faces. This is a difficult example to classify based only on the information of the hole. The 3D-GradCAM rendering correctly identifies the non-manufacturable hole and as a result the DLDFM network also predicts the part to be non-manufacturable.

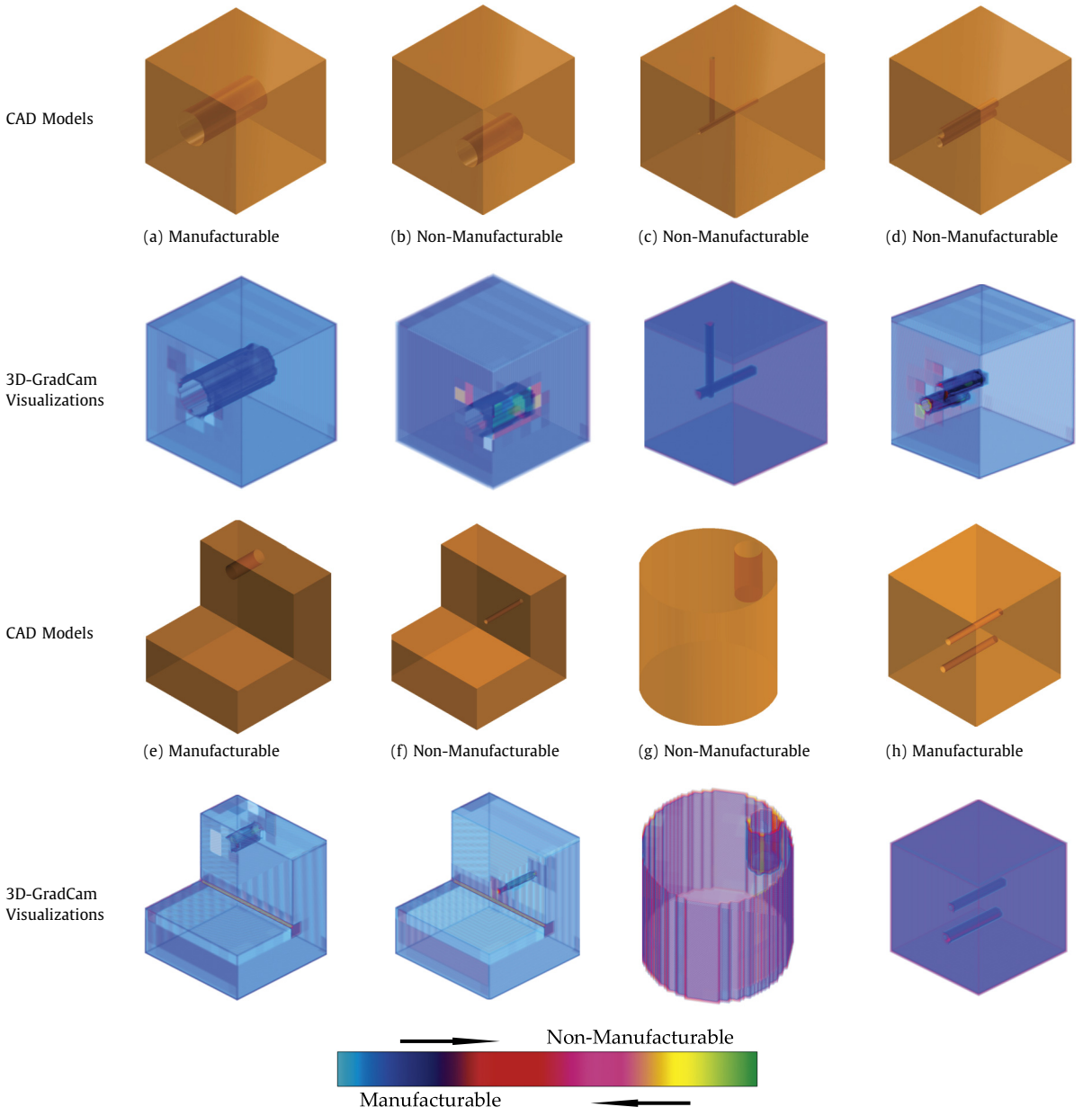
6.4. Relationship between data size and accuracy

We performed an experiment the assess the effect of number of data samples on training the network. As discussed earlier, the training samples were selected from a set of more than 800,000 CAD models. We trained the DLDFM network with different number of training samples and recorded the variation in the training and validation accuracy (Fig. 8). The

Table 4
Comparison of the performance of the DLDFM on the training data with traditional DFM based only on the depth to diameter ratio of the drilled holes.

CAD Models	3334		4361		5387		8405	
DFM	Hole ratio	DLDFM	Hole ratio	DLDFM	Hole ratio	DLDFM	Hole ratio	DLDFM
True Positive	575	684	912	1053	1145	1339	1875	2222
True Negative	676	868	821	990	1011	1191	1568	1731
False Positive	165	59	208	67	266	73	316	126
False Negative	251	56	239	70	271	90	473	153
Training accuracy	0.7504	**0.9310**	0.7949	**0.9372**	0.8006	**0.9394**	0.8136	**0.9340**

Table 5
Illustrative examples of manufacturability prediction and interpretation using the DLDFM framework.



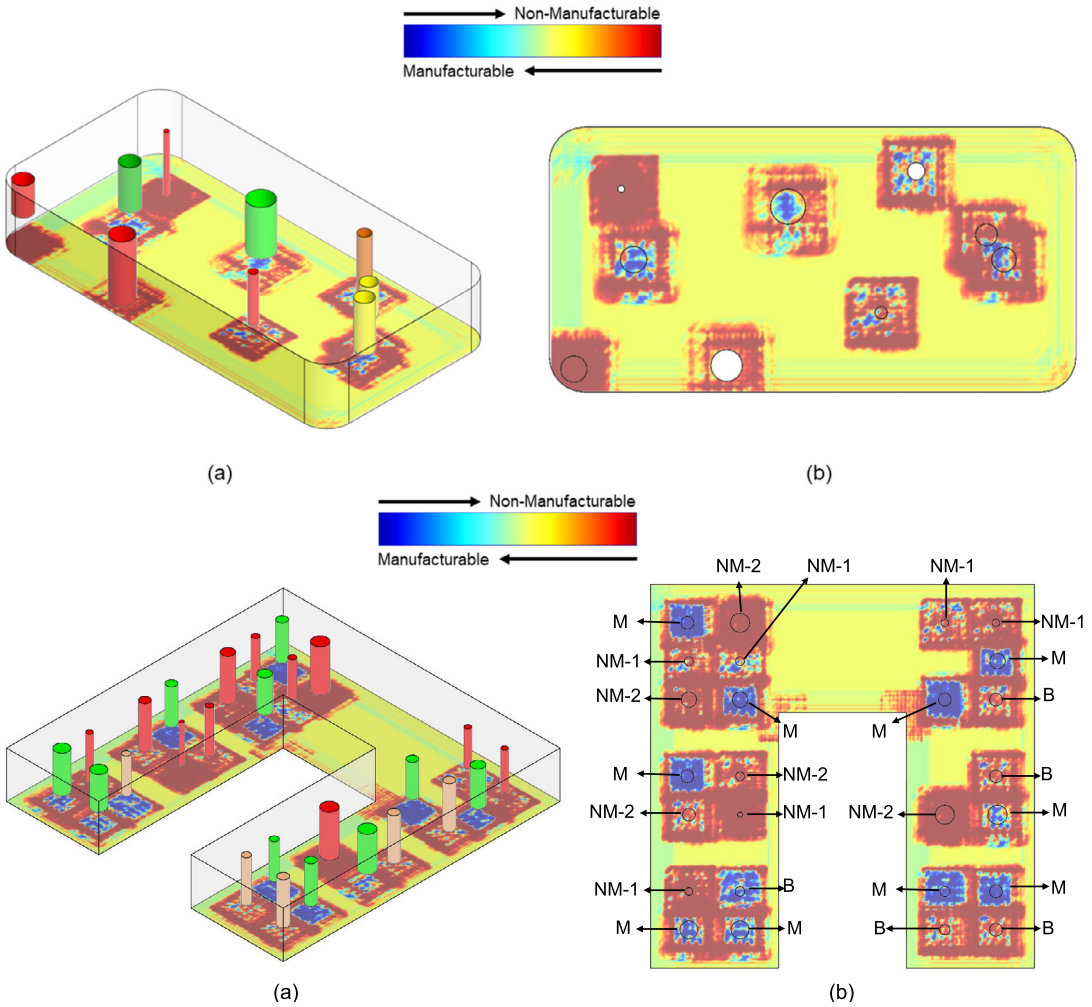


Fig. 9. Test results on the large scale CAD model using DLDFM framework to detect manufacturable and non-manufacturable drilled holes. (a) Isometric view of the test case with DLDFM inference mapped to the CAD model for comparison. (b) Top view with the holes aligned with the inference map. Each hole is marked as M – manufacturable or as NM – non-manufacturable based on the reason for non-manufacturability: NM-1 – non-manufacturable due to D/d ratio; NM-2 – Non-manufacturable due to small amount of material in the depth direction; B – borderline cases.

training accuracy does not vary, while the validation accuracy improves with the addition of more training data. The only limitation in increasing the training data size is computational cost of training from the large data.

In order to compare our DLDFM method with other feature based detection method, we need to assume prior knowledge of the geometry of the feature. We compare the performance of DLDFM with a hole-ratio based feature detection system, since the hole-ratio is the most dominant feature that can be used for feature recognition. This is found in many CAD tools such as SolidWorks (SolidWorks Corp., 2017). A hole-ratio based manufacturability analysis will fail for other thickness based rules discussed in Section 2, which makes it non-reliable. However including such thickness-based checks is not trivial for such a hand-crafted system. The false positive and false negative examples for hole-ratio based system is higher than DLDFM as shown in Table 4 thus demonstrating the reliability of DLDFM and its capability to learn multiple complex features, which one would have to hand-craft to use a feature detection algorithm to analyze the manufacturability of a part.

6.5. Manufacturability analysis of a realistic part

In order to infer the manufacturability of a large scale realistic CAD model, we use the trained 3D-CNN model (Section 4) to infer the manufacturability of small slices of the voxelized geometry. If the initial grid resolution chosen for the training is $l \times m \times n$, then the large scale CAD model is voxelized using the same grid size, resulting in a larger grid resolution, $l_1 \times m_1 \times n_1$. We pad this voxel grid with $l/2, m/2, n/2$ voxels of zero value in all the directions to correctly analyze the boundaries. We then slice $l \times m \times n$ voxels to obtain one inference; we stride by one in each direction to obtain a map of size $l_1 \times m_1 \times n_1$ inferring the spatial non-manufacturability of the large CAD model. Note that some of the slices

having partial coverage of the features will be inferred as non-manufacturable, thus raising a false alarm, but when the hole is bounded correctly by the slice, the inference is comparable to the performance on the previous training data set. As shown in the Fig. 9 there is a square region surrounding the features identified in the large CAD model, with false non-manufacturability inference. However, the central region in the patch indicates the true manufacturability inference of the feature, i.e. whether the drilled hole is manufacturable or not. According to the color map shown in Fig. 9, the central blue colored regions inside holes corresponds to their manufacturability and likewise the central red colored regions correspond to non-manufacturability. Comparing the inference of the DLDFM with the original CAD model in Fig. 3 we see that all the green (manufacturable) holes align to the blue regions in the inference and the red (non-manufacturable) holes align to the red regions as expected. The same is true in another example of a complex large part with many holes resembling an injection mold die with cooling holes.

7. Conclusions

In this paper, we demonstrate the feasibility of using 3D-CNNs to identify local features of interest using a voxel-based approach. The 3D-CNN was able to learn local geometric features directly from the voxelized model, without any additional shape information. As a result, the 3D-CNN was able to identify the local geometric features irrespective of the external object shape. Hence, a 3D-CNN can be used effectively to identify local features, which in turn can be used to define a metric that may be used for successful object classification. In addition, using the 3D-GradCAM eliminates the black box notion about CNNs; the DLDFM framework provides feedback about the source of non-manufacturability. The feedback is helpful to understand which particular local feature among various other features in a CAD geometry accounts for the non-manufacturability and possibly modify the design appropriately.

We apply our local feature detection tool to build a deep-learning-based DFM (DLDFM) framework, a novel application of deep learning for cyber-enabled manufacturing. To the best of our knowledge, this is the first application of deep learning to learn the different DFM rules associated with design for manufacturing. In this paper, our DLDFM framework was able to successfully learn the complex DFM rules for drilling, which include not only the depth-to-diameter ratio of the holes but also their position and type (through hole vs. blind). The framework can be extended to learn manufacturable features for a variety of manufacturing processes such as milling, turning, etc. We envision training multiple networks for specific manufacturing processes, which can be concurrently used to classify the same design with respect to their manufacturability using different processes. Thus, an interactive decision-support system for DFM can be integrated with current CAD systems, which can provide real-time manufacturability analysis while the component is being designed. This would decrease the design time, leading to significant cost-savings.

Acknowledgements

This paper is based upon research partially supported by the National Science Foundation under Grant No. CMMI: 1644441. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GTX TITAN Xp GPU used for this research. The data and code used in this paper can be found at http://web.me.iastate.edu/adamlab/Index.html.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., 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., 2015. TensorFlow: large-scale machine learning on heterogeneous systems. http://tensorflow.org/. Software available from tensorflow.org.

Boothroyd, G., Dewhurst, P., Knight, W., 2002. Product Design for Manufacture and Assembly. M. Dekker.

Bralla, J.G., 1999. Design for Manufacturability Handbook. McGraw-Hill.

Chollet, F., 2015. Keras. https://github.com/fchollet/keras.

Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.

Huang, J., You, S., 2016. Point cloud labeling using 3D convolutional neural network. In: Proc. of the International Conf. on Pattern Recognition (ICPR), vol. 2.

Krishnamurthy, A., Khardekar, R., McMains, S., 2009. Optimized GPU evaluation of arbitrary degree NURBS curves and surfaces. Comput. Aided Des. 41 (12), 971–980.

Lee, H., Grosse, R., Ranganath, R., Ng, A.Y., 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM, pp. 609–616.

Lore, K.G., Akintayo, A., Sarkar, S., 2017. Llnet: a deep autoencoder approach to natural low-light image enhancement. Pattern Recognit. 61, 650–662.

Maturana, D., Scherer, S., 2015a. 3D convolutional neural networks for landing zone detection from LIDAR. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 3471–3478.

Maturana, D., Scherer, S., 2015b. Voxnet: a 3d convolutional neural network for real-time object recognition. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. IEEE, pp. 922–928.

Riegler, G., Ulusoys, A.O., Geiger, A., 2016. Octnet: learning deep 3d representations at high resolutions. ArXiv preprint arXiv:1611.05009.

Sarkar, S., Venugopalan, V., Reddy, K., Giering, M., Ryde, J., Jaitly, N., 2015. Occlusion edge detection in rgb-d frames using deep convolutional networks. In: Proceedings of IEEE High Performance Extreme Computing Conference. Waltham, MA.

Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., Batra, D., 2016. Grad-cam: why did you say that? Visual explanations from deep networks via gradient-based localization. ArXiv preprint arXiv:1610.02391.

Socher, R., Huval, B., Bath, B., Manning, C.D., Ng, A.Y., 2012. Convolutional-recursive deep learning for 3D object classification. In: Advances in Neural Information Processing Systems, pp. 656–664.

SolidWorks Corp., 2017. SolidWorks. www.solidworks.com.

Spatial Corporation, 2014. ACIS Geometric Modeler: User Guide, Version 26.0.
Wang, C., Siddiqi, K., 2016. Differential geometry boosts convolutional neural networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 51–58.
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 28th IEEE Conference on Computer Vision and Pattern Recognition. CVPR2015.
Zeiler, M.D., 2012. Adadelta: an adaptive learning rate method. ArXiv preprint arXiv:1212.5701.
Zhu, Z., Wang, X., Bai, S., Yao, C., Bai, X., 2016. Deep learning representation using autoencoder for 3D shape retrieval. Neurocomputing 204, 41–50.