3D model classification using convolutional neural network

JunYoung Gwak
Stanford
jgwak@cs.stanford.edu

Abstract

Our goal is to classify 3D models directly using convolutional neural network. Most of existing approaches rely on a set of human-engineered features. We use 3D convolutional neural network to let the network learn the features over 3D space to minimize classification error. We trained and tested over ShapeNet dataset with data augmentation by applying random transformations. We made various visual analysis to find out what the network has learned. We extended our work to extract additional information such as pose of the 3D model.

1. Introduction

Consider a 3D model in Figure 1. Just with a quick glimpse of it, we can accurately classify that this is a sofa. This ability to infer the class of a 3D model is important for various problems interacting with the real world such as autonomous driving. Likewise, the class of the 3D model provides cues to pose, grasping, affordance, and other properties. For similar reasons, object classification of a 2D image has traditionally been one of the most tackled problems in computer vision. However, there has been a little effort to classify a 3D model as a whole. This is partially due to lack of training data and difficulty of building intuitive hand-engineered features on 3D space.

In this paper, we prepared training data using ShapeNet[2] dataset - a richly-annotated, large-scale dataset of 3D shapes - along with data augmentation by applying random transformations to the model on the fly. Moreover, we trained a 3D convolutional neural network to let the network learn the best features over 3D space to classify 3D models. Then, we made a thorough analysis of the network to investigate what the network has learned. Additionally, we extended this work by extracting additional semantic information of the model such as pose.

1.1. Related Work

Some of the previous works involve classification of 3D objects over intuitive hand-engineered features[1, 3, 6]. In our work, we let the network learn the features by training a 3D convolutional neural network.

There exists many works which utilizes convolutional neural network to classify 2D images[4, 5]. Our work shares similar motivation and network structure. However, our input is in 3D and we convolute over 3D space to directly extract 3D features.

Wu et al.[7] approached this problem using a generative model based on a probability distribution using convolutional deep belief net. Our approach is a discriminative model learning a direct mapping from voxel to classification.

For analysis of the network, we plotted the activation and the gradient of channel with respect to data[8]. Our work verified that the same technique can be applied to visualize 3D convolutaionl networks.

2. Method overview

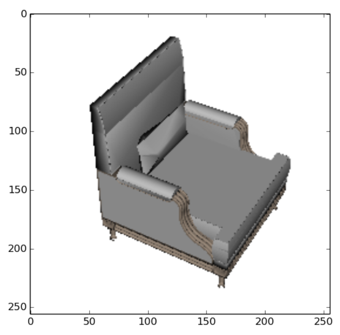
2.1. Input data

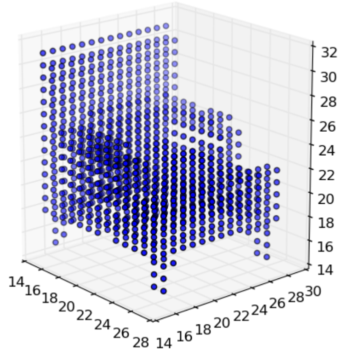
One of the challenges of training a neural network is in preparing a large amount of input data. Our initial concern has been lifted with advent of ShapeNet[2] - richly-annotated, large-scale dataset of 3D shapes. Among 3 million models in it, we used a part of the ShapeNetCore, 18 common daily categories with more than 45000 3D models.

Since we train the network over many iterations, we need a lot more data than ShapeNetCore. Therefore, we virtually augment the dataset by applying random transformation as followed.

$$V_{\text{transformed}} = \begin{bmatrix} \cos(r) & \sin(r) & 0 & tx \\ -\sin(r) & \cos(r) & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & s \end{bmatrix} \cdot V$$

where V is a matrix of vertices in homogeneous space, r is degree of yaw rotation, tx, ty, tz is degree of x, y, z translation, and s is scale of the transformed model. We randomly chose the parameters above to augment the data. Our rotation ranges from $-\pi$ to π, translation from 0 to 0.2,


(a) 3D model


(b) Voxels

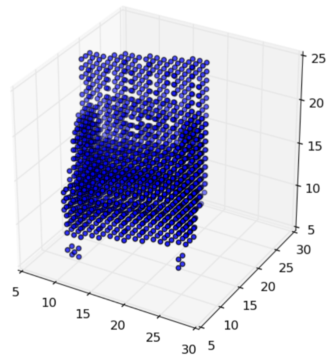

(c) Data-augmented Voxels

Figure 1: Sample input data pipeline. We first augment the input data by applying random transformation to the 3D model. Then, we voxelize the augmented model to fix the input shape of the neural network.

Layer	Filter shape	Output shape
input		(32, 1, 32, 32, 32)
conv1	(32, 1, 3, 3, 3)	(32, 32, 32, 32, 32)
pool1	(2, 2, 2)	(32, 32, 16, 16, 16)
conv2	(64, 32, 3, 3, 3)	(32, 64, 16, 16, 16)
pool2	(2, 2, 2)	(32, 64, 8, 8, 8)
conv3	(128, 64, 3, 3, 3)	(32, 128, 8, 8, 8)
pool3	(2, 2, 2)	(32, 128, 4, 4, 4)
conv4	(256, 128, 3, 3, 3)	(32, 256, 4, 4, 4)
pool4	(2, 2, 2)	(32, 256, 2, 2, 2)
fc5	(1024)	(32, 1024)
fc6	(1024)	(32, 1024)
softmax7		(32, 18)

Table 1: Detailed network configuration. Please note that ReLU and Dropout layers always follows after convolutional and fully connected layers.

and scale from 0.7 to 1.2. A sample input data with data augmentation can be found in Figure 1.

Finally, neural network requires the input data to be in fixed shape. Therefore, mesh representation of the 3D model should be converted to a fixed-sized data without loss of 3D information of the model. Therefore, we convert the 3D model to a single channel binary voxel.

2.2. 3D convolutional neural network configuration

From an input 3D model, we wish to learn the correct class. We train the network in following two steps. First, we pre-process the data by applying random transformation to the mesh and then voxelize it. Then, we train the network by updating weight based on the gradient of the classification cost with respect to each parameter.

Our network is composed with the following layers:

1. **3D Convolutional layer**: 3D Convolutional layer convolutes the data by computing local weighted sum using learned weight. It takes input of size (batch size, channel size, x size, y size, z size) and convolutes with filter of size (filter size, input channel size, filter x size, filter y size, filter z size). The convolutional layer learns the 3D feature to be used for classification.

2. **Rectified Linear Unit**(ReLU): ReLU layer applies a simple activation function: $f(x) = \max(0, x)$ where x is the input to a neuron. This rectifier introduces non-linearity with very efficient computation and simple gradient propagation.

3. **Pooling**: Pooling layer reduces the size of the data by choosing the largest value of the pool. This layer allows us to reduce the size of the input and to learn abstract features efficiently.

4. **Fully Connected Layer**(FC): Fully connected layer takes all neurons in the previous layer and connects it to every single neuron it has. This layer allows high-level reasoning of low-level local features learned from preceding layers.

5. **Dropout Layer**: Dropout layer prevents overfitting by dropping out some unit activations in a given layer by setting them to zero.

6. **Softmax**: Softmax regression is a generalization of logistic regression to the case where we want to handle multiple classes. This final layer allows us to compute the regularized likelihood of each class based on a linear classifier.

All of the layers above are put together to build a 3D classification network. Please check Table 1 for details of the network configuration.

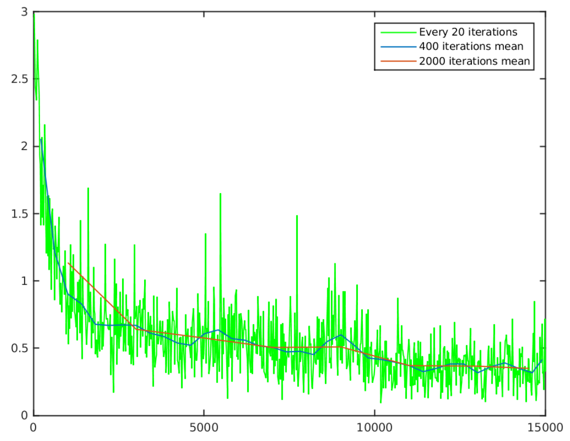


Figure 2: Training loss vs number of iterations.

2.3. Training the network

Given the training data and the network configuration above, we train the network to classify voxels. We update weights of each convolutional and fully connected layer using batch gradient descent with momentum. Following is the formulation of the update where n is the size of the batch, η is the learning rate, α is the momentum rate, $\triangle w$ is the momentum, and w is the weight.

$$\triangle w := \eta \sum_{i=1}^{n} \triangledown Q_i(w) + \alpha \triangle w$$
$$w := w - \eta \triangle w$$

While training, we tracked the change of training loss over number of training iterations. Then, we adaptively lowered the learning rate whenever the loss stopped decreasing. For example, in Figure 2, we lowered the training rate by 1/10 at 10000 iterations to decrease the training loss further down.

3. Result

We achieved test accuracy of 89.34%. Please check Figure 3 for confusion matrix of the classification result. As a baseline of this work, we can compare the classification accuracy with that of Wu et al.[7], who tried classification on a smaller set of ShapeNet. Wu et al. achieved total accuracy of 57.9% on 10 classes while ours achieved test accuracy of 89.34% on 18 classes. We achieved better result since our network learns discriminative model of class probability given voxels while Wu et al. learns generative model of the voxels and prior of the data to produce class likelihood.

From the confusion matrix, we observed two properties of the network. First, it prefers to output the classes with

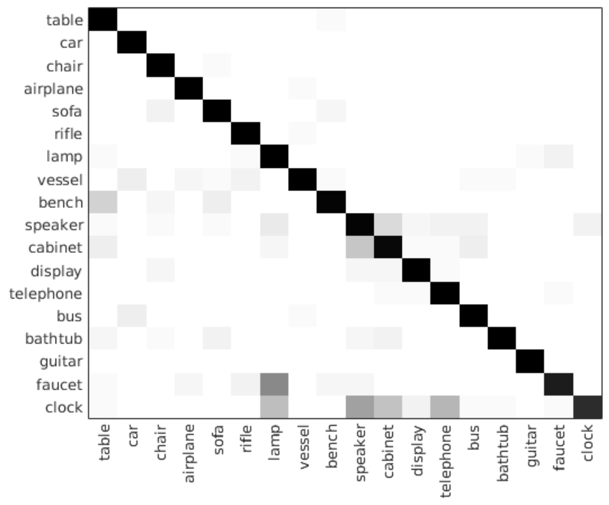


Figure 3: Confusion matrix of the classification results.

more models. This is due to a bias in the number of training data in each class that the softmax has learned. Second, it prefers to output the classes with simple shape. The classes that misleads others tend to have simple shape. For example, the network tends to output cabinet or speaker which usually has a simple blob over classes with complex shape such as airplane or car.

We made further analysis of the network with various approaches to visualize what the network has learned.

3.1. Analysis 1: activation output

We plotted the output of ReLU layers as voxel to visualize the activation.

Please check Figure 4 for some visualization of convolutional channel activations. We can observe that initial convolutional layers perform basic filtering operations, extracting simple shapes such as horizontal or vertical shapes.

3.2. Analysis 2: Gradient of activation

Next, we plotted the gradient of channel activation with respect to the input data. This will visualize which part of the input contributed to the activation of a specific channel.

Please check Figure 5 for some sample gradient of channel with respect to input data. First row, channel 66 of convolutional layer 4, seems to have learned leg. Similarly, channel 190 of convolutional layer 4 seems to have learned wheel of a vehicle. We can observe that deeper convolutional layers tend to learn high level concepts of the 3D objects.

3.3. Analysis 3: Inputs activating a channel

Finally, we took average of 10 models that has highest activation on each channel. This will again visualize the

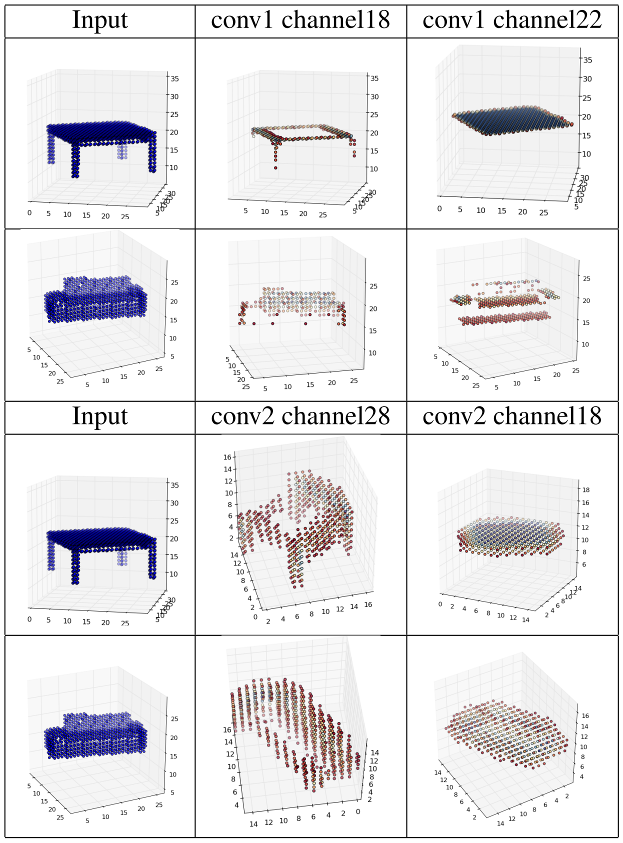


Figure 4: Channel activations

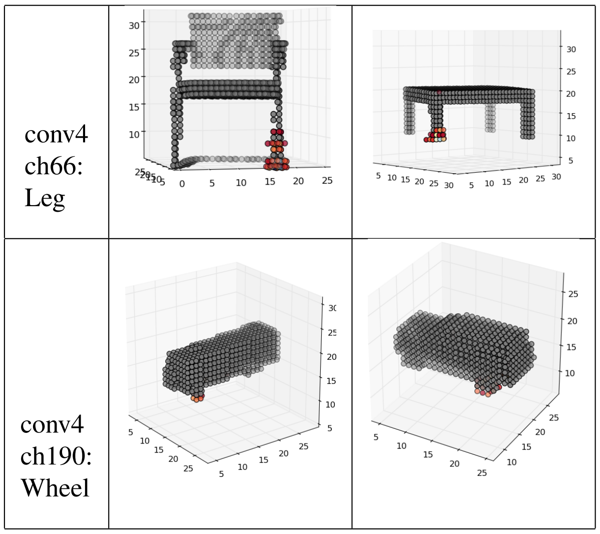


Figure 5: Gradient of channel with respect to input data

high level shape of what the channel has learned.

As shown in Figure 6, channel 4 learned interesting high-level shapes. Some learned dominant shape such as circle, rod, and rectangle plate. Others learned class specific shape

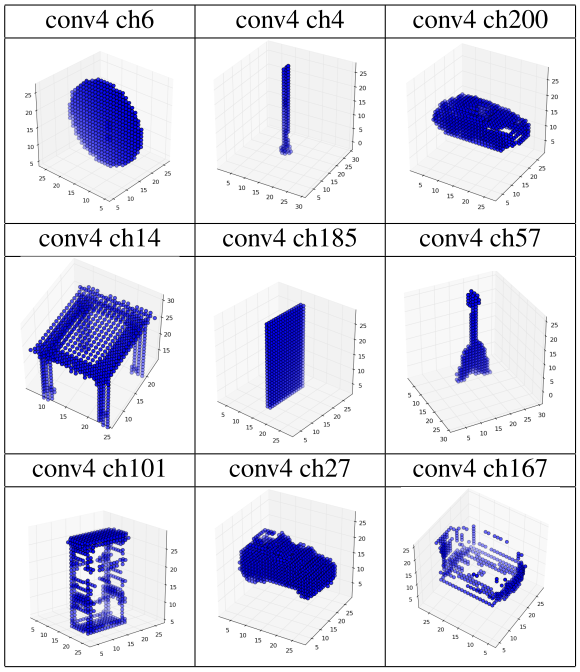


Figure 6: Average of top 10 input that activates the channel

such as car, desk, and guitar.

Through all of the analysis above, we found that the network learns simple filtering operations at the beginning of the convolutional layers and some high-level concept of the 3D models at the end of the convolutional layers.

4. Model pose estimation

As an extension of this work, we explored the possibility of the network extracting further information about the model. Specifically, we explored the possibility of our network estimating the pose of the 3D model along with the class.

In our data augmentation step in Section 2.1, we showed that a random yaw rotation has been applied to all of the training and testing data. We discretized the degree of rotation into 8 bins from $-\pi$ to π. Such discretized bins along with class is used as a new label of the data.

We achieved this goal by adding another fully connected layer and softmax layer at the end of the network. The second softmax layer classifies model pose into 8 bins as defined above. Then, we defined the training cost as following:

$$\text{total\_cost} = \text{classification\_cost} + \lambda \cdot \text{pose\_cost}$$

where $\lambda = 0.5$. The network structure and all other training sequences stay the same.

For this experiment, we achieved pose accuracy of 0.8736. As shown in Figure 7, most of the confusion came

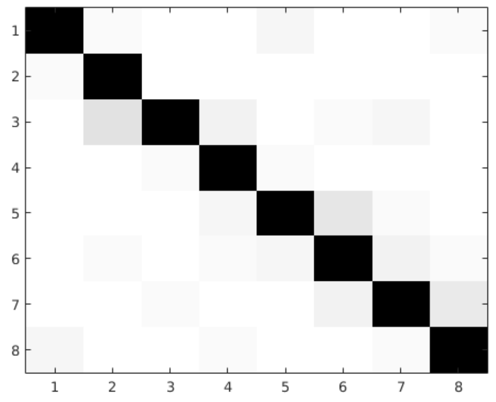


Figure 7: Confusion matrix of model pose estimation.

from pose nearby or 180° apart. This is a promising result suggesting the possibility of the network to extract various semantic data of the model in addition to its class.

5. Conclusion

We proposed a method to classify 3D models as a whole using 3D convolutional neural network. We used ShapeNet dataset along with data augmentation to prepare enough data to train the network over long iterations. We built and trained a network to achieve a reasonable test accuracy on our dataset. We made a thorough analysis on the network to investigate what the network learned. We found that the beginning convolutional layers tend to do simple filtering operations while higher convolutional layers tend to make high-level understanding of the 3D model. Moreover, we explored the possibility of the network to extract useful semantic data on top of model class from the input 3D model.

Acknowledgement

This project is part of a research at Computational Vision and Geometry Lab, Stanford. The author received advice about design and code from Christopher Choy(chrischoy@ai), a Ph.D. student in the lab. This project is also a joint project with CS221. I allocated advanced analysis of the network and pose estimation of the 3D model for CS229.

References

[1] I. Atmosukarto and L. G. Shapiro. A learning approach to 3d object representation for classification. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 267–276. Springer, 2008. 1

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1

[3] M. A. Kassimi, O. El Beqqali, S. Mohamed, and F. Morocco. 3d model classification and retrieval based on semantic and ontology. 2011. 1

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 1

[6] A. Teichman, J. Levinson, and S. Thrun. Towards 3d object recognition via classification of arbitrary object tracks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4034–4041. IEEE, 2011. 1

[7] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 1, 3

[8] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015. 1