3D Capsule Networks for Object Classification from 3D Model Data

Ayesha Ahmad, Burak Kakillioglu and Senem Velipasalar
Electrical Enginnering and Computer Science Department
Syracuse University
Syracuse, NY, USA
{aahmad,bkakilli,svelipas}@syr.edu

***Abstract*—Many of the existing object classification methods today rely on convolutional neural networks (CNNs), which are very successful in extracting features from the data. However, CNNs cannot sufficiently address the spatial relationship between features and require large amounts of data for training. In this paper, a new architecture is proposed for 3D object classification, which is an extension of the Capsule Networks (CapsNets) to 3D data. Our proposed 3D CapsNet architecture preserves the orientation and spatial relationship of the extracted features, and thus requires less data to train the network. We compare our approach with a ShapeNet inspired model, and show that our method provides performance improvement especially when training data size gets smaller. We also compare and evaluate several different versions of the 3D Capsnet architecture.**

***Index Terms*—Capsule networks, 3D, object classification.**

I. Introduction

Object recognition is a process for identifying an object in a digital image, 3D space or video. Object recognition algorithms typically rely on matching, learning, or pattern recognition algorithms using appearance-based or feature-based techniques. Object recognition comprises a deeply rooted and ubiquitous component of modern intelligent systems. The application of the technology related to 3D object recognition and analysis is increasing day-by-day. Moreover, the effectiveness of 3D object recognition is increasing as researchers are developing and implementing new algorithms, models and approaches. Application areas for which 3D object recognition is used include manufacturing industry, video surveillance, autonomous driving, urban planning, safety and control, and augmented reality.

Many of the existing 2D and 3D classification methods rely on convolutional neural networks (CNNs), which are very successful in extracting features from the data. However, CNNs cannot sufficiently address the spatial relationship between features due to the max-pooling layers, and they require large amounts of data for training. Deficiency of data is a pronounced concern while training using a pure CNN-based architecture. The problem becomes even more pronounced when it comes to 3D data. There is a dearth of annotated datasets available for 3D data to train models, with even fewer training data. Despite the advent of 3D sensors, the number of 3D datasets for classification is low. Most commonly used datasets contain artificial data created from Computer Aided Design (CAD) models. 3D CAD has several benefits, since it allows visualization and optimization of designs, avoids unnecessary costs due to human error, and provides reproducibility of the experiments under various conditions such as different viewpoints, different sizes etc. One such dataset is ModelNet [1].

It is indeed a challenge to be able to use the data collected from 3D scanners by first constructing 3D models and then recognizing objects in these models. There has been several CNN-based approaches for object classification from 3D data. However, CNN-based approaches require larger datasets. Capsule Networks (CapsNets) [2] have been introduced recently and has been tested on the MNIST [3], CIFAR [4] and Small NORB [5] datasets, which are 2D datasets with images. Capsules have encoding for poses and orientations of the object, i.e. neural activities are different for same objects with different poses. Results in [2] show that Capsule Networks are better at identifying multiple objects and also at generalizing among viewpoints than CNNs. The motivation behind development of capsules is close depiction of neurons arrangement in the brain.

In this paper, we propose a 3D CapsNet architecture, extended for 3D data, to address the problem of 3D object classification from 3D volumetric data. We also compare and evaluate several different architectures. The results show that 3D CapsNet provides promising results, which are better than a CNN-based approach, especially when smaller size training dataset is used.

II. Related Work

Despite its 2D structure, a depth image contains spatial information in 3D world. With the advent of 3D sensing devices, depth image understanding became popular in the literature. The intensity values in a depth image represent the distance of the object from a viewpoint. You can color code these to visually represent the close and far objects efficiently. Although depth images contain spatial information about the object or environment and are used for many 3D vision tasks, depth image-based methods are often regarded

The information, data, or work presented herein was funded in part by National Science Foundation (NSF) under Grant 1739748 and by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000940. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

as 2.5D approaches. Different from this, 3D voxel grid-based approaches were presented in the literature. One of the first examples of voxel grid-based 3D classification works is ShapeNet [1], wherein point clouds of CAD objects are voxelized into $30 \times 30 \times 30$ grids, and a CNN architecture, with 3D convolutional and 3D max pooling layers, is presented. Similar to 2D CNNs, it has stacked fully-connected layers after convolutional layers for object class prediction. VoxNet [6] approach also voxelizes every object from 12 viewpoints into $32 \times 32 \times 32$ voxel grids. Different from [1], they define three different encoding schemes for occupancy grid generation.

In addition to depth image-based and 3D voxel grid-based approaches, there are other methods for 3D object classification. PointNet [7] proposes a CNN that accepts raw point cloud data, and thus does not require voxelization. They sub-sample 2048 points from each point cloud, and give them as input object to CNN. They predict an affine transformation matrix by a mini-network and directly apply this transformation to the coordinates of input points. Another approach, MVCNN [8], uses 2D rendered images of 3D point clouds from many different viewpoints around gravitational axis. They train 2D CNNs to classify each render and train another CNN for aggregating multiple views better. Panorama [9], extracts panoramic view of 3D objects and defines a CNN architecture for predicting the classes of objects from their panoramic representations.

III. Capsule Networks

A. Capsules

A capsule is a group of neurons, which together perform internal computations on inputs, and encapsulate the output into a small vector, which is capable of representing different properties of the same entity. In [2], the deficiencies of previous object classification methods, such as CNNs and Scale Invariant Feature Transform (SIFT) [10], are discussed.

CNNs, in general, are invariant to features but not equivariant. Equivariance is the detection of objects that can transform to each other. For instance, if an object is rotated at an angle, then by the property of equivariance, capsules can recognize that the object is rotated at that angle, without necessitating training with that variation of image. This is a unique quality of capsules over its predecessor, CNNs, which required training with all variations of the object such as orientation, pixel intensities, scale etc.

Capsules have two primary components. One is the locally invariant probability that an entity is present. Second is the set of the instantiation parameters, also known as pose, that are equivariant. It is important to have these two constituents because they help in recognizing the whole object by recognizing their parts.

B. Squashing Function

The squashing function is applied to output of capsule to normalize the length of capsule vectors. It is a non-linearity function just like ReLU, Sigmoid, etc. However, unlike ReLU, which works well with scalars, squashing function has proven to work better with vectors which are the output of capsules. The function squashes capsules, which are essentially vectors of activations, to 0 if the output is a short vector and tries to constrain the output vector to 1 if the vector is long. The squashing function is defined in equation (1).

$$\upsilon_j = \frac{\| s_j \|^2}{1 + \| s_j \|^2} \frac{\| s_j \|}{\| s_j \|^2} \tag{1}$$

C. Dynamic Routing Algorithm

A routing algorithm is used to resolve which capsule gets activated for the incoming data. Dynamic routing is a very important networking technique that helps select a path according to the real-time layout changes. Dynamic routing algorithm is employed between the primary and class capsule layers and is used to achieve an agreement between the primary and class capsule layers. Dynamic routing helps to strengthen prediction value by using an agreement protocol. The lower level capsule sends its input to the higher level, which agrees with its input. Weight matrices are updated using this agreement between the two levels of capsules. The routing algorithm performs a similar function as the max pooling layer. However, while the max pooling layer chooses the most prominent features eliminating the non-prominent ones, the routing algorithm does not eliminate the features, but routes to the right feature instead.

D. Decoder

The decoder takes the output of the class capsules and reconstructs the object from it. Main purpose of having a decoder after final capsule layer is to enhance the encoding ability of the capsule network and better represent the object in the smaller domain. Therefore, a weighted reconstruction loss term is added to the overall loss that penalizes inaccurate reconstructions of an object so that network optimizes itself to better represent the objects in the final capsule layer.

$$L_R = \frac{1}{30 \times 30 \times 30} \sum_{i \in voxelgrid} (\mathbf{X}_i - \mathbf{R}_i)^2 \tag{2}$$

is the loss for single object where $\mathbf{X} \in G^3$ is the given object and $\mathbf{R} \in G^3$ is reconstruction where $G = \{0, 1\}$.

In [2], authors use two fully connected layers to reconstruct digits from the digit (class) capsules. Besides, by default, they used true class label in the reconstruction rather than the predicted one.

E. Loss Function

Sabour et al. [2] introduced margin loss to optimize the training of Capsule Networks. The margin loss for each capsule is defined as follows [2]:

$$L_k = T_k max(0, m^+ - ||v_k||)^2 + \lambda(1 - T_k) max(0, ||v_k|| - m^-)^2 \tag{3}$$

where k denotes the class capsule index, and T_k is 1 for true class. As in [2], we set $m^- = 0.1$, $m^+ = 0.9$ and $\lambda = 0.5$. Therefore, for C-many classes, overall loss function for batch size of N is defined as:

$$loss = \frac{1}{N}\sum_{i=0}^{N}\sum_{k=0}^{C} L_k^i + \gamma L_R^i \quad (4)$$

where γ is scale-down factor, set to 0.4, for reconstruction loss term.

Reconstruction loss is used as regularization to learn a global linear manifold between a whole object and the pose of the object as a matrix of weights via unsupervised learning. As such, the translation invariance is encapsulated in the matrix of weights, and not in the neural activity, making the neural network translation equivariant.

IV. Proposed Method

Our method is an extension of Capsule Networks [2] to 3D volumetric data, and will henceforth be referred to as 3D CapsNets.

A. 3D Capsule Networks

Our 3D CapsNet consists of several layers. Fig. 1 shows an example 3D CapsNet architecture. The Convolution layer extracts the basic features in the 3D data to create activities for these features. 3D convolutions involve filters of 3 dimensions (x, y and z), and thus are also known as spatial convolutions. This filter is moved in the three dimensions producing 3D output. In our implementation, a batch normalization [11] is used after convolutional layers. Primary capsule helps in implementing inverse graphics. We used Leaky ReLU activation in convolutional layers and in the primary capsule layer. Hierarchy of parts is a concept used, which explains that a higher level visual entity is present if several lower level visual entities can agree on their predictions for its pose. In the concept of capsules, lower level capsule helps the higher level predict if an entity is present. Primary capsules are thus bottommost level of the multi-dimensional objects. Class capsule has 16D output per object class. Dynamic routing algorithm is employed between the primary and class capsule layers to achieve an agreement. A non-linear squash function is used within the routing algorithm in order to change the length of vector to less than one, yet preserving the direction of the vector, thereby representing the weights as probabilities with direction.

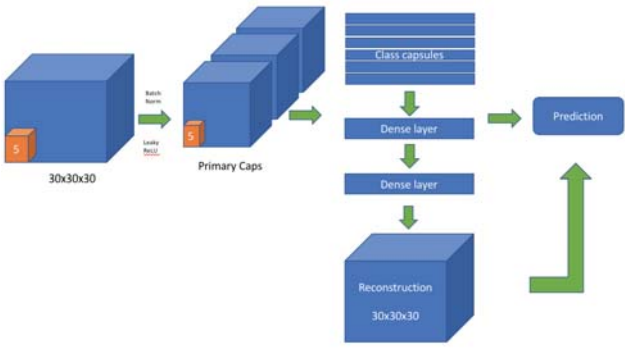


Fig. 1. 3D Capsule Network (Architecture-1)

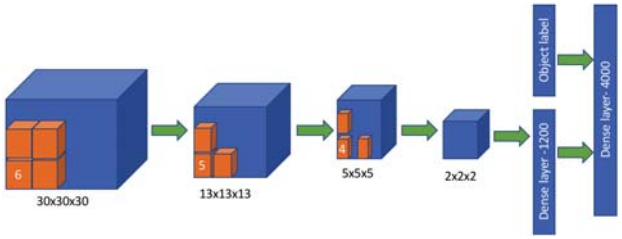


Fig. 2. Base Model architecture with two 3D convolutional layers, each followed by a max-pooling layer and ReLU activation, and 2 fully-connected layers.

A decoder mechanism is also added in our model in order to reconstruct the object to calculate reconstruction loss during training. There are three fully connected layers in our model's decoder. Fully connected layers use ReLU activation except the last one which uses Sigmoid activation.

V. Experimental Results

A. ModelNet Dataset and Benchmark

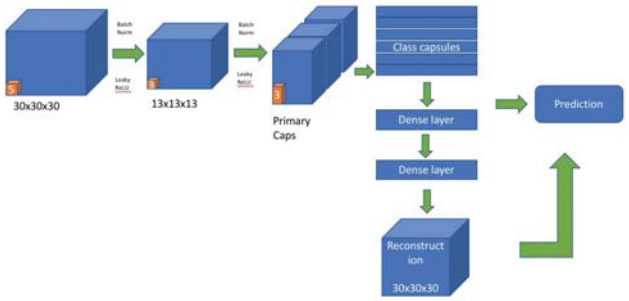
Princeton ModelNet project [1] provides a comprehensive collection of 3D CAD models of objects covering most common object categories. In this work, we have used the 40-class subset as well as the 10-class subset of the full dataset. The object classes in the 10-class subset are bathtub, bed, chair, desk, dresser, monitor, nightstand, sofa, table, and toilet. 40-class subset has additional 30 object categories. Number of samples in 10-class subset and 40-class subset are 4900 and 147,732, respectively.

B. Base Model

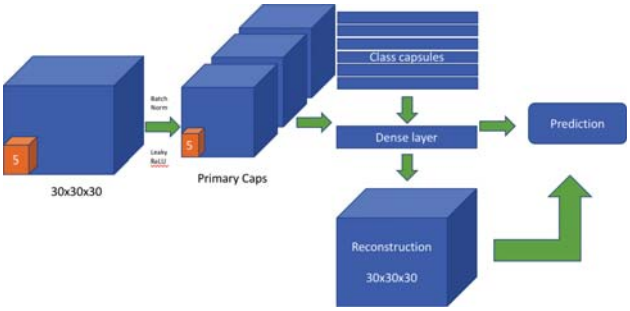
In our experiments, we used a base model architecture, similar to the 3D CNN architecture in ShapeNet [1], to compare with our model. We used this base model to show that the integration of capsule layers will boost the 3D object classification performance, even with smaller training data. The base model, shown in Fig 2, has two convolutional layers, max pooling layers, and two fully connected layers instead of capsule layers. It employs 3D maxpooling layers for dimension reduction and rotational/translational invariance.

C. Different 3D CapsNet Configurations

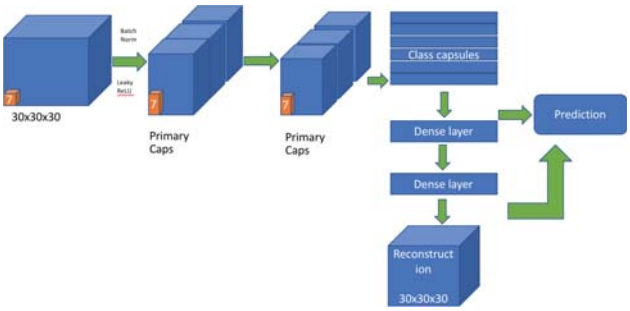
We have defined four unique architectures to measure the effects of different configurations of convolutional and capsule layers. Each configuration is given in Table I. 3D convolutional layers (Conv) have various kernel sizes with stride 1 in different architectures. Primary capsule layers (P.C.) have $7 \times 7 \times 7$ kernel with stride 1. First fully connected layers (FC) in decoder part of 3D CapsNets have length 512 and second fully connected layers, if any, have length 1024. Last FC layers have dimension of $30 \times 30 \times 30$ for the reconstructed object. Each 3D CapsNet architecture and base model is trained on



Architecture 2



Architecture 3



Architecture 4

Fig. 3. Different 3D CapsNet Architectures.

TABLE I
3D CAPSNET CONFIGURATIONS

Arch	L1	L2	L3	L4	L5	L6	L7
Base	Conv	MP	Conv	MP	FC	FC	
1	Conv	P.Caps	Caps	FC	FC	FC	
2	Conv	Conv	P.Caps	Caps	FC	FC	FC
3	Conv	P.Caps	Caps	FC	FC		
4	Conv	P.Caps	P.Caps	Caps	FC	FC	

the same training set which contains 40% of the ModelNet-10 dataset. Accuracy results are given in Table II.

D. Discussion of Experiments

We have made an extensive set of experiments with four different 3D CapsNet architectures with different number of layers and configurations. These architectures are summarized in I and illustared in Fig. 3. We trained our models on both ModelNet10 and ModelNet40 with varying training/test size ratios. In our analysis, we observed that the best overall performance is achieved by Architecture-3, which has the most shallow structure. It is because capsule layers are very successful at capturing better representations.

TABLE II
PERFORMANCE COMPARISON WITH 40% TRAINING DATA

	ModelNet-10	ModelNet-40
Base Model	88.30%	85.60%
Arch 1	90.75%	88.70%
Arch 2	90.63%	87.28%
Arch 3	91.37%	89.66%
Arch 4	91.30%	87.38%

VI. CONCLUSION

In this paper, we have proposed 3D Capsule Network solutions to perform object classification from 3D data, which is present in the form of 3D binary occupancy grids. We have proposed shallow architectures that can recognize objects better than shallow architectures based on pure convolutional neural networks. We have used 3D convolutional layers to extract the features and proposed capsule architectures to capture the spatial relationships in the 3D data better. It is claimed that Capsule Networks require less training data. We have been able to prove this to be true for 3D data as well in our experiments. We have compared our approach with ShapeNet on the ModelNet dataset, and showed that our method provides performance improvement especially when training data size gets smaller. With only 40% training data, we have been able to achieve 91.37% accuracy for ModelNet10 and 89.66% for ModelNet40.

REFERENCES

[1] 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*, 2015, pp. 1912–1920.

[2] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[3] Y. LeCun, "The mnist database of handwritten digits," *http://yann.lecun.com/exdb/mnist/*, 1998.

[4] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[5] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–104.

[6] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.

[7] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

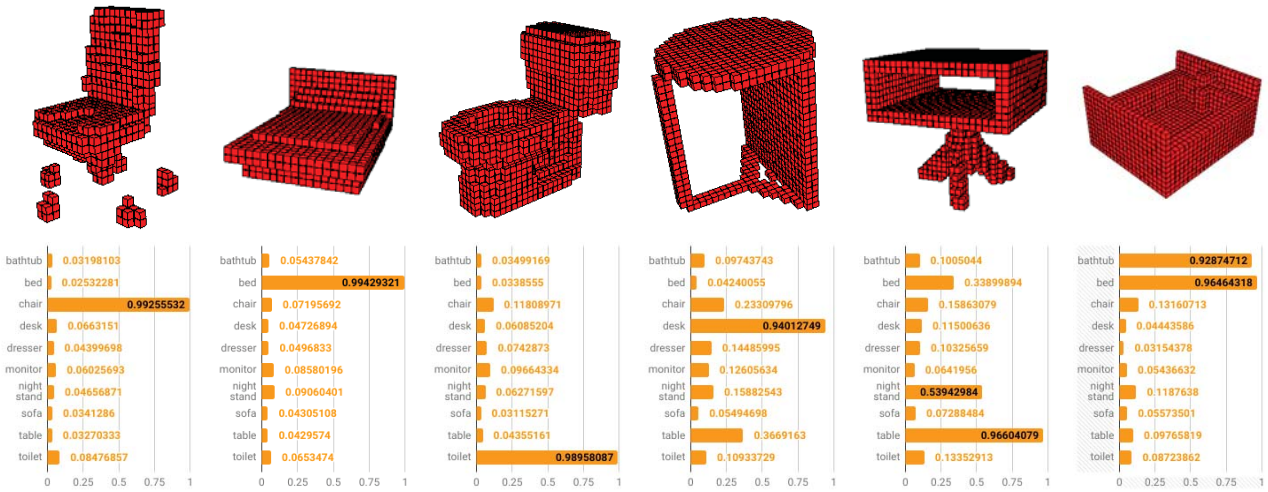


Fig. 4. Examples of three correct prediction (left) and three false prediction (right)

[8] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[9] K. Sfikas, T. Theoharis, and I. Pratikakis, "Exploiting the PANORAMA Representation for Convolutional Neural Network Classification and Retrieval," in *Eurographics Workshop on 3D Object Retrieval*, I. Pratikakis, F. Dupont, and M. Ovsjanikov, Eds. The Eurographics Association, 2017.

[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.