IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 15, 2020, accepted January 29, 2020, date of publication February 5, 2020, date of current version February 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971950

3D Capsule Networks for Object Classification With Weight Pruning

BURAK KAKILLIOGLU[1], (Student Member, IEEE), AO REN[2], (Student Member, IEEE), YANZHI WANG[2], (Member, IEEE), AND SENEM VELIPASALAR[1], (Senior Member, IEEE)

[1]Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244, USA
[2]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA

Corresponding author: Burak Kakillioglu (bkakilli@syr.edu)

The information, data, or work presented herein was funded in part by National Science Foundation (NSF) under Grant 1739748, Grant 1816732 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.

ABSTRACT The proliferation of 3D sensors, due to the increased demand for 3D data, induced the 3D computer vision research in the last decade, and 3D data processing has gained a lot of interest. As in many other applications in computer vision, deep learning-based methods were quickly applied to 3D data classification and have become the state-of-the-art in this area. More recently, *capsule networks*, which are novel neural structures, have been introduced to enhance the ability of neural networks to better capture the parts-relationship, which yields more accurate classification with less training data. Moreover, deploying deep machine learning models on mobile platforms requires the models to be optimized due to limited memory and computational constraints. In this work, we propose methods to boost the accuracies of a standard 3D CNN-based and a Capsule Network-based classifier, help the training to better generalize the data distribution with limited data, and optimize the models for resource-constrained environments, such as mobile platforms. We also show that the introduction of capsules to 3D object classification pipeline improves the classification performance with limited training data, while a specifically optimized weight pruning method keeps the model compact enough for mobile deployment. Our broad spectrum of experiments show that proposed methods improve the performance of the base model while significantly reducing the memory and computation requirements.

INDEX TERMS 3D, admm, capsule networks, classification, modelnet, network optimization, shapenet, pruning.

I. INTRODUCTION

Deep learning-based methods have become the state-of-the-art in many fields, including machine learning, computer vision and natural language processing. Approaches based on deep learning, such as [1] and [2], have become the de-facto standard for object detection applications. However, they require significant amount of annotated training data. Moreover, these models are computationally expensive, and have large network structures with significant number of parameters.

The proliferation of 3D sensors induced the 3D computer vision research in the last decade, and deep learning for 3D data understanding has gained a lot of interest. As mentioned earlier, machine learning models are data driven and require lots of positive and negative examples to be trained. On the other hand, the amount of 3D data is significantly lower compared to the available 2D image datasets, and labeling/annotating 3D data is harder. Moreover, the appearance and representation of 3D objects change for different viewpoints, which makes the learning in 3D even harder. Due to these constraints and challenges, models that can be trained with less data while maintaining the classification accuracy are more desirable. There have been various generic strategies proposed in the literature for machine learning with less data, such as data augmentation, transfer learning, and generative models [3]–[8].

3D objects are usually rigid bodies with an underlying simple or complex shape definition. Although some perturbations, such as rotation, scaling and shifting, may reflect on the external attributes, such as position in the Euclidean space, orientation, and volume, the internal characterization of the

The associate editor coordinating the review of this manuscript and approving it for publication was Orazio Gambino.

shape remains same. Therefore, an analysis scheme must be invariant to these external perturbations while carefully capturing the internal geometric properties of the object, and account for the validity of the geometric positioning and size of these parts with respect to each other. This is also an important factor for 2D image classification. However, the third dimension and much lower data resolution could potentially reduce the chance of recovery further, compared to 2D counterpart.

More recently, Capsule Networks were proposed with a novel activation routing mechanism [9]. Capsule network is claimed to be robust to changes of the parts in various instantiation parameters, such as pose, deformation, velocity, texture etc. A capsule is a vector of neurons, which encapsulates multiple activations in one group. A capsule layer consists of many capsules. A special agreement mechanism, between subsequent capsule layers, is used for training these layers. The encapsulation of activations and agreement mechanism encourage each capsule to be responsible for capturing how an entity is represented in the dataset, instead of only indicating its existence as in traditional neural activations. This property greatly increases the explainability and the accountability of the model during and after it is trained.

In our preliminary work [10], [11], we introduced a 3D Capsule Network, referred to as 3D CapsNet, to perform object classification from 3D volumetric data. Despite its ability to extract features from 3D objects, the large model size of the 3D CapsNet hurdles its deployment on resource-constrained platforms, such as mobile phones, IoT devices, and unmanned vehicles etc.

Model compression of neural networks has been proven to be an effective method to reduce the network model size so that the state-of-the-art deep neural networks can be implemented on FPGAs and ASICs [12]–[14]. In this work, we propose an optimized neural network model for 3D object classification by using a weight pruning approach to significantly compress our proposed 3D CapsNet, so that it can be implemented on mobile platforms or edge devices. Our pruning approach is based on the Alternating Direction Method of Multipliers (ADMM), which achieves state-of-the-art pruning ratio. The approaches proposed in this paper can also make an existing model more accurate, especially with limited training data, and significantly reduce the number of parameters of a model.

In this paper, one goal is to show that a network's classification accuracy can be improved, especially with limited training data, while significantly reducing the network size at the same time. We also show that these strategies can be applied to other architectures without loss of generality. Therefore, in addition to comparing with other existing work, we also test our methods on a base model for a more commensurate comparison, and provide thorough experimental analysis on different datasets and with different data encodings.

The contributions of this paper include the following:

- A 3D object classification method, referred to as 3D CapsNet, which captures part-relationships better and requires less data for training.
- An ADMM-based weight pruning method for significantly reducing the number of parameters of our proposed 3D CapsNet without accuracy loss.
- A comprehensive set of experiments including comparison of accuracy, number of weights and compression ratio on different datasets.
- A detailed analysis showing the performance on decreasing amounts of training data.
- An optimization to the dynamic routing mechanism for faster computation while maintaining the classification accuracy.

The work proposed in this paper is different from, and improved compared to our previous work [10], [11] in multiple ways including (i) the development of a weight pruning approach based on ADMM and applying this approach to the proposed 3D CapsNet to significantly reduce the number of weights and the memory requirements of the model for embedded deployment, and increase the accuracy at the same time; (ii) introducing an optimization to the dynamic routing mechanism for faster computation while maintaining the classification accuracy; (iii) performing a comprehensive set of experiments comparing accuracy, number of weights and compression ratio on different datasets and showing the significant decrease in the number of weights together with the effect of weight pruning on the classification accuracy; (iv) comparison with additional existing work and a base model; (v) providing a detailed analysis by using different data splits showing the performance on decreasing amounts of training data.

The rest of this paper is organized as follows: The related work is discussed in Section II. The proposed 3D CapsNet and its optimization with the weight pruning algorithm are described in Sections III and IV, respectively. Experimental results are presented together with a discussion in Section V, and the paper is concluded in Section VI.

II. RELATED WORK

A. 3D OBJECT CLASSIFICATION

There have been various works for 3D object classification [15]–[26]. In contrast to 2D case [27], [28], 3D data has different data representations, and requires careful model design for each type of data representation. Below, different studies will be discussed separately based on the 3D representation that they use.

1) METHODS BASED ON DEPTH MAPS

These methods are often called 2.5D approaches, as they actually do not process in 3D, despite the fact that data contains spatial information. The data is a 2D depth map and is processed either as a single channel image or as the

fourth channel together with RGB color channels. [15] is one of the early studies, which uses convolutional features of depth maps for transfer learning. [16] fuses convolutional features, which are extracted from depth maps and color images separately in different branches, for hand-held object recognition.

2) MULTI-VIEW AND PANORAMIC METHODS

Not all the spatial object classification models utilize a 3D sensor. [17] proposes a multi-view Convolutional Neural Network (CNN) model for classification of 3D objects by fusing 2D rendered views of the objects from various different viewpoints. Similarly, [18] proposes a CNN model that utilizes 2D renderings of the objects together with viewpoint information in both guided and unguided settings. [19] proposes a novel 2D panoramic representation for 3D object classification.

3) 3D VOXEL GRID-BASED METHODS

Initial works, which process the 3D data actually in 3D domain by encoding the information as voxel-grids, use 3D convolutional kernels for feature learning and extraction. An earlier work in this category is 3DShapeNets [20], which proposes a 3D CNN-based model for object classification using volumetric grids with next-best-view approach, which benefits from multiple views. The 3D information is encoded into $30 \times 30 \times 30$ volumetric grid as binary occupancies. The work in [20] was introduced together with the ModelNet dataset and benchmark, which enabled further improvement by others. Another 3D convolution-based method, which uses 3D voxel grids, is proposed in [21]. It compares different voxel grid representations, such as binary and probabilistic occupancy grids, as well as data augmentation techniques for 3D object recognition in spatial domain. [22] uses $30 \times 30 \times 30$ binary voxel grids from several different viewpoints and proposes two neural architectures to classify them. The first network performs auxiliary training by subvolume supervision, and the second network performs anisotropic probing by an elongated kernel to capture the global structure of the 3D volume.

4) POINT CLOUD-BASED METHODS

In recent literature, voxel grid-based representation was very popular because of its simplicity and well-organized nature, which fits well with the neural architectures. However, processing 3D voxel grids in a neural network setting is a demanding task. [23] and [24] criticized voxel grid-based methods for having $O(n^3)$ time and space complexity, and for requiring 3D convolution operation. [23] and [24] also discuss the advantages of processing 3D point clouds in a neural network directly, which is a more natural and detailed 3D representation, which contains more information while usually requiring much less memory. However, the problem with processing point clouds is that they are unordered set of points. The same point cloud data (set) with N points can be represented in $N!$ different ways. Thus, this requires a model to be invariant to permutation of point ordering. [23] proposes the PointNet architecture, wherein the input to the neural network is a 3D Point cloud ($P_{N\times 3}$) with N points. It first applies a transformation network *T-Net* to get the canonical rotation of the input, and applies a Multi-Layer Perceptron (MLP) network to each point individually. Then it applies a commutative function (max-pooling) over the point dimension to get a point order invariant latent space representation. This representation is then fed into another MLP classifier to get the softmax probabilities. [24] proposes an order equivariant layer for feature extraction from individual points while preserving their order. Then similar to [23], it uses a commutative function (averaging) to get the latent space representation of the object from the extracted features by equivariant layers. [25] is the proceeding study of [23]. It uses the same PointNet structure internally, but it applies PointNet to several hierarchical parts of the object separately and extracts local features from the object. More recently, another deep learning model for 3D point clouds is proposed for addressing contextual information between local regions implicitly via attention mechanism with LSTM (Long-Short Term Memory) units [26].

B. NETWORK OPTIMIZATION

In order to achieve high accuracy, the state-of-the-art approaches tend to build deeper and wider networks containing millions of weights. However, the significantly large number of weights incurs massive computation and storage burden, hindering the deployment of the state-of-the-art deep learning methods on resource-constrained platforms, such as mobile phones and embedded devices. It has been extensively studied and shown that there exists inherent redundancy in these weights, and there have been increasing research efforts on removing this redundancy, which is known as weight pruning [13], [14], [29], [30].

A heuristic pruning method is used in [13] by directly removing the weights with small magnitudes and retraining the network. This work achieves $9\times$ weight reduction on AlexNet [27] for ImageNet dataset, but it only achieves $2.7\times$ reduction on the convolutional layers of AlexNet, which accounts for the main computations and number of weights in the state-of-the-art deep neural networks (DNNs), such as GoogleNet [31] and ResNet-50 [28], etc. [28], [31]. Low-rank matrix factorization [14], [29] is another way of pruning by decomposing the original weight matrix into the linear composition of a set of low-rank weight matrices. Even though these methods can achieve good compression ratio by constraining the rank to a small number, they also incur significant ($>3\%$) accuracy loss. NeST [30] achieves weight reduction by learning a compact architecture, and provides $15.7\times$ and $30.2\times$ overall reduction on AlexNet and VGG, respectively, but only $2\times$ to $5\times$ reduction are obtained on the convolutional layers of AlexNet and VGG [32].

III. 3D CAPSULE NETWORKS

As mentioned in [9], CNNs do not necessarily impose any restrictions on relative positioning of parts with respect to

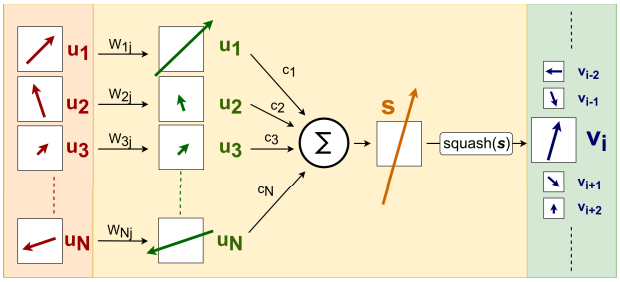


FIGURE 1. Routing of information from the first capsule layer to the capsule v_i in the second capsule layer.

each other, and need large amounts of training data to generalize the classifier. All low-level details are sent to all of the higher level neurons, which then perform further convolutions to check whether certain features are present. This is done by striding the receptive field and then replicating the knowledge across all different neurons. Capsule networks (CapsNet) were proposed [9] to address these problems.

In a CapsNet, neurons (which output scalar values) are replaced with capsules (which output vectors). A capsule contains several neurons, and together they represent instantiation parameters of a feature. Like neurons, capsules are also stacked into layers. However, unlike convolutional layers, capsule layers will activate only certain capsules that best represent the incoming image (or just a feature). A CapsNet addresses the aforementioned issues of CNNs by treating different orientations of an image as the same object. Hence fewer training examples suffice during the training of CapsNets. The CapsNet solves the issue of translational invariance by preserving the geometric dependence of features.

Capsule Networks were originally proposed for 2D data [9]. The model used in this work, namely 3D Capsule Network, is an extension of the 2D Capsule Networks to 3D data, and will be referred to as 3D CapsNet. To account for the complexity of 3D data, in comparison to MNIST dataset that was used in [9], 3D CapsNet uses additional layers to improve the model. In our experiments, we use the 3D CapsNet architecture shown in Fig. 2, and improve the performance of our previous work [10] by incorporating ADMM-based weight pruning optimization and routing mechanism optimization, which are detailed in Sections IV and V-F, respectively. We also perform additional comparative experiments with the state-of-the art models and a base model on multiple datasets with different training/testing splits to show the significant decrease in the number of weights together with the effect of weight pruning on the classification accuracy.

A. CAPSULE LAYERS

Although the idea of capsules was first introduced by Hinton in [33], their first use in a deep, specifically convolutional, neural network is in [9], thanks to the proposed *dynamic routing between capsules* algorithm. A capsule is basically a small group of neurons, each of which is responsible for various properties of a particular entity. Each capsule is represented as a vector in the layer and a layer is composed of multiple capsules. A special function, which is called squashing function (see Eqn. (1)), ensures that the length of each vector is squashed into [0, 1) range. Therefore, squashing function also works as a regularizer. The length of the capsule vector indicates how well the entity, of which the capsule is responsible for, is represented in the data. If a particular entity does not exist in the data, the length of the capsule, which is responsible for that entity, is expected to be close to zero. The squashing function is defined as following, where s_j is the vector that is accumulated by the contributions from the previous capsule layer and v_j is the final capsule vector, which is the squashed version of s_j:

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

B. DYNAMIC ROUTING

The reason that capsules have not appeared in deep neural networks until recently is that there was not a practical and effective way to train them in a neural network. The first study to train a capsule network introduced the dynamic routing mechanism, which routes the information from previous capsule layer to the next one by an agreement [9]. If the agreement between the capsules at lower and higher level is high, then the input at the lower capsule is sent to the higher one due to a greater coupling coefficient between these capsules. This provides a dynamic mechanism for lower level features to contribute to a feature at the higher level if they are related. Internally, a prediction $\hat{\boldsymbol{u}}_{j|i}$ of the next layer is made by a dot product of the capsules at current layer ($\boldsymbol{u}_{j|i}$) with a weight matrix $\boldsymbol{W}_{ij}$ as given in Eqn. (2). This initial guess is similar to dense network between each capsules. The coupling coefficients c_{ij}, which are log probabilities, are updated iteratively by an agreement between consecutive capsule layers. At the final iteration, these coefficients are used as weights to either suppress or encourage the contribution of lower level capsules to certain higher ones. The weighted sum of the predictions ($\hat{\boldsymbol{u}}_{j|i}$) are calculated with c_{ij}'s and then squashed as defined in Eqn. (1) to produce the capsules at the next layer (v_j). Fig. 1 illustrates routing operation for capsule v_i.

$$s_j = \sum_i c_{ij} \hat{\boldsymbol{u}}_{j|i}, \quad \hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij} \boldsymbol{u}_{j|i} \tag{2}$$

C. 3D OBJECT CLASSIFICATION WITH CAPSULE NETWORKS

A capsule network can be defined as a convolutional deep neural network, which contains at least two capsule layers, where a specialized routing happens in between. A capsule layer can be used as a stand alone classifier as in [9]. Otherwise, it can be used as a regular feature extractor and its output is routed to another feature layer or to a classifier.

In our model, we use a series of 3D convolutional layers as the initial feature extractor, which takes 3D voxel grids

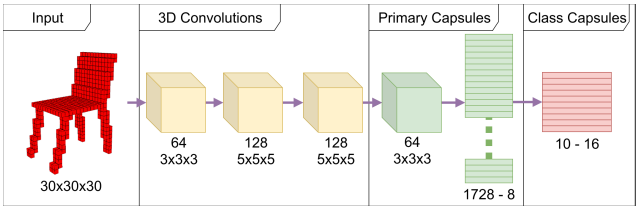


FIGURE 2. 3D Capsule Network architecture for ModelNet-10 classification. Three convolutional layers with 3D kernels extract low level features from input data. Primary capsule applies one more layer convolutional feature extraction and re-organizes the activations as capsules as 8 dimensional vectors. The iterative dynamic routing happens between primary and class capsules, where the features are classified into 10 categories.

as inputs. These initial layers are responsible for extracting basic features, such as curvatures, corners, edges etc., as well as more abstract features, such as table tops, chair legs etc. A pair of capsule layers follows the convolutional layers for better investigation of the extracted features' interrelations and classification. It should be noted that the minimum number of capsule layers in a capsule network is two given the fact that learning on capsules is done by the dynamic routing by agreement algorithm between consecutive capsule layers during back propagation. The structure of the proposed 3D CapsNet model is shown in Fig. 2. It should also be noted that it does not contain pooling layers for rotation and translation invariance. Instead, it accommodates the variance of conceptually and visually similar, but different objects.

IV. NETWORK OPTIMIZATION

As mentioned in Section II, it has been shown in prior work that there is significant redundancy in CNNs, and pruning, which reduces the number of weights in neural networks, has been extensively studied [12], [13], [34].

Our proposed 3D CapsNet is intended for 3D object classification tasks. Compared with the CNNs designed for 2D datasets, the added third dimension introduces more redundancy in the network. Moreover, we add extra convolutional layers with large kernels (e.g., $5 \times 5 \times 5$) to extract the pose features of the 3D objects. Although these added layers significantly improve the accuracy of 3D CapsNets, they also add to the redundancy. ADMM-based weight pruning has been proven to be an effective pruning method, and achieves the state-of-the-art pruning ratio [12]. Therefore, we integrate the ADMM-based pruning into our 3D CapsNet to significantly reduce the redundancy while maintaining our 3D CapsNet's accuracy, so that it would be possible to deploy our 3D CapsNet on resource-constrained platforms.

A. ADMM METHOD

ADMM has been demonstrated to be a powerful optimization framework for solving a non-convex optimization problem that is difficult to solve directly [35], [36]. Through decomposing the original problem into subproblems that can be solved relatively easily, and through iteratively solving the subproblems, the original problem can converge to a satisfactory solution. For example, the optimization problem

$$\min_x \; f(x) + g(x) \tag{3}$$

can be solved with ADMM if two conditions are satisfied: i) $f(x)$ is differentiable, and ii) $g(x)$ has some special structure that can be exploited. By introducing an auxiliary variable z, the problem can be re-written as

$$\begin{aligned} &\min_{x,z} \; f(x) + g(z), \\ &\text{subject to } x = z. \end{aligned} \tag{4}$$

Next, the above problem can be decomposed into two subproblems on x and z, using augmented Lagrangian [37]. The first is $\min_x f(x) + q_1(x)$, where $q_1(x)$ is a quadratic function. In the case of training neural networks, subproblem 1 can be solved via back-propagation. Subproblem 2 is $\min_z g(z) + q_2(z)$, where $q_2(z)$ is a quadratic function. Since function g is required to have some special structure, it can be solved analytically. The problem (4) is solved via iteratively solving the two subproblems.

B. DESIGN DETAILS OF ADMM PRUNING ON 3D CAPSULES

In order to apply ADMM-based weight pruning to the proposed 3D CapsNets, we need to formulate the problem as shown in (4). First of all, we will add constraints to each layer of the neural network as $S_l = \{\textit{the number of nonzero wegiths at } \; l_{th}\textit{layer is less than } \alpha_l\}$, $l = 1, \ldots, N$, and integrate the constraints into the loss function. Then the pruning problem becomes

$$\begin{aligned} &\underset{\{W_l\},\{b_l\}}{\text{minimize}} \quad f\left(\{W_l\}_{l=1}^N, \{b_l\}_{l=1}^N\right), \\ &\text{subject to} \quad W_l \in S_l, \;\; l = 1, \ldots, N. \end{aligned} \tag{5}$$

where W_l and b_l are weights and biases in the l_{th} layer, respectively. By introducing an auxiliary variable $Z_l = W_l$ and an indicator function

$$g(W_l) = \begin{cases} 0 & \text{if } W_l \in S_l, \\ +\infty & \text{otherwise,} \end{cases}$$

the problem (5) can be rewritten as

$$\begin{aligned} &\underset{\{W_l\},\{b_l\}}{\text{minimize}} \quad f\left(\{W_l\}_{l=1}^N, \{b_l\}_{l=1}^N\right) + \sum_{l=1}^N g(Z_l), \\ &\text{subject to} \quad W_l = Z_l, \;\; l = 1, \ldots, N. \end{aligned} \tag{6}$$

Next, ADMM plays a role in solving problem (6). It decomposes the problem into two subproblems, using augmented Lagrangian. The first subproblem is written as

$$\underset{\{W_l\},\{b_l\}}{\text{minimize}} \; f\left(\{W_l\}_{l=1}^N, \{b_l\}_{l=1}^N\right) + \sum_{l=1}^N \frac{\rho_l}{2} \| W_l - Z_l^k + U_l^k \|_F^2, \tag{7}$$

where $U_l^k := U_l^{k-1} + W_l^k - Z_l^k$ is the dual variable updated in each iteration, and ρ_l is referred to as the penalty term. This

subproblem can be seen as a loss function with a dynamically updated weight decay regularization, where the regularization target Z_l^k is solved in the second subproblem. Therefore, the first subproblem can be solved via back-propagation.

On the other hand, the second subproblem is written as

$$\underset{\{Z_l\}}{\text{minimize}} \quad \sum_{i=l}^{N} g(Z_l) + \sum_{l=1}^{N} \frac{\rho_l}{2} \|W_l^{k+1} - Z_l + U_l^k\|_F^2. \quad (8)$$

Since $g(\cdot)$ is an indicator function, it has an analytical solution that can be written as

$$Z_l^{k+1} = \Pi_{S_l}(W_l^{k+1} + U_l^k), \quad (9)$$

where $\Pi_{S_l}(\cdot)$ represents the Euclidean projection of $W_l^{k+1} + U_l^k$ onto the set S_l. The details of the solution to this subproblem is problem-specific. For our weight pruning problem, the solution of Z_l^{k+1} is obtained by keeping the first α_l weights that have the largest magnitude. In other words, the weights at the l_{th} layer are sorted according to their magnitudes $|w_{ij}|$. Then the top α_l weights are kept and the rest are eliminated.

In each iteration, subproblem 1 and subproblem 2 are solved, and the solutions W_l^{k+1} and Z_l^{k+1} are used to update the dual variable U_l^{k+1}. After one to several tens of iterations, the problem (5) will converge and a satisfactory solution can be obtained. It should be noted that the solution is not guaranteed to be optimal because of two reasons: first, ADMM is used to solve a problem that itself is hard to solve directly, and thus ADMM can guarantee a satisfactory solution rather than the optimum; second, the solution to the loss function $f\left(\{W_l\}_{l=1}^N, \{b_l\}_{l=1}^N\right)$ cannot be guaranteed to be optimal. However, we can still claim that the solution is good enough to produce high performance for a neural network.

Since 3D CapsNet is responsible for classifying 3D objects that impart 4D activations, and 5D kernels as a result, it is more difficult for 3D CapsNet to converge when the ADMM-based weight pruning is applied. Therefore, in order to make the ADMM-based weight pruning over 3D CapsNets more stable, we adopt a dynamic penalty term adjusting strategy [37]. That is, we evaluate the primal residual $r^k = ||W_l^{k+1} - W_l^k||_2^2$ and dual residual $s^k = ||Z_l^{k+1} - Z_l^k||_2^2$ in each iteration. When the primal residual is greater than the dual residual by u, that is $\frac{r^k}{s^k} > u$, the penalty ρ_l is increased by t. On the other hand, if the dual residual is greater than the primal residual by u, that is $\frac{s^k}{r^k} > u$, the penalty ρ_l is divided by t. The typical values for u and t are 10 and 2, respectively.

After solving the ADMM-based pruning, we still need to run a masked re-training, because the pruned 3D CapsNet still contains many nonzero weights with small magnitudes that are close to zero. These weights need to be set to zeros and masked during the re-training stage, that is, those zero weights will not be updated.

V. EXPERIMENTAL RESULTS

We have conducted an extensive set of experiments with the proposed 3D CapsNets and network optimization on various datasets with different data splits (by using decreasing amounts of data for training). ModelNet dataset is one of the synthetic datasets, which contains various categories of 3D CAD objects from public online sources with ground truth [20]. ModelNet has two benchmarks, namely ModelNet-10 and ModelNet-40, which are 10-class and 40-class subsets of most common objects, respectively. ShapeNetCore55 is another synthetic dataset, similar to ModelNet, which contains CAD models of 55 most common objects [38]. In our experiments, in order to show the performance of the proposed 3D CapsNets on decreasing amounts of training data, we used varying portions of training sets while keeping the validation and testing sets fixed.

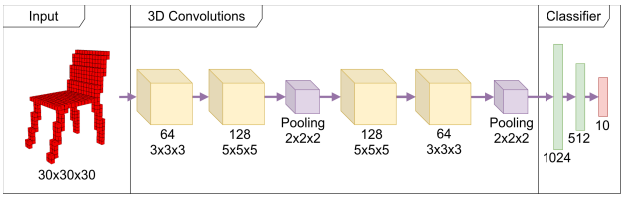


FIGURE 3. Base model architecture, which is a standard 3D CNN, is designed and used for commensurate comparison with the proposed 3D CapsNets. The role and significance of the base model is that it is composed of basic building blocks of a traditional CNN in order to show that the evident improvement is due to capsules, rather than complex and domain-dependent modifications.

TABLE 1. Performance comparison on standard datasets without weight pruning*.

Training Set	Model Type	Accuracy %
ModelNet-10 Default Split (78%‡)	3DShapeNets [20]	83.50
	VoxNet [21]	92.00
	Point2Sequence [26]	94.05
	Base Model	93.04
	3DCapsNet	**94.51**
ModelNet-40 Default Split (78%‡)	3DShapeNets [20]	77.30
	VoxNet [21]	85.90
	Point2Sequence [26]	**91.98**
	Base Model	86.13
	3DCapsNet	89.74
ShapeNetCore55 20%	Base Model	63.18
	3DCapsNet	**75.97**

*‡See the footnote of Table 2.

A. MODEL ARCHITECTURE AND EXPERIMENTAL SETTINGS

Our proposed 3D CapsNet architecture is shown in Fig. 2. It contains four 3D convolution layers followed by two capsule layers. The first capsule layer consists of 1728 8-dimensional vectors as primary capsules. The second capsule layer, namely class capsules, consists of C-many 16 dimensional vectors as class capsules, where C is the number of categories in the training set. Thus, the class capsule layer changes across different datasets. Since the class capsule is already being used for classification, there are no dense layers in our 3D CapsNet architecture. We defined and trained a base model, which is similar to the VoxNet [21] in its architecture, but with wider and deeper layers, i.e., more

TABLE 2. Performance comparison with smaller training sets and weight pruning*.

Training Set	Model Type	Accuracy %		# of Weights (M).		Compression Ratio
		Before †	After †	Before †	After †	
ModelNet-10 30%‡	VoxNet [21]	86.12	n/a	**0.92**	n/a	n/a
	Point2Sequence [26]	90.55	n/a	5.54	n/a	n/a
	Base Model	91.77	93.20	47.61	9.47	5.03×
	3DCapsNet	**93.87**	**93.44**	35.15	**1.05**	33.48×
ModelNet-10 60%‡	VoxNet [21]	90.11	n/a	**0.92**	n/a	n/a
	Point2Sequence [26]	92.04	n/a	5.54	n/a	n/a
	Base Model	91.48	92.25	47.61	10.49	4.54×
	3DCapsNet	**94.16**	**94.68**	35.15	**2.30**	15.28×
ModelNet-40 30%‡	VoxNet [21]	79.53	n/a	**0.92**	n/a	n/a
	Point2Sequence [26]	81.45	n/a	5.54	n/a	n/a
	Base Model	79.26	82.79	37.37	4.29	8.71×
	3DCapsNet	**82.71**	**83.62**	5.51	**0.72**	7.65×
ModelNet-40 60%‡	VoxNet [21]	84.01	n/a	**0.92**	n/a	n/a
	Point2Sequence [26]	**86.34**	n/a	5.54	n/a	n/a
	Base Model	83.65	85.95	37.37	5.99	6.24×
	3DCapsNet	85.61	**87.31**	5.51	**0.83**	6.64×

*Original published codes of the compared models are used to retrain and test their models with mentioned datasets. Same data splitting is used in all experiments in each horizontal block. †Before and After refers to before weight pruning and after weight pruning, respectively. ‡Percentages mean the ratio of the size of the training set to the size of the dataset.

parameters, in order to make a fair and commensurate comparison with our 3D CapsNet architecture. The base model, shown in Fig. 3, contains four convolutional layers, as in the 3D CapsNets, which are followed by three dense layers with dropout for classification. The role and significance of the base model is that it is composed of basic building blocks of a traditional CNN in order to show that the evident improvement is due to capsules, rather than complex and domain-dependent modifications. All of the trainings is performed in a supervised manner, i.e., with ground truth labels, via backpropagation using Adam optimization [39] with minibatches of 32. We used 0.0005 as the learning rate during the pretraining of 3D CapsNet as well as during the two weight optimization stages defined in Section IV. Our experiments are performed on NVIDIA Titan X (Pascal) GPU with Tensorflow implementations of the presented models.

As elaborated in Section I, traditional deep learning methods require large amounts of training data to perform well. One of our goals in this paper was to show that the proposed approach can perform better than other methods, especially when limited amount of training data is available. The performance improvement provided by our approach becomes more apparent when training data size gets smaller, since other traditional approaches need large amounts of training data. This is the reason that we performed experiments by using decreasing amounts of training data to compare different methods. For instance, the default (provided by the authors) training set size is 78% of the entire data in the ModelNet-40 dataset. We randomly subsampled only the training data in a stratified manner to generate different training data corresponding to 30% and 60% of the entire dataset. It should be noted that the test set remains same and does not overlap with any of the training set. All accuracies reported for a dataset are evaluated on the same test set. We presented the results in Table 2, while the Table 1 shows the results with the original training set sizes. This is discussed in more detail in Section V-B.

B. CLASSIFICATION RESULTS

We evaluated the classification accuracy of 3D CapsNets on both ModelNet-10 and ModelNet-40 [20] subsets as well as on the ShapeNetCore55 [38] dataset. We consistently show across these datasets that when the given training data progressively gets smaller, the performance margin between our model and other approaches increase. This signifies the better generalization capability of our model with less data. The results are presented in Tables 1 and 2, where we compare our results with the state-of-the-art and other approaches. In fact, there are many results are reported on ModelNet benchmark and the performance of the latest models are similar to each other. Therefore, to compare with our model we chose the Point2Sequence model [26], which is one of the recent works among the state-of-the-art, and the VoxNet model [21], which is one of the pioneer works and has a simple traditional CNN architecture. Although the VoxNet is an earlier approach with lower accuracy, it is a perfect candidate for highlighting the improvement by the capsule layers in our work, similar to the purpose of the base model mentioned earlier.

Table 1 summarizes the results obtained with different methods without weight pruning so that the performance of the proposed 3D CapsNet (before pruning) can be compared with others. For Table 1, the results on the ModelNet-10 and ModelNet-40 sets have been obtained by using the default training-testing split of the ModelNet dataset, in which the training data is the 78% of all data. As can be seen, with the ModelNet-10 dataset, the proposed 3D CapsNet provides the highest accuracy. However, overall, its advantages and strength are not as pronounced when using a larger portion of the training dataset, i.e. when more data is available, since the proposed approach outperforms others, especially when

limited amount of training data is available. More discussion on this is provided below by referring to Table 2.

Table 2 shows the classification performance of and the number of parameters for the compared models before and after ADMM-based weight pruning is applied, for the ModelNet-10 and ModelNet-40 datasets. Moreover, Table 2 presents these results for decreasing amounts of training data, which shows the strength of the proposed 3D CapsNet when smaller amounts of training data is used. It can be observed from Tables 1 and 2 that when the training set is larger our model's performance is on par with [26], which is among the-state-of-the-art. However, as the size of the training data decreases, as shown in Table 2, our model performs better than all the other methods, and our model is more robust to decreasing the size of the training set than any other method. This indicates a better generalization of the objects' geometric properties by agreement of their parts. More specifically, for ModelNet-10 dataset, when 60% of the dataset is used for training, the proposed method provides 4.05% and 2.12% increase in accuracy compared to VoxNet and Point2Sequence, respectively. When the training data size is decreased to only 30% of the dataset, the performance improvement provided by our method increases, and it provides 7.75% and 3.32% increase in accuracy compared to VoxNet and Point2Sequence, respectively. It should be noted that when 30% of the data is used for training, there are only 1052 and 3693 training samples for ModelNet-10 and ModelNet-40 datasets, respectively. When training dataset size is to 60% of all data, there are only 2104 and 7386 training samples for ModelNet-10 and ModelNet-40 datasets, respectively. In order to show that these results are not due to a specific split, we have also performed a three-fold cross validation when 30% of the dataset is used for training. For this, we first separated the test set. Then, folds have been created by randomly sampling (without replacement) the remaining training set in a stratified manner (proportion of each object class is preserved in the subsampled sets). In other words, three different subsets were sampled from the training data and the subset size corresponded to 30% of the size of the entire ModelNet10 or ModelNet40 dataset (training + test sets). After three-fold cross validation, for ModelNet-10 dataset, the average accuracies obtained are 93.21% and 90.95% for our proposed 3DCapsNet and Point2Sequence, respectively. For ModelNet-40 dataset, the average accuracies obtained are 82.75% and 81.22% for our proposed 3DCapsNet and Point2Sequence, respectively. Thus, even before weight pruning, our proposed 3DCapsNet performs better when only limited amount of training data is available. It can also be seen from Tables 1 and 2 that the base model we employ performs comparably to VoxNet [21], which are similar in their architecture with some subtle differences in hyperparameters. For instance, the base model has deeper and wider convolutional layers than VoxNet and it has larger dense layers.

Table 2 also shows the significant decrease in the number of network parameters obtained with the ADMM-based pruning approach that we use on the base model and the 3D CapsNet. More discussion on this will be provided in Section V-C. Overall, the proposed 3D CapsNet model performs better than the other methods, even with much fewer parameters after weight pruning. We also observe that 3D CapsNet consistently performs better on the objects with rounded parts. This shows that 3D CapsNets are more capable of adapting surfaces with smoother curvatures, while maintaining the accuracy of the objects with more aggressive curvatures, such as sharp edges. In order to show the superiority of our model in extreme scenario, Table 3 presents the individual class accuracies of the proposed 3D CapsNets trained on the ModelNet-10 dataset with only 5% training set split ratio.

ShapeNetCore55, on the other hand, is a very challenging dataset due to large variations within a class. Furthermore, it is highly unbalanced, and it did not yield to a stable training when the whole dataset is fed during the training at once. In order to evaluate the classification accuracy of 3D CapsNets, we followed an iterative approach to train our network with ShapeNetCore55 data. We divided the dataset into batches of 10 classes, which are sorted in descending order based on the number of samples. We started the training with the data only in the first batch of classes, which has the largest representation in the dataset. The network quickly adapted to the given training set and we introduced second batch of class together with the previous one. We repeated this until all 55 classes are finally started to being trained stably. We sampled 20% of the data in a stratified manner based on the class labels as the training set and used the rest as the test set. Due to the complexity of the training procedure, we only compare the base network and the proposed 3D CapsNet model. Since ADMM-based weight pruning is not applied to models trained on ShapeNetCore55 dataset, the results are presented in Table 1. The large margin proves the superiority of the proposed model on this very challenging dataset.

Fig. 7 shows a few correct and incorrect classification samples from ModelNet-10 with 3D CapsNets. We observe that our architecture distinguishes the object with strong confidence in most of the cases. The failure cases of 3D CapsNets include the ones in which the objects are similar in their appearance and nature, such as desk and table, table and night-stand. Since there is no softmax applied and the squashing function normalizes the length of each capsule vector to 1, represented values on the table are the length of the corresponding capsules in the class capsule layer. Therefore the value indicates how strong the given capsule, which is the class itself for the last layer, is represented in the object, rather than indicating the probability of belonging to that class.

C. NETWORK OPTIMIZATION

From Table 2, it can be seen that almost all proposed CapsNet models provide higher accuracy than their base model counterparts, even though the base models have more weight parameters. It can demonstrate that CapsNet has stronger capability to extract 3D features with smaller amounts of

TABLE 3. Individual class accuracies (%) on ModelNet-10 with 5% training set without weight pruning.

	Bathtub	Bed	Chair	Desk	Dresser	Monitor	NightStand	Sofa	Table	Toilet	Overall
Base model	24.44	**93.05**	**97.72**	48.55	**69.30**	**83.89**	46.81	90.95	83.33	74.05	81.12
3D CapsNet	**60.00**	92.48	97.24	**52.28**	66.23	80.54	**65.53**	**94.12**	**83.33**	**82.70**	**84.13**

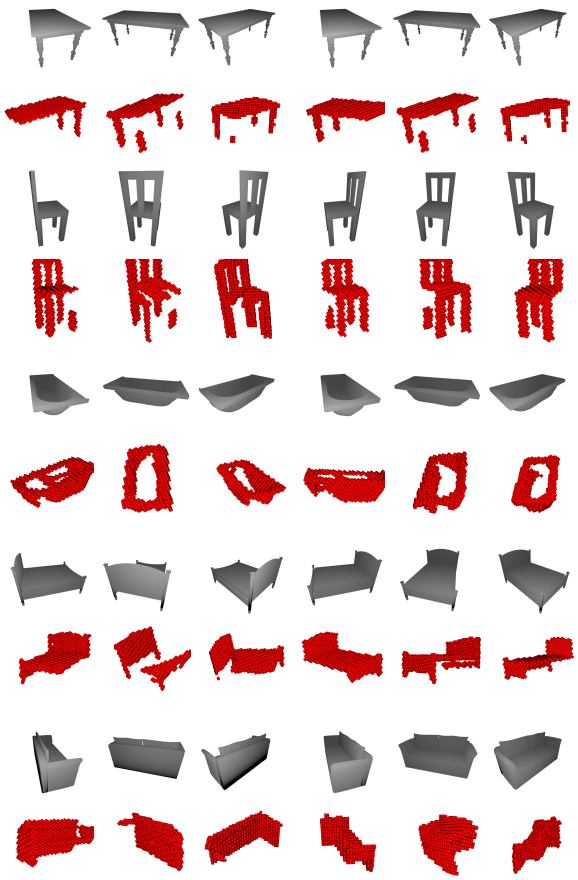


FIGURE 4. Rendered meshes (top) from ShapeNet database and corresponding voxel grid representations (bottom) of their visible surfaces. Viewpoints of voxel grid representations are adjusted to illustrate the effect of self-occlusion better. Objects from top to bottom: table, chair, bathtub, bed, sofa.

training data. Moreover, the CapsNet models can achieve higher compression ratio than their base model counterparts. This is because the capsules are more powerful in extracting pose information, so less parameters are needed for CapsNet to maintain the accuracy. When comparing the models trained with 30% data with the models trained with 60% data, it can be found that higher compression ratios are achieved on the 30% data-trained models. This is because the training data size is smaller, so the 30% data-trained models are more over-parameterized than the 60% data-trained models. Overall, 3D CapsNet models, on which ADMM-based weight pruning applied, perform the best among all other methods in terms of classification accuracy with very significant model compression and fewer network parameters.

TABLE 4. Partial data classification results.

Model	Test Set	5%	40%
Base Model	Individual	76.7	84.94
3DCapsNet	Individual	**79.31**	**87.68**
Base Model	Voting	82.06	91.12
3DCapsNet	Voting	**85.41**	**94.41**

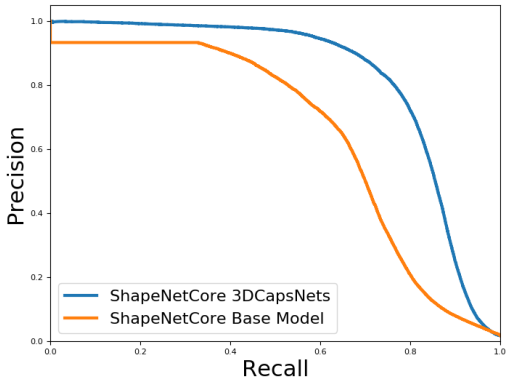


FIGURE 5. Average Precision-Recall curves for partial data training.

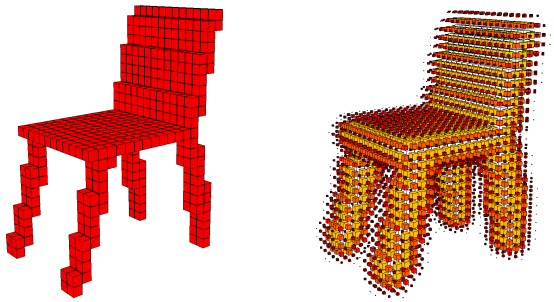


FIGURE 6. 3D binary voxel grid (left) and TDF encoded 3D voxel grid (right). Smaller and darker (red) points are far from a surface whereas lighter (yellow) and larger points are close to a surface.

D. PARTIAL DATA RESULTS

Synthetic CAD models are usually the exact replica of the objects and they possess all of their geometric properties. However, when an actual object is captured by a sensor or rendered in a software, the captured data usually corresponds to only a small fraction of the object. First, only the surface can be captured, and the internal parts are completely invisible to the sensor or the renderer. Second, the back part of the object cannot be captured from a single viewpoint, since object itself occludes the sensor's or renderer's view. A well-designed classifier model should be able to work with such

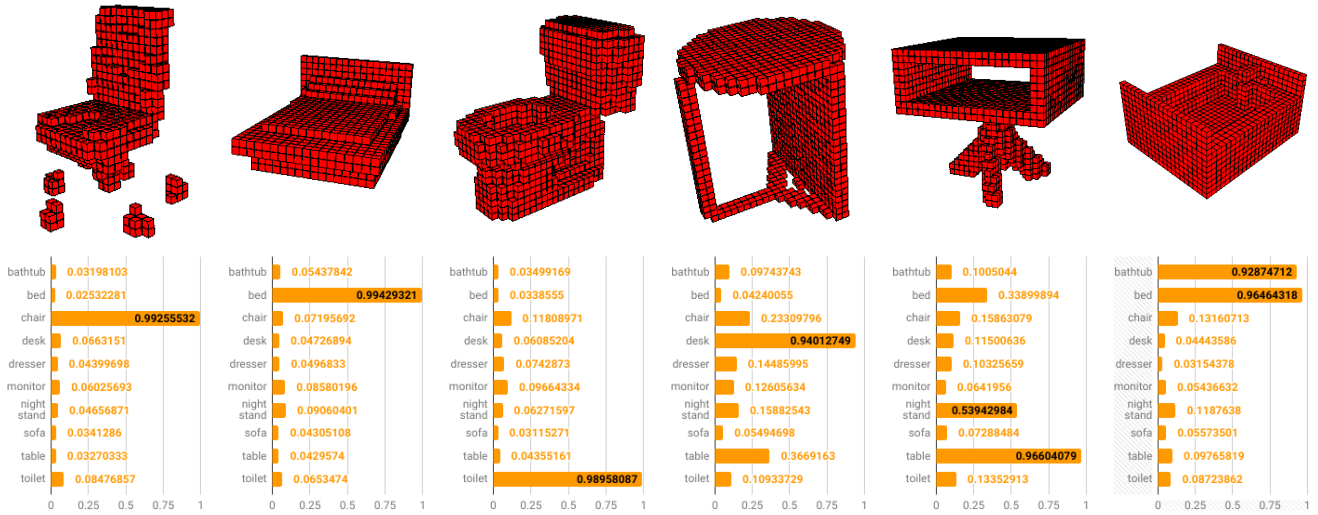


FIGURE 7. Examples of three correct (left) and three false classifications (right). We observed that falsely classified objects are usually confused with semantically or geometrically similar classes, such as table-desk, nightstand-table, bed-bathtub etc.

data since it is not very likely to have the full object in an application. With this in mind, we tested our approach with 3D surfaces from different views of the objects using a small subset of ShapeNetCore55 dataset. We used Blender software [40] to get the depth map of the object meshes from various viewpoints. These depth maps are then converted to point clouds and voxelized. We extracted surface data from 12 different viewpoints for each mesh in the dataset. Fig. 4 shows mesh samples of 5 common objects and voxel grid representations corresponding to their rendered depth maps from 6 different viewpoints.

Four experiments have been performed on both 3D CapsNet architecture and the base model. Each of these experiments are the combination of following two conditions: 1) With and without test-time voting, 2) 5% and 40% training dataset. Test-time voting is the majority voting accuracy where the predicted label of an object is determined based on the prediction of its different viewpoints. Without majority voting (i.e., individual), all test data is calculated independently in the overall accuracy calculation. Table 4 shows that, with or without majority voting, 3D CapsNet performs considerably better than the base model in both 5% and 40% training data split rates. Fig. 5 shows the precision-recall curve comparsion for partial data results of the 3D CapsNet and base model with varying threshold.

E. INPUT ENCODING RESULTS

The representation of the data is of great importance in 3D. An object can be represented by various different ways, which may or may not convert to each other. Point cloud representation of a 3D object, P, can be generated from a depth map M by the equation $P_{N\times 3} = f(M_{H\times W}, \theta)$, where θ is the intrinsic camera parameters and f is a linear mapping and bijection. If the initial representation of the object is a 3D mesh, such as a synthetic object, the 3D point cloud representation can be created by pseudorandom sampling of the mesh representation. Then these point clouds are quantized into regular 3D grids, which is also known as voxelization, for processing in a neural architecture. Although some information (detail) is lost while quantizing a 3D point cloud into a voxel grid (voxelization), the resulting voxel grid is still rich in preserving geometric properties of the input object.

We used the standard binary occupancy voxel grid encoding throughout all the experiments above. In order to validate the classification accuracy of 3D CapsNets, we performed a comparison analysis with other 3D encodings. In the first scheme, point normal vectors are used instead of binary occupancies. We used 0.5% of the size of the object as normal estimation radius. The shape of input object became $30\times30\times30\times3$ where the last dimension holds the XYZ values of normal vectors while leaving the values for empty regions as zero.

Beyond the binary occupancies and point normals, there is an encoding scheme called Truncated Distance Function (TDF), proposed in [41], in which each voxel contains a floating point number value which is the distance to the closest surface. TDF is the absolute value of TSDF (Truncated signed distance function) encoded data, which is often used in computer graphics [42]. Examples of a TDF encoded voxel grid can be seen in Fig. 6. We tested our algorithm with TDF encoded voxel grids. With TDF encoded data, the network is fed with some information about the surfaces beforehand, which helps the training to catch curvature features better. The shape of the TDF encoded data is still $30 \times 30 \times 30$ as in the binary grids, while the values become continuous between $[0 - 1]$.

Table 5 shows the comparison result of three encoding schemes used to train 3D CapsNets. Although the accuracy increase is not very large, we observe that introducing

TABLE 5. Accuracy comparison with different input encodings.

	ModelNet-10	ShapeNetCore55
Base Model	84.58%	85.60%
3D CapsNets (Binary)	86.53%	88.67%
3D CapsNets (Normal)	86.91%	89.24%
3D CapsNets (TDF)	87.75%	89.70%

prior information for surface and curvature with point normals or TDF helps in classification.

F. ROUTING MECHANISM OPTIMIZATION

We observed in our experiments that tuning the number of dynamic routing iterations does not lead to considerable improvement. In fact, we achieved the same accuracy with routing the capsule activation once without any iteration, i.e., when the number of iterations is one. Therefore, we optimized the original dynamic routing mechanism. Because we hypothesize that each capsule will learn certain entity, the relationship between these abstractions between layers should also be learnable. Thus, in our modified procedure, instead of predicting correspondence coefficients iteratively in a loop based on the predictions of next layer, we made these parameters trainable. We also used 70% dropout during training in order to avoid overfitting of these coefficients to certain capsule connections. These learned coefficients are then applied as weights to predicted capsules of next layer. This masking procedure can be thought of as a soft-attention mechanism [43], which is deterministic in nature, between capsules in consecutive layers. Finally, the squashing is applied to get the capsule of the next layer. The accuracy results of the non-iterative method is on par with the vanilla 3D CapsNets while runtime speed gets 15% faster with the optimized routing.

VI. CONCLUSION

We proposed an improved version of our 3D object classification method, referred to as 3D CapsNet, which captures part-relationships better and requires less data for training. Despite its ability to extract features from 3D objects, the large model size of the proposed 3D CapsNet hurdles its deployment on resource-constrained platforms, such as mobile phones, IoT devices, and unmanned vehicles. In order to address this, we have also introduced an ADMM-based weight pruning method to significantly reduce the number of parameters of our proposed 3D CapsNet with a further improvement on the accuracy. We have shown the performance of our proposed approach on comprehensive set of experiments including comparison of accuracy, number of weights and compression ratio between different approaches on different datasets. The detailed analysis have shown that our proposed 3D CapsNet models, to which ADMM-based weight pruning is applied, performs the best among all other methods in terms of classification accuracy with very significant model compression and fewer network parameters. The results also show the strength of the proposed 3D CapsNet (before and after weight pruning) especially when smaller amounts of training data is used. Moreover, we further optimized our model for efficient computation by modifying the original dynamic routing mechanism while maintaining the classification accuracy.

Our presented ADMM-based weight pruning approach is a technique, which can be easily applied to other models to improve their performance further while shrinking their size for efficient memory utilization. This is, in fact, one of the reasons that we have compared our 3D CapsNet with a base model, which is composed of basic building blocks of a traditional CNN, in order to showcase the evident improvement achieved due to capsules.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[3] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," 2017, *arXiv:1701.06547*. [Online]. Available: https://arxiv.org/abs/1701.06547

[4] Y. Liu, Y. Zhou, X. Liu, F. Dong, C. Wang, and Z. Wang, "Wasserstein GAN-based small-sample augmentation for new-generation artificial intelligence: A case study of cancer-staging data in biology," *Engineering*, vol. 5, no. 1, pp. 156–163, Feb. 2019.

[5] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.

[6] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, Dec. 2015.

[7] T. Wang, Z. Miao, Y. Chen, Y. Zhou, G. Shan, and H. Snoussi, "AED-Net: An abnormal event detection network," *Engineering*, vol. 5, no. 5, pp. 930–939, Oct. 2019.

[8] T. Wang, Y. Chen, M. Zhang, J. Chen, and H. Snoussi, "Internal transfer learning for improving performance in human action recognition for small datasets," *IEEE Access*, vol. 5, pp. 17627–17633, 2017.

[9] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3859–3869.

[10] B. Kakillioglu, A. Ahmad, and S. Velipasalar, "Object classification from 3D volumetric data with 3D capsule networks," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 385–389.

[11] A. Ahmad, B. Kakillioglu, and S. Velipasalar, "3D capsule networks for object classification from 3D model data," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 2225–2229.

[12] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "ADMM-NN: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2019, pp. 925–938.

[13] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[14] T. Chen, J. Lin, T. Lin, S. Han, C. Wang, and D. Zhou, "Adaptive mixture of low-rank factorizations for compact neural modeling," in *Proc. NIPS Workshop CDNNRIA*, Montreal, QC, Canada, Dec. 2018. [Online]. Available: https://openreview.net/forum?id=B1eHgu-Fim and https://nips.cc/Conferences/2018/Schedule?showEvent=10941

[15] L. A. Alexandre, "3D object recognition using convolutional neural networks with transfer learning between input channels," in *Intelligent Autonomous Systems*. Cham, Switzerland: Springer, pp. 889–898, 2016.

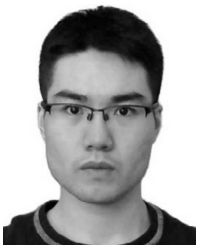
[16] X. Lv, X. Liu, X. Li, X. Li, S. Jiang, and Z. He, "Modality-specific and hierarchical feature learning for RGB-D hand-held object recognition," *Multimed Tools Appl*, vol. 76, no. 3, pp. 4273–4290, Feb. 2017.

[17] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015.

[18] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3813–3822.

[19] K. Sfikas, T. Theoharis, and I. Pratikakis, "Exploiting the PANORAMA representation for convolutional neural network classification and retrieval," in *Proc. Eurograph. Workshop 3D Object Retr.*, Lyon, France, 2017, pp. 1–7.
[20] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1912–1920.
[21] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.
[22] C. R. Qi, H. Su, M. Niebner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5648–5656.
[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," 2016, *arXiv:1612.00593*. [Online]. Available: https://arxiv.org/abs/1612.00593
[24] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2017, pp. 3391–3401.
[25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.
[26] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network," in *Proc. AAAI*, vol. 33, Aug. 2019, pp. 8778–8785.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2012, pp. 1097–1105.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[29] A. M. Grachev, D. I. Ignatov, and A. V. Savchenko, "Neural networks compression for language modeling," in *Proc. Int. Conf. Pattern Recognit. Mach. Intell.* Cham, Switzerland: Springer, 2017, pp. 351–357.
[30] X. Dai, H. Yin, and N. K. Jha, "NeST: A neural network synthesis tool based on a grow-and-prune paradigm," *IEEE Trans. Comput.*, vol. 68, no. 10, pp. 1487–1497, Oct. 2019.
[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[33] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. Int. Conf. Artif. Neural Netw.* Berlin, Germany: Springer, 2011, pp. 44–51.
[34] K. Guo, S. Han, S. Yao, Y. Wang, Y. Xie, and H. Yang, "Software-hardware codesign for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 18–25, Mar. 2017, pp. 18–25.
[35] H. Ouyang, N. He, L. Tran, and A. Gray, "Stochastic alternating direction method of multipliers," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 80–88.
[36] T. Suzuki, "Dual averaging and proximal gradient descent for online alternating direction multiplier method," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 392–400.
[37] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *FNT Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.
[38] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, and H. Su, "Shapenet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*. [Online]. Available: https://arxiv.org/abs/1512.03012
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[40] B. O. Community, *Blender—A 3D Modelling and Rendering Package*. Amsterdam, The Netherlands: Blender Foundation, Stichting Blender Foundation, 2018.
[41] A. Zeng, S. Song, M. Niebner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1802–1811.
[42] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1996, pp. 303–312.
[43] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.

BURAK KAKILLIOGLU (Student Member, IEEE) received the B.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA. In 2019, he was a Research Intern at the Computer Vision Technologies Group, SRI International, Princeton, NJ, USA. His research interests include 3D data understanding, computer vision, machine learning, and embedded systems.



AO REN (Student Member, IEEE) received the B.S. degree in integrated circuit design and integrated systems from the Dalian University of Technology, Dalian, China, in 2013, and the M.S. degree in computer engineering from Syracuse University, Syracuse, NY, USA, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. His research interests include neuromorphic computing, deep neural network acceleration, and low power design.

YANZHI WANG (Member, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, in 2009, and the Ph.D. degree in computer engineering from the University of Southern California, in 2014. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Northeastern University. Besides, he works on the application of deep learning and machine intelligence in various mobile and IoT systems, medical systems, and UAVs, and the integration of security protection in deep learning systems. His works have been published in top venues in conferences and journals (e.g., ASPLOS, MICRO, HPCA, ISSCC, AAAI, ICML, ICLR, ECCV, ACM MM, CCS, VLDB, FPGA, DAC, ICCAD, DATE, LCTES, INFOCOM, ICDCS, and Nature SP), and have been cited for around 4,000 times according to Google Scholar. His current research interests are the energy-efficient and high-performance implementations of deep learning and artificial intelligence systems. He has received four Best Paper Awards, has another seven Best Paper Nominations and two Popular Papers in IEEE TCAD. His group is sponsored by NSF, DARPA, IARPA, AFRL/AFOSR, and industry sources.

SENEM VELIPASALAR (Senior Member, IEEE) received the B.S. degree in electrical and electronic engineering from Bogazici University, Istanbul, Turkey, in 1999, the M.S. degree in electrical sciences and computer engineering from Brown University, Providence, RI, USA, in 2001, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 2004 and 2007, respectively.

From 2001 to 2005, she was with the Exploratory Computer Vision Group, IBM T. J. Watson Research Center, NY, USA. From 2007 to 2011, she was an Assistant Professor with the Department of Electrical Engineering, University of Nebraska-Lincoln, Lincoln. She is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, Syracuse University. The focus of her research has been on mobile camera applications, wireless embedded smart cameras, multicamera tracking and surveillance systems, and automatic event detection from videos. Dr. Velipasalar received a Faculty Early Career Development Award (CAREER) from the National Science Foundation, in 2011. She was a recipient of the 2014 Excellence in Graduate Education Faculty Recognition Award. She is the coauthor of the paper, which received the 2017 IEEE Green Communications and Computing Technical Committee Best Journal Paper Award. She received the Best Student Paper Award at the IEEE International Conference on Multimedia and Expo, in 2006. She was a recipient of the EPSCoR First Award, IBM Patent Application Award, and Princeton and Brown University Graduate Fellowships. She is a member of the Editorial Board of the IEEE TRANSACTIONS ON IMAGE PROCESSING and Springer *Journal of Signal Processing Systems*.