# Dominant Set Clustering and Pooling for Multi-View 3D Object Recognition.

Chu Wang[1]
http://www.cim.mcgill.ca/~chuwang/
Marcello Pelillo[2]
http://www.dsi.unive.it/~pelillo/
Kaleem Siddiqi[1]
http://www.cim.mcgill.ca/~siddiqi/

[1] School of Computer Science
McGill University
Montréal, Canada

[2] DAIS / ECLT
Ca' Foscari University of Venice
Italy

**Abstract**

View based strategies for 3D object recognition have proven to be very successful. The state-of-the-art methods now achieve over 90% correct category level recognition performance on appearance images. We improve upon these methods by introducing a view clustering and pooling layer based on *dominant sets*. The key idea is to pool information from views which are similar and thus belong to the same cluster. The pooled feature vectors are then fed as inputs to the same layer, in a recurrent fashion. This recurrent clustering and pooling module, when inserted in an off-the-shelf pretrained CNN, boosts performance for multi-view 3D object recognition, achieving a new state of the art test set recognition accuracy of 93.8% on the ModelNet 40 database. We also explore a fast approximate learning strategy for our cluster-pooling CNN, which, while sacrificing end-to-end learning, greatly improves its training efficiency with only a slight reduction of recognition accuracy to 93.3%. Our implementation is available at `https://github.com/fate3439/dscnn`.

## 1 Introduction

Appearance based object recognition remains a fundamental challenge to computer vision systems. Recent strategies have focused largely on learning category level object labels from 2D features obtained from projection. With the parallel advance in 3D sensing technologies, such as the Kinect, we also have the real possibility to seamless include features derived from 3D shape into recognition pipelines. There is also a growing interest in 3D shape recognition from databases of 3D mesh models, acquired from computer graphics databases [1] or reconstructed from point cloud data [18].

3D object recognition from shape features may be categorized into *view-based* versus *volumetric* approaches. View based approaches including those in [6, 12, 13, 15] create hand-designed feature descriptors from 2D renderings of a 3D object by combing information across different views. 3D object recognition is then reduced to a classification problem on the designed feature descriptors. More recent methods in this class combine CNN features from the multiple views to boost object recognition accuracy [2, 9, 18, 20]. Volumetric approaches rely on 3D features computed directly from native 3D representations, including

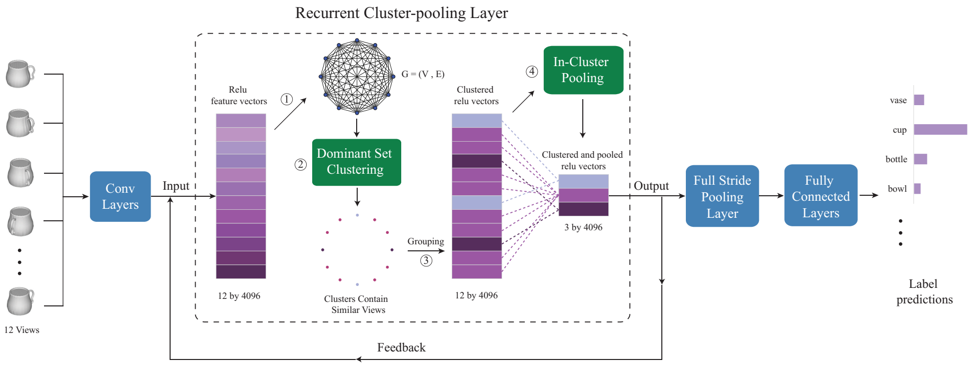

Figure 1: Our recurrent clustering and pooling CNN. We insert our recurrent cluster-pooling layer (dashed box) after the relu layers (relu6 and relu7 are typical choices) of an Imagenet pretrained VGG-m network. A set of multi-view 2D feature maps (12 views in this example), obtained by projection of the 3D mesh model, is used as input to our system. The proposed layer takes relu vectors of the previous layer as its input and performs dominant set clustering on a view similarity graph followed by within cluster pooling in a recurrent manner. Once the clusters are stable the outputs serve as inputs to a full stride pooling layer, following which the multi-view relu vectors (12 by 4096) become one unified vector (1 by 4096), which is processed by fully connected layers to produce predicted object category labels.

meshes, voxelized 3D grids and point clouds [8, 10, 11, 19]. The state-of-the-art here is the use of 3D convolutional neural networks on discretized occupancy grids for feature extraction for further processing or for direct classification [14, 18, 22, 23]. At the present time, at least when evaluated on popular 3D object model databases, the view based approaches [9, 20] outperform the volumetric ones [14, 23], as reported in the extensive comparison in [18].

The state-of-the-art view based methods differ in the manner in which multi-view information is fused. In the MVCNN approach [20], a view pooling layer is inserted in a VGG-m style network to perform multi-view feature fusion of CNN relu vectors. This view pooling layer performs a full stride channel-wise max pooling to acquire a unified feature vector, following which fully connected layers are used to predict category labels. In the pairwise decomposition method in [9], two CNNs are used, one for view pair selection and one for pairwise label prediction. The two CNNs each use a VGG-m structure [5] but they have to be trained separately, which is costly. At the expense of increased training cost, the pairwise formulation out performs the MVCNN approach.

In the present article we propose a revised architecture which aims to overcome potential limitations of the above two strategies, namely: 1) the winner-take-all pooling strategy in [20], which could discard information from possibly informative views, and 2) the pairwise formulation of [9]. The key contribution is the introduction of a recurrent clustering and pooling module based on the concept of dominant sets, as illustrated in Figure (1). The 2D views of an object are abstracted into relu vectors, which serve as inputs to this new layer. Within this layer we construct a view similarity graph, whose nodes are feature vectors corresponding to views and whose edge weights are pairwise view similarities. We then find dominant sets within this graph, which exhibit high within cluster similarity and between cluster dissimilarity. Finally, we carry out channel wise pooling but only from *within* each dominant set. If the dominant sets have changed from the previous iteration, they are fed back as new feature vectors to the same layer, and the clustering and pooling process is repeated. But if not, they are fed forward to the next full stride pooling layer. In contrast to the MVCNN approach of [20] we only pool information across similar views. The recurrent nature allows for the feature vectors themselves to be iteratively refined. Our recurrent cluster-pooling layer, followed by a full stride view pooling layer, can be inserted after a pre-trained network's relu layers to yield a unified multi-view network which can be trained in an end-to-end manner.[1] Our experimental results in Section (3) show that when inserted before the view pooling layer in the MVCNN architecture of [20], our recurrent clustering and pooling unit greatly boosts performance, achieving new state-of-the-art results in multi-view 3D object recognition on the ModelNet40 dataset.

## 2 Recurrent Clustering and Pooling Layer

The recurrent clustering and pooling layer requires the construction of a view similarity graph, the clustering of nodes (views) within this graph based on dominant sets, and the pooling of information from within each cluster. We now discuss these steps in greater depth and provide implementation details.

### 2.1 A View Similarity Graph

First, a pairwise view similarity measure is defined for any two views in the set of rendered views of a 3D object. We then construct a view similarity graph $G = (V, E, w)$ where views $i, j \in V$ are distinct nodes and each edge $E(i, j)$ has a weight $w(i, j)$ corresponding to the similarity between the views $i$ and $j$. This results in a complete weighted undirected graph $G = K_n$, where $n$ is the number of views.

Different rendered views have different appearances, which are in turn captured with low dimensional signatures by the relu features from a CNN (we typically apply relu6 or relu7 vectors). A very convenient notion of pairwise view *similarity* between the appearance images of views $i$ and $j$ is therefore given by the inner product of the corresponding CNN relu feature vectors $r_i$ and $r_j$:

$$w(i, j) = r_i \cdot r_j. \tag{1}$$

We exploit the property that the components of relu feature vectors from a CNN, where relu stands for "rectified linear units", are non-negative and have finite values that tend to lie within a certain range. The larger $w(i, j)$ is, the more similar the two views $i$ and $j$ are.

### 2.2 Dominant Set Clustering

We now cluster views within the view similarity graph, based on the concept of dominant sets [16, 17]. The views to be clustered are represented as an undirected edge-weighted graph with no self-loops $G = (V, E, w)$, where $V = \{1, ..., n\}$ is the vertex set, $E \subseteq V \times V$ is the edge set, and $w : E \to \mathbb{R}_+^*$ is the (positive) weight function. As explained in Section (2.1), the vertices in $G$ correspond to relu vectors abstracted from different rendered views of a given object, the edges represent view relationships between all possible view pairs, and

[1]Note that the within-cluster pooling operation is fixed: it is either max or average pooling throughout the recurrence.

the edge-weights encode similarity between pairs of views. We compute the affinity matrix of $G$, which is the $n \times n$ nonnegative, symmetric matrix $A = (a_{ij})$ with entries $a_{ij} = w(i, j)$. Since in $G$ there are no self-loops, all entries on the main diagonal of $A$ are zero.

For a non-empty subset $S \subseteq V$, $i \in S$, and $j \notin S$, let us define

$$\phi_S(i, j) = a_{ij} - \frac{1}{|S|} \sum_{k \in S} a_{ik} \tag{2}$$

which measures the (relative) similarity between vertices $j$ and $i$, with respect to the average similarity between $i$ and its neighbors in $S$. Next, to each vertex $i \in S$ we assign a weight defined (recursively) as follows:

$$w_S(i) = \begin{cases} 1, & \text{if} \quad |S| = 1, \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(j, i) w_{S \setminus \{i\}}(j), & \text{otherwise.} \end{cases} \tag{3}$$

As explained in [16, 17], a positive $w_S(i)$ indicates that adding $i$ to the elements in $S$ will increase its internal coherence, whereas a negative weight indicates that adding $i$ will cause the overall coherence of $S$ to be decreased. Finally, we define the total weight of $S$

$$W(S) = \sum_{i \in S} w_S(i) \; . \tag{4}$$

**Definition 2.1.** A non-empty subset of vertices $S \subseteq V$ such that $W(T) > 0$ for any non-empty $T \subseteq S$, is said to be a *dominant set* if:

1. $w_S(i) > 0$, for all $i \in S$,
2. $w_{S \cup \{i\}}(i) < 0$, for all $i \notin S$.

It is evident from its definition that a dominant set satisfies the two properties of a cluster that we desire, namely, it yields a partition of $G$ with a high degree of intra-cluster similarity and inter-cluster dissimilarity. Condition 1 indicates that a dominant set is internally coherent, while condition 2 implies that this coherence will be decreased by the addition of any other vertex. A simple and effective optimization algorithm to extract a dominant set from a graph based on the use of *replicator dynamics* can be found in [4, 16, 17], with a run time complexity of $\mathcal{O}(V^2)$, where $V$ is the number of vertices in the graph. We adopt this algorithm in our implementation.

## 2.3 Clustering, Pooling and Recurrence

After obtaining a partition into dominant sets, we propose to perform pooling operations only *within* each cluster. This allows the network to take advantage of informative features from possibly disparate views. The resultant relu vectors are then fed back to the beginning of the clustering and pooling layer, to serve as inputs for the next recurrence. This process is repeated, in a cyclical fashion, until the clusters (dominant sets) do not change. During the recurrences, we alternate max and average pooling, which allows for further abstraction of informative features. A full stride max pooling is applied to the relu vectors corresponding to the final, stable, dominant sets. We carry out experiments to demonstrate the effect of different recurrent structures and pooling combinations in Section (3.2).

Figure (2) depicts the cluster-pooling recurrences for a mug, with 12 initial input views. After the first stage there are 3 clusters, one corresponding to side views (blue), a second

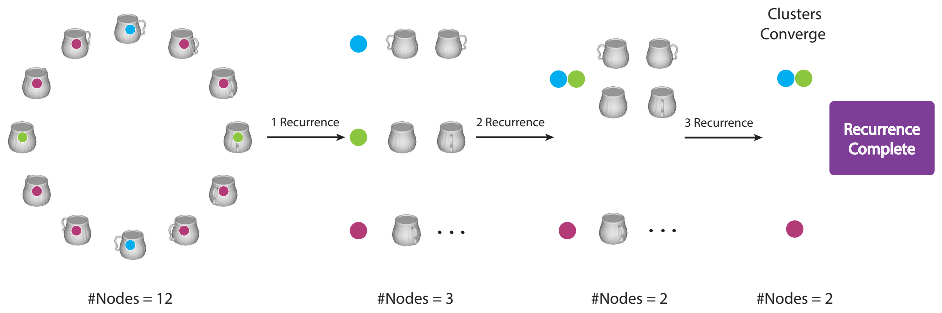

Figure 2: We provide an illustration of the clustering and pooling recurrences that occur for the case of a mug with 12 initial views. See text for a discussion.

corresponding to front or rear views (green) and a third corresponding to oblique views (maroon). At this stage the three relu feature vectors represent pooled information from within these distinct view types. After the second recurrence, information from the green and blue views is combined, following which no additional clusters are formed.

The proposed recurrent clustering and pooling strategy aims to improve discrimination between different input objects. When input relu features are changed, our method increases the likelihood of variations in the aggregated multi-view relu output. In contrast, a single max pooling operation, as in the MVCNN approach of [20], will result in the same output, unless the change in the inputs cause the present max value to be surpassed. The ability to capture subtle changes in relu space helps in discrimination between similar object categories. Dominant set clustering seeks to prevent outliers in a given cluster, and the within cluster pooling decreases the likelihood that the aggregated result is determined by the single relu vector that gives the maximum response.

## 2.4 Back propagation

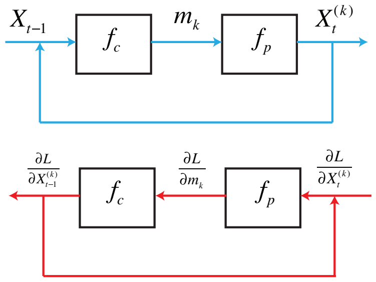
$X_{t-1}$
$f_c$
$m_k$
$f_p$
$X_t^{(k)}$
$\frac{\partial L}{\partial X_{t-1}^{(k)}}$
$\frac{\partial L}{\partial m_k}$
$\frac{\partial L}{\partial X_t^{(k)}}$

Figure 3: The recurrent clustering and pooling layer's forward and backward pass with respect to the k-th cluster. Recurrence: (t-1).

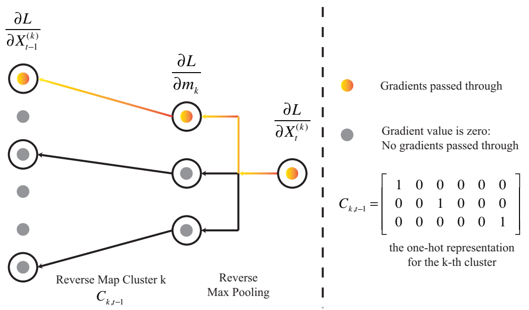
$\frac{\partial L}{\partial X_{t-1}^{(k)}}$
$\frac{\partial L}{\partial m_k}$
$\frac{\partial L}{\partial X_t^{(k)}}$
Gradients passed through
Gradient value is zero: No gradients passed through
$C_{k,t-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
the one-hot representation for the k-th cluster
Reverse Map Cluster k $C_{k,t-1}$
Reverse Max Pooling

Figure 4: A toy example with 6 input views. The back propagation for the k-th cluster output with a max within-cluster pooling is illustrated in this example.

There are no parameters to be learned in our recurrent clustering and pooling layer, therefore we aim to derive the gradients w.r.t this layers' input given gradients w.r.t output to facilitate learning at preceding layers. Details of the forward pass and back propagation steps are illustrated in Figure (3). Here $f_c$ represents the dominant set clustering unit which takes as input the (t-1)-th recurrence $X_{t-1} \in \mathbb{R}^{n_{t-1} \times d}$ and outputs cluster assignments, where $n_{t-1}$ is the number of input nodes and $d$ is relu vector's dimension. $f_p$ stands for the within-cluster pooling unit, which takes as input relu vectors belonging to the k-th resultant cluster $m_k \in \mathbb{R}^{c_k \times d}$ and outputs a pooled relu vector, where $c_k$ is the number of nodes in k-th cluster. During the *forward* pass, we first acquire cluster assignments for the (t-1)-th recurrence using the dominant set algorithm $C_{k,t-1} = f_c\left(A_{t-1}\right)$, where $k$ stands for an arbitrary resultant cluster and $A_{t-1}$ stands for the affinity matrix of the constructed similarity graph. A one hot cluster representation matrix $C_{k,t-1} \in \mathbb{R}^{c_k \times n_{t-1}}$ is constructed in the following manner. For an arbitrary cluster $k$ containing input nodes $\{k_1, k_2, k_3, ..., k_{c_k}\}$, the i-th row in its cluster representation matrix is a one-hot vector encoding value $k_i$. Now, inputs belonging to the k-th cluster can be represented as

$$m_k = C_{(k,t-1)} X_{t-1}. \tag{5}$$

The within-cluster pooling unit will then give $X_t^{(k)} = f_p\left(m_k\right)$, where $X_t^{(k)} \in \mathbb{R}^{1 \times d}$ is the pooled relu vector of cluster k. The above process applies to all resultant clusters.

To establish the formulas for *back propagation* of the recurrent clustering and pooling layer, we define the cross entropy loss functions as $L$. We note that the backward pass requires the same amount of recurrence as the forward pass, but the direction of data flow is opposite as shown in Figure (3). During the backward pass of (t-1)-th recurrence, we iteratively loop through clusters to accumulate the gradients of the loss function w.r.t the (t-1)-th recurrence's input defined as $\frac{\partial L}{\partial X_{t-1}}$. For a given resultant cluster $k$, gradients will be mapped back only to those input nodes that belong to this very cluster. If we define the gradients w.r.t the recurrent layer's inputs of cluster k as $\frac{\partial L}{\partial X_{t-1}^{(k)}}$, gradients w.r.t output as $\frac{\partial L}{\partial X_t^{(k)}}$, and w.r.t pre-pooling as $\frac{\partial L}{\partial m_k}$, we have the following equations:

$$\frac{\partial L}{\partial m_k} = f_p^{-1}\left(\frac{\partial L}{\partial X_t^{(k)}}\right) \tag{6a}$$

$$\frac{\partial L}{\partial X_{t-1}^{(k)}} = C_{k,t-1}^T \frac{\partial L}{\partial m_k} \tag{6b}$$

which are derived by reversing the operations of pooling and then clustering. Gradients with respect to the recurrent layer's input are then given by

$$\frac{\partial L}{\partial X_{t-1}} = \sum_k \frac{\partial L}{\partial X_{t-1}^{(k)}} = \sum_k C_{k,t-1}^T f_p^{-1}\left(\frac{\partial L}{\partial X_t^{(k)}}\right). \tag{7}$$

The toy example in Figure (4) illustrates a scenario with 6 input views using within-cluster max pooling. When back propagating the gradients, the orange cells represent the relu positions where gradients have passed through. We note that the grey cells' gradients will always remain zero and gradients w.r.t input $X_{t-1}$ will only have non-zero values at nodes belongs to the k-th cluster.

## 3 Experiments

**Network Setup.** We use the same baseline CNN structure as in [9, 20], which is a VGG-M network [5] containing five convolutional layers followed by three fully connected layers. We follow the same process of network pretraining and task specific network fine tuning as in [9, 18, 20]. Specifically, we use the Imagenet 1k pretrained VGG-m network provided in

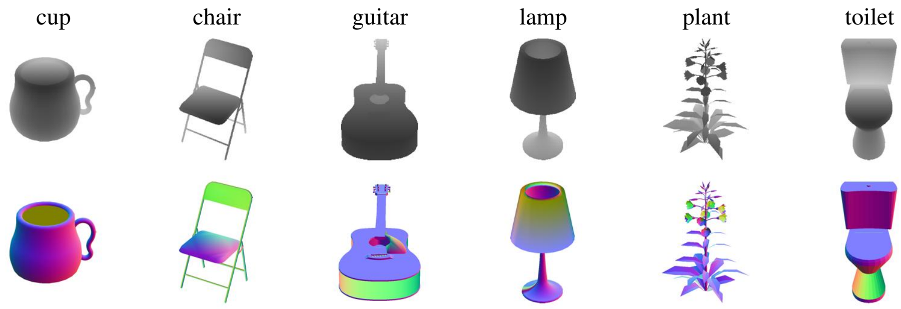

Figure 5: Rendered views of ModelNet40 objects using additional feature modalities. Top row: depth map. Bottom row: surface normals rendered with an RGB colormap, with green pointing towards the viewer, blue pointing upwards and red pointing right.

[5] and fine tune it on the ModelNet 40 training set after our recurrent clustering and pooling layer is inserted. In our experiments, we insert customized layers after the relu6 layer.

**Dataset.** We evaluate our method on the Princeton ModelNet40 dataset [1] which contains 12, 311 3D CAD models from 40 categories. This dataset is well annotated and many state-of-the-art approaches have reported their results on it [9, 18, 20]. The dataset also provides a training and testing split, in which there are 9,843 training and 2,468 test models [2]. We use the entire training and testing set for experiments in Sections (3.2) and we provide results for both the full set and the subset when comparing against related works in Section (3.3).

**Rendering and additional feature modalities.** We render the 3D mesh models by placing 12 centroid pointing virtual cameras around the mesh every 30 degrees with an elevation of 30 degrees from the ground plane, which is the first camera setup in [20]. In our experiments we also include additional feature types beyond the grey scale (appearance) images, to encode surface geometry. Surface normals are computed at the vertices of each 3D mesh model and then bilinearly interpolated to infer values on their faces. For depth, we directly apply the normalized depth values. We then render these 3D mesh features using the multi-view representations in [20] but with these new features. We linearly map the surface normal vector field $(n_x, n_y, n_z)$ where $n_i \in [-1, 1]$ and $i = x, y, z$, to a color coding space $(C_x, C_y, C_z)$ where $C_i \in [0, 255]$ for all $i = x, y, z$, to ensure that these features are similar in magnitude to the intensity values in the grey scale appearance images. Examples of the computed rendered feature types are shown in Figure (5).

## 3.1 Training and Testing Procedure

We explore two training approaches for our system in Figure (1). In *Fast training* we use the Imagenet pretrained VGG-m network in [5] to compute the relu7 feature vectors of each view of each training 3D mesh, by forward passing the 2D views into it. We then forward pass the relu7 vectors to our recurrent clustering and pooling layer. In our experiments we use a universal clustering scheme at each recurrence for all training and testing objects, by

[2] Qi *et al.* [18] used this entire train/test split and reported average class accuracy on the 2,468 test objects. Su *et al.* [20] used a subset of train/test split comprising the first 80 objects in each category in the train folder (or all objects if there are fewer than 80) and the first 20 objects of each category in the test folder, respectively.

| Cluster Pooling Structure | Fast | End-to-End |
|---|---|---|
| mvcnn (f-max) | 89.8 | 91.5 |
| DS-avg-f-max | 90.4 | 91.1 |
| DS-max-f-avg | 91.2 | 91.3 |
| (DS-alt)-f-max | 91.9 | 92.2 |

Table 1: Comparison of different pooling structures and training types.

| RGB | Depth | Surf | Accuracy |
|---|---|---|---|
| ✓ | ✗ | ✗ | 91.9 |
| ✓ | ✓ | ✗ | 92.9 |
| ✓ | ✗ | ✓ | 92.1 |
| ✗ | ✓ | ✓ | 92.9 |
| ✓ | ✓ | ✓ | $\boxed{93.3}$ |

Table 2: The benefits of additional feature modalities (see text).

averaging over the affinity matrices of all training objects. Ideally category-specific clustering is preferred for better category level recognition accuracy, but we do not have access to labels at test time. The universal clustering scheme is computationally efficient and the consistency it grants helps improve recognition accuracy. We record each recurrence's universal clustering scheme w.r.t training objects to formulate a recurrent clustering hierarchy for end-to-end training. After the full stride pooling layer, we have a single fused relu7 feature vector on which an SVM classifier is trained. At test time, we follow the same routine for a test object and apply the clustering scheme we used during training. The trained SVM classifier is then applied to predict its object category label. We note that there is no CNN training at all since we applied an Imagenet pretrained VGG-m network with no fine-tuning. In *End-to-end training* we directly feed in rendered 2D maps of training objects to the unified network in Figure (1) to perform the forward and backward passes in an end-to-end manner, using the recorded recurrent clustering hierarchy. The weights for the VGG-m network's layers before and after the recurrent clustering and pooling layer are jointly learned during training time. At test time, we send a test object's rendered 2D maps to the network to acquire its predicted object category label.

## 3.2 Model Ablation Study

**Recurrent Clustering and Pooling Structure** In our evaluation "f-max" stands for a full stride channel-wise max pooling, "ds-avg" stands for one recurrence of the clustering and pooling layer with within-cluster average pooling and "(ds-x)" stands for recurrent clustering and pooling until the clusters are stable. We examine the benefits of both recurrence and pooling with 3 variations: 1) ds-avg-f-max, 2) ds-max-f-avg, which uses only one phase of clustering and pooling but with different pooling operations and 3) (ds-alt)-f-max which uses recurrent clustering while alternating max and average pooling. Table (1) shows the results of these variations, together with the baseline "f-max" used by the MVCNN method [20]. Recurrent clustering and pooling is indeed better than a non-recurrent version and alternating max-avg within-cluster pooling followed with a full stride max pooling performs better than the other variants. We further note that end-to-end training performs better than non-end-to-end fast training.

**Additional Feature Types** We now explore the benefit of additional feature modalities using "(ds-alt)-f-max" clustering and pooling. We run experiments using the fast training scheme introduced in section (3.1). The results in Table (2) show that even without fine tuning, the network pretrained on pure appearance images can be generalized to handle different feature modalities. The additional feature types significantly boost the recognition accuracy of our recurrent cluster and pooling structure with a combination of appearance, depth and surface normals giving the best test set recognition accuracy of 93.3% with no CNN training.

| Cluster Pooling Structure | End-to-End? | 3 Views | 6 Views | 12 Views | 24 Views |
|---|---|---|---|---|---|
| mvcnn | ✗ | 90.76 | 91.16 | 89.8 | 91.93 |
| mvcnn | ✓ | 91.33 | 92.01 | 91.49 | 92.42 |
| Ours | ✗ | 91.45 | 91.61 | 91.89 | 92.70 |
| Ours | ✓ | 92.1 | 92.22 | 92.18 | 93.0 |

Table 3: Effect of the number of views.

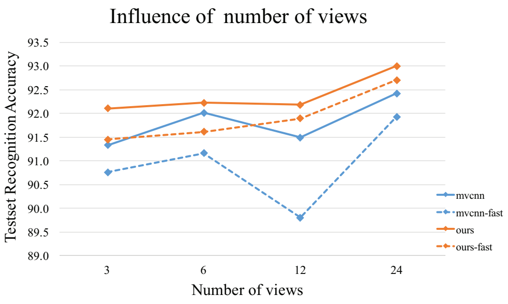

Figure 6: Effect of the number of views.

**Effects from number of Views** We evaluate the effect of the number of views on our recurrent clustering and pooling CNN and on MVCNN [20] in Table (3) and Figure (6). Our method is consistently better than the MVCNN approach, with a steady improvement of recognition accuracy as the number of views increases. We further note that for MVCNN there is an evident drop in performance when increasing from 6 views to 12 views, illustrating a potential drawback of its single full stride max pooling strategy.

## 3.3 Comparison with the present State-of-the-art

We now compare our method against the state-of-the-art view based 3D object recognition approaches [9, 18, 20] in Table (4). Two accuracies groups are reported: one for the subset used in MVCNN [20] and one for the full set used in [9, 18].[3] For Johns *et al.* [9] and Qi *et al.* [18], we quote the recognition accuracies reported in their papers. For MVCNN [20], we quote their reported subset results but reproduce experimental results on the full set. In [9, 18, 20], two types of view sampling/selection strategies are applied, as mentioned in Table (4).[4]

In addition to recognition accuracy, we also consider training time consumption. We measure the training time cost per epoch by $\mathcal{C} = \Phi\mathcal{N}_v\mathcal{N}_c$ where $\mathcal{N}_v$ denotes the number of rendered views, $\mathcal{N}_c$ denotes the number of VGG-m like CNNs trained and $\Phi$ stands for, for a single CNN, the *unit training cost* in terms of computational complexity for one epoch over a given 3D dataset when only one view is rendered per object. We denote the computational complexity for our fast training scheme, which consists of 1 epoch of forward passing and SVM training, as $\varepsilon$, since this is less costly than $\Phi$. We estimate that $\varepsilon < 0.5\Phi$.

The results in Table (4) show that our recurrent clustering and pooling CNN out performs the state-of-the-art methods in terms of recognition accuracy, achieving a 93.8 % full test set accuracy on ModelNet40 and 92.8% on the subset. When fast (non end-to-end) training is applied, the method still achieves state-of-the-art recognition accuracies of 93.3% and 92.1% with a greatly reduced training cost. Our results presently rank second on the ModelNet40

[3]Judging by the description in [9] "ModelNet10, containing 10 object categories with 4,905 unique objects, and ModelNet40, containing 40 object categories and 12,311 unique objects, both with a testing-training split." we assume they used the entire train/test split. Note that 9, 843 training and 2, 468 test models result in a total of 12,311 objects in the dataset.

[4] The term "30° elevation" means only views at an elevation of 30° above the ground plane, constrained to rotations about a gravity vector, are selected. The term "Uniform" means all uniformly sampled view locations on the view sphere. More specifically in Johns *et al.* [9] where a CNN based next-best-view prediction is applied for view point selection, the authors select the best 12 over 144 views to perform a combined-voting style classification. Therefore their CNNs are trained on a view base of 144 rendered views per object.

| Method | View Selection | # Views | Feature Types | base CNNs | Training Cost per Epoch | Subset Accuracy | Fullset Accuracy |
|---|---|---|---|---|---|---|---|
| Pairwise [9] | 30° | best 12 of 144 | RGB | 2x vgg-m | 288Φ | n/a | 90.7 |
| | uniform | best 12 of 144 | RGB + Depth | 4x vgg-m | 576Φ | n/a | 92.0 |
| Qi-MVCNN [18] | uniform | 20 | RGB + Sph-30 + Sph-60 | 3x alexnet | $60\phi = 5\Phi$ | n/a | 91.4 |
| Su-MVCNN [20] | 30° | 12 | RGB | 1x vgg-m | 12Φ | 89.9 | 91.5 |
| Ours-Fast | 30° | 12 | RGB | 1x vgg-m | $\varepsilon < 0.5\Phi$ | 90.4 | 91.9 |
| | 30° | 12 | RGB + Depth + Surf | 1x vgg-m | $3\varepsilon < 1.5\Phi$ | 92.1 | 93.3 |
| Ours-End-To-End | 30° | 12 | RGB | 1x vgg-m | 12Φ | 91.5 | 92.2 |
| | 30° | 12 | RGB + Depth + Surf | 3x vgg-m | 36Φ | 92.8 | 93.8 |

Table 4: A comparison against state-of-the-art view based 3D object recognition methods. For additional feature types, we used the same acronyms as is in Section (3.2). For Qi *et al.* [18], Sph-30 and Sph-60 stand for 30 and 60 uniform samples on a sphere. The training cost per epoch provides an estimate of the CNN training time consumption for each method. In [9], 2 CNNs need to be trained per feature modality, making it more costly. $\phi$ denotes the unit cost for alexnet where $\Phi \approx 12\phi$, as observed in our experiments.

Benchmark [1].[5] The top performer is the method of Brock *et al.* [3], which is a voxel-based approach with 3D representation training, focused on designing network models. In contrast, our method is a 2D view-based approach which exploits strategies for view feature aggregation. As rightly observed by a reviewer of this article, the 95.54% accuracy on ModelNet40 achieved in [3] is accomplished with an ensemble of 5 Voxception-ResNet [7] (VRN) architecture (45 layers each) and 1 Inception [21] like architecture. Training each model from the ensemble takes 6 days on a Titan X. In our approach, fine-tuning a pretrained VGG-M model after inserting our recurrent layer takes 20 hours on a Tesla K40 (rendering 12 views for ModelNet40). When only one VRN is used instead of the ensemble, Brock *et al.* achieve 91.33% on ModelNet40, while we achieve 92.2% using only RGB features.

## 4 Conclusion

The recurrent clustering and pooling layer introduced in this paper aims to aggregate multi-view features in a way that provides more discriminative power for 3D object recognition. Experiments on the ModelNet40 benchmark demonstrate that the use of this layer in a standard pretrained network achieves state of the art object category level recognition results. Further, at the cost of sacrificing end-to-end training, it is possible to greatly speed up computation with a negligible loss in multi-view recognition accuracy. We therefore anticipate that the application of a recurrent clustering and pooling layer will find value in 3D computer vision systems in real world environments, where both performance and computational cost have to be considered.

**Acknowledgments** We are grateful to the Natural Sciences and Engineering Research Council of Canada for funding, to Stavros Tsogkas and Sven Dickinson for their helpful comments, and to the reviewers whose constructive feedback greatly improved this article.

[5]As of July 17th, 2017.

## References

[1] The Princeton ModelNet Dataset. http://modelnet.cs.princeton.edu/. Online; accessed: July 2017.

[2] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. Gift: A real-time and scalable 3D shape search engine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2016.

[4] Samuel Rota Bulò and Marcello Pelillo. Dominant-set clustering: A review. *European Journal of Operational Research*, 2017.

[5] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[6] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3D model retrieval. In *Computer Graphics Forum*, volume 22, pages 223–232. Wiley Online Library, 2003.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Berthold Klaus Paul Horn. Extended Gaussian images. *Proceedings of the IEEE*, 72 (12):1671–1686, 1984.

[9] Edward Johns, Stefan Leutenegger, and Andrew J. Davison. Pairwise Decomposition of Image Sequences for Active Multi-View Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proceedings of the Symposium on Geometry Processing*, 2003.

[11] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3D SURF for robust three dimensional classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.

[12] David G Lowe. Local feature view clustering for 3D object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[13] Diego Macrini, Ali Shokoufandeh, Sven Dickinson, Kaleem Siddiqi, and Steven Zucker. View-based 3-D object recognition using shock graphs. In *Proceedings of the IEEE International Conference on Pattern Recognition*, 2002.

[14] Daniel Maturana and Sebastian Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015.

[15] Hiroshi Murase and Shree K Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[16] Massimiliano Pavan and Marcello Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[17] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1), 2007.

[18] Charles R. Qi, Hao Su, Matthias Niessner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and Multi-View CNNs for Object Classification on 3D Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3D shape surfaces using geometry images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[20] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[21] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[22] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[23] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.