3D_DEN: Open-ended 3D Object Recognition using Dynamically Expandable Networks

Sudhakaran Jain and Hamidreza Kasaei

Abstract—Service robots, in general, have to work independently and adapt to the dynamic changes in the environment. One important aspect in such scenarios is to continually learn to recognize new objects when they become available. This combines two main research problems namely continual learning and 3D object recognition. Most of the existing research approaches include the use of deep Convolutional Neural Networks (CNNs) focusing on image datasets. A modified approach might be needed for continually learning 3D objects. A major concern in using CNNs is the problem of catastrophic forgetting when a model tries to learn new data. In spite of various recent proposed solutions to mitigate this problem, there still exist a few side-effects (such as time/computational complexity) of such solutions. We propose a model capable of learning 3D objects in an open-ended fashion by employing deep transfer learning-based approach combined with dynamically expandable layers, which also makes sure that these side-effects are minimized to a great extent. We show that this model sets a new state-of-the-art standard not only with regards to accuracy but also for computational complexity.

Index Terms—Continual learning, open-ended learning, 3D object recognition, catastrophic forgetting, dynamic network architectures.

I. INTRODUCTION

SERVICE robots are generally used in domestic human-centric environments where they have to work independently, adapt to the dynamic changes in the environment and efficiently manipulate objects to achieve some specific goal assigned to it. One of the main concerns in such scenarios is learning about new object categories in an open-ended fashion using a limited number of training instances of the object categories. Generally, such a learning process is also called open-ended/continual or lifelong learning [1] that poses a big challenge in the field of Artificial General Intelligence. Hence, it can be seen as a vital ability required for a robot to perform its day today work.

Even though a lot of attention has been given to continually learning scenarios, researchers have mainly focused to test their proposed model on 2D image datasets, usually MNIST [2]. On the other hand, very less focus has been given to the problem of continuous learning on 3D objects. Inspite of the presence of various different papers to improve 3D object classification, open-ended recognition of 3D objects still lacks focus and has a lot of research scope for improvement.

In recent times, the research community has been giving much attention to deep Convolutional Neural Networks (CNNs) for continual learning tasks on image datasets. When there is a fixed set of object categories and large number of instances for each category which are sufficiently similar to the test images, CNN-based approaches yield good results. But in open-ended scenarios where the model needs to learn newer categories using very few training examples presented over a period of time, these assumptions may not be applicable. CNNs are considered to be incremental in nature but do not support open-ended learning as latter demands the network topology to be restructured every time when a new category comes into play. In simple terms, in incremental learning, the target set of classes is predefined in the beginning itself and representation of these classes is improved over time, where as in open-ended learning the set of classes grow continuously. Moreover, CNNs demand lot of training data to yield better performance output which is their important limitation. Catastrophic forgetting [3] is another major limitation of CNNs. This problem can be defined as forgetting the older tasks while the model is being trained further on newer tasks. It usually happens due to modifying previously learned weights upon start of training the network for newer tasks. In addition, learning to recognize 3D objects adds more complications to the problem.

We try to overcome these issues by proposing a novel deep learning dynamic architectural design and methods for training them to improve open-ended learning for 3D object recognition. Figure [1] shows how our model looks like. After conducting various experiments on a dataset with different extensive settings, we prove that our model performs better than the current state-of-the-art [4], not only in terms of accuracy, but also reduces the computational complexity to its one-third.

The remainder of this paper is organized in the following way. Section [II] reviews related work of continual learning and 3D object recognition. Next, the detailed methodologies of our proposed model namely 3D_DEN are explained in Section [III]. Section [IV] is about the experimental setup, results and discussion where we explain in-detail about the performance of our model compared to existing state-of-the-art. This is followed by conclusions and final remarks on our direction of future work in the Section [V].

II. RELATED WORK

In this study we are looking at the problem of open-ended 3D object recognition, which in itself has two sub-problems, namely continual learning and 3D object recognition. Both of these have a deep history of research in machine learning, computer vision and robotics, resulting in many different approaches. In this section, we review a few recent efforts.
Several approaches have been proposed to tackle continual learning problem, involving regularization techniques for weight changes, dynamic architectural networks and memory replay \[5\]. Memory replay can further be subdivided into rehearsal and generative replay \[6\].

Regularization approaches alleviate catastrophic forgetting by imposing constraints on the update of the neural weights. However, the use of simple regularization like L2-regularizer prevents the model from learning new knowledge for the new tasks, which results in sub-optimal performances on later tasks. To overcome this limitation, a method called Elastic Weight Consolidation (EWC) \[7\] was proposed that regularizes the model parameter at each step with the model parameter at previous iteration via the Fisher information matrix for the current task, which enables to find a suitable solution for both tasks. Few works explored neural networks that can dynamically increase its capacity during training of multi-class classification tasks. One such impressive work was made in the paper \[8\], where they proposed an idea to increase the size of the network for each new task. The inference for new task is made using the existing nodes and newly added nodes. But, the main drawback was that the model always required more and more memory with the incoming new tasks. As the tasks only used nodes added till their training stage during inference, only a part of the network was used at inference time. This indeed indicated a very inefficient use of memory. J. Yoon et al. \[9\] recently proposed a method to incrementally train a network for multi-class classification, where the network not only grows in capacity, but forms a hierarchical structure as new classes arrive at the model. The model, however, grows and branches only to the topmost layers. Referring to these approaches, a new efficient model has been proposed in the paper \[10\] which the authors called ‘Dynamically Expandable Neural Network’ that combines both regularization and dynamic architectural methods. The authors have proved that through Selective Re-training, Dynamic Network Expansion, and Network Splitting, better results can be achieved not only concerning accuracy but also in terms of computational complexity. The paper has made sure of efficient memory usage without compromising the overall accuracy. Their model was tested on several datasets, including 2D image datasets with smaller dimensions like MNIST \[2\] and its variations. One major disadvantage of using these above dynamic architectures is that, they assume each task to be a multi-class classification problem. Due to this, during inference, we have to provide a task ID to select the sub-network from the final model which may then be used to provide output. Not all continual learning problems will have this scenario, such as our problem at hand, where each new task is a new incoming category to be learned. Having said that, these papers form an excellent basis for our work.

There has been few interesting approaches specific to tackle the problem of open-ended object recognition. One such method is the instance-based approach proposed in the paper \[11\]. Such an approach demands lesser training time and is very useful when you have very less instances of training data. According to the algorithm discussed in this paper, the model initially considers each new object as a new class and later clusters them together rather than assigning labels to each sample. Even though this proposed model is for unsupervised object recognition in open real-world ergocentric scenarios, the instance-based approach discussed here can also be considered for supervised learning. One such similar instance-based learning model called OrthographicNet is proposed in the paper \[4\]. The interactive learning model generates scale and rotation invariant global features of a given object, so that same or similar objects can be recognized from different perspectives. Firstly, three orthographic projection views of an object are obtained. These views are decided based on the principle component axis obtained from the eigenvalue decomposition of the object point-cloud. Each view is fed to a Convolutional Neural Network (CNN), pre-trained on ImageNet \[12\], to obtain view-wise features of the object. Finally, the obtained CNN features are merged using an element-wise max-pooling function to generate a global representations for the given 3D objects. During supervised training, when an object from a new category arrives, an instance based global representation is created and initialized with the set of views of the target object. This global representation for the category is updated whenever more objects from this category arrives. In short, we can say that the category is represented by set of known
instances’ features. During inference, the feature representation of the unknown object is compared with the set of global representations of all categories using distance similarity measure. This model is considered to be the state-of-the-art for highest accuracy in open-ended 3D object recognition scenarios tested on Princeton ModelNet and Washington RGB-D Object datasets \[13\], \[14\]. But the main drawback of such instance-based learning model is that it may only be able to learn limited number of categories. Poor performance for categories with similar features is another concern. This may indeed be a big problem when number of known categories become very high. Moreover, it also occupies memory to store global representations of all categories along with their object views. This problem is mainly due to usage of fixed architectural model with constant number of neurons for feature extraction and representation.

Another important paper \[15\] discussed a new model named PointNet to recognize 3D objects by taking the input of objects in the form of point-cloud. Although this model cannot deal with open-ended learning scenarios, it serves as a base architecture for many other research papers on open-ended learning. This network has three key modules: the max pooling layer as a symmetric function to aggregate information from all the points, a local and global information combination structure, and two joint alignment networks that align both input points and point features. Symmetry function is responsible to make the model invariant to input permutations and to aggregate the information from each point. The joint alignment network applies affine transformations to the input so that the model is invariant to certain geometric transformations during semantic labeling. These local features are sent to max pooling layer to obtain global features. Finally, a linear function is applied over it to get the corresponding softmax output. The paper \[16\] came up with modifications to PointNet and called it as L3DOC model. The core idea of the proposed model is to factorize PointNet in a perspective of lifelong/open-ended learning, while capturing and storing the shared point-knowledge in a perspective of layer-wise tensor factorization architecture. To further transfer the task-specific knowledge from previous tasks to the new coming classification task, a memory attention mechanism is proposed to connect the current task with relevant previously tasks, which can effectively prevent catastrophic forgetting via soft-transferring previous knowledge. This model is considered to be the state-of-the-art for computational complexity on Princeton ModelNet datasets. During their evaluation they had compared their model results with \[10\], but however, the input given to it was only the 2D part from the 3D input image. Hence, the results obtained from \[10\] was much worse compared to theirs.

Our work differs from other similar above mentioned papers \[8\], \[10\] in various aspects. First key difference is the type of tasks learned by the model. As stated earlier, our work tries to address the problem where training a new task means to learn and classify a new category with existing known ones, whereas these papers assume every new task to be a multi-class classification that are independent of each other. Secondly, compared to \[10\] which re-trains all the parameters in the network, we only re-train the newly added nodes during Dynamic Expansion to prevent catastrophic forgetting. Hence, we do not need a separate method named Network Splitting discussed in \[10\] that avoids any drastic shift in old parameter values. Lastly, we employ deep transfer learning approach by using a pre-trained network in our architecture for feature extraction instead of training deep convolution layers from scratch.

III. Methodology

As our work aims to solve the above discussed problems in open-ended 3D object recognition using a dynamic network architecture, the model should ideally learn new object categories in open-ended manner to an unbounded extent, and at the same time retain the ability to recognize previously known objects.

Open-ended learning concerns all three issues namely time, computation and storage complexity. In addition, dynamic architectures which we will be using, are known to have very high time/computational cost. However, as our problem at hand is specific to domestic robots, we can assume that the storage complexity to be less significant. Thus, our focus is to reduce the computational/time complexity of our dynamic architectural model, with a little or no compromise in overall accuracy. In other words, we aim to make a model which is compact too.

Similar to OrthographicNet, our model takes three projection view images for each 3D object instance during training. But, rather than having different CNNs for every view image, these view images are converted to grey-scale single channel images and then combined to form a 3-channel image which becomes our input. Our model design is a pre-trained network (without its output layer) attached to two Dynamically Expandable Network (DEN) layers which are Fully-Connected (FC). We believe that the usage of a pre-trained network and having only one CNN will reduce the computational and time complexity to a great extent. Having said that, these FC layers will be responsible for maintaining better overall accuracy by learning new distinguishing features for upcoming newer categories during training. Throughout the incremental training, weights of the network are kept sparse using L1-Regularization. Figure 1 shows the architecture of our model. We introduce modern supervised training methods, few of them similar to the ones discussed in the paper \[10\] with modifications as explained in following subsections. The learning procedure consists of a series of tasks. The initial task \((t = 1)\) is a binary classification of two categories trained according to below Equation 1

\[
\min_{W^{t=1}} L \left( W^{t=1}; D_{t=1} \right) + \mu \sum_{l=1}^{L} \left\| W^{t=1}_l \right\|_1
\]

where \(L\) is task specific loss function, \(1 \leq l \leq L\) represents the \(l_{th}\) layer in the model. \(W^{t=1}_l\) represent the weight matrix of the layer \(l\) and \(\mu\) is the parameter for L1-Regularization. So now onwards, the rest sequence of upcoming tasks represent addition of new unknown categories to be learnt with respect to already known ones.
Data Sampling Procedure: Ideally, due to the problem of catastrophic forgetting, gradient based models have to retrained again from scratch when newer data becomes available. In our proposed model we try to solve this problem by using the simplest method which is sampled rehearsal of older data along with new data during training each task. This procedure is carried out by sampling the old data for same number of instances that the new category data has. We keep equal sampling probability for all the categories in old data and also maintain a minimum threshold $\rho$, on number of instances to be drawn from an old category for each new task. Algorithm 1 describes our sampling process. Although, we reduced the training data by using sampling, we still have to store the whole dataset during our continual training process. An ideal solution for generic open-ended learning should eradicate this storage problem completely. We leave this problem for future work.

Selective Retraining: When a new task ($t>1$) arrives, firstly, a new output node (for new category) is added and the weights between top-most hidden layer to the output layer are retrained using the Equation 2. Then, the nodes in this hidden layer that have non-zero weights to output of new category are selected. Further, we perform a breadth-first search among rest of the sparse FC-layers(DEN) to select all the corresponding nodes which are connected to the previously selected nodes. These nodes are the ones affected by the new task. Hence, we retrain only them separately as a sub-network using the Equation 3. Please note that we also make sure there is no drastic change in their weights by imposing a regularization constraint in the loss function. Figure 2(a) and Algorithm 2 illustrate the working of this method.

![Fig. 2. Depiction of training methods used for our model: (a) Selective Retraining: First, the hidden-to-output nodes are selected, then we perform breadth-first search using the selected nodes to identify rest of the nodes that are responsible for new output. Selected sub-network is represented by red color; (b) Dynamic Expansion: New nodes are added to DEN layers and trained, while rest of the nodes are kept constant. In the end, new nodes that were deemed useless are removed. Here, newly added nodes are represented by red color.](image)

**Algorithm 1**: Data Sampling

| Input: Dataset $D = (D_1, \ldots, D_T)$, task $t (>1)$ |
| Output: $D_t$ |
| $s = \text{len}(D_t)/(t-1)$ |
| if $s < \rho$, then |
| $s = \rho$ |
| for $i = 1, \ldots, t-1$ do |
| Sample $s$ elements from $D_i$ and add to $D_t$ |
| end |
| Add $D_t$ to $D_i$ |

**Algorithm 2**: Selective Retraining

| Input: $D_t$, $W^{t-1}$ |
| Output: $W^t$ |
| Initialize $l \leftarrow L-1, S = \{o_t\}$ |
| Solve Eq. 3 to obtain $W^{l,t}_{L,l}$ |
| Add neuron $i$ to $S$ if the weight between $i$ and $o_t$ in $W^{l,t}_{L,l}$ is not zero |
| for $l = L-1, \ldots, 1$ do |
| Add neuron $i$ to $S$ if there exists some neuron $j \in S$ such that $W^{l-1}_{l,i,j} \neq 0$ |
| end |

**IV. Experimental Results**

Three types of experiments were carried out to evaluate the proposed approach. First, we present a systematic open-ended
evaluation of the proposed 3D_DEN approach in the context of object recognition task. Second, we perform an offline evaluation using a similar architecture of 3D_DEN, but with fixed size of FC-layers. Lastly, we also performed a real-robot demonstration in the context of serve_a_drink scenario to show the strength of the proposed approach concerning real-time performance. In the following subsection, for each type of experiments we first describe the experimental setup and then discuss the obtained results.

We used Princeton ModelNet40 dataset [13] for performing all our experiments. It contains 12,311 CAD models from 40 different object categories, which were divided into 9,843 training samples and 2,468 testing samples. This dataset in image view format was obtained from the paper [19]. Out of 12 views for each object we selected three best representatives manually by inspecting them with respect to their angular positions, and utilized them for training. This indeed reduces computational complexity making the model more compact. Each view image was converted to single channel(greyscale) with dimensions 128 × 128.

A. Open-Ended Training Procedure

In this round of experiment, we evaluated three different pre-trained networks to find out the best architecture for our 3D_DEN model in terms of accuracy and computational cost. Two of the networks are popular feature extractors namely VGG16 [20] and MobileNet-v2 [21], both pre-trained on ImageNet [12]. The third network is a custom model that is built based on the architecture of MobileNet-v2, that is trained on ModelNet10 dataset [13] and later serving as pre-trained network during experiment. The main training procedure is done in a supervised manner by combining three grey-scale orthographic view images in the dataset and feeding them as input to the model, while the label of the object is fed as output. The model learns these input representations to match the outputs using the conventional back-propagation technique [22]. The intention of considering various pre-trained architectures was to find if there are any notable difference in feature extraction when the input for pre-training are not RGB images but rather three single channel view images. Table I gives more insights on these networks’ properties.

Since the order of introducing tasks may influence the performance, we performed 10 trials for each of the networks. In each trial the model was trained from scratch on the ModelNet40 dataset in an open-ended fashion and overall accuracy was noted down. Apparently, every trial consisted of 39 tasks, i.e., one less than total number categories because initial task is binary classification. The order in which the categories appeared for training was random in every trial. Additionally, we also keep a check on test accuracy during the trials. Algorithm 3 describes this process in a step-wise manner. After the completion of trials, measurements based on different metrics were computed. It should be noted that the classical form of evaluation which considers the accuracy as the main metric cannot be used for open-ended performance evaluation. We therefore consider three main metrics introduced recently to compare and discuss the performance of our models [23][3]. These metrics include: (i) Global Classification Accuracy (GCA), which describes the average of final accuracies for all trials; (ii) Average Protocol Accuracy (APA), which describes the average accuracy over all tasks; (iii) Average number of Learned Categories (ALC) during each trial. Unlike OrthographicNet, we do not stop a trial when test accuracy falls below a specific threshold, rather only when all tasks are trained. This was done to thoroughly check the ability of each model to learn newer tasks and decide which performs the best. Thus, ALC was 40 for all our models. Additionally, to evaluate the computational complexity of models, we consider the total number of parameters in the model.

Results: During our extensive set of open-ended training with different settings in the experiment, we compared the performances of our 3D_DEN models with OrthographicNet (current state-of-the-art). Table II shows the results. We can clearly see that that our 3D_DEN model with Mobilenetv2 as pretrained network performed the best in terms of both accuracies namely GCA and APA, making it the new state-of-the-art model for open-ended evaluation on this dataset. Whereas, OrthographicNet was able to learn only 38 categories after

| Table I |
|---|
| Properties of Feature Extractor Networks used in Experiment |
| Model | Feature Length | Size | Depth |
| VGG16 | 4 × 4 × 512 | 528 MB | 23 |
| MobileNet-v2 | 1280 | 14 MB | 88 |

Algorithm 3: Dynamic Expansion

Input: \( D_t, \tau \)
Output: \( W^t \)
Add \( k \) neurons \( h^N \) in all layers
Solve for Eq. 4 for all layers
for \( t = 1, \ldots, L \) do
    | Remove useless neurons in \( h^N \)
end

Algorithm 4: Training Procedure

Input: Dataset \( D = (D_1, \ldots, D_T) \)
Output: \( W^t \)
for \( t = 1, \ldots, T \) do
    if \( t = 1 \) then
        | Train the network weights \( W^1 \) using Eq. 2
    else
        \( D_t = \) DataSampling\( (D, t) \)
        \( W^t = \) SelectiveRetraining\( (W^{t-1}) \)
        if \( A_t < \tau \) then
            | \( W^t = \) DynamicExpansion\( (W^t) \)
        end
    end
end
Fig. 3. Summary of open-ended evaluation: (top) shows the timeline of accuracies for all three models. We can notice 3D_DEN_MobileNet outperforms the rest; (middle) shows training time(approx) needed by our models while learning new tasks. Note the difference between 3D_DEN_VGG16 and rest of them, which is due to the size of the pre-trained network used; (bottom) shows the increment in number of neurons in DEN layers for all models. With careful inspection, we can notice that the steady increase starts only after a specific threshold for every model. Which it had to be stopped as it reached the saturation point. In particular, our model achieved 80.71% as GCA and 84.16% as APA on the test data while using only one third of parameters of OrthographicNet. Thus, it not only gives better results but reduced the computational complexity to a great extent.

| Model                  | GCA(%) | APA(%) | ALC | #Parameters       |
|------------------------|--------|--------|-----|-------------------|
| OrthographicNet [4]    | 66.54  | 74.70  | 38  | $3 \times 3.53 = 10.59$ M |
| 3D_DEN (ours-VGG16)   | 77.60  | 83.92  | 40  | 138.35 M         |
| 3D_DEN (ours-MobileNet)| 80.71  | 84.16  | 40  | 3.53 M           |
| 3D_DEN (ours-Custom)  | 56.93  | 68.21  | 40  | 3.53 M           |

One other major observation here is the slope of the top plot, which looks a bit steep towards the start but later this steepness decreases with upcoming tasks. With further inspection, we can see that each model has a threshold value in x-axis after which the steepness decreases more significantly. To interpret the reason behind this phenomenon, lets consider 3D_DEN_VGG16 and 3D_DEN_MobileNet for which this threshold is around task-15 in x-axis. By referring to bottom plot which describes about the increment of neurons during training, task-15 seems to be the point after which both these
models start to add neurons to its DEN layers. This indeed explains that using our second approach which is Dynamic Expansion, yields better results than selective retraining. Even though, both seem to suppress catastrophic forgetting, the latter seems to do the job more efficiently.

B. Offline Evaluation using Grid Search

Through this evaluation procedure, we try to find an approximate number of neurons in FC-layers and appropriate optimizer required for a model that can achieve best accuracy during offline training on our dataset. Our main intention was to compare these results with our best performing main model (3D_DEN_MobileNet) which is trained in open-ended fashion. Precisely, we will observe how the DEN layers of our main model differs from the optimal size of FC-layers in the model obtained here and also seek to find if there is any differences in optimizer used.

In this round of experiment, the model training was governed by Grid Search algorithm. Here, we performed a series of training, where the model was trained from scratch on the whole dataset in an offline fashion, i.e., on all 40 categories at once. Each training procedure had a different combination of FC-layer sizes and optimizer. The parameter range for layer size and optimizer used during the process along with the best configuration results obtained are shown in Table III. Also, the top three configuration with respect to accuracy is shown in figure 4.

![Fig. 4. Results of offline evaluation using grid search: num_units1 and num_units2 are the FC-layers of the model. The best obtained parameters are connected using the green-colored line. Whereas, the red and purple-colored lines show the parameters which achieved second best and third best results, respectively.](image)

**Results:** Table III and Figure 4 shows the results of offline evaluation, where we can see that the best size of FC-layers were 2048 and 512 (so total of 2560), yielding to best accuracy of 90.72%. Now, by comparing this to our 3D_DEN_MobileNet model (referring to Figure 3-bottom) to these results, we observe that our model used only around 1500 neurons in total for both DEN layers which indeed sounds good when we correlate them with their respective accuracies they achieved. Also, to our surprise, we observed that offline evaluation gave best accuracy using SGD optimizer, whereas open-ended evaluation models always gave best results using Adam optimizer.

C. Evaluation on Real-Time Robot

A real-robot experiment was carried out to check the performance of the proposed model in real-time. Precisely, the aim of this experiment was to see if the robot can recognize a set of table-top objects using 3D_DEN to accomplish a given task, i.e., serve_a_drink. In particular, the task here was to pour a drink from a bottle into cups present on the table. The experimental setup is depicted in Figure 5. It consists of a table, a Kinect sensor, a UR5e robotic-arm as the primary sensory-motor embodiment for perceiving and acting upon its environment. There are four instances of three object categories on the table: two cups, a bottle, and a vase object with flowers. This is a suitable set of objects for this test since similar instances to the selected objects exist in ModelNet40 dataset. Towards this goal, we integrated our trained model into a cognitive robotic system presented in [24] as a ROS service [25].

**Results:** To accomplish the serve_a_drink task, the robot should be able to detect the pose and recognize the label of all table-top objects. Afterward, it has to grasp the bottle object and transport it on top of each active cup and serve the drink. The robot should finally return to the initial pose. Towards this goal, the robot first segments

![Fig. 5. Our experimental setup for real-robot experiment consists of a Kinect sensor to perceive the environment, and a UR5e robot to act upon the environment. Throughout the serve_a_drink experiment, we use four instances of three object categories including bottle, cup, and plant. It should be noted that similar instances to the selected objects exist in the ModelNet40 dataset.](image)

| Grid Parameters | Range        | Best Parameters | Best Accuracy (%) |
|-----------------|--------------|-----------------|-------------------|
| FC-layer 1      | 512–2048     | 2048            | 90.72             |
| FC-layer 2      | 128–512      | 512             |                   |
| Optimizer       | SGD, Adam    | SGD             |                   |

**TABLE III**

**GRID SEARCH PARAMETERS FOR OFFLINE EVALUATION**
However, by seeing the trend of gradual decrease in accuracy throughout our open-ended training experiment, we can say the model has to be eventually stopped from learning after a point in order to maintain certain level of overall accuracy. With these drawback in mind, we can say that our model is not capable of learning tasks to an unbounded extent as we had wanted it to be. But, these observations will prove to be vital while extending our work in future.

Another problem is with the converting of view images to single channels and combining them. In spite of the fact that it reduced the computational complexity of our model to a great extent(also making it compact), it did incur a limitation to our approach, which is the inability of our model to learn color features of the 3D object. Moreover, as the model demands input in such a way, there is a small additional computational cost for pre-processing here. Having said that, if computational complexity becomes lesser concern in certain scenarios due to the advancements in hardware, then the performance of the model can be improved further by using more number of views than just three.

Lastly, lets come to the storage problem which our model does not address. Even though, our model uses data sampling to reduce the rehearsal training data, this still means that dataset has to be stored during whole learning process. In contrast to our sampling method, the training procedure of OrthographicNet is governed by a teaching protocol named simulated teacher which interacts with the learning agent (neural network model). This protocol uses three main functions: Teach-Ask-Correct. The main purpose of these functions are to teach, test and correct the model when it makes mistakes in recognizing a particular object category. Hence, the rehearsal happens here only when required and not on a fixed basis like in our approach. Nevertheless, in both the cases, the dataset always has to be stored in memory. We seek to find a solution to this problem in future.

V. Conclusion and Future Scope

In this paper, we proposed a deep learning based approach named 3D_DEN that makes use of dynamic architectural design to learn 3D objects in open-ended fashion. This approach not only achieves better results in terms of accuracy, but also proves to be very efficient in terms of computational complexity, which is indeed considered as a major concern while using dynamic architectures in general. While this model can be seen as new state-of-the-art benchmark, it should also be noted that this model paves a new path(by using dynamic architecture) for solving the problem of continual learning using 3-dimensional data in robotics domains. It can indeed also be considered in other domains that use 3D data, but ofcourse, with some improvements/changes to adapt to those domains.

Even though we did achieve satisfactory performance using our model, there were few areas in which we need to work on in future. As we discussed in above section, one major flaw is the control on dynamic increase in DEN layers. We seek to solve this issue by coming with a better optimization strategy to reduce the usage on neurons. Moreover, as of

---

1 A video demonstration is available at: https://youtu.be/tf4trRMvQ0Y
now, we are adding a constant number of neurons(hardcoded) during Dynamic Expansion. Deciding appropriate values for other hyper-parameters(and certain thresholds) is one more concern. In future, we will try to find a strategy which will decide these values using one more learning-based approach. Indeed, there exists a paper [26] which proposed a method using reinforcement learning to predict optimal values for these hyper-parameters. This seems be a good starting point for further improving our work.

As said earlier, storage and rehearsal of data is one other major concern. There has been few notable papers to eliminate the rehearsal strategies by making use of generative approaches [27], [28]. One such latest work that is also similar to our work (to some extent) is the paper [29] which seems to give promising results by combing various recent strategies to mitigate catastrophic forgetting. Another relevant brain-inspired implementation can be seen in paper [30], which neither uses generative approach nor attention mechanisms to address continual learning problem. In future, we seek to integrate one or more of these above strategies in our work.

REFERENCES

[1] S. Thrun, “A lifelong learning perspective for mobile robot control,” in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’94), vol. 1, 1994, pp. 23–30 vol.1.

[2] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[3] M. McCloskey and N. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” Psychology of Learning and Motivation - Advances in Research and Theory, vol. 24, no. C, pp. 109–165, 1 1989.

[4] H. Kasten, “OrthographicNet: A deep learning approach for 3D object recognition in open-ended domains,” CoRR, vol. abs/1902.03057, 2019. [Online]. Available: http://arxiv.org/abs/1902.03057

[5] G. I. Parisi, R. Kemker, J. L. Part, C. Kanau, and S. Wermter, “Continual lifelong learning with neural networks: A review,” CoRR, vol. abs/1802.07569, 2018. [Online]. Available: http://arxiv.org/abs/1802.07569

[6] T. Lesort, V. Lomonaco, A. Stojan, D. Maltoni, D. Filliat, and N. D. Rodríguez, “Continual learning for robotics,” CoRR, vol. abs/1907.00182, 2019. [Online]. Available: http://arxiv.org/abs/1907.00182

[7] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, “Overcoming catastrophic forgetting in neural networks,” CoRR, vol. abs/1612.00796, 2016. [Online]. Available: http://arxiv.org/abs/1612.00796

[8] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” CoRR, vol. abs/1606.04671, 2016. [Online]. Available: http://arxiv.org/abs/1606.04671

[9] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, “Error-driven incremental learning in deep convolutional neural network for large-scale image classification,” MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia, pp. 177–186, 11 2014.

[10] J. Lee, J. Yoon, E. Yang, and S. J. Hwang, “Lifelong learning with dynamically expandable networks,” CoRR, vol. abs/1708.01547, 2017. [Online]. Available: http://arxiv.org/abs/1708.01547

[11] L. Eruculiani, F. Giunchiglia, and A. Passerini, “Continual egocentric object recognition,” 2019. [Online]. Available: https://arxiv.org/abs/1912.05029

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in CVPR09, 2009.

[13] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, “3D shapenets for 2.5D object recognition and next-best-view prediction,” CoRR, vol. abs/1406.5670, 2014. [Online]. Available: http://arxiv.org/abs/1406.5670

[14] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view rgb-d object dataset,” 05 2011, pp. 1817–1824.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3D classification and segmentation,” 2016, cite arxiv:1610.09178, 2019. [Online]. Available: https://arxiv.org/abs/1610.05071

[16] Y. Liu, Y. Cong, and G. Sun, “LiD3C: Lifelong 3D object classification,” arxiv preprint arXiv1912.06135, 2019. [Online]. Available: https://arxiv.org/abs/1912.06135

[17] T. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, and S. J. Young, “A network-based end-to-end trainable task-oriented dialogue system,” in CoRR, vol. abs/1804.04562, 2016. [Online]. Available: http://arxiv.org/abs/1804.04562

[18] J. M. Alvarez and M. Salzmann, “Learning the number of neurons in deep networks,” CoRR, vol. abs/1611.06321, 2016. [Online]. Available: http://arxiv.org/abs/1611.06321

[19] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, “Multi-view convolutional neural networks for 3D shape recognition,” in Proc. ICCV, 2015.

[20] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in CoRR, vol. abs/1409.1556, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[21] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for 2.5D object recognition and next-best-view prediction,” arXiv preprint arXiv1912.06135, 2019. [Online]. Available: https://arxiv.org/abs/1912.06135

[22] S. H. Kasaei, N. Shafii, L. S. Lopes, and A. M. Tomé, “Interactive open-ended object affordance and grasp learning for robotic manipulation,” in 2019 IEEE/RSJ International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 3747–3753.
[25] Stanford Artificial Intelligence Laboratory et al., *Robotic Operating System*. [Online]. Available: [https://www.ros.org](https://www.ros.org)

[26] J. Xu and Z. Zhu, “Reinforced continual learning,” *CoRR*, vol. abs / 1805.12369, 2018. [Online]. Available: [http://arxiv.org/abs/1805.12369](http://arxiv.org/abs/1805.12369)

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: [http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf](http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf)

[28] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, “Adversarial autoencoders,” in *International Conference on Learning Representations*, 2016. [Online]. Available: [http://arxiv.org/abs/1511.05644](http://arxiv.org/abs/1511.05644)

[29] S. Ebrahimi, F. Meier, R. Calandra, T. Darrell, and M. Rohrbach, “Adversarial continual learning,” *arXiv preprint arXiv:2003.09553*, 2020. [Online]. Available: [https://arxiv.org/abs/2003.09553](https://arxiv.org/abs/2003.09553)

[30] X. Ji, J. Henriques, T. Hazelaars, and A. Vedaldi, “Automatic recall machines: Internal replay, continual learning and the brain,” *arXiv preprint arXiv:2006.12323*, 2020. [Online]. Available: [https://arxiv.org/abs/2006.12323](https://arxiv.org/abs/2006.12323)