Open-Ended Learning Approaches for 3D Object Recognition

1nd Sarandis Doulgeris
Rijksuniversiteit Groningen
S4928628
Serres, Greece
s.doulgeris@student.rug.nl

2st Pedro Rodriguez de Ledesma Jimenez
Rijksuniversiteit Groningen
S4745779)
Madrid, Spain
p.rodriguez.de.ledesma.jimenez@student.rug.nl

***Abstract*—Three-dimensional (3D) object recognition is a technique for identifying objects in images or point clouds. The goal of such techniques is to teach a computer to gain a level of understanding of what an image contains. We can use a variety of machine learning or deep learning approaches for object recognition. In our assignment, the object recognition is split into two parts. The first part is about implementing/optimizing offline 3D object recognition systems, which take an object view as input and produces the category label as output. The second part of this assignment is dedicated to testing your approach in an open-ended fashion. In this part, the number of categories is not predefined in advance, and the knowledge of agent/robot is increasing over time by interacting with a simulated teacher.**

I. Introduction

For a robot's functionality, 3D object recognition is a crucial phase. In order for it to interact with the user and the environment, the robot has to be very accurate in the recognition of its surroundings. The input of the object recognition stage is a result of several activities.

First a sensor (e.g. a camera) an image frame is extracted. Pre-processing techniques are applied in order to discard unwanted information and clutter. Afterwards, objects are extracted using clustering procedures. Object tracking is then used to know the location of the object in the the robot's "field of vision". Filtering techniques in the form of control theory systems are used to predict the next pose of the object.

The result is inserted in the Object Representation module. This paper work, starts from this module. Object-descriptors are used to give a suitable representation as input to the Object Recognition module. They are categorized in two parts:

- Hand-crafted feature
- Deep transfer learning

These categories will be further discussed later.

As a learning and adaptation approach we considered the instance-based learning, where an object category is represented and compared by a set of known instances. For the comparison we use a nearest neighbor classifier. This task can also be implemented using a probabilistic model (i.e. Bayesian Learning) where the object category with the highest likelihood is picked to label the object.

The optimization experiment is divided into two sections:

- Offline 3D object recognition
- Open-Ended learning scenario

In each approach we worked with a different dataset.

II. Offline 3D object recognition with hand-crafted object representation

The Offline evaluation needs the set of categories of the objects to be predefined. Then the classical ***"Train-then-Test"*** scheme if followed. The model's performance is evaluated on its ***accuracy*** in correctly classifying the object as well as in the training and testing time (***computational time***). For that a k-fold Cross Validation is used with $k = 10$. A dataset with 10 fixed categories is used as an input. Each category consists of various point clouds from different angles of the object/category. As we mentioned, this point cloud is fed in the object-descriptor to interpret the point cloud. A C++ code for 10-fold cross validation was provided. There are 4 available object descriptors (*GOOD, ESF, VFH, GRSD*). It was asked to pick two hand-crafted 3D object recognition algorithms out of them and optimize the model based them. GOOD and ESF descriptors were picked.

In the training phase each representation by the descriptor is saved in the memory. In the testing phase the new object representation is compared with the already saved representations. For this comparison a simple K-NN algorithm is used. This is also provided in the C++ code. Moreover different distance function for the K-NN algorithm are also provided.

A. GOOD: A global orthographic object descriptor

This descriptor makes use of the LRF (Local Reference Frame) and more specifically the orthographic projections of the object on the three orthogonal planes, *XoY, YoZ, XoZ* [1]. These three projections are segmented into a number of bins and the number of points in each bin is calculated. The bin grid is transformed into a histogram. The three histograms are then concatenated into one. First one is the histogram of the pane with the biggest entropy and second is the one with the highest variance of the two remaining as presented in Fig.1. The object-descriptor's performance was tested in its descriptiveness, scalability, robustness and efficiency. The results showed the only parameter of the algorithm, which is *number of bins* n has an effect of the descriptiveness efficiency and robustness. The number of bins should be optimal to reduce computational time, memory usage, and sensitivity to noise. Presented many categories, GOOD can categorize them

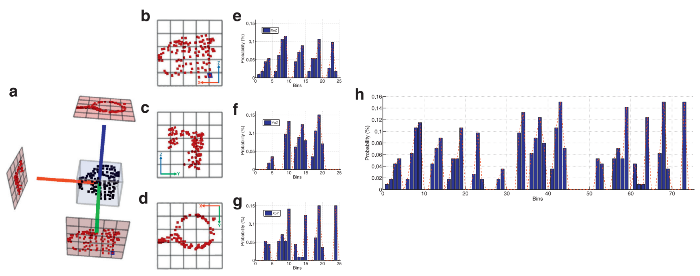


Fig. 1: An illustrative example of the producing a GOO shape description for a mug object: (a) The mug object and i bounding box, reference frame and three projected views; tl object's points are then projected onto three planes; therefoi XoZ (b), YoZ (c) and XoY projections (d) are create Each plane is segmented into bins and the number of poi falling into each bin is counted. Accordingly, three distribution matrices are obtained for the projections; afterwards, each distribution matrix is converted to a distribution vector, (i.e. (e), (f) and (g)) and two statistic features including entropy and variance are then calculated for each distribution vector; (h) the distribution vectors are consequently concatenated together using the statistics features, to form a single description for the given object. The ordering of the three distribution vectors is first by decreasing values of entropy. Afterwards the second and third vectors are sorted again by increasing values of variance [1]

maintaining high accuracy in estimations. Also it performs way better than rest in the presence of noise when the number of bins has a low value (e.g. *number_of_bins* = 5). It is also the most computational time efficient.

B. Ensemble of Shape Functions for 3D Object Classification

ESF (Ensemble of Shape Functions) is a global shape descriptor based on the three distinct shape functions describing distance, angle and area distributions on the surface of the partial point cloud [2]. It is able to cope with differences in sensor characteristics and thus enables robust real-time classification.

This descriptor is a unit of ten 64-bin sized histograms of shape function. The first three histograms are derived from the angle computed between two lines. Three points are selected randomly and the angle enclosed is encoded. This histogram is further divided into three distinct histograms representing the ON, Off and MIXED angles. This is done by taking the line opposing the angle and tracing it with the help of the 3D Bresenham algorithm in a voxel grid with side length of 64. This coarse voxel grid serves as an approximation of the real surface. This process increases the descriptiveness of the descriptor. To characterize the distribution of the voxels along the line, the authors added the ratio of line distance. This ratio is equal to zero if the line falls off the surface, equal to one if inside, and equal to the numbers of the voxels, along with the connection, if the line is mixed [3] (last histogram).

The next three histograms are derived from the area that three randomly selected points form and are also categorized in the same manner as ON, OFF, MIXED. The rest histogram are derived from the lines that are formed from two points again categorized as mentioned. An example of the process is shown in Fig.2 and a representation of a real object (banana) is shown in Fig.3

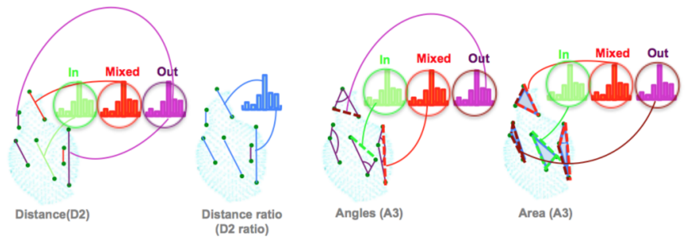


Fig. 2: The Ensemble of Shape Function (ESF). Illustration of how shape functions are computed for a point cloud. Left: point distance distributions. The green histogram represents points in, the red histogram points out, and the purple represents mixed points; Middle left: Distance Ratio; Middle right: angle distributions; Right: the area covered by triplets of sampled points

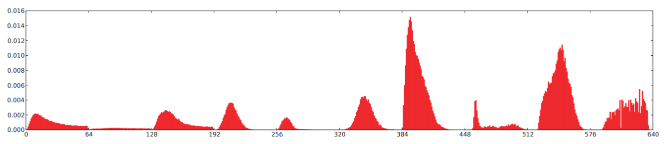


Fig. 3: The ESF descriptor calculated on a view of a banana with its ten 64-bin sub-histograms of shape functions. The shape functions from left to right: Angle (in,out,mixed), Area (in,out,mixed), Distance (in,out,mixed) and ratio of line distances.

The results from testing this descriptor showed that it provides great separability between categories and is an accurate algorithm. If the view of the object is different it can still recognize it correctly. It is also a fairly time efficient. One drawback for the ESF is that is very prone to noise fluctuations in the point cloud.

C. K-NN Algorithm and Distance Functions

Now that we have selected our descriptors and saved our representation in the perception memory, we have created our database which will be used for comparison when a new object is presented. For the comparison in our instance-based model we use the simplest algorithm for that procedure which is the *k-nearest neighbor* or simple *K-NN* algorithm. When a new histogram is presented, it is compared with all the histograms in the database. The label of the histogram with the lowest distance value is selected as output. The comparison is done for each bin sample and summed for the whole histogram.

In order to calculate the distance, we have to inform the algorithm about how it should calculate the distance. A variety of distance functions are provided.

D. Process of optimization

All of the code was provided. We selected an Object-descriptor and tested for various distance functions as well as different number of neighbours the accuracy of the model. In the end a trade-off between accuracy and computation time was made in order to select the best model.

For the GOOD, we tested also for different number_of_bins. The results are exported in a *.txt* file. We took that file and did a little reprocessing in order to transform it into a proper *.csv* file. That enabled us to make use of the *pandas* library in *Python*. With that we were able to extract key information fast and accurate to select the best parameters that optimized the model. The criteria for evaluation of the model:

- Instance-Accuracy (Ins-Acc). This is the accuracy of the model in each specific category.
- Average-Class-Accuracy (AVG-Class-Acc). This is the overall accuracy of the model.
- Computational time. This is the time that it takes for the model to train as well as test the dataset.

E. Results

In the GOOD approach a different set of parameters achieved the best Instance-Accuracy is the same and the Average-Class-Accuracy as can be seen in Fig.4.

	index	10	120
0	EXP	11	121
1	Obj.Disc.	GOOD	GOOD
2	#bins	15	30
3	K	1	1
4	Dist.Func.	motyka	bhattacharyya
5	Ins-Acc	0.9674	0.9674
6	Avg-Class-Acc	0.9597	0.9608
7	Time(s)	3.941	4.736

Fig. 4: Table representation of the best results for GOOD

A compromise was made because Instance-Accuracy is the same and the Average-Class-Accuracy are pretty similar. Computation time was the factor that made us choose. The mean computational time for all the experiments is $computational\_mean = 3.81$. The model with the following parameters, (*#bins=15, K=1, Dist.Func.=motyka*) has the closest computational time to the mean. These are the pare meters we chose. In previous research it was shown that GOOD works better for 15 and 30 bins. The best parameter for the nearest algorithm is 1. This is logical because similar histogram from different categories will not take part in the decision the closest neighbor.

The motyka distance function [4], takes as nominator the highest coordinate for every bins of the histogram and adds them

$$d_{mot} = \frac{\sum_{i=1}^{d} max(P_i, Q_i)}{\sum_{i=1}^{d}(P_i + Q_i)} \tag{1}$$

for all the bins making the biggest vector that can be made if we combine each bin bar plot from the histogram. As denominator we have a of all the coordinates for every bin which translates the area formed by the bars in the histogram plot.

In the ESF approach, things were easier as optimal Instance-Accuracy and the Average-Class-Accuracy values were achieved by the same parameters as shown if Fig.5. The parameters that were chosen are *K = 1, Dist.Func = chiSquared*.

	index	0	0
0	Exp	1	1
1	Obj.Desc.	ESF	ESF
2	K	1	1
3	Dist.Func.	chiSquared	chiSquared
4	Ins-Acc	0.9739	0.9739
5	Avg-Class-Acc	0.972	0.972
6	Time(s)	12.38	12.38

Fig. 5: Table representation of the best results for ESF

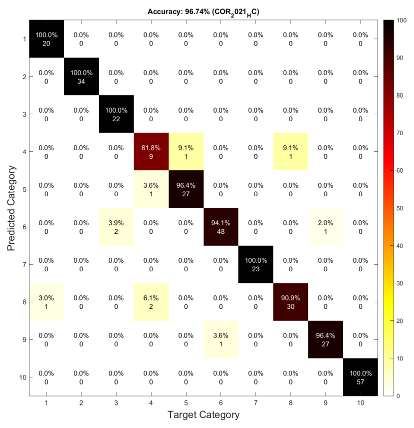


Fig. 6: Heat-map of the optimized GOOD

The heat-map of the best experiment for both descriptors is represented in Fig.6 and Fig.7. The heat-map shows the accuracy of the model on each class. The darker the colour the better the accuracy. These experiments did not achieve better results because of class imbalance. Class imbalance refers to an uneven number of samples for each category. This puts an

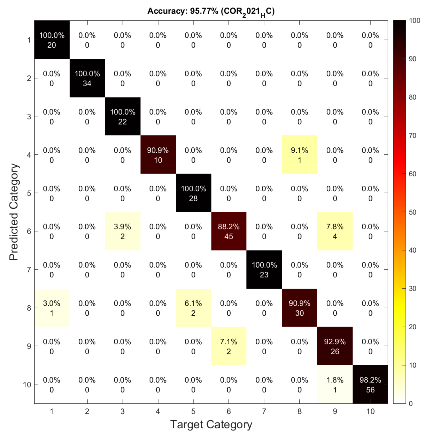


Fig. 7: Heat-map of the optimized ESF descriptor

extra burden to the model in recognizing and detecting the object from those classes. K-fold cross validation was a good way to overcome this problem but still we can see that the ”Fork” class (Category 4), the one with the smallest amount of samples, is the most difficult to detect. The simplest way to fix this, is to just simply get even samples for each class. So we need to increase the samples for the classes that suffer.

III. Offline 3D object recognition with deep-transfer learning object representation

Another approach to develop an accurate model for object representation are the use of convolutional neuronal network with deep-transfer learning techniques.

Convolutional neural networks (CNN) are a class of artificial neural network that are designed to automatically learn by using multiples blocks, suck as convolution layers, pooling layers, and fully connected layers. In this approach, these CNNs will receive as an input an instance that will be transformed into a feature vector, that will subsequently be compared the same way that the histogram of the hand-crafted approach. In this part, the *mobileNetV2* and *vgg19_fc1* architectures are going to be tested.

The neural networks that we are going to test, use deep-transfer learning techniques. That means that each network has already been pre-trained for a related task and are going to be used as the starting points to find the optimal model. More information about the functionality of the networks and the advantages will be discussed in the next chapter. The only certainty is that these CNNs are more capable to recognizing the objects in our dataset. The dataset used in previous experiments (hand-crafted object representation) is used for these experiments also, but is only used for testing. Finally, the same criteria for evaluation of the model are used (Ins-Acc, AVG-Class-Acc, Computational time).

A. Parameters

The experiments described and exposed in these sections are configured with different approach of the following parameters:

- Network architecture: *mobileNetV2* and *vgg19_fc1*.
- Matrix element Pooling: Pooling layers are used for feature matrix dimensions reduction by summarizing the features presented in a region of the feature map generated by the convolutional layer. As a consequence, the computational time to create an accurate model is reduced due to the decrease of the number of parameters needed to learn. This makes the model more robust to variations in the position of the features in the input image. In this part we are going to work with the following types of polling layers:
  - MAX: layer that calculates the maximum value of a feature map region and gives it as an output.
  - AVG: layer that calculates the mean value of a feature map region and gives it as an output.
  - APP pooling: layer that adds all the values from a feature map region and gives it as an output.
- Distance function: *cosine, gower, chiSquared, kLDivergance, symmetricKL, motyka, euclidean, intersection, dice, bhattacharyya, sorensen, canberra, pearson, neyman*.
- K: (1,3,5,7,10).

B. MobileNetV2 Architecture

MobileNetV2 is a mobile general purpose architecture created and designed for detection and classification. The main feature of this CNN is the small number of operations and low memory usage while maintaining good performance compared to other CNN. As a consequence, this type of CNN achieves similar performance in relation with other state-of-art networks while being simpler and faster [8].

The architecture of MobileNetV2 includes a convolution layer with 32 filters, followed by 19 residual bottleneck layers. In Fig.8 the convolutional blocks of the MobileNetV2 architecture is presented.

C. Vgg19_fc1 Architecture

The VGG19 is a variant of the VGG (Visual Geometry Group) architecture which contains a total of 19 layers, in particular 16 convolution layers, 3 fully connected layers, 5 MaxPool layers and 1 SoftMax layer9.

The main purpose for which VGG net was designed was to win the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) but it has been used in many other ways. This architecture is suitable for classification tasks that mainly focuses on the accuracy.

D. Experiment and Results

After running multiples experiments with the mentioned configurations, the performance of each approach is evaluated, as done before, based on: instance accuracy, average class accuracy and computational time. In Fig.10 the best results in terms Instance-Accuracy and the Average-Class-Accuracy of all the experiments using the MobileNetV2 architecture are shown:

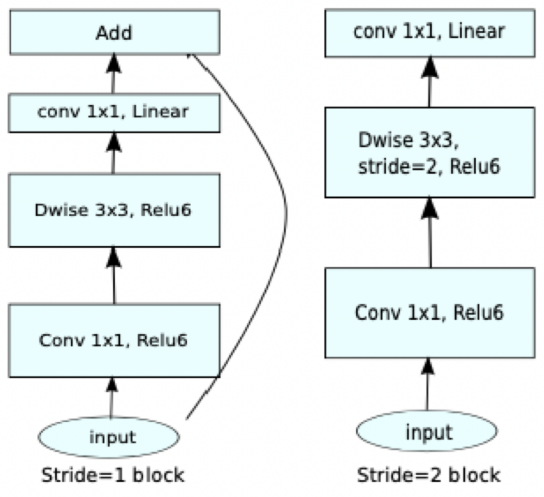


Fig. 8: Representation of the MovileNetV2 convolutional blocks

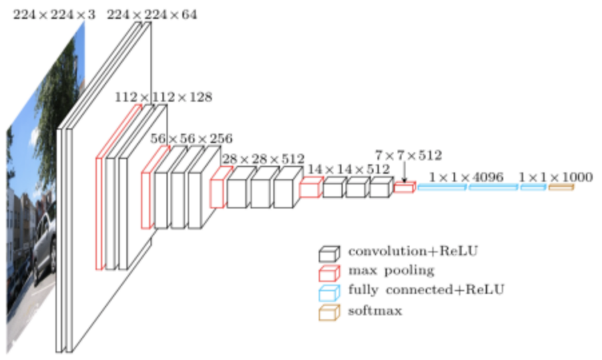
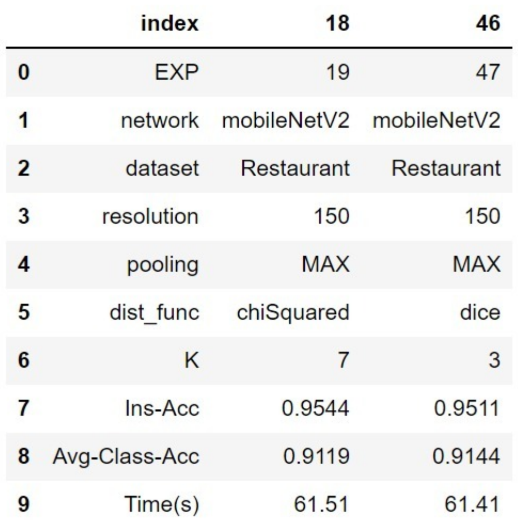


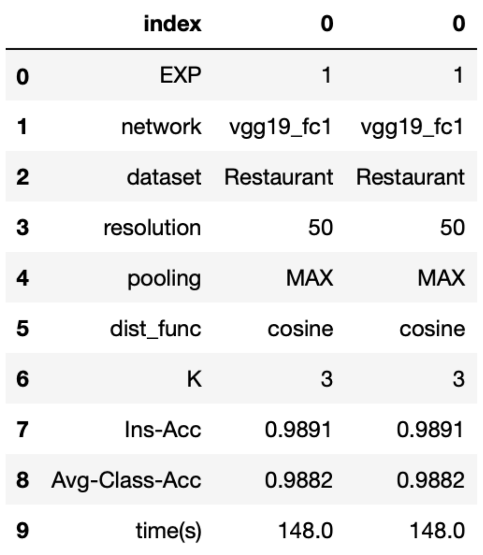
Fig. 9: Representation of the Vgg19_fc1 architecture



	index	18	46
0	EXP	19	47
1	network	mobileNetV2	mobileNetV2
2	dataset	Restaurant	Restaurant
3	resolution	150	150
4	pooling	MAX	MAX
5	dist_func	chiSquared	dice
6	K	7	3
7	Ins-Acc	0.9544	0.9511
8	Avg-Class-Acc	0.9119	0.9144
9	Time(s)	61.51	61.41

Fig. 10: Representation of the best results of experiments launch with MobileNetV2 architecture

On the other hand, we evaluated the performance of the Vgg19_fc1 architecture with different configurations of the parameters mentioned above. In Fig.11 we can observe the experiments with the best performance in terms of instance accuracy and average class accuracy.



	index	0	0
0	EXP	1	1
1	network	vgg19_fc1	vgg19_fc1
2	dataset	Restaurant	Restaurant
3	resolution	50	50
4	pooling	MAX	MAX
5	dist_func	cosine	cosine
6	K	3	3
7	Ins-Acc	0.9891	0.9891
8	Avg-Class-Acc	0.9882	0.9882
9	time(s)	148.0	148.0

Fig. 11: Table representation of the best results of experiments launch with Vgg19_fc1 architecture

We can clearly see that all the specifications of each network are shown in our experiments. Vgg19_fc1 architecture which mainly focus of accurate models, had 98.91% instance accuracy and the same average class accuracy. On the other hand MobileNetV2 achieves good accuracy while keeping the computational time at the lowest level. As a consequence, the vgg19_fc1 CNN architecture significantly greater performance than the MobileNetV2. The difference in computational time is substantial. For that reason, we have selected the MobileNetV2 architecture with the TableI.

Configuration with the best performance	
Network	MobileNetV2
Resolution	150
Pooling Function	MAX
Distance function	chi-squared
K-nearest-neighbors	7

TABLE I: Parameters of optimised mobileNetV2 network

IV. Open-ended learning system approach

The open-ended learning implies that the set of categories for learning are unknown to the robot. The training instances are extracted from on-line experiences of the robot, and thus become gradually available over time, rather than being completely available at the beginning of the learning process. On-line evaluation methodologies are necessary to evaluate these kind of systems with the simultaneous nature the open-ended learning. There are different scenarios and characteristics of an open-ended learning system.

1) Supervised: A human is added to the system as an instructor. He provides the labels of unknown objects and can also correct the robot if it wrongfully categorizes an object.
2) On-line: Learning procedure is held while the robot is running.
3) Opportunistic: The robot should always be prepared to accept new examples when one is observed
4) Incremental: The robot is able to adjust specific categories when new instances of those are taught.
5) Concurrent: Robot is able to handle multiple learning problems at the same time.

Our model should have the on-line characteristic. The learning procedure should also be supervised. One approach is to build a deep CNN and train it. However, there are several limitation in open-ended domains, as introduction of new categories forces a reconstruction of the network. A suggested solution is OrthographicNet [10]. The three basic views (top view, side view, front view) from the input category are each fed to a CNN. Each output vector is fed to a pooling layer where the final feature vector is extracted 12. Then instead of having an output layer to choose the category, the OrthographicNet is linked with an on-line classifier which handles the problem of the open-ended object category learning and recognition.

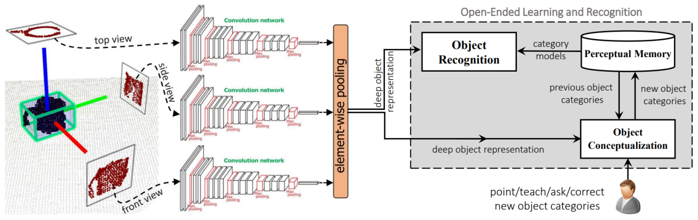


Fig. 12: OrthographicNet as open-ended object recognition pipeline [10]

A. Open-Ended Evaluation

In a human-robot interaction, the robot is always keeping track of the objects but also the human that places the objects in its field of vision. The human/instructor gives guidance to the robot with a properly built Graphical User Interface (GUI). Through that, the instructor can provide labels for unknown objects but also correct the robot when it mislabels one, in real time.

Open-ended evaluation tries to mimic the human-robot interaction. The algorithm is based in a `Test-then-Train` scheme. This procedure is given to us in the form of C++ file called `simulated_teacher` for both hand-crafted and deep-learning approaches. This allows the possibility to perform multiple experiments and test different experimental conditions in a fraction of time. The `simulated_teacher`, has three basic actions:

- Teach: Introduce a new category to the robot/agent.
- Ask: Request the agent to recognize the category of the object.
- Correct: Provide the correct label when misclassification happens.

The algorithm that implements the procedure is shown in Fig.13.

```
Introduce $Category_1$
$n \leftarrow 1$
repeat
    $n \leftarrow n+1$                                          ▷ Ready for the next category
    Introduce $Category_n$
    $k \leftarrow 0$
    $c \leftarrow 1$
    repeat                                                      ▷ question / correction iteration
        Present a previously unseen instance of $Category_c$
        Ask the category of this instance
        If needed, provide correct feedback
        $c \leftarrow (c == n)$ ? $1 : c+1$
        $k \leftarrow k+1$
        $s \leftarrow$ success in last $k$ question/correction iterations
    until (($s > \tau$ and $k >= n$)                           ▷ accuracy threshold crossed
            or (user sees no improvement in success))           ▷ breakpoint reached
until (user sees no improvement in success)                     ▷ breakpoint reached
```

Fig. 13: Algorithm foe the teaching protocol.

The evaluation begins with no previous knowledge. Three views from a random object are picked and introduced. The agent stores this views in his memory and the category is learned. After a new category is learned, the teacher introduces a new category but also checks for all the previous knowledge, in case a known category is forgotten. This evaluation is done by picking never-seen before views from a learned category and asking the agent to recognize the object. If the classification is wrong, teacher provides a corrective feedback, and the new instance is stored. This keeps the amount of instances in the memory for a category at the minimum level. This recognition performance is continuously calculated and compared with a threshold value called `protocol_accuracy` via a sliding window of size $3n$ where *"n"* is the number of learned categories. If the recognition performance is higher than the threshold value, the teacher randomly chooses and introduces a new category. If the value of lower than the threshold for a great amount of iterations (e.g. 100), the experiment stops based on the conclusion that the agent can no longer learn any more categories. The experiment also stops when there are no new categories to be learned.

B. Experiment

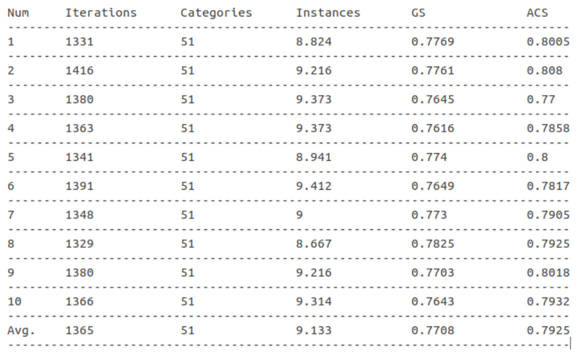
Based on the obtained results with the best configurations for both hand-crafted and deep transfer learning approaches, we update the parameters of the `simulated_teacher` launch file. The order of category introduction plays a big role in the overall learning ability (accuracy) of the robot. If for example the the first two categories have a very similar point cloud (apple, orange), the robot might not be able to learn any new category and the process might terminate. That is why we are running 10 experiments and keep the order of introduced categories the same. One key aspect of this assignment to investigate the impact of different `protocol_threshold` values in the recognition performance as well as the overall time of the recognition of the whole dataset. The dataset that is used (*Washington RGB-D Object Dataset*) contains points clouds of different everyday objects in RGB-D fashion. It has 51 categories and in each category there are 300 instances (point clouds of objects).

C. Results

The first round of experiments was carried out with a *protocol accuracy* set to 0.67. Both hand-crafted and deep-transfer learning approaches where used. The number of experiments in each set is set to 10 for each approach and an average value for each metric is extracted. The metrics than each set of experiment will be evaluated are:

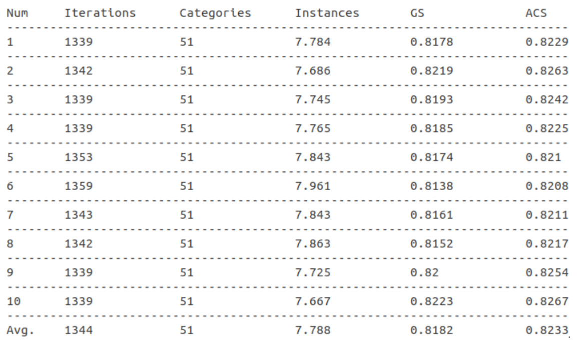
- global classification accuracy
- number of learned categories
- number of stored instances per category
- protocol accuracy

In each following round of experiments, only the hyper-parameter of `protocol_accuracy` changes. The discrete values that are assigned to it are $[0.7, 0.8, 0.9]$. In Fig.14, the template of the results representation is shown.



Num	Iterations	Categories	Instances	GS	ACS
1	1331	51	8.824	0.7769	0.8005
2	1416	51	9.216	0.7761	0.808
3	1380	51	9.373	0.7645	0.77
4	1363	51	9.373	0.7616	0.7858
5	1341	51	8.941	0.774	0.8
6	1391	51	9.412	0.7649	0.7817
7	1348	51	9	0.773	0.7905
8	1329	51	8.667	0.7825	0.7925
9	1380	51	9.216	0.7703	0.8018
10	1366	51	9.314	0.7643	0.7932
Avg.	1365	51	9.133	0.7708	0.7925

(a) Hand-Creafted Approach



Num	Iterations	Categories	Instances	GS	ACS
1	1339	51	7.784	0.8178	0.8229
2	1342	51	7.686	0.8219	0.8263
3	1339	51	7.745	0.8193	0.8242
4	1339	51	7.765	0.8185	0.8225
5	1353	51	7.843	0.8174	0.821
6	1359	51	7.961	0.8138	0.8208
7	1343	51	7.843	0.8161	0.8211
8	1342	51	7.863	0.8152	0.8217
9	1339	51	7.725	0.82	0.8254
10	1339	51	7.667	0.8223	0.8267
Avg.	1344	51	7.788	0.8182	0.8233

(b) Deep-Transfer Learning approach

Fig. 14: First round of experiments. Protocol accuracy= 0.67. Columns from left to right: *Number of experiment, Iterations until the termination of procedure, Categories learned in the experiment, Mean value of instances saved in the memory for each learned category, Global classification accuracy, Protocol's accuracy.*

As we can observe in both approaches, the agent is capable of learning all the categories with very similar experiment execution time (20 iterations margin). Memory-wise the Deep-Transfer Learning approach is more efficient because it stores about 1.5 object instances less than the Hand-crafted which translates to about 100 views extra storage space. Finally that approach is more accurate as it achieves a higher F1 score (GS column) and higher protocol accuracy (ACS column).

The GS column or F1 score metric, is a harmonic mean value of the precision and recall of the experiment. Precision is the *positive predictive value* of the model and recall is the model's *sensitivity* (predictive accuracy). The ACS column the overall value of the protocol's accuracy and evaluation procedure throughout the whole experiment.

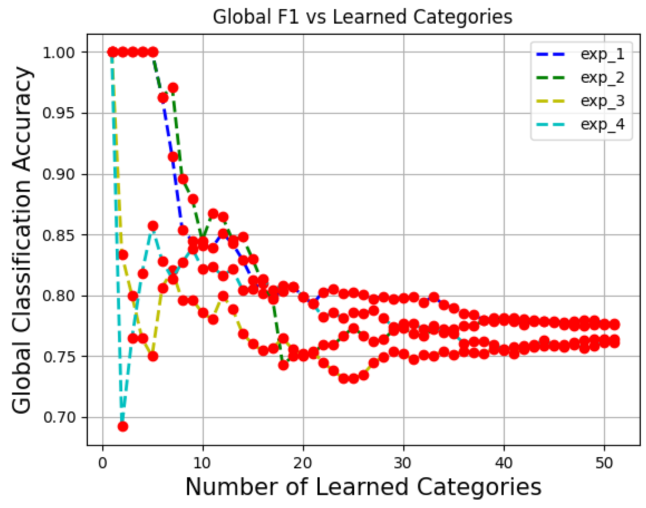


Fig. 15: F1 metric evaluation when a new category is learned

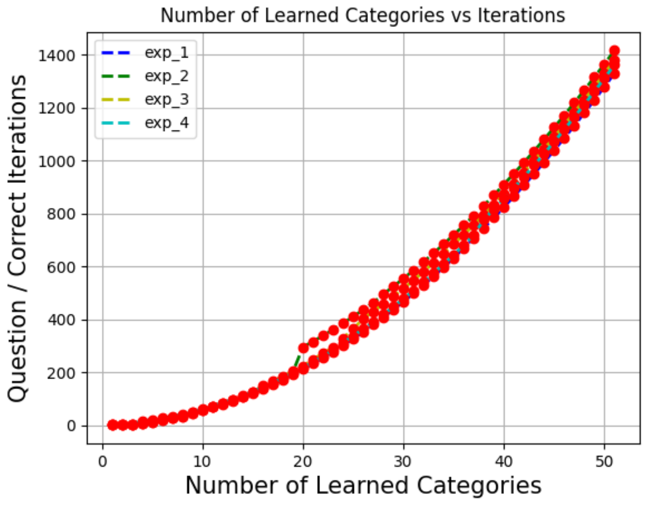


Fig. 16: Agent's learned categories based on protocol's evaluation iteration.

As an example, four plots (Fig.15, Fig.16, Fig.17, Fig.18) for each metric are provided for visualization of the results. Only four experiments can be plotted at a time. A similar pattern for each experiment is expected as we can see from the previous tables. This is due to the threshold value as well as the fixed introduction of categories to the agent. Because we don't put a high restriction on the robots recognition performance, it fairly unlike that the operation will be terminated due to the conclusion that the agent is not in a position to learn any more categories. This also gives the robot the freedom to store more instances per category, because it is *"allowed"* to make more mistakes. The next round of experiments is done by setting the *protocol value* equal to 0.7. That begin said, we do not except things to change drastically or hardly at all.

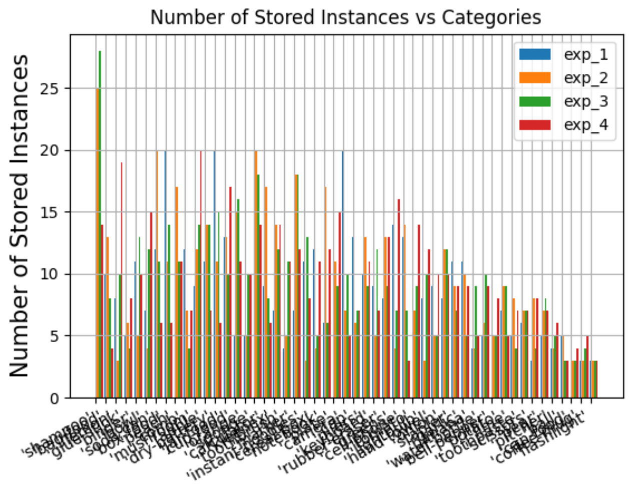


Fig. 17: Stored instances for each category for each experiment.

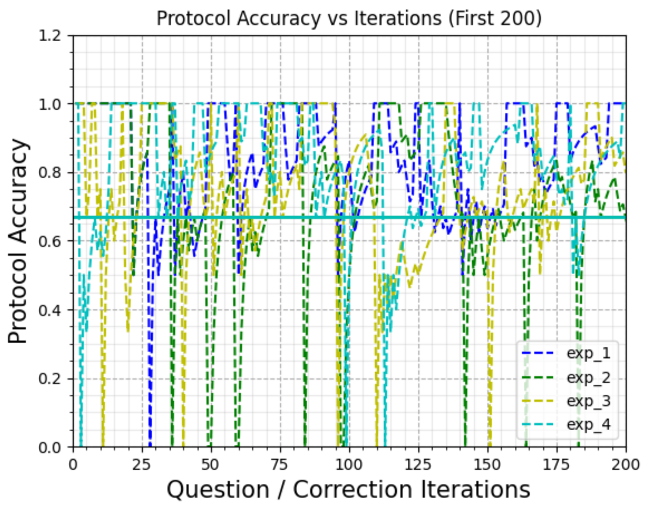
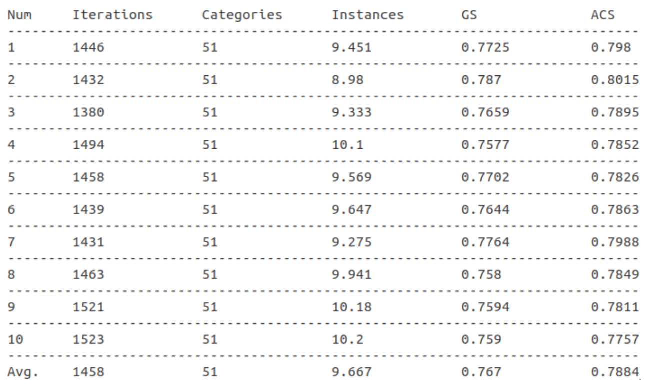
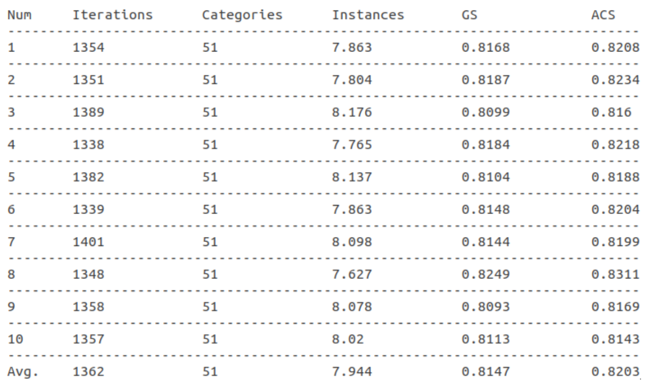


Fig. 18: Protocol's accuracy evaluation on each iteration. Vertical line is the protocol threshold value.



Num	Iterations	Categories	Instances	GS	ACS
1	1446	51	9.451	0.7725	0.798
2	1432	51	8.98	0.787	0.8015
3	1380	51	9.333	0.7659	0.7895
4	1494	51	10.1	0.7577	0.7852
5	1458	51	9.569	0.7702	0.7826
6	1439	51	9.647	0.7644	0.7863
7	1431	51	9.275	0.7764	0.7988
8	1463	51	9.941	0.758	0.7849
9	1521	51	10.18	0.7594	0.7811
10	1523	51	10.2	0.759	0.7757
Avg.	1458	51	9.667	0.767	0.7884

(a) Hand-Creafted Approach



Num	Iterations	Categories	Instances	GS	ACS
1	1354	51	7.863	0.8168	0.8208
2	1351	51	7.804	0.8187	0.8234
3	1389	51	8.176	0.8099	0.816
4	1338	51	7.765	0.8184	0.8218
5	1382	51	8.137	0.8104	0.8188
6	1339	51	7.863	0.8148	0.8204
7	1401	51	8.098	0.8144	0.8199
8	1348	51	7.627	0.8249	0.8311
9	1358	51	8.078	0.8093	0.8169
10	1357	51	8.02	0.8113	0.8143
Avg.	1362	51	7.944	0.8147	0.8203

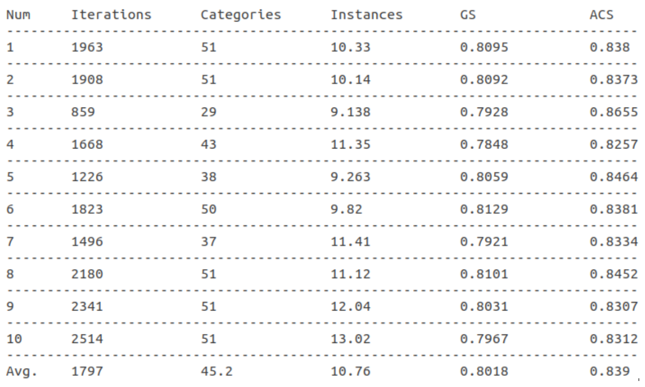
(b) Deep-Transfer Learning approach

Fig. 19: Next round of experiments. Protocol accuracy= 0.7.

As we expected no essentials differences between there two rounds of experiments is noticed (Fig.19). For the Hand-crafted approach we see that this 0.03 value difference, puts a tight constraint of the agent, by increasing his computational time to an average of 100 iterations. In both approaches the accuracy drops and also more instances per class are stored in the memory.

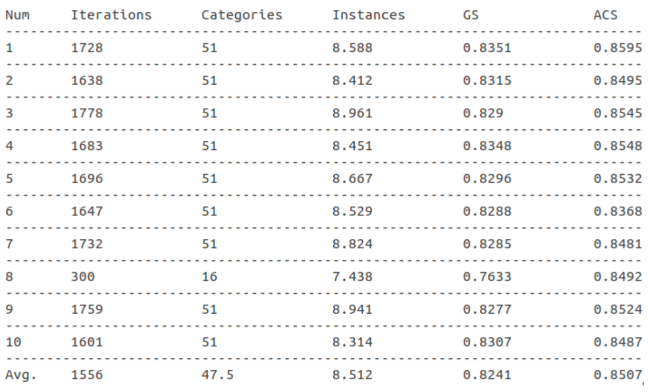
In the following round of experiments, a much quicker termination of the procedure for some experiments or a much more time-consuming for other is expected . This leads to smaller average value of learned categories. The results of this round of experiment for both approaches is presented in Fig.20. Deep-Transfer Learning approach is much more consistent without many variations throughout the round of experiments. Also it is more precise in the robot's guesses and far more memory efficient. We highlight four experiments for the Hand-Crafted approach, (*Num*$= [2, 3, 7, 10]$). In experiment $N_o = 2$ the agent is able to learn all the categories but in very long time (1908 iterations). In contrast to the experiments of the previous round, The agent's accuracy is higher than before (Fig.21) with only one more saved instance at average for each category. In experiment $N_o = 3$ the operation has to terminate after 859 iterations. Also in the $N_o = 7$ experiment, the experiment takes more time than the previous one, only to learn 37 categories. The final experiment is the less time-efficient of them all, but manages to learn all 51 categories.

The most important information that we gain is that the overall performance as well as the protocol's accuracy (Fig.24) get higher values. The same things apply to the Deep-Transfer Learning approach with only difference that the latter



Num	Iterations	Categories	Instances	GS	ACS
1	1963	51	10.33	0.8095	0.838
2	1908	51	10.14	0.8092	0.8373
3	859	29	9.138	0.7928	0.8655
4	1668	43	11.35	0.7848	0.8257
5	1226	38	9.263	0.8059	0.8464
6	1823	50	9.82	0.8129	0.8381
7	1496	37	11.41	0.7921	0.8334
8	2180	51	11.12	0.8101	0.8452
9	2341	51	12.04	0.8031	0.8307
10	2514	51	13.02	0.7967	0.8312
Avg.	1797	45.2	10.76	0.8018	0.839

(a) Hand-Creafted Approach



Num	Iterations	Categories	Instances	GS	ACS
1	1728	51	8.588	0.8351	0.8595
2	1638	51	8.412	0.8315	0.8495
3	1778	51	8.961	0.829	0.8545
4	1683	51	8.451	0.8348	0.8548
5	1696	51	8.667	0.8296	0.8532
6	1647	51	8.529	0.8288	0.8368
7	1732	51	8.824	0.8285	0.8481
8	300	16	7.438	0.7633	0.8492
9	1759	51	8.941	0.8277	0.8524
10	1601	51	8.314	0.8307	0.8487
Avg.	1556	47.5	8.512	0.8241	0.8507

(b) Deep-Transfer Learning approach

Fig. 20: Next round of experiments. Protocol accuracy$=0.8$.

is a more efficient. Not only higher accuracies are achieved, but also less instances in average are stored in the memory. The only drawback is that the agent cannot learn 10% of the categories.

In the most strict experiment, where the $protocol\ threshold = 0.9$, we can clearly observe, the effect that a high threshold value has for an agent's ability to learn. In Fig.25 the metrics for each experiments are presented.

Neither approach is able to learn all the categories all the experiments stop due to the fact that the agent is not able to learn any more categories in the last k iterations.Also, this extra difficulty on the agent, makes it store almost the same number of instances for each category in a much shorter experiment.One good outcome is that the model's as well as the protocol's accuracy have a high value. Let's inspect the experiment with the less iterations ($Num = 6$) and especially Fig.26 which shows the protocol's accuracy throughout the experiment.We observe that after the introduction of some categories (*ball, flashlight, rubber-eraser*) the agent spends a lot of time under the threshold value (0.9) meaning that it is hard for him to distinguish these objects from other stored categories. If we observe the experiment's log but also Fig.27

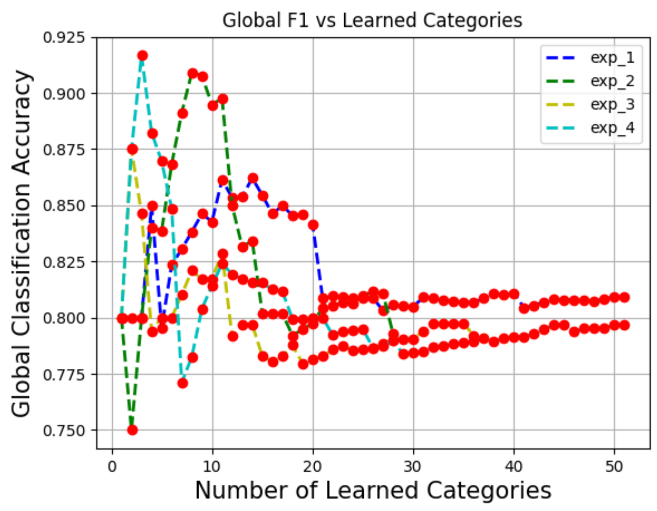


Fig. 21: F1 metric evaluation when a new category is learned

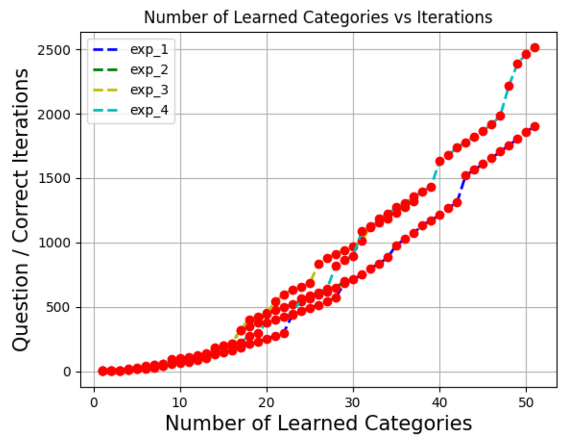


Fig. 22: Agent's learned categories based on protocol's evaluation iteration.

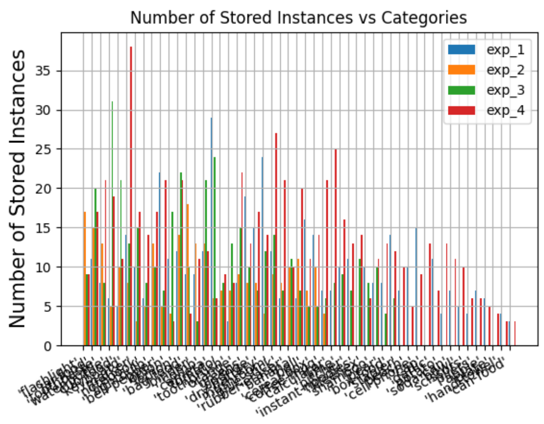


Fig. 23: Stored instances for each category for each experiment

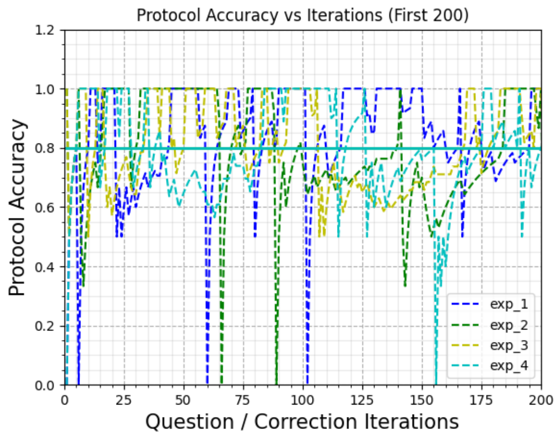
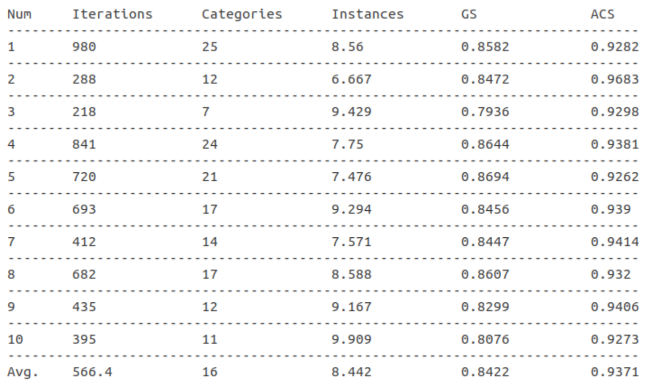
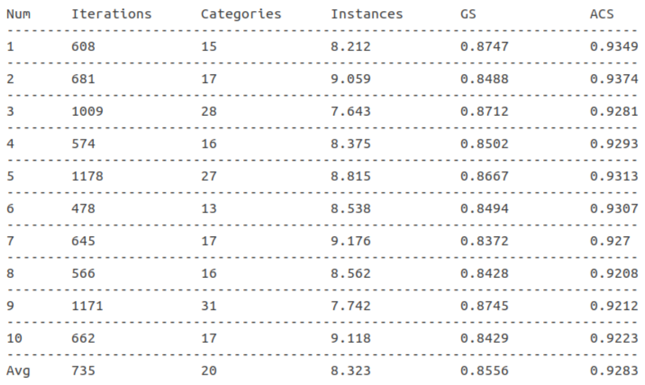


Fig. 24: Protocol's accuracy evaluation on each iteration. Vertical line is the protocol threshold value.



Num	Iterations	Categories	Instances	GS	ACS
1	980	25	8.56	0.8582	0.9282
2	288	12	6.667	0.8472	0.9683
3	218	7	9.429	0.7936	0.9298
4	841	24	7.75	0.8644	0.9381
5	720	21	7.476	0.8694	0.9262
6	693	17	9.294	0.8456	0.939
7	412	14	7.571	0.8447	0.9414
8	682	17	8.588	0.8607	0.932
9	435	12	9.167	0.8299	0.9406
10	395	11	9.909	0.8076	0.9273
Avg.	566.4	16	8.442	0.8422	0.9371

(a) Hand-Creafted Approach



Num	Iterations	Categories	Instances	GS	ACS
1	608	15	8.212	0.8747	0.9349
2	681	17	9.059	0.8488	0.9374
3	1009	28	7.643	0.8712	0.9281
4	574	16	8.375	0.8502	0.9293
5	1178	27	8.815	0.8667	0.9313
6	478	13	8.538	0.8494	0.9307
7	645	17	9.176	0.8372	0.927
8	566	16	8.562	0.8428	0.9208
9	1171	31	7.742	0.8745	0.9212
10	662	17	9.118	0.8429	0.9223
Avg	735	20	8.323	0.8556	0.9283

(b) Deep-Transfer Learning approach

Fig. 25: Next round of experiments. Protocol accuracy= 0.9.

we see what clearly happens. Category ball is misclassified as a mushroom, pear and orange, orange as ball and so on. This confusion combined with the high threshold value, is the reason that the experiment is terminated only after 412 iterations.

D. Conclusions

A series of experiments was performed to calculate the learning performance of an agent in an open-ended domain. An algorithm for evaluation was implemented and different kind of criteria for evaluation were applied. Both Hand-Crafted and Deep-Transfer Learning approaches were tested. Bases on the experiments both approaches perform very similar but the latter one does it in less time (fewer iterations) and uses less memory by needing less instances, in average, stored per category. When the agent has more freedom (smaller threshold value), the accuracy of the model is not remarkable but manages to learn all the categories and the process ends for that reason. When the criteria gets stricter the agent finds it difficult to learn all categories, needs about one more instance per category but achieves higher accuracies. One thought about future experiments in to test the concept of ”adapting threshold”. This means that throughout the experiment, the threshold value will vary based on the value of the performance. This can help the agent achieve better accuracies while still maintaining a high value of learned categories.

As it applies to humans, the more ”successful” you want to be in one field the more you are likely to restrict yourself in order to achieve your goals. That might make you ”forget” about other things that might be essential and meaningful. If you try to be more balanced you can achieve more things but in a ”less successful” way.

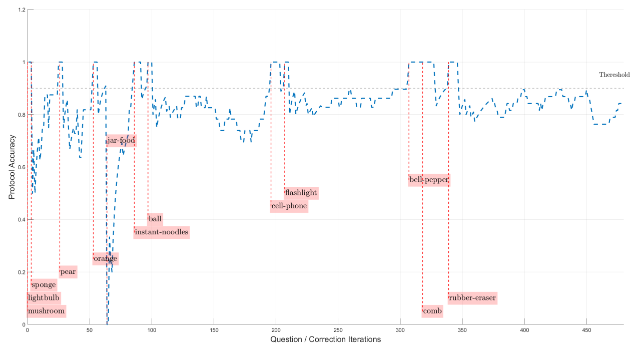


Fig. 26: Protocol accuracy on experiment 6.

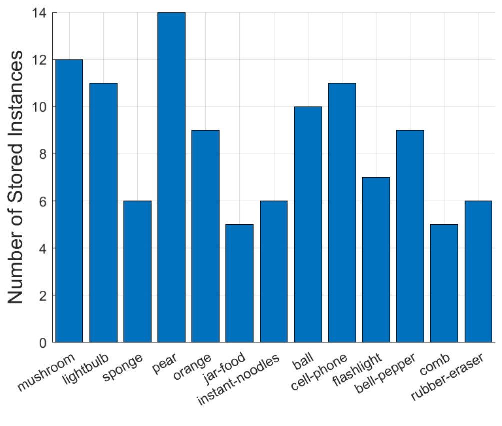


Fig. 27: Stored instances for each category in experiment 6.

REFERENCES

[1] S. Hamidreza Kasaei. Ana Maria Tomé. Luís Seabra Lopes. Miguel Oliveira, ”GOOD: A global orthographic object descriptor for 3D object recognition and manipulation”, Pattern Recognition Letters, Volume 83, Part 3, 2016, Pages 312-320, ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2016.07.006.

[2] W. Wohlkinger and M. Vincze, ”Ensemble of shape functions for 3D object classification,” 2011 IEEE International Conference on Robotics and Biomimetics, 2011, pp. 2987-2992, doi: 10.1109/ROBIO.2011.6181760.

[3] Himri K. Ridao P. Gracias N., ”3D Object Recognition Based on Point Clouds in Underwater Environment with Global Descriptors: A Survey,” Sensors 2019, 19, 4451, https://doi.org/10.3390/s19204451.

[4] Cha, Sung-Hyuk. (2007). Comprehensive Survey on Distance/Similarity Measures Between Probability Density Functions. Int. J. Math. Model. Meth. Appl. Sci.. 1.

[5] Chuanqi Tan1. Fuchun Sun2. Tao Kong1. Wenchang Zhang1. Chao Yang1. Chunfang Liu2. “A Survey on Deep Transfer Learning ”, https://arxiv.org/pdf/1808.01974.pdf

[6] Hossein Gholamalinezhad1. Hossein Khosravi. ”Pooling Methods in Deep Neural Networks, a Review”,https://arxiv.org/pdf/2009.07485.pdf

[7] Mark Sandler. Andrew Howard. (2018). ”MobileNetV2: The Next Generation of On-Device Computer Vision Networks ”, https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html. I

[8] Mark Sandler. Andrew Howard. Menglong Zhu. Andrey Zhmoginov. Liang-Chieh Chen, ”MobileNetV2: Inverted Residuals and Linear Bottlenecks”, https://arxiv.org/abs/1801.04381.

[9] Karen Simonyan. Andrew Zisserman. (2015). ”VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION ”, https://arxiv.org/pdf/1409.1556v6.pdf

[10] Kasaei, Hamidreza. (2020). OrthographicNet: A Deep Transfer Learning Approach for 3D Object Recognition in Open-Ended Domains. IEEE/ASME Transactions on Mechatronics. PP. 1-1. 10.1109/TMECH.2020.3048433.