Boosting Deep Open World Recognition by Clustering

Boosting Deep Open World Recognition by Clustering

Dario Fontanel*,1, Fabio Cermelli*,1,2, Massimiliano Mancini[3], Samuel Rota Bulò[4], Elisa Ricci[5] and Barbara Caputo[1,2]

***Abstract*—While convolutional neural networks have brought significant advances in robot vision, their ability is often limited to *closed world* scenarios, where the number of semantic concepts to be recognized is determined by the available training set. Since it is practically impossible to capture all possible semantic concepts present in the real world in a single training set, we need to break the closed world assumption, equipping our robot with the capability to act in an *open world*. To provide such ability, a robot vision system should be able to (i) identify whether an instance does not belong to the set of known categories (i.e. open set recognition), and (ii) extend its knowledge to learn new classes over time (i.e. incremental learning). In this work, we show how we can boost the performance of deep open world recognition algorithms by means of a new loss formulation enforcing a global to local clustering of class-specific features. In particular, a first loss term, i.e. *global clustering*, forces the network to map samples closer to the class centroid they belong to while the second one, *local clustering*, shapes the representation space in such a way that samples of the same class get closer in the representation space while pushing away neighbours belonging to other classes. Moreover, we propose a strategy to learn class-specific rejection thresholds, instead of heuristically estimating a single global threshold, as in previous works. Experiments on three benchmarks show the effectiveness of our approach.**

***Index Terms*—Deep Learning for Visual Perception, Visual Learning, Recognition**

I. Introduction

A long-standing goal of artificial intelligence and robotics is implementing agents able to interact in the real world. In order to achieve this goal, a crucial step is making the agent able to understand the current state of the surrounding environment. Within this context, visual cameras are one of the most powerful and information-rich sensors, thus a lot of research efforts have been spent on improving robot vision systems. Due to their effectiveness in addressing visual problems, deep neural networks have been used in many robotic tasks such as egomotion estimation [1], depth prediction [2], [3], object grasping [4], [5] and semantic segmentation [6], [7]. Despite their effectiveness, deep neural networks limit their understanding to the particular set of knowledge present in the training set they are tuned on, relying on the *closed world assumption* (CWA). Obviously, this is a fundamental drawback if we want to apply any visual system, especially a recognition based one, in the real world. Indeed, the world contains an infinite set of possible input conditions (e.g. various illumination, environments) and semantic concepts: capturing them in a single training set is practically unfeasible. Under these perspectives, we would like to make our algorithm both robust to unseen input conditions as well as being able to detect and learn novel semantic concepts. While previous work tried to address the first problem in the context of domain adaptation [8], [9], [10] and generalization [11], little attention has been posed to the second one. Here we show how we can break the CWA developing a visual system able to work in the *open world*.

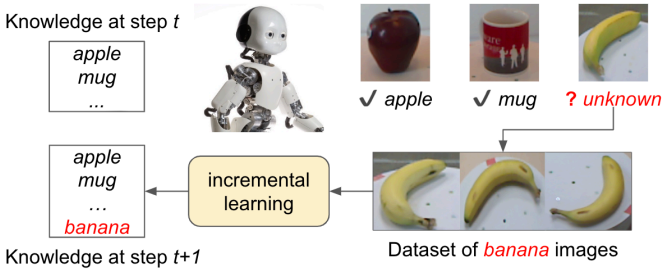


Fig. 1: In the open world scenario a robot must be able to classify correctly known objects, (*apple* and *mug*), and detect novel semantic concepts (e.g. *banana*). When a novel concept is detected, it should learn the new class from an auxiliary dataset, updating its internal knowledge.

To clarify our goal, let us consider the example shown in Fig. 1. The robot has a knowledge base composed by a limited number of classes. Given an image containing an unknown concept (e.g. banana), we want the robot to detect it as unknown and being able to add it to its knowledge base in subsequent learning stages. To accomplish this goal, it is very important for a robot vision system to have two crucial abilities: (i) it must be able to recognize already seen concepts and detect unknown ones (i.e. open set recognition), and (ii) it must be able to extend its knowledge base with new classes (i.e. incremental learning), without forgetting the already learned ones and without access to old training

Manuscript received: February, 24, 2020; Revised April, 22, 2020; Accepted June, 19, 2020.

This paper was recommended for publication by Editor Cesar Cadena Lerma upon evaluation of the Associate Editor and Reviewers' comments.

*indicates equal contribution

[1]D. Fontanel, F. Cermelli and B. Caputo are with Politecnico di Torino, Turin, Italy. {dario.fontanel, fabio.cermelli, barbara.caputo}@polito.it

[2]F. Cermelli and B. Caputo are with Italian Institute of Technology, Genoa, Italy.

[3]M. Mancini is with Sapienza University of Rome, Rome, Italy and University of Tübingen, Tübingen, Germany. mancini@diag.uniroma1.it

[4]S. Rota Bulò is with Mapillary, Graz, Austria. samuel@mapillary.com

[5]E. Ricci is with Fondazione Bruno Kessler, Trento, Italy and University of Trento, Trento, Italy. eliricci@fbk.eu

Digital Object Identifier (DOI): see top of this page.

sets, avoiding *catastrophic forgetting* [12]). While open set recognition [13], [14], [15] and incremental learning [16], [17], [18], [19] are well-studied problems in the literature, few works proposed a solution to solve them together [20], [21], [22]. Standard approaches for open world recognition (OWR) equip the nearest class mean (NCM) classification algorithm with a rejection option based on an estimated threshold. While standard approaches [20], [21] use shallow features, only recently it has been showed how deep neural networks can be successfully employed also in the OWR scenario [22]. In this work we follow the deep learning based approach of [22] but we take a step forward. In fact, we argue that it is crucial to force the deep architecture used as feature extractor to cluster appropriately samples belonging to the same class, while pushing away samples of other classes. For this reason, we introduce a global clustering loss term that aims at keeping closer the features of samples belonging to the same class to their class centroid. Furthermore, we show how the soft nearest neighbor loss [23], [24] can be successfully employed as a local clustering loss term in order to force pair of samples of the same class to be closer in the learned metric space than points of other classes. Additionally, differently from previous works [20], [22] we avoid to estimate a global rejection threshold on the model predictions based on heuristic rules but we (i) define an independent threshold for each class and (ii) we explicitly learn the thresholds by using a margin-based loss function which balances rejection errors on samples of a reserved memory held-out from the training. We evaluate the effectiveness of our method on Core50 [25], RGB-D Object Dataset [26] and CIFAR-100 [27] datasets, showing that introducing the two complementary clustering loss and learning the rejection thresholds outperforms previous approaches.

Contributions. To summarize, the contributions of this paper are as follows:

- We introduce two clustering losses to effectively localize samples of the same class in the representation space, while separating them from points belonging to other classes;
- We propose an effective method to detect unknown samples based on learned class-specific rejection thresholds;
- We demonstrate the superiority of our method over state of the art, reporting a quantitative analysis and an extensive ablation of the components of our model.

II. Related Works

The necessity of breaking the CWA for robot vision systems [28] has lead various research efforts on understanding how to extend pre-trained models with new semantic concepts while retain previous knowledge. To this extent, recent years have seen a growing interest on topics such as continual [29] and incremental learning [19], [18], [30]. In [31], the authors study how to update the visual recognition system of a humanoid robot on multiple training sessions. In [18], a variant of the Regularized Least Squares algorithm is introduced to add new classes to a pre-trained model. In [32], a growing dual-memory is proposed to dynamically learn novel object instances and categories. In [33] the authors proposed to learn an embedding in order to perform fast incremental learning of new objects. Another solution to this problem can exploit the help of a human-robot interaction, as in [19] where a robot incrementally learns to detect new objects as they are manually pointed by a human.

While these approaches focus on incremental and continual learning, acting in the open world requires both detecting unknown concepts automatically and adding them in subsequent learning stages. Towards this objective, in [20] the authors introduced the OWR setting, as a more general and realistic scenario for agents acting in the real world. In [20], the authors extend the Nearest Class Mean (NCM) classifier [34], [35] to act in the open set scenario, proposing the Nearest Non-Outlier algorithm (NNO). In order to estimate whereas a test sample belongs to the known or unknown set of categories, this method introduces a rejection threshold that, after the first initialization phase, is kept fixed for subsequent learning episodes. In [21], the authors proposed to tackle OWR with the Nearest Ball Classifier, with a rejection threshold based on the confidence of the predictions. Recently, in [22], the NNO algorithm of [20] has been extended by employing an end-to-end trainable deep architecture as feature extractor, with a dynamic update strategy for the rejection threshold. In this work, we show how we can improve the performances of NCM based classifier for OWR through a global to local clustering loss. Moreover, differently for previous works, our rejection threshold is class-specific and is explicitly learned rather than fixed based on heuristic strategies.

III. Our Method

In this section we describe our OWR method. We start by formalizing the OWR problem and describing the DeepNNO framework [22] which serves as our starting point. We then discuss our core components, the global to local clustering and how we learn the class-specific rejection thresholds.

A. Problem Definition

The goal of OWR is producing a model capable of (i) recognizing known concepts (i.e. classes seen during training), (ii) detecting unseen categories (i.e. classes not present in any training set used for training the model) and (iii) incrementally add new classes as new training data is available. Formally, let us denote as $\mathcal{X}$ and $\mathcal{K}$ the input space (i.e. image space) and the closed world output space respectively (i.e. set of known classes). Moreover, since our output space will change as we receive new data containing novel concepts, we will denote as $\mathcal{K}_t$ the set of classes seen after the t_{th} incremental step, with $\mathcal{K}_0$ denoting the category present in the first training set. Additionally, since we aim to detect if an image contains an unknown concept, in the following we will denote as unk the special unknown class, building the output space as $\mathcal{K}_t \cup \{unk\}$. We assume that, at each incremental step, we have access to a training set $\mathcal{T}_t = \{(x_1^t, c_1^t), \cdots, (x_{N_t}^t, c_{N_t}^t)\}$, with $N_t = |\mathcal{T}_t|$, $x^t \in \mathcal{X}$, and $c^t \in \mathcal{C}_t$, where $\mathcal{C}_t$ is the set of categories contained in the training set $\mathcal{T}_t$. Note that, without loss of generality, in each incremental step, we assume to see

a new set of classes $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ if $i \neq j$. The set of known classes at step t is computed as $\mathcal{K}_t = \cup_{i=0}^{t} \mathcal{C}_i$ and given a sequence of S incremental steps, our goal is to learn a model mapping input images to either their corresponding label in $\mathcal{K}_S$ or to the special class unk. In the following we will split the classification model into two components: a feature extractor f that maps the samples into a feature space and a classifier g that maps the features into a class label, i.e. $g(f(x)) = c$ with $c \in \{\mathcal{K}_S, unk\}$.

B. Preliminaries

Standard approaches to tackle the OWR problem apply non-parametric classification algorithms on top of learned metric spaces [20], [21]. A common choice for the classifier g is the Nearest Class Mean (NCM) [34], [35]. NCM works by computing a centroid for each class (i.e. the mean feature vector) and assigning a test sample to the closest centroid in the learned metric space. Formally, we have:

$$g^{\text{NCM}}(x) = \arg\min_{c \in \mathcal{C}_t} d(f(x), \mu_c) \tag{1}$$

where $d(\cdot, \cdot)$ is a distance function (e.g. Euclidean) and μ_c is the mean feature vector for class c. The standard NCM formulation cannot be applied in the OWR setting since it lacks the inherent capability of detecting images belonging to unknown categories. To this extent, in [20] the authors extend the NCM algorithm to the OWR setting by defining a rejection criterion for the unknowns. In this extension, called Nearest Non-Outlier (NNO), class scores are defined as:

$$s_c^{\texttt{NNO}}(x) = \mathcal{Z}(1 - \frac{d(f(x), \mu_c)}{\tau}), \tag{2}$$

where τ is the rejection threshold and $\mathcal{Z}$ is a normalization factor. The final classification is held-out as:

$$g(x) = \begin{cases} unk & \text{if } s_c^{\texttt{NNO}}(x) \leq 0 \ \forall c \in \mathcal{K}_t, \\ g^{\text{NCM}}(x) & \text{otherwise.} \end{cases} \tag{3}$$

Following [34], in [20] the features are linearly projected into a metric space defined by a matrix W (i.e. $f(x) = W \cdot x$), with W learned on the first training set $\mathcal{T}_0$ and kept fixed during the successive learning steps. The main limitation of this approach is that new knowledge will be incorporated in the classifier g without updating the feature extractor f accordingly. In [22], it is shown how the performance of NNO can be significantly improved by using as f a deep architecture trained end-to-end in each incremental step. The proposed algorithm, DeepNNO, trains the deep neural network by minimizing the binary cross-entropy loss:

$$\ell(x_i, c_i) = \sum_{c \in \mathcal{C}_t} \mathbb{1}_{c=c_i} \log(s_c^{\texttt{DNNO}}(x_i)) + \mathbb{1}_{c \neq c_i} \log(1 - s_c^{\texttt{DNNO}}(x_i)) \tag{4}$$

where $s_c^{\texttt{DNNO}}(x)$ is the class scores computed as $s_c^{\texttt{DNNO}}(x) = e^{-\frac{1}{2}||f(x)-\mu_c||^2}$. Differently from [34], [20], the underlying feature representation of the data changes along with the parameters of the backbone architecture. As a consequence, it is not possible to fix the class-specific centroids, especially in the incremental learning setting, since changes in the network parameters will create a shift among the computed old class centroids and the current network activations. Such shift cannot be recovered, since the training sets $\mathcal{T}_i$ with $i < t$ are not available. To overcome this problem, DeepNNO proposes to (i) update online the class centroids and (ii) perform rehearsal using as memory stored samples of old classes. Additionally, DeepNNO uses the network at the previous learning step to compute a distillation loss [36], [16] on the network activations, reducing the catastrophic forgetting problem by preventing them from deviating from the features used to discern old classes.

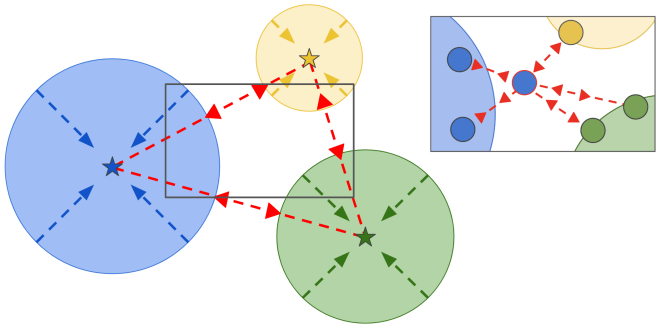


Fig. 2: Overview of the proposed global to local clustering. The global clustering (left) pushes sample representations closer to the centroid (star) of the class they belong to. The local clustering (right), instead, forces the neighborhood of a sample in the representation space to be semantically consistent, pushing away samples of other classes.

Finally, [22] updates online the rejection threshold during training with an heuristic rule that raises the threshold whenever the network predicts true positives or negatives and lowers it whenever the network predicts false positives or negatives. The final classification is held-out as in Eq.(3).

While we will base our architecture and classifier on [22], we argue that DeepNNO has two main drawbacks. First, the learned feature representation f is not forced to produce predictions clearly *localized* in a limited region of the metric space. Indeed, constraining the feature representations of a given class to a limited region of the metric space allows to have both more confident predictions on seen classes and producing clearer rejections also for images of unseen concepts. Second, having an heuristic strategy for setting the threshold is sub-optimal with no guarantees on the robustness of the choice. In the following, we will detail how we provide solutions to both problems.

C. Boosting Deep Open World Recognition

To obtain feature representations clearly localized in the metric space based on their semantic, we propose to use a pair of losses enforcing clustering. In particular, we use a *global* term which forces the network to map samples of the same class close to their centroid (Fig.2, left) and a *local* clustering term which constrains the neighborhood of a sample to be semantically consistent, i.e. to contain samples of the same class (Fig.2, right). In the following we describe the two clustering terms.

Global Clustering. The global clustering term aims to reduce the distance between the features of a sample with the centroid of its class. To model this, we took inspiration from what has been proposed in [34] and we employ a cross-entropy loss with the probabilities obtained through the distances among samples and class centroids. Formally, given a sample x and its class label c, we define the global clustering term as follows:

$$\ell_{GC}(x,c) = -\log \frac{s_c(x)}{\sum_{k\in\mathcal{K}_t} s_k(x)}. \quad (5)$$

The class-specific score $s_c(x)$ is defined as:

$$s_c(x) = \frac{e^{-\frac{1}{T}||f(x)-\mu_c||^2}}{\sum_{k\in\mathcal{C}_t} e^{-\frac{1}{T}||f(x)-\mu_k||^2}} \quad (6)$$

where T is a temperature value which allows us to control the behavior of the classifier. We set T as the variance of the activations in the feature space, σ^2, in order to normalize the representation space and increase the stability of the system. During training, σ^2 is the variance of the features extracted from the current batch while, at the same time, we keep an online global estimate of σ^2 that we use at test time. The class mean vectors μ_i with $i \in \mathcal{K}_t$ as well as σ^2 are computed in an online fashion, as in [22].

Local Clustering. To enforce that the neighborhood of a sample in the feature space is semantically consistent (i.e. given a sample x of a class c, the nearest neighbours of $f(x)$ belong to c), we employ the soft nearest neighbour loss [23], [24]. This loss has been proposed to measure the class-conditional entanglement of features in the representation space. In particular, it has been defined as:

$$\ell_{LC}(x,c,\mathcal{B}) = -\log \frac{\sum_{x_j\in\mathcal{B}_c\setminus\{x\}} e^{-\frac{1}{T}||f(x)-f(x_j)||^2}}{\sum_{x_k\in\mathcal{B}\setminus\{x\}} e^{-\frac{1}{T}||f(x)-f(x_k)||^2}} \quad (7)$$

where T refers to the temperature value, $\mathcal{B}$ is the current training batch, and $\mathcal{B}_c$ is the set of samples in the training batch belonging to class c. Instead of performing multiple learning steps to optimize the value of T as proposed in [24], we use as $T = \sigma^2$ as we do in Eq. 6.

Intuitively, given a sample x of a class c, a low value of the loss indicates that the nearest neighbours of $f(x)$ belong to c, while high values indicates the opposite (i.e. nearest neighbours belong to classes $i \in \mathcal{K}_t$ with $i \neq c$). Minimizing this objective allows to enforce the semantic consistency in the neighborhood of a sample in the feature space.

Reducing catastrophic forgetting through distillation. As highlighted in the previous sections, to avoid forgetting old knowledge, we want the feature extractor to preserve the behaviour learned in previous learning steps. To this extent, we follow standard rehearsal-based approaches for incremental learning [16], [37], [22], [38] and we introduce (i) a memory which stores the most relevant samples for classes in $\mathcal{K}_t$ and (ii) a distillation loss which enforces consistency among the features extracted by f and ones obtained by the feature extractor of the previous learning step, f_{t-1}. Formally, the distillation loss is computed as:

$$\ell_{DS}(x) = ||f(x) - f_{t-1}(x)||. \quad (8)$$

This loss is minimized only for incremental training steps, hence, only when $t > 1$.

Overall, we train the network to minimize on a batch of samples $\mathcal{B} = \{(x_1, c_1), \cdots, (x_{|\mathcal{B}|}, c_{|\mathcal{B}|})\}$ the following loss:

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{(x,c)\in\mathcal{B}} \ell_{GC}(x,c) + \lambda\, \ell_{LC}(x,c,\mathcal{B}) + \gamma\, \ell_{DS}(x) \quad (9)$$

with λ and γ hyperparameters weighting the different components. We set $\lambda = \gamma = 1$ in all experiments.

Learning to detect the unknown. In order to extend our NCM-based classifier to work on the open set scenario, we explicitly learn class-specific rejection criterions. As illustrated in Fig. 3, for each class c we define the *class-specific* threshold as the maximal distance Δ_c for which the sample belongs to c. Under this definition, our classifier is:

$$g(x) = \begin{cases} unk & \text{if } d(f(x),\mu_c) > \Delta_c,\ \forall c \in \mathcal{K}_t, \\ \text{argmin}_c d(f(x),\mu_c) & \text{otherwise} \end{cases} \quad (10)$$

with $d(x,y) = \frac{1}{\sigma^2}||x-y||^2$. Instead of heuristically estimating or fixing a maximal distance, we explicitly learn it for each class minimizing the following objective:

$$\ell_{MD}(x,c) = \sum_{k\in\mathcal{K}_t} \max(0, m\cdot(\frac{1}{\sigma^2}||f(x)-\mu_k||^2 - \Delta_k)) \quad (11)$$

where $m = -1$ if $c = k$ and $m = 1$ otherwise. The ℓ_{MD} loss leads to an increase of Δ_c if the distance from a sample belonging to the class c and the class centroid μ_c is greater than Δ_c. Instead, if a sample not belonging to c has a distance from μ_c less then Δ_c, it increases the value of Δ_c.

Overall, the training procedure of our method is made of two steps: in the first we train the feature extractor on the training set minimizing Eq. 9, while in the second we learn the distances Δ_c on a set of samples which we held-out from training set. To this extent, we split the samples of the memory

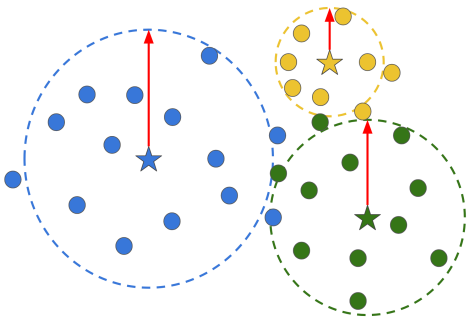


Fig. 3: Overview of the learning of the *class-specific* rejection thresholds. The small circles represent the samples in the held out set. The dashed circles, having radius the maximal distance (red), represent the limits beyond which a sample is rejected as a member of that class. As it can be seen, the class-specific threshold is learned to reduce the rejection errors. Best viewed in colors.

in two parts, one used for updating the feature extractor f and the centroids μ_c and the other part for learning the Δ_c values.

IV. Experiments

In this section, we first introduce the experimental setting and the metrics used for the evaluation, then we report results of our experiments and an ablation study of our contributions.

A. Experimental Setting

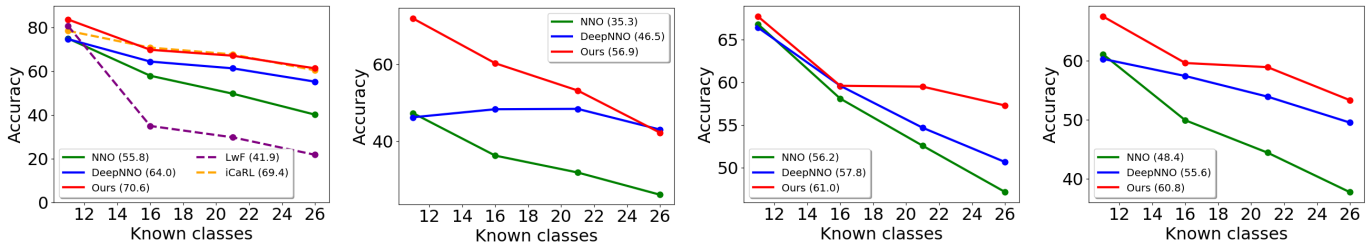
Datasets and Baselines. We assess the performance of our model on three datasets: RGB-D Object [26] Core50 [25] and CIFAR-100 [27]. The RGB-D Object dataset [26] is one of the most used dataset to evaluate the ability of a model to recognize daily-life objects. It contains 51 different semantic categories that we split in two parts in our experiments: 26 classes are considered as known categories, while the other 25 are the set of unknown classes. Among the 26 classes, we consider the first 11 classes as the initial training set and we incrementally add the remaining classes in 4 steps of 5 class each. As proposed in [26], we sub-sample the dataset taking one every fifth frame. For the experiments, we use the first train-test split among the original ones defined by the authors [26]. In each split one object instance from each class is chosen to be used in the test set and removed from the training set. This split provides nearly 35,000 training images and 7,000 test images. Core50 [25] is a recently introduced benchmark for testing continual learning methods in an egocentric setting. The dataset contains images of 50 objects grouped into 10 semantic categories. The images have been acquired on 11 different sequences with varying conditions. Following the standard protocol described in [25], we select the sequences 3, 7, 10 for the evaluation phase and use the remaining ones to train the model. Due to these differences in conditions between the sequences, Core50 represents a very challenging benchmark for object recognition. As in the RGB-D Object dataset, we split it into two parts: 5 classes are considered known and the other 5 as unknown. In the known set, the first 2 classes are considered as the initial training set. The others are incrementally added 1 class at a time. CIFAR-100 [27] is a standard benchmark for comparing incremental class learning algorithms [16]. It contains 100 different semantic categories. We follow previous works [22] splitting the dataset into 50 known and 50 unknown classes and considering 20 classes as the initial training set. Then, we incrementally add the remaining ones in steps of 10 classes. We evaluate the performance of our method in the OWR scenario comparing it to DeepNNO [22] and NNO [20], using the implementation in [22] for the latter. We further compare our method with two standard incremental class learning algorithms, namely LwF [39] (in the MC variant of [16]) and iCaRL [16]. Both LwF and iCaRL are designed for the closed world scenario, thus we use their performances as reference in that setting, without open-ended evaluation. For each dataset, we have randomly chosen five different sets of known and unknown classes. After fixing them, we run the experiments three times for each method. The results are obtained by averaging the results among each run and order.

Networks architectures and training protocols. Following previous works, we use a ResNet-18 architecture [40] for all the experiments. For RGB-D Object dataset and Core50, we train the network from scratch on the initial classes for 12 epochs and for 4 epochs in the incremental steps. For CIFAR-100, instead, we set the epochs to 120 for the initial learning stage and to 40 for each incremental step. We use a learning rate of 0.1 and batch size 128 for the RGB-D Object dataset, while we use 0.01 and 64 for Core50. We train the network using Stochastic Gradient Descent (SGD) with momentum 0.9 and a weight decay of 10^{-3} on both datasets.We resize the images of RGB-D Object dataset to 64×64 pixels and the images of Core50 to 128×128 pixels. We perform random cropping and mirroring for all the datasets. Moreover, for the set of held-out samples, we also perform color jittering varying brightness, hue and saturation. For the baselines, we use the same network architecture and training protocol defined in [22]. We also employ the same strategy for memory management, considering a fixed size of 2000 samples and constructing each batch by drawing 40% of the instances from memory. Differently from [22], we never see during training 20% of the samples present in memory, using them only to learn the values the class-specific threshold values Δ_k.

Metrics We use 3 standard metrics for comparing the performances of OWR methods. For the closed world we show the global accuracy *with* and *without* rejection option. Specifically, in the closed world *without rejection* setting, the model is tested only on the known set of classes, *excluding* the possibility to classify a sample as *unknown*. This scenario measures the ability of the model to correctly classify samples among the given set of classes. In the closed world *with rejection* scenario, instead, the model can either classify a sample among the known set of classes or categorize it as *unknown*. This scenario is more challenging than the previous one because samples belonging to the set of known classes might be misclassified as *unknowns*. For the open world we use the standard OWR metric defined in [20] as the average between the accuracy computed on the closed world *with rejection* scenario and the accuracy computed on the open set scenario (i.e. the accuracy on rejecting samples of unknown classes). Since the latter metric creates biases on the final score (i.e. a method rejecting every sample will achieve a 50% accuracy), we introduced the OWR-H as the harmonic mean between the accuracy on open set and the closed world with rejection scenarios to mitigate this bias.

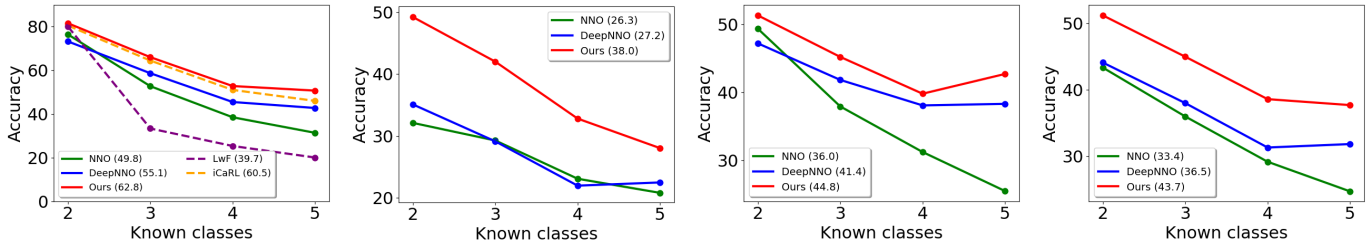
B. Quantitative results

We report the results on the RGB-D Object dataset in Fig. 4. Considering the closed world without rejection, reported in Fig. 4a, we note that our method is able to improve the feature representation, outperforming DeepNNO by 5.6% of accuracy on average and NNO by 14.8%. The reason for the improvement comes from the introduction of the global and local clustering loss terms, which allows the model to better aggregate samples of the same class and to better separate them from samples of other classes. Comparing our



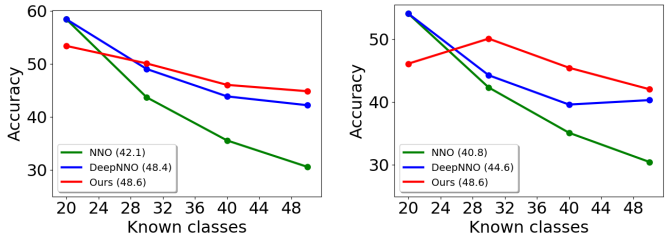
(a) Closed World Without Rejection (b) Closed World With Rejection (c) Open World Recognition Average (d) Open World Recognition Harmonic Mean

Fig. 4: Comparison of NNO [20], DeepNNO [22] and our method on RGB-D Object dataset [26]. The numbers in parenthesis denote the average accuracy among the different incremental steps.



(a) Closed World Without Rejection (b) Closed World With Rejection (c) Open World Recognition Average (d) Open World Recognition Harmonic Mean

Fig. 5: Comparison of NNO [20], DeepNNO [22] and our method on Core50 dataset [25]. The numbers in parenthesis denote the average accuracy among the different incremental steps.



(a) Open World Recognition Average (b) Open World Recognition Harmonic Mean

Fig. 6: Comparison of NNO [20], DeepNNO [22] and our method on Cifar dataset [27]. The numbers in parenthesis denote the average accuracy among the different steps.

model with the incremental class learning approaches LwF and iCaRL, we can see that our approach is highly competitive, surpassing LwF with a large gap while being comparable with the more effective iCaRL. We believe these are remarkable results given that the main goal of our model is not to purely extend its knowledge over time with new concepts. The comparison on the closed world with rejection, shown in Fig. 4b, demonstrates that our method is also more confident on the known classes, being able to reject a lower number of known samples. In particular, our method is more confident on the first incremental steps, and obtains, on average, an accuracy of 10.3% more than DeepNNO. Considering the open world metrics, our method is superior to previous works. From the results of OWR, reported in Fig. 4c, we see that our method reaches performance similar to DeepNNO in the first steps, while it outperforms it in the latest ones. However, considering the OWR-H (Fig. 4d), our method is better in all the incremental steps. This is because previous methods are biased towards rejecting more samples, as it is demonstrated by the lower closed world with rejection performance they achieve. On the contrary, our learned rejection criterion, coupled with our clustering losses, allows to achieve a better trade-off between the accuracy of open set and closed world with rejection. Overall, our method improves on average by 4.8% and 5.2% with respect to DeepNNO in the OWR and OWR-H metrics respectively.

In Fig. 5 we report the results on the Core50 [25] dataset. Similarly to the RGB-D Object dataset, our method achieves competitive results with respect to incremental class learning algorithms designed for the closed world scenario, remarkably outperforming iCaRL by 4.7% of accuracy in the last incremental step. It also achieves a superior performance in both closed world, without and with rejection option with respect to state-of-the-art OWR algorithms, outperforming NNO by 13.01% and DeepNNO by 7.74% on average in the first (Fig. 5a) and by more than 10% for both NNO and DeepNNO in the latter (Fig. 5b). In particular, it is worth noting how both DeepNNO and NNO are not able to properly model the confidence threshold, rejecting most of the sample of the known classes. Indeed, by including the rejection option the accuracy drops to 27.2% and 26.3% respectively for DeepNNO and NNO, while our model reaches an average accuracy of 38.0%. In Fig. 5c and Fig. 5d, we report the OWR performances (standard and harmonic) on Core50. Our method outperforms DeepNNO by 3.4% and 7.2% in average respectively in standard OWR and OWR-H metrics, confirming the effectiveness of our clustering losses and learned class-specific maximal distances.

Finally, in Fig. 6 we report the results on the CIFAR-100 dataset in terms of the OWR (Fig. 6a) and OWR-H metrics

Method	Known Classes				OWR	
	11	16	21	26	[20]	H
GC	66.0	57.3	58.6	53.3	58.8	58.7
LC	64.1	56.0	57.9	56.4	58.6	58.4
Triplet	62.1	54.9	54.8	49.5	55.4	55.4
GC + LC	**67.7**	**59.6**	**59.5**	**57.3**	**61.0**	**60.8**

TABLE I: Ablation study on the global (GC), local clustering (LC) and Triplet loss on the OWR metric. The right column shows the average OWR-H over all steps.

Method	Class specific	Multi stage	Known	Unknown	Diff.
DeepNNO [22]			84.4	98.8	14.4
Ours	✓		83.0	98.6	15.6
		✓	4.4	26.9	22.6
	✓	✓	27.4	65.2	**37.8**

TABLE II: Rejection rates of different techniques for detecting the unknowns. The results are computed using the same feature extractor on the RGB-D Object dataset.

(Fig. 6b). Even in this benchmark, our approach achieves superior results, on average, than previous methods. Our model achieves lower performances with respect to NNO and DeepNNO only in the initial training stage. However, in the incremental learning steps our model clearly outperforms both methods, demonstrating its ability to learning and recognizing in an open-world without forgetting old classes. In fact, considering the incremental steps, the average improvement of our model over NNO are of 10% in both OWR and OWR-H metrics, while over DeepNNO are of 2% for the OWR and 4.5% for the OWR-H metric.

C. Ablation study.

Our approach is mainly built on three components, i.e. global clustering loss (GC), local clustering loss (LC) and the learned class-specific rejection thresholds.In this section we analyze each proposed contribution. We start from the two clustering losses and then we compare the choice we made for the rejection with other common choices.

Global and local clustering. In Table I we compare the two clustering terms considering the open world recognition metrics in the RGB-D Object dataset. By analyzing the two loss terms separately we see that, on average, they show similar performance. In particular, using only the global clustering (GC) term we achieve slightly better performance on the first three incremental steps, while on the fourth the local clustering (LC) term is better. However, the best performance on every step is achieved by combining the global and local clustering terms (GC + LC). This demonstrates that the two losses provide different contributions, being complementary to learn a representation space which properly clusters samples of the same classes while better detecting unknowns. Finally, since our loss functions and triplet loss [41] share the same objective, i.e. building a metric space where samples sharing the same semantic are closer then ones with different semantics, we report in Table I also the results achieved by replacing our loss terms with a triplet loss [41]. As the Table shows, the triplet loss formulation (Triplet) fails in reaching competitive results with respect to our full objective function, with a gap of more than 5% in both standard OWR metric and OWR harmonic mean. Notably, it achieves lower results also with respect to all of the loss terms in isolation and the superior performances of LC confirm the advantages of SNNL-based loss functions with respect to triplets, as shown in [24].

Detecting the Unknowns. In Table II we report a comparison of different strategies to reject samples on the RGB-D Object dataset [26]. In particular, using the same feature extractor, we compare the proposed method to learn the class-specific maximal distances with three baselines: (i) we adopt the strategy proposed by DeepNNO [22], (ii) we learn class-specific maximal distances but during training (i.e. without our two-stage pipeline) and (iii) we learn a single maximal distance which applies to all classes using our two-stage training strategy. The comparison is performed considering the difference of the rejection rates on the known and unknown samples. For the known class samples, we report the percentage of correctly classified samples in the closed-world that are rejected when the rejection option is included. We intentionally remove the wrongly classified samples since we want to isolate rejection mistakes from classification ones. On the unknown samples, we report the open-set accuracy, i.e. the percentage of rejected samples among all the unknown ones. In the third column, we report the difference among the open-set accuracy and the rejection rate on known samples. Ideally, the difference should be as close as possible to 100%, since we want a 100% rejection rate on unknown class samples and 0% on the known class ones. From the table, we see that the highest gap is achieved by the class-specific maximal distance with the two-stage pipeline we proposed, which rejects 27.4% of known class samples and 65.2% on the unknown ones. The gap with the other strategies is remarkable. Using the two stage-pipeline but a class-generic maximal distance leads to a low rejection rate, both on known and unknown samples, achieving a difference of 22.6%, which is 15.2% less than using a class-specific distance. On the other hand, estimating the confidence threshold as proposed in DeepNNO [22] or without our two-stage pipeline provides a very high rejection rate, both on known and unknown classes, which lead to a difference of 14.4% and 15.6% for DeepNNO and the single-stage strategy respectively, the lowest two among the four strategies. In fact, computing the thresholds using only the training set biases the rejection criterion on the overconfidence that the method has acquired on this set. At test time, this causes the model to consider the different data distribution (caused by e.g. different object instances) as a source for rejection even if the actual concept present in the input is known. Using the two-stage process allows to overcome this bias, tuning the rejection criterion on unseen data on which the model cannot be overconfident.

V. Conclusion and Future Works

In this work we presented an approach to tackle the open world recognition problem in robot vision. As in previous works, we base our approach on a NCM classifier built on top of deep features, and we boost the OWR performances of this framework by training the deep architecture to minimize a global to local semantic clustering loss. This loss allows to reduce distances of samples of the same class in the feature

space while separating them from points belonging to other classes, thus better detecting unknown concepts. Moreover, we avoid heuristic estimates of a rejection criterion for detecting unknowns by explicitly learning class-specific distances beyond which a sample is rejected. Quantitative and qualitative analysis on standard recognition benchmarks show the efficacy of our approach and choices, outperforming previous state-of-the-art OWR algorithms. While here we considered the OWR scenario, there are still many directions that could be explored for enabling robots to learn autonomously in the real word. One could be extending the approach to the Web-aided OWR scenario considered in [22]. In particular, when training images are autonomously retrieved from the Web, they come with an inherent noisy labelling. In this context our model will not be able to work well since the noise will be transmitted to the class-specific cluster center hampering the efficacy of the recognition model. Moreover, it would be interesting to analyze the OWR problem in an active learning context [42]. Finally, another interesting research direction would be extending OWR approaches to more complex tasks, such as object detection and segmentation, where the ability to distinguish among background/stuffs [43] and actual unknown objects and the background shift problem illustrated in [44].

REFERENCES

[1] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with cnns for frame-to-frame ego-motion estimation," *RA-L-16*, vol. 1, no. 1.

[2] O. H. Jafari, O. Groth, A. Kirillov, M. Y. Yang, and C. Rother, "Analyzing modular cnn architectures for joint depth prediction and semantic segmentation," in *ICRA-17*.

[3] M. Mancini, G. Costante, P. Valigi, T. A. Ciarfuglia, J. Delmerico, and D. Scaramuzza, "Toward domain independence for learning-based monocular depth estimation," *RA-L-17*, vol. 2, no. 3.

[4] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *IROS-16*.

[5] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *IJRR-18*, vol. 37, no. 4-5.

[6] M. Schwarz, A. Milan, A. S. Periyasamy, and S. Behnke, "Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter," *IJRR-18*, vol. 37, no. 4-5.

[7] G. L. Oliveira, C. Bollen, W. Burgard, and T. Brox, "Efficient and robust deep networks for semantic segmentation," *IJRR-18*, vol. 37.

[8] M. Wulfmeier, A. Bewley, and I. Posner, "Addressing appearance change in outdoor robotics with adversarial domain adaptation," in *IROS-17*.

[9] ——, "Incremental adversarial domain adaptation for continually changing environments," in *ICRA-18*.

[10] M. Mancini, H. Karaoguz, E. Ricci, P. Jensfelt, and B. Caputo, "Kitting in the wild through online domain adaptation," *IROS-18*.

[11] M. Mancini, S. R. Bulo, B. Caputo, and E. Ricci, "Robust place categorization with deep domain generalization," *RA-L-18*, vol. 3, no. 3.

[12] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[13] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *T-PAMI-12*, vol. 35, no. 7.

[14] V. Fragoso, P. Sen, S. Rodriguez, and M. Turk, "Evsac: accelerating hypotheses generation by modeling matching scores with extreme value theory," in *ICCV-13*.

[15] F. Li and H. Wechsler, "Open set face recognition using transduction," *T-PAMI-05*, vol. 27, no. 11.

[16] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *CVPR-17*.

[17] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori, "Incremental semiparametric inverse dynamics learning," in *ICRA-16*.

[18] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta, "Incremental robot learning of new objects with fixed update time," in *ICRA-17*.

[19] S. Valipour, C. Perez, and M. Jagersand, "Incremental learning for robot perception through hri," in *IROS-17*.

[20] A. Bendale and T. Boult, "Towards open world recognition," in *CVPR-15*.

[21] R. De Rosa, T. Mensink, and B. Caputo, "Online open world recognition," *arXiv:1604.02275*, 2016.

[22] M. Mancini, H. Karaoguz, E. Ricci, P. Jensfelt, and B. Caputo, "Knowledge is never enough: Towards web aided deep open world recognition," in *ICRA-19*.

[23] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Artificial Intelligence and Statistics*, 2007.

[24] N. Frosst, N. Papernot, and G. E. Hinton, "Analyzing and improving representations with the soft nearest neighbor loss," in *ICML-19*.

[25] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *CoRL-17*.

[26] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *ICRA-11*.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical report, University of Toronto., Tech. Rep., 2009.

[28] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford *et al.*, "The limits and potentials of deep learning for robotics," *IJRR-18*, vol. 37, no. 4-5.

[29] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information Fusion*, vol. 58, pp. 52–68, 2020.

[30] F. Cermelli, M. Mancini, E. Ricci, and B. Caputo, "The rgb-d triathlon: Towards agile visual toolboxes for robots," *IROS-19*.

[31] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale, "Teaching icub to recognize objects using deep convolutional neural networks," in *Machine Learning for Interactive Systems*, 2015.

[32] G. I. Parisi, J. Tani, C. Weber, and S. Wermter, "Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization," *Frontiers in neurorobotics*, vol. 12, 2018.

[33] M. Lagunes-Fortiz, D. Damen, and W. Mayol-Cuevas, "Learning discriminative embeddings for object recognition on-the-fly," in *ICRA-19*.

[34] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Metric learning for large scale image classification: Generalizing to new classes at near-zero cost," in *ECCV-12*.

[35] S. Guerriero, B. Caputo, and T. Mensink, "Deep nearest class mean classifiers," in *ICLR-WS*, 2018.

[36] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv 1503.02531*, 2015.

[37] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *ECCV-18*.

[38] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *ECCV-18*.

[39] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR-16*.

[41] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks."

[42] G. I. Parisi and C. Kanan, "Rethinking continual learning for autonomous agents and robots," *arXiv preprint arXiv:1907.01929*, 2019.

[43] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014.

[44] F. Cermelli, M. Mancini, S. R. Bulò, E. Ricci, and B. Caputo, "Modeling the background for incremental learning in semantic segmentation," *CVPR-20*.