Incremental Robot Learning of New Objects with Fixed Update Time

Raffaello Camoriano*†‡⋄, Giulia Pasquale*†‡⋄, Carlo Ciliberto⋄,
Lorenzo Natale†, Lorenzo Rosasco⋄‡ and Giorgio Metta†

***Abstract*— We consider object recognition in the context of lifelong learning, where a robotic agent learns to discriminate between a growing number of object classes as it accumulates experience about the environment. We propose an incremental variant of the Regularized Least Squares for Classification (RLSC) algorithm, and exploit its structure to seamlessly add new classes to the learned model. The presented algorithm addresses the problem of having an unbalanced proportion of training examples per class, which occurs when new objects are presented to the system for the first time.**

We evaluate our algorithm on both a machine learning benchmark dataset and two challenging object recognition tasks in a robotic setting. Empirical evidence shows that our approach achieves comparable or higher classification performance than its batch counterpart when classes are unbalanced, while being significantly faster.

I. Introduction

In order for autonomous robots to operate in unstructured environments, several perceptual capabilities are required. Most of these skills cannot be hard-coded in the system beforehand, but need to be developed and learned over time as the agent explores and acquires novel experience. As a prototypical example of this setting, in this work we consider the task of visual object recognition in robotics: Images depicting different objects are received one frame at a time, and the system needs to incrementally update the internal model of known objects as new examples are gathered.

In the last few years, machine learning has achieved remarkable results in a variety of applications for robotics and computer vision [1], [2], [3]. However, most of these methods have been developed for off-line (or "batch") settings, where the entire training set is available beforehand. The problem of updating a learned model online has been addressed in the literature [4], [5], [6], [7], but most algorithms proposed in this context do not take into account challenges that are characteristic of realistic lifelong learning applications. Specifically, in online classification settings, a major challenge is to cope with the situation in which a novel class is added to the model. Indeed, $1)$ most learning algorithms require the number of classes to be known beforehand and not grow indefinitely, and $2)$ the imbalance between the few examples of the new class (potentially just one) and the many examples of previously learned classes can lead to unexpected and undesired behaviors [8]. More precisely, in this work we theoretically and empirically observe that the new and under-represented class is likely to be ignored by the learned model in favor of classes for which more training examples have already been observed, until a sufficient number of examples are provided also for such class.

Several methods have been proposed in the literature to deal with class imbalance in the batch setting by "rebalancing" the misclassification errors accordingly [8], [9], [10]. However, as we point out in this work, rebalancing cannot be applied to the online setting without re-training the entire model from scratch every time a new example is acquired. This would incur in computational learning times that increase at least linearly in the number of examples, which is clearly not feasible in scenarios in which training data grows indefinitely.

In this work we propose a novel method that learns incrementally both with respect to the number of examples and classes, and accounts for potential class unbalance. Our algorithm builds on a recursive version of Regularized Least Squares for Classification (RLSC) [11], [12] to achieve fixed incremental learning times when adding new examples to the model, while efficiently dealing with imbalance between classes. We evaluate our approach on a standard machine learning benchmark for classification and two challenging visual object recognition datasets for robotics. Our results highlight the clear advantages of our approach when classes are learned incrementally.

The paper is organized as follows: Sec. II overviews related work on incremental learning and class imbalance. In Sec. III we introduce the learning setting, discussing the impact of class imbalance and presenting two approaches that have been adopted in the literature to deal with this problem. Sec. IV reviews the recursive RLSC algorithm. In Sec. V we build on previous Sec. III and IV to derive the approach proposed in this work, which extends recursive RLSC to allow for the addition of new classes with fixed update time, while dealing with class imbalance. In Sec. VI we report on the empirical evaluation of our method, concluding the paper in Sec. VII.

II. Related Work

Incremental Learning. The problem of learning from a continuous stream of data has been addressed in the literature from multiple perspectives. The simplest strategy is to retrain the system on the updated training set, whenever a new example is received [13], [14]. The model from the previous iteration can be used as an initialization to learn the new predictor, reducing training time. These approaches require to store all the training data, and to retrain over all the points

*Equal contribution

†iCub Facility, Istituto Italiano di Tecnologia (IIT), Genoa, Italy. Email: {name.surname}@iit.it

⋄LCSL, IIT and MIT, Cambridge, USA.

‡DIBRIS, Università degli Studi di Genova, Genoa, Italy.

at each iteration. Their computational complexity increases at least linearly with the number of examples.

Incremental approaches that do not require to keep previous data in memory can be divided in *stochastic* and *recursive* methods. Stochastic techniques assume training data to be randomly sampled from an unknown distribution and offer asymptotic convergence guarantees to the ideal predictor [6]. However, it has been empirically observed that these methods do not perform well when seeing each training point only once, hence requiring to perform "multiple passes" over the data [15], [16]. This problem has been referred to as the "catastrophic effect of forgetting" [4], which occurs when training a stochastic model only on new examples while ignoring previous ones, and has recently attracted the attention of the Neural Networks literature [17], [7].

Recursive techniques are based, as the name suggests, on a recursive formulation of batch learning algorithms. Such formulation typically allows to compute the current model in closed form (or with few operations independent of the number of examples) as a combination of the previous model and the new observed example [5], [18]. As we discuss in more detail in Sec. IV, the algorithm proposed in this work is based on a recursive method.

Learning with an Increasing Number of Classes. Most classification algorithms have been developed for batch settings and therefore require the number of classes to be known a priori. However, this assumption is often broken in incremental settings, since new examples could belong to previously unknown classes. The problem of dealing with an increasing number of classes has been addressed in the contexts of transfer learning or *learning to learn* [19]. These settings consider a scenario where T linear predictors have been learned to model T classes. Then, when a new class is observed, the associated predictor is learned with the requirement of being "close" to a linear combination of the previous ones [20], [21], [22]. Other approaches have been recently proposed where a class hierarchy is built incrementally as new classes are observed, allowing to create a taxonomy and exploit possible similarities among different classes [13], [23]. However, all these methods are not incremental in the number of examples and require to retrain the system every time a new point is received.

Class Imbalance. The problems related to class imbalance were previously studied in the literature [8], [10], [9] and are addressed in Sec. III. Methods to tackle this issue have been proposed, typically re-weighting the misclassification loss [20] to account for class imbalance. However, as we discuss in Sec. V-B for the case of the square loss, these methods cannot be implemented incrementally. This is problematic, since imbalance among multiple classes often arises in online settings, even if temporarily, for instance when examples of a new class are observed for the first time.

III. Classification Setting and the Effect of Class Imbalance

In this section, we introduce the learning framework adopted in this work and describe the disrupting effect of imbalance among class labels. For simplicity, in the following we consider a binary classification setting, postponing the extension to multiclass classification to the end of the section. We refer the reader to [9] for more details about the Statistical Learning Theory for classification.

A. Optimal Bayes Classifier and its Least Squares Surrogate

Let us consider a binary classification problem where input-output examples are sampled randomly according to a distribution ρ over $\mathcal{X} \times \{-1, 1\}$. The goal is to learn a function $b^* : \mathcal{X} \to \{-1, 1\}$ minimizing the overall *expected* classification error

$$b^* = \underset{b:\mathcal{X}\to\{-1,1\}}{\operatorname{argmin}} \int_{\mathcal{X}\times\{-1,1\}} \mathbf{1}(b(x) - y)\ d\rho(x, y), \quad (1)$$

given a finite set of observations $\{x_i, y_i\}_{i=1}^n$, $x_i \in \mathcal{X}$, $y_i \in \{-1, 1\}$ randomly sampled from ρ. Here $\mathbf{1}(s) : \mathbb{R} \to \{0, 1\}$ denotes the binary function taking value 0 if $s = 0$ and 1 otherwise. The solution to Eq. (1) is called the *optimal Bayes classifier* and it can be shown to satisfy the equation

$$b^*(x) = \begin{cases} 1 & \text{if } \rho(1|x) > \rho(-1|x) \\ -1 & \text{otherwise} \end{cases}, \quad (2)$$

for all $x \in \mathcal{X}$. Here we have denoted by $\rho(y|x)$ the *conditional* distribution of y given x and in this work we will denote by $\rho(x)$ the *marginal* distribution of x, such that by Bayes' rule $\rho(x, y) = \rho(y|x)\rho(x)$. Computing good estimates of $\rho(y|x)$ typically requires large training datasets and is often unfeasible in practice. Therefore, a so-called *surrogate* problem (see [9], [24]) is usually adopted to simplify the optimization problem at Eq. (1) and asymptotically recover the optimal Bayes classifier. In this sense, one well-known surrogate approach is to consider the least squares expected risk minimization

$$f^* = \underset{f:\mathcal{X}\to\mathbb{R}}{\operatorname{argmin}} \int_{\mathcal{X}\times\{-1,1\}} (y - f(x))^2\ d\rho(x, y). \quad (3)$$

The solution to Eq. (3) allows to recover the optimal Bayes classifier. Indeed, for any $f : \mathcal{X} \to \mathbb{R}$ we have

$$\int (y - f(x))^2 d\rho(x, y) = \int\int (y - f(x))^2 d\rho(y|x) d\rho(x)$$
$$= \int \left[(1 - f(x))^2 \rho(1|x) + (f(x) + 1)^2 \rho(-1|x)\right] d\rho(x),$$

which implies that the minimizer of Eq. (3) satisfies

$$f^*(x) = 2\rho(1|x) - 1 = \rho(1|x) - \rho(-1|x) \quad (4)$$

for all $x \in \mathcal{X}$. The optimal Bayes classifier can be recovered from f^* by taking its sign: $b^*(x) = sign(f^*(x))$. Indeed, $f^*(x) > 0$ if and only if $\rho(1|x) > \rho(-1|x)$.

Empirical Setting. When solving the problem in practice, we are provided with a finite set $\{x_i, y_i\}_{i=1}^n$ of training examples. In these settings the typical approach is to find an estimator $\hat{f}$ of f^* by minimizing the *regularized empirical risk*

$$\hat{f} = \underset{f:\mathcal{X}\to\mathbb{R}}{\operatorname{argmin}} \quad \frac{1}{n}\sum_{i=1}^n (y_i - f(x_i))^2 + R(f), \quad (5)$$

where R is a so-called *regularizer* preventing the solution $\hat{f}$ to overfit. Indeed, it can be shown [9], [25] that, under mild assumptions on the distribution ρ, it is possible for $\hat{f}$ to converge in probability to the ideal f^* as the number of training points grows indefinitely. In Sec. IV we review a method to compute $\hat{f}$ in practice, both in the batch and in the online settings.

B. The Effect of Unbalanced Data

The classification rule at Eq. (2) associates every $x \in \mathcal{X}$ to the class y with highest likelihood $\rho(y|x)$. However, in settings where the two classes are not balanced this approach could lead to unexpected and undesired behaviors. To see this, let us denote $\gamma = \rho(y=1) = \int_{\mathcal{X}} d\rho(y=1,x)$ and notice that, by Eq. (2) and the Bayes' rule, an example x is labeled $y = 1$ whenever

$$\rho(x|1) > \rho(x|-1)\frac{(1-\gamma)}{\gamma}. \tag{6}$$

Hence, when γ is close to one of its extremal values 0 or 1 (i.e. $\rho(y=1) \gg \rho(y=-1)$ or vice-versa), one class becomes clearly preferred with respect to the other and is almost always selected.

In Fig. 1 we report an example of the effect of unbalanced data by showing how the decision boundary (white dashed curve) of the optimal Bayes classifier from Eq. (2) varies as γ takes values from 0.5 (balanced case) to 0.9 (very unbalanced case). As it can be noticed, while the classes maintain the same shape, the decision boundary is remarkably affected by the value of γ.

Clearly, in an online robotics setting this effect could be critically suboptimal for two reasons: 1) We would like the robot to recognize with high accuracy even objects that are less common to be seen. 2) In incremental settings, whenever a novel object is observed for the first time, only few training examples are available (in the extreme case, just one) and we need a loss weighting fairly also underrepresented classes.

C. Rebalancing the Loss

In this paper, we consider a general approach to "rebalancing" the classification loss of the standard learning problem of Eq. (1), similar to the ones in [8], [9]. We begin by noticing that in the balanced setting, namely for $\gamma = 0.5$, the classification rule at Eq. (6) is equivalent to assigning class 1 whenever $\rho(x|1) > \rho(x|-1)$ and vice-versa. Here we want to slightly modify the misclassification loss in Eq. (1) to recover this same rule also in unbalanced settings. To do so, we propose to apply a weight $w(y) \in \mathbb{R}$ to the loss $\mathbf{1}(b(x) - y)$, obtaining the problem

$$b^*_w = \underset{b:\mathcal{X}\to\{-1,1\}}{\operatorname{argmin}} \int_{\mathcal{X}\times\{-1,1\}} w(y)\mathbf{1}(b(x)-y)\ d\rho(x,y).$$

Analogously to the non-weighted case, the solution to this problem is

$$b^*_w(x) = \begin{cases} 1 & \text{if } \rho(1|x)w(1) > \rho(-1|x)w(-1) \\ -1 & \text{otherwise} \end{cases}. \tag{7}$$

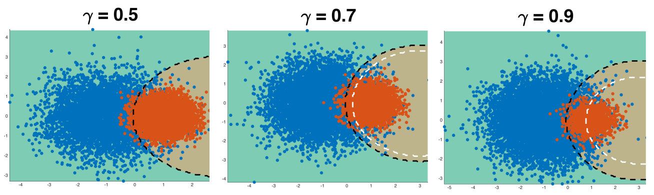

Fig. 1: Bayes decision boundaries for standard (dashed white line) and rebalanced (dashed black line) binary classification loss for multiple values of $\gamma = \rho(y=1)$ from 0.5 to 0.9. Data are sampled according to a Gaussian $\rho(x|y) \sim \mathcal{N}(\mu_y, \sigma_y)$ with $\mu_1 = (-1,0)^\top$, $\mu_{-1} = (1,0)^\top$, $\sigma_1 = 1$ and $\sigma_{-1} = 0.3$. The boundaries coincide when $\gamma = 0.5$ (balanced data), while they separate as γ increases.

In this work we take the weights w to be $w(1) = 1/\gamma$ and $w(-1) = \frac{1}{1-\gamma}$. Indeed, from the fact that $\rho(y|x) = \rho(x|y)(\rho(y)/\rho(x))$ we have that the rule at Eq. (7) is equivalent to

$$b^*_w(x) = \begin{cases} 1 & \text{if } \rho(x|1) > \rho(x|-1) \\ -1 & \text{otherwise} \end{cases}, \tag{8}$$

which corresponds to the (unbalanced) optimal Bayes classifier in the case $\gamma = 0.5$, as desired.

Fig. 1 compares the unbalanced and rebalanced optimal Bayes classifiers for different values of γ. Notice that rebalancing leads to solutions that are invariant to the value of γ (compare the black decision boundary with the white one).

D. Rebalancing and Recoding the Least Squares Loss

Interestingly, the strategy of changing the weight of the classification error loss can be naturally extended to the least squares surrogate. If we consider the weighted least squares problem,

$$f^*_w = \underset{f:\mathcal{X}\to\mathbb{R}}{\operatorname{argmin}} \int_{\mathcal{X}\times\{-1,1\}} w(y)(y - f(x))^2 d\rho(x,y), \tag{9}$$

we can again recover the (weighted) rule $b^*_w(x) = sign(f^*_w(x))$ like in the non-weighted setting. Indeed, by direct calculation it follows that Eq. (9) has solution

$$f^w_*(x) = \frac{\rho(1|x)w(1) - \rho(-1|x)w(-1)}{\rho(1|x)w(1) + \rho(-1|x)w(-1)}. \tag{10}$$

If we assume $w(1) > 0$ and $w(-1) > 0$ (as in this work), the denominator of Eq. (10) is always positive and therefore $sign(f^*_w(x)) > 0$ if and only if $\rho(1|x)w(1) > \rho(-1|x)w(-1)$, as desired.

Coding. An alternative approach to recover the rebalanced optimal Bayes classifier via least squares surrogate is to apply a suitable coding function to the class labels $y = \{-1, 1\}$, namely

$$f^*_c = \underset{f:\mathcal{X}\to\mathbb{R}}{\operatorname{argmin}} \int_{\mathcal{X}\times\{-1,1\}} (c(y) - f(x))^2 d\rho(x,y), \tag{11}$$

where $c : \{-1, 1\} \to \mathbb{R}$ maps the labels y into scalar codes $c(y) \in \mathbb{R}$. Analogously to the unbalanced (and uncoded) case, the solution to Eq. (11) is

$$f_c^*(x) = c(1)\rho(1|x) - c(-1)\rho(-1|x), \tag{12}$$

which, for $c(y) = w(y)$, corresponds to the numerator of Eq. (10). Therefore, the optimal (rebalanced) Bayes classifier is recovered again by $b_w^*(x) = sign(f_c^*(x))$.

E. Multiclass Rebalancing and Recoding

In the multiclass setting, the optimal Bayes decision rule corresponds to the function $b^* : \mathcal{X} \to \{1, \dots, T\}$, assigning a label $t \in \{1, \dots, T\}$ to $x \in \mathcal{X}$ when $\rho(t|x) > \rho(s|x)\ \forall s \neq t$, with $t, s \in \{1, \dots, T\}$. Consequently, the rebalanced decision rule would assign class t, whenever $\rho(y = t|x)w(t) > \rho(y = s|x)w(s)\ \forall s \neq t$, where the function $w : \{1, \dots, T\} \to \mathbb{R}$ assigns a weight to each class. Generalizing the binary case, in this work we set $w(t) = 1/\rho(y = t)$, where we denote $\rho(y = t) = \int_{\mathcal{X}} d\rho(t, x)$, for each $t \in \{1, \dots, T\}$.

In multiclass settings, the surrogate least squares classification approach is recovered by adopting a *1-vs-all* strategy, formulated as the vector-valued problem

$$f^* = \underset{f:\mathcal{X}\to\mathbb{R}^T}{\operatorname{argmin}} \int_{\mathcal{X}\times\{1,\dots,T\}} \|e_y - f(x)\|^2 d\rho(x, y), \tag{13}$$

where $e_t \in \mathbb{R}^T$ is a vector of the canonical basis $\{e_1, \dots, e_T\}$ of $\mathbb{R}^T$ (with the t-th coordinate equal to 1 and the remaining 0). Analogously to the derivation of Eq. (4), it can be shown that the solution to this problem corresponds to $f^*(x) = (\rho(1|x), \dots, \rho(T|x))^\top \in \mathbb{R}^T$ for all $x \in \mathcal{X}$. Consequently, we recover the optimal Bayes classifier by

$$b^*(x) = \underset{t=1,\dots,T}{\operatorname{argmax}} f(x)_t, \tag{14}$$

where $f(x)_t$ denotes the t-th entry of the vector $f(x) \in \mathbb{R}^T$.

The extensions of recoding and rebalancing approaches to this setting follow analogously to the binary setting discussed in Sec. III-D. In particular, the coding function $c : \{e_1, \dots, e_T\} \to \mathbb{R}$ consists in mapping a vector of the basis e_t to $c(e_t) = e_t/\rho(y = t)$.

Note. In previous sections we presented the analysis on the binary case by considering a $\{-1, 1\}$ coding for class labels. This was done to offer a clear introduction to the classification problem, since we need to solve a single least squares problem to recover the optimal Bayes classifier. Alternatively, we could have followed the approach introduced in this section where classes have labels $y = \{1, 2\}$ and adopt surrogate labels $e_1 = [1, 0]^\top$ and $e_2 = [0, 1]^\top$. This would have led to training two distinct classifiers and choosing the predicted class as the $argmax$ of their scores, according to Eq. (14). The two approaches are clearly equivalent since the Bayes classifier corresponds respectively to the inequalities $\rho(1|x) > \rho(-1|x)$ or $\rho(1|x) > \rho(2|x)$.

IV. RLSC and Recursive Formulation

In this section we review the standard algorithm for Regularized Least Squares Classification (RLSC) and its recursive formulation used for incremental updates.

A. Regularized Least Squares for Classification

We address the problem of solving the empirical risk minimization introduced in Eq. (5) in the multiclass setting. Let $\{x_i, y_i\}_{i=1}^n$ be a finite training set, with inputs $x_i \in \mathcal{X} = \mathbb{R}^d$ and labels $y_i \in \{1, \dots, T\}$. In this work, we will assume a linear model for the classifier $\hat{f}$, namely $f(x) = W^\top x$, with W a matrix in $\mathbb{R}^{d\times T}$. We can rewrite Eq. (5) in matrix notation as

$$\widehat{W} = \underset{W\in\mathcal{R}^{d\times T}}{\operatorname{argmin}} \|Y - XW\|_F^2 + \lambda\|W\|_F^2 \tag{15}$$

with $\lambda > 0$ the *regularization parameter* and $X \in \mathbb{R}^{n\times d}$ and $Y \in \mathbb{R}^{n\times T}$ the matrices whose i-th rows correspond respectively to $x_i \in \mathbb{R}^d$ and $e_{y_i} \in \mathbb{R}^T$. We denote by $\|\cdot\|_F^2$ the squared Frobenius norm of a matrix (i.e. the sum of its squared entries).

The solution to Eq. (15) is

$$\widehat{W} = (X^\top X + \lambda I_d)^{-1} X^\top Y \in \mathbb{R}^{d\times T}, \tag{16}$$

where I_d is the $d \times d$ identity matrix (see for instance [26]).

Prediction. According to the rule introduced in Sec. III-E, a given $x \in \mathbb{R}^d$ is classified according to

$$\hat{b}(x) = \underset{i=1,\dots,T}{\operatorname{argmax}}\ \hat{f}(x)_i = \underset{i=1,\dots,T}{\operatorname{argmax}}\ (\widehat{W}^{(i)})^\top x, \tag{17}$$

with $\widehat{W}^{(i)} \in \mathbb{R}^d$ denoting the i-th column of $\widehat{W}$.

B. Recursive Formulation

The closed form for the solution at Eq. (16) allows to derive a recursive formulation to incrementally update $\widehat{W}$ in fixed time as new training examples are observed [5]. Consider a learning process where training data are provided to the system one at a time. At iteration k we need to compute $W_k = (X_k^\top X_k + \lambda I_d)^{-1} X_k^\top Y_k$, where $X_k \in \mathbb{R}^{k\times d}$ and $Y_k \in \mathbb{R}^{k\times T}$ are the matrices whose rows correspond to the first k training examples. The computational cost for evaluating W_k according to Eq. (16) is $O(kd^2)$ (for the matrix products) and $O(d^3)$ (for the inversion). This is undesirable in an online setting where k can grow indefinitely. To this end, we now review how W_k can be computed incrementally from W_{k-1} in $O(Td^2)$. To see this, first notice that, by construction,

$$X_k = [X_{k-1}^\top, x_k]^\top \qquad Y_k = [Y_{k-1}^\top, e_{y_k}]^\top,$$

and therefore, if we denote $A_k = X_k^\top X_k + \lambda I_d$ and $b_k = X_k^\top Y_k$, we obtain the recursive formulations

$$\begin{aligned} A_k &= X_k^\top X_k + \lambda I_d \\ &= X_{k-1}^\top X_{k-1} + x_k^\top x_k + \lambda I_d \\ &= A_{k-1} + x_k^\top x_k + \lambda I_d \end{aligned} \tag{18}$$

and

$$b_k = X_k^\top Y_k = X_{k-1}^\top Y_{k-1} + x_k e_{y_k}^\top = b_{k-1} + x_k e_{y_k}^\top. \quad (19)$$

Computing b_k from b_{k-1} requires $O(d)$ operations (since e_{y_k} has all zero entries but one). Computing A_k from A_{k-1} requires $O(d^2)$, while the inversion A_k^{-1} requires $O(d^3)$. To reduce the cost of the (incremental) inversion, we recall that for a positive definite matrix A_k for which its Cholesky decomposition $A_k = R_k^\top R_k$ is known (with $R^k \in \mathbb{R}^{d\times d}$ upper triangular), the inversion A_k^{-1} can be computed in $O(d^2)$ [27]. In principle, computing the Cholesky decomposition of A_k still requires $O(d^3)$, but we can apply a rank-one update to the Cholesky decomposition at the previous step, namely $A_k = R_k^\top R_k = R_{k-1}^\top R_{k-1} + x_k x_k^\top = A_{k-1} + x_k x_k^\top$, which is known to require $O(d^2)$ [28]. Several implementations are available for the Cholesky rank-one updates; in our experiments we used the MATLAB routine CHOLUPDATE.

Therefore, the update W_k from W_{k-1} can be computed in $O(Td^2)$, since the most expensive operation is the multiplication $A_k^{-1}b_k$. In particular, this computation is independent of the current number k of training examples seen so far, making this algorithm suited for online settings.

V. Incremental RLSC with Class Extension and Recoding

In this Section, we present our approach to incremental multiclass classification where we account for the possibility to extend the number of classes incrementally and apply the recoding approach introduced in Sec. III. The algorithm is reported in Alg. 1.

A. Class Extension

We propose a modification of recursive RLSC, allowing to extend the number of classes in constant time with respect to the number of examples seen so far. Let T_k denote the number of classes seen up to iteration k. We have two possibilities:

1) The new example (x_k, y_k) belongs to one of the known classes, i.e. $e_{y_k} \in \mathbb{R}^{T_{k-1}}$, with $T_k = T_{k-1}$.
2) (x_k, y_k) belongs to a new class, implying that $y_k = T_k = T_{k-1} + 1$.

In the first case, the update rules for A_k, b_k and W_k explained in Section IV-B can be directly applied. In the second case, the update rule for A_k remains unchanged, while the update of b_k needs to account for the increase in size (since $b_k \in \mathbb{R}^{k\times(T_{k-1}+1)}$). However, we can modify the update rule for b_k without increasing its computational cost by first adding a new column of zeros $\mathbf{0} \in \mathbb{R}^d$ to b_{k-1}, namely

$$b_k = [b_{k-1}, \mathbf{0}] + x_k^\top e_{y_k}, \quad (20)$$

which requires $O(d)$ operations. Therefore, with the strategy described above it is indeed possible to extend the classification capabilities of the incremental learner during online operation, without re-training it from scratch. In the following, we address the problem of dealing with class imbalance during incremental updates by performing incremental recoding.

Algorithm 1: Incremental RLSC with Class Recoding

```
Input: Hyperparameters $\lambda > 0$, $\alpha \in [0,1]$
Output: Learned weights $W_k$ at each iteration
Initialize: $R_0 \leftarrow \sqrt{\lambda} I_d,\ b_0 \leftarrow \emptyset,\ \gamma_0 \leftarrow \emptyset,\ T \leftarrow 0$

Increment: Observe input $x_k \in \mathbb{R}^d$ and output label $y_k$:
    if $(y_k = T+1)$ then
        $T \leftarrow T+1$
        $\gamma_{k-1} \leftarrow [\gamma_{k-1}^\top, 0]^\top$
        $b_{k-1} \leftarrow [b_{k-1}, \mathbf{0}]$, with $\mathbf{0} \in \mathbb{R}^d$
    end if
    $\gamma_k \leftarrow \gamma_{k-1} + e_{y_k}$
    $\Gamma_k \leftarrow k \cdot \mathrm{diag}(\gamma_k)^{-1}$
    $b_k \leftarrow b_{k-1} + x_k^\top e_{y_k}$
    $R_k \leftarrow$ CHOLESKYUPDATE$(R_{k-1}, x_k)$
    $W_k \leftarrow R_k^{-1}(R_k^\top)^{-1} b_k (\Gamma_k)^\alpha$
return $W_k$
```

B. Incremental Recoding

The main algorithmic difference between standard RLSC and the variant with recoding is in the matrix Y containing output training examples. Indeed, according to the recoding strategy, the vector e_{y_k} associated to an output label y_k is coded into $c(e_{y_k}) = e_{y_k}/\rho(y = y_k)$. In the batch setting, this can be formulated in matrix notation as

$$W = (X^\top X + \lambda I_d)^{-1} X^\top Y \Gamma,$$

where the original output matrix is replaced by its encoded version $Y^{(c)} = Y\Gamma \in \mathbf{R}^{n\times T}$, with Γ the $T \times T$ diagonal matrix whose t-th diagonal element is $\Gamma_{tt} = 1/\rho(y = t)$. Clearly, in practice the $\rho(y = t)$ are estimated empirically (e.g. by $\hat{\rho}(y = t) = n_t/n$, the ratio between the number n_t of training examples belonging to class t and the total number n of examples).

The above formulation is favorable for the online setting. Indeed, we have

$$X_k^\top Y_k \Gamma_k = b_k \Gamma_k = (b_{k-1} + x_k^\top y_k)\Gamma_k, \quad (21)$$

where Γ_k is the diagonal matrix of the (inverse) class distribution estimators $\hat{\rho}$ up to iteration k. Γ_k can be computed incrementally in $O(T)$ by keeping track of the number k_t of examples belonging to t and then computing $\hat{\rho}_k(y = t) = k_t/k$ (see Alg.1 for how this update was implemented in our experiments). Note that the above step requires $O(dT)$, since updating the (uncoded) b_k from b_{k-1} requires $O(d)$ and multiplying b_k by a diagonal matrix requires $O(dT)$. All the above computations are dominated by the product $A_k^{-1}b_k$, which requires $O(Td^2)$. Therefore, our algorithm is computationally equivalent to the standard incremental RLSC approach.

Coding as a Regularization Parameter. Depending on the amount of training examples seen so far, the estimator k_t/k could happen to not approximate $\rho(y = t)$ well. In order to mitigate this issue, we propose to introduce a parameter $\alpha \in [0,1]$ and raise Γ_k element-wise to the power of α (indicated by $(\Gamma_k)^\alpha$). Indeed, it can be noticed that for $\alpha = 0$ we recover the (uncoded) standard RLSC, since $(\Gamma_k)^0 = I_T$, while $\alpha = 1$ applies full recoding. In Sec. VI-C we discuss an efficient heuristic to find α in practice.

Incremental Rebalancing. Note that the loss-rebalancing algorithm (Sec. III-D) cannot be implemented incrementally. Indeed, the solution of the rebalanced empirical RLSC is

$$W_k = (X_k^\top \Sigma_k X_k + \lambda I_d)^{-1} X_k^\top \Sigma_K Y_k, \qquad (22)$$

with Σ_k a diagonal matrix whose i-th entry is equal to $(\Sigma_k)_{ii} = 1/\hat{\rho}(y = t_i)$, with t_i the class of the i-th training example. Since Σ_k changes at every iteration, it is not possible to derive a rank-one update rule for $(X_k^\top \Sigma_k X_k + \lambda I_d)^{-1}$ as for the standard RLSC.

VI. Experiments

We empirically assessed the performance of Alg.1 on a standard benchmark for machine learning and on two visual recognition tasks in robotics. To evaluate the improvement provided by the incremental recoding when classes are imbalanced, we compared the accuracy of the proposed method with the standard recursive RLSC presented in Sec. IV-B. As a competitor in terms of accuracy, we also considered the rebalanced approach presented in Eq. (22) (which, we recall, cannot be implemented incrementally).

A. Experimental Protocol

We adopted the following experimental protocol[1]:

1) Given a dataset with T classes, we simulated a scenario where a new class is observed by selecting $T-1$ of them to be “balanced” and the remaining one to be under-represented.
2) We trained a classifier on the balanced classes, using a randomly sampled dataset containing n_{bal} examples per class (specified below for each dataset). We sampled a validation set with $n_{bal}/5$ examples per class.
3) We incrementally trained the classifier from the previous step by sampling online n_{imb} examples for the T-th class. Model selection was performed *using exclusively the validation set of the balanced classes*, following the strategy described in Sec. VI-C.
4) To measure performance, we sampled a separate test set containing n_{test} examples per class (both balanced and under-represented) and we measured the accuracy of the algorithms on the test set while they were trained incrementally.

For each dataset, we averaged results over multiple independent trials randomly sampling the validation set. In Table I we report the test accuracy on the imbalanced class and on the entire test set.

[1]Code available at `https://github.com/LCSL/incremental_multiclass_RLSC`

B. Datasets

MNIST [29] is a benchmark composed of 60K 28×28 greyscale pictures of digits from 0 to 9. We addressed the 10-class digit recognition problem usually considered in the literature, but using $n_{bal} = 1000$ training images per class. The test set was obtained by sampling $n_{test} = 200$ images per class. We used the raw pixels of the images as inputs for the linear classifier.

iCubWorld28 [30] is a dataset for visual object recognition in robotics, collected during a series of sessions where a human teacher showed different objects to the iCub humanoid robot [31]. We addressed the task of discriminating between the 28 objects instances in the dataset, using all available acquisition sessions per object and randomly sampling $n_{bal} = 700$ and $n_{test} = 700$ examples per class. We performed feature extraction as specified in [30], i.e. by taking the activations of the *fc*7 layer of the CaffeNet Convolutional Neural Network [32].

RGB-D Washington [33] is a visual object recognition dataset comprising 300 objects belonging to 51 categories, acquired by recording image sequences of each object while rotating on a turntable. We addressed the 51-class object categorization task, averaging results over the ten splits specified in [33] (where, for each category, a random instance is left out for testing). We subsampled one cropped RGB frame every five from the full dataset, following the standard procedure. We sampled $n_{bal} = 500$ and $n_{test} = 400$ images per class and performed feature extraction analogously to *iCubWorld28*, using the output of CaffeNet’s $fc6$ layer.

C. Model Selection

In traditional batch learning settings for RLSC, model selection for the hyperparameter λ is typically performed via hold-out, k-fold or similar cross-validation techniques. In the incremental setting these strategies cannot be directly applied since examples are observed online, but a simple approach to create a validation set is to hold out every i-th example without using it for training (e.g., we set $i = 6$). At each iteration, multiple candidate models are trained incrementally, each for a different value of λ, and the one with highest validation accuracy is selected for prediction.

However, following the same argument of Sec. III, in presence of class imbalance this strategy would often select classifiers that ignore the under-represented class. Rebalancing the validation loss (see Sec. III) does not necessarily solve the issue, but could rather lead to overfitting the under-represented class, degrading the accuracy on other classes since errors count less on them. Motivated by the empirical evidence discussed below, in this work we have adopted a model selection heuristic for λ and α in Alg. 1, which guarantees to not degrade accuracy on well-represented classes, while at the same time achieving higher or equal accuracy on the under-represented one.

Our strategy evaluates the accuracy of the candidate models on the incremental validation set, but *only for classes that have a sufficient number of examples* (e.g., classes with fewer examples than a pre-defined threshold are not used

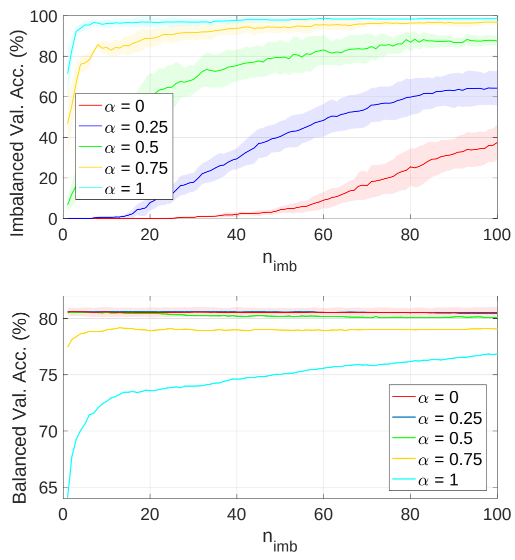

Fig. 2: Classification accuracy on *iCubWorld28* imbalanced (Top) and balanced (Bottom) test classes for models trained according to Alg.1 with varying α and best λ within a pre-defined range (chosen at each iteration and for each α). Growing α from 0 to 1 allows to find a model that maintains the same performance on known classes while improving on the under-represented one.

for validation). Then, we choose the model with largest $\alpha \in [0, 1]$ for which such accuracy is higher or equal to the one measured for $\alpha = 0$, namely without coding. Indeed, as can be seen in Fig. 2 for validation experiments on *iCubWorld28*, as α grows from 0 to 1, the classification accuracy on the under-represented class increases, Fig. 2 (Top), while it decreases on the remaining ones, Fig. 2 (Bottom). Our heuristic chooses the best trade-off for α such that performance does not degrade on well-known classes, but at the same time it will often improve on the under-represented one.

D. Results

In Table I we report the results of the three methods on *MNIST*, *iCubWorld28* and *RGB-D* for a single under-represented class (digit "8", class 28 and *tomato*, respectively). We observed a similar behaviour for other classes. We show both the accuracy on all classes (Total Acc., Left) and on the under-represented one (Imbalanced Acc., Right). We note that, on the under-represented class, Alg. 1 (*RC*) consistently outperforms the RLSC baseline (*N*), which does not account for class imbalance and learns models that ignore the class. Also the total accuracy of *RC* results higher. Interestingly, on the two robotics tasks, *RC* outperforms the loss rebalancing approach (*RB*), particularly when very few examples of the under-represented class are available. This is favorable since, as we said, the rebalancing approach cannot be implemented incrementally (Sec. V-B).

To offer a clear intuition of the improvement provided by our method, in Fig. 3 we show the accuracy of Naïve RLSC

TABLE I: Incremental classification accuracy for Naïve (N) RLSC, Rebalanced (RB) and Recoding (RC) (see Alg. 1). Following the procedure described in Sec. VI-C, we set $\alpha = 0.7$ for *MNIST*, $\alpha = 0.6$ for *iCubWorld28* and $\alpha = 0.7$ for *RGB-D*.

Dataset	n_{imb}	Total Acc. (%)			Imbalanced Acc. (%)		
		N	RB	RC	N	RB	RC
MNIST n_{bal} = 1000	1	79.2 ± 0.3	**79.7 ± 0.4**	**79.7 ± 0.6**	0.0 ± 0.0	7.4 ± 7.7	**9.5 ± 4.9**
	5	79.1 ± 0.3	**82.5 ± 0.7**	80.3 ± 0.6	0.0 ± 0.0	**39.6 ± 6.2**	17.5 ± 6.6
	10	79.2 ± 0.3	**83.6 ± 0.7**	81.0 ± 0.6	0.0 ± 0.0	**49.5 ± 5.7**	25.1 ± 5.3
	50	79.2 ± 0.3	**85.5 ± 0.3**	83.9 ± 0.5	0.0 ± 0.0	**73.5 ± 3.3**	49.1 ± 3.5
	100	79.2 ± 0.4	**85.9 ± 0.4**	85.1 ± 0.5	2.0 ± 0.9	**75.5 ± 2.7**	62.7 ± 2.9
	500	85.5 ± 0.3	**86.2 ± 0.3**	**86.1 ± 0.3**	66.9 ± 1.1	**78.5 ± 0.9**	**77.8 ± 1.1**
iCub n_{bal} = 700	1	77.6 ± 0.3	76.8 ± 0.1	**77.7 ± 0.3**	0.0 ± 0.0	0.4 ± 0.6	**8.0 ± 11.4**
	5	77.6 ± 0.3	77.9 ± 0.1	**78.6 ± 0.3**	0.0 ± 0.0	8.1 ± 3.9	**38.5 ± 9.7**
	10	77.6 ± 0.3	78.3 ± 0.4	**78.9 ± 0.2**	0.0 ± 0.0	23.7 ± 10.8	**49.6 ± 5.6**
	50	77.7 ± 0.2	**80.0 ± 0.2**	**80.0 ± 0.1**	5.4 ± 4.1	73.9 ± 7.3	**75.0 ± 5.5**
	100	78.6 ± 0.1	**80.2 ± 0.1**	80.1 ± 0.2	39.1 ± 3.6	85.9 ± 4.0	**86.5 ± 3.0**
	500	80.2 ± 0.2	**80.1 ± 0.1**	**80.1 ± 0.2**	89.3 ± 2.5	93.8 ± 2.0	**94.8 ± 1.9**
RGB-D n_{bal} = 500	1	80.4 ± 2.2	78.6 ± 3.2	**83.3 ± 3.2**	0.0 ± 0.0	62.0 ± 42.1	**72.2 ± 26.3**
	5	80.4 ± 2.2	83.0 ± 2.1	**83.9 ± 2.6**	0.0 ± 0.0	91.7 ± 12.8	**99.9 ± 0.3**
	10	80.4 ± 2.2	**83.8 ± 1.8**	**83.6 ± 2.6**	2.8 ± 2.4	94.7 ± 8.4	**100.0 ± 0.0**
	50	82.3 ± 2.2	**84.3 ± 1.9**	83.5 ± 2.9	96.6 ± 3.7	**100.0± 0.0**	**100.0 ± 0.0**
	100	82.4 ± 2.1	**84.4 ± 2.0**	83.5 ± 2.8	100.0 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**
	500	82.3 ± 2.1	**84.1 ± 2.0**	**84.1 ± 2.8**	100.0 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**

(Red) and Alg. 1 (Blue), separately on the under-represented class (Top), the balanced classes (Middle), and all classes (Bottom), as they are trained on new examples. For this experiment we let each class be under-represented and averaged the results. It can be noticed that Alg. 1 is much better on the imbalanced class, while being comparable on the balanced ones, resulting in overall improved performance.

We point out that the total accuracy on all datasets for $n_{imb} = 500$ is comparable with the state of the art. Indeed, on *MNIST* we achieve $\sim 86\%$ accuracy, which is slightly lower than the one reported in [29] for a linear classifier on top of raw pixels (this is reasonable, since we are using much fewer training examples). The total accuracy of Alg. 1 on *RGB-D* is approximately 84%, which is comparable with the state of the art on this dataset [3]. On the *iCubWorld28* dataset we achieve $\sim 80\%$ accuracy, which is in line with the results reported in Fig. 8 of [34] (extended version of [30]).

VII. Conclusion

In this paper we addressed the problem of learning online with an increasing number of classes. Motivated by the visual recognition scenario in lifelong robot learning, we focused on issues related to class imbalance, which naturally arises when a new object/category is observed for the first time. To address this problem, we proposed a variant of the recursive Regularized Least Squares for Classification (RLSC) algorithm that (i) incorporates new classes incrementally and (ii) dynamically applies class recoding when new examples are observed. Updates are performed in constant time with respect to the growing number of training examples. We evaluated the proposed algorithm on a standard machine learning benchmark and on two datasets for visual recognition in robotics, showing that our approach is indeed favorable in online settings when classes are imbalanced.

We note that, in principle, for the experiments where we used features extracted from a Convolutional Neural Network, we could have also directly trained the network online, by Stochastic Gradient Descent (backpropagation). While works empirically investigating this end-to-end approach in

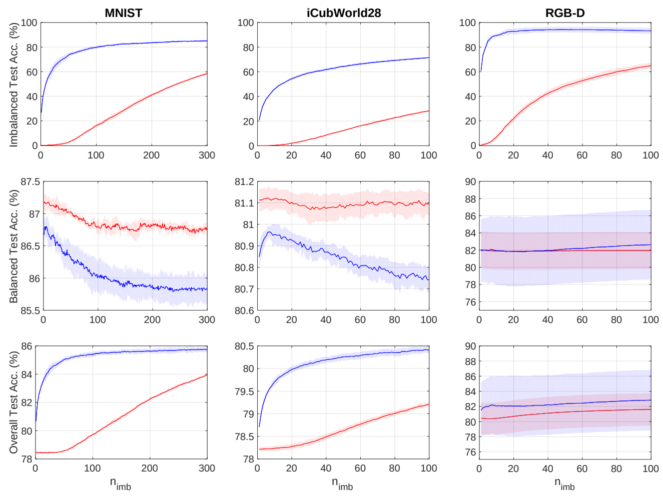

Fig. 3: Average test classification accuracy of the standard incremental RLSC (Red) and the variant proposed in this work (Blue) over the imbalanced (Top), balanced (Middle) and all (Bottom) classes. The models are incrementally trained as n_{imb} grows, as described in Sec. VI-A.

settings where new classes are to be progressively included into the model exist [35], this is still a largely unexplored field, the study of which is not in the scope of this work. The method we propose allows to update a predictor without using training data from previous classes in a fast and stable way, and, by relying on rich deep representations learned offline, is proven to be competitive with the state of the art, while being more suitable for online applications.

Future research will focus on strategies to exploit knowledge of known classes to improve classification accuracy on new ones, following recent work [20], [21], [22], [23].

Acknowledgment

The work described in this paper is supported by the Center for Brains, Minds and Machines, funded by NSF STC award CCF-1231216 and by FIRB project RBFR12M3AC, funded by the Italian Ministry of Education, University and Research. We acknowledge NVIDIA Corporation for the donation of the Tesla k40 GPU used for this research.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.
[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-scale Image Recognition," *arXiv preprint 1409.1556*, 2014.
[3] M. Schwarz, H. Schulz, and S. Behnke, "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features," in *ICRA*, 2015.
[4] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, 1999.
[5] A. H. Sayed, *Adaptive Filters*. Wiley-IEEE Press, 2008.
[6] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, 2011.
[7] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint 1312.6211*, 2013.
[8] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, 2001.
[9] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.
[10] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, 2009.
[11] R. M. Rifkin, "Everything old is new again: a fresh look at historical approaches in machine learning," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.
[12] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, 2003.
[13] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang, "Error-driven incremental learning in deep convolutional neural network for large-scale image classification," in *ACM International Conference on Multimedia*, 2014.
[14] L. C. Jain, M. Seera, C. P. Lim, and P. Balasubramaniam, "A review of online learning in supervised neural networks," *Neural Computing and Applications*, 2014.
[15] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," *ICML*, 2016.
[16] J. Lin, R. Camoriano, and L. Rosasco, "Generalization Properties and Implicit Regularization for Multiple Passes SGM," *ICML*, 2016.
[17] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, "Compete to compute," in *NIPS*, 2013.
[18] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, "Incremental support vector learning: Analysis, implementation and applications," *Journal of Machine Learning Research*, 2006.
[19] S. Thrun, "Is learning the n-th thing any easier than learning the first?" *NIPS*, 1996.
[20] T. Tommasi, F. Orabona, and B. Caputo, "Safety in numbers: Learning categories from few examples with multi model knowledge transfer," in *CVPR*, 2010.
[21] T. Tommasi, F. Orabona, M. Kaboli, and B. Caputo, "Leveraging over prior knowledge for online learning of visual categories," in *BMVC*, 2012.
[22] I. Kuzborskij, F. Orabona, and B. Caputo, "From n to n+ 1: Multiclass transfer incremental learning," in *CVPR*, 2013.
[23] Y. Sun and D. Fox, "Neol: Toward never-ending object learning for robots," in *ICRA*, 2016.
[24] P. Bartlett, M. Jordan, and J. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, 2006.
[25] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.
[26] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[27] G. H. Golub and C. Van Loan, *Matrix computations*. Johns Hopkins Univ., 1996.
[28] Å. Björck, *Numerical Methods for Least Squares Problems*. SIAM, 1996.
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *IEEE Proceedings*, 1998.
[30] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale, "Teaching iCub to recognize objects using deep Convolutional Neural Networks," in *ICML Workshop on Machine Learning for Interactive Systems*, vol. 43, 2015, pp. 21–25.
[31] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. Von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, *et al.*, "The iCub Humanoid Robot: An Open-systems Platform for Research in Cognitive Development," *Neural Networks*, vol. 23, no. 8, pp. 1125–1134, 2010.
[32] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia*, 2014.
[33] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *ICRA*, 2011.
[34] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale, "Real-world Object Recognition with Off-the-shelf Deep Conv Nets: How Many Objects can iCub Learn?" *ArXiv preprint 1504.03154*, 2015.
[35] C. Käding, E. Rodner, A. Freytag, and J. Denzler, "Fine-tuning deep neural networks in continuous learning scenarios," in *ACCV Workshop on Interpretation and Visualization of Deep Neural Nets (ACCV-WS)*, 2016.