Open World Recognition in Image Classification

Daniele Giannuzzi
s290231@Studenti.polito.it

Francesco Scorca
s288876@studenti.polito.it

Paolo Calderaro
s288831@Studenti.polito.it

Abstract

The proposed work is meant as an exploration of incremental learning techniques. We start from the implementation of three baselines: fine-tuning, Learning Without Forgetting, iCaRL framework. Furthermore, different ablation studies are carried, analysing variations in losses and classifiers. We then tested models able to reject samples, labelling them as unknown class: this scenario is tested in both open and closed world, with and without the rejection capability. In the last part a modification in the nearest-mean-of-exemplars classifier is proposed, accounting for the different spreads of the classes in the feature space.

1. Introduction

Continual learning, considering the computer vision field, refers to the ability of a visual object classification system to incrementally learn about new classes once acquired data on them. The problem gets even more sophisticated if we leave the closed world assumption (CWA), in which the model assumes that every class to recognize has already been encountered during training. If this does not hold we are in the open world scenario and the model has to distinguish between what it knows and what it does not by labelling instances as unknown, if supposed to belong to a never seen class. One of the main problems of the incremental learning scenario is catastrophic forgetting, i.e. adapting a model to new data results in performance degradation on previous tasks. The following work aims at exploring this scenarios and some methods to face its challenges.

2. Related Works

In this section we report some research efforts addressing continual learning and open world recognition.
A strategy to face catastrophic forgetting is storing previous experience explicitly or implicitly, for instance by collecting or generating samples from encountered classes for for rehearsal or pseudo-rehearsal. In [11] the authors introduce methods to synthesize class-conditional input images characterized by high diversity among them, by using information stored in the batch normalization layers of a trained network. Other methods exploit modularity for localizing inference to a subset of the network, an example is [10] developing a growing architecture which retains lateral connections to previously frozen modules. In [1] the authors equip convolutional layers with task-specific gating modules whose objective is selecting which filters to apply on the given input. Another common approach consists in reducing change in parameters if it causes performance downgrade on prior tasks, for instance [2] introduce a loss penalizing the changes in the computed classifiers attention maps. Regarding open World Recognition, [8] exploits CNNs to obtain feature representations used to compute class-probability scores to compare to a global rejection threshold, heuristically estimated, for rejecting samples; [3] introduce learned class-specific thresholds and a loss enforcing the network to map samples belonging to same class closer to their class centroid and closer among them.

3. Baselines

In this section we compare three different approaches for image classification, starting from fine-tuning, then adding measures to prevent catastrophic forgetting arriving to iCaRL. As data-set we use the CIFAR-100 data and we train all 100 classes in batches of 10 classes at a time. The evaluation metric is a standard multi-class accuracy on the test set, containing samples belonging to all the classes observed. Since the order in which the classes are met by the network influences the results, we run this benchmark three times with different class orders and report averaged results through the runs. For all methods we train a 32-layers ResNet[4] using standard backpropagation with mini-batches of size 64 and a weight decay parameter of 0.00001 and a momentum parameter of 0.9. Each training step consists of 70 epochs, the learning rate starts at 2.0 and is divided by 5 after 49 and 63 epochs. Furthermore, we performed random cropping and flipping on the training-set for data augmentation.

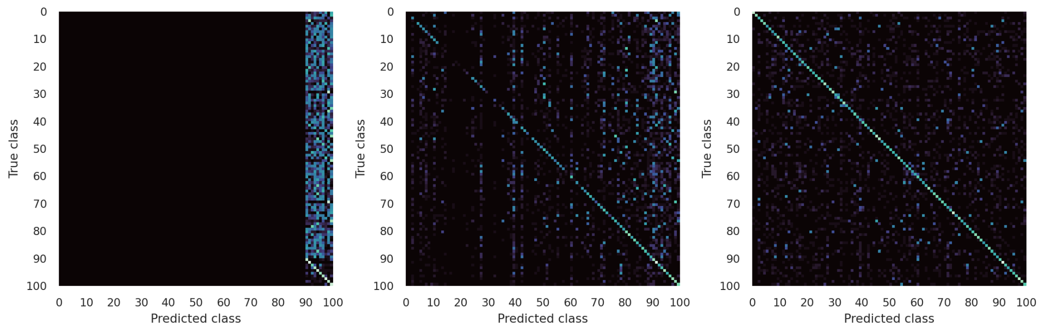

Figure 1: Heatmap of the baselines, from left to right: (a) Finetuning, (b) LwF.MC, (c) iCaRL.

3.1. Fine-tuning

This is the simplest form of transfer learning consisting in training the same network in subsequent steps adding ten neurons (one for each new class) to the output layer. It is supposed to provide the worst results since the network will simply overwrite his weights.

3.2. LwF.MC

This is the same approach of LwF[7], implemented as described in iCaRL[9], where MC stands for multi-class. Besides the classification loss, it is exploited a distillation loss which ensures that the discriminative information learned previously is not lost. The resulting loss, comprehensive of distillation and classification, is the following:

$$L = -\sum_{x_i, y_i} [\sum_{y=s}^{t} \delta_{y=y_i} \log g_i^y + \delta_{y \neq y_i} \log(1 - g_i^y) + \sum_{y=1}^{s-1} q_i^y \log g_i^y + (1 - q_i^y) \log(1 - g_i^y)] \tag{1}$$

where x_i, y_i are the i-th sample input and label, t is the number of classes observed so far, $s - 1$ the number of classes before this split. g_i^y, q_i^y are the sigmoid output of the y-th neuron, for the i-th sample respectively of the new and the old network, which needs to be stored to evaluate the new samples and produce these targets.

3.3. iCaRL

In this framework the network is a trainable feature extractor followed by a single classification layer. A new countermeasure to catastrophic forgetting is introduced: the storage of exemplars, i.e. K images[1] from the previously encountered classes. At each incremental step iCaRL builds an augmented training set consisting of the currently available training examples together with the stored exemplars. The model is trained minimizing the loss described in (1).

[1]the exemplars need to be stored as images, since their feature representation will change for each incremental step

iCaRL manages to keep fixed the dimension of exemplars set storing for each class $m = \frac{K}{t}$ [2]images, exploiting two routines: the first one consists in herding[3], which constructs the exemplars set as a prioritized list; the latter is the removing routine, which simply keeps the first m images. For the testing phase iCaRL uses the nearest-mean-of-exemplars (NME) classifier, a variation of the nearest-class-mean classifier. An image is assigned to the class whose prototype, which is is the average feature vector of all exemplars for a class, is nearest to the image feature representation.

3.4. Results

In Figure 2 we report the performance in terms of accuracy of each baseline method and it is possible to compare the drop trend of accuracy for the different models. Further details can be found in Table 5 in appendix B. As expected, iCaRL significantly outperforms the other methods. This puts the use of rehearsal in the spotlight, which is quite intuitive, since we are explicitly storing information from the

[2]we recall that K is the total number of images, t is the number of classes observed so far

[3]this algorithm is shown in appendix B, Table 4

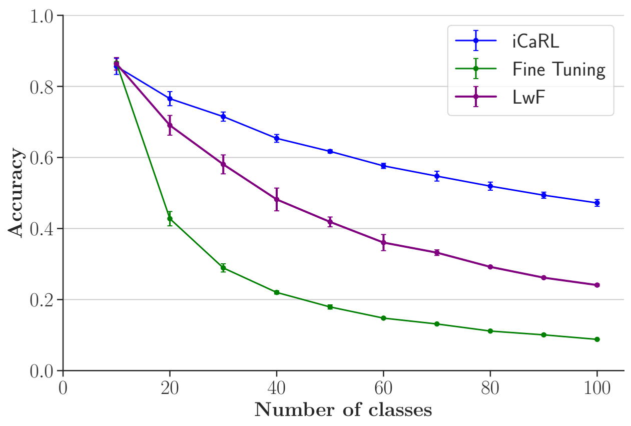

Figure 2: Accuracy performance of difference baselines.

previous step. Considering the performance of Fine-tuning, we can observe that in the last step it correctly classifies less than a sample out of ten, showing the catastrophic forgetting of the previous classes and the complete inadequacy of this method. As we can further see from the heatmap in Figure 1, as expected Fine-tuning forgets the existence of the previous classes, predicting all the samples as belonging to the last ten. LwF.MC mitigates this, but still shows a bias towards the last classes, since more prediction appear in the right-most part. Finally, iCaRL presents the most homogeneous heatmap, imposing itself as an unbiased predictor.

4. Ablation studies

In this section we focus on evaluating different classification strategies and different combinations of loss functions. Our objective is to find out how the different components of iCaRL influence the overall performances and better understand this framework behaviour and possible limitations.

4.1. Loss ablation study

In this part we focus on the loss function. We have two types of loss to evaluate: the classification loss and the distillation loss. We fixed a Cross Entropy loss as classification loss, since it is widely spread and accepted in the literature:

$$L_{CE} = \sum_{i=1}^{N} y_i log(p_i) \tag{2}$$

Where N is the total number of classes, p_i the output of a softmax applied on the network logits and y_i a binary indicator with 1 only for the true class label of the sample.

L1/L2 loss. Also known as MAE (Mean Absolute Error) and MSE (Mean Squared Error) loss:

$$L_{MAE} = |x_n - y_n| \qquad L_{MSE} = (x_n - y_n)^2 \tag{3}$$

With these losses we aim at minimizing respectively the absolute and the squared difference between the target class y_n of a sample n and his estimated value x_n. We tested these losses since they are among the simplest loss formulation. As we can see, since the L2 loss squares the error between target class and his estimated value, it will penalize more larger errors with respect to L1 loss. This aspect makes this loss more sensitive to outliers.

Less Forget Constraint. Hou at al. in [6] use a distillation loss minimizing the cosine divergence of the features extracted by the old model and the new one:

$$L_{LFC}(x) = 1 - \langle \widehat{f}'(x), \widehat{f}(x) \rangle, \tag{4}$$

where $\widehat{f}'(x)$ and $\widehat{f}(x)$ are respectively the normalized features extracted by the old model and those by the current one, while$\langle\rangle$ refers to the cosine similarity. This formulation discourages changes in the relative orientation of features extracted by the networks, not constraining variations in the magnitudes, since normalization is applied in the loss. Furthermore this loss is weighted by an adaptive coefficient accounting for the degree of need to preserve the previous knowledge:

$$\lambda = \lambda_{base} \sqrt{\frac{|C_n|}{|C_o|}}, \tag{5}$$

with $|C_n|$ and $|C_o|$ number of new and old classes, λ_{base} weight at step 0, set to 5 as in the referred paper.

Knowledge Distillation loss. This loss has been applied for the first time in incremental learning by Li and Hoiem in [7] and consists of the following formula:

$$L_{KD}(\mathbf{y_o}, \mathbf{\hat{y}_o}) = -\sum_{i=1}^{l} y_o'^{(i)} \log \hat{y_o}'^{(i)}, \tag{6}$$

where l is the number of labels, y_o' and $\hat{y_o}'$are the modified versions of recorded and current probabilities:

$$y_o'(i) = \frac{(y_o^{(i)})^{\frac{1}{T}}}{\sum_j (y_o^{(j)})^{\frac{1}{T}}}, \qquad \hat{y}_o'(i) = \frac{(\hat{y}_o^{(i)})^{\frac{1}{T}}}{\sum_j (\hat{y}_o^{(j)})^{\frac{1}{T}}} \tag{7}$$

The rationale behind this modification of the cross-entropy loss consists in the fact that much of the information about the learned function resides in the ratios of very small probabilities in the soft targets[5], which have negligible influence on the cost function during the transfer stage. Raising the logits to values of T>1 produces a softer probability distribution over classes, addressing the mentioned phenomenon. We set the temperature T=2, following the referred literature.

4.2. Loss conclusions

Since the loss defines how the model is trained, it is necessary to perform again the tuning of the learning hyper-parameters. We found good results by decreasing the learning rate (0.1 for L_{KD} and 0.3 for L_{L1} , L_{L2} , L_{LFC}) and increasing weight decay to 10^{-4} for all new losses combinations, fixing the remaining hyper-parameters as described in Section 3. We compare our losses combinations to the BCE + BCE (Binary Cross Entropy both for classification and distillation) used in the official framework.
As we can see from Figure 4, the combination with L2 loss performs slightly better then the L1 loss and it is the most stable across splits. The Knowledge Distillation loss performs well on first splits, while performance get lower when more classes are added. Conversely, LFC presents an opposite behaviour, better preserving performance in the last steps. We conjecture that this loss constraints too heavily

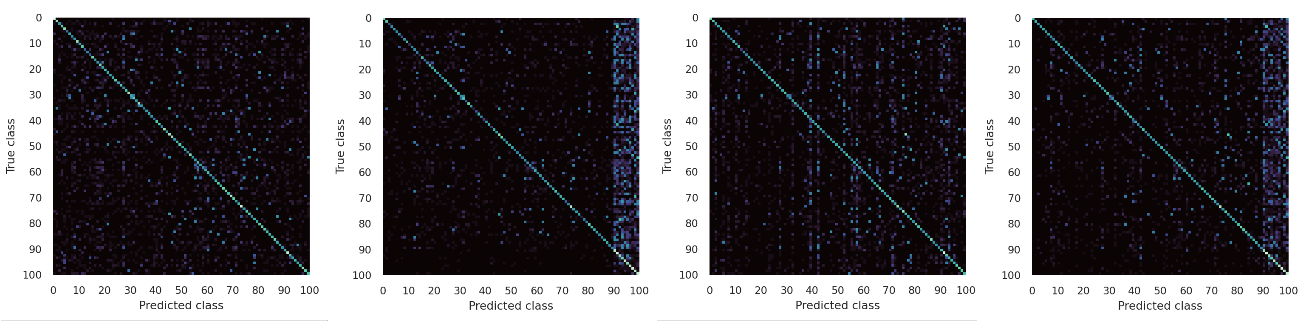

Figure 3: Heatmap of the model trained, from left to right: KNN trained only on exemplar, KNN trained on all data, SVC trained only on exemplar, SVC trained on all data.

the feature space in this framework, limiting the acquiring of new knowledge, while this is more and more mitigated by the decreasing weighting parameter. Thus we hypothesize that further investigation on heuristics on this parameter can improve also the initial steps results. The average accuracy can be seen in Table 4 (appendix B). We can state from the results of this study that the best suitable choice for the iCaRL framework is still a pair of BCE losses, since it is the best combination among the ones tested.

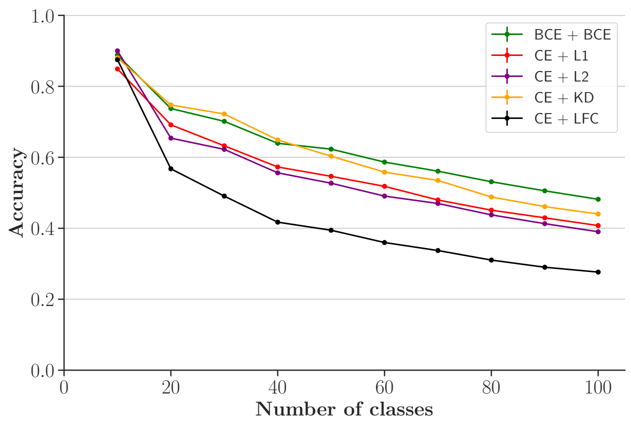

Figure 4: Accuracy performance of different loss combinations tested.

4.3. Classifier ablation study

In the following experiments we replaced the NME classification strategy proposed by the authors of iCaRL with two different classifiers: i) K-Nearest Neighbors classifier, ii) Support Vector Classifier.

KNN Classifier. KNN classifier is a non-parametric classifier that decides the class of a sample by comparing it to the K nearest neighbors labels from the training set. We choose this classifier due to its conceptual similarity to NME, since both strategies have a distance-based approach to classification.

Model	Parameter	Best configuration	Average accuracy (%)
KNN	K: 8 → 15 weight: {distance, uniform}	{ K : 10, weight : distance }	all data: 51.9 ± 2.0 only exemplars: 59.1 ± 2.1

SVC	C: {1, 0.1, 0.001}
	tolerance: {1e-4, 1e-3, 1e-2}
	kernel: {linear,rbf}
		{ C : 1,
		tolerance: 0.001,
		kernel : linear }
			all data: 56.1 ± 0.9
			only exemplars: 56.4 ± 2.0

Table 1: Different hyperparameters configurations tried for the classifiers tested. Last column refers to the best configuration accuracy, averaged through all splits for three different runs.

We used a consistent configuration of the model throughout the incremental training: hyper-parameters of the model were fixed before starting the training of the network and remained the same through all the different splits. We trained the model, at the end of each split testing with two different training-sets: the union of exemplars and training data for the new classes of the current split or only exemplars. We expect for the first training set that feeding the KNN model with an unbalanced dataset will cause a bias towards the classes with more samples.

SVC. Support Vector Machine were the state-of-the-art classification methods before the spread of Deep Learning. Support Vector Classifier tries to separate the different classes by building hyper-planes in the features spaces and finding the hyper-plane that separates classes with highest margin.
Also for SVC we used a consistent configuration throughout all the steps of the incremental training and we tested the two training scenarios. For the first training scenario (training with all data) we tested a class-weighted SVC that assigns weights to the classes depending on the number of training samples of each class and modifies the prediction accordingly. Due to that, we expect to have similar results in both training scenarios. We did experiments with rbf kernel as well as with the linear version of SVC, to check if we could achieve good results without any kernel transformation.

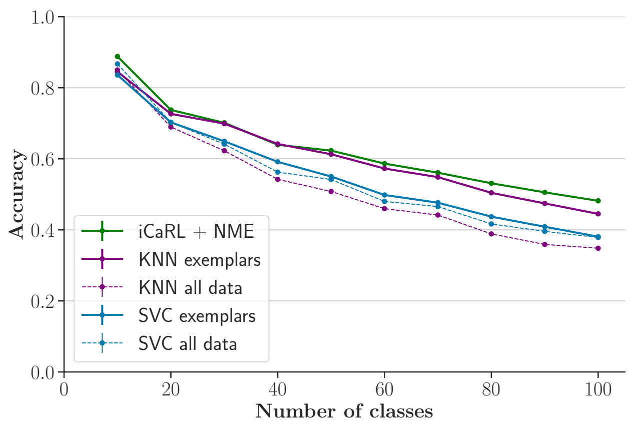

Figure 5: Accuracy performance of different classifiers tested. All data refers to training the model with both exemplar and training set of the current split.

4.4. Classifier conclusions

The list of hyper-parameters tested for both models can be found in Table 1. We can see from Figure 5 that the best-performing configuration among the new proposed is KNN using only exemplars, with an average accuracy of 59.1%. This configuration handles well catastrophic forgetting, but it performs slightly worse than iCaRL with NME: this may be due to the fact that NME is more robust to noise in data since it computes classes averages, which reduce the contribution of noise in the prediction. For the KNN model we can also point out from the confusion matrices in Figure 3, that training the model with all the data available heavily skew the prediction towards the last ten classes, losing 7% of accuracy on average with respect to training with a balanced dataset. So, as expected, KNN model benefits from a training set composed of only exemplars. For the SVC model the best performance configuration is the one with linear kernel, with an average accuracy of 56%. The performance across the two training scenarios are comparable, even if the model trained on all the data still presents skewness towards last ten classes, probably because the model did not handle well the unbalance of classes in the last split.

5. Open World Recognition

Until now, all experiments have been conducted in a standard incremental training scenario also known as closed world without rejection scenario. Nevertheless, recognition in the real world is quite different: the datasets are dynamic and novel categories must continuously be identified and added. In this context, open world recognition assumes incomplete knowledge of the world during testing. Hence, at test time the system could have to deal with several unseen categories. Our objective is to handle these unknown classes at prediction time, by giving to the model the chance to reject predictions if it is not confident. To address this problem, we divided the CIFAR-100 dataset in 50 classes for incremental training and 50 for the testing phase. Summarizing, in the closed world scenario, the model is tested only on classes seen up to the current training split. The ideal result is that model rejects nothing. In the open world scenario, the test dataset is composed only of unknown classes. So we expect that the rejection rate is near to 1. We introduced in the model the capability of rejection, by implementing a naïve rejection strategy, consisting in labelling a sample as unknown if its maximum probability to belong to a class, assigned by the network by means of a softmax layer, is lower than a fixed threshold t. We tested different values of this hyperparameter, obtaining satisfying results for $t = 0.80$.

Incremental split	1	2	3	4	5
Closed World	0.152	0.301	0.383	0.506	0.569
Open World	0.423	0.583	0.692	0.742	0.787

Table 2: Rejection rate over all splits for closed world and open set scenarios.

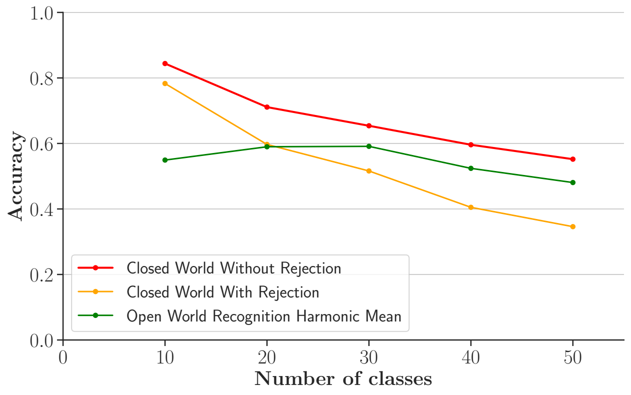

Figure 6: Accuracy performance of different scenarios

The metrics involved are the following: the accuracy for closed world without and with rejection and the open world harmonic mean, which is the harmonic mean computed at each incremental step between the accuracy of closed world with rejection and the accuracy of rejecting unknown samples of open set scenario. Table 2 shows the rejection rates, presenting an increasing trend which suggests that the network becomes more and more unconfident as new data arrive. This is mirrored in the drop of performance, observable in Figure 6, between the model with and without rejection: although knowing all the classes, the first model tends to reject always more samples, suggesting that this approach could nullify the network classification capability in the long run. The results with harmonic mean are more stable, mitigated by the open world performance, for which this more and more conservative trend is well suited.

6. Extension: NME with Distances Mean

Since we are working with the NME, in order to classify a sample we only consider the distance between its feature representation and the current classes means. This solution does not take into account that a class could be more or less scattered than others in the feature space. We tried to improve the performances of iCaRL by including this kind of information during testing. In Figure 7 we notice how using half of exemplars of iCaRL does not reduce significantly the performances, which is a clear benefit of herding: even if exemplars are less, the most representative ones are preserved. Consequently, it is possible to exploit the second half of memory to store other samples to determine the sparsity of the class cluster in the feature space. Therefore, we shaped a second type of exemplars set, made of samples at approximately the distance characterizing the spread of a class, computed as the mean distance of the samples from the prototypes. This is described in Algorithm 1, which points out how, in the novel exemplar set, samples of a specific class are ordered by the absolute value of the difference between class distances mean and the distance of the sample from the prototype.

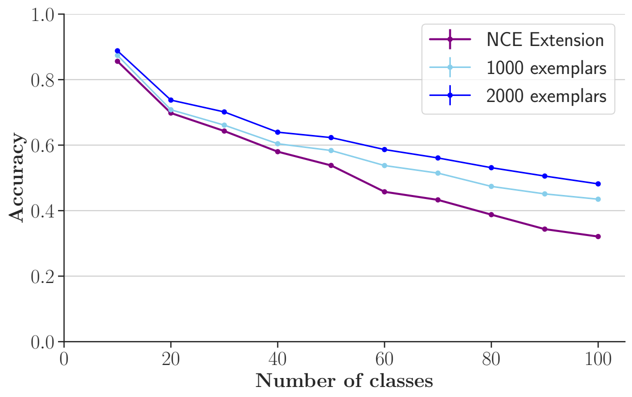

Figure 7: Accuracy performance of our extension compared with iCaRL

Algorithm 1 Construct Second Exemplar Set

```
input image set $X = \{x_1, ..., x_n\}$ of class $y$
input $m$ target number of exemplars
require current feature function $\varphi : X \rightarrow R^d$
    $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$    //current class mean
    $\hat{d} \leftarrow \frac{1}{n} \sum_{x \in X} \|(\varphi(x) - \mu)\|$    //class distances mean
    for k = 1, ... , m do
        $p_k \leftarrow \underset{x \in X}{\mathrm{argmin}} \left| \hat{d} - \|\varphi(x) - \mu\| \right|$
    end for
    $P \leftarrow (p_1, ..., p_m)$
output second exemplar set $P$
```

Then we perform for both exemplars set the same reduction routine used by iCaRL.

Similarly to herding, the selection of exemplars is prioritized: this is useful to have a more accurate estimate of the distances mean than a random selection of samples. At test time, a sample x^q is labelled y^q evaluating the minimum ratio between the distance of the sample from the prototype μ_c and the distances mean $\hat{d}_c$ for set of learned classes C.

$$y^q = \underset{c \in C}{\mathrm{argmin}} \frac{\|\varphi(x^q) - \mu_c\|}{\hat{d}_c} \tag{8}$$

Results, in Figure 7, showed a slight drop in performance compared to iCaRL. We assume that this is due to the fact that classes with higher sparsity have a distances mean that is large enough to unbalance predictions to their advantage[4]. Furthermore, the distance mean is an approximation for all the dimensions: since a class can have images farther in one dimension than in others, this approximation can lead to misleading results. Another significant consideration is that iCaRL uses exemplars to obtain prototypes, which are already approximations of the real class means. The same happens for the distance mean exemplars, which introduce another approximation, the one of the distances mean for each class, which could have affected the performance by adding an extra bias.

7. Conclusions

We developed this work as an exploration of the class-incremental setting, by deepening and varying iCaRL components, with the purpose of understanding its limitations and the challenges of this setting. The proposed ablation studies achieved no further improvements with respect to the official version of iCaRL, nevertheless allowed to enlighten some mechanics, as the necessity of managing the exemplars classifier-wise, or the necessity of investigating heuristics for weighting the losses. Then we analyzed the implications of breaking the closed world assumption, facing them with a naïve approach, establishing a widely improvable baseline, suggesting the inadequacy of shallow methods. Finally we introduced a method aiming to code the dispersion of a class in its features space, exploiting a second exemplar set. We propose to extend this method with the following future works:

1. estimating the dispersion of a class by means of the variance of its feature representations, rather than the mean distance from its centroid;

2. updating the prototypes with different methods[5], in order to fully adopt the exemplars set for class-dispersion estimation.

[4]this unbalance is shown in the heatmap in Figure 9, appendix B
[5]a first attempt is described in appendix A

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning, 2020. 1

[2] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing, 2019. 1

[3] Dario Fontanel, Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Boosting deep open world recognition by clustering. *IEEE Robotics and Automation Letters*, 5(4):5985–5992, Oct 2020. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 3

[6] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[7] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017. 2, 3

[8] Massimilano Mancini, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, and Barbara Caputo. Knowledge is never enough: Towards web aided deep open world recognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2019. 1

[9] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, July 2017. 2

[10] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2016. 1

[11] Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion, 2020. 1

[12] Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning, 2020. 8

A. Semantic Drift Compensation

When training the network it is possible to observe a drift in the 64-dimensional representation provided at each step, hence in the position of the real means, which we estimate by means of exemplars. In Figure 8 it is possible to observe this drift. This has been obtained by storing all the samples of a single class, extracting their iCaRL features representation and mapping them in a 2-dimensional space by means of Principle Component Analysis (PCA). Furthermore, we collected 100 exemplars and projected the estimates of the mean at the two steps, which do not coincide with the real ones: this phenomenon degrades performance and gets more and more marked as the exemplar set gets smaller, hence for each incremental step. Following [12] we propose a different estimate of the classes means for incremental steps, which takes into account the representations shifts. First we need to introduce the semantic drift, defined as the difference of a class mean at the current step (which is not available) and the class mean at the step in which was obtained by the training set:

$$\Delta_{c^s}^{s\to t} = \mu_{c^s}^t - \mu_{c^s}^s, \tag{9}$$

where s is the split in which the class was met the first time, t is the current split, μ_{c^s} is the mean of a class c^s (we could use another index to refer to the specific class, but it is omitted to ease the notation). In order to obtain $\mu^t_{c^s}$, first we measure the drift of the current data during the last training step:

$$\delta_i^{t-1\to t} = z_i^t - z_i^{t-1}, \tag{10}$$

where z_i refers to the vector representation of a sample in the current training set.

The semantic drift is then approximated by the following formula:

$$\hat{\Delta}_{c^s}^{t-1\to t} = \frac{\sum_i \omega_i \delta_i^{t-1\to t}}{\sum_i \omega_i} \tag{11}$$

$$\omega_i = e^{-\frac{||z_i^{t-1} - \mu_{c^s}^{t-1}||^2}{2\sigma^2}} \tag{12}$$

where we set the hyperparameter σ= 0.2, following the previously mentioned paper. For every prototype we are computing a weight per vector drift, dependent from the distance of the two. This set of weights is used to compute the semantic drift in two subsequent steps of a class prototype as the weighted sum of the drifts of the available samples. Eventually, a recursive scheme is applied to update all previously learned prototypes at each step:

$$\mu_{c^s}^t = \mu_{c^s}^{t-1} + \hat{\Delta}_{c^s}^{t-1\to t} \tag{13}$$

Unfortunately the efforts put in this direction did not produce the hoped results and we could not investigate and tackle the reasons because of lack of computational and time resources. Despite of this we mention this method, since promising for further works, especially if combined with the proposed extension, allowing to employ the entire exemplars set for the estimation of the classes spreads.

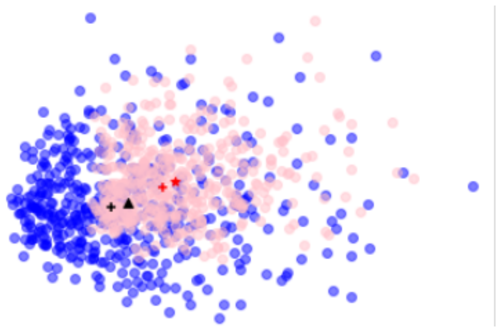
Figure 8: Image features of a single class mapped in 2D space by means of PCA. In blue images representation at first split, with black triangle as real mean. In pink images representation at second split, with red red star as real mean. Crosses represent mean of 100 exemplars.

B. Additional material

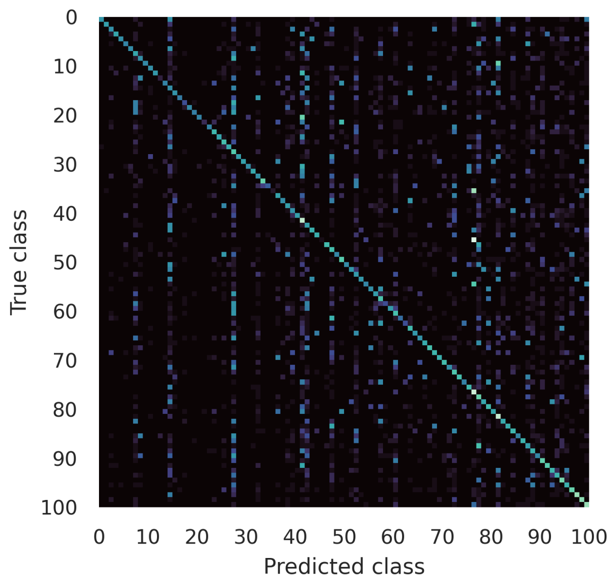

Figure 9: Heatmap of the last split of iCaRL with the new classifier. Vertical lines suggest a bias towards some classes which probably have the largest spread in the features space.

```
**Herding algorithm**
**input** image set $X = \{x_1, ..., x_n\}$ of class $y$
**input** $m$ target number of exemplars
**require** current feature function $\varphi : X \rightarrow R^d$
    $\mu \leftarrow \frac{1}{n} \sum_{x \in X} \varphi(x)$    //current class mean

    **for** k = 1, ... , m **do**
        $p_k \leftarrow \underset{x \in X}{\operatorname{argmin}} \left\| \mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1} \varphi(p-j)] \right\|$
    **end for**
    $P \leftarrow (p_1, ..., p_m)$
**output** exemplar set $P$
```

Table 3: Herding algorithm used in iCaRL to construct an exemplar set.

Loss combination	Avg. accuracy (%)
$L_{BCE} + L_{BCE}$	62.1± 1.2
$L_{CE} + L_{KD}$	60.8 ± 1.4
$L_{CE} + L_{L1}$	55.8 ± 0.4
$L_{CE} + L_{L2}$	54.6 ± 0.7
$L_{CE} + L_{LFC}$	43.2 ± 2.1

Table 4: Average accuracy over all splits. Results comes from three run on different class splits.

Incremental split	1	2	3	4	5	6	7	8	9	10	Avg. Accuracy
FineTuning (%)	86.5 ± 0.3	42.7 ± 2.0	28.9 ± 1.1	22.0 ± 0.4	17.9 ± 0.6	14.7 ± 0.1	13.1 ± 0.1	11.1 ± 0.1	10.0 ± 0.1	8.7 ± 0.01	25.6 ± 0.5
LwF.MC (%)	86.3 ± 1.7	69.0 ± 2.7	58.0 ± 2.6	48.2 ± 3.2	41.8 ± 1.3	36.0 ± 2.2	33.2 ± 0.7	29.2 ± 0.2	26.1 ± 0.1	24.1 ± 0.2	45.2 ± 1.5
iCaRL (%)	85.6 ± 2.2	76.5 ± 1.9	71.5 ± 1.3	65.3 ± 1.1	61.7 ± 0.5	57.6 ± 0.7	54.7 ± 1.4	51.9 ± 1.1	49.4 ± 0.9	47.2 ± 0.9	62.1 ± 1.2

Table 5: Average accuracy over all splits. Results comes from three run on different class split.