Open World Recognition in Image Classification

Andrea Rubeis
Politecnico di Torino
s290216@studenti.polito.it

Gianluigi Lopardo
Politecnico di Torino
s277268@studenti.polito.it

Marco Momo
Politecnico di Torino
S279653@studenti.polito.it

Abstract

One of the main obstacles to the spread of deep learning today is its restriction to closed world scenarios. However, several solutions have been proposed to address the problem: in this paper, we analyse some state-of-the-art methods capable of operating in an open world in image classification. We apply slight variations to the best known methods and then propose some of our own modifications. This work is carried out along two main axes:

- *incremental learning: to gradually learn new classes;*
- *open set: to recognise whether the classes belong to the knowledge that the model already possesses.*

The implementation is available in our GitHub repository.[1].

1. Introduction

Natural learning systems are inherently incremental: new knowledge is learned continuously over time while existing knowledge is retained. However, while deep learning methods have brought state-of-the-art results in Computer Vision, they still suffer from *catastrophic forgetting*: there is a severe performance degradation when incrementally learning new classes if previous data is not available.

Currently, neural network architectures require the entire data set, consisting of all samples of old and new classes, to update the model. In addition to becoming untenable due to memory issues when the number of classes (and thus images) grows, this demand also encounters legal problems.

Many strategies have been proposed to overcome this problem. A popular solution is the use of *distillation* - an additional loss that aims to maintain the previously acquired knowledge - and the selection of a few exemplars from the old classes as a representatives of past data.

Unfortunately, this is not sufficient if we want our system to behave correctly in the real world. In fact, we must also require the model to be able to detect whether or not an instance belongs to its knowledge. To do this, at the top of our incremental model we must define a *rejection strategy* - a criterion that tells the model whether an instance belongs to the known categories.

[1] https://github.com/gianluigilopardo/Open-World-Recognition

2. Related Works

One of the earliest contributions in deep learning approaches to incremental learning was **LwF** [4], which proposed adding a distillation loss in addition to the classification loss, to encourage retention of prior knowledge. Specifically, LwF adds a *softmax with temperature distillation* to the standard cross entropy.

A further improvement to this approach has been made by **iCarl** [5]. The additional idea is to store a (possibly small) representative subset of images for each class learnt (called *exemplars*), intelligently chosen and managed to limit the memory required. A deep network is used as feature extractor and the *NME (nearest-mean-of-examplars-rule)* classifier is applied.

[1] follows the strategy of [5] of keeping the exemplars but learns a feature representation and a classifier jointly, using the deep architecture in a end to end fashion. [6] found that the incremental learning methods that use exemplars from old classes are biased toward new classes and propose the addiction of the bias correction layer and a two stage training for balancing this bias.

3. Dataset

For the evaluation of the algorithms we use the popular CIFAR-100 dataset, containing 600 images (500 for training and 100 for testing) for each of the 100 classes. We apply transformations to the training datasets (random cropping, flip) to perform data augmentation and we normalize train and test images. We consider two version of this dataset. For the first one, we randomly divide the dataset on 10 sets of 10 classes. At each incremental step the training data is made by the new 10 classes, while we test on all the discovered classes. For the second one instead, we randomly divide the dataset in two halves. The first 50 classes are considered as *known* classes and are managed as in the

previous version. While the last 50 classes are all treated as *unknown* and they are grouped in a single set for test the rejection capability.

4. Methods

In this section we describe and compare different incremental learning algorithms, provide an ablation study on different classifiers and losses, and finally evaluate the classification confidence in closed and open world scenarios. We use ResNet-32 as deep architecture for every algorithm.

4.1. Finetuning

First of all, we need a benchmark to compare the performance of the methods. We therefore incrementally train the network without taking any action to address the catastrophic forgetting issue. We use *Binary Cross Entropy* (BCE) as classification loss and the fully connected of the network as classifier. We will use the accuracy of this model as a lower bound.

4.2. LwF and iCaRL

We implement **LwF** and **iCaRL** as detailed in [5]. In the first, we equip our architecture with a distillation loss.

In the latter, in addition to the distillation loss, the exemplars of the old classes are stored and we use NME-classifier instead of the FC layer. This classifier computes the average feature vector μ_y among the exemplars for each class y. Each new image x is then assigned to the class y^* whose average feature vector is closest to $f(x)$:

$$y^* = \arg\min_{y=1,\dots,t} \|f(x) - \mu_y\|_2 , \quad (1)$$

where $f(x)$ is the features vector extracted from x.

4.3. Ablation Study

The main components of our methods are classification loss, distillation loss and classification algorithm. To assess the impact on the final performance of each of these, we conduct an ablation study comparing the accuracies obtained from new combinations of them. The results for each model are reported in *Figure 2*.

4.3.1 SVM Classifier

The SVM classifier is algorithm that consists in looking for the hyperplanes that best separate the classes in the space of the inputs. We applied this classifier to vectors obtained using the network as **features extractor**. Similar to NME, this is a geometric approach; it has two main parameters: C and kernel. For setting these values, we have performed fine-tuning processes: the kernel is tuned once and for all, while the value of C is calculated by a Grid-Search at each iteration.

4.3.2 Bias Correction Layer

As shown in [6], incremental learning methods using exemplars are biased towards new classes, and the more incremental batches arrive, the stronger the bias. **BiC** exploit this issue by adding a Bias Correction Layer on top of the deep architecture that performs a linear transformation outputs of the model of the new classes. This layer has only two parameters, α and β, shared by all new classes. Let o be the new classes logits, the *Bias Correction Layer* performs the operation $x = \alpha \cdot o + \beta$, while the old outputs are kept unchanged.

The effectiveness of the method is based on a two-step sequential training. Specifically, at each incremental step, we extract a small balanced *validation set* from the available training data (new class data and exemplars). The remaining part is used for training the architecture, then we freeze it and we train the bias correction on the validation set. We will use this model for a further analysis in Section 4.4 and we refer it as **BiC Method**.

4.3.3 Distillation losses on features

The idea here is to apply the distillation loss to the output of the feature maps. In this case, we try both with a L1 Loss (Least Absolute Deviations) and a L2 Loss (Least Square Errors), keeping the classification loss and the classifier as in iCaRL proposal. The L1 and L2 Losses are defined as:

$$L1Loss(x) = \|f_{old}(x) - f_{new}(x)\|_1 ,$$

$$L2Loss(x) = \|f_{old}(x) - f_{new}(x)\|_2 ,$$

where $f_{old}(\cdot)$ and $f_{new}(\cdot)$ are the feature extractors respectively for the old and new models.

Based on an intuition in [3], we refine the model by adding a balancing step: as the ratio of new to old classes decreases, we want the distillation loss to become more effective. So, we multiply the distillation loss by parameter γ, that works as follows:

$$\gamma = \gamma_{base}\sqrt{|C_n|/|C_o|}, \quad (2)$$

where, at each task, C_n is the number of new classes, C_o is the number of old classes and $\gamma_{base} = 5$.

4.3.4 Cosine loss

In [3] a further method for incremental learning is presented, consisting of three components: a new distillation loss, a new classifier and a new separation loss. In this project we study and re-propose their weighted distillation loss together with the iCaRL components. The structure is:

- *classification loss*: Cross Entropy Loss, that minimizes the distance among the predicted labels and the true ones;

- *distillation loss*: Cosine Similarity Loss, that minimize the cosine among two vectors. In this case is used to force the features vector at a given task to stay as close as possible to the correspondent one at the previous task:

$$\ell_{cosine}(x) = 1 - \cos\left(f_{old}(x), f_{new}(x)\right)$$

- *classifier*: as classifiers we tested both the NME from iCaRL and the Linear Fully Connected layer of the network.

4.4. Open World Recognition

Now we endow **iCaRL** with a *rejection capability* such that it can classify an instance as *unknown* if its predicted probability is lower than a given threshold.

For this goal, we take the output of the model and obtain probabilities through a *softmax* transformation. We then take the maximum probability and the corresponding label. If this probability is higher than the threshold, we classify the instance with the corresponding label, otherwise we predict it as unknown.

This leads the model to perform a more rigorous evaluation, since in order to assign a certain class to an instance, a minimum degree of confidence with the choice is required.

We first use a fixed threshold and then repeat the training on a set of new thresholds to understand their impact. Moreover, since *iCaRL* does not use probabilities in its classification mechanism it is not suitable for this task, so we repeat the experiments with *BiC Method*.

4.5. Our Modifications

4.5.1 KNN on probability distributions

The output of the last fully connected layer in a neural network, passed through a SoftMax layer, can be interpreted as a probability vector.

During classification, usually, the standard approach is to pass the object through the network, take the output vector and select from it the maximum probability; the correspondent class is the one assigned to the object. In a complex problem with many classes, such as the ones we are addressing in this project, it can be interesting to look not only at the current predicted classes, but also at other highly probable. One possible way to address this issue is to consider all the probabilities during the classification phase, instead of only the highest. Our idea is to use as classification algorithm a simple KNN; for an instance x, the input features are here the probabilities given by the last FC layer plus a SoftMax activation function. The KNN model is trained at each task, by performing a grid search for tuning parameter K.

4.5.2 Distillation among the whole network

Increasing knowledge of a model is a matter of trade-off between conservatism and progressivity of parameters during the training phase. We propose to reach this trade-off by introducing a new distillation loss; differently from the other seen in this report, our loss function has a random behaviour in selecting which layer should act. The distillation loss works as follows:

- at each epoch, a random network layer i is selected;
- for each image, the output of the layer i from both the new and the old versions of the network are computed;
- the cosine loss is calculated among those two features vectors.

Then the distillation loss and the classification loss are summed and back-propagated among the whole network.

4.5.3 Analysis on Exemplars choice

iCaRL [5] NME approach for exemplars choice could lead to good results when the images in a class are very similar to each other, but in case of different domains or dissimilar instances, this representation is not so useful: the network learns the images that are easy to classify, but fails with the others. So our proposal is to examine the impact in the choice of exemplars by comparing NME with two implementations:

- among m exemplars, $m/2$ are selected by taking the closest to the centroid, while the other half part is selected among those element that have the highest distance from the centroid (FME); the idea is to incorporate in the NME some variation given by instances that we can call "outliers" for our problem;
- all the exemplars are chosen by using the FME, Farest Mean Exemplar. The idea, in this case, is to try to capture all the variations in the features space.

4.5.4 Rejection strategy

Taking inspiration from [2], we learn class-specific rejection thresholds from data by minimizing a loss. Based on the analysis on the global thresholds we believe that **BiC Method** can be very promising for further analysis. So, we use this model as incremental classifier.

In each incremental step we learn the new model as usual. Then, we freeze the model, we learn a vector of thresholds Δ with many elements as the number of already seen classes by minimizing the following loss

$$L(\Theta) = \sum_{(x,y)\in\Theta} \sum_{k=1}^{K_t} \max\left(0, (m_k^y(\pi_k(x) - 0.99 \cdot \Delta_k))\right),$$

where Θ is a data batch, K_t is the number of already seen classes and π is the output probability vector of our model. m_k^y is equal to -1 if $k = y$ and 3 otherwise.

The loss is minimized over an held-out data portion. Specifically, we double the size of the validation set of BiC Method and we split it in two parts, one to learn the bias correction and one to learn the threshold. The first time we implement the method we set $m_k^y = -1$ if $k = y$ and 1 otherwise and we do not multiply the threshold by a factor but we find that the learned threshold was too low. We multiply the threshold by 0.99 in order to make the training procedure more elastic, *i.e.*, during training we shrink the threshold and we make the optimization less strict.

5. Results and Discussions

In this section we show the performances of our algorithms executed on three different random seeds (42, 24 and 1993) and averaged out. For sections 5.1, 5.2, 5.4.1, 5.4.2 and 5.4.3 we use the first versione of the dataset. While for sections 5.3 and 5.4.4 we use the second one.

5.1. Finetuning, LwF and iCaRL

We train our model and test it for 70 epochs with a batch size of 128, we initially set the learning rate to 2 and then divided by 5 in epoch 49 and 63. The optimization step is taken according to the *SGD* algorithm with momentum equals to 0.9, weigth decay 0.00001, while the loss and the classifier depends on the algorithm used. We kept a total of 2000 exemplars. As we expected, we can see that finetuning has the worst performances, indeed the testing accuracy in the second incremental step dramatically decreases from around 0.83 to 0.39 till drop to a value lower than 0.1. While introducing further improvements in LWF and in iCaRL we get a final classification accuracy of 0.24 and 0.4 respectively.

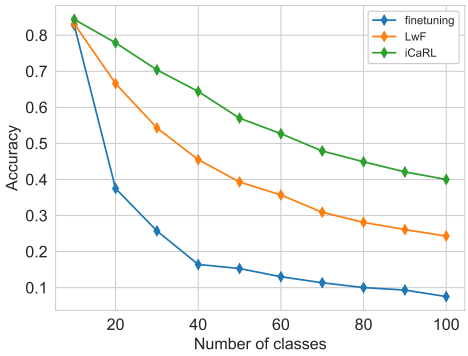

Figure 1. Comparison between finetuning, LwF and iCaRL.

Name	Classifier	Class. Loss	Dist. Loss
A	SVM	BCE *(o)*	BCE *(o)*
B	NME	BCE *(o)*	L2 *(f)*
C	NME	BCE *(o)*	L1 *(f)*
D	FC-Linear	CE *(o)*	Cosine *(f)*
E	NME	CE *(o)*	Cosine *(f)*
F	FC+bias corr.	CE *(o)*	Softmax *(o)*

Table 1. Ablation study, combining different losses and classifiers. *(o)* and *(f)* respectively indicate that the loss is applied on outputs or on features.

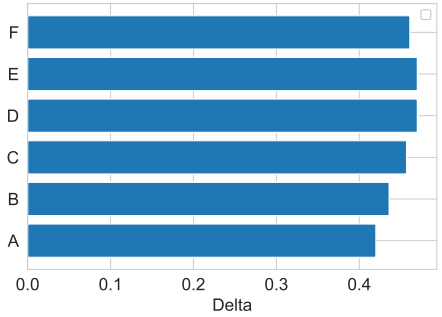

Figure 2. Ablation study

5.2. Ablation Study

We call *Delta* the evaluation metric used in this experiment, it measure the difference between the initial accuracy and the final one. That is, the accuracy degradation during the whole incremental learning, allowing us to focus on the catastrophic forgetting issue. Ideally we want it as small as possible. Looking at *Table 1* it's possible to see all the explored configurations; parameters are kept equals to iCaRL's, except for F where we set the initial learning rate to 0.1 and for D and E where we use those indicated in [3]. *Figure 2* shows the results for our implementations. Best results are achieved by using as classifier the SVM on features, keeping the same classification and distillation loss as iCaRL. It seems logical that BCE Loss performs better since for classification tasks, penalizing misclassifications is an important aspect. As for the Cosine Loss, it performs better with the NME Classifier than with the FC; in this case probably a tuning on the γ_{base} parameter could improve the results.

5.3. Open World Recognition

For evaluating the effectiveness of our rejection strategies we use the second version of the dataset and we consider 3 different criteria:

- *closed world without rejection accuracy* (CWA), the standard incremental learning scenario.
- *closed world with rejection accuracy* (CWA rej), as in

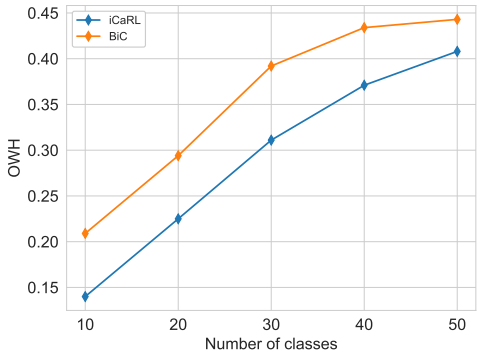

Figure 3. Comparison between iCaRL and BiC with a naive rejection strategy: fix a global probability threshold to 0.5.

	Number of classes				
	10	20	30	40	50
CWA	0.835	0.745	0.682	0.625	0.566
CWA rej	0.829	0.733	0.657	0.59	0.517
OWA	0.077	0.133	0.203	0.271	0.337
OWH	0.14	0.225	0.311	0.371	0.408

Table 2. Result table for the naive rejection strategy applied to iCaRL.

the previous point but in addiction our model now has also the rejection capability.

- *open world accuracy* (OWA): at each incremental step we test our model on the second part of the dataset which contains only unknown classes.

The last two evaluation metrics are the most important for this setting. The former evaluates the capability of our model to recognize that a class is known and correctly classify it. The latter evaluates the capability of our model to understand that a concept is still not known. In order to have a joint evaluation of them we consider the *Open World Harmonic mean* (OWH) which is the harmonic mean between the two. As first experiment we fix a global threshold to 0.5 and we evaluate the rejection capability of **iCaRL** and **BiC Method**. Figure 3 reports curves in terms of *OWH* of the two methods. As expected, BiC is better in each incremental step. On the other hand, it is unusual that the curves have an increasing trend, the reason can be found by looking at Table 2. Indeed, we can see that the *CWA rej* decreases while the *OWA* increases. We believe that the poor rejection capacity in the first steps is due to fact that whenever only few concepts are known the classification criteria are still very general making understanding what is not known even more difficult. Finally, we evaluate the other fixed thresholds, and find out that 0.8 is the best one concerning the harmonic mean, as shown in Table 3.

Thresholds	Number of classes				
	10	20	30	40	50
0.3	0.018	0.042	0.098	0.15	0.18
0.4	0.095	0.153	0.254	0.311	0.338
0.5	0.209	0.294	0.392	0.434	0.443
0.6	0.345	0.428	0.484	0.505	0.488
0.7	0.444	0.517	0.538	0.532	0.501
0.8	**0.524**	**0.578**	**0.566**	**0.537**	**0.484**

Table 3. Comparison in terms of Open-World Harmonic mean, between different fixed-global rejection thresholds.

5.4. Our Modifications

5.4.1 KNN on probability distributions

This approach doesn't lead to an improvement on accuracy comparing to iCaRL results; but the main goal of this method was to prove that the classification using the output probabilities has some potentiality, especially in case into the dataset there are similar classes that can be confused one to each other.

Example. In order to analyze the potential of this method, let's take an example: image 987 from the test dataset of CIFAR-100 has as class label "Girl" (class 35). Using the network FC output, we obtain the wrong result, because we obtain that the highest probability corresponds to class 11; but we notice that the second one is class 35, that is the correct one. When we pass this probabilities vector to the KNN model, it correctly classifies the instance. In this case, so, our approach is able to solve the issue.

5.4.2 Distillation among the whole network

The results of this part are available in *Figure 4*, where we compare iCaRL and our method *Random Distillation Loss* RDL. Although the accuracy is not satisfactory, an analysis of the type of classes predicted shows that distillation loss over the entire network actually increases conservativeness. In fact, with a starting learning rate of 0.5 and the rest of the parameters equal to [3], we are able to force the model into predicting older classes. Further analysis on the number of predicted elements for each class showed the great importance of the parameter γ_{base}, which with a more precise finetuning could strongly and positively impact the final results.

5.4.3 Analysis on Exemplars choice

Here the analysis compare iCaRL results with our two approaches. Results are available in *Figure 5*, from where it turns out that our implementation with the mixed strategy actually improves iCaRL solution (delta is around 2% better). As expected, taking exemplars so that they better capture variation and also the various types of class images

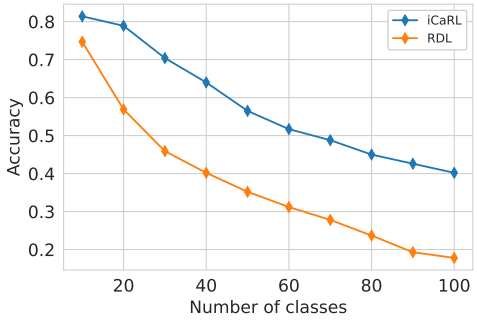

Figure 4. Distillation Among Network Analysis

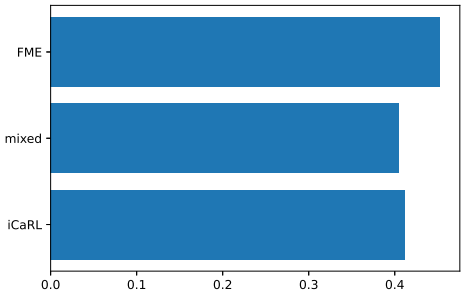

Figure 5. Exemplars Analysis

leads to better representation of the dataset, and thus better results.

5.4.4 Rejection strategy

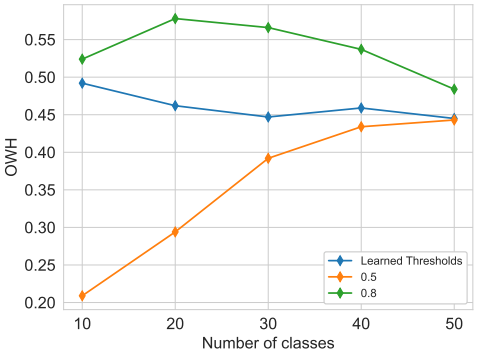

Figure 6. Comparison between class-specific and fixed-global thresholds

Here we report the results that we find after modifying the way on which we learn the rejection thresholds. We use *Adam* as optimizer with initial learning rate set to 0.01 and we decrease it by a factor of $1/100$ every 50 epochs for a total of 250. We set the weight decay to 0.0002.

	Number of classes				
	10	20	30	40	50
CWA	0.784	0.728	0.663	0.607	0.547
CWA rej	0.695	0.652	0.585	0.532	0.477
OW	0.382	0.358	0.361	0.405	0.418
OWH	0.492	0.462	0.447	0.459	0.445

Table 4. Results table for our proposed modification for learning class-specific rejection threshold from held-out data.

Table 4 shows that the model performs well in closed world as well as in the open set scenario.
Figure 6 compares our proposal with the naive strategies of global thresholds. We observe that the performances are still lower then the those with global threshold 0.8, so this global threshold seems to be a very good candidate. However, we believe that this may not be always true and 0.8 can be only a good guess for this specific dataset and model. Moreover, it seems to follow a decreasing trend, so if we could performs more incremental steps, the performances could become poorer. While our proposal appear to be more stable and reliable.

6. Conclusion

We have experimented that the use of distillation and the selection of few exemplars can alleviate the problem of catastrophic forgetting. However, the models performances are still not satisfying.

In the open world scenario we have understood that thresholds learned from data are more reliable. Moreover, the learning procedure is computationally efficient and require only a little held-out data portion. Finally, more tailored losses can be designed for obtaining better thresholds.

References

[1] F.Castro, M. Marin-Jimenez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. 2018.

[2] D. Fontanel, F. Cermelli, M. Mancini, S. Bulò, E. Ricci, and B. Caputo. Boosting deep open world recognition by clustering. October 2020.

[3] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. June 2019.

[4] Zhizhong Li and Derek Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016.

[5] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725, 2016.

[6] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. 2019.