Open World Recognition in Image Classification

Alessia Scandaliato
Politecnico di Torino
s280067@studenti.polito.it

Chiara Tomei
Politecnico di Torino
s277790@studenti.polito.it

Daniela Martorana
Politecnico di Torino
s267742@studenti.polito.it

1. Introduction

A major problem in artificial intelligence is the difficulty of having new knowledge without incurring in the Catastrophic Forgetting event. Incremental learning systems that learn about new concepts over time are one of the greatest forward steps in this context.
Many artificial object recognition models can only be trained in batches, where all classes are known in advance, and the training data of all classes can be accessed simultaneously. It becomes necessary to find more efficient strategies to handle large-scale classification problems. Many approaches have been found to manage this issue and in this paper we will show some of them. In addition, we implemented an ablation study in which different losses combinations and classifiers are applied. Moreover, a Deep Open World Recognition framework has been implemented. The ability of convolutional neural networks is often limited to a closed world scenario, where the number of classes to be recognized is limited by the available training set. For this reason we need to add the capability of rejecting images that belong to unknown classes. Finally we propose our modifications:
1. A modification in the metric used for the rejection strategy.
2. The addition of two layers in the neural network to extract the most relevant features

2. Related Works

Some related works regarding incremental learning and open world scenario are presented below.
Finetuning. This is a sort of baseline, it learns and optimizes the parameters of a network for new incoming classes without preventing the catastrophic forgetting phenomenon. Indeed, the shared parameters among tasks change without new indications for the original parameters and thus only the last categories analyzed can be recognized, while the others parameters are progressively overwritten.
Learning without Forgetting. [2] uses only examples for the new task and optimizes both for high accuracy for the new task and for preservation of responses on the existing tasks from the original network. It uses the concept of Distillation Loss, applied to the output of the network at the previous steps in order to encourage the output probabilities of new classes to be close to the original output. It is important to point out that the functioning of this method probably depends on how similar the old task data are to those of the new task.
iCaRL. iCaRL [4] allows learning in a class incremental way, by keeping only a small portion of training data (called exemplars) from previously learned classes and by adding progressively new classes. The storage and the selection of exemplars are based on a technique called *Herding*. Moreover, in order to avoid catastrophic forgetting is used a combination of knowledge distillation and classification loss. Finally, iCaRL uses a technique named nearest-mean-of exemplars (*NME*) to classify test samples.
BDOC [1], **Knowledge is never enough** [3]. BDOC tries to break the closed world assumption. It tries to identify whether an instance belongs to the set of known categories or not. Among its different contributions it proposes a strategy to learn class specific rejection thresholds. In [3] the authors extend the NCM algorithm to the Open World Recognition (OWR), starting from the so called Nearest Non-Outlier (NNO) rejection criterion and improving it by an algorithm named DeepNNO.

3. Background

In this section we will describe the baseline settings used during the experiments carried out.

3.1. Dataset

All the experiments were carried out on CIFAR-100 Dataset. It is composed of $60k$ images distributed in 100 different classes. In particular some splits are already provided, indeed the dataset is internally divided in $50k$ images for the training procedure (500 per class) and $10k$ for testing (100 per class). At each learning step a batch of 10 classes was used to train the model, the test set instead, includes all the images of the classes seen so far. We have randomized

the order of the classes by means of three random seeds.

3.2. Network

In this study the model used was a ResNet-32 optimized for the CIFAR-100 Dataset. The main difference with respect to the other ResNet is in the first layer: since the input images on the used dataset are 32x32, it is composed of one convolutional layer 3x3 with respect to the original 7x7, with stride and padding 1, to match the output size with the input size, followed by a batch normalization. The ResNet has been used for the training, classification and features extraction procedures. In order to do so, since generally the net predicts the classes labels, another modification concerns the last layer of the net: in iCaRL the last Fully Connected layer was used just for the training and another function is used for the features extraction. The FC is set from the beginning to have 100 output nodes.[1]

4. Method

In this section, will be presented some of the models introduced above with our implementations. For what concerns the incremental learning part, all the models will be trained progressively on a new set of classes that will arrive at each iteration and will be tested on all classes seen so far. All the models that will be presented are trained with the hyper-parameters of [4], that are the following: Batch size: 128, LR: 2, Momentum: 0.9, Weight Decay: 10^{-5}, Number of epochs: 70, Gamma: 0.2, Step sizes for the LR are set at the 49 and at 63 epoch. It has been used the ResNet32 described above as reported in [4]. Each model has been performed with 3 different random seeds in order to shuffle the classes at the beginning in a different way, with the aim of capturing variations among the different runs, or vice versa, some stable behaviour. The implementation of all models relies on the use of PyTorch library in order to exploit the GPU capabilities provided by this.

Fine Tuning Baseline.
The first model that has been implemented is Fine Tuning. For each group of class, a complete training of 70 epochs is carried out, with a validation phase in order to choose the best model. The loss used was the BCEwith-LogitLoss function, that is applied on the output of the network on the current 10 classes. Clearly, for the first group of classes the accuracy is high, but for all other groups, the catastrophic forgetting phenomenon occurs. Indeed, the parameters of the network are optimized just for the last group of data and so the parameters learned before are overwritten and forgotten. As shown in the confusion matrix in Figure 2, the model at the end is able to predict correctly only the last 10 classes.

[1]The code of the implementation of the ResNet can be found at: https://github.com/Alessia-Sc/Open-World-Recognition.git

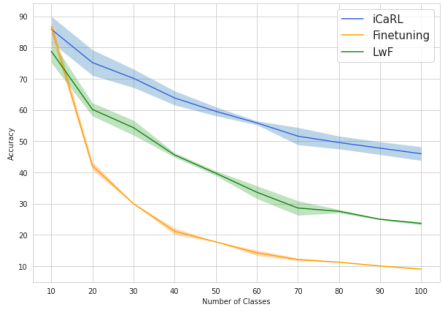

Figure 1. Comparison of mean accuracy and variance, between different techniques, with three different seeds.

Learning Without Forgetting.
Also this model is trained in an incremental way. Lwf has the purpose to learn the new tasks' parameters without degrading performance on old tasks, sharing old parameters, and without having access to the old training data, because this is infeasible for memory constraints. This is done by exploiting the concept of Distillation Loss, explained above. It is computed with BCEwith-LogitLoss function, like for the classification loss. The loss is composed by the sum of two factors (that represents the union of classification and distillation loss). For the classification loss the output of the current network is compared to the ground truth, in the distillation loss the output of the current network is compared to the output of the new images forwarded throw the network at the previous stage.
By minimizing this total loss, the model can learn new classes but also reduce the forgetting of old ones. The results show that this method provides a big improvement with respect to Fine Tuning, in that LwF recognizes and classifies well also the old classes, but the confusion matrix in Figure 2 shows that there is still a bias toward the learning of the new classes and that the older are progressively forgotten. This problem, that is addressed by iCaRL, is probably due to the fact that the model does not retain any data relating to the previous classes.

iCaRL.
In order to improve the performances of LwF, iCaRL [4] introduced two main contributions. First it introduces the concept of exemplars and second it changes the way the model classifies.
More in detail, for each class, the exemplars are the images forwarded in the network that are the closest to the vector that represents the mean of that class. The number of total exemplars is fixed and it's equal for each class, so as other classes progressively arrive, the number of exemplars for each class is reduced. This concept, combined with the concept of distillation, proved to be fundamental for what con-

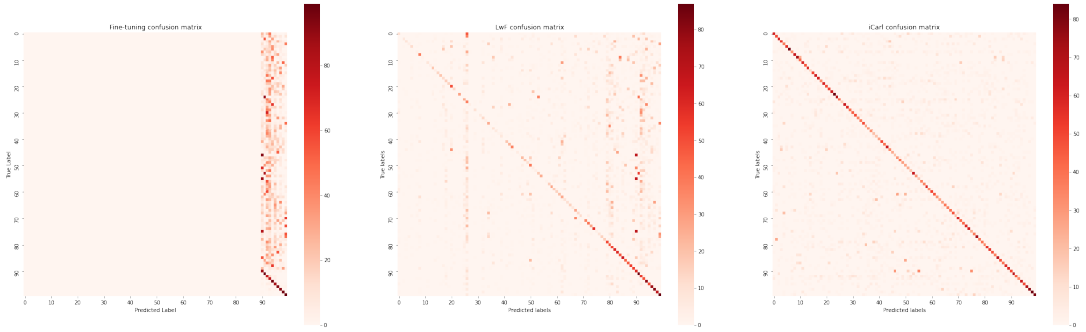

Figure 2. Confusion Matrix computed with each different techniques adopted. (Left: Fine Tuning, Center: LwF, Right: iCaRL)

cerns the purpose of not forgetting old classes and keeping the memory usage limited. Indeed, each time a new class arrives, the network is trained on the images of new classes plus the exemplars of previously learned classes. However, progressively, as the number of classes increases, performances will inevitably tend to decline, but this decreasing trend is much less steep than the trend of the other techniques, as shown in the Figure 1.
In practice, for each group of 10 classes, the parameters of the network are updated throw the *update representation* function. Here the examplars together with the images of the new incoming classes are forwarded to the network. The loss function is computed as in LwF. After this step, the set of exemplars is reduced, in order to leave space to exemplars of the new classes. After updating the variable *m*, number of exemplar per class, the firsts *m* exemplars are selected. They are sorted by ascending closed distance from the class mean, by exploiting the concept of *Herding*. The exemplars per class are selected through the *construct exemplars set* function. For each class, the mean μ of the features vectors of images ($\varphi(x)$) have been computed and then *m* images that meet the following function have been selected:

$$p_k \leftarrow \arg\min_{x \in X} \|\mu - \frac{1}{k}[\varphi(x) + \sum_{j=1}^{k-1} \varphi(p_j)]\|$$

Finally for the classification, the model vectorizes the features of the exemplars of each class, computes the mean vector and then assigns to the input image, the label of the class which mean vector is the closest to the features vector $\varphi(x)$ of the current image, by exploiting the concept of NME classifier (instead of using a FC).
The confusion matrix in Figure 2 shows less misclassification than the one in LwF: the diagonal is more highlighted and there are very few points outside.

The Figure 1 shows a comparison between the results obtained with our implementation of the three different techniques. Throw the variance, it can be seen that a different seed, and thus a different order of classes, influences the accuracy and the trend of the model.

5. Ablation Study

In this section we will describe the ablation study we have made about losses and classifiers. In each case that we have analysed we have kept the same baseline of iCaRL. For this section we have decided to compute every run using only the seed that performs better in the iCaRL's section. This choice is justified by the fact that in this study the main purpose is to show the trend.

5.1. Losses

Regarding the different combinations of losses, we have decided to replace the combination BCE + BCE proposed in iCaRL with the following combinations.

5.1.1 BCE + L1

As a first combination we have chosen to keep the BCE loss for the classification part. It is particularly suited for binary problems and is composed of two types of contributions: the probability of having classified the class correctly and the probability of having misclassified it.
We decided to use L1-norm loss for distillation, it minimizes the sum of the absolute differences (S) between the target value (y_i) and the estimated values ($f(x_i)$), in our case between the output of the old network and the new one.

$$S = \sum_{i=1}^{n} |y_i - f(x_i)| \quad (1)$$

L1 loss is not good in detecting outliers. This may be helpful in studies where outliers may be safely and effectively ignored. If it is important to pay attention to any outliers and find a more stable solution, the method of L2 (or MSE) is a better choice and for this reason we will analyze it later.
As shown in Figure 3, with this combination of losses, in the firsts iterations the model works very well. However, the robustness of BCE loss overcomes the L1 loss, making the performances of model decrease quite rapidly from the fifth

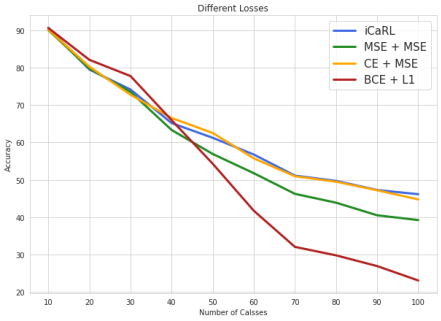

Figure 3. Comparison between different combinations of losses.

iteration, ending up with performances lower than the other losses taken into consideration. Also the opposite is important: the classification loss should not be overwhelmed by distillation one.

5.1.2 MSE + MSE

Then, we have decided to use the Mean Square Error both for the classification and distillation losses. It minimize the sum of the square of the differences (S) between the target value (y_i) and the estimated values ($f(x_i)$):

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{2}$$

Intuitively, since MSE squares the error, the model is much more sensitive to big errors. For the classification part the loss receives as input the *softmax* of the output of the images given by the current model and, as target, the corresponding labels one-hot encoded. For the distillation part the loss takes the *sigmoid* functions applied on the outputs of new and old model. The two resulting losses are then summed. As can be noticed in Figure 3, using this type of loss the model outperforms iCaRL in the short term but at the same time the performance seems to fall down on the finals batches and we can suppose that for more than 100 classes this could be a bigger problem.

5.1.3 CE + MSE

Finally, we have chosen the Cross Entropy loss for the classification loss. The CE loss is a non-binary version of the already seen BCE loss. The penalty is logarithmic in nature yielding a large score for large differences close to 1 and small score for small differences tending to 0.

$$-\sum_{c=1}^{M} y_{o,c} \log p_{o,c} \tag{3}$$

So, according to the formula, $y_{o,c}$ is the binary indicator that tells us if the class c is or not the correct classification for the sample o, $p_{o,c}$ indicates the probability that the sample o belongs to the class c.
For the distillation part we decided to use the MSE, after having applied the *softmax* function to the outputs.
Finally, the two different losses have been summed and averaged according to the batch size. In this particular combination of losses the learning rate is scaled by a magnitude and set to 0.2 instead of 2.
In the Figure 3 it can be seen that this combination of losses could be an alternative to iCaRL, since the performance tend to be very close to the BCE one.

5.2. Classifiers

In this section we have decided to investigate the behaviour of three different models comparing them with the Nearest Mean of Classifier (NME) used in iCaRL:
SVM, the best classification model before the incoming of Deep Learning,
KNN, similar to the roots behind NME and
Random Forest, a simpler classifiers based on decision rules.
All the classifiers were trained on the exemplar sets, constructed with iCaRL, over each batch of 10 classes.

5.2.1 Support Vector Machine (SVM)

The first classifier that has been tested was SVM classifier with RBF kernel. After some trials we found that the best setting was with C=1. This classifier overall gives us the best results among all three variations as it can be seen in the Figure 4. Its performances are similar to the one of iCaRL, but it remains lower.

5.2.2 K-Nearest Neighbors (KNN)

For the second classifier we have tuned the different values of K as long as we found the best one around 10.
As can be seen in Figure 4, KNN struggles a lot in learning new classes as the number of steps increases, and in general it provides worse results with respect to NME. We can suppose that the algorithm found difficulties in separating classes one another and tune the size of neighborhood is not sufficient for getting good results. Another possible problem that affects KNN is the curse of dimensionality.

5.2.3 Random Forest

With this method, the decision rules split the space and classify the images. We set the number of estimators to 100 and it was found to be less efficient than the other classifiers. Also here, a possible reason for that could be the curse of dimensionality. Moreover, decision rules usually do not work so well in the context of image classification since the categories of images are so many and it is difficult for random forest to split the space properly.

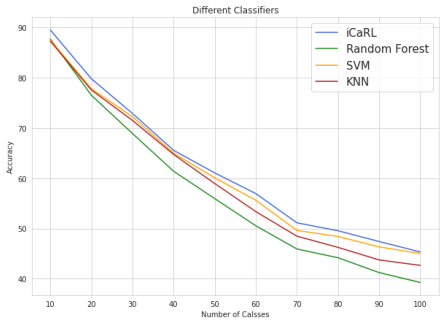

Figure 4. Comparison between different classification models.

To conclude, especially when the number of exemplars starts to decrease, none of these models can achieve NME performances. Indeed, iCaRL outperforms very well and the simplicity and effectiveness of it is rewarded for this type of classification problem, as described in [4].

6. Open World Set

In this section we will present our approach to the problem of an open world scenario. We divided CIFAR100 in two sections, splitting 50 classes for the known set and 50 classes for the unknown set. The Net is trained only on the known classes, in an incremental way, and then tested both on known and unknown classes. We started testing our rejection capability in the closed set scenario, then we added the open world scenario, in order to adjust the threshold properly and reach a good harmonic mean of the accuracy on both scenarios. We started with a naive rejection strategy using as classifier the complete ResNet with the Fully Connected layer. We set the threshold equal to 0.5 in order to compare it with all the probabilities of each instance to belong to different classes.
In order to compute the accuracy, every time that an instance has a probability to belong to a class > 0.5 is classified as known, accordingly to the net prediction, otherwise is classified as unknown. Specifically, if the current instance belong to the closed set scenario and it is classified as unknown in computing the accuracy that prediction will be considered as misclassified. Otherwise if an instance that belongs to the open set scenario is classified as unknown that prediction will be considered as correct. The results obtained are shown in the table below.

	Harmonic means				
N. of Classes	10	20	30	40	50
$\tau = 0.5$	73.47	66.17	61.95	55.05	40.66

Table 1. Harmonic Means

Different threshold will be tested in our improvement section.

7. Improvements

In this study have been implemented two different improvements, one regarding the different rejection strategies in the open world scenario, and the other one concerning the structure of the network used for iCaRL implementation.

7.1. Open World improvement

In our implementation we decided to follow a strategy closer to iCaRL and so we kept the NME classifier and extended his functionality to act also in an open set scenario, working on a threshold compared with a score computed on the distance.
We have kept the same baseline of iCaRL and we have introduced a score function applied to the distance obtained through NME. For this section we have taken inspiration from [1] and [3]. Differently from the NNO and DeepNNO approach we don't have transformed the distances into probabilities, we have just used similar formulas applied on the distance.

First Approach. In this first approach we calculated our score according to:

$$S_c^F(x) = 1 - \frac{d(f(x), \mu_c)}{\tau} \tag{4}$$

In this case the threshold that influences the rejection appears directly inside the score computation as τ. Using this metric we have set as a condition that the images with a score > 0 will be defined as known and so classified, according to NME, to the nearest class, while if the score computed is < 0 the image will be classified as unknown and the accuracy will be computed accordingly.

Second Approach. The second approach instead, takes inspiration from the DeepNNO calculation of probability, indeed our second score is computed as follow:

$$S_c^S(x) = e^{-\frac{1}{2}(d(f(x),\mu_c))^2} \tag{5}$$

Here the distance is computed as before, the threshold instead is not set inside the score but directly before the computation of the accuracy, which is managed as before.
We found out that the best trade-off is provided by τ = 0.5 for the first score (4) and a threshold of 0.88 for the second score (5).

We specify that as regards the first score it was necessary to decrease the threshold to obtain more skills in rejection, while for the second one it was necessary to increase the threshold to reject more images.

Third Approach. In the last improvement, we applied a mobile threshold in the baseline approach with FC. Indeed,

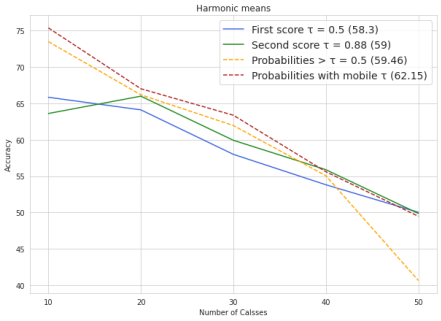

Figure 5. Comparison between different rejection strategies and thresholds.

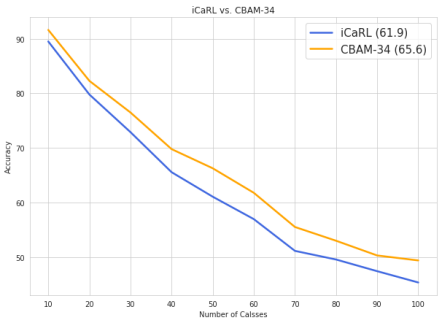

Figure 6. Comparison between different classification models.

we have noticed that as the number of classes increase, the fixed threshold is too low and the rejection capability performs poorly. For this reason, we have tried to gradually raise it, as new classes arrive.
It is important to point out that we tuned parameters and we set thresholds empirically without an exhaustive research over them.

We can analyze the results of our improvements in the Figure 5. In particular we can see that in the baseline implementation with the FC the performance drops dramatically as the number of classes increase. Instead, (4) and (5) have a less steep behaviour, especially in the last classes.
From this observation, we had the intuition of adding a mobile threshold directly on the baseline. This last improvement turned out to be the best.
In future works we would like to investigate better the mobile threshold also for (4) and (5) and tune it in a more exhaustive way and try to find a mathematical formulation.

7.2. Network Improvement

We have considered an improvement of the network to extract better features, since one of the major shortcomings is the low-data problem, that makes the features of each test images not representative for the true class distribution. For this reason we looked for a network that would place more attention on the features of the selected images.
We were inspired by [5] where Spatial and Channel attention modules were applied. The idea was to add these modules to an existing ResNet, which, given a class feature map, allow to highlight the target object and emphasize the meaningful features. As can be noticed in the Figure 7, the modules are applied to an intermediate feature map at every convolutional block of the network.

Channel Attention Module, focuses on "what". To do that it aggregates the channel information using *max-pooling* operations that highlights the main element of the image, and *average-pooling* operations that makes the picture softer. As shown in Figure 8, both elements are then forwarded to a shared network composed of a multi-layer perceptron with one hidden layer. After the MLP a *sigmoid* function is applied and the channel attention module is created.

Spatial Attention Module, focuses on "where" and complements the Channel attention one. It uses the two outputs that are pooled along the channel axis and forwards them to a convolution layer. Also in this case a *sigmoid* function is applied and the spatial attention module is created.

We applied these two modules on different ResNet. In particular we started from 32 and 18 in order to do a direct comparison with iCaRL model. Then we also explored other structures: ResNet 20 and 34. As can be seen in Table 2, our implementation with "CBAM" overcome iCaRL almost in all experiments. The Figure 6 shows that CBAM 34 in particular gave a not negligible improvement. We tried also more deep architecture but they were too much computationally and time expensive and moreover the results on the firsts iterations weren't the best ones.

8. Conclusion

In the table 2 have been summarized the results obtained in this work.

8.1. Future works

- First, an idea of future improvement of iCaRL baseline could be to increase the number of exemplars as the classification goes on and the number of classes increases; or to distribute them unevenly for classes, based on the accuracy of the predictions on them.

- The second idea regards a different balancing between distillation and classification loss, especially for the combination of losses with different strength.

Method	Accuracy mean
Fine Tuning	25.65
LwF	43
iCaRL	61.9
KNN	59.45
SVM	60.76
RF	57.16
MSE+MSE	58.52
CE+MSE	62.04
BCE+L1	52.43
iCaRL ResNet18	62.7
CBAM32	62.95
CBAM18	65.49
CBAM34	65.64
CBAM20	61.7

Table 2. Different Methods Comparison

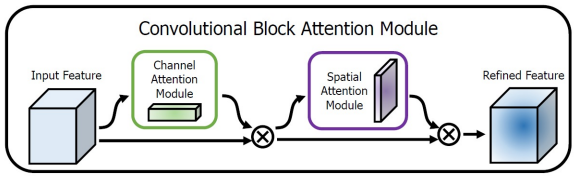

Figure 7. CBAM Scheme.

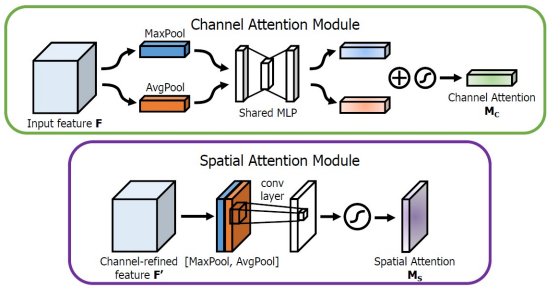

Figure 8. Channel and Spatial Attention Modules.

References

[1] M. Mancini S.R. Bulò E. Ricci D. Fontanel, F. Cermelli and B. Caputo. Boosting deep open world recognition by clustering. *IEEE Robotics and Automation Letters*, 2020.

[2] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*.

[3] E. Ricci P. Jensfelt M. Mancini, H. Karaoguz and Barbara Caputo. Knowledge is never enough: Towards web aided deep open world recognition. *ICRA 2019*.

[4] G. Sperl S.-A. Rebuff, A. Kolesnikov and C. H. Lampert. icarl: Incremental classifier and representation learning. *CVPR-17*.

[5] J.-Y. Lee S. Woo, J. Park and I.S. Kweon. Cbam: Convolutional block attention module. *ECCV 2018*.