Incremental Learning Project - Machine Learning and Deep Learning

Piero Ciffolillo S278184

Mattia Cappelli S278421

Abstract

An important open problem in deep learning is the problem of the incremental learning. Typically, when a convolutional neural network receives incrementally batches of training data, it is not able to adapt itself learning new data and at the same time remembering the old ones: what really happens is an issue known as catastrophic forgetting.

To let CNN able to avoid that behaviour is very important when we have stream of data, without the possibility to always recover old ones, as in many real cases happen, but it is hard to achieves principally because the network has in parallel to maintain itself capable to change, and it has structurally limited resources. This is an equilibrium problem well known as stability-plasticity dilemma.

In the field of image classification many researches recently proposed their solutions. In this paper, there are point out some state-of-the-art methods and then we will explore some of them looking for the critical points in order to improve them.

1. Background

The overview of existing solutions explores different baselines: finetuning training, Learning without Forgetting and iCaRl methods. On the other hand the upper bound result considered is achieved with the joint training, when all data are used for the training as in an unique batch (not incremental setting).

The dataset used is the CIFAR-100 that contains 60000 images divided in 50000 for the training set and 10000 for the test set. The classes are 100 with 500 samples for each one in the training set and 100 in the case of the test set. For the incremental learning, the dataset is divided into 10 batches composed of 10 classes.

The experiments in this paper are developed in Pytorch. For all of them the convolutional neural network is a ResNet-32 version of the Cifar Resnet (a ResNet optimized for CifarDataset), the optimizer is the SGD and the training epochs for each batch are 70, with mini-batches of size equal to 128. As scheduler for the learning rate we used a multi-step scheduler, enabling to change LR at determined epochs.

1.1. Upper bound

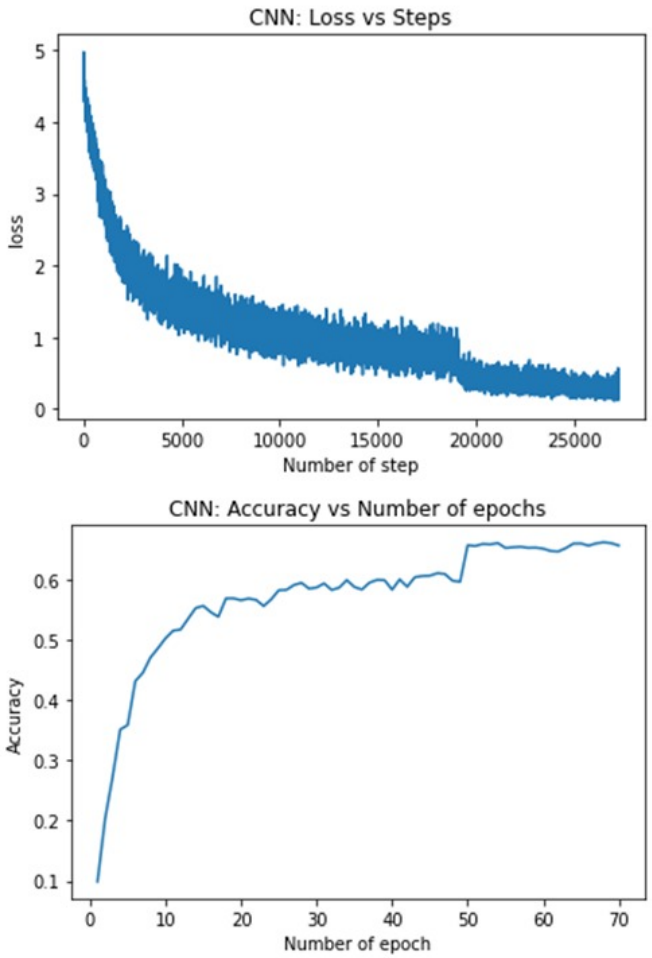

Figure 1. Accuracy scores over epochs and loss over steps for joint training.

For the upper bound the model is trained with the whole training set. The mean accuracy is about 0.67 with a variance of 0.85. From the confusion matrix in **Figure 2** , it's

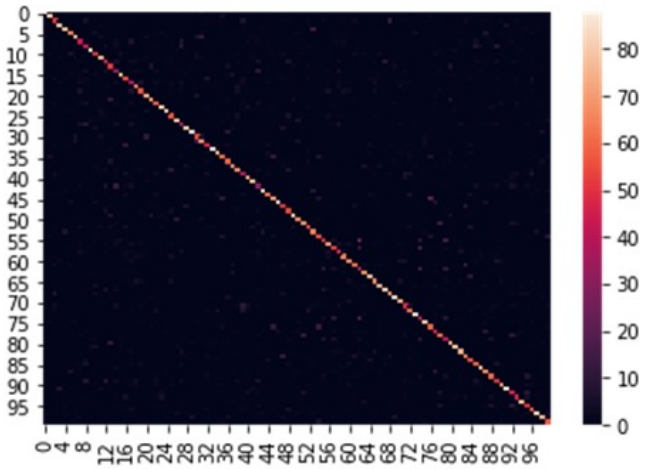

Figure 2. Confusion matrix in joint training.

visible the diagonal that prove that the model is able to recognize the respective feature in the right way.

1.2. Fine-tuning

The simplest method to deal with the Incremental learning is the training of the net incrementally without any precautions. Given n batches for training and test set take the i-th training batch for train and for the test the union between the current and the previous test batches for the test.

With this approach, the behaviour of the accuracy is clearly different and even if the accuracy on the first batches is higher with respect to the joint training, starting also from the second class the accuracy decreases. As we can see from the confusion matrix in **Figure 3** of the last batch the predictions are sparse and the classes that the model identifies better are the last 10 thus the model has not any remembrance of the previous classes.

1.3. Learning without Forgetting

The Learning without Forgetting paper intruduces a solution enabling the network to avoid the remarked degradation behaviour that we demonstred with the fineting. Such a solution is based on a distilling procedure, that saving the network trained on the previous batch, uses the softmax logits of images of the current batch passed on the old network as targets in an additional loss term, that is called distillation loss.

Knowledge distillation is in general a method useful for model compression, in which there are two models, the first of them is pre-trained, and one has to transfer knowledge from the first to the second one. Such a training setting is sometimes referred to as "teacher-student", where the pre-trained is the teacher and the new model is the student. In our case the teacher is the old network and the student is the current network, that tries to mimic outputs for old classes

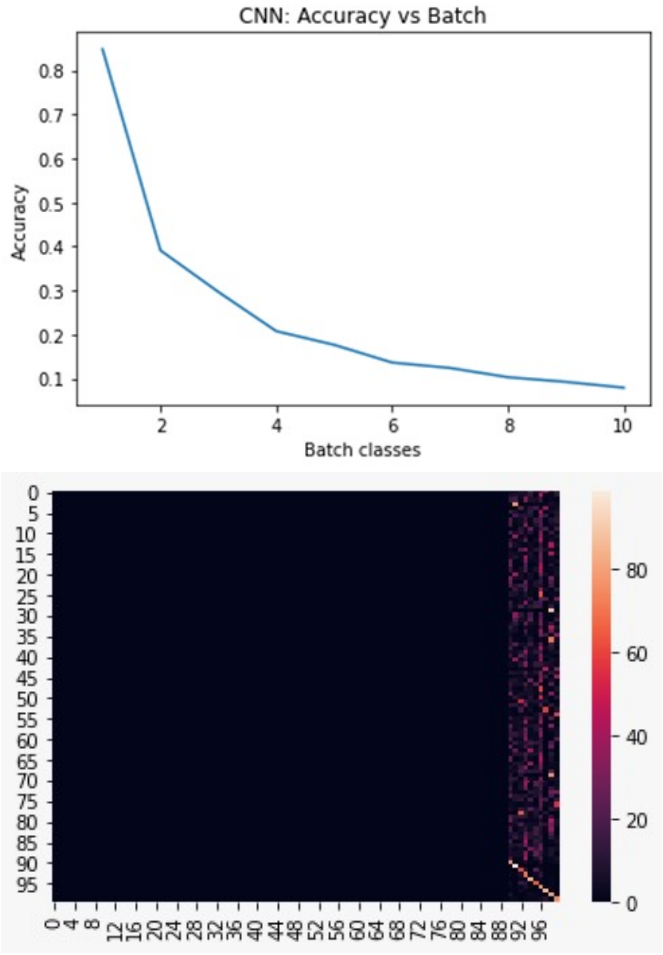

Figure 3. Confusion matrix and accuracy scores over batches with fine-tuning method.

of the teacher in order to not forget those classes. This loss introduces in other words a regularization during the training on the new classes, avoiding let the network change in a way that allows catastrophic forgetting. If we denote - for a generic input x of the network - the output logits of the old and new models as

$$\vec{o}_{old}(x) = [o^1_{old}(x), ..., o^n_{old}(x)]$$

and

$$\vec{o}_{new}(x) = [o^1_{new}(x), ..., o^n_{new}(x), o^{n+1}_{new}(x), ..., o^{n+10}_{new}(x)]$$

respectively - considering 10 new classes in the current batch and n classes seen in the previous batches - the distillation loss is formulated as follows:

$$Loss_{distill} = -\sum_{i=1}^{n} \pi^i_{new}(x) \log[\pi^i_{old}(x)]$$

where $\pi^i = \textbf{softmax}(o^i)$.

The **classification** loss is instead a classical Cross Entropy loss, **computed only on the 10 new outputs** . The total loss is the sum of the two parts.

1.3.1 Implementation details

We use hyperparameters described in Table 1, reaching a mean accuracy of on the last batch, so in line with results of LwF paper. As shown in the accuracy graph and the confusion matrix the method alleviates an heavy degradation till the last batch but the main problem remains a visible forgetting just from the second batch. **Detailed scores** are listed in the **Appendix** together those of each Section.

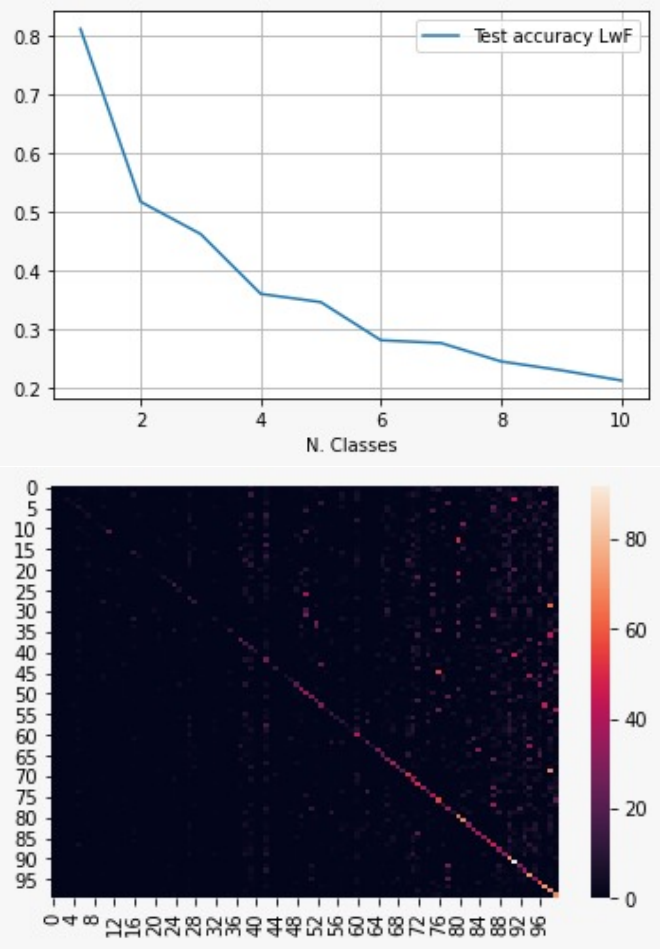

Figure 4. Confusion matrix and accuracy scores over batches with Lwf. They are visually better than fine-tuning but also so far from joint training.

Initial LR	Milestones	Gamma	Weight decay
2	[49, 63]	0.2	1e-05

Table 1. Hyperparameters setting table for the baselines.

1.4. iCarl

The incremental Classifier and Representation Learning paper presents a method to overcome some problem of incremental learning. The model uses the distillation loss like in the LwF method, and the loss utilized for the basic version is the binary cross-entropy loss. There are two main differences from the LwF method, the use of memory and of the NME for the classification. The memory is utilized to store some exemplars that they will be used during the training to 'refresh' the past classes in the next batches. The memory is fixed to a total of 2000 exemplars and for each step is reassigned based on the number of classes seen. Indeed the number of exemplars is equal to the number of the memory size divided the seen classes, in this way the number of exemplars is partitioned between all classes to have the same number of exemplars for each class. Specifically we merged the uploaded list of selected exemplars with each new training batch before training the net at each new incremental step.

The exemplar selection consists to find the better samples basing on a metric that takes care iteratively of the previously selected exemplars and of the class mean of the features extracted. This selection is done for each class. The exemplar set is a priority list, so during the new batch, the exemplars of the previous classes are simply reduced to the new sized calculated removing the lasts exemplars that exceed from the size.

The Nearest-Mean-Exemplar classifier works calculating first the mean of the features for each class, then found the nearest mean to a features image and assigns as a prediction the number of the class of the nearest mean (for the previous class the mean is calculated on the exemplars features but for the current batch the mean is calculated on the all training batch). This classifier reveals itself more performative compared to the FC layer as intrinsically less sensible to a bias toward new classes.

1.4.1 Implementation details

The learning rate start from 2 and then is divided by 5 at epoch 49 and 63, the total number of epochs is 70, the weight decay is set to 0.00001 and the mini-batch-size is 128. The training data are augmented with a horizontal flip transformation. The convolutional neural network used is the ResNet32 both for the feature extraction and the classifications. The normalization is done after the features extractions and after each operation on it like the calculation of the means. The distillation loss is implemented in the same way described for the LwF method, but using BCE (Specifically, we used BCEWithLogits by Pytorch that requires one hot targets for the class. part and sigmoid of old outputs of the old net as target for the distill. part).

Results for each batch are listed in Table 2.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	89.1	73.0	69.07	63.2	60.06	56.95	54.96	51.46	48.19	46.3	61.23
2204	80.4	74.95	69.4	64.95	62.06	59.45	54.83	52.4	48.2	45.21	61.18
1345	91.9	78.75	70.17	65.22	61.64	58.55	54.23	51.15	47.39	46.5	**62.55**
Mean on seeds	87.13	75.57	69.54	64.46	61.25	58.32	54.67	51.67	47.93	46.0	61.65

Table 2. These are the results for each seed achieved with the baseline implementation of iCarl.

2. Ablation study

In this section are described some variations on the iCARL model in order to improve it. The variations regard the classification model and the type of losses utilized.

2.1. Different Classifiers

For the classification, we tried three models: KNN, Cosine Similarity and the SVM, applied at the feature representation of test images after training the network for each batch.

The best model considering the three seed is the NME but the difference from the Cosine similarity in terms of accuracy are very small. Indeed the Cosine Similarity is very similar to the NME as the main difference is the type of metrics utilized. The KNN is set with a number of neighbours equal to 5 and the accuracy trend is worst than the previous classifier.

The KNN is a good classifier considering the online training task that the incremental learning requires, it's very easy to use with new data from different size and type but from the other hand it's too general and we can't infer any information of the data except the K. Another pro is that it is very sensitive to the outliers (in the last batches, the previous class that are fewer than the new one can be seen in this way). The SVM is unexpectedly the worst between the three and it is set with C=1 and the RBF-kernel. The main problem of the SVM for this task is the Hyperparameters selection that is very difficult to be set considering the great changes in the training at each batch. The principal reason to choose this classifier is that it works very well with high dimensional data.

For the implementation of KNN and SVM, we used the model provided by the sklearn package and regarding the data utilized for test and train we used the features extracted from the CNN and then normalized.

2.2. Modification of Losses

We use three different combination of losses for the classification and the distillation, with respect to the original version of iCarl: Cross Entropy for both (we implemented an our version for the distillation instead using the one provided by Pytorch), Cross Entropy for classification and Binary Cross Entropy for distillation, Cross Entropy for classification and Smooth L1 Loss for distillation.

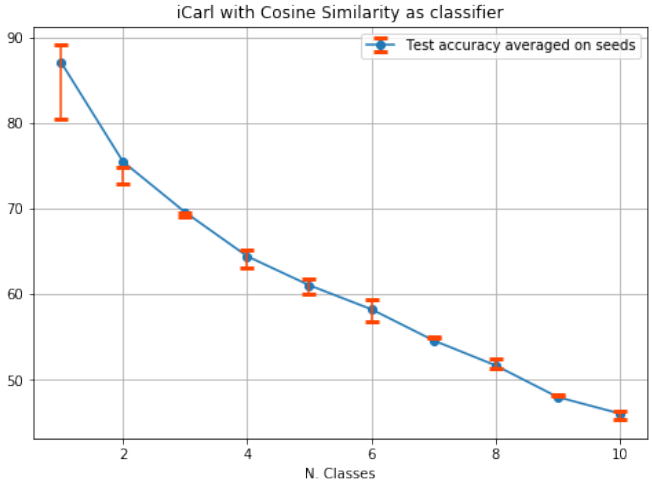

Figure 5. Results averaged on different seeds (the error bar represents the difference of the mean from maximal and minimal scores reached on a batch) using Cosine Similarity instead NME classifier.

We differentiates between the iCarl implementation also in the application of the classification loss on all outputs classes, not only on that of the current batch. The presence of exemplars in fact allows also to use old labels during the training, and when using Cross entropy as classification loss this reveals bettering the results. More specifically, for a generic input data x:

$$Loss_{class} = -\sum_{i=1}^{n+10} \delta_{y=i} \log[p_i(x)]$$

where y is the ground truth label of the input, delta is the indicator function and p is the softmax of logits of the i-th outputs.

A difference between CE and BCE is in the use of sigmoid activation instead softmax. In that case the loss computed for every CNN output vector component is not affected by other component values. So in principle it could be useful for multi-label classification, where the insight of an element belonging to a certain class should not influence the decision for another class. Nevertheless, we find in general - with our implementation - ourselves more comfortable using CE as classification part, regarding tuning and subjecting it to modifications, so regarding effective-

ness and stability in conclusion. We modified in fact also the basic setting - in which probably is really BCE the best choice - for example increasing the initial LR (to 2-04), or rather subjecting the network to a stronger regularization. On the other hand we found BCE very efficacious in distilling.

Moreover we choose also the smooth L1 loss because it is able to take the best features both from L1 loss and L2 loss. Creates a criterion that uses a squared term if the absolute element-wise error falls below 1 and an L1 term otherwise. It gives in general stability to the training phase. This loss is used for the distillation part, so we pass to it old classes logits of new and old network, while for the classification is used always the the CE. This combinations works good and the model is able to achieve almost the same performance of our basic iCarl implementations. In Fig. 6 is shown the best trial version that has a mean accuracy of 62.68.

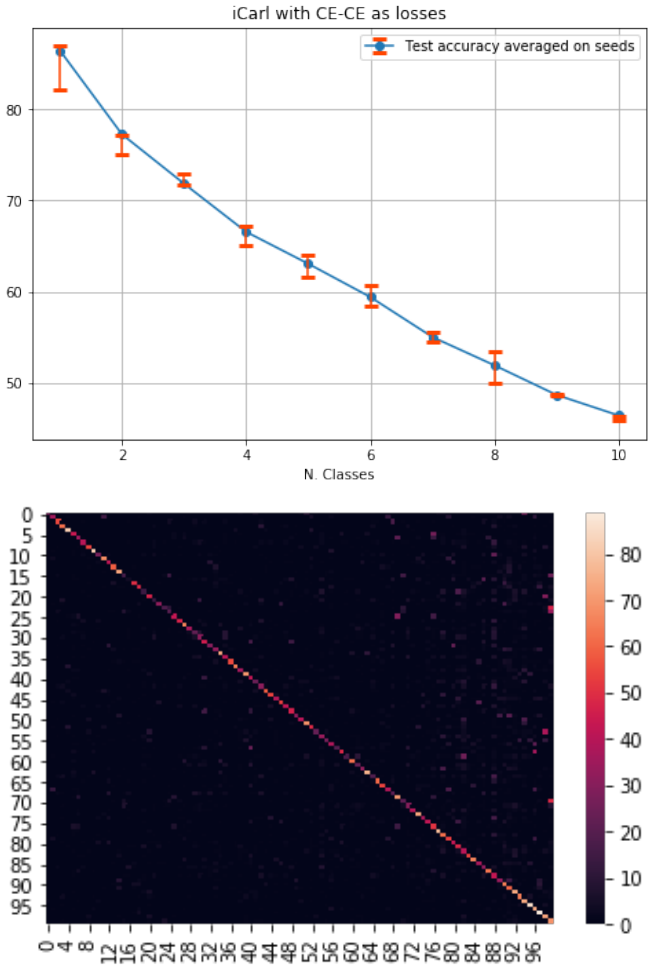

Figure 6. Results averaged on different seeds using CE-CE as losses. Also the confusion matrix (referred only on seed 1993) shown quite goods capacity in to learn new classes and remember old ones.

3. Our proposal

The problem of catastrophic forgetting is related to a limited capacity of the network to learn parameters for new task without forget that for old ones. LwF alleviates this problem with the distillation loss, able to maintain information of the old models through reproducing their outputs; iCarl with exemplars enhances that "remembering" process endows new tasks with little samples of previous data during the training. The latter model, besides to be meaningful over-performing, let us to think about the problem from the statistical points of view of balancing and re-sampling.

The underlying problem is a bias of the classifier towards new classes: a possible way to solve it is to find the most representative samples to allocate in a limited memory and to re-weight the classification process considering the total samples imbalance in each task.

In other words, the role of exemplars in every incremental steps make every single task compose of two sub-task: the first is the original classification task, that due to the minority presence of exemplars could be seen as an imbalanced classification task, while the second is the exemplars selection, that instead can be seen as an optimal undersampling task.

Our proposal consists to deal with both these problems from these point of view adopting new ideas and solutions.

3.1. Imbalanced Classification approach

We tried to improve the iCarl method with a different approach, the idea is to think at this problem like an imbalanced dataset. Indeed due to the limitations of the memory, at each batch, we have 500 samples for the current classes and a less number for the previous classes depending on the seen classes.

Possible existing approach to deal with this issue are re-sampling methods during the training, like under-sampling the majority class - risking to exclude important images - or over-sampling the minority class - repeating the same images from them risking to overfit the model. One can easily do that with a provided "sampler" argument in the Dataloader function of Pytorch: we tried but results are not particular interesting due to explained drawbacks. Another approach is to use weight for classes in the loss. The basic method consists to set these weights inversely proportional to the square root of class frequency. As explained in the paper "Class-Balanced Loss Based on Effective Number of Samples" [reference], this is not the best way to construct a class-balanced loss. During the training there could be information overlapping among data of a class, so higher is the number of samples of this class lower is the marginal benefit a model can extract from the data. For example this information overlapped in CNN could be due to data augmentation.

Ablation trial	Classification Loss	Distillation Loss	Initial LR	Milestones	Gamma	Weight decay	Average accuracy
1	CE	CE	0.2	[40, 60]	0.2	2e-04	**62.68**
2	CE	BCE	0.2	[40, 60]	0.2	2e-04	62.63
3	CE	Smooth L1 Loss	0.2	[40, 60]	0.2	2e-04	62.16

Table 3. Hyperparameters setting table for the ablation study trials (the average accuracy is referred to the mean on the seeds).

One can take it into consideration using an "effective number of samples" instead of empirical class frequency. The effective number can be imagined as the actual volume that will be covered by n samples into a volume N made of all possible data in the class. This is the formulation to calculate an effective number E, for a generic class y with n samples:

$$E_{n,y} = (1 - \beta^n)/(1 - \beta),$$

where

$$\beta = (N - 1)/N$$

Beta is a number between zero and one, while N is an hyperparameter that we consider equal for all the classes in the dataset. This modeling could be explained through its asymptotic properties: when N is 1 (beta=0) actually there is no need to weight because all images for each class represents the same information, while when N is very large (beta=1) the probability of overlapping is reduced such that the effective number is exact n.

The other important part is composed of the **Focal Loss**. This kind of loss fits very well the imbalanced problems and it's implemented as an extension of the binary cross-entropy loss, here reported in its classification version:

$$Loss_{FOCAL} = -\sum_{i=n}^{10} (1 - s_i(x))^{\alpha} \log[s_i(x)]$$

where s is the sigmoid function of the i-th output of the network for an input x, while (1-s) powered to alpha is a modulating factor. Note that for simplicity, using the same notations as cross-entropy, we have re-defined these outputs as:

$$o^i(x) = \begin{cases} o^i(x) \text{if} & i = y \\ -o^i(x) & \text{otherwise} \end{cases}$$

The focal loss down-weight numerous samples identified as easy samples and focus the train on the minority samples.Focal loss dynamically scales the binary cross-entropy loss in this case, where the scaling factor automatically decays to 0 as the confidence in the correct class increases. Increasing s to 1, the modulating factor decrease to 0 and the losses of the well-classified examples are down-weighted. Finally the definition of Class-Balanced Loss, that is weighted with the inverse of the effective number of samples:

$$CB(s, y) = (1/E_{n,y}) * Loss_{FOCAL}$$

Regarding the implementations we tried two version. In the first one we used the Class balanced loss both for the distillation and classification loss and in the latter we only used it as a classification loss. (Actually we try also to use normal BCE instead focal, but we don't report results because we don't see improvements, besides considering focal more interesting for this application). The first one is the best and gives a mean accuracy of 0.4357 on the last batch. The balancing used as in the canonical case instead gives a mean accuracy of 0.39 on the last batch. Regarding the total behaviour of the accuracy, in both the two implementations, it starts in line with the iCarl results but on the last batches there is an higher variance.

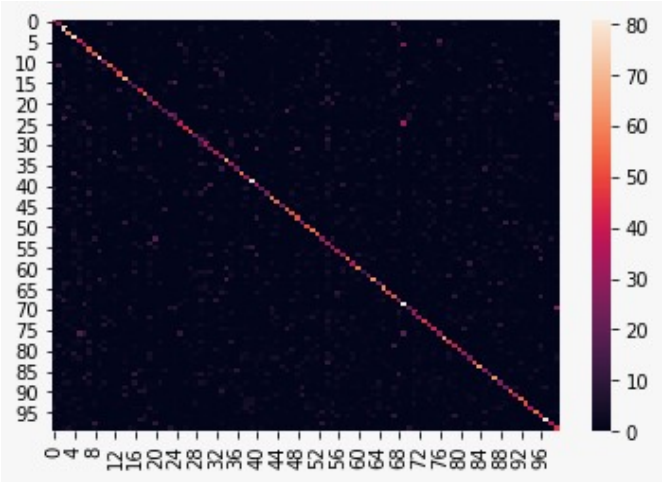

Figure 7. Confusion matrix for the experiment using CB+CB as losses.

Concerning the method without distillation, we expected to see that the balancing could have repair a bit forgetting, even without reaching other methods that make distilling. In fact our results sustain that to regularize the network saving the information of old outputs as distillation does prevents forgetting better respect to regularize only using a sophisticated weighting between old and new data in the loss. To distill is a strategy more directly connected to the incremental learning problem, so more able to look for balance in term of plasticity and stability for the network.

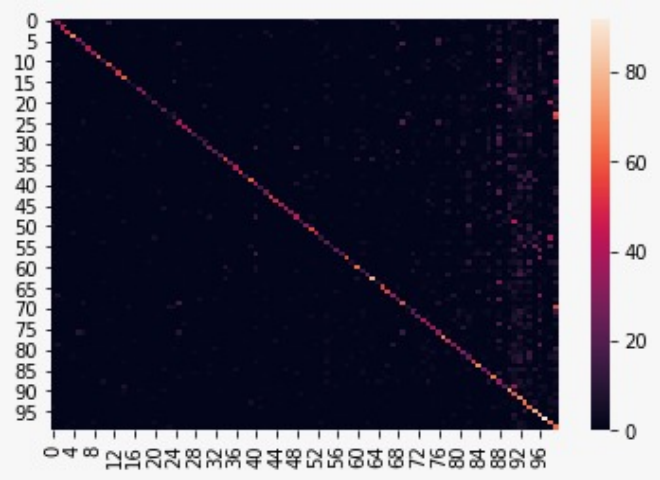

Figure 8. Confusion matrix for the experiment using only CB as classification loss.

Despite the expectations, also the second approach - the one using class balancing loss for both distillation and classification - does not improve results obtained with other methods. There are many possible aims to justify this behaviour. The first problem is imputable to the hyperparameters, indeed this loss has two hyperparameters that could depend on the data distribution, thus in the incremental learning tack where the data set is dynamic is very difficult to find a right set of hyperparameters. Moreover, mixing the balancing loss with the distillation equals to mix two different regularization strategy that are hard to balance, so this hardness traduces itself into a additional difficulty in the model selection.

In conclusion, despite the performance is a bit lower than that of other experiments, especially this last discussed method perform quite good a resistance to catastrophic forgetting: in **Fig.7** is shown a good distribution in this sense of points for the confusion matrix, while in **Fig. 8** there is the version without distillation that is visibly worse at remembering old classes, due to a more concentrated distribution of light on the right.

3.1.1 Implementation details

For the first batch, the scale factor alfa of the focal loss is set to 0.6 and the beta number for the calculus of the effective number of samples is set to 0.9999. For the rest of the batches, alfa is set to 0.3 and beta to 0.999999. We tried also to implement a schedelur function that make alfa depending on the number of exemplars for class at each batch, but the fixed value of 0.3 results to work better. Regarding learning rate, weight decay, optimizer and the other hyperparameters, we used the same used for the basic version of iCarl. (Results at Table 4)

3.2. Alternatives exemplars selection methods

Figure 9. Representation of exemplars selection process with iCarl herding on class 2.

We try to improve iCarl's method of exemplars selection. In the iCarl's paper is shown that random sampling achieves similar results compared to iCarl herding. So we try to analyze how in the features space samples from old classes are distrubuted, in order to establish if actually better exemplars to select exists, if there are better methods to select them, so indeed how much an accurate exemplars selection can positively affect performance of an incremental classification task.

3.2.1 SVM

As first trial we introduce an SVM implemented on the features representation of all training samples from classes of the current batch, after the training of the network. The advantage to use an SVM classifier is that to train the model means to use a subspace of the training set (made by the also known as support vectors) to compute hyperplanes that separates regions in a way useful for classification.

In our case, exemplars may be seen as subspaces from each batch of classes particularly good for maintaining a minimal degradation for classification performance over the incremental steps. So our aim is to select exemplars as extracted support vectors from each batch of classes, with the purpose to observe a better exemplars supported, so a more stable, classification accuracy on the test set (computed with an eventually analogous SVClassifier).

The main problem we encounter with this approach is the tuning of the used SVM. Since for different incremental steps we have to select different numbers of exemplars, hyperparameters for SVM we are interested in are not always the same. In particular for what concern the number of support vector used - so the number of exemplars we want to extract - is crucial the value of C. Generally highest is the C, lower is the number of support vectors used by the algorithm, because the increasing misclassification penalty represented by C induces to permit fewer support vectors

Method	Alfa	Beta	Initial LR	Milestones	Gamma	Weight decay	Average accuracy
CB+CB	0.3	0.999999	2	[49, 63]	0.2	1e-05	57.96
CB	0.3	0.99999	2	[49, 63]	0.2	1e-05	56.91

Table 4. Hyperparameters setting (alfa and beta are referred to those from the second batch) table for our Imbalanced Classification approach (the average accuracy is referred to the mean on the seeds).

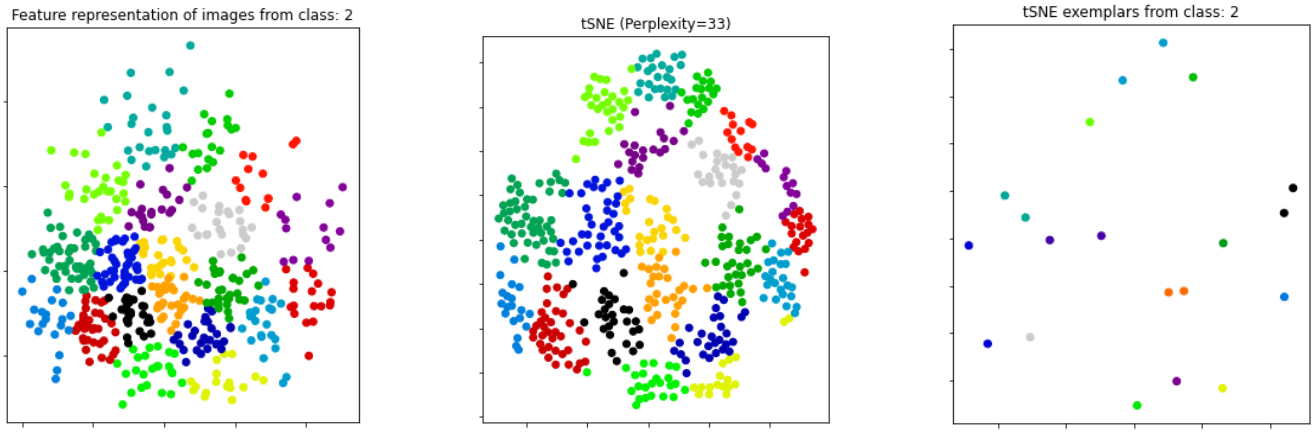

Figure 10. Visualization of exemplars selection process with K-Means on class 2 (22 clusters).

to stay in the margin. On the other hand such a solution increase the risk of overfitting, since an increasing misclassification penalty tends to over specialize the model on the training samples; so this could means to lose the original idea of using support vectors as best classification performing samples. Similarly, losing this idea is also the solution in which the right number of exemplars is selected independently from the number of support vectors used, i.e. without tuning the SVM for every batch. In conclusion we did't find a good balance allowing us to see improvements in comparison to iCarl herding or random selection, despite the initial idea to "remember" the most difficult samples for classification (that is another way to see support vectors) seems us to be winning for the task to maintain good results when the classes increase. At posteriori, actually, we realize that even theoretically this is criticizable: to record only data that are inside the margin could make on the contrary more challenging to not forget. Specifically we use same hyperparameters of the best ablation study, fixing C in a range of [1, 10, 100, 1000], or increasing it over steps always in this range. The mean accuracy reached on the last step is approximately 0.41 - with an average on the 10 batches of 0.59 - in the first case, with a little bettering in the second case. Anyway we don't observe meaningful behaviour for our analysis so do not report detailed results.

3.2.2 Clustering

As second trial we switch to a different approach. Starting from the hypotesis that every class could contain a proper

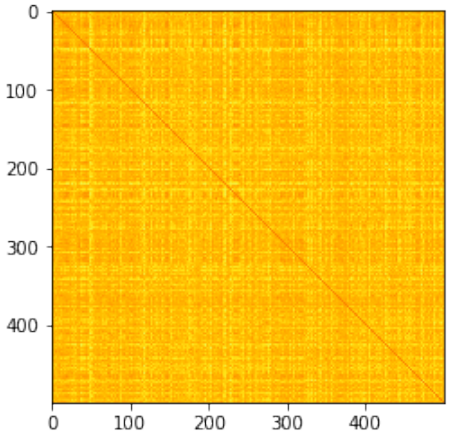

Figure 11. Affinity matrix obtained with spectral embedding for class 2 (22 clusters). Clusters are visually not well defined, however some points are more related than others.

own variability, like sub-classes, and so could be helpful to extract representative images of each sub-class, for each class, in order to maintain this relative variability even when samples from old classes are significatively reducted, we implement a clustering-based method for exemplars selection.

The first advantage with respect to an SVM based method is that in this way the number of exemplars is directly scalable. The core idea is that the most representative samples of a particular cluster in a class are that nearest to the cluster centroid. So we apply **K-means** on the features

representation of each class of the current batch after training the net, imposing to look to a number of clusters - so a number of centroids - accordingly to the number of exemplars that is possible to store for that specific batch.

Actually there are different version we experimented: setting the number of cluster costant (near to 20) for all batch of classes, and reducing adequately the number of samples nearest to the controids to save with the incremental steps; or changing directly the number of clusters at each batch step and maintaing the number of selected exemplars from every cluster fixed at one. In particular the second solution reveal itself better and more stable, slightly outperforming (round the +1 percent) our implementation of iCarl. It is noticeable the fact that with the latter, when the number of images to store for each class is big (e.g. 200 with the first batch) the K-means algorithm has to discriminates between a lot of clusters for every features-represented class. This is clearly not totally consistent with a class set size of 500 images, nevertheless it allows to perform an homogeneous sampling from this set and that is in line with the purpose of our selection method.

Anyway, it is possible to observe that when the numbers of clusters decrease clusters-like gruops really emerges in the features space of a class: for this visualization task we implement tSNE (after applying a PCA reduction to 2D to the features).

Figs. 9 and 10 visualizes the (normalized) features representation of the same random class, and the exemplars extraction with classical method of iCarl (herding) and our K-Means variation. In Figure 10 in particular there is a tSNE representation of 22 clusters with perplexity equal to 33. In terms of exemplars selected, it is hard to denote remarkable differences. Nevertheless, results (reported in detail in the Appendix) support the hypothesis that clustering based method is able to maintain a major relative diversification in the memory for old classes, allowing to achieve a slightly better result. In fact the best implementation of our proposal obtain a mean accuracy (reported also in **Table 5**) over classes of 63.73, versus 62.68 (**Table 3**) of our best implementation of iCarl. In addition we observe that this variation seem to reach notable more stable results over different runnigs trials in comparison to the basic one.

As one further trial we experiment an exemplars recomputation (instead elimination of the exceeding ones) for old classes, passing to our exemplars selection function also their reduced sets: this trial degrades a little the performance, probably as consequence of a more difficult effectiveness of clustering with reduced number of samples.

Herding is based also on an ordered list of exemplars useful when they have to be removed; in our case what works better is to not prioritize them (a trial was made ordering the list with nearness to an unique central centroid computed for each class) maybe because the advantage in diversification instead of the centralization.

Moreover we attempt to use **Spectral clustering** (lightly modifying its sklearn function in order to obtain the clusters centroids once applied the version of the algorithm that use K-means after the spectral embedding) instead K-means. Generally it is useful when the structure of the individual clusters is highly non-convex, but this is in effect not the case, in fact it does not outperforms K-means: with spectral clustering the mean accuracy on the last step is approximately 0.456, with an average over all steps of 0.61, so actually there is no improvement. Nevertheless it is interesting to observe the affinity matrix (**Fig. 11**) computed for a single class with this method, that is an alternative way to visualize similarities between images in the generic class.

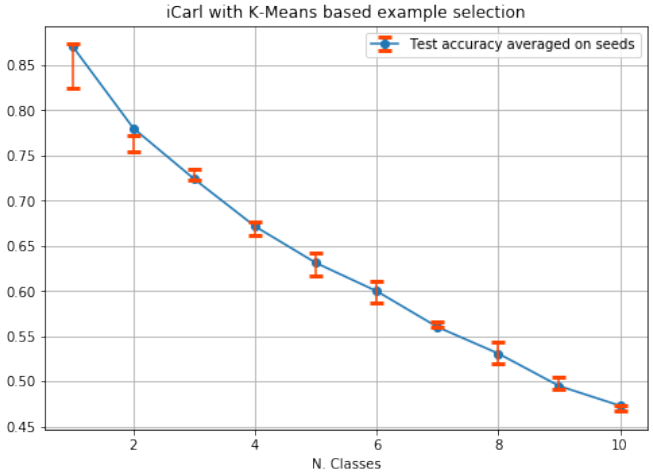

Figure 12. Results averaged using CE-CE as losses with K-means strategy.

3.2.3 Implementation details and final results

So the **best** version confirmed to be the one described above with **K-means** and a variable number of centroids. We experiment this clustering selection variation with two combination of losses, the best we prove in Section 2.2, so CE-CE (**Fig.12**) and CE-BCE. Concerning the classifier we used also Cosine Similarity - that in Section 2.1 demonsted to achieve comparable results to NME. Regarding the initial LR, we find that for our variation works optimally 0.1, instead of 0.2 that is find to be optimal for the ablation studies cases section.

Regarding the CE-CE we add another "trick" on the computation of losses, consisting in multiplying them with a lambda factor in this way:

$$Loss = \lambda Loss_{distill} + (1 - \lambda) Loss_{class},$$

with $\lambda = n/100$.

Lambda is 0 for the first batch since all classes are new, while in the last batches it is nearly 1, indicating the importance to maintain the old classes.

Class+Dist Loss	Init LR	Steps	γ	W.D.	Av. accuracy
CE+CE	0.1	[40, 60]	0.2	2e-04	**63.73**
CE+BCE	0.1	[40, 60]	0.2	2e-04	63.35

Table 5. Hyperparameters setting table for our clustering based selection method (the average accuracy is referred to the mean on the seeds with NME).

3.3. Additional trials

Initially we tried to face the problem of bias toward new classes adding after the FC layer a bias correction layer (take inspiration from the paper Large Sale Incremental Learning) estimating two parameters for a linear correction with a small balanced validation set of old and new classes. Despite it seems to be a simple and effective solution, we finally preferred a setting that avoid complications due to presence of validation sets and that does not use the FC layer as classifier, since we have found other classifiers on features space preferable.

References

https://arxiv.org/abs/1606.09282 "LwF"

https://arxiv.org/abs/1611.07725 "iCarl"

https://arxiv.org/pdf/1512.03385 "ResNet"

https://arxiv.org/abs/1905.13260 "BiC"

https://github.com/sairin1202/BIC

https://arxiv.org/pdf/1708.02002.pdf "Focal Loss for Dense Object Detection"

https://arxiv.org/abs/1901.05555 "Class-Balanced Loss Based on Effective Number of Samples"

Here are available our codes:

https://github.com/pierociffo/ILProject-MLDL

A. Appendix: Tables of detailed results

A.1. BASELINES

10	20	30	40	50	60	70	80	90	100
8.11	5.17	46.2	36.02	34.62	28.13	27.65	24.52	23.02	21.29

Table 6. Results for each batch achieved with Lwf.

A.2. ABLATION STUDIES

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	89.1	72.55	66.13	56.88	54.26	50.58	48.9	42.8	40.72	38.28	56.02
2204	80.4	75.1	67.47	59.92	56.56	51.97	48.13	44.73	38.63	35.19	55.81
1345	91.9	78.05	66.1	60.92	55.28	51.73	46.89	42.66	38.13	38.07	56.97
Mean on seeds	87.13	75.23	66.57	59.24	55.37	51.43	47.97	43.4	39.16	37.18	56.27

Table 7. Results for each seed achieved with SVM as classifier in iCarl.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	89.1	72.85	68.97	63.05	59.96	56.78	55.01	51.3	48.2	46.27	61.15
2204	80.4	74.8	69.53	65.22	61.74	59.38	54.91	52.49	48.07	45.32	61.19
1345	91.9	78.8	70.23	65.0	61.4	58.52	53.84	51.2	47.62	46.52	62.5
Mean on seeds	87.13	75.48	69.58	64.42	61.03	58.23	54.59	51.66	47.96	46.04	61.61

Table 8. Results for each seed achieved with Cosine Similarity as classifier in iCarl.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	89.1	72.95	66.43	57.95	53.86	48.93	47.76	42.68	40.14	38.05	55.79
2204	80.4	74.3	67.83	59.85	56.74	51.93	48.07	44.85	39.19	35.59	55.88
1345	91.9	77.6	66.67	61.45	55.26	51.15	45.64	42.83	38.46	37.48	56.84
Mean on seeds	87.13	74.95	66.98	59.75	55.29	50.67	47.16	43.45	39.26	37.04	56.17

Table 9. Results for each seed achieved with KNN as classifier in iCarl.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	86.9	75.15	71.63	65.33	61.6	58.35	54.69	50.2	47.8	45.45	61.71
2204	82.2	77.3	72.97	67.1	64.28	60.6	55.57	53.31	48.66	45.77	62.78
1345	90.2	79.45	70.83	67.3	63.68	59.23	55.03	52.31	49.1	46.97	63.41
Mean on seeds	86.43	77.3	71.81	66.58	63.19	59.39	55.1	51.94	48.52	46.06	62.63

Table 10. Results for each seed achieved using CE+BCE as losses.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	88.1	76.6	72.23	64.6	60.06	55.63	53.01	48.84	46.48	45.11	61.07
2204	81.6	78.15	71.83	67.25	62.88	59.42	55.19	52.71	48.08	45.97	62.31
1345	91.1	83.6	71.9	66.38	62.44	58.35	53.91	50.68	47.4	45.44	63.12
Mean on seeds	86.93	79.45	71.99	66.08	61.79	57.8	54.04	50.74	47.32	45.51	62.16

Table 11. Results for each seed achieved using CE+SmoothLoss as losses.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	86.9	75.1	71.77	65.12	61.56	58.42	54.57	50.06	48.79	46.45	61.87
2204	82.2	77.2	72.93	67.17	64.1	60.68	55.59	53.47	48.63	45.9	62.79
1345	90.2	79.45	70.83	67.42	63.58	59.2	55.0	52.33	48.72	47.05	63.38
Mean on seeds	86.43	77.25	71.84	66.57	63.08	59.43	55.05	51.95	48.71	46.47	62.68

Table 12. Results for each seed achieved using CE+CE as losses.

A.3.1 OUR PROPOSAL: CLASS BALANCING

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	86.7	70.9	65.87	59.82	55.8	51.43	50.71	47.51	44.3	42.36	57.54
2204	80.0	74.15	65.63	60.62	57.28	55.18	51.03	48.04	46.41	43.23	58.16
1345	90.1	76.5	65.03	60.0	55.48	52.2	48.93	46.95	44.5	42.08	58.18
Mean on seeds	85.6	73.85	65.51	60.15	56.19	52.94	50.22	47.5	45.07	42.56	57.96

Table 13. Results for each seed using the class balancing approach. The class balancing loss is used for both distillation and classification.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	86.0	72.55	67.3	58.43	55.48	50.1	48.23	42.5	39.44	37.67	55.77
2204	80.7	75.9	69.57	62.12	57.86	54.7	48.11	45.25	40.16	38.79	57.32
1345	90.0	80.3	68.53	62.15	56.52	51.72	47.4	42.41	39.33	38.03	57.64
Mean on seeds	85.57	76.25	68.47	60.9	56.62	52.17	47.91	43.39	39.64	38.16	56.91

Table 14. Results for each seed using the class balancing approach. The class balancing loss is used only for classification and there is no distillation.

A.3.2 OUR PROPOSAL: CLUSTERING

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	87.4	75.8	72.87	67.22	63.34	59.12	56.21	51.79	49.74	46.84	63.03
2204	80.5	78.45	73.0	68.65	64.46	61.42	56.8	54.51	50.46	47.86	63.61
1345	91.0	82.55	72.43	68.8	64.72	60.08	56.19	52.95	49.29	47.51	64.55
Mean on seeds	86.3	78.93	72.77	68.23	64.17	60.21	56.4	53.08	49.83	47.4	**63.73**

Table 15. Results for each seed achieved using CE+CE as losses and clustering for exemplars selection.

Seed	10	20	30	40	50	60	70	80	90	100	Mean on batches
1993	87.3	75.45	72.23	66.1	61.72	58.62	56.04	51.91	49.12	46.7	62.52
2204	82.5	77.15	73.47	67.62	64.28	61.12	56.56	54.27	50.53	47.3	63.48
1345	91.4	81.4	71.43	67.7	63.3	60.18	55.4	53.02	48.78	47.9	64.05
Mean on seeds	87.07	78.0	72.38	67.14	63.1	59.97	56.0	53.07	49.48	47.3	63.35

Table 16. Results for each seed achieved using CE+BCE as losses and clustering for exemplars selection.