Incremental Learning via Double Distillation

Giulia Tortoioli
giulia.tortoioli@studenti.polito.it

Giovanni Cioffi
giovanni.cioffi@studenti.polito.it

Fabio Frattin
fabio.frattin@studenti.polito.it

Abstract

This work deals with many different approaches to the Incremental Learning setting. First, the problem of catastrophic forgetting is introduced, and some existing attempts to tackle the problem, such as LwF and iCaRL, are presented. Then, some ablation studies in terms of losses and classification methods in the iCaRL framework are conducted, either making small changes or trying different approaches proposed in the literature. Finally, we propose our variation, with the aim of building a less biased classifier through the concept of Double Distillation, entirely turning the learning phase into the distillation from two different models, one holding the knowledge on the old classes and the other on the new ones.

1. Introduction

1.1. Incremental Learning

Thinking about how humans learn, an incremental setting is the closest to the reality. We are constantly learning concepts coming across new information every day, and we usually do not completely forget what we already know. On the other hand, our learning abilities are limited and so is our memory, thus if something is not reviewed periodically, we may not remember what we once knew.

Trying to extend this idea to the world of the CNNs, the implementation is not trivial. In fact, most of the training methods rely on a batch setting, meaning that all the classes and the related data are accessible in advance. When trying to learn something new in a further moment in time, if no specific measure is taken and previous data are not available anymore, the model almost immediately and inevitably loses the past knowledge to re-set on the new task. This is known as *catastrophic forgetting*.

More specifically, the most accepted framework for incremental learning requires that:

- classes of data come in different batches over time;
- the performances on old and new tasks are comparable;
- only a small and fixed amount (in some cases none) of previous data can be kept in memory;
- the amount of computational power and memory required can grow with the number of classes, but should remain bounded;

Later sections of the paper will show different existing techniques to face this issue.

1.2. Dataset

All the experiments have been performed on the CIFAR-100 dataset. It contains data of 100 different classes, which are split at the beginning of the analysis into 10 batches of 10 classes each. At each training step, a new batch is loaded as training data and the model is required to learn these new classes without forgetting the previously learned ones.

All the results here reported relate to our specific splitting of the classes, which is kept fixed along all the different methods to produce comparable results.

1.3. Implementation details

Where not differently specified, we use a 32-layer ResNet. It can be seen as composed of two parts: a feature extractor f followed by a fc-layer, that can be direcly used as a classifier. Being our studies focused on iCaRL and its variations, we stick to their implementation parameters and optimization methods: the number of training epochs is set to 70; the learning rate starts at 2 and is multiplied by 0.2 after 49 and 63 epochs; the weight decay is set to 0.00001 and the momentum to 0.9; the optimization phase is carried out through backpropagation via stochastic gradient descent, with minibatches of 128 images.

These parameters are to be considered the default ones when nothing different is specified.

The results reported in terms of accuracy or other measures are obtained through an average among 4 runs of the algorithm and have been found to be quite stable.

For more implementation details, please rely on the code at https://github.com/frattinfabio/project_IL.

1.4. Notation

From this point on, we will use the following notation, unless differently specified:

- (x, y): training sample x belonging to class y.
- $\delta_y(x)$: discrete function being 1 if y is the class label of x, else 0;
- $f(x)$: feature representation extracted from x;
- θ^y: fc-layer weights for class y;
- $o_y^t(x)$: output of the network at time t for the class y, more specifically $\langle f(x), \theta^y \rangle$;
- Y^t: new set of classes coming at time t;
- K^t: set of classes known at time t;
- $g_y^t(x)$: sigmoid output of the network at time t for class y, more specifically

$$g_y^t(x) = \frac{1}{1 + exp(-o_y^t(x))} \tag{1}$$

- $s_y^t(x)$: softmax output of the network at time t for class y, more specifically

$$s_y^t(x) = \frac{exp(o_y^t(x))}{\sum_{y \in K^t} exp(o_y^t(x))} \tag{2}$$

2. Related works

2.1. Finetuning

Finetuning means learning without adopting any strategy to avoid catastrophic forgetting. The network is trained using just the new data available through a loss function which only cares about maximizing the class output for the true class label and minimizing all the others regardless. In this way the network almost immediately (after few epochs) completely forgets what it learned in the previous phase.

2.2. Learning without Forgetting

The idea of *Learning without Forgetting* is using a distillation loss in order to drive the network to remember past knowledge by adding a term which penalizes changes in old classes' outputs with respect to the ones obtained in the previous incremental step. The loss formulation used for this model will be explained in the next section (3).

2.3. iCaRL

A more complete implementation was introduced by Rebuffi *et al*. [5] with iCaRL. iCaRL added two main components to the previous model (LwF):

- *Nearest Mean of Exemplars* classifier;
- *prioritized exemplar selection* based on herding.

In order to preserve better the previous knowledge a fixed number K of exemplars ($K = 2000$ in our case) equally distributed among the past classes is trained with new classes data at each incremental step.

The classification task through NME consists in finding the label y^* for which the distance between the average feature vector of exemplars for class y and the feature vector of the image to be classified is the minimum.

Moreover the set of exemplars is chosen with *herding* strategy, or rather a greedy search to find a group of old samples of which the average feature vector well approximates the the average feature vector over all training samples for a specific class. Hence the exemplars are selected iteratively by making at each iteration the best "local" choice, selecting the exemplar whose contribution to the current exemplar set mean would best approximate the class mean. This way, a prioritized list is created. Reducing the exemplar set is made by removing elements with the lowest priority, thus the ones at the end of the list.

The loss function which is used in iCaRL (and also in our implementation of LwF) is a Binary Cross Entropy loss which includes two contributes: the first drives the network to a correct class indicator for new classes (L_{class}) while the second aims at preserving the outputs of the previous step (L_{dist}) (3).

$$L_{class}(x) = \sum_{y \in Y^t} \delta_y(x) \log g_y^t(x) + [1 - \delta_y(x)] \log[1 - g_y^t(x)]$$

$$L_{dist}(x) = \sum_{y \in K^{t-1}} g_y^{t-1}(x) \log g_y^t(x) + [1 - g_y^{t-1}(x)] \log[1 - g_y^t(x)]$$

$$L_{tot}(x) = \frac{1}{|Y^t|} L_{class}(x) + \frac{1}{|K^{t-1}|} L_{dist}(x) \tag{3}$$

After the implementation of iCaRL using *herding* for the exemplars selection, we tried to use a *random sampling* strategy which gave better results in terms of accuracy. For this reason also in the following models a random selection will be used since in most cases it allows to generalize better.

3. Ablation study: Losses

Starting from iCaRL framework with NME classifier, we experimented the use of different couples of losses, each one composed by a classification component and a distillation component.

The losses are computed and averaged for the current batch of images, then they are summed up using a weighting approach in order to obtain a balanced final loss with respect to the cardinality of new classes and old ones.

By minimizing the classification loss, we encourage the output of the network to match the ground truth labels (i.e. the targets) while classifying the new set of classes in each incremental step. The distillation loss, instead, is thought to preserve the knowledge about old classes in order to avoid catastrophic forgetting. It takes as targets the outputs of the net at the previous step whereas it takes as input the outputs of the net at the current state. A more detailed explanation of how and why distillation works is made in a later section (5.1).

In the following sections, we are going to discuss the implementation of three different couples of losses. The first two couples use cross-entropy loss for classification and respectively Hinton Loss and Kullback-Leibler divergence loss for distillation, whereas the last couple use MSE loss both classification and distillation tasks.

The learning rate is set to 0.1 when using cross-entropy loss and 5 for the MSE loss. It is divided by 10 at each milestone.

3.1. Cross Entropy for Classification

Cross-entropy $H(p, q)$ assumes the presence of two probability distributions over data, the true distribution p, which in our case is the empirical distribution of the ground truth labels, and the estimated distribution q of the predictions of the net, both defined over a discrete variable x, which represents the training images. The empirical distribution for each image simply assigns probability 1 to the class of that image, and 0 to all other classes. The formula of the cross entropy is the following:

$$H(p, q) = -\sum_{\forall x} p(x) \log q(x) \tag{4}$$

For our purposes of classification loss in incremental leaning, this loss consists in the product between the target one-hot vector and the logarithm of the softmax function applied to the output. It maps the output of the net to a probability distribution over the predicted output classes. Since in our case the training is carried out per batches of images, the loss is averaged by dividing the summation by the cardinality of images for that batch. For the sake of simplicity, the formula below (5) shows a general loss suitable for a single image.

$$L_{ce}(x) = -\sum_{y \in Y^t} \delta_y(x) \log s_y^t(x) \tag{5}$$

Since $\delta_y(x)$ takes value of 1 only for the ground truth label of the image x, for each new class y, only one term will be added up to the loss. When the net outputs low values for the true label, meaning that the prediction made by the current state of the network is likely to be wrong, the logarithm of the softmax results in a high contribution for the loss function that must be minimized. On the contrary, higher values of the net outputs will give lower contribution for the loss; with this procedure, the output of the net will be encouraged to be adherent to the target. The behaviour of the logarithm of a pseudo-probability is shown in the Figure below.

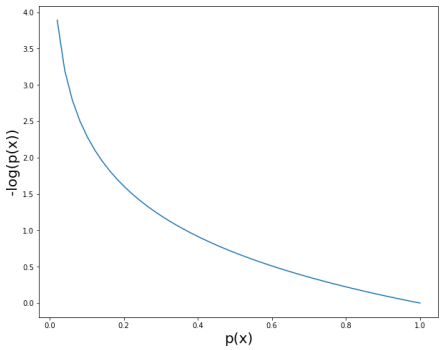


Figure 1: Behaviour of the negative logarithm of a probability

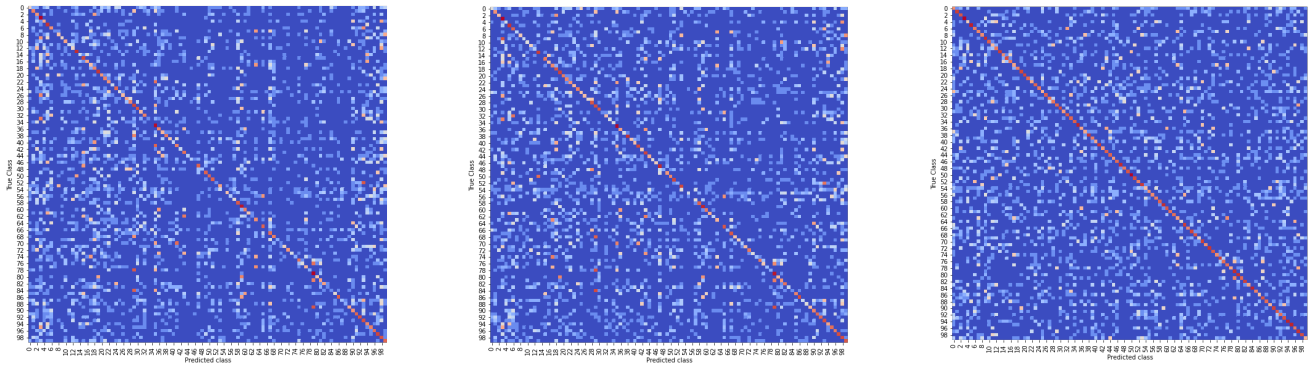
3.2. Hinton Loss

The first variation was done using cross-entropy loss as classification loss and another loss inspired by Hinton *et al.* [2] as distillation loss. This combination of losses is similar to the one used by Li and Hoiem [4].

$L_{hinton}(x)$ is the summation over the set of old classes K^{t-1} of the products between the target, which are the sigmoid outputs of the old classes obtained with the net at $t-1$ and the logarithm of the softmax applied to the output of the net at the current step t divided by the so called *softmax temperature* T, which is intended to increase the sensitivity to low probability candidates. The formula of the temperature based softmax function is the following:

$$s_y^t(x, T) = \frac{exp(o_y^t(x)/T)}{\sum_y exp(o_y^t(x)/T)} \tag{6}$$

Temperature is set to 2 in order to produce a softer probability distribution over classes, meaning that less values in the probability distribution are close to zero. $T = 1$, instead, would give back the original softmax.



(a) $L_{ce} + L_{KL}$ (b) $L_{ce} + L_{hinton}$ (c) $L_{MSE,class} + L_{MSE,dist}$

Figure 2: Confusion matrices on CIFAR-100 training groups of 10 classes at each incremental step with different loss pairs (classification and distillation loss). L_{ce} + L_{KL} divergence on the left, L_{ce} + L_{hinton} in the middle, $L_{MSE,class}$ + $L_{MSE,dist}$ on the right. From left to right, the pattern is increasingly homogeneous, meaning that no intrinsic bias towards specific classes is present

The formula of $L_{hinton}(x)$ is showed below:

$$L_{hinton}(x) = -\sum_{y \in K^{t-1}} s_y^{t-1}(x,T)\ \log s_y^t(x,T) \qquad (7)$$

Thus, Hinton loss is similar to a cross-entropy loss that acts on the previous class outputs preserving them by increasing the loss as they tend to change along the incremental process as an effect of catastrophic forgetting. After the computation of the mean of the losses for the batch, it is multiplied by T^2 as suggested by Hinton *et al.*, since he gradient of the loss with respect to the model weights scales as $\frac{1}{T^2}$.

3.3. Kullback Leibler divergence

The second variation for distillation loss was carried out using the Kullback-Leibler divergence.

It measures the "distance" between two probability distributions over the same variable x. For the distillation purpose, the *true* probability distribution p and q are respectively represented by the outputs of the net at the previous and the current step for the old classes to be preserved. There is a strong correlation between cross-entropy $H(p,q)$ and Kullback-Leibler divergence:

$$D_{KL}(p||q) = H(p,q) - H(p) \qquad (8)$$

The difference between them is the term $H(p)$, the entropy of the empirical distribution of the previous net outputs. Since p is fixed, the entropy does not change with the parameters of the model during the training process and can be disregarded in the loss function. Thus, Kullback-Leibler loss can be reconducted to a cross-entropy loss using as targets the softmax output of the network at the previous step.

$$L_{KL}(x) = \sum_{y \in K^{t-1}} s_y^{t-1}(x)\left[\log s_y^{t-1}(x) - \log s_y^t(x)\right] \qquad (9)$$

3.4. MSE Loss

Another experiment was carried out using MSE loss (Mean Squared Error) as both classification and distillation loss.

$L_{MSE,class}$ computes the mean square error between the output of the current net passed through the sigmoid function and the one-hot vector $\delta(x)$ for the new set of classes. $L_{MSE,dist}$, computes the mean squared error between the sigmoid outputs of the net at the previous step and the sigmoid output of the net at the current step. MSE classification and distillation loss formulas are shown below:

$$L_{MSE,class} = \frac{1}{|Y^t|}\sum_{y \in Y^t}(g_y^t(x) - \delta_y(x))^2 \qquad (10)$$

$$L_{MSE,dist} = \frac{1}{|K^{t-1}|}\sum_{y \in K^{t-1}}(g_y^t(x) - g_y^{t-1}(x))^2 \qquad (11)$$

We though about using MSE both for classification and distillation rather than combining them with another type of loss, such as cross-entropy, for their particular quadratic form, different form the linearitmic one of the other losses presented before.

3.5. Results

Figure 3 shows the overall accuracy score against the number of classes learned at that stage of the incremental process, which is the average of the performances on old and new classes with respect to the different distillation losses used. Models trained with L_{ce} as classification loss, combined with L_{hinton} and L_{KL} end up with lower overall accuracy with respect to iCarL, 31.92% with the former, 29.72% with the latter. The variation carried out with L_{MSE}, instead, reaches similar performances to iCarL with a final overall accuracy of 43.38% with respect to 44.26% of iCarL. The average overall accuracy is slightly

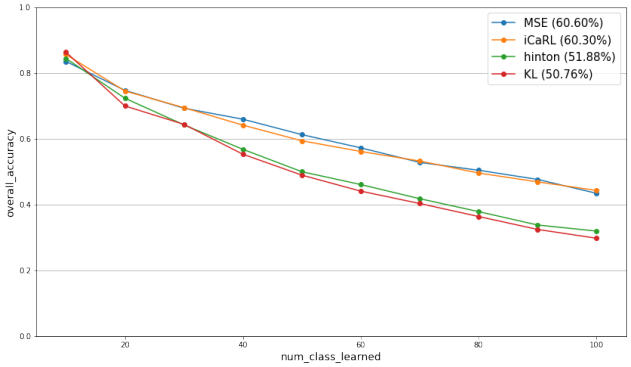


Figure 3: Overall accuracy loss vs number of class learnt depending on different couples of classification and distillation loss. Average of the overall accuracy at each step is shown in brackets in the legend.

higher for the model trained with L_{MSE}, though.

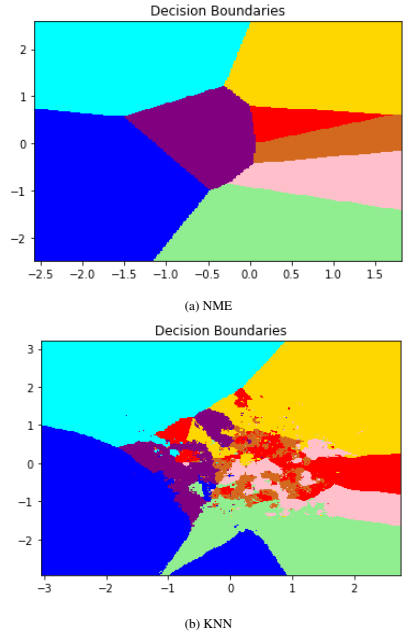
We plotted confusion matrices through heatmaps (Figure 2) for each of the three considered configurations of losses in order to obtain further insights about their behaviour. The diagonal entries represent the correct predictions, whereas the off-diagonal entries are the errors in classification. The model trained with L_{MSE} shows the most homogeneous pattern over classes, both in terms of correct predictions, represented by the pronounced red points on the principal diagonal line and in terms of errors. Thus, this configuration has no intrinsic bias towards specific classes between the 100 it encounters during the incremental process. On the contrary, the models trained with Hinton Loss or Kullback-Leibler divergence as distillation losses present similar appearance: they show a less homogeneous pattern which is slightly biased towards the first incremental groups and the last one with many non-zero entries concentrated on the left side and on the right with respect to the center. Moreover, they have a less pronounced red diagonal, meaning that they tend to make more errors.

4. Ablation study: Classifiers

After the evaluation on the losses, we tried different models for classification based on past works and on an accurate analysis of the task. We studied different approaches to address incremental learning for image classification and we applied some of them to our problem in order to see the strong points and even the limitations of each method.

4.1. K-Nearest Neighbor Classifier

NME classifier adopted by iCaRL uses a method based on the distances between images' and classes' mean feature representation. A classifier which has a similar behaviour since it uses distances for labelling data is the K-Nearest Neighbor (KNN) classifier. KNN algorithm finds the K



(a) NME

(b) KNN

Figure 4: Decision boundaries related to NME and KNN. For the sake of visualization, the number of classes has been reduced to 4 per batch and the image represent the situation at the second step, thus having seen 8 classes. A dimensionality reduction through t-SNE has been performed. It is clear that the boundaries obtained through NME are more prone to generalize, while the KNN tends to overfit more on the training data.

nearest (in terms of Euclidean distance) train datapoints to predict the label of the new sample either by majority voting or by weighting each "vote" on the distance from the sample.

Since KNN suffers from unbalanced classes distribution, we decided to resize new classes during the classification by randomly undersampling the data up to the number of old classes' exemplars. Note that this is in line with the random sampling approach to the exemplar selection. Regarding the numbers of neighbors, we observed that the classifier works better with values higher than 7, so we applied a local grid search on a set of values ([9,11,13,15]) every time a training phase ended.
In order to make KNN classifier more suitable for our task we also changed the default setting of classifier's choice of nearest neighbors. In fact, in our implementation the classifier assigns weights proportionally to the inverse of the distance from the query point. Adopting this approach, points nearer to the new one will influence more the prediction and it should be more unlikely that they belong to a different class. Our assumption was that a uniform weighting strategy could have suffered from the high number of classes and neighbors considered.

KNN did not reach the average accuracy results of NME in iCaRL, but on the other hand with KNN a higher balancing between new and old classes was reached. The better

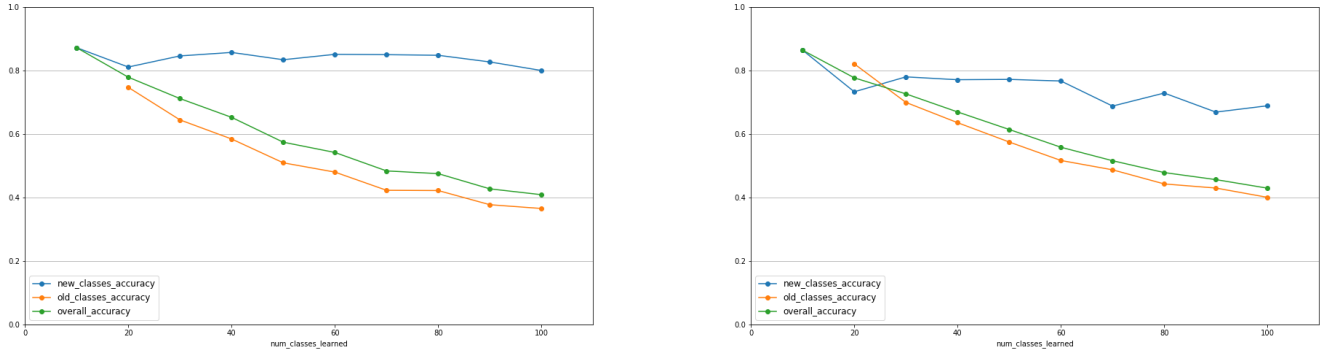


Figure 5: Accuracy scores after each incremental step using original IL2M (left) and IL2M with the correction factor α (right). It can be observed that the factor α allows more balanced predictions between new and old classes as well as a higher overall accuracy.

performance of the NME might be explained by the difficulty into well generalize for a distance based algorithm like KNN when working in a very high-dimensional space (each feature vector has 64 elements) with a lot of data, as shown in Figure 4.

4.2. Incremental Dual Memory Classifier

A totally different approach was proposed by Belouadah and Popescu [1] that introduced Incremental Learning with Dual Memory (IL2M). This method uses statistics about past classes computed when they were originally trained to rectify their prediction scores in the current incremental state. IL2M rectification is based on the idea that the classification is more accurate when it is made on all available data and that the more are the training data the higher will be the prediction scores of the classes.

Their idea of correcting the unbalance towards the new classes then shifts from a distillation approach: no distillation loss is used. The knowledge about past classes is preserved through the storing of the exemplars (first memory) and statistics computed for each class on the training phase during which they were originally learned (second memory). The rectification of the output scores of the network by means of those statistics is supposed to reduce the bias.

The method proposed in IL2M paper to compensate the existing bias between new and old classes scores is a rectification to be applied to past class predictions:

$$o_y^{t,rect}(x) = \begin{cases} o_y^t(x) \times \frac{\mu^{t^*}(y)}{\mu^t(y)} \times \frac{\mu(\mathcal{M}_t)}{\mu(\mathcal{M}_{t^*})}, & \text{if } pred = new \\ o_y^t(x), & \text{otherwise} \end{cases} \tag{12}$$

where: t^* represents the state in which y was learned and t is the current state, so μ^{t^*} and μ^t are the related mean prediction scores. $o_y^t(x)$ is the raw prediction for class y in the current state and the values $\mu(\mathcal{M}_{t^*})$ and $\mu(\mathcal{M}_t)$ correspond to the confidences in states t and t^*. Analysing the formula it can be noticed that the rectification is applied to past classes predictions only if the image is labelled as belonging to a new class.

This method was devised to avoid catastrophic forgetting for very large datasets and moreover the number of exemplars used by IL2M is higher in percentage with respect to our old samples (2000 in total). In fact trying to apply this model to our problem we observed that there was still a high imbalance between new classes and past classes scores. For this reason we tried to apply a modification to the rectification factor for new classes' outputs by multiplying it with a constant α:

$$o_y^{t,rect}(x) \leftarrow \alpha\, o_y^{t,rect}(x). \tag{13}$$

We used the Cross Entropy loss as described in (5) because it has the best performance in predicting new classes. The learning rate is set to 0.1 and it is divided by 10 at each training milestone. The value of α which worked the best for our purpose of balancing the prediction scores between old and new classes was $\alpha = 1.3$, as shown in Figure 6. Furthermore, also the overall accuracy increased thanks to the contribution of the additional factor.

4.3. Cosine Normalization Classifier

Another work which inspired us for its innovative approach was the one proposed by Hou *et al.* [3]. Their work introduced some new components after analysing the main critical issues related to incremental learning. Their attention was mainly focused on the bias towards new classes which affects the training process, leading to imbalanced magnitudes between weight vectors of new and old classes and to a deviation in the previous knowledge. The solution to these problems was the introduction of *cosine normalization* for balancing magnitudes and of *less-forget constraint* for preserving the geometric configuration of old classes.

The *cosine normalization* is applied the last layer of the CNN and is formulated as follows:

$$p_y(x) = \frac{\exp(\eta\langle\bar{\theta}_y, \bar{f}(x)\rangle)}{\sum_y \exp(\eta\langle\bar{\theta}_y \bar{f}(x)\rangle)}, \tag{14}$$

where $\bar{\theta}_y$ and $\bar{f}(x)$ are the l_2-normalized vector of the weights and the feature extractor respectively and the scalar product measures their cosine similarity. Since the inner product gives a result in the range [-1,1], a learnable factor η is introduced to adapt the values to the softmax function used in the learning phase.
Using this normalization the weight bias between new and old classes is eliminated.

The second objective is to preserve past knowledge, in particular the relationship between features and weights of old classes. *Less-forget constraint* has the function of preserving the angle between the normalized vectors and at the same time control the orientation of the features in order to avoid a rotation. The formulation of this constraint used as distillation loss on the features is the following:

$$L_{dis}(x) = 1 - \langle\bar{f}^*(x), \bar{f}(x)\rangle, \tag{15}$$

where $\bar{f}^*(x)$ and $\bar{f}(x)$ are the normalized features of the original model and those of the current one respectively. The objective of $L_{dis}(x)$ is to shape the orientation of the features extracted by the current network basing on the features of the original model.

To stick with the implementation provided by Hou *et al*., the number of epochs is increased to 160, the learning rate to 0.1 and divided by 10 after 80 and 120 epochs. Also the weighting of the losses is different then usual: the distillation loss importance decreases for each incremental step since the ratio between the number of old classes and new ones increases and there is less need to preserve the previous knowledge. For this reason an adaptive factor λ is applied to $L_{dis}(x)$:

$$\lambda = \lambda_{base}\sqrt{\frac{|Y^t|}{|K^{t-1}|}}, \tag{16}$$

Where λ_{base} is a constant to be tuned with respect to the specific implementation. The value that worked the best for us was λ_{base} = 5.

From this moment on and for every table or image, all the references to result obtained through *IL2M* or *Cosine normalization* will be based on our specific implementation and not on the achievements of their original authors.

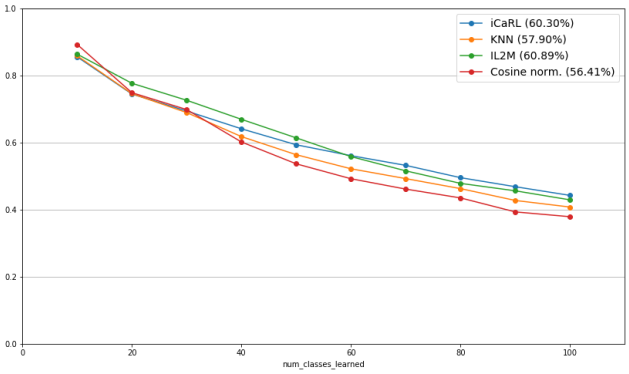


Figure 6: Overall accuracy values after each incremental step using iCarl with random selection and our implementations of KNN, IL2M and Cosine Normalization classifiers. Average accuracy among all incremental steps is indicated in brackets in the legend.

5. Our variation

One of the biggest limitations of the iCaRL approach is the imbalance towards the new classes, which are not only more represented in the training dataset, but also benefit from a stronger constraint due to the nature of the loss proposed.

Although it could make sense, from a human-learning point of view, remembering better the last thing you saw, it is not the way we would like a classifier to behave: ideally the relative performance on a specific task should be indipendent by the moment in the learner history such a task was learned.

Since we cannot have a great impact on the amount of training data (although some exemplar-generative approaches can be useful as described by Wu *et al*. [6]), the focus of our work is on the loss function and on a way to try to make it fairer with respect to the class imbalance.

The main idea is to make the learning about the new classes through a distillation technique, matching what already happens for the old classes, in order to treat them more equally.

5.1. Distilling Knowledge

Hinton *et al*. introduced the idea of distillation as a way to compress the knowledge of a group of different learners into a single, simpler model to have an easier deployment phase.

Usually, what is supposed to embed the knowledge about a task in a neural network are the learned parameter values. This point of view may be limiting assuming we want to compress the knowledge of two (or more) task-specific models, which then may have very different parameters, into one single model able to perform both tasks: if we change the parameters, the knowledge would be lost. To make the knowledge independent on the specific values of

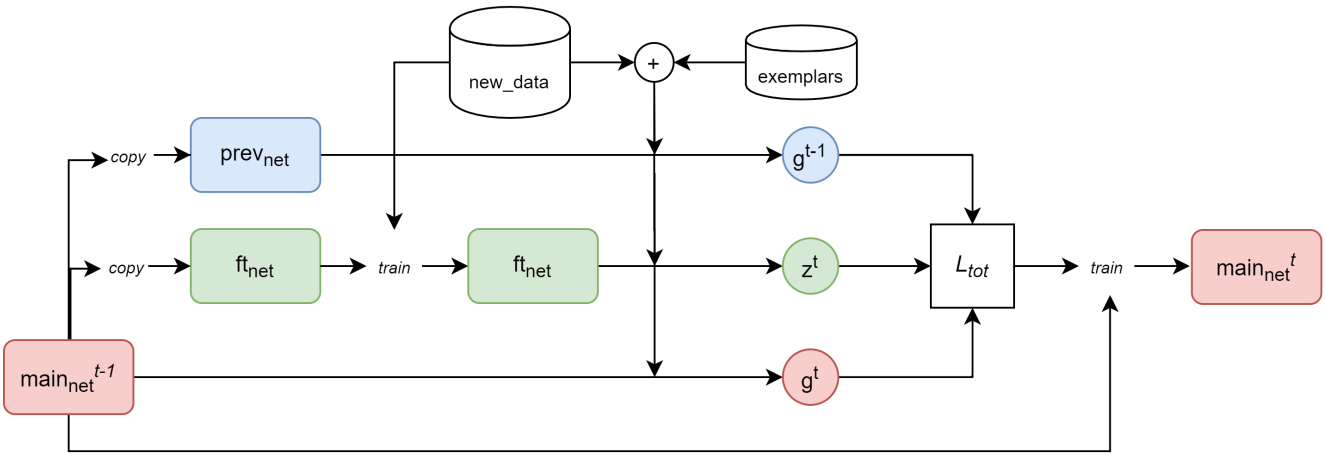


Figure 7: A qualitatve diagram of the functioning of a single incremental step. First, two copies of the network are created. One of them is *finetuned* using only the data from the new classes. When training the main network, both the exemplars and the new classes data are used. The sigmoid outputs of the other two networks are used to compute the loss through which the main network itself is updated.

the parameters, we have to shift to a more abstract interpretation: at the end, what is really relevant in terms of accuracy is the mapping function learned by the model. If we were able to reproduce the same (or at least similar) mapping behavior with a different set of parameters, the knowledge of the model would remain roughly the same: this is what the **distilling** approach tries to achieve.

The usual training is based on maximizing the probability of the correct answer and to minimize, at the same time, the probabilities for all the other answers, regardless how those answers may be related to the correct one. In other words, we try to learn an ideal distribution where the probability of the correct answer is 1 and 0 for all the others. Obviously, this is a big simplification because, for example, given an image of a dog, we would expect the probability of misclassifying it as a lion to be much higher than the one of misclassifying it as a car. On the other hand, this simplification is inevitable since normally we don't have the real distribution we want to tend to, and the best we can do is to optimize the performance on the training data.

Contrariwise, when we want to distill the knowledge of different models into a single one, we do know how a good representation of the real distribution looks like (assuming the task-specific models achieve good performances on their own). We can use the class probabilities produced by the task-specific models as **soft-targets** instead of the **hard-targets** previously mentioned (1 for the correct class, 0 for the others), and then minimize the distance between the targets and the current predicted probabilities, thus having a better generalization and extracting a broader knowledge.

5.2. Double Distillation

As previously mentioned, one of the reasons of the imbalance towards the new classes is the fact that while the distillation uses soft targets, the classification uses hard targets. To re-balance the situation, we would need something to extract the soft-targets for the classification from, a fairly good representation of the distribution among the new classes for the training data.

Since the targets for the old classes are taken from the network at the previous incremental step (which is supposed to perform well on the old classes), we would need a "complementary" network able to tell us how a good (but feasible) network would behave on the new tasks. In fact, one could think to the hard-targets as some sort of soft-targets based on a "ideal" model, able to perfectly classify each input image, whose knowledge we are trying to distill into our model. Nevertheless, such a model is not practically feasible.

In the first steps of our study, it was shown that the fine-tuning approach produces a model with high performances on the new tasks, while completely forgetting about the others. Although such a model is not what we are aiming at for the old tasks, its behavior on the new tasks is the best one we can achieve and it is something feasible, not ideal. In the same way we use the output of the network at the previous step as soft-targets for the old tasks, we could use the outputs of the fine-tuned network as soft-targets for the new tasks.

Before training the network on new classes, it is then required to save two copies of the old state of the network. One of them will be used, as in the iCaRL framework, to extract the soft-targets for the old classes; the other one will be trained via a fine-tuning approach using only the new training data: once trained, its outputs will be used as soft-targets for the new classes.

More specifically, the training phase is carried out like

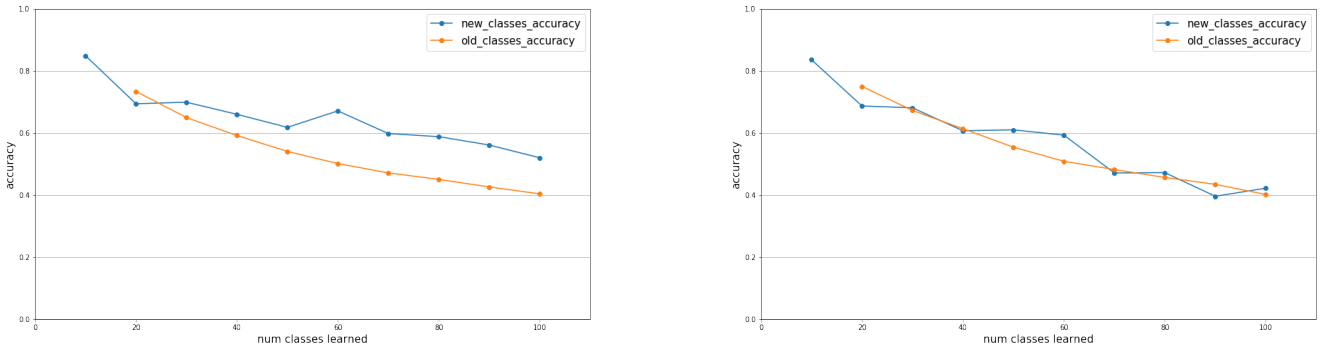


Figure 8: Comparison between accuracy on old and new classes in iCaRL (left, average overall accuracy 0.603) and Double Distillation (right, average overall accuracy 0.585).

this (Figure 7):

1. two copies of the current state of the network ($main_{net}$) are created, respectively $prev_{net}$ and ft_{net}.

2. ft_{net} is fine-tuned using only the new data (no exemplars) and minimizing a loss having the same shape of the classification term of the one described at (3), but taking into consideration all the classes and not only the new ones.

3. $main_{net}$ is trained using all the available data (new classes data and the exemplars) minimizing the L_{tot} loss, where z_y^t, g_y^{t-1} and g_y^t indicate the sigmoid outputs for the class y obtained respectively from ft_{net}, $prev_{net}$ and $main_{net}$:

$$L_{dist^{new}}(x) = \sum_{y \in Y^t} z_y^t(x) \log g_y^t(x) + [1 - z_y^t(x)] \log[1 - g_y^t(x)]$$

$$L_{dist^{old}}(x) = \sum_{y \in K^{t-1}} g_y^{t-1}(x) \log g_y^t(x) + [1 - g_y^{t-1}(x)] \log[1 - g_y^t(x)]$$

$$L_{tot}(x) = \frac{1}{|Y^t|} L_{dist^{new}}(x) + \frac{1}{|K^{t-1}|} L_{dist^{old}}(x) \quad (17)$$

At the end, what this approach is trying to do is to **distill** the knowledge of two different models, well performing on some specific tasks, into one single model able to accomplish both tasks with a comparable accuracy.

Note that both $main_{net}$ and ft_{net} are initialized with the weights of the model at the previous incremental step, thus being initially biased towards the old classes. This should help in tackling the training data imbalance.

5.3. Results

The results we obtained partly match the expected outcome described in the previous section.

From the accuracy point of view, the class imbalance basically disappeared. The hard-target approach led to an average difference of 10.2% between old and new classes accuracy, against a 2.2% achieved by the soft-target one (Figure 8). It is interesting to notice how, in both cases, when the number of exemplars is close to the number of new training data (like in the first incremental step where there are 500 images per classes for the new classes and 200 exemplars for the old ones) the model is better in remembering than learning.

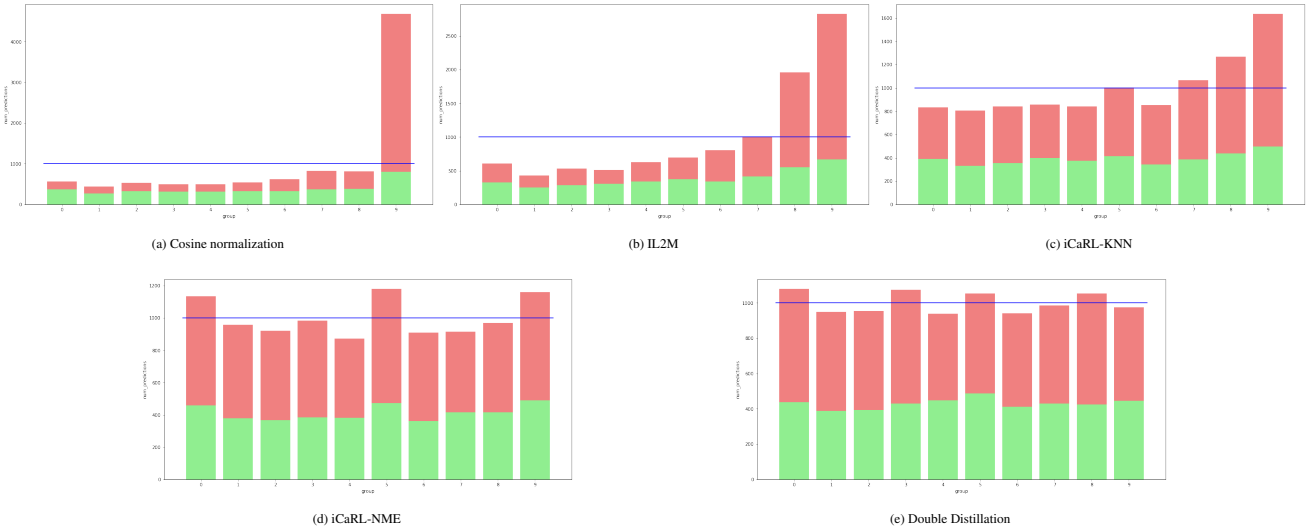
On the other hand, we expected to achieve this result through an increased accuracy on the old classes rather than a worse performance on the new ones, which is indeed what happens in practice: the light improvement on the old tasks is not enough to have the same level of overall accuracy.

To better evaluate the bias of the different models, they were performed both a quantitative and a qualitative analysis of the predictions at the final stage, when all the 100 class have been seen and learned.

Method	class MRE	group MRE
iCaRL-NME	22.45%	9.52%
iCaRL-KNN	30.12%	19.42%
Double Dist.	**19.60%**	**6.13%**
IL2M	37.20%	33.03%
Cosine norm.	46.76%	53.41%

Table 1: *Mean relative error* between the number of predictions and the acutal number of test images per class/group.

On the quantitative side, it was inspected only the difference between the number of predictions the model was supposed to make and the one it made. We see that the Double Distillation model is the one with the lowest mean differences both between the true number of test image per class and per group (Table 1). Even though it is only an evaluation from the quantitative point of view (all the predictions are counted, regardless of their matching with the truth), it



(a) Cosine normalization (b) IL2M (c) iCaRL-KNN

(d) iCaRL-NME (e) Double Distillation

Figure 9: Stacked group prediction bar plots for the different approaches. Each bar shows the correct (green) and wrong (red) number of predictions for a specific group of classes. The blue line indicates the correct number of images belonging to each group (100 images per class in CIFAR-100)

helps in understanding if the model is biased towards some classes/groups, not necessarily the newest ones.

Nevertheless, a qualitative approach is needed to understand which classes between the old and the new ones influence the most these differences. Interestingly, as shown in Table 2, it is not completely correct to say that an higher level of accuracy on the new classes means that the model has *learned better* those classes. In fact, especially the models achieving the highest values of accuracy on the new classes were found to be the least *precise*, meaning that their good performances are due to the fact that fact that they mostly predict images as belonging to new classes even when this is not true (Figure 9).

Method	new classes		old classes	
	recall	precision	recall	precision
iCaRL-NME	0.53	0.45	0.44	0.45
iCaRL-KNN	0.60	0.37	0.43	0.47
Double Dist.	**0.48**	**0.49**	**0.46**	**0.46**
IL2M	0.77	0.27	0.39	0.54
Cosine norm.	0.92	0.20	0.35	0.61

Table 2: Recall and precision for the old and new classes at the last incremental step achieved by the different approaches.

With this analysis we wanted to point out that just looking at the accuracies doesn't fully tell what the model is learning or forgetting. We think that also other measures, like the *precision* and the *recall*, should be taken into consideration when evaluating an incremental task like this one, since they tell us more about what and if the model has learned or forgotten.

References

[1] E. Belouadah and A. Popescu. Il2m: Class incremental learning with dual memory. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 583–592, 2019.

[2] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015.

[3] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] Z. Li and D. Hoiem. Learning without forgetting, 2016.

[5] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning, 2016.

[6] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu. Incremental classifier learning with generative adversarial networks, 2018.