Incremental Learning project

Emanuele Dri
Politecnico di Torino
emanuele.dri@studenti.polito.it

Abstract

The ability to add new classes to an already trained model without having to fully retrain it is known as incremental learning (IL). As today incrementally learning new classes is an open problem in data science mostly because learning these new classes makes neural networks quickly forget knowledge related to old data, which is referred to as catastrophic forgetting. However progress is constantly being made and different solutions have being already proposed to mitigate this phenomenon and to approach the ideal joint training case in terms of performances. In this work some of these techniques are explored and analyzed in order to assess pros and cons and give an overview of this expanding field of research. In particular this analysis will focus on three different approaches: finetuning, Learning without Forgetting and finally the strategy presented by the iCaRL paper with related ablation studies and proposals for further improvements.

1. Introduction

Modern machine learning algorithms perform very well in the classical batch setting: data are given prior to training, hence hyperparameter optimisation and model selection can be based on the full dataset and training can rely on the assumption that the data and its underlying structure are static. However real world applications often are based on a sequential data stream which means that the system needs to learn new data as it comes available and retain previous knowledge of old data. Moreover often old data is no more available making therefore impossible to train the model from scratch every time new data arrives, which in addition would result in a very computationally expensive and time consuming approach. Still it's not trivial to build a model able to incrementally learn new classes without forgetting the old ones and which is trained each time only on new data. In fact if we simply add new classes to an already trained model the inevitable consequence will be *catastrophic forgetting* [8] which consists in forgetting previously learned information when past data is not available

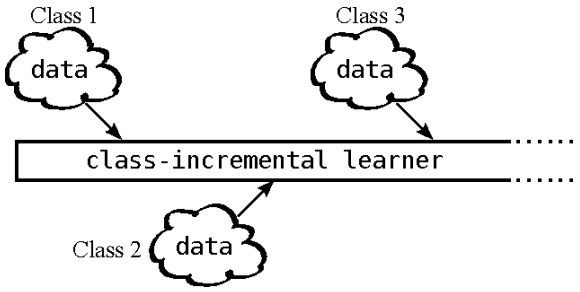


Figure 1: Class-incremental learning: at first the algorithm is trained on a dataset containing only data of the initial class or classes, then at each incremental step new classes arrive and the algorithm should be able at any time to classify all the instances of classes it has seen so far.

To maintain good performances also on old classes and therefore overcame catastrophic forgetting several models have been proposed. With the aim of comparing some of the most prominent ones the first step was to set as baseline the simplest approach: *Finetuning*, this algorithm does not use any strategy to remember what was learned in the past, so implementing it will make possible to observe catastrophic forgetting in action. Moreover this baseline will be useful to measure the improvements made by the other methods implemented for this analysis: *Learning without Forgetting (LwF)* [7] and *iCaRL* [10] (Incremental Classifier and Representation Learning). *LwF* computes in addition to the classical classification loss, a distillation term which ensures that the discriminative information learned previously is not lost during the new learning step: the lower the loss, the better old classes are remembered. This mechanism allows to partially preserve the knowledge of previous classes.

To a certain degree iCaRL builds on the Learning Without Forgetting architecture enhancing it with two smart strategies:

1. a fixed number of the most representative observations for each class seen so far is saved and subsequently used for *prototype rehearsal*

2. a *nearest mean* classifier is used to mitigate the bias towards new classes.

On iCaRL a certain number of ablation studies was also performed and proposals for improvements were advanced.

2. Related work

Several publications each year present new solutions to face the challenges related to class-incremental learning. This section is an attempt to briefly summarize the most significant ones which progressively led to the current state of the art algorithms.

First attempts to learn data representations in an incremental fashion can already be found in the classic neural network literature, in particular, in 1989 McCloskey *et al*. [8] described the problem of *catastrophic forgetting* as the phenomenon that training a neural network with new data causes it to overwrite (and thereby forget) what it has learned on previous data.

A major step in dealing with this problem was the utilisation of *knowledge distillation*. Originally proposed by Hinton *et al*. [3] to transfer information between diverse networks, knowledge distillation is instead used within a single network between different time points by the incremental learning methods presented here. This allows to prevent knowledge acquired on old tasks from deteriorating when adding the new ones. This approach gives the basis for the work made by Li and Hoiem in Learning without Forgetting [7]. Figure 2 shows an overview of the key steps of the LwF algorithm and how the distillation term is used to force the network to retain previously acquired knowledge.

Further improvements in the field of IL were achieved with the introduction of a new strategy: *knowledge rehearsal*, i.e. a continuous stimulation of the network not only with the most recent, but also with earlier data. Of course for this purpose it's necessary to store some elements of old tasks (*exemplars*). Different works explored this idea, implementing it in different ways. Rebuffi *et al*. [10] were certanly among the most successful with iCaRL, which makes use of a combination of a nearest-mean-of-exemplars classifier, a method for prioritized exemplars selection (*herding*) and the use of the above-mentioned distillation loss (Figure 3). Their analysis showed that the system is able to significantly outperform Learning without Forgetting, finetuning and fixed representation in different settings and made it the baseline for many subsequent studies.

In recent years a new impulse to research came from the use of *Generative Adversarial Networks* (GANs) *e.g*. [1] to model the underlying distributions of old classes and select additional real exemplars as anchors to support the learned distribution. When learning from new class samples, synthesized data generated by GANs and real exemplars stored in the memory for old classes can be jointly reviewed to

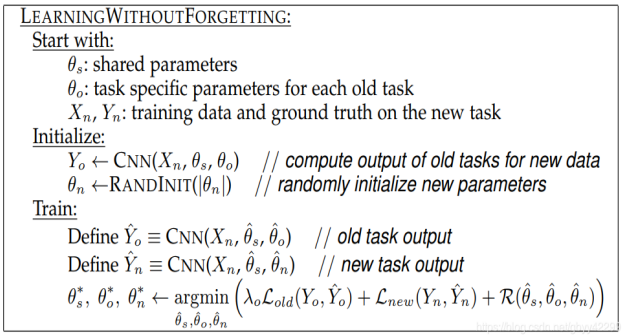


Figure 2: Procedure for Learning without Forgetting.

mitigate catastrophic forgetting. Another aspect of IL on which many recent studies focused their attention was that of imbalance between previous and new data, the work of S. Hou *et al*. [5] dealt exactly with this problem, developing in response a new framework for incrementally learning a unified classifier, i.e. a classifier that treats both old and new classes uniformly[1].

However reviewing the state of the art algorithms in terms of performances it can't be neglected the work of J. Rajasegaran *et al*. [9] which implemented a system that learns a set of generalized parameters, that are neither specific to old nor new tasks. In this pursuit, a novel meta-learning approach is introduced, one that aims at maintaining an equilibrium between all the encountered tasks. This is ensured by a new meta-update rule which avoids catastrophic forgetting. In comparison to previous meta-learning techniques, this approach is task-agnostic. When presented with a continuum of data, the model automatically identifies the task and quickly adapts to it with just a single update. This system led to significant improvements over the state of the art methods (*e.g*. a 21.3% boost on CIFAR100 with 10 incremental tasks).

3. Experiments

Details and results of the experiments conducted are described in this section. The source code can be retrieved at https://github.com/usesnames/IL-project

3.1. Setting

Dataset. The experiments are conducted on a popular dataset for multi-class incremental learning, i.e. CIFAR100 [6]. CIFAR100 is composed of 60000 images with a resolution of 32*32 pixels and belonging to 100 different classes. Every class has 500 images for training and 100 images for testing. During the execution the 100 classes

[1] This work will be further analyzed when proposing improvements for iCaRL

Algorithm 3 iCaRL UPDATEREPRESENTATION

input $X^s, \dots, X^t$ // training images of classes $s, \dots, t$
require $\mathcal{P} = (P_1, \dots, P_{s-1})$ // exemplar sets
require Θ // current model parameters
// form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s,\dots,t} \{(x,y) : x \in X^y\} \cup \bigcup_{y=1,\dots,s-1} \{(x,y) : x \in P^y\}$$

// store network outputs with pre-update parameters:
for $y = 1, \dots, s-1$ **do**
$q_i^y \leftarrow g_y(x_i)$ for all $(x_i, \cdot) \in \mathcal{D}$
end for
run network training (*e.g.* BackProp) with loss function

$$\ell(\Theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Big[\sum_{y=s}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1-g_y(x_i)) + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1-q_i^y)\log(1-g_y(x_i))\Big]$$

that consists of *classification* and *distillation* terms.

Figure 3: Training step of iCaRL.

come in 10 batches of 10 classes each. Testing is instead performed using the test images of all the classes seen so far.

Network. A 32-layer ResNet optimize for CIFAR100 [2] is utilized. However the network presented in [2] is slightly modified in order to simplify feature extraction when needed, in fact in the forward method is possible to retrieve as an alternative to the scores, the normalized feature representations.

Hyperparameters. In order to have a fair comparison among results of different methods the hyperparameters used are the same of those proposed in the iCaRL paper [10] for all the strategies implemented. Therefore the system is trained with a SGD (*stochastic gradient descent*) optimizer, a weight decay equals to 0.00001, mini-batches of size 128, 70 epochs and an initial learning rate of 2.0, this is divided by 5 after 49 and 63 epochs. For the iCaRL method the fixed number of total exemplar, K, is set to 2000 as in the original paper.

3.2. Implementations

Three different strategies are implemented and their results are compared to underline the effectiveness of proposed measures.

3.2.1 Finetuning

Finetuning learns an ordinary multi-class network without taking any measure to prevent catastrophic forgetting. It consists in learning a multi-class classifier for new incoming classes by finetuning the previously learned multi-class classification network, i.e. it starts with a new network which is subsequently trained on the given first task, then it uses the parameters of the old network as a starting point to train on the second task, and so on. This approach results in very poor performances on classes belonging to old batches, in fact it represent the perfect situation in which catastrophic forgetting can be observed: after training, the new weights are appropriate only for the current images but meaningless for the older ones.

3.2.2 Learning without Forgetting

The Learning without Forgetting approach makes use of convolutional neural networks (CNN) in which the network with predictions of the previously learned tasks is enforced to be similar to the network with the current task by using knowledge distillation, i.e. the transferring of knowledge from a large, highly regularized model into a smaller one [4]. According to the LwF algorithm, given a set of shared parameters θ_s across all tasks, it optimizes the parameters of the new task θ_n together with θ_s imposing the additional constraint that the predictions on the samples of the novel task using θ_s and the parameters of old tasks θ_0 do not shift significantly in order to remember θ_0. Given the training data on the new task (X_n, Y_n), the output of old tasks for the new data Y_0, and randomly initialized new parameters θ_n, the updated parameters $\theta_s^*, \theta_0^*, \theta_n^*$ are given by:

$$\theta_s^*, \theta_0^*, \theta_n^* \leftarrow \underset{\hat{\theta_s},\hat{\theta_0},\hat{\theta_n}}{\operatorname{argmin}} (L_{old}(Y_0, \hat{Y_0}) + L_{new}(Y_n, \hat{Y_n}) + R(\hat{\theta_s}, \hat{\theta_0}, \hat{\theta_n}))$$

where $L_{old}(Y_0, \hat{Y_0})$ and $L_{new}(Y_n, \hat{Y_n})$ minimize the difference between the predicted values and the ground-truth values of old and new tasks and R is a regularization term to prevent overfitting. It must be noted that while the original paper [7] proposes a *Multinomial Logistic loss* for the classification term and a modified *Cross Entropy loss* for the distillation term (the latter is multiplied by a loss weight to balance new/old tasks) here instead the implementation proposed by iCaRL and based on a smart utilisation of the *Binary Cross Entropy loss with Logits* will be used:

$$l(x,y) = -\sum_{i=1}^{C} (y_i \log(p(y_i)) + (1-y_i)\log(1-p(y_i)))$$

Where the y_i are the ground truth labels (after *one-hot encoding*) for images belonging to the new classes and the outputs of the previous network (passed through a *sigmoid* function) if the image belongs to the old ones, while $p(y_i)$ is always the predicted binary probability (*sigmoid*) of the class i using the current network. This will make possible to have comparable results with the ones obtained by iCaRL

(obviously using their same hyperparameters). Moreover the network chosen instead of AlexNet is the already mentioned ResNet32.

3.2.3 iCaRL

iCaRL learns classifiers and a feature representation simultaneously from a data stream in a class-incremental form. For classification, iCaRL relies on sets of exemplar images that it selects dynamically out of the data stream. There is one such exemplar set for each observed class so far, and iCaRL ensures that the total number of exemplar images never exceeds a fixed parameter K. More specifically iCaRL implements a mean-of-exemplars classifier to classify images into the set of classes observed so far: to predict a label, y^*, for a new image, x, it computes a prototype vector for each class observed so far $\mu_1, ..., \mu_t$ where $\mu_c = \frac{1}{|P_c|} \sum_{p \in P_c} \psi(p)$ is the average feature vector of all exemplars for a class c. It also computes the feature vector $\gamma(x)$ of the image that has to be classified and assigns the class label with most similar average feature vector:

$$y^* = argmin_c ||\gamma(x) - \mu_c||^2$$

However for the system on which this analysis is based differs from the original implementation when it comes to compute the mean of current classes, in fact since they are still available, all the training images and not only exemplars are used when computing the average feature vector of new classes (*nearest-class-mean classifier*).

The nearest-mean-of-exemplars rule does not have decoupled weight vectors, the class-prototypes therefore automatically change whenever the feature representation changes, making the classifier robust against changes of the feature representation.

For training, every time data for new classes is available iCaRL calls an update routine (Figure 3) that adjusts the network parameters and exemplars based on the additional information available in the current training data. Like for Learning without forgetting a *Binary Cross Entropy Loss with Logits* is used to update the network. The main difference is the presence of the exemplars. In fact the training phase takes into account both new images and exemplars and these result very useful to prevent forgetting old tasks. Under the hood, iCaRL makes use of a convolutional neural network. This network is interpreted as a *trainable feature extractor* followed by a single classification layer with as many output nodes as the total number of classes, this makes the model less general but actually the same results are obtainable using a net that increments its output nodes every time new classes are observed.

At the end there are two modifications to plain finetuning that aim at preventing or at least mitigating catastrophic forgetting. First, the training set is augmented: it consists not only of the new training examples but also of the stored exemplars. By this it is ensured that at least some information about the data distribution of all previous classes enters the training process. Second, the loss function is augmented as well. Again besides the standard classification loss it also contains the distillation loss.

Whenever iCaRL encounters new classes it adjusts its exemplar sets. All classes are treated equally in this, i.e. when t classes have been observed so far and K is the total number of exemplars that can be stored, iCaRL will use $m = K/t$ exemplars for each class. By this it is ensured that the available memory budget for exemplars is always fully used, but never exceeded. Two routines are responsible for exemplar management: one to select exemplars for new classes and one to reduce the sizes of the exemplar sets of previous classes. Exemplars $p_1, ..., p_m$ are selected and stored in an iterative way until the target number, m, is met. In each step of the iteration, one more example of the current training set is added to the exemplar set, namely the one that causes the average feature vector over all exemplars to best approximate the average feature vector over all training examples. Thus, the exemplar "set"is really a prioritized list (the same prioritized construction is used in *herding* [11] to create a representative set of samples from a distribution). The order of its elements matters, with exemplars earlier in the list being more important. The procedure for removing exemplars is therefore particularly simple: to reduce the number of exemplars from any m to m, one discards the exemplars $p_{m+1}, ..., p_m$.

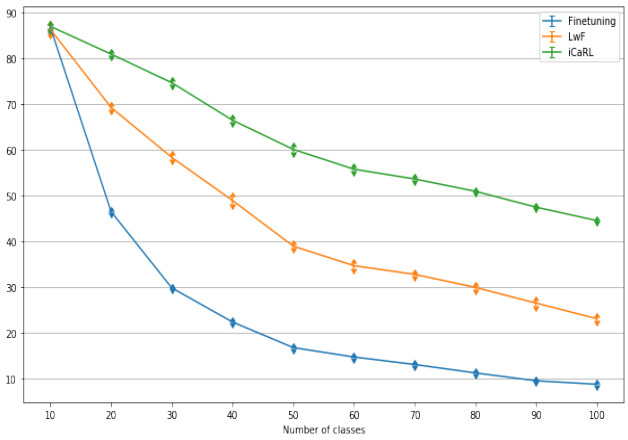
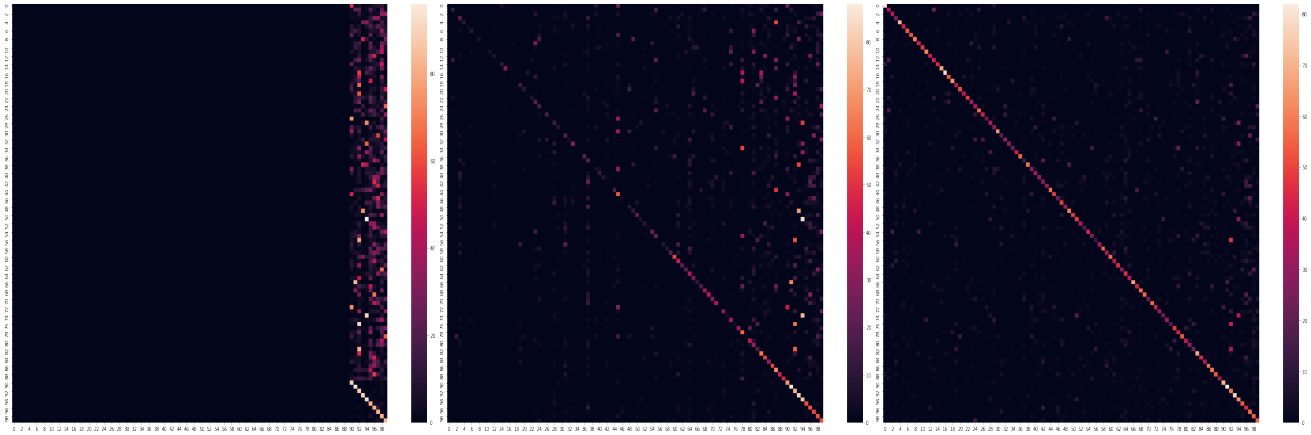


Figure 4: Accuracy obtained for Finetuning, Learning without Forgetting and iCaRL

3.3. Results

Figure 4 shows the results of different methods on the CIFAR100 dataset: finetuning achieves the worst accuracy scores due to catastrophic forgetting, Learning without For-



(a) Finetuning (b) LwF (c) iCaRL

Figure 5: Finetuning, LwF and iCaRL confusion matrices

getting improves on the previous approach thanks to the addition of the distillation term and finally iCaRL significantly outperforms both of them with the introduction of exemplars and of the nearest-mean classifier.

For a further insight we look at the confusion matrix of each model (figure 5): We can see that while in finetuning all predicted class labels come from the last batch of classes that the network has been trained with (one could say that the finetuned network simply forgot that earlier classes even exist), the LwF matrix still shows an inhomogeneous pattern but it's able to predict labels of older classes and even if biased towards the newest ones it represents a great improvement over finetuning. In conclusion iCaRL's confusion matrix looks quite homogeneous over all classes, This shows that iCaRL has almost no intrinsic bias towards or against classes that it encounters early or late during learning. However we are still far away from joint-training scores and further improvements are surely necessary.

4. Ablation studies

To further inspect the iCaRL mechanism, a series of additional experiments were performed, in particular alternatives for the losses and the classifier are proposed and tested changing the original implementation.

4.1. Loss alternatives

Three different combinations of classification and distillation loss are tested in order to study their impact on the model.

4.1.1 CE and BCE

The first alternative combination maintains the BCE with Logits for the distillation term and changes the classification one with a *Cross Entropy loss* which is defined for each image as follows:

$$CE = -\sum_{i}^{C} y_i log(p(y_i))$$

Where y_i are the ground truth and $p(y_i)$ are the CNN scores (passed through the *softmax* activation function) for each class i in C. Making the *softmax* explicit and removing the

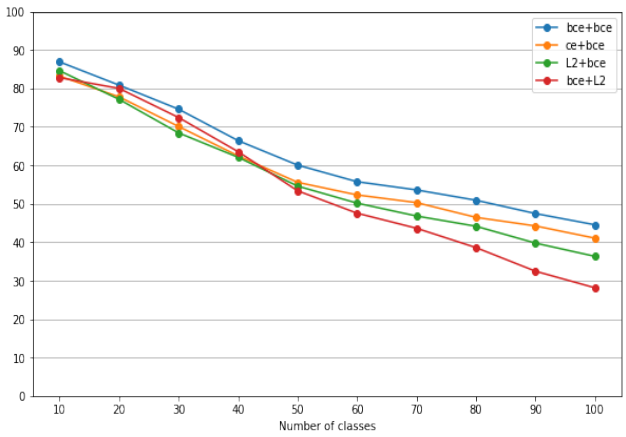
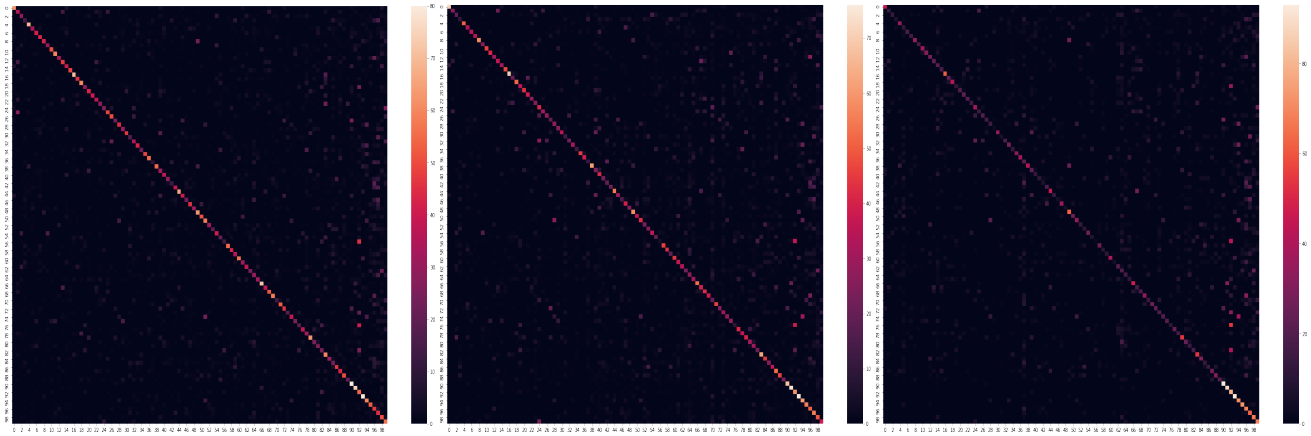


Figure 6: Accuracy of iCaRL changing the combination of losses



(a) ce + bce (b) L2 + bce (c) bce + L2

Figure 7: confusion matrices with different loss combinations

zero-terms, we obtain:

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

Where S_p is the CNN score for the positive class. Note that this value is computed for each image and then the average over the batch is taken. It's important to underline that in this case the original iCaRL learning rate is divided by ten following an heuristic principle to obtain a more reasonable value (0.2). This loss does not require a binary form for the labels and it must be noted that here the loss terms coming from the negative classes are zero. However, the loss gradient respect those negative classes is not cancelled, since the Softmax of the positive class also depends on the negative classes scores. The BCE instead is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values.

4.1.2 L2 and BCE

Also for this combination only the classification term is changed, this time it is replaced with an *L2 loss* which is based on the euclidean distance between label and prediction, so for each image:

$$l = (\mathbf{y} - \hat{\mathbf{y}})^2$$

Where $\hat{\mathbf{y}}$ is the softmax of the network output and $\mathbf{y}$ is the one hot encoding of the label. Also here this value is calculated for each image and then the mean over the entire batch is computed. This loss measures the distance between the probability predicted and the one hot encoding. This is interesting because this loss penalizes higher probabilities assigned to the wrong classes, and also low probability for the correct one.

4.1.3 BCE and L2

In this case the L2 loss is used for the distillation term while BCE remains the choice for the classification part. The inputs for the distillation are again the sigmoid of the old network outputs and this combination is tested to see if the L2 loss described before can outperform the BCE as distillation term.

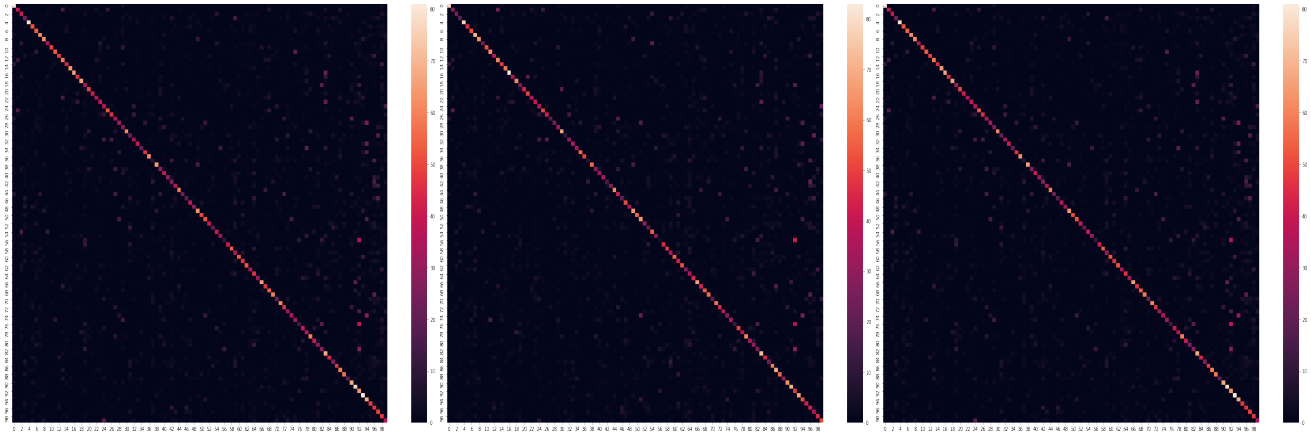
Results. Figure 6 shows that no improvements over the original version were achieved changing the loss combination. In particular for the last batches of classes the results get worse and especially when substituting the distillation terms the results obtained are far from optimal. Observing in Figure 7 the confusion matrix of the last combination, an accentuated bias towards the newest classes can be noted.

4.2. Classifier alternatives

Also here three alternative algorithms are proposed as classifiers. Note that only the exemplars features are used to train these classifier in order to have balancing among new and old classes, in fact if all the current training images are used, the major quantity of data for new classes will bias the classifier, greatly deteriorating test accuracy. Moreover in this case features are extracted after exemplars construction so that also current classes are represented.

4.2.1 SVM

The objective of the support vector machine algorithm is to find the hyperplane in the features space that distinctly classifies the data points with the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more



(a) SVM (b) KNN (c) Random forest

Figure 8: confusion matrices with different classifiers

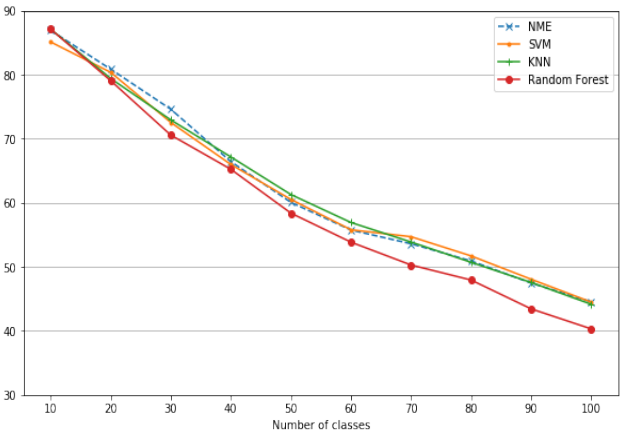


Figure 9: Accuracy of iCaRL changing the classifier

confidence. Support Vector Machine is tested for its robustness and because can work well (exploiting the *kernel trick*) also when data points are not linearly separable. This said, using it with iCaRL gave the best results when implementing the linear version of SVM, therefore data could be somewhat linearly separable. Moreover among the three values tried for the C hyperparameter (the default value 1.0, one order of magnitude more and one order less), that indicates the measure in which mistakes are allowed, the best value was exactly the default one.

4.2.2 KNN

KNN was tested to see if it could bring improvements over the nearest class mean of exemplars system, both of them are built on a mechanism based on distances but the first differs from the latter because instead of taking the minimum Euclidean distance between the features of the image and the mean of exemplars among all seen classes, it assigns the predicted label by majority voting looking at the distances from the nearest K neighbors. It must be noted that the optimal choice of the value K is highly data-dependent. Here [5, 10, 15] were tried as values of K and 10 gave the best result.

4.2.3 Random forest

Random forest consists of an ensemble of individual decision trees. Each individual tree in the random forest outputs a class prediction and the final one is decided by majority voting. The reason why it was tested is that a large number of relatively uncorrelated models (trees) operating as a committee could often improve the performances of a model. Uncorrelated models can in fact produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this effect is that often while some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. Again variations of the default hyperparameters were tempted (doubling or halving the default number of threes, changing the minimum number of samples for a leaf node to 10 and 20) but none of these outperformed the default values of 100 threes and one sample as minimum for a leaf node.

Results. As observable in Figure 9 both the SVM and the KNN classifier performed very similarly to the iCaRL's NME classifier, Random forest instead gets worse approaching the last batches of classes. In Figure 8 the patterns look quite homogeneous for all the three proposed alternatives and no significant bias seems observable even with no improvements over iCaRL's base implementation.

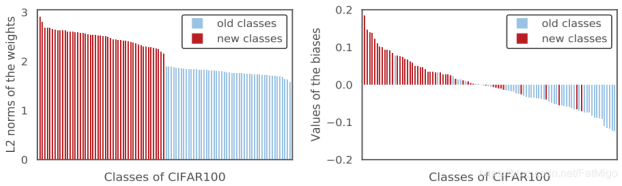


Figure 10: Visualization of the weights and biases in the last layer for old and new classes in the incremental setting of CIFAR100 by iCaRL.

5. Proposals for improvement

In order to establish on which portion of the iCaRL's algorithm should focus to increase its performances, the first step was to analyze one of its weakest points: how it tackles the significant imbalance between old and new classes. To deal with the issue, iCaRL proposes the already mentioned nearest-mean-of-exemplars classification strategy, while doing this it makes improvements over LwF and other methods, its performance on long sequences of classes is still not satisfying. In this work, the aim is to tackle this problem by incorporating a custom implementation of some of the components originally proposed by S. Hou *et al.* [5]: *cosine normalization* and *less-forget constraint*, which address the classes imbalance.

5.1. Cosine normalization

As shown in figure 10, in iCaRL the magnitudes of both the embeddings and the biases for the new classes are significantly higher than those for the old classes. This results in the bias in the predictions that favor new classes. To address this issue, the paper proposes the use of a cosine normalization in the last layer:

$$p_i(x) = \frac{\exp\left(\eta \left\langle \bar{\theta}_i, \bar{f}(x) \right\rangle\right)}{\sum_j \exp\left(\eta \left\langle \bar{\theta}_j, \bar{f}(x) \right\rangle\right)}$$

where $\bar{v} = v/||v||_2$ denotes the l2-normalized vector, and $\langle \bar{v_1}, \bar{v_2} \rangle = \bar{v}_1^T \bar{v}_2$ measures the cosine similarity between two normalized vectors. The learnable scalar η is introduced to control the peakiness of softmax distribution since the range of the cosine similarity is restricted to [1,1]. Cosine normalization can effectively eliminate the bias caused by the significant difference in magnitudes. It is also noteworthy that due to cosine normalization, the scores before softmax all lies in the same range (i.e.[1,1]) and thus are comparable. Geometrically, the normalized features and the class embeddings lie on a high-dimensional sphere.

5.2. Less-forget constraint

One of the practical challenges for incremental learning is how to less forget the previous knowledge. To this

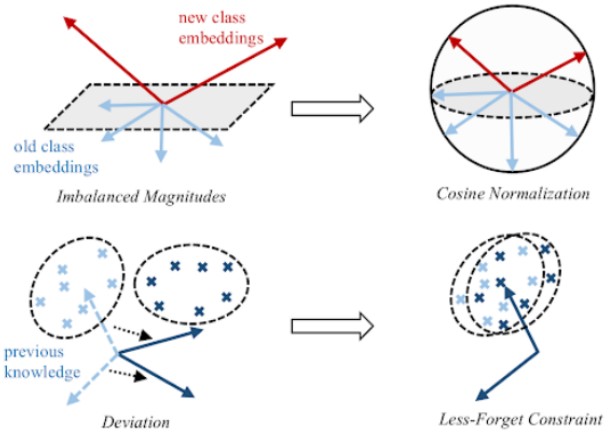


Figure 11: Illustration of the adverse effects caused by the imbalance between old and new classes in multi-class incremental learning, and how this approach tackles them.

end, the paper introduces a less-forget constraint through a new loss L_{dis}^G, which provides a stronger constraint on the previous knowledge compared to iCaRL's distillation loss. Specifically, the latter mainly considers the local geometric structures, i.e. the angles between the normalized features and the old class embeddings. This constraint is not able to prevent the embeddings and the features from being rotated entirely. To enforce a stronger constraint on the previous knowledge, the idea is to fix the old class embeddings and compute a novel distillation loss on the features as below:

$$L_{\text{dis}}^{\text{G}}(x) = 1 - \left\langle \bar{f}^*(x), \bar{f}(x) \right\rangle$$

where $\bar{f}^*(x)$ and $\bar{f}(x)$ are respectively the normalized features extracted by the original model and those by the current one. L_{dis}^G encourages the orientation of features extracted by current network to be similar to those by the original model. The loss is bounded ($L_{dis}^G \leq 2$). The rationale behind this design is that the spatial configuration of the class embeddings, to a certain extent, reflects the inherent relationships among classes. Hence, to preserve the previous knowledge, a natural idea is to keep this configuration. With the old class embeddings fixed, it is then reasonable to encourage the features to be similar as in L_{dis}^G.

5.3. Implementation and results

As already underlined, a custom implementation of these measures is adopted for this analysis, this means that three fundamental differences exist between the version used here and the one proposed by S. Hou *et al.* [5]:

1. The original paper uses for the distillation term an adaptive loss weight which is defined as

$$\lambda = \lambda_{base} \sqrt{|C_n|/|C_o|}$$

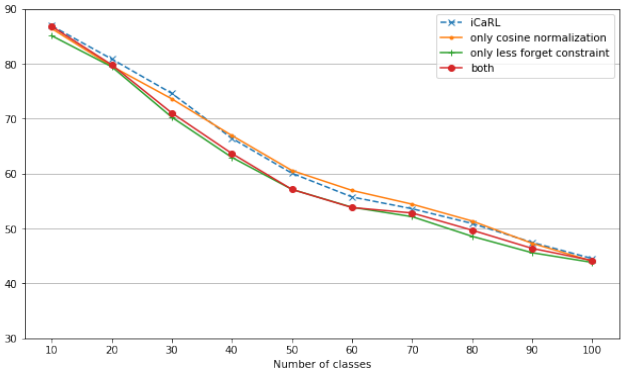


Figure 12: Accuracy of the iCaRL implementation compared with these of the proposed variations

where $|C_o|$ and $|C_n|$ are the number of old and new classes in each phase, λ_{base} is a fixed constant for each dataset (5 for CIFAR100). In general, λ increases when the ratio of the number of new classes to that of old classes increases. Instead here the formula used[2] is:

$$\lambda = \lambda_{base}\sqrt{|C_o|/|C_n|}$$

with λ_{base} set at 6 after finetuning. Now λ increases so that the network will give more importance to the distillation term if there are more old classes already seen.

2. Here the iCaRL's nearest mean classifier is kept as a further measure for dealing with bias between old and new classes, instead in the original implementation the final layer of the network is used to compute the predictions.

3. For this analysis the setting is as close as possible to the one of iCaRL, which means that the algorithm sees ten new classes every iteration, starting from zero and finishing at one hundred, differently from the original paper in which the system is firstly trained on fifty classes and then starts receiving the remaining fifty in batches of five or ten per iteration. Moreover the hyperarameters are also kept identical to those of iCaRL to have a fair comparison, the only exception is the learning rate which is set to 0.2 in order to be suitable for the Cross Entropy loss used for the classification term.

Results. As it can be seen in Figure 12 no improvements were achieved over the iCaRL implementation. To further inspect the reasons, the system was tested also with only one of the two variations proposed. It must be noted that while the cosine normalization added to the original algorithm gives results in line with those observed without implementing it, for the less forget constraint the scores are even worse of those obtained by iCaRL, at least until the last batches where it actually returns on similar levels. These absence of improvement can be explained taking into consideration the different setting in which the experiment was conducted with respect to the original paper and the different choice of hyperparameters done in order to more fairly compare the results with those of the iCaRL strategy.

[2] actually the following formula is also the one used in the online python implementation of the paper, however no explanation for this discrepancy with the paper itself was to be found.

6. Conclusions

This work compared several strategies to tackle IL and tried to improve one of the most prominent ones: iCaRL. Two modifications to the original iCaRL implementation were tested with that aim: *cosine normalization* and *less-forget constraint*, both of them deal with the bias toward new classes proposing solutions to normalize the probabilities in one case and to retain knowledge on previous classes in the other. Further possibilities for improvements could reside in the third proposal of the S. Hou *et al.* paper [5] not tested here: a margin ranking loss to ensure that old and new classes are well separated. Specifically, for each reserved sample x, the authors tried to separate the ground-truth old class from all the new classes by a margin, using x itself as an anchor. They considered the embedding of the ground-truth class as positive. To find the hard negatives, an online mining method is proposed. Those new classes that yield highest responses to x will be selected as hard negative classes and their embeddings used as negatives for the corresponding anchor.

However the expanding field of research which is that of incremental learning cannot even remotely be fully represented by this analysis, in particular approaches based on *meta-learning* are emerging as serious candidates to drastically decrease *forgetting* and reach results that until recently were thought possible only in non incremental settings. On this note one of the best examples is the already mentioned paper by J. Rajasegaran *et al. iTAML: An Incremental Task-agnostic Meta-learning Approach* [9] whose experiments demonstrate impressive improvements across a range of classification datasets including ImageNet, CIFAR100, MNIST, SVHN and MS-Celeb.

References

[1] C. He, R. Wang, S. Shan, and X. Chen. Exemplar-supported generative reproduction for class incremental learning, 2018. 2

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learn-ing for image recognition, 2015. 3

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

[4] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2014. 3

[5] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing, 2019. 2, 8, 9

[6] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009. 2

[7] Z. Li and D. Hoiem. Learning without forgetting, 2017. 1, 2, 3

[8] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem, 1989. 1, 2

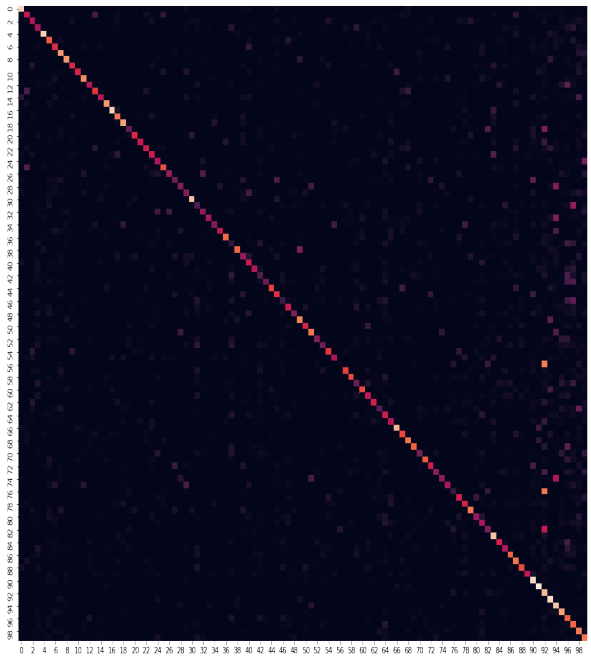
[9] J. Rajasegaran, S. Khan, M. Hayat, F. S. Khan, and M. Shah. itaml: An incremental task-agnostic meta-learning approach, 2020. 2, 9

[10] S. A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning, 2017. 1, 2, 3

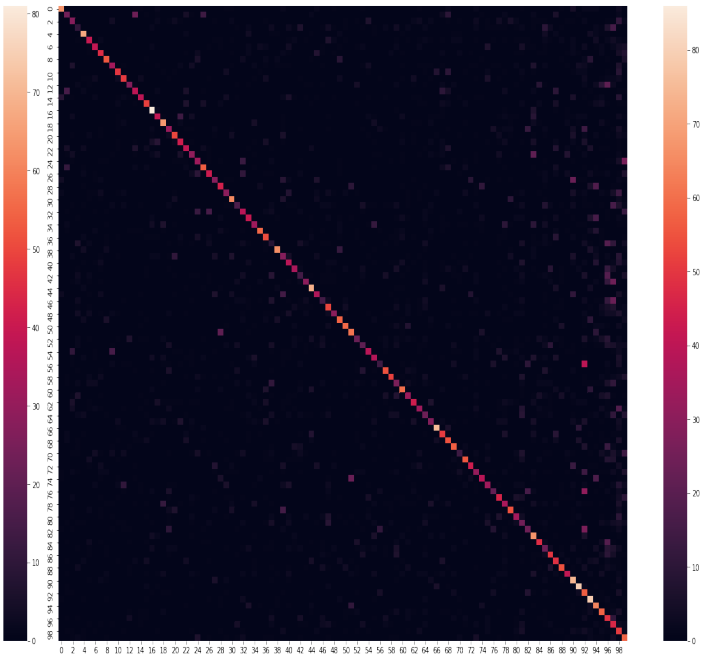
[11] M. Welling. Herding dynamical weights to learn, 2009. 4

Appendices

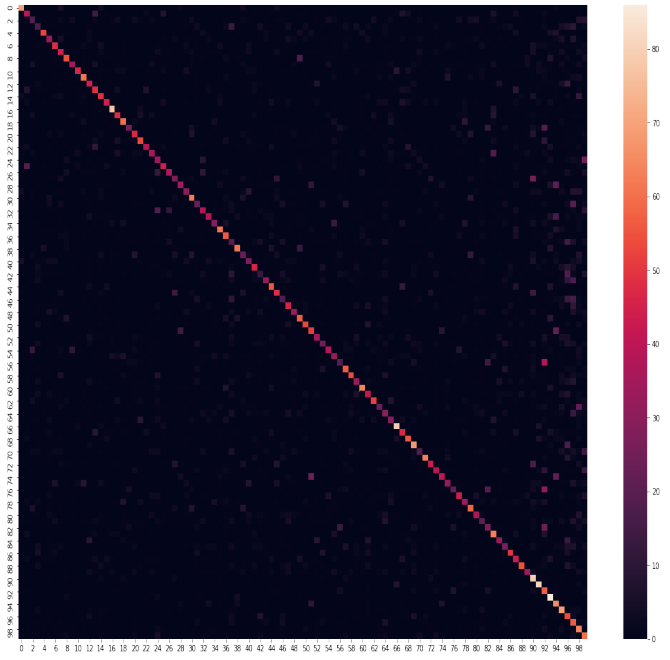
A. Confusion matrices for improvement proposals



(a) only cosine normalization



(b) only less forget constraint



(c) Cosine normalization + less forget constraints