Incremental Learning in Image Classification

Cavagnero Niccolò
s277755@studenti.polito.it

Mascarello Andrea
s276798@studenti.polito.it

Serra Alessio
s277756@studenti.polito.it

Abstract

While for living beings learning is inherently incremental, one of the major differences and drawbacks of artificial neural networks is the incapability of such systems to learn new concepts without forgetting old knowledge if not being re-trained on all previous data due to the so called Catastrophic Forgetting. For this reason, we are going to analyze two of the most common state-of-the-art techniques, Learning Without Forgetting and iCaRL, whose aim is to prevent or at least to mitigate this phenomenon, trying to improve them.

1. Introduction

In recent years, many attempts to deal with the Catastrophic Forgetting issue have been made but the two most successful and widely used methods are the addition of a regularization term that encourages the network to reproduce results at previous states, i.e. the distillation loss introduced in Learning Without Forgetting [1], and the rehearsal of samples of old classes as shown in iCaRL [2].

Alongside the classification loss for current classes, the distillation loss consists in using the outputs of the nodes corresponding to previous classes to force the network to reproduce the same outputs of those nodes at time *t-1*, helping the model to maintain the previously learned representation.

Formally:

$$L = -\sum_{c \in K^{t-1}} [p^c_{\theta^{t-1}} log(p^c_{\theta^t}) + (1 - p^c_{\theta^{t-1}}) log(1 - p^c_{\theta^t})]$$

where $p^c_{\theta^t}$ is the binary probability of the class c using the network θ at time t and K^{t-1} is the set of classes known at time *t-1*.

In iCaRL, S. A. Rebuffi et al showed that the rehearsal of old samples, combined with the distillation loss, enables the network to outperform the models built only with this second one. However, this violates one of the core assumptions of an incremental setting, i.e. the availability of samples of previous classes: this issue could be solved, as proposed in [3] and [4], through the use of GANs able to reproduce those samples, but in this case too the network would be trained using all classes and so it still cannot be considered a proper incremental setting. However, exemplars are widely used in the literature and current state-of-the-art results are achieved by means of them.

2. Baselines

First, we replicated some of the experiments showed in iCaRL using the CIFAR100 dataset training the various models in a class-incremental way on the available training data using batch of 10 classes and then evaluating them on all the classes seen up to that moment.

As done in the original paper, the images are preprocessed by subtracting the per-pixel mean and during the training phase the dataset is augmented by means of a random cropping with pad = 4 and a random horizontal split.

Following again the original implementation, we used a 32-layers ResNet, a learning rate that starts at 2.0 and is divided by 5 after 49 and 63 epochs, mini-batches of size 128 and a weight decay of 0.00001. The maximum number of exemplars is set to 2000. The source code is available at: `https://github.com/NiccoloCavagnero/IncrementalLearning`.

All the experiments are carried out with the classes randomly arranged in fixed batches and each experiment is repeated with three different random splits.

We reported both averages and standard deviations of the accuracies obtained at each incremental step together with the average incremental accuracy among the different steps, excluding the first one that is not properly incremental.

With these settings we first tried to simply fine-tune the network on the new classes and as expected Catastrophic Forgetting causes the model to classify all of the samples with the labels of the last seen batch.

Next, adding the distillation loss the network is able to remember some of the old classes but it still suffers of a strong bias towards the last ones.

We then reported the configuration called 'hybrid1' showing this way both the impact of the rehearsal, comparing it to LwF, and the effectiveness of the second main

Batch	1	2	3	4	5	6	7	8	9	10	Avg
FineTune (%)	84.6±3.1	45.0±0.2	28.9±1.1	21.4±0.3	18.1±0.6	14.6±0.2	12.6±0.3	11.5±0.05	9.50±0.2	8.9±0.2	18.9±0.6
LwF (%)	84.1±3.1	67.6±2.4	56±3.6	47.5±3.7	43.7±3.8	38.6±3.0	32.2±2.5	30.3±1.4	26.6±1.3	24.7±0.5	40.8±2.5
hybrid1 (%)	87.6±2.4	76.7±1.0	**71.2±2.1**	63.6±1.4	60.7±2.4	54.9±1.9	47.8±0.8	45.4±0.3	41.0±0.9	37.5±0.2	55.4±1.3
iCaRL (%)	84.5±2.5	**76.9±0.8**	70.5±3.0	**64.4±1.7**	**61.1±2.9**	**57.6±2.3**	**53.7±1.4**	**51.4±0.9**	**48.2±0.8**	**45.6±0.7**	**58.8±1.7**

Table 1. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps for Fine-Tuning, LwF, hybrid1 and iCaRL.

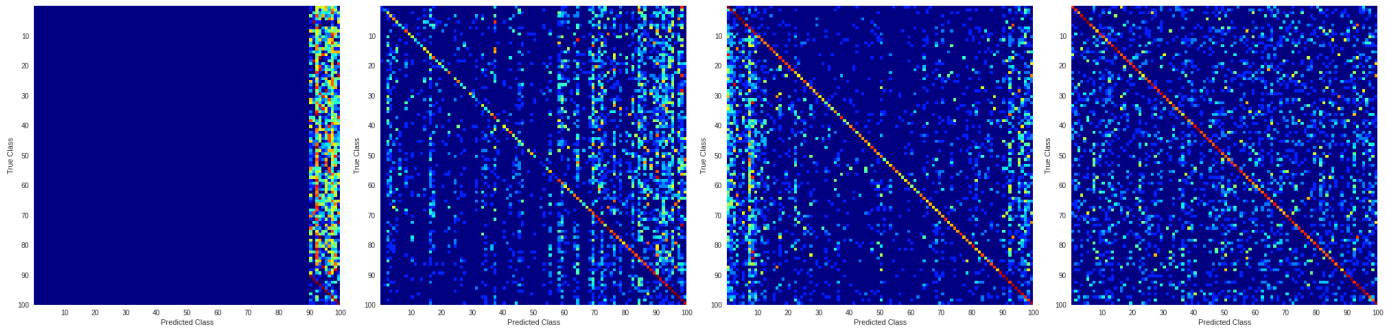

Figure 1. Confusion matrices of different methods with entries transformed by $\log(1+x)$ for better visibility. From left to right: Fine-Tuning, Lwf, hybrid1 and iCaRL.

component of iCaRL, the Nearest Mean of Exemplars, or NME, classifier against the use of the fully connected layer trained with the network.

Using the prototype rehearsal showed in iCaRL, thus ensuring that at least some information about the distribution of the previously seen classes enters into the training phase, the network ability in remembering old knowledge is greatly enhanced but only thanks to NME the errors start to be approximately equally distributed among all the incremental steps with only a small bias towards the new ones, avoiding the strong bias of the fully connected layer towards first and last seen classes.

3. Ablation Study

Once the base models have been built, we tried to use different types of losses and classifiers in order to find out if there are better choices than the ones originally used by the authors.

Since the herding phase suggested in [2] is very expensive and does not provide tangible improvements, as also noticed in [5] and [6], all next experiments are carried out selecting the exemplars in a random fashion.

3.1. Losses

We chose to investigate the following three types of losses:

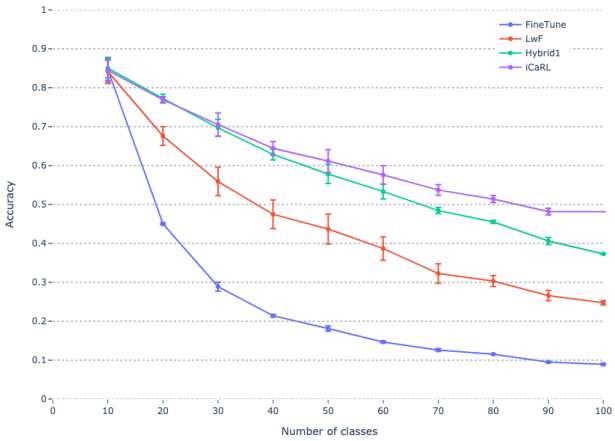

Figure 2. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps of different baseline settings.

- L2, one of the simplest possible losses:

$$L2 = \sum_{i=1}^{N} (y_i - \hat{y_i})^2 \tag{1}$$

- Cross Entropy, one of the classic losses for neural networks:

$$H = -\sum_{i=1}^{N} y_i \log \hat{y_i} \tag{2}$$

- KLDivergence, also called Relative Entropy, that

Batch	1	2	3	4	5	6	7	8	9	10	Avg
L2 (%)	86.4±2.0	**78.8±0.7**	**71.5±2.3**	**65.6±0.8**	**62.5±1.6**	**59.3±1.0**	**55.2±0.1**	**52.9±0.9**	**48.9±0.5**	**46.6±0.8**	**60.1±1.0**
CE (%)	85.1±2.0	73.1±1.0	64.0±1.2	57.5±0.3	54.3±1.2	50.7±0.5	46.4±0.8	44.5±0.8	41.5±0.4	41.4±0.5	52.6±0.9
KLDiv (%)	85.1±2.5	71.8±0.4	63.5±1.3	56.2±0.5	52.4±1.2	48.4±1.0	44.1±1.1	42.0±0.6	38.6±0.7	36.5±0.6	50.4±1.0
BCE (%)	84.5±2.5	76.9±0.8	70.5±3.0	64.4±1.7	61.1±2.9	57.6±2.3	53.7±1.4	51.4±0.9	48.2±0.8	45.6±0.7	58.8±1.7

Table 2. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps for L2, CrossEntropy, KLDivergence, BCE(iCaRL).

should behave similarly to the former:

$$D_{KL} = \sum_{i=1}^{N} y_i \log \frac{y_i}{\hat{y}_i} \tag{3}$$

While Cross Entropy is widely used in the literature [1], [5], [6], [7], [8], [9], [10] and [11], L2 and KLDivergence are less common but they should achieve good results according to [12].

All these types of losses are used for both classification and distillation since they may give very different numerical results, even on different scales, and this should have been taken into account with a weighting parameter in order to properly balance classification and distillation.

The hyper-parameters of the network are the original ones, that have proved to be quite effective also with the other losses.

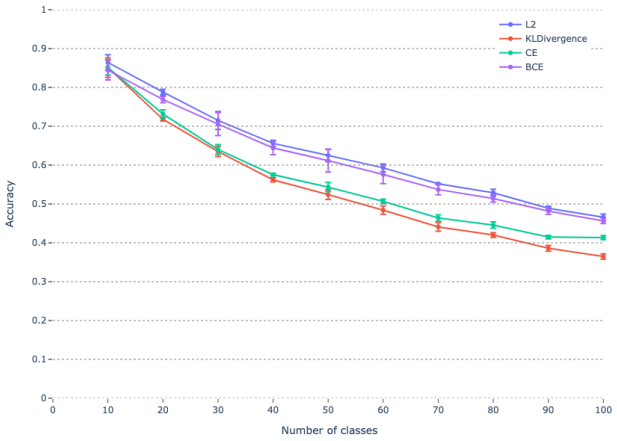

Figure 3. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps of different types of losses.

Since the Cross Entropy and the KLDivergence require the outputs of the network to be probabilities, for these two losses we substitute the sigmoid with a softmax layer for the first and a logarithmic softmax layer for the second one.

In terms of accuracy L2 gives slightly better results with respect to BCE and it is also more stable across different splits, while Cross Entropy and KLDivergence perform quite poorly.

This may be due to the fact that, using a one hot encoding for classification, for each sample the only non-zero contribution comes from the output node corresponding to the true label, while in BCE or L2 this does not happen.

Moreover, again because of the one hot encoding, Cross Entropy and KLDivergence are mathematically equivalent in classification so the first must be better than the second in distilling previously learnt knowledge: indeed the KLDivergence starts to achieve lower scores from the fourth batch and the gap between the two widens as the number of learnt classes increases, i.e. when there is more the necessity of preserve old knowledge.

It must be noticed that the lower results may be caused by the softmax layer, since adding it to the model cause a decrease in performance even when using BCE or L2: we believe that the softmax does not allow to preserve the knowledge since the output of each node depends on the other nodes. Moreover, there is no way that the outputs of the network can match the old outputs used for distillation since both sum to 1 but new outputs depend on more nodes.

3.2. Classifiers

With respect to the classifiers, we chose to investigate three different models:

- KNN, due to its conceptual similarity to NME;
- SVM with RBF kernel, the state-of-the-art technique for classification before Deep Learning;
- Gaussian Naïve Bayes, because differently from the previous ones it is not a distance based method and does not suffer both data on different scales or high dimensional.

Since the choice of the classifier does not affect the updating representation phase of iCaRL, we used again the original hyper-parameters for the network while for the first two classifiers, we inspected their behavior trying with different hyper-parameters.

In order to always train the classifiers with a balanced dataset, only the exemplars are used as training data.

Batch	1	2	3	4	5	6	7	8	9	10	Avg
KNN (%)	85.8±2.6	78.2±0.3	70.0±2.5	64.0±1.8	58.8±3.8	53.1±3.5	48.0±1.0	44.1±0.7	40.6±0.9	38.3±0.6	55.0±1.7
SVM (%)	85.4±3.0	**78.3±0.4**	**72.0±2.4**	**66.6±2.0**	**64.7±0.7**	**59.0±2.4**	**54.1±0.8**	50.2±0.5	46.7±0.8	44.2±0.3	**59.5±1.3**
Naïve Bayes (%)	83.3±3.1	76.6±0.9	70.3±1.6	63.7±1.3	61.3±2.5	56.6±2.0	51.9±0.6	49.2±0.7	44.7±0.5	41.9±0.9	57.3±1.4
FC-Layer (%)	87.6±2.4	76.7±1.0	71.2±2.1	63.6±1.4	60.7±2.4	54.9±1.9	47.8±0.8	45.4±0.3	41.0±0.9	37.5±0.2	55.4±1.3
NME (%)	84.5±2.5	76.9±0.8	70.5±3.0	64.4±1.7	61.1±2.9	57.6±2.3	53.7±1.4	**51.4±0.9**	**48.2±0.8**	**45.6±0.7**	58.8±1.7

Table 3. K-Nearest Neighbours, Support Vector Machine, Gaussian Naïve Bayes, FC-Layer(hybrid1) and NME(iCaRL). SVM and KNN refer to the best configuration of these models among the ones inspected.

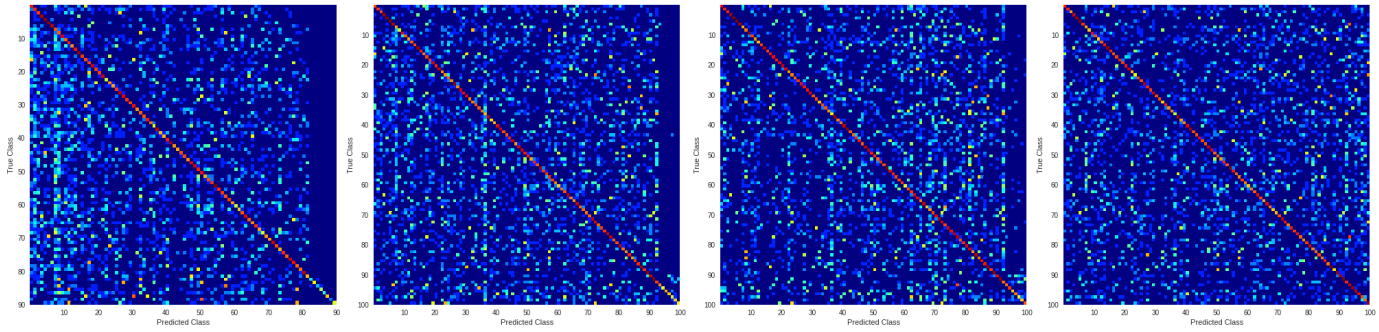

Figure 4. Confusion matrices of different classifiers with entries transformed by $\log(1+x)$ for better visibility. From left to right: KNN, SVM, Naïve Bayes and NME(iCaRL). SVM and KNN refer to the best configuration of these models among the ones inspected.

KNN performances increase as the parameter K is increasing, probably because the effect of outliers is reduced and because the predictions are taken out more similarly to NME that considers all training samples, through their means of class.

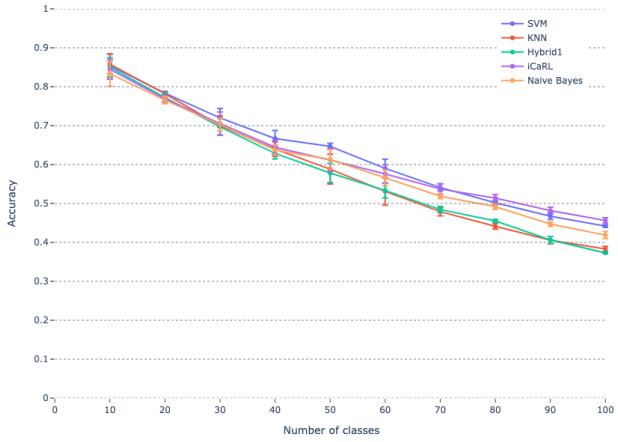

Figure 5. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps of different types of classifiers. SVM and KNN refer to the best configuration of these models among the ones inspected.

KNN shows a strong bias towards old classes and seems less able to learn new concepts as the number of steps increases.

Batch	K=1	K=3	K=5	K=7
1	84.2±2.5	84.7±2.7	85.0±2.4	85.8±2.6
2	76.8±1.2	77.2±0.4	77.8±0.4	**78.2±0.3**
3	67.2±2.7	69.0±2.6	69.3±2.9	**70.0±2.5**
4	59.2±2.1	61.4±1.7	62.7±1.2	**63.4±1.8**
5	54.9±4.9	56.9±3.5	57.9±3.5	**58.8±3.8**
6	50.0±3.9	51.3±2.4	52.6±3.5	**53.1±3.6**
7	44.5±1.8	46.1±0.5	46.8±0.9	**47.9±1.0**
8	40.4±0.5	42.1±0.3	43.0±0.5	**44.1±0.7**
9	36.7±2.4	38.4±0.5	39.5±1.1	**40.6±0.9**
10	34.7±1.4	36.0±0.6	37.3±0.4	**38.3±0.6**
Avg	51.6±2.3	53.1±1.5	54.1±1.7	**55.0±1.8**

Table 4. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps for KNN as parameter K varies.

For the Kernel SVM, different values of C have been tested: 0.001, 0.01, 0.1, 1. This model seems to be the most accurate and balanced classifier among the ones inspected and does not seem to show any kind of bias, differently from KNN and Naïve Bayes. The highest scores are achieved with the default value C = 1 and until the seventh incremental step SVM is able to outscore NME, that however is better at the end when there are fewer exemplars.

Naïve Bayes can match the NME results up to the fifth step and then the gap from iCaRL begins to get wider as the number of exemplars decreases. Moreover this classifier seems to show a small bias towards more recent classes.

It is also interesting that, somehow, almost no sample

Batch	C=0.001	C=0.01	C=0.1	C=1
1	84.2±2.6	84.6±2.3	85.2±2.9	85.4±3.0
2	76.9±0.9	77.1±0.7	78.1±0.4	**78.3±0.4**
3	70.3±1.7	70.3±1.6	71.1±2.5	**72.0±2.4**
4	64.6±0.8	64.6±0.8	65.5±1.0	**66.6±2.0**
5	61.4±2.5	61.4±2.5	62.1±2.8	**64.7±0.8**
6	56.5±2.5	56.5±2.5	57.6±2.0	**59.0±2.4**
7	51.5±1.2	51.5±1.2	52.4±1.2	**54.1±0.8**
8	48.4±1.3	48.4±1.3	48.6±1.2	**50.2±0.5**
9	45.5±0.3	45.5±0.3	46.0±0.3	**46.7±0.8**
10	41.4±1.1	41.4±1.1	42.1±0.8	**44.2±0.3**
Avg	57.4±1.5	57.4±1.5	58.1±1.5	**59.5±1.3**

Table 5. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps for SVM as parameter C varies.

of older classes is assigned to the ones of the last batch, but we do not know the reason behind this strange behavior that is not present in NME or in the fully connected layer and that seems to appear only during the last incremental steps.

Indeed, until the third batch all the classifiers give approximately the same scores but when the number of exemplars starts to decrease none of these models is able to match NME's performances, with the exception of SVM that achieves better results at least up to the seventh batch.

It is worth to notice that hybrid1's fully connected layer achieves circa the same results as KNN that is the worst classifier, implying that the quality of taught logits is very low.

4. iCaRL2

In this section we discuss the differences and improvements that constitute our evolution of iCaRL, which we simply called iCaRL2.

Our model tries to address two problems which iCaRL suffers of, such as the limited exploiting of available knowledge and the fully connected layer's bias towards the first and the last seen classes.

The first improvement to iCaRL is a simple change in the hyper-parameters: using a higher weight decay helps reducing both the overfitting and the plasticity of the network, naturally allowing to preserve more knowledge. However, the reduced plasticity prevents the network to adapt in the last incremental steps and for this reason we chose for a decreasing policy of the weight decay: at each incremental step the value of the weight decay is lowered. We used 10 evenly spaced points between 0.0001 and 0.00001.

Next, taking inspiration from Castro et al [9] we introduce a new loss that allows to better exploit the data:

- the classification loss is computed on all the classes, and not only on the last incremental batch;
- for the distillation loss, the target logits of classes learned at time *t* are taught by the network at the state *t*;
- the distillation loss is weighted by an increasing parameter λ, as also proposed in [5], [6], [7], [8], [10], [12] and [13], that starts from 0.3 and is augmented of 0.25 at each incremental step (this values are based on our empirical results);
- Temperature with τ = 1/2 is used for the distillation forcing the network to minimize even the smaller errors learning this way a finer representation, similarly to what is done in [1], [5], [7], [9], [10] and [11];
- BCEWithLogits is replaced by a sigmoid + L2, because the use of the Temperature would require a sigmoid + BCE that is numerically unstable and because we also previously found that L2 gives slightly better results than the original type, being also more stable across different random splits.

Thus, our loss can be expressed as:

$$L = \sum_{c \in K^t} (y_c - p^c_{\theta^t})^2 + \lambda \sum_{c \in K^{t-1}} [(p^c_{\theta^{t-i}})^\tau - (p^c_{\theta^t})^\tau]^2$$

where y_c is the one hot encoding of the true label, $p^c_{\theta^t}$ is the binary probability of the class *c* using the network θ at time *t*, K^t is the set of classes known at time *t*, *i* is the number of steps after the class *c* has been learnt and τ is the Temperature.

The last improvement, again inspired by [9], is the addition of a stabilization phase of 10 epochs after each incremental step to adjust both the representation and the fully connected layer using:

- only the exemplars as training data;
- a smaller learning rate of 0.08, the last value during the representation updating;
- the classification loss only;
- the initial weight decay, since we don't want drastic changes in the weights even without the distillation loss.

This phase greatly reduces the bias in the fully connected layer, especially during the last tasks, leading to a more

Batch	1	2	3	4	5	6	7	8	9	10	Avg
iCaRL (%)	84.5±2.5	76.9±0.8	70.5±3.0	64.4±1.7	61.1±2.9	57.6±2.3	53.7±1.4	51.4±0.9	48.2±0.8	45.6±0.7	58.8±1.7
new_decay (%)	88.1±2.0	81.5±1.3	75.1±1.3	69.0±1.0	65.4±2.0	61.3±1.6	56.6±0.7	53.6±0.1	49.3±0.4	46.6±0.6	62.0±1.1
new_loss (%)	88.6±1.7	82.9±1.5	76.4±2.3	69.4±1.7	67.0±2.2	62.9±1.8	58.8±1.4	56.1±0.4	52.3±0.5	48.8±0.3	63.8±1.4
iCaRL2 (%)	88.4±1.7	**83.5±1.0**	**76.6±2.0**	**70.5±0.9**	**67.5±2.2**	**63.7±1.2**	**59.3±0.9**	**56.9±0.7**	**53.1±0.8**	**50.5±0.4**	**64.6±1.2**

Table 6. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps for iCaRL, hybrid models and iCaRL2.

balanced and robust classifier that is also able to teach better logits to new configurations, further helping in the preservation of previous knowledge.

Indeed, the accuracy achieved by the balanced fully connected layer is similar to the one of NME, with only a small gap of 1-2% in the last batches, differently from hybrid1, whose performances start to collapse after the sixth batch up to having a 8% gap at the end.

Further improvements and experiments could involve the use of old fully connected layers only instead of the full network or limiting the number of teachers to the last k network states, reducing this way the amount of memory needed for the model.

Also a self adapting rule for λ should be investigated taking into account, for example, the number of classes observed and the number of new classes that should be learnt, similarly to what is proposed in [10].

5. Ablation Study II

In order to show the effectiveness of each component of iCaRL2, similarly to what has been done in the original paper, we report the results of different hybrid configurations, starting from the base iCaRL model and incrementally adding for each configuration one of our improvements arriving to the final model:

- "new_decay" refers to the original iCaRL model with the addition of the decreasing policy for the weight decay;
- "new_loss" refers to the previous configuration with the addition our loss function;
- "iCaRL2" refers instead to the full model with stabilization phase.

As we can easily notice from Figure 6 each configuration achieves slightly better scores than the previous one, proving the contribution of each different component to the final result.

The addition of the decreasing policy for the weight decay alone enables the model to be more precise during the first incremental steps but in the last ones the accuracy drops

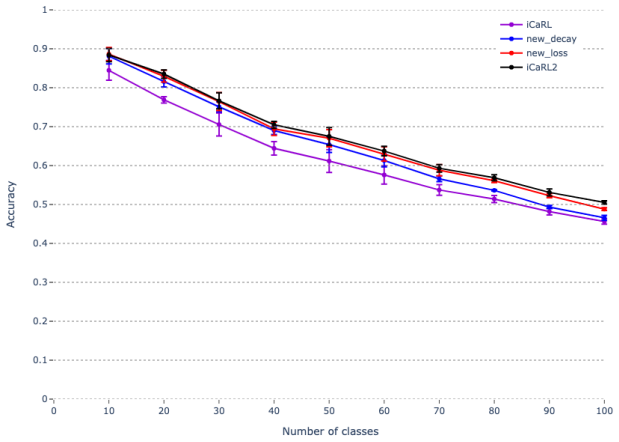

Figure 6. Multi-class accuracy across all classes seen up to a certain point and average multi-class accuracy across all incremental steps for iCaRL, iCaRL2 and hybrid models.

to iCaRL's scores. Only with the newly introduced loss the network is able to outscore the former model also in the last steps, and thanks to the stabilization phase it gains an additional 1-2% of accuracy.

As noticeable from Figure 6 and Table 6, the final version of our model, iCaRL2, outperforms the base iCaRL across all steps up to achieving at time t higher accuracies than iCaRL at *t-1*.

6. Conclusions

After the analysis of iCaRL in which we also explored the possibility of using different types of losses and classifiers, we introduced an evolution of this technique, iCaRL2, that, despite the drawback of the necessity of storing one version of the network for each incremental step, is able to outperform not only the former model but also End-to-End Incremental Learning [9] which has been the first source of inspiration for our improvements and that, together with iCaRL, is among the state-of-the-art techniques for Incremental Learning.

References

[1] Zhizhong Li, Derek Hoiem, *Learning without Forgetting*, 2017

[2] Rebuffi et al, *iCaRL: Incremental Classifier and Representation Learning*, 2017

[3] C. Wu et al, *Memory Replay GANs: learning to generate images from new categories without forgetting*, 2018

[4] H. Shin et al, *Continual learning with Deep Generative Replay*, 2017

[5] K. Javed et F. Shafait, *Revisiting Distillation and Incremental Classifier Learning*, 2019

[6] L.Guo et al, *Exemplar-Supported Representation for Effective Class-Incremental Learning*, 2020

[7] U. Michieli et al, *Knowledge Distillation for Incremental Learning in Semantic Segmentation*, 2020

[8] H. Jung et al, *Less-forgetting Learning in Deep Neural Networks*, 2016

[9] Castro et al, *End-to-End Incremental Learning*, 2018

[10] Y. Wu et al, *Large Scale Incremental Learning*, 2019

[11] G. Hinton et al, *Distilling the Knowledge in a Neural Network*, 2015

[12] J. Zhang et al, *Class-incremental Learning via Deep Model Consolidation*, 2020

[13] K. Shmelkov et al, *Incremental Learning of Object Detectors without Catastrophic Forgetting*, 2017