Incremental Learning Project

Armando La Rocca
Politecnico di Torino
armando.larocca97@gmail.com
s279401@studenti.polito.it

Marco Ratta
Politecnico di Torino
marcoratta97@gmail.com
s279637@studenti.polito.it

Abstract

Incremental learning is one of the most challenging open problems in image recognition. It consists in creating a multiclass classifier which can learn new classes incrementally with the only constraints of having a limited amount of storage memory and receiving the novel classes few at a time. The main problem related to Incremental learning is catastrophic forgetting which is the phenomenon by which the network forgets the previous classes when new ones are given to be learned.
It's been proven that classical training algorithms applied on CNN fail to keep the information previously acquired if they are not available anymore and that's why research is so active in this field.
In order to prevent catastrophic forgetting some strategies have been adopted over time such Learning Without Forgetting (LWF) and Icarl (state of art), which base the most of their improvements in the use of a distillation loss which aims to encourage the network to be similar to its non updated version.
In this project we investigate both these approaches and we try to give our contribution to the task.

1. Introduction

In the first part of the paper some algorithms that try to solve this problem are analyzed. In particular are the Fine tuning approach, Learning without forgetting and Icarl. After a brief introduction to this methods in Section 5 our implementation with the obtained results is reported.
The second part of the paper consists of two ablation studies using the Icarl algorithm as starting point. The first study is related to the losses, while the second one is focused on the classifiers.
In the last part some changes that we consider interesting to analyze are proposed, in order to try to improve the results. The first is related to the hearding step, the second is based on a possible improvement of the distillation part, the third is an interpretation of the end-to-end approach, while the last one is a modification of the NMC. [2]
The code related to the project is available at the following repository:
`https://github.com/armando-larocca/Project-IL-`.

2. Comparable methods

In this section is provided a broad description of the three considered approaches for incremental learning. The main goal for all of them is avoiding catastrophic forgetting during the training of the new classes.
Fine Tuning
The main idea of this algorithm is making a training sequentially to the new tasks. There are different approaches that can be adopted to avoid the forgetting, as, for example, freezing the specific parameters related to the old tasks and train only the parameters related to the new task and fine-tuning the shared ones. In our implementation we chose to train the model on the new classes without freezing parameters but fine-tuning all the network and adding specific parameters to the new task. This approach is indeed not efficient because we don't take any measure to avoid the forgetting, but this is mainly a way to demonstrate the effects of a more traditional approach and to underline the differences with the other methods.
Learning without forgetting
This approach is based on the preservation of the old tasks and the optimal learning of the new ones using only samples of the new task. This is possible using the concept of knowledge distillation which tries to preserve the output related to the old tasks without losing efficiency in learning the new ones. [4]
Icarl
This approach follows the same suggestion introduced by LwF but adds the concept of features and exemplars. Basically, this algorithm stores a fixed number of images for each old class using the hearding principle. This images are then used for both making a classification with a Nearest Mean classifier after being processed by a feature extractor and for the training step. [5]

3. Learning without Forgetting (LwF)

The main goal of this algorithm, as the name suggests, is to learn new tasks without forgetting the old ones and also without using data of previous tasks. This approach increases the performances with respect to the fine tuning and also provides good computational performances.
Considering a CNN with shared parameters θ_s and task specific θ_o we want to add and learn new specific task parameters θ_n. The dimension of θ_n is equal to the dimension of the new classes that we want to learn while the dimension θ_o corresponds to the number of classes learned so far. So we consider a training set X_n, Y_n related to the new task.
We can consider two different steps:

1. Initialization
2. Training

3.1. Initialization

In this first step the outputs of the new task's images related to the old network Y_o are recorded. This is necessary because, as we previously said, the model is trained only on the new task images but we want also to avoid the catastrophic forgetting. In this sense the concept of knowledge distillation is used with the old network. This concept was proposed for a different usage but is a useful tool which can be adapted to this problem.[3]
Then the CNN is modified by adding the parameters θ_o randomly initialized.

3.2. Training

In the second phase from the new task images X_n the outputs related to the old tasks $\hat{Y_o}$ and the new one $\hat{Y_n}$ are extracted. To avoid the catastrophic forgetting two losses are used i.e. the classification loss which trains the network on the new task and the distillation loss that tries to fit the parameters to the old network. The overall loss is specified below :

$$L(\theta) = \lambda_o L_{dist}(Y_o, \hat{Y_o}) + L_{class}(Y_n, \hat{Y_n})$$

where L_{dist} represents the distillation loss and L_{class} is the classification loss.
λ_0 is a loss balanced weight that is used to increase the contribute of the distillation loss.
In the original paper the proposed classification loss is CrossEntropy loss and the distillation one is a modified cross entropy with temperature that is described in the Section 6.3. However, in this case we used the loss proposed by the Icarl paper that is described in the Section 4.1 in order to make a fair comparison among all the methods.
Then the network is trained minimizing the proposed loss and updating the parameters.

4. Icarl

In this section we analyze the algorithms involved in Icarl step by step, with emphasys on the parts which are new with respect to the other state of art approaches.
Let's suppose we have images belonging to N different classes and we want to learn to recognize these images incrementally (which means we have only images of few classes available at the same time), in this case a complete Icarl step requires 3 different steps:

1. Training
2. Test
3. Construction of exemplars

4.1. Training

The first part of Icarl approach consists in the training of the CNN. Differently from the fine-tuning approach, the images provided for Icarl training belong both to old and new classes; in particular the maximum number of storable images belonging to old classes, which are named "Exemplars" is fixed a priori. So the current training set is composed by all images belonging to new classes (which are the ones that are available at this time) plus a little amount of old exemplars which are needed, in order to avoid the already mentioned catastrophic forgetting.
For the same reason the loss used is also different and it's formed by the sum of 2 different losses which are a classical CrossEntropyLoss for classification and a new component as explained in Section 3.2. The overall loss which is used to update the gradients of the CNN takes this form:

$$l(\theta) = -\sum_{(x_i,y_i)\in\mathcal{D}} \Bigg[\sum_{y=j}^{t} \delta_{y=y_i} \log g_y(x_i) + \delta_{y\neq y_i} \log(1 - g_y(x_i)) + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1 - q_i^y) \log(1 - g_y(x_i)) \Bigg] \quad (1)$$

For every couple of image-label (x_i, y_i) in the training set $\mathcal{D}$ two losses are calculated; the first one is the cross entropy classification loss calculated only on the new classes (labeled in the formula from s to t), the second one is a binary crossentropy loss calculated only for the old classes (labeled in the formula from 1 to $s-1$).

4.2. Construction of Exemplars

Once the training procedure is complete, it's reasonable to think that in a real scenario the images belonging to the current tasks would not be available anymore; which would bring our algorithm to forget those classes in future.Icarl

deals with this problem by storing a few exemplars of each learned class.
In order to decide which images to store, Icarl provides a sort of herding procedure, which aims to select only those images which better represents the class in terms of features.
In particular for each already learned class a feature class-mean is computed as follows

$$\mu_y = \frac{1}{\mid P_y \mid} \sum_{p \in P_y} \phi(p)$$

Then for each image belonging to the class y the features $\phi(x_i)$ are computed. The k-th exemplar is selected by minimizing the difference between the feature class-mean μ_y and the average of the feature vectors of the $k-1$ already stored exemplars plus the feature vector of the current image $\phi(x_i)$. Among all the images x_i the one which minimizes the quantity described above is chosen. The procedure is shown below:

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

The exemplars are now ordered with a priority, which means that p_1 corresponds to the image which better represents the class in terms of features.
This is a good point because when new classes are learned and we need to reduce the exemplar set of each class, due to the computational storage limits, it will be sufficient to cut the vector to its first m terms and we can be sure that the exemplar set is still representative of the class.

4.3. Test

After each session of training, the CNN is tested on a test set which has the same number of exemplars for each learned class. Differently from other approaches as LWF, the classification method is not based on the use of the outputs of the FC layers but it's based on the outputs of the feature extractor layers. The method provided by Icarl is called "Nearest Mean of Exemplar" (NME) and it's described as follows.
For each learned class - as we said before - only a few number of exemplars are stored; for each of those exemplars the output of the feature extractor is computed and, by averaging the feature vectors obtained, a class-mean is calculated.
The herding procedure applied in the construction of the exemplar set ensures us that, no matter how few are the exemplars, the class-mean is actually representative. The calculation of the feature class-mean for the class y is shown below:

$$\mu_y = \frac{1}{\mid P_y \mid} \sum_{p \in P_y} \phi(p)$$

Made this procedure for every already learned class, we have a feature-mean vector:

$$\mu = \{\mu_1, \mu_2, \dots, \mu_s\}$$

Now, for each image in the test set the feature vector $\phi(x)$ is computed and the norm of the difference between $\phi(x)$ and each component μ is computed. The label y^* assigned to the image corresponds to the one that minimizes the norm of the error. The procedure is shown below:

$$y^* = \arg\min_{y=1 \dots t} \|\phi(x) - \mu_y\|$$

5. Experiments

5.1. Cifar100

For the following experiments the dataset Cifar100 is used. It contains 60000 images of dimension 32x32x3 that are divided in 100 classes. For each class there are 500 images for the training and 100 for the test.
With regards to the incremental approach, the dataset is randomly splitted in 10 batches composed by 10 classes that are serially used.
The test set is composed in a similar way, but joining the previous batch with the new one with an incremental approach. Hence at the end of the incremental training the network is evaluated on all the test set.

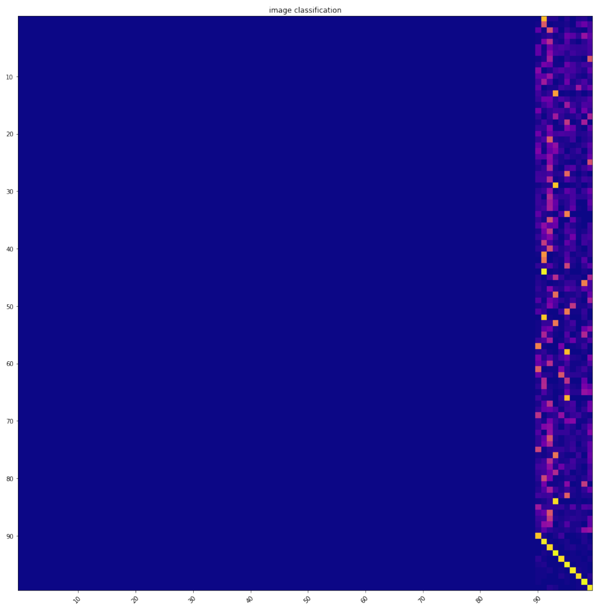
5.2. Architecture

Our model is composed of two main components which are the feature extractor and the fully connected layer.
The feature extractor is a ResNet-32 adapted for a 32x32x3 without the last layer that is the FCL. This part of the architecture is used to obtain the features that have a 64x1x1 dimension.
The last part of the architecture is composed by a FCL that has as many outputs as the number of the classes. In contrast with the Icarl implementation, we chose to adopt an incremental approach even considering the architecture. In this sense for each incremental step the number of outputs in the fully connected is updated by adding, in our case, ten more outputs for each step. This choice probably affected our results but we preferred to adopt a more general approach because in the largest part of the cases when we consider an incremental approach we don't know a-priori the total number of the classes.

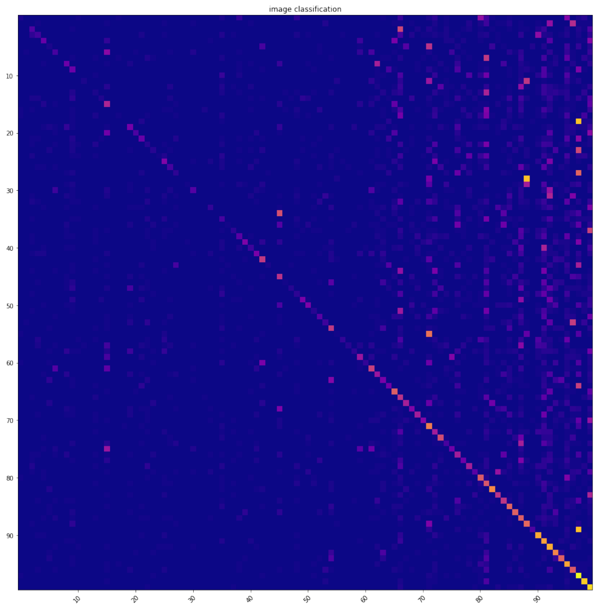
5.3. Configuration

For all the experiments the images are normalized with the CIFAR100 mean and standard deviation:
mean= $(0.5071, 0.4867, 0.4408)$
standard deviation = $(0.2675, 0.2565, 0.2761)$

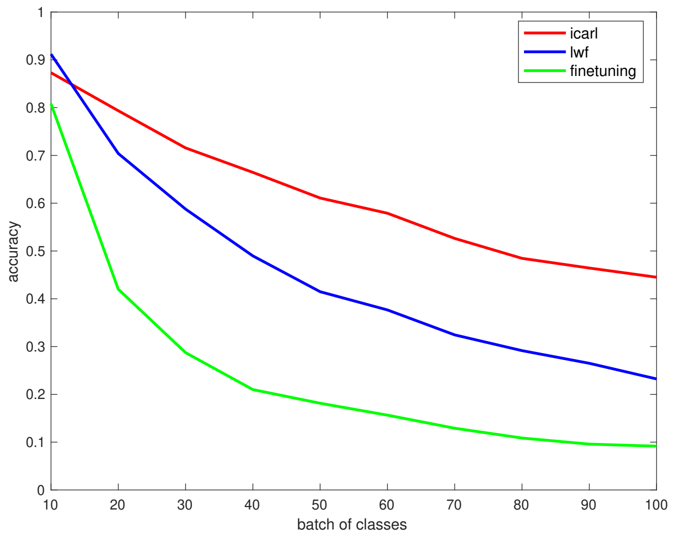
Data augmentation is applied, it is specifically used a Random Crop and an Horizontal flip with 50% probability. The model is trained with stochastic gradient descent with mini-batches of 128 samples.
In this section Lwf is implemented , using an incremental approach with the loss proposed in Section 4.1 while for fine-tuning a Cross Entropy loss is used. In all these experiments the model is trained for 70 epochs considering a multiple step size of 49 and 63 with a multiplicative factor of 0.2. The weight decay is 0.0001 while the learning rate used was of 2 in LwF and Icarl while 0.2 with fine tuning.



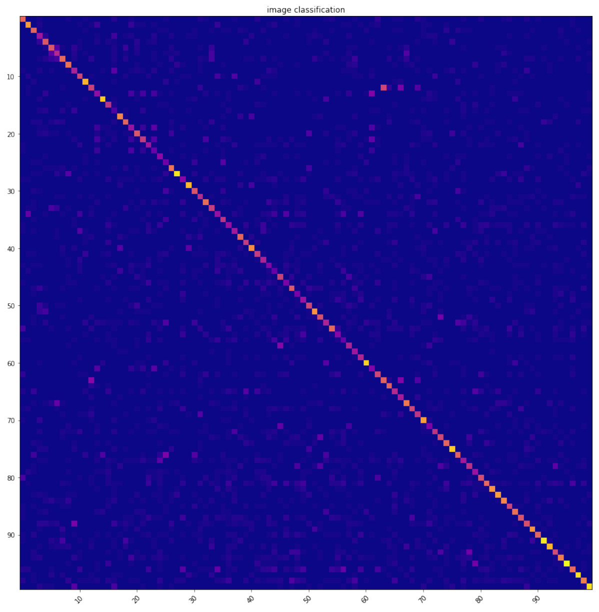
(a) Fine Tuning



(b) LwF



(a) Baselines Test accuracy



(b) Icarl

Figure 2: These figures represent the different results obtained implementing the baselines

5.4. Results

The results obtained with this algorithms underline that a traditional training approach as the one proposed in the fine tuning cannot solve the problem of catastrophic forgetting. In contrast, the distillation concept introduced in Lwf gives great improvements and Icarl, which combines the distillation with the exemplars and the NCM, is the one that reaches the best results.
The results obtained with LwF and FT are aligned with

the ones proposed in the Icarl paper, but with the Icarl algorithm the results are lower of few percentage points. A possible motivation could be the fact that, in Icarl original implementation, the incremental approach is not followed as FC has always 100 outputs and the hyperparameters are optimized for this configuration.

Average accuracy		
Icarl	LwF	FT
61.57	45.98	24.88

6. Loss ablation study

In this section we provide an ablation study based on 3 new different choices of losses applied on Icarl. The goal of this study is possibly finding a better performing loss or alternatively confirm that the choice made in Icarl is actually the best performing one.
For each loss the mathematical formula is provided with a brief explanation; furthermore the reasons of our choices are presented with respect to the improvements or the behaviour we expect.

6.1. Cross Entropy + BCE

Description
For this first attempt we choose a Cross Entropy Loss for classification and a Binary Cross Entropy Loss for the distillation. The reason why we decided to change the classification Loss is basically due to the fact that the cross entropy loss is known in literature as one of the best performing losses for multiclass classification.
The formula of the cross entropy loss is the following:

$$Class(\theta) = -\sum_{c=1}^{M} \delta_{i,c} \log(p_{i,c})$$

where M is the total number of classes, $\delta_{i,c}$ is a binary index which goes to one if the predicted label of the observation i is c, and $p_{i,c}$ is the probability that the observation i is of class c, which comes from the $c-th$ output of the FC layer.
The main difference between the BCE provided in Icarl and the CE used here is that in this case we are considering not only a classification on new classes but also a classification among old classes. This choice has been made in order to reinforce the already acquired knowledge on old classes.
For the distillation loss we use the same BCE as Icarl, so

$$Dist = \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1-q_i^y)\log(1-g_y(x_i))$$

The main issue related to this choice, coupled with the classification CE, is due to the learning rate. In fact the Cross Entropy wants smaller LR to have good performances, while the BCE wants higher ones. After some attempts, our trade-off made us decide for LR=0.1, which is the one we used for our experiments.

Results
Analyzing the accuracy on our experiments we can state that this choice offers worst results compared with the ones obtained with Icarl.
Especially from the confusion matrix, we can state that the problem does not involve the classification loss, which gives us good performances but in the distillation.
The algorithm worse recognizes old classes with respect to Icarl and this can be related to the trade-off made on the learning rate. As we previously said, the two losses we have used have different formulation and, because of that, they need different LR. We suppose that setting a LR in order to make both losses work good enough, make us difficult to make them perform individually at their best and that's why our performances give us no improvements, although they are somehow comparable with Icarl.

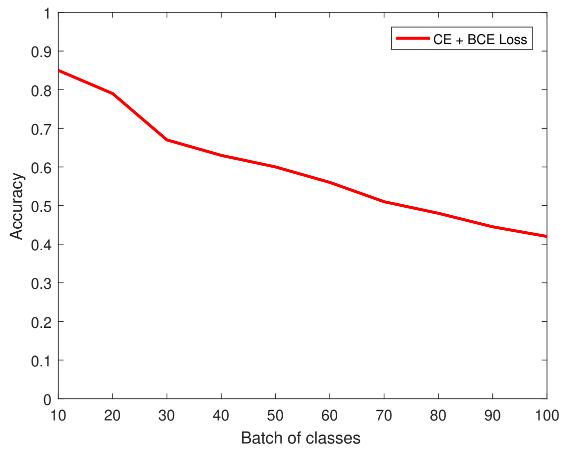


Figure 3: Test accuracy CE+BCE

6.2. Cross Entropy + L2 Loss

Description
For this attempt we keep a Cross Entropy Loss for classification, as it gave us good results in the previous analysis, while for distillation we follow a different approach.
The loss we use is a L2 Loss calculated on the features at the previous step. Calling $\phi^*(x)$ the features calculated with the CNN at the previous step and $\phi(x)$ the features calculated with the updated CNN the expression is the following:

$$L^2(\theta) = \|\phi^*(x) - \phi(x)\|_{L^2}$$

The reason why we tried this approach is related to the fact that our classification is based on the output of the feature extractor. In Icarl the assumption made during distillation is that the feature extractor is trained jointly with the FC, in the sense that minimizing the distance between outputs of old and new CNN implies directly minimizing the distance between the outputs of the 2 feature extractors. The approach is reasonable; this attempt though aims to train the feature extractor, at least in the distillation part, by minimizing its ”own” loss which means minimizing directly the L^2 norm of the difference vector.

Results

From the experiments made with this loss we obtain results comparable with our version of Icarl. Both the trend of the accuracy and the absolute results are similar to the ones obtained by Icarl, which means the choice is consistent.

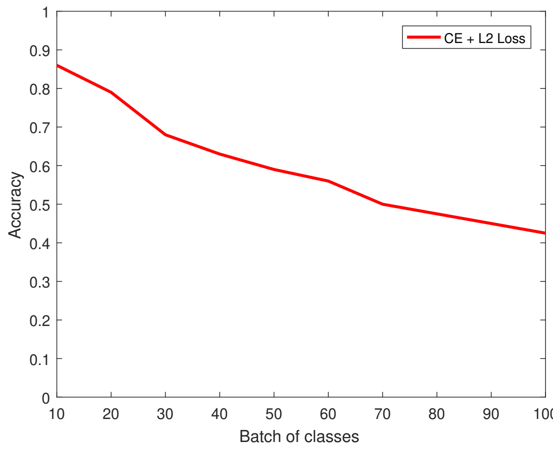


Figure 4: Test accuracy L2

6.3. Cross Entropy + Softmax with Temperture

Description

For this analysis we keep the cross entropy loss as classification loss and we use a new distillation loss called ”Softmax with Temperature”.

This procedure consists in bringing the outputs of the fully connected layers, passing them through a softmax - which returns for each neuron a logit corresponding to the probability that an image belongs to the corresponding class - and computing the loss on these normalized logits.

The formula of the losses is

$$SwtLoss = \sum_{y=1}^{s-1} k_y^*(x_i) \log g_y^*(x_i)$$

where $k^*(x_i)$ and $g_y^*(x_i)$ are respectively the outputs of the old and the current networks modified as follow:

$$k_{(j)}^*(x_i) = \frac{(k_{(j)}(x_i))^{1/T}}{\sum_{h=1}^{s-1}(k_{(h)}(x_i))^{1/T}}$$

$$g_{(j)}^*(x_i) = \frac{(g_{(j)}(x_i))^{1/T}}{\sum_{h=1}^{s-1}(g_{(h)}(x_i))^{1/T}}$$

We notice from the expression that actually this loss is a modified cross-entropy loss, which increases the weight for smaller probabilities depending on the value of T which is called temperature and which represents an hyperparameter to tune. The greater is T, the more the lowest logits will count in the computation of the loss, and this should encourage the network to better encode similarities among the classes.

This approach is the one used also in LWF and that’s the main reason we decided to test it in our task.

As written in the LWF paper we set the temperature to 2, as is the one which better performs after a gridsearch.

Results

As expected, the loss offers good performances, and the accuracy plot shows us that the trend is comparable with the one of Icarl. More experiments have been done also to test the loss with different values of the hyperparameter T but the results haven’t changed much.

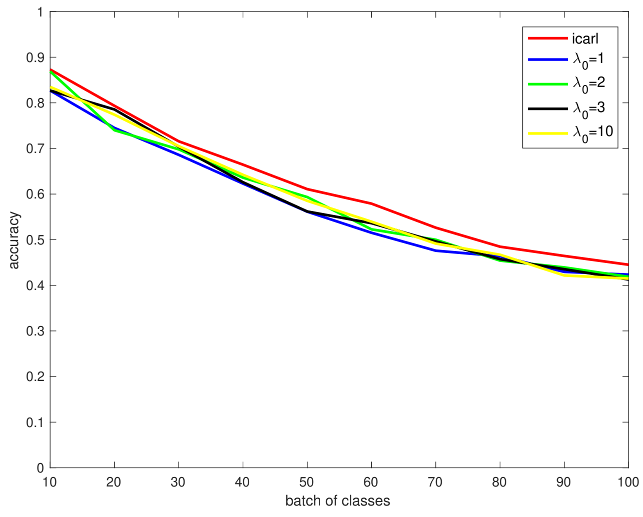


Figure 5: Test Accuracy CE+SWT

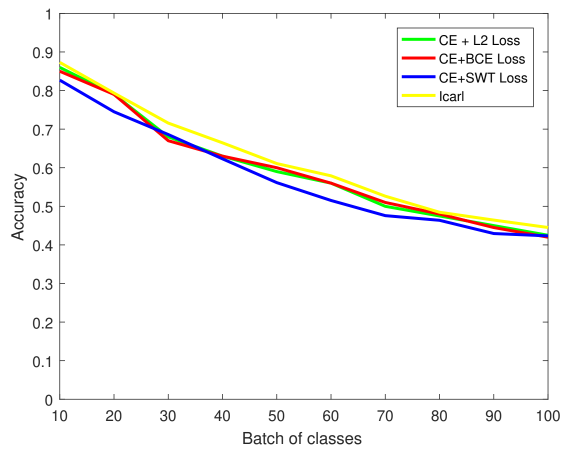


Figure 6: Losses Comparison

Average accuracy			
Icarl	CE+BCE	CE+L2	CE+SWT
61.57	59.55	59.60	57.50

7. Classifier ablation study

In Icarl the classifier takes as input the features obtained using the feature extractor on the exemplars that are chosen using the hearding method. The choice of the right classifier is very important for the performances of the model.
As we previously said the number of exemplars is constant and fixed (K=2000) and the number of exemplars for each class is obtained dividing them for the number classes learned so far. For this reason the dataset is balanced anyway we have to consider that the higher is the number of classes the lower is the number of exemplars for each class, which is equivalent to a lower number of training samples for each class.
Another important aspect when we consider a classifier is the dimension of the input, that in our case corresponds to the dimension of the feature that is 64x1x1.
Facing these two issues brings the choice of classifier towards the more general problem of curse of dimensionality.
In Icarl the proposed classifier is the nearest-mean classifier but in this section the use of three other classifiers is experienced and for each one results are analyzed. For this ablation study the architecture, the loss and the hyperparameters proposed in the section 1 are used.

7.1. K Nearest Neighbors

Description
Basically, the KNN classifier works assigning to the input that we want to classify the label corresponding to the most frequent class considering only the K nearest labeled samples.
Assuming that each class distribution is dense in the feature space, we have chosen the KNN classifier, as we expect that the features related to images of the same class will be close in the feature space.
However we have also to remind that the KNN performs better when we have an high number of samples for each class and, as in this case the number decreases with the introduction of new classes, it may fail with the decrease of the exemplars. Another weak point of this choice is related to the high dimensional feature space because KNN is based on the distance, and in an high dimensional space this can affect the efficiency.
Experiments
For our experiments we used two values of K. This hyperparameter is used to set the number of neighbours considered for the classification and it is very important to set the sensibility to the noise in the data.

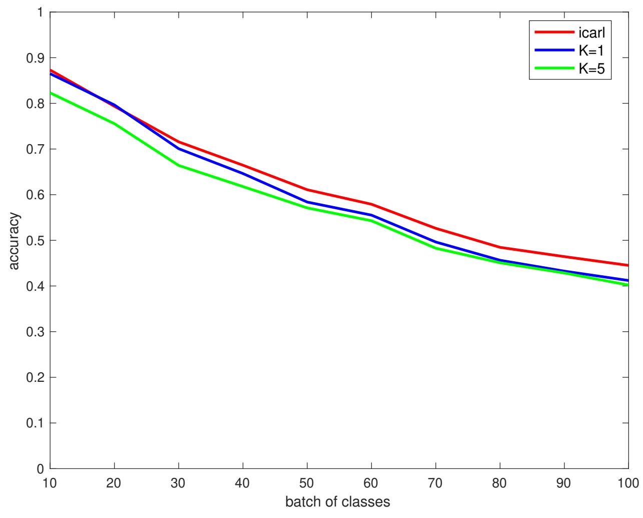


Figure 7: KNN Test accuracy

7.2. Support Vector Machine

Description
Overall, the SVM finds hyperplanes that separate the space in order to divide labeled samples of different classes of the dataset.
Choosing the SVM we made a similar analysis of the KNN but in this case we consider that SVM is less affected by the problem of the reduction of the samples because the choice of the hyperplanes is made considering a subset of features. Another advantage is that SVM should have good performances for high dimensional inputs.
The negative aspect is that the tuning of the parameters and a good choice of the kernel is fundamental to avoid the

overfitting and in this case the tuning is a very expensive operation.

Experiments In our experiments we chose to use a linear kernel and an rbf kernel. For each kernel two different values of the hyperparameter C were tried. With regard to the rbf kernel the scaled value of γ was used, it can be expressed in the following way:

$$\gamma = 1/(N * Var(X))$$

where X is the input and N is the number of samples.

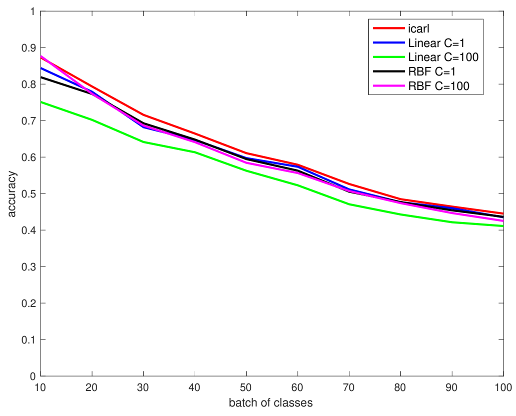


Figure 8: SVM Test accuracy

7.3. Multi Layer Perceptron

Description
The MLP is a supervised algorithm that is capable to learn a function that links the input and the output data, thanks to a training procedure on labeled data.
The main reason using MLP is that, as it doesn't make any priori assumption on the output function it should be able to stochastically approximate even very complex problems. Some weak points are the decreasing number of samples per class and the high dimensionality of the data, already mentioned above.
Experiments
In our experiments we have chosen to use two different optimizers that are SGD and Adam. Our classifier has only one hidden layer with 100 nodes. With Adam the default learning rate 0.001 was used, while for SGD a LR of 0.01 was selected. In both cases was used ReLu as activation function.

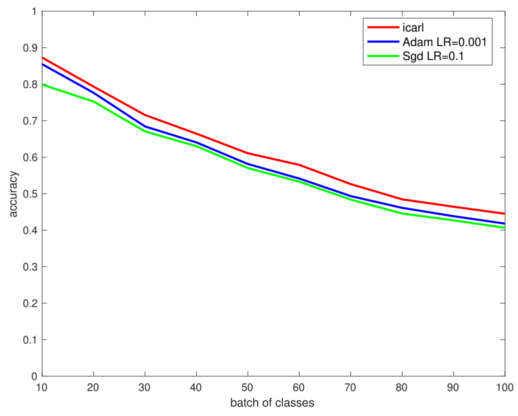


Figure 9: MLP test accuracy

7.4. Results

Overall, all the classifiers had lower performances than the NMC proposed by Icarl. In the following graphs we show a comparison between the best results given by all the combinations of hyperparameters tried on each classifier.
The best one was the SVM with RBF kernel and C=1; furthermore from the figure12 it can be seen that the RBF kernel gives slightly better performances than the linear, and that using C=1 offers better results.
Multi layer perceptron and KNN have very similar trends. The first one gives the same trend for both the optimizers used. However in this case we have to consider that there are many other combinations of hyperparameters that should be investigate.
With regard the KNN with K=1 the performances are a little bit higher than with K=5 but both have similar trends.

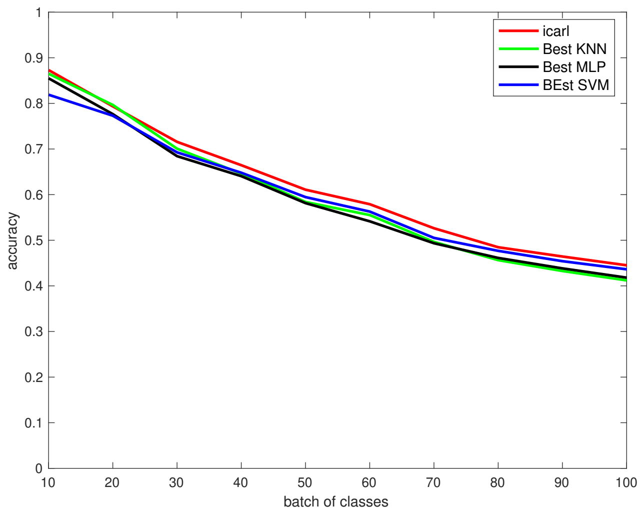


Figure 10: Classifiers' comparison

Average accuracy				
Icarl	KNN	SVM	MLP	Icarl HB1
61.57	59.45	59.62	58.90	56.81

8. Our implementation

8.1. Old Networks Memory

Description

As we have explained in the previous sections both Icarl and LWF make use of a distillation loss in order to avoid catastrophic forgetting.
The base concept adopted is that, beyond classification loss which is responsible of the weights update, a second term is added to the overall loss in order to make the network differ as less as possible from the previous one. The two terms are in some way in competition and the second one can be seen as a regularization term, made with the purpose of not letting the classification be too dominant.
The fact that distillation is made on the net at the previous step is suboptimal from our point of view, because potentially, as the algorithm forgets (not catastrophically, but still does), after a certain number of training iterations, using the net at the just previous step can become a bad approximation, mostly with respect to the first classes more. This concept is related to a propagation of the forgetting error, which we briefly explain using a real scenario.
Let's imagine that already 80 classes are learned and the current network is able to predict the images of the first 10 classes with a 30% accuracy, if we use this network for the distillation loss of the 6th decine training, we are comparing, for the first 10 classes, the current outputs with the ones of a pretty bad network so our target is a net which gives bad prediction and which is in a certain way outdated. This error propagates over time an it can be potentially dangerous if we suppose to have many many classes.

Implementation

Our idea consists in saving the network at each training iteration, and use all the N-1 saved networks as a reference for the new network. In particular the first network stored, which will correspond to the 10 output network trained on the first 10 classes, will be used as target for distillation for the first 10 neurons of the new model and so on. We think that using this first network as reference for the first 10 neurons training at each iterations can be better, as the outputs calculated on this network will be always the ones corresponding to the best predictions of the first 10 classes and cannot risk to become outdated.
One of the main problems in this procedure is related to the storage memory which, saving a net at each iteration can easily saturates. In order to do so we thought an easy algorithm in order to store a maximum of N networks in a smart way. Let's say we have 200 classes which we want to learn on batches of 10 classes, and we want to store a maximum of 10 networks; for the first 10 iterations we'll save the 10 networks, for the 11th step we delete the first network and we add the last one remaining with 10 networks. Now, the total number of current outputs, 110 in the case, are divided by 10 and the results will be the number of outputs that each stored net will have to calculate as target of the distillation loss. In this way the memory occupied remains the same and the predictions remains accurate enough.
Another problem is related to the lack of generality, in fact for example this approach can not be used with distillation losses which use features as L2 loss described above, that's because the dimension of features is fixed (in the case 64) and so the correspondence feature-class is not 1-1.

Experiments

We made experiments using this approach and the results obteined are comparable with our version of Icarl, benefits though are not evident, probably because our variation is made to be effective with a large amount of classes (actually a large number of training iterations). Anyway we consider this variation a possible improvement.

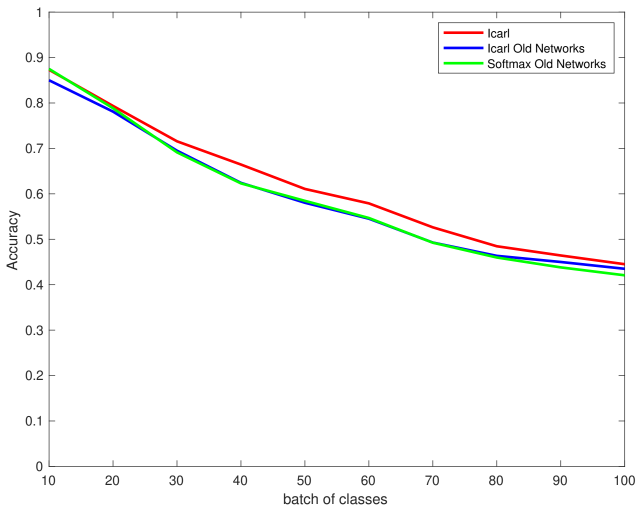


Figure 11: Old Networks Comparison

8.2. Updated Herding

Description
As we have seen in the previous paragraphs Icarl stores its previous classes knowledge in form of exemplars and the way Icarl picks the exemplars is regulated by an herding procedure which we have already described. Our modification consists in a variant of this herding procedure, which spots a problem on the reduction of the exemplar sets. We noticed that Icarl at the point it learns new classes reduces the exemplar sets by cutting the vector containing them at the first K/N elements, ensuring that the first elements of the vector are the most representative of the class in terms of features. The problem is that features are updated step by step during the consecutive trainings and the order of the exemplars in the set does not refer to the updated feature extractor but to the feature extractor of the old network, the one corresponding to the net updated at the time the exemplars were calculated. The exemplars in Icarl in fact are computed and stored just once, and then the order of the exemplar set remains the same. That is the point we have modified.

Implementation
For our implementation we construct the exemplars following the same protocol provided by Icarl, so choosing the ones corresponding the most to the mean of the current features for each class; but this procedure is not made only for new classes exemplars, on the contrary it affects in a way also the previous classes. For each training step, exemplars are built for the current classes and reordered for the previous ones in order to provide an updated herding based on an updated feature extractor. We are aware of the fact the features of a “new” network can be suboptimal with respect to the old ones for the old classes, but we think that for our purpose it’s better to choose updated exemplars because it should give benefits at the time of classification, as features correspond to the network which at the current step is actually classifying. The main issue related to this approach regards the time required for the runs. Cutting a vector is way faster than recompute the construction of the exemplar set; anyway, as the number of exemplar is fixed we are not really worried about that as the time required will not explode.

Experiments
Experiments using this herding procedure does not give evident improvements but we think it’s because the feature of images of each class are already pretty close to each other, so the herding is more a detail than a substancial change.
This observation is also proven by some experiments we have made picking random exemplars and getting competitive results as well.

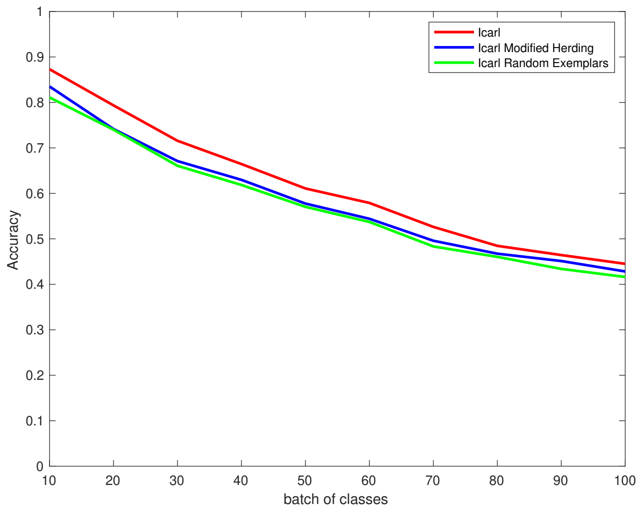


Figure 12: Exemplar Choice Comparison

8.3. End to end incremental learning

For the following proposal we take inspiration from the paper *End-to-End Incremental Learning*.[1]
For this approach the same architecture presented in the previous sections is used but we introduce the Cross distillation loss and the balance fine-tuning. As in ICarl the model is trained on the new tasks images and on the exemplars of the old classes.
Cross distillation loss
This loss is similar to the one proposed in the Section 6.3 but in this case it’s used in a different way.
There is a classification loss that takes all the outputs of the FC layer and a distillation loss that is the sum of the of N

Cross-entropy with temperature, where N is the number of the old tasks. In formula :

$$L(\theta) = L_{class}(\theta) + \sum_{i=1}^{N} L_{dist}^{(i)}(\theta)$$

where $L_{class}(\theta)$ is the Cross Entropy and $L_{dist}(\theta)$ is the Cross Entropy with the temperature referred to the i-esima class.

In this way each distillation loss related to the previous task has the same weight because each task is composed by the same number of classes. Even though as classification loss is a mean over all classes (so its magnitude doesn't increase) and distillation loss is the sum of many losses as old tasks its contribution to the overall loss increases.

Balance fine-tuning
This approach is based on the fact that the model is trained on a disproportionate dataset because the number of exemplars for each class is lower than the number of samples of the new task. In this case training a model with a classification loss on all the tasks benefit the learning of new classes.
A possible solution is making an additional training step considering a dataset obtained reducing the number of samples of the new classes, to balance this inequality. In this step to avoid the forgetting of the new classes we use a distillation loss on them while we use a Cross Entropy on all the outputs. A consideration is that the first batch of classes was considered as a traditional training and is performed without the Fine tuning step because the dataset it is trained on is already balanced due to the fact we have not exemplars.

Experiments
For the following experiments we considered some combinations of this two possibilities in order to better understand their efficiency.

Hybrid1 This model was obtained using the Cross-distillation loss on Icarl,a learning rate of 0.1 is the sum with a weight decay of 0.0001 and as Icarl was trained for 70 epochs.

Hybrid2 This model was obtained using the loss proposed by Icarl and with the Fine-Tuning step. For the fine-tuning was used a learning rate of 0.01 and a weight decay of 0.0001 while for the training step of 0.1. Then different combinations of epochs for both the steps and of weight decay are tried for the training step. We made a training step of 55 epochs and weight decay of 0.0001 and a BFT step of 30 epochs (Set1),than we tried a training step of 50 epochs and a BFT of 20 with a weight decay first of 0.0001 (Set2) and then of 0.00001 (Set3) for the first step.

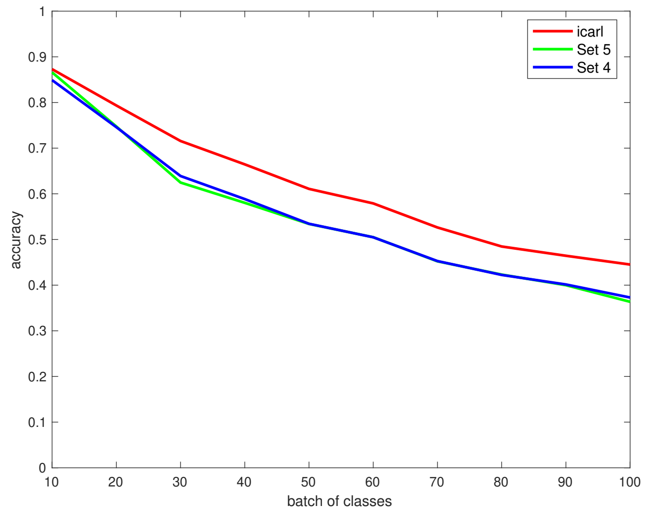


Figure 13: Hybrid 2 test accuracy

End-to-end approach This approach was quite similar to the one proposed in the paper. We used the Cross distillation loss for the first step and then fine tuning step. For the first step we used a learning rate of 0.1 while for the BFT of 0.01. For both the steps a weight decay of 0.0001 was used. The model was trained considering 40 epochs for the training step and 40 for BFT (Set4) and also 40 and 30 (Set 5)

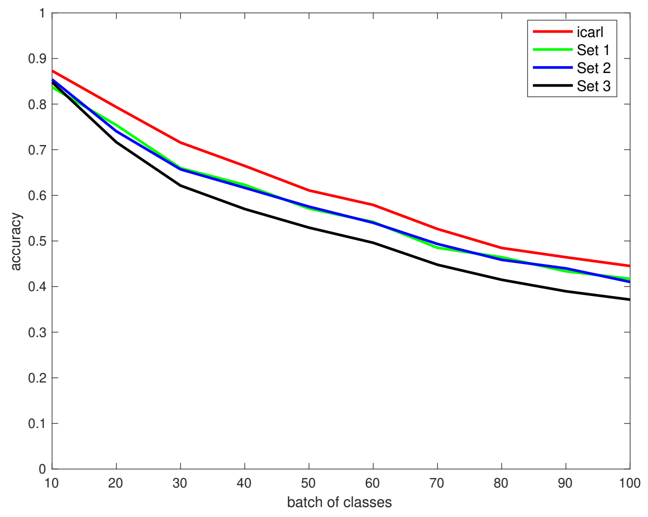


Figure 14: End-to-end test accuracy

Results
Overall, for what regards the hybrid2 it is clear that using an higher weight decay with the BFT approach improves

the results, while increasing the number of epochs doesn't improve the results and this is evident also in the end-to-end approach.
At the end, making a comparison, the Hybrid2 configuration close the training with similar values of Icarl. However the trend is quite lower and this means that the BFT has not a good effect on the algorithm.
The cross-distillation loss proposed had not a good impact on the performances of the model above all in the first batches. This can be explainable with the fact that the distillation loss increases its weight over the iterations and, probably in the first iterations its weight is not enough to balance the classification. Also in this case the balance fine-tuning doesn't provide any better relevant improvements.

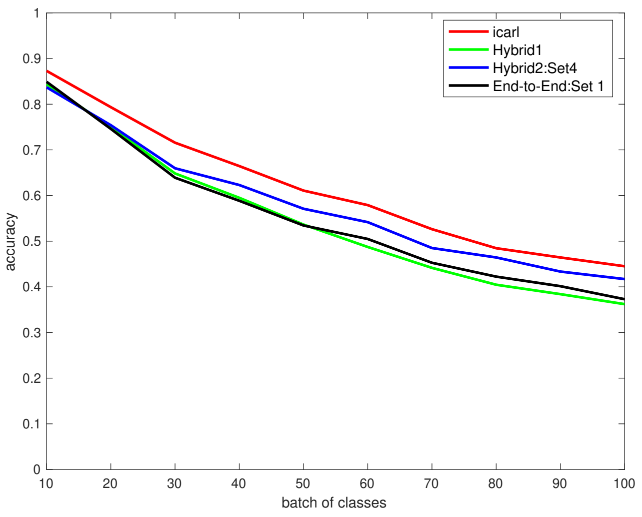


Figure 15: End-to-end comparison

8.4. Normal Distribution Based Classifier

Description

Icarl uses NME as classifier, which is an easy and very straight forward method to predict the label of an image using its features. Even though we think that this approach is not so accurate and may fail in some borderline cases. The features of every class follow a certain distribution and what Icarl does is basically condense all the informations of this distribution in its mean; this of course is not incorrect at all as the expected value is the first order approximation of a distribution but can be a too strong constraint.
Let's imagine for example that the classifier has to deal with two classes which have similar characteristics (let's say donkeys and horses for examples). What we can expect is that the mean of the features of these 2 classes is very close, so the discrimination can become a big problem and the classifier may fail.
Our improvement aims to generalize the NME classifier to be more effective in these situations, and in general to be more precise.

Implementation

In order to overcome this problems our idea is determining a distribution of probability for each class based on its features and using it instead of the mean for classification.
When the method Classify is called the mean of exemplars and the approximated covariance matrix are computed for each class and a gaussian 64 dimensions distribution is built by estimating its parameters with the ones above. Then when an image has to be classified its features are computed and the value of the distribution in that point is calculated. The label is chosen as the one whose distribution maximize this quantity.
One of the main problems using this approach is that potentially the features of images belonging to one class can be very close and the computation of the multivariate gaussian can be difficult, as it requires the inverse of the covariance matrix that with this setting is very close to be singular (or anyways is badly conditioned). Moreover, when the number of exemplars becomes lower than 64 the covariance matrix has to be estimated from a number of points lower than the dimension of the distributions and this also gives some singularity problems. We observed the effect of this problem in our first attempt which showed the sudden decrease of the accuracies after the 4th batch, which is also the step where exemplars stored for each class become less than 64.
In order to overcome both this problems we tried some strategies to regularize the variance of the distribution and prevent the bad conditioning of the covariance matrix.
In particular we adopted 3 different methods of shrinkage: the first by simply adding a term of extravariance on the diagonal of the matrix by summing the Identity matrix multiplied by a factor which we have optimized, the second by using the Ledoit Wolf algorithm provided by Scikit learn and the third using OAS algorithm which is a generalization of the previous one under the assumption of gaussian distributions. As we can see in Figure 18 even with regularization this approach fails with the basic settings of Icarl in particular in the last batches. The problem states in the way Icarl stores the exemplars. In fact using our approach we want the exemplars to be representative of the distribution, not to be the closest to the mean; in fact if this happens over time the distributions tend to collapse in their means and the variances tends to zero, which is bad for classification. For this reason we improved our algorithm by picking random exemplars (assuming that random choice is the best way of approximating a population).

Experiments

We make some experiments using the 3 different approaches. First of all we tried Ledoit Wolf algorithm which gave us bad results, then we tried OAS algorithm which gave us relevant improvements in terms of accuracies and which is just 1/2 percentage points under Icarl and then we tried the diagonal approach optimizing the parameter λ over the set $[0.001, 0.01, 0.1]$. In particular with the optimum value of lambda we reaches the same performances of Icarl. The graphs shows the results of our experiments.

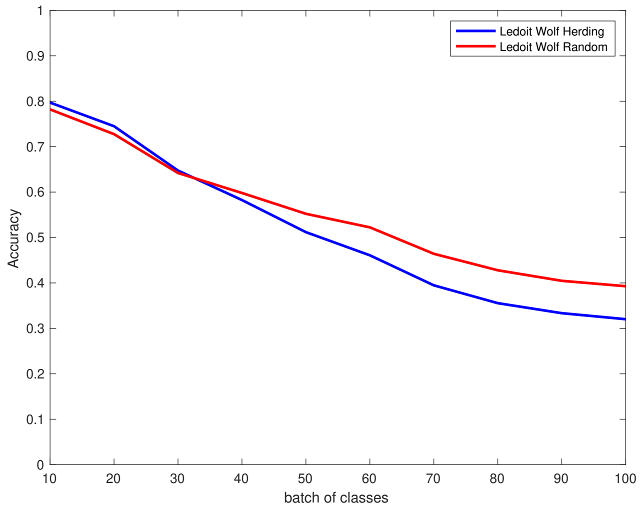


Figure 16: Herding VS Random Exemplars

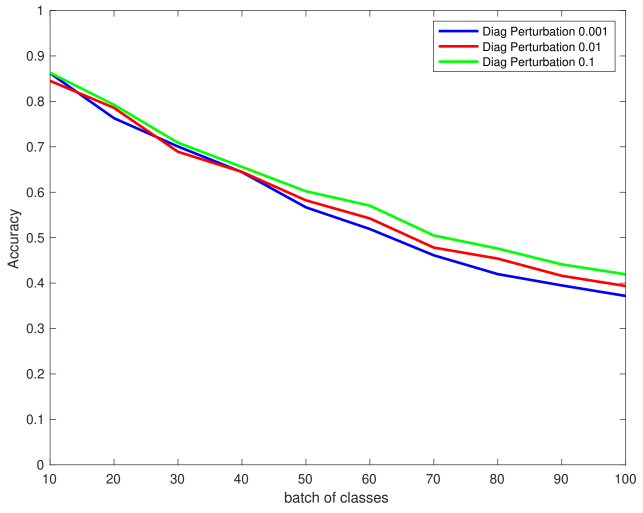


Figure 17: Diagonal Perturbation Tuning

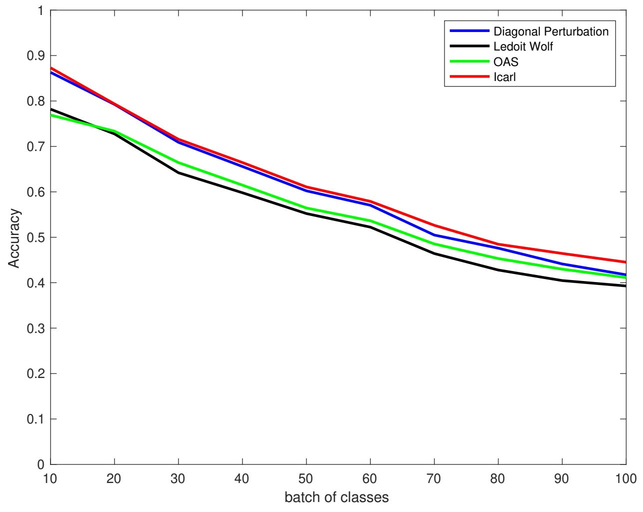


Figure 18: Variance Optimization Comparison

Average accuracy				
Icarl	Old Net	New Heard.	End-to-end	NDBC
61.57	59.17	58.42	55.11	60.34

9. Conclusions

From our work we have proven that Icarl's results are still the most competitive in the incremental learning scenario.

Changing losses and the classifiers didn't give us improvements and the couple NME, BCEWithLogits loss is still the most effective in terms of efficiency.

Our new implementations didn't offer any improvement either and we don't think any of them can actually make the difference in the task.

The analysis made on NDBC though opened us some interesting perspective for possible future deepenings. Especially we think that one promising way which can be pursued is developing a new features-based distillation loss discriminating, for each class, on the features with smallest variance (as they are also the ones which better encode the class).

Of course there are a lot of other interesting approaches which could be validated and analyzed but they all require a consistent amount of time; that's why we had to choose only few to present, discarding for example some ideas based on changing the architecture of the CNN or going deeper in the curse of dimensionality problem.

Anyway this project offered us a good opportunity to understand the complexity of the topic and projected us in a possible research context.

We enjoyed this aspect and we think it could be interesting to keep on investigating on this field.

References

[1] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. *CoRR*, abs/1807.09536, 2018.

[2] S. Guerriero, B. Caputo, and T. Mensink. Deepncm: Deep nearest class mean classifiers. In *International Conference on Learning Representations - Workshop (ICLRw)*, 2018.

[3] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[4] Z. Li and D. Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016.

[5] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.