Incremental Learning in Image Classification

Daniele Genta
s276922@studenti.polito.it

Davide Massimino
s274023@studenti.polito.it

Christian Paesante
s270102@studenti.polito.it

Politecnico di Torino

Abstract

An open research problem, in Deep Learning, arises when a large training data-set isn't available in advance yet the system is challenged with continuous stream of data that has to be used to train the model properly.
In this project, after having obtained an Upper Bound through Joint Training, we aim to present the main techniques that are used to address incremental learning, such as: Fine Tuning, Learning Without Forgetting and iCaRL.
Finally, after having examined the limitations of the previously mentioned methods, we try to overcome them with our contribution in order to propose an improved model.
The experiments have been carried out on the CIFAR100 data-set, under an Incremental setting.

1. Introduction

In a real world scenario, many problems are incremental. Basically, we'd like to replicate humans' learning, which consists in not forgetting previously learnt concepts when presented with new ones. This kind of problem is particularly important in computer vision and image classification in a *multi-class* setting.
So, we seek that our model is capable of being trained continuously on new data, Figure 1, instead of being trained on a large data-set offline once (this, instead, will be the upper bound for our experiments).
We'd ideally like to have is a model able to learn new classes while preserving the knowledge, i.e. parameters, from the old ones.
So, in an Incremental Learning setting, different classes occur at different times and the model should be able to provide a competitive multi-class classifier on the classes seen so far with a bounded amount of computational requirements and memory footprint, this means that limited or no access to previous data is granted.

We have compared different relevant approaches for Incremental Learning on *Cifar100*.
The first naive approach to address Incremental Learning might be to operate *Fine Tuning* of the model on unseen data. Unfortunately, this strategy is not optimal and ends up in a drop of the accuracy of the model (this scenario is commonly referred as *catastrophic forgetting* [4]). In order to address this issue, the literature has proposed different approaches over the years.
In this paper, we implemented [1] the *Learning without Forgetting* [3] and *iCaRL* [6] methodologies, both of them try to preserve the knowledge of the previously seen data through *knowledge distillation*. iCaRL moreover implements various techniques in order to mitigate the effect of catastrophic forgetting further, these are the management of the *exemplars* and a classifier based on the *Nearest Class Mean (NCM)* concept.
Although, these methods provide a significant improvement over Fine Tuning, the performances are still far from the Joint Training approach. In fact, the performances degrade along with the increasing number of classes.
In the last section, we aim to improve the limitations of the presented models, in particular the imbalance between previous and new data [2] and the ability of the network to maintain knowledge about observed classes.

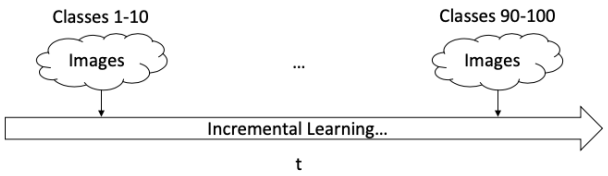

Figure 1. Incremental learning schema

[1] https://github.com/danielegenta/Progetto-MLDL

1.1. Literature

The literature about Continual Learning is very active at the moment. Researchers are focused on trying to reduce the gap between batch learning and incremental stream learning, especially when the number of classes, i.e. tasks, increases [8].
Addressing the task via generative adversarial networks, i.e GANs, is becoming possible since the quality of the images being generated is high, however this may result in excessive complexity [7].
Another approach might be to use progressive neural networks to avoid forgetting, by retaining a pool of pre-trained model during training, and learn lateral connections from these to extract useful features for the new task [5].
Targeting the class imbalance between the old classes seen at previous stages and the new ones at the current step, is another methodology being studied [2].

1.2. Data-set

The data-set that has been used throughout the project is *CIFAR100*. It consists in 60'000 labelled images that are equally distributed among 100 classes. A predefined split is available and divide the data-set into 50'000 training images and 10'000 test images. The size of each image is 32x32.
In order to mimic an Incremental Learning scenario, we divided both the train and the test set into 10 groups of 10 classes each. We have trained and tested our model on a group at a time. Since the split of the data-set plays a major role in the experiments' results, we consistently tried three different splits and averaged the results.

1.3. Model

The model that has been used throughout the project is an ad-hoc version of *Resnet 34* [1] for the CIFAR100 data-set. The model accepts as input 32x32 images, in order to improve generalization the train set images are firstly padded and then cropped to a 32x32 size, moreover it is performed a Horizontal Flipping transformation with probability 0.5. The test images do not undergo the previous transformations. Both the test and the train images are normalized using the CIFAR100 mean and standard deviation.
An alternative, in case the input data-set is not known apriori, would be to implement the commonly used 0.5 mean and standard deviation to normalize the images, the results wouldn't change drastically.

2. Baselines

In the following sections we present different methodologies to carry out an Incremental Learning task.
Firstly, we consider Joint Training, which consists in training the data-set on all the classes seen so far. This method will yield an Upper Bound for our actual Incremental Learning models.
Then, we consider Fine Tuning which aims to update the network's weights as new classes are learnt.
Lastly, we consider Learning without Forgetting and iCaRL, two methodologies from the literature to address the Incremental Learning task properly, by making use of knowledge distillation.
Each model will be presented together with its limitations.

The experiments have been carried out in the *Google Colab* environment, and are the result of different trials using three different random splits of the data-set. The results are reported in terms of accuracy and heat-map representing the confusion matrix obtained on the test set.
The models that will be described make use of a *Convolutional Neural Network* (CNN) and in particular the Resnet 34. This network can be interpreted as a trainable feature extractor, followed by a single classification layer with as many output nodes as the classes observed so far.
All the experiments carried out in the following sections have been performed using the iCaRL hyper-parameters reported in the following table, in order to have coherent baselines.
A further study might focus on performing hyper-parameter tuning on each different model in order to obtain a comparison between the best possible models' results.

Experiments' parameters	
Parameter	Value
LR	2
MOMENTUM	0.9
WEIGHT DECAY	1e-5
NUM. EPOCHS	70
MILESTONES	49, 63
BATCH SIZE	128
GAMMA	0.2

Table 1. Parameters used in all the experiments presented in the next sections.

2.1. Joint Training

The first considered approach is Joint Training.
This simple method consists in defining the training set of our model as the union of the images belonging to the new classes to be processed and the images belonging to all the classes observed so far. This means that, for each batch of classes, the parameters of the network are optimized on all the observed images.
So, compared to the classic fine tuning, which uses only the latest classes for training, the model is now trained simultaneously on all the classes known so far.
In this way, Catastrophic Forgetting is avoided because the network is not actually performing an Incremental Learning

task. In fact, we could label the result obtained with Joint Training as an Upper Bound for the methods that will be implemented in the following sections.
The results obtained on the new task and the original tasks are both good, in terms of accuracy, Figure 2.

Due to its nature, Joint Training has enormous limitations from an application point of view.
Increasing the number of classes to be learned, the size of the training set also increases. This results in a memorization problem that is not easy to manage. Moreover, the execution time might become excessively long due to the many iterations being performed.
Furthermore, in many applications that deal with image sequences it is simply not possible to trace and store the data of previous tasks. In cases like these, the Joint Training cannot be used since it requires the batches of previous data.

As expected, by performing the Joint Training on the 10 CIFAR100 class groups and evaluating the quality of the model on the test set after each training phase, we obtain excellent results, Figure 2, 3.
In fact, by training the network on the totality of the images every time a new task is added, the model is able to maintain an accuracy value whose range is between 70% and 80%. On the other hand, as expected, the time spent training the net on all classes was extremely long.

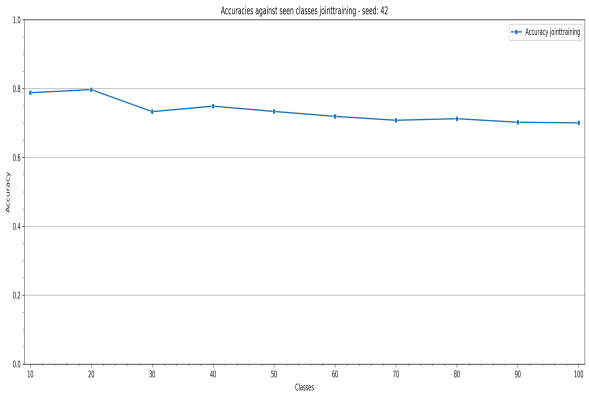

Figure 2. Accuracy against classes - Joint Training setting

2.2. Fine Tuning

In this section we demonstrate that without an ad-hoc method, the network is unable to learn new classes without dramatically forgetting the previous ones.
We perform a Fine Tuning of the model, in which the network is trained sequentially on 10 groups of the dataset which in turn contain 10 classes each, the total is 100 classes.

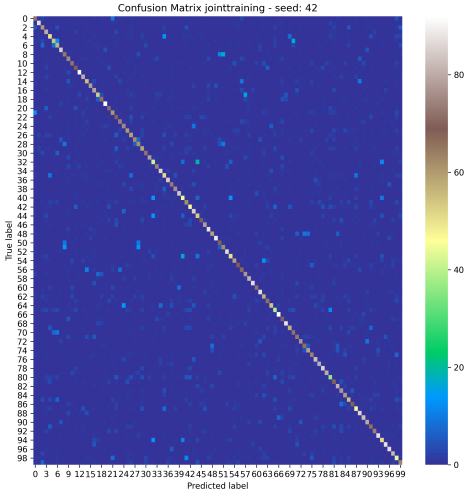

Figure 3. Confusion matrix - Joint Training setting

Firstly, we implement a Sequential Learning test baseline that evaluates the network only on the last batch of observed classes.
As we expected, in this case, the accuracy obtained on the evaluation set is excellent and, for each group of classes, it has a value between 78% and 93%, Figure 4. In fact, the network is validated on the last 10 classes on which it was trained and consequently is able to recognize accurately the features that characterize them.
On the other hand, being ours a Incremental Learning problem, we don't just observe the results of the latest classes, but we also focus on the classes learnt previously.

We therefore perform a second test, considering as the evaluation set all the classes observed so far.
Although the results obtained on the training set are good, the accuracy obtained from the test set during the execution of the various trains worsens drastically with the increase of the number of classes learned.
After the first group of classes the accuracy obtained is in line with the one of the validation set, but from the second group there is a major drop that brings the accuracy down to 45%. This decreasing trend continues with the addition of new groups and the final value of the accuracy after the train on the last group is 1%, Figure 5, 6.
This phenomenon is referred to as Catastrophic Forgetting and derives from a structural limitation of connection networks. The architecture of our network is based on a single set of shared weights that give the model the ability to distinguish and generalize patterns, because of this characteristic the network is very sensitive to the addition of new classes that disturbs drastically the previously learned pat-

terns.
To mitigate the Catastrophic Forgetting there are several methods, we will describe two paradigms addressing the issue: Learning Without Forgetting (LwF) and iCaRL.

2.3. Learning without Forgetting

A first approach to manage an Incremental Learning problem by avoiding running into Catastrophic Forgetting is to use Learning without Forgetting (LwF).
This method aims to improve the classification of both new and old tasks by combining the parameters obtained in output from the network on all the tasks seen up to that moment. While the parameters of the new classes are obtained in the current training phase, the parameters of the old classes must be saved so that they can be used in subsequent training phases.
In this section we first present an overview of the LwF architecture and its main components, then our experiments and results obtained by implementing it.

2.3.1 Architecture

The LwF can be described as a modification of the Joint Training where, instead of using the old images to expand the training set and train the network on all the classes seen up to that moment, the parameters returned from the previous classifications are used combined with the parameters obtained in the current classification phase. In this way, the training set is kept unchanged.
One of the advantages that the LwF brings over the Joint Training is certainly the reduction of the execution time. Once a new group of classes has been learnt and the results of the model have been stored, it will no longer be necessary to re-train the network on the same samples to preserve performance and this speeds up execution a lot.
Moreover, using LwF we're able to reach a true Incremental Learning setting, since we do not need to store observed images at all, resulting in a lower memory footprint and computational requirements.

Given a CNN, the shared parameters that are used to classify images are defined as θ_s, those of a task already learned as θ_o and those of a new task to be learned as θ_n. The goal to be achieved is to define θ_n and use it together with θ_o to identify parameters that can work well on both new and old classes.

Since ours is a classification problem, the model output's for each image is a set of label probabilities. When a new class is added, we insert a new node in the output layer of the network in order to manage the increase of classes number.
The classification strategy of LwF is simple: the labels with the highest probability will be chosen as the class of our image. During each training phase we record the responses of the network to the new group of images, and using a copy of the old network we also record the responses to the new images based on θ_o.
The network parameters' are updated by minimizing a loss function consisting of two main parts: classification loss which encourages predictions made on new images to be consistent with true labels in order to correctly classify new groups, and the distillation loss that pushes the output probabilities for each image to be close to the recorded output from the old network so as not to forget the tasks previously learned.
λ_o is a loss balance weight, set to 1 for our experiments.
The increase of λ will favor the performance of the old tasks compared to those of the new activity. R corresponds to a simple weight decay of 1e-5.
At the end of the training the network is characterized by parameters capable of recognizing the new classes without forgetting the previous ones.

2.3.2 Experiments

As in the previous experiments, we train the model by applying three different splits of the data set and we average the results in order to obtain a more robust baseline. From the graphs, Figure 7, 8, we notice that the results obtained are in line with those of the official LwF paper.
The only difference obtained is that in the first 30 classes the standard deviation of the results is slightly higher than the official reference, this problem could be due to a non-optimal split of the data.
Compared to Fine Tuning, the model with the LwF achieves better performances which lead to an accuracy on the last batch of classes which is between 20% and 25%. By looking at the heat-map representing the confusion matrix for LwF, we can confirm that the performances are decaying with the number of observed classes.

2.3.3 Conclusions and limitations of the model

As expected, the use of LwF mitigates the problem of catastrophic forgetting, in fact the collapse of accuracy with the increase in classes observed is reduced and better performances are achieved compared to Fine Tuning. Furthermore, LwF allows a true Incremental Learning setting, which doesn't require the storage of previous data.
Despite this, the results obtained in the second batch classes are already very far from those of the Joint Training and the accuracy falls to 20% when working with all the 100 classes, which is a very poor result.
In the next section our goal will be to further improve the network performances by applying a new paradigm, iCaRL.

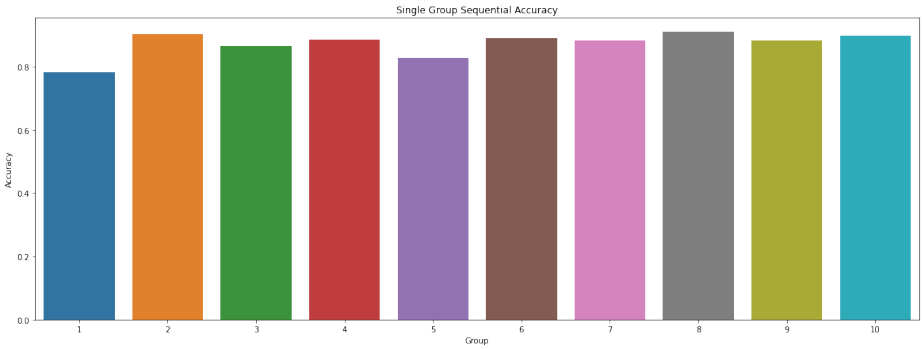

Figure 4. Accuracy against single batch of classes - Sequential Learning

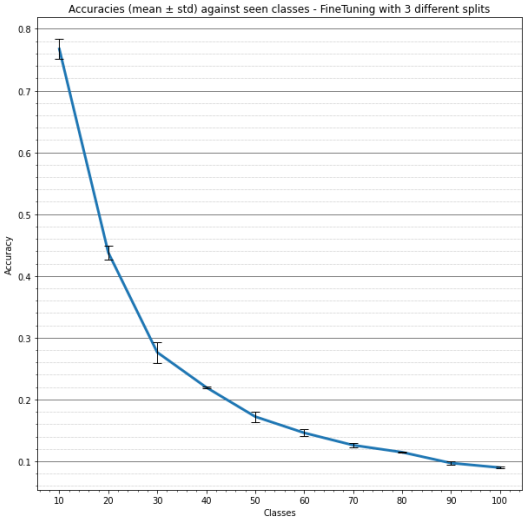

Figure 5. Accuracy against classes - Fine tuning setting

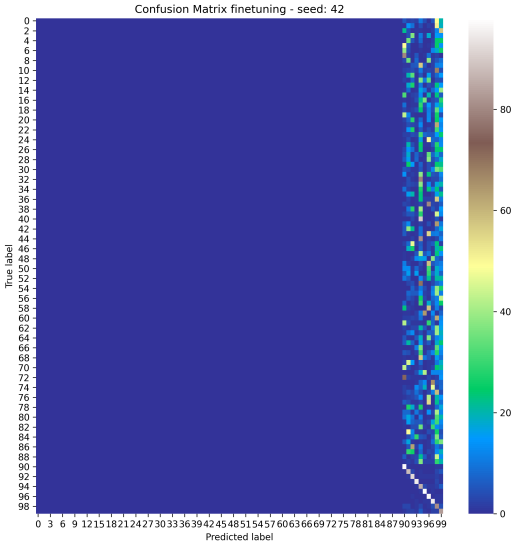

Figure 6. Confusion matrix - Fine Tuning setting

LEARNING WITHOUT FORGETTING:
start with:
θ_s: shared parameters
θ_o: task specific parameters for each old task
X_n, Y_n: training data and true labels of the new task
Initialize:
$Y_o \leftarrow CNN(\mathrm{X}_n,\theta_s,\theta_o)$
$\theta_n \leftarrow RandInit(|\theta_n|)$
Train:
Define $\hat{Y}_o \equiv CNN(\mathrm{X}_n,\hat{\theta}_s,\hat{\theta}_o)$
Define $\hat{Y}_n \equiv CNN(\mathrm{X}_n,\hat{\theta}_s,\hat{\theta}_n)$
$\theta_s^*, \theta_o^*, \theta_n^* \leftarrow \underset{\hat{\theta}_s,\hat{\theta}_o,\hat{\theta}_n}{\arg\min}(\lambda_o L_{old}(Y_o, \hat{Y}_o) + L_{new}(Y_n, \hat{Y}_n) + R(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n))$

2.4. iCaRL

In order to overcome the LwF limitations, we implemented the iCaRL methodology thoroughly. What differs from LwF is that iCaRL introduces new elements to tackle the Incremental Learning challenges.
In fact, iCaRL is founded on three key components:

- classification by nearest mean of exemplars rule,
- prioritized exemplars selection based on herding,
- representation learning using knowledge distillation and prototype rehearsal.

In this section we first present an overview of the iCaRL architecture and its main components and then our experiments and results obtained by implementing it.

2.4.1 Architecture

A brief explanation of the main components of iCaRL will now be presented.
As for classification, iCaRL relies on sets of exemplar images that are selected dynamically out of the data stream. There is an exemplar set for each class learnt so far and its size varies with the total number of observed classes. The total number of exemplar images is fixed, in order to provide a memory efficient method, this is represented by the parameter K, set to 2000 in the iCaRL paper.

The classification strategy for iCaRL is called *Nearest Mean of Exemplars (NME)* and it works as follow.
In order to predict a label for an unseen image, it computes a prototype vector of each class observed so far.
Each element of the vector is the mean feature vector of all exemplars for a certain class computed on all the exemplars for a certain class plus all the images belonging to that class,

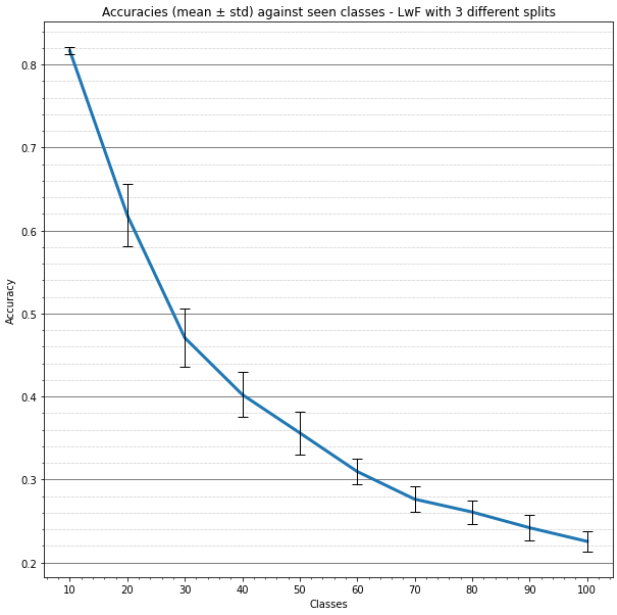

Figure 7. Accuracy against classes - LwF setting

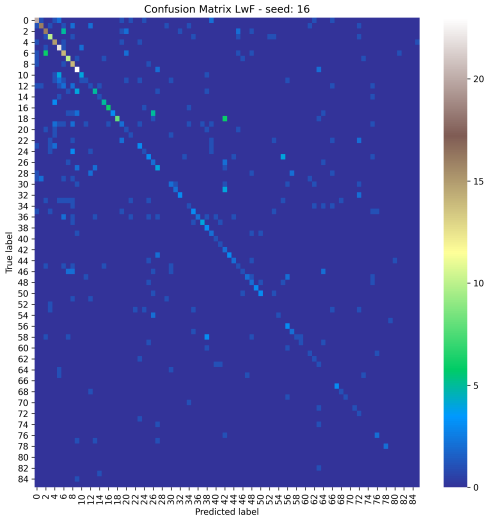

Figure 8. Confusion matrix - LwF setting

this reduces the noise when the number of exemplars per class is low. Then, the feature vector for the image that has to be classified is computed as well.
The class label is assigned via considering the most similar prototype (*prototype rehearsal*), according to the formula: Equation 1.

$$y^* = \underset{y=1,\ldots,t}{\arg\min} ||\varphi(x) - \mu(x)|| \quad (1)$$

Algorithm 1: iCaRL incremental train algorithm

Result: P $\leftarrow (P_1, ..., P_t)$
// New exemplar sets
Input $X^s, ..., X^t$
// training examples for batch of classes
Input K
// memory size
Require Θ
// current model parameters
Require P = ($P_1, ..., P_(s-1)$)
// current exemplars set
$\Theta \leftarrow UpdateRepresentation(X^s, ..., X^t, P, \Theta)$
m $\leftarrow K/t$
for y **in** classes_so_far:
  $P_y \leftarrow ReduceExemplarSet(P_y, m)$
end for
for y **in** new_classes:
  $P_y \leftarrow ConstructExemplarSet(X_y, m, \Theta)$
end for

Regarding the representation learning, iCaRL constructs firstly an augmented data-set containing the current training samples together with the stored exemplars.
Then, the current network is evaluated for each example and the resulting network outputs for all previous classes are stored.
Finally, the network parameters are updated by minimizing a loss function that is made up of two weighted parts: the classification loss that for each new image image helps the network to output the correct class indicator for new classes and the distillation loss, that helps the network to somehow reproduce the scores stored previously for old classes (this is actually the same concept of knowledge distillation presented in the LwF section).

As for the exemplars management, iCaRL uses a number of exemplars for each class that is equal to *m = K/t*, where t is the number of classes observed so far. The memory budget is in fact fixed throughout the execution of the application and it is equal to K.
The exemplars are selected by using herding. Operatively, each exemplar set is constituted of the m images whose features are more similar to the current class mean.
The exemplar set does not admit duplicates, as it would imply a waste of limited resources.
Each exemplar set's size is reduced when a new batch of classes is observed. Computing again m, where t is the updated number of classes, we keep only the first m exemplars for each class, i.e. exemplar set.
The exemplar sets construction and reduction are applied once for each batch of classes observed, this respects the Incremental Learning setting since the training data for each class are needed only when they're available and there is no need to store them further.

2.4.2 Experiments

Similarly to the previous experiments, we run the model with three different splits and averaged the results in order to obtain a more robust baseline. As it can be seen from the following plots, Figure 9 and 10, we managed to replicate the results presented in the official iCaRL paper.
Our results are in practice slightly worse in terms of accuracy, but this might be due to a non-optimal random splitting. With respect to the previous methods such as Fine Tuning and LwF, we can notice that iCaRL actually performs better, especially on the last batches of classes where the accuracy doesn't drop and remains in the realm of 45% to 50%.
The high variance underlined by the error plot, Figure 9, confirms that changing the split could lead to quite different results. The heat-map representing the confusion matrix computed on the test set confirms the progressive decaying in the performances, as more classes are observed.

2.4.3 Conclusions and limitations of the model

We're overall satisfied with the performances of iCaRL and, as expected, it achieves better results than the models presented in the previous section.
However, the results obtained are still far, in terms of accuracy, from those obtained with a Joint Training approach, i.e. a system trained in a batch setting.
This phenomena becomes more and more evident as the number of observed classes increases.

2.5. Baselines Conclusions

In this section we've implemented different methodologies, Figure 11, in order to address an Incremental Learning task.
As expected, Fine Tuning the network each time a new batch of classes is examined results in a Catastrophic Forgetting phenomena. LwF makes use of knowledge distillation in order to mitigate the problem and iCaRL takes it a step further making use of tools like the exemplars management and a classifier based on exemplars.
iCaRL achieves the best results in terms of accuracy, but still they're far from what it can be achieved in a batch setting, i.e. Joint Training.
In the following section we aim to carry out an ablation study in order to evaluate each component of iCaRL and some alternatives to it.
Our aim will be to deeply understand the intrinsic limita-

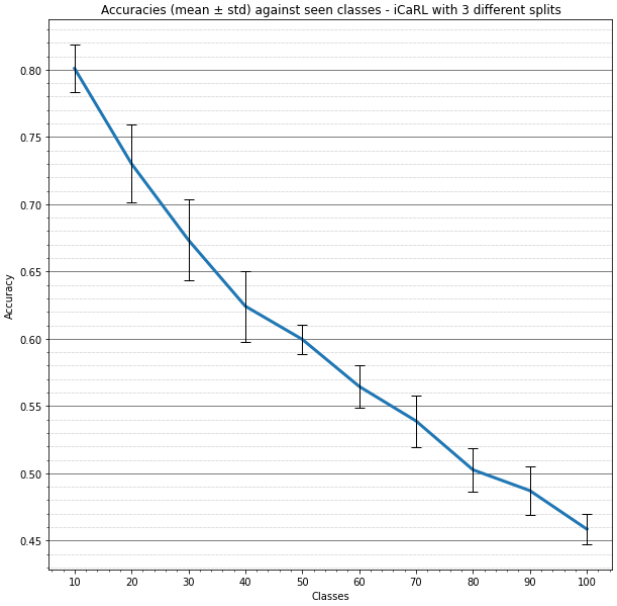

Figure 9. Accuracy against classes - iCaRL setting

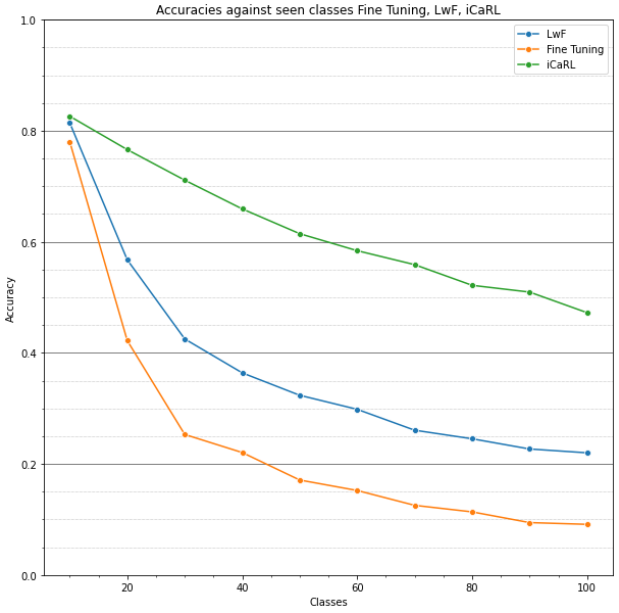

Figure 11. Classes against Accuracy for the different models

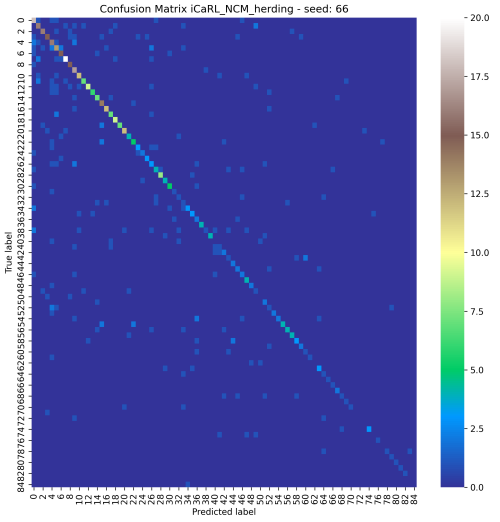

Figure 10. Confusion matrix - iCaRL setting

tions of iCaRL as well as its points of strength in order to be able, in the last section, to propose an improvement.

3. Ablation Study

In this section we perform multiple tests on our iCaRL baseline by trying out different sets of losses and classifiers. We also perform an implementation of the Hybrid 1 model presented in the iCaRL paper and a comparison between random exemplars selection and the default strategy based on herding.
Our aim is to test and verify what is the influence of each of the components presented in the official paper and their contribution to the overall performances, secondly we inspect the goodness of different choices with respect to the original reference.

3.1. Losses

In this section our aim is to inspect different combination of classification losses L_{class} and distillation losses L_{dist}. Since different losses produce values scaled differently, a loss balance weight λ_0 was required to being able to balance how much the network should remember, hence giving more weight to the L_{dist} or forget, hence giving more weight to L_{class}). Making λ_0 larger will favor the old class performance over the new classes. Consequently our final loss was composed as following: Equation 2

$$L(x) = (1-\alpha)L_{class}(x) + \lambda_0 \alpha L_{dist}(x) \tag{2}$$

with α defined as:

$$\alpha = \frac{n_{oldclasses}}{n_{oldclasses} + n_{newclasses}} \tag{3}$$

As the formulation (Equation 3) says, α is needed to make a weighted average of the two losses. This was required since PyTorch averages the contribution of each class. Using two losses, the contribution of every old class is divided by the number of old classes and the contribution of new classes is divided by the number of new classes. Hence re-balancing

the contribution to the total loss $L(x)$ of each loss was similar to move the importance of L_{class} to L_{dist} as new groups of data come in.

The losses considered where the following:

- Cross Entropy Loss
- Binary Cross Entropy Loss
- Mean Squared Error Loss (L2 Loss)

with the following formulations:

$$L_{CE}(x_i, y_i) = -\boldsymbol{p(y_i)} \cdot \log(\boldsymbol{p(x_i)}) = -\sum_{y=1}^{|C|} p_y(y_i) \log(p_y(x_i))$$

$$L_{BCE}(x_i, y_i) = -\sum_{y=1}^{|C|} p_y(y_i) \log(p_y(x_i)) + + (1 - p_y(y_i)) \log(1 - p_y(x_i))$$

$$L_{MSE}(x_i, y_i) = \|\boldsymbol{p(y_i)} - \boldsymbol{p(x_i)}\|_2^2$$

Where x_i, y_i are respectively the logit outputs and the ground truth of the image i and $\boldsymbol{p(x)}$ is a function returning the probability vector of the input x. If x is a vector of logits, a sigmoid or a softmax is applied to it and returned. If x is a label, $\boldsymbol{p(x)}$ returns the one-hot encoding of that label, hence still returning a vector of probabilities. $p_y(x)$ hence denote the element y of the probability vector $\boldsymbol{p(x)}$. This means that $p_y(x_i)$ corresponds to the probability of class y.

Please note that in both LwF and iCaRL, the ground truth of the image i in our distillation loss is $\boldsymbol{q_i}$, the outputs of the old network, which is still a vector of logits. Hence $\boldsymbol{p(y_i)}$ will correspond to $\boldsymbol{p(q_i)}$. So, $p_y(q_i)$ corresponds to the probability of class y of the old network.

We may notice how $L_{BCE}(x_i, y_i)$ does not match the following theorical formulation:

$$L_{BCE}(x_i, y_i) = -\sum_{y=1}^{|C|=2} p_y(y_i) \log(p_y(x_i))$$

However in this scenario it was extended to a multiclass application which for each class y takes into account both the contributes of true and false positives.

These generalized formulations are built to work both as classification losses and distillation losses in both iCaRL and LwF. For L_{BCE} and L_{CE}, the number of classes $|C|$ will be equal to $n_{newclasses}$ in case the loss is used as L_{class} and to $n_{oldclasses}$ if the loss is used as L_{dist}.

3.1.1 LwF losses

The combinations of losses tested for LwF are the ones in Table 2. The results are available in Figure 12.

Combination of losses tested	
L_{class}	L_{dist}
L_{CE}	L_{BCE}
L_{CE}	L_{MSE}
L_{BCE}	L_{MSE}

Table 2. Combinations of losses tested for LwF.

The hyperparameters used were the same used for the baselines, except for λ_0 and the learning rate set as written in Table 3.

Parameters used for each combination of losses tested		
L_{class} + L_{dist}	Learning rate	λ_0
$L_{CE} + L_{BCE}$	0.3	0.1
$L_{CE} + L_{MSE}$	0.3	0.01
$L_{BCE} + L_{MSE}$	2	0.01

Table 3. Parameters used for the losses tested in LwF.

As we can see from the results, no combination of losses could match with L_{BCE} used both as classification and distillation loss. We can think to this result on how the L_{BCE} is computed. For each class y, it sums the cost of having misclassified the class y plus the cost of having classified another class. Hence, it heavily penalizes punctual misclassification, while L_{CE} and L_{MSE} penalize an overall misclassification. This L_{BCE} behaviour helps a lot to effectively update weights to learn new classes while maintaining more knowledge of previous tasks.

3.1.2 iCaRL losses

The combinations of losses tested for iCaRL are the ones in Table 4. The results are available in Figure 13.

The hyperparameters used were the same used for the baselines, except for λ_0 and the learning rate set as written in Table 5.

As we can see from the results, no combination of losses could match with L_{BCE} used both as classification and distillation loss. Similar considerations can be made to

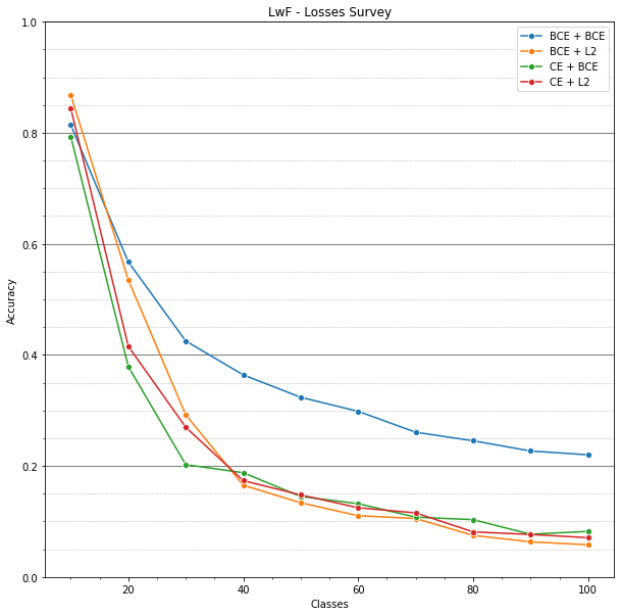

Figure 12. LwF Classes against Accuracy - Different combination of classification and distillation losses. In the legend they are denoted as "L_{class} + L_{dist}". L2 denotes L_{MSE}.

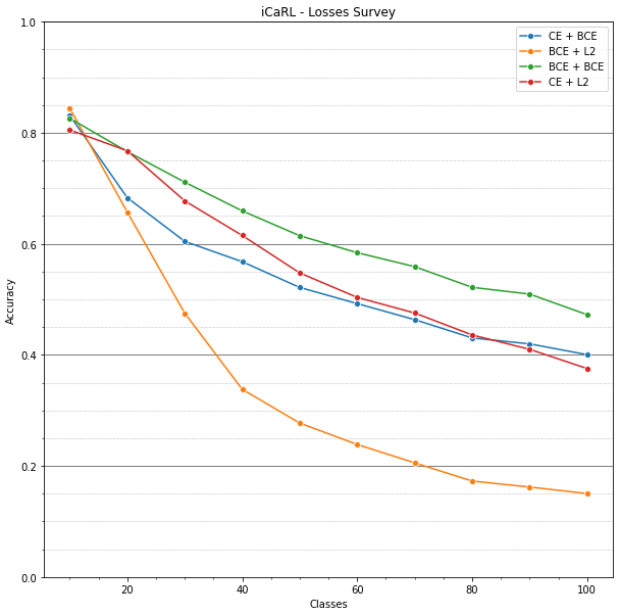

Figure 13. iCaRL Classes against Accuracy - Different combination of classification and distillation losses. In the legend they are denoted as "L_{class} + L_{dist}". L2 denotes L_{MSE}.

Combination of losses tested	
L_{class}	L_{dist}
L_{CE}	L_{BCE}
L_{CE}	L_{MSE}
L_{BCE}	L_{MSE}

Table 4. Combinations of losses tested for iCaRL.

Parameters used for each combination of losses tested		
L_{class} + L_{dist}	Learning rate	λ_0
$L_{CE} + L_{BCE}$	0.3	0.1
$L_{CE} + L_{MSE}$	0.3	0.1
$L_{BCE} + L_{MSE}$	2	0.01

Table 5. Parameters used for the losses tested in LwF.

the ones exposes in the previous test conducted on LwF. We can see how two combinations have closer performances to the ones obtained with $L_{BCE} + L_{BCE}$ and the gap is somehow narrower than the one we had with LwF. This is due to the fact that in iCaRL we are using exemplars in the training of new classes and this helps refreshing the network knowledge. The bad performances resulted by using $L_{BCE} + L_{MSE}$ may be due to non optimal parameter optimization. In fact, using L_{MSE} as distillation loss was really cumbersome to configure in terms of hyper-parameters and λ_0.

3.2. Hybrid 1 Implementation

In this section we aim to inspect the contribution of the *NME classifier* proposed in the iCaRL paper by confronting it with a plain *Fully Connected Classifier (FCC)*, similar to the one implemented in LwF.
As previously explained, the NME classifier proposed by iCaRL selects a class for an unseen image by considering which is the exemplar set whose mean, in terms of features, is the more similar with respect to the features of the given image. Using the FCC, instead, we use the plain network's output directly for classification.
The results are reported below, Figure 14.
By inspecting the plot, we can notice that the NME classifier proposed in the iCaRL paper actually achieves better performances, in terms of accuracy.
This is inline with what is stated in the paper and the effect could be bigger if we consider smaller and smaller batch sizes. In our own implementation the performances gap becomes greater with the increasing number of observed classes.

3.3. Classifiers

In this section we try out different classifiers in place of the iCaRL official *NME* classifier. The considered classifiers are: *Cosine Similarity, KNN and Linear SVC*. The implementations of these will be now explained in detail.

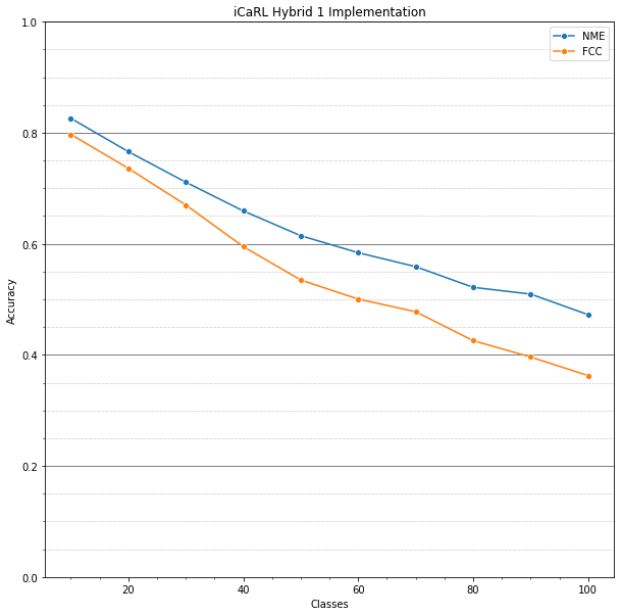

Figure 14. Classes against Accuracy - Classifiers, NME vs FCC

3.3.1 Cosine Similarity

The first approach we considered was to implement a classifier based on the Cosine Similarity. We have chosen this approach since it works similarly to the default NME classifier but computes the similarity differently and specifically it uses the formula in Equation 4 instead of using the similarity prototype, Equation 1, defined for the iCaRL baseline.

$$similarity = \frac{\varphi(x) \cdot \mu(x)}{||\varphi(x)||||\mu(x)||} \tag{4}$$

As we could expect, the change of the similarity does not affect significantly the results, Figure 15, which are lower with respect to the NME classifier but still comparable. This is because we maintain practically the same classification approach and we just change how the similarity is computed.

3.3.2 KNN

Secondly, we implemented a Classifier based on the *K-nearest-neighbors (KNN)* schema.
The KNN classifier decides the class of a test image by comparing it to the class of the K_{nn} nearest neighbors.
Operatively, we have fitted a KNN model to the exemplars data-set each time a new batch of classes is considered. This KNN model is then used during the classification step to make predictions about unseen classes from the test set.
In order to use the KNN model, the choice of the parameter K_{nn} is critical. Since our KNN model is just based on exemplars, we thought to make K_{nn} a variable that decreases during the Incremental Learning task, Equation 5. In fact, it starts from $K_{nn} = 20$ at the first step (200 exemplars/class) and maintain dynamically the same ratio throughout the execution of the application.

$$K_{nn} = \left\lceil \frac{exemplars per class}{10} \right\rceil \tag{5}$$

The performances, Figure 15, are lower than those achieved by iCaRL default NME classifier but the drop in the accuracy is not drastic. This might be because we fit the KNN model on the exemplars data-set which contains representative values per class and well separated points between different classes.
We have chosen KNN classifier due to its implementation' simplicity. One of the key limitations of KNN is that if the features data-set is large, then it becomes onerous. By using the exemplars data-set to fit the model we accomplish to overcome this limitation. Another limitation is that choosing K_{nn} improperly could yield poor results, in fact a K_{nn} which is too small imply a model sensible to outliers, while if K_{nn} is too large we could consider instances of other classes in the classification.

3.3.3 Linear SVC

The last Classifier we consider in our ablation study is the *Linear Support Vector Classification (Linear SVC)*. As for the KNN classifier we decided to fit the model on the exemplars data-set of iCaRL in order not to consider the noise in the training set and lessen the computational requirements.
The LinearSVC Classifier tries to separate the different classes by linear hyper-planes. We decided to use LinearSVC since it's computational requirements are not as heavy as KNN, since it is not interested in all the points of the data-set yet the ones on the margin (so called Support Vectors).
We implemented a basic version of the LinearSVC classifier by keeping the default value for the parameter C (cost for misclassification). The performances, Figure 15, of the LinearSVC Classifier are lower with respect to the default NME classifier, yet they're still comparable in terms of accuracy, moreover they are almost aligned to what we obtained in the Cosine Similarity Classifier implementation.
Again, we think that this is because we fit the LinearSVC model on the exemplars data-set which contains representative points per class and little noise compared to the original training set.

3.3.4 Classifiers conclusions

Figure 15 let us point out that the best scoring classifier is the one adopted in the iCaRL paper as default, i.e. NME.

We can also notice that the trend of the accuracies against the number of observed classes is similar trying out other classifiers and that we do not observe drastic drops in the performances of the model, i.e. *catastrophic forgetting*.
Our hypothesis of such a behaviour is that the other iCaRL implemented components, such as the exemplars management help the network in achieving performances higher than Learning Without Forgetting.
Finally, in our tests the better alternative classifier has been the one based on the Cosine Similarity, owing to its similar approach to NME; the worse instead has been KNN.

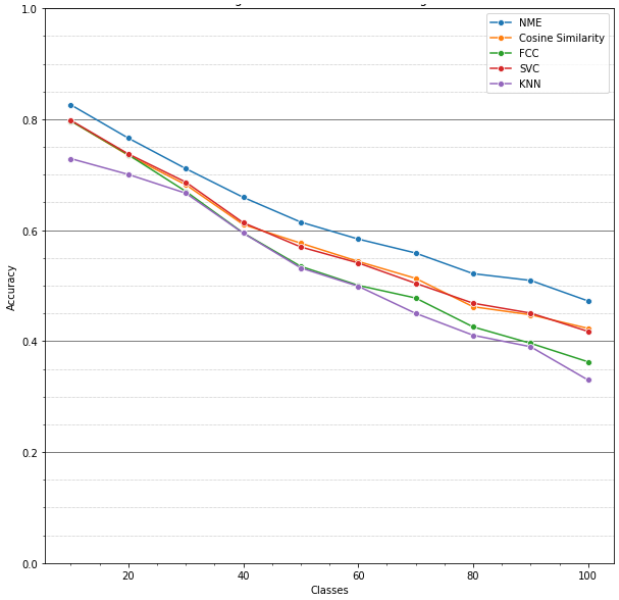

Figure 15. iCaRL Classifiers Survey

3.4. iCaRL exemplars selection

In this section our aim is to inspect which is the contribution of the exemplars' selection based on *herding* proposed by the iCaRL paper.
As previously explained, the selection of exemplars in iCaRL is based on the concept of making the average feature vector over all exemplars the best approximation of the average feature vector over all the training samples. This technique produces a prioritized list in which the position of each exemplar in the training set matters. This strategy is also computationally quite expensive.
We tried to perform a *random selection of exemplars*, maintaining the constraint of not having duplicates inside each exemplar set.
The results are reported below, Figure 16. By inspecting the plot, we can point out that the herding methodology achieves better performances yet the drop in the accuracy is not drastic and could also be partially owing to the variance intrinsic in each run, although the split compared is the same.

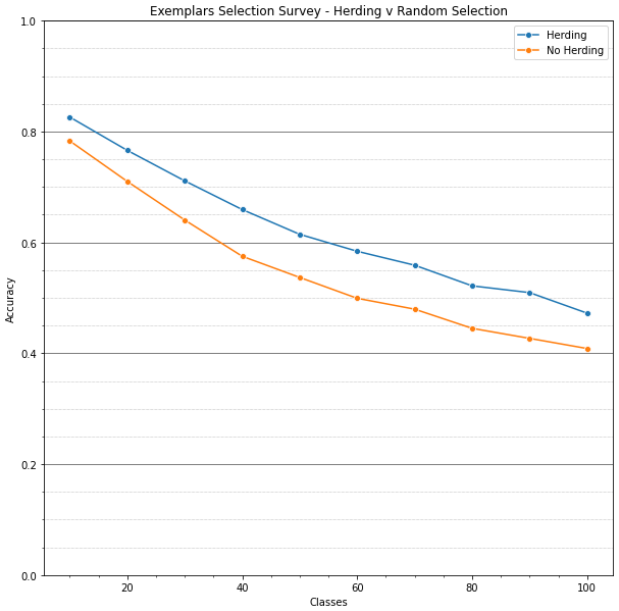

Figure 16. Classes against Accuracy - Random Exemplars Selection vs Herding

3.5. iCaRL conclusions and limitations

In this section we report the considerations about our implementation of iCaRL and the intrinsic limitations of the model.
By implementing iCaRL, we accomplished to obtain results comparable with those of the official paper.
Although they represent a step forward with respect to previous methodologies such as Fine Tuning and LwF, the performances are still far from what can be obtained in a batch learning setting.
The iCaRL model has in fact certain limitations, for instance iCaRL uses just one *teacher net* in order to maintain knowledge of previously observed classes and furthermore there is an imbalance in the number of images belonging to the current examined batch of classes and the images related to the previously observed classes (contained in the exemplars data-set). This limitations will be our starting point for the last section of this project. The techniques proposed to overcome them are explained in the next section.

4. Improvements

In this final section of the project, our aim is to propose some variations in order to overcome some of the iCaRL limitations, such as:

- use of a single teacher net to maintain the knowledge of old tasks,

- imbalance between the size of the new task data-set, with respect to the old tasks data-set (represented by the exemplars.

We will tackle these two points by developing two parallel branches.
In the first, our aim is to enhance the knowledge of old tasks by initially increasing the number of teachers and then changing paradigm to try to completely avoid catastrophic forgetting, this will be done by trying Progressive Neural Networks.
Secondly, we try to reduce the imbalance between new and old tasks, in terms of data-sets size, by reducing the number of training samples and in parallel expanding the exemplars data-set via data augmentation.
It is important to point out that in the latter step our memory K, reserved to storing exemplars will be kept fixed with respect to what has been proposed in the iCaRL paper.

4.1. Two teachers

The iCaRL model mitigates the Catastrophic Forgetting by using the distillation loss that allows to compare the output obtained from the old network on the new images with the output obtained from the previously learnt classes. In this way, the new model tends to remember as much as possible the weights that characterize the old net and therefore continues to correctly classify the old images.
To enhance this feature of iCaRL we decided not to limit ourselves to calculating a single distillation loss relative to the old model, but to expand the idea by using a further distillation loss that allows us to compare the output obtained from the previous network to the old one on the new images with the output obtained on the images already used.
Using this new distillation loss, the optimization problem will give further weight to the previous networks by improving our model's ability to remember the old tasks.

λ_j is the balance weight of the new distillation loss, its value has been conservatively set to 0.5 so that the weights of the previous model do not affect the learning of the new classes too much, limiting their accuracy.

4.1.1 Experiments

In this section the hyper-parameters have been kept coherent with the baselines detailed in the previous sections, in particular we kept Table 1 as a reference. The model was trained using L_{BCE} both as L_{class} and L_{dist}. As done with the previous experiments, we tried our modified model with three different splits of the data-set and averaged the results to have a robust baseline.
Looking at Figure 17, we can notice that the new model performances' are initially in line with the values achieved by iCaRL without modifications, but already from the third group of classes, the accuracy has a slight reduction.

TWO TEACHERS:
start with:
θ_s: shared parameters
θ_o: task specific parameters for each old task
θ_j: task specific parameters for each task prior to the old one
X_n, Y_n: training data and true labels of the new task
Initialize:
$Y_o \leftarrow CNN(\mathrm{X}_n,\theta_s,\theta_o)$
$Y_j \leftarrow CNN(\mathrm{X}_n,\theta_s,\theta_j)$
$\theta_n \leftarrow RandInit(|\theta_n|)$
Train:
Define $\hat{Y}_o \equiv CNN(\mathrm{X}_n,\hat{\theta}_s,\hat{\theta}_o)$
Define $\hat{Y}_j \equiv CNN(\mathrm{X}_n,\hat{\theta}_s,\hat{\theta}_j)$
Define $\hat{Y}_n \equiv CNN(\mathrm{X}_n,\hat{\theta}_s,\hat{\theta}_n)$
$\theta_s^*, \theta_o^*, \theta_j^*, \theta_n^* \leftarrow \underset{\hat{\theta}_s,\hat{\theta}_o,\hat{\theta}_j,\hat{\theta}_n}{\arg\min} (\lambda_j L_{old-old}(Y_j, \hat{Y}_j) + \lambda_o L_{old}(Y_o, \hat{Y}_o) + L_{new}(Y_n, \hat{Y}_n) + R(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_j, \hat{\theta}_n))$

Once the network has been trained on all classes, we obtain a final accuracy that is slightly above 40%.

4.1.2 Results and further studies

The results obtained are slightly lower than those achieved in the baseline implementation.
The use of the new loss probably disturbs too much the learning of the new classes and this entails a deterioration in the classification performance.
As it can be seen from the confusion matrix, Figure 18, with an increase in the number of considered classes, the model decreases its accuracy in labeling new images.
The change therefore did not bring substantial advantages compared to the classic iCaRL methodology, but this new model might be considered a starting point on which applying new updates, e.g. a Progressive Neural Networks approach.

4.2. Task-specific networks

In this variant of the iCaRL model we decided to make another step further in the limitation of the knowledge forgetting. Instead of having one network which tries to learn incrementally new groups of classes, for each of those we train a new ResNet. The iCaRL model will be responsible to incrementally learn to discriminate images belonging to different groups.
The classification of the images is then splitted in two steps: first the image group is predicted using the iCaRL model, then, using the classifier trained for that group, the image class is predicted. This strategy was inspired from the Progressive Neural Network paper [5]. The paper aims to elim-

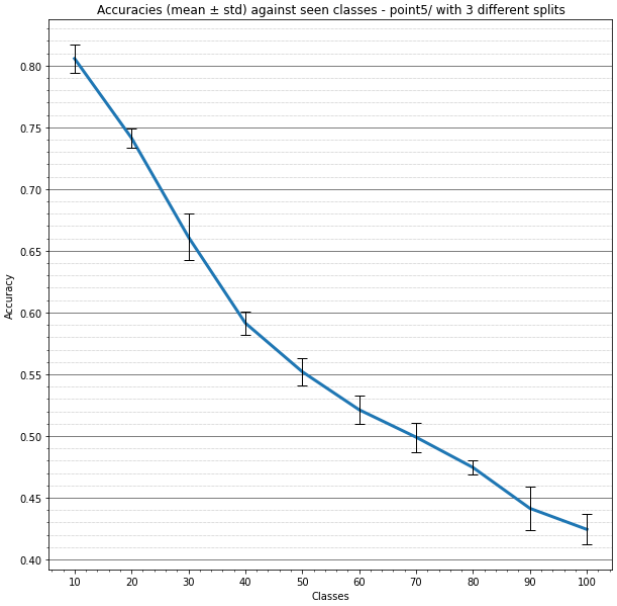

Figure 17. Accuracy against classes (3 seeds) - iCaRL setting, 2 teachers approach

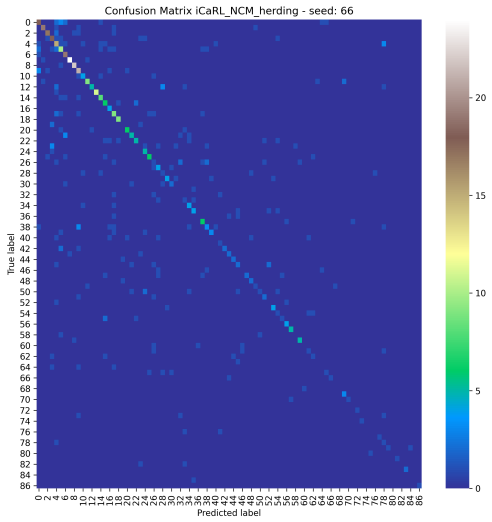

Figure 18. Confusion matrix - iCaRL setting, 2 teachers approach

inate the knowledge forgetting using a specifically trained classifier for each new task, but one of its limits is that the application of the model is task-aware, which in our case means that the user must be aware of the group the image belongs to. We tried to solve that by applying iCaRL on the prediction of the group (or task) before choosing the correct classifier. This is well represented in Figure 19.

Moving the Incremental Learning problem from classes to groups had the following benefits:

- with the same total number of exemplars K, for each group the number of exemplars was higher compared to the number of exemplars which were available for each class. We know that iCaRL performances improve storing an higher number K of exemplars, in other words iCaRL performs better with an higher number of exemplars for each label K/t.
- the number of labels to be learned by the iCaRL internal network was lower for each incremental training. This could improve the convergence rate of the iCaRL model faster. So, reducing the number of labels iCaRL has to remember could potentially simplify the incremental training, boosting the performances thanks to the previous consideration.
- at each incremental training, the number of images available for each label to learn was higher with higher variability, potentially helping to learn group-invariant features leading to a simpler classification of the correct group.

In the PNN paper the task-specific networks shared some connections between them. For sake of simplicity, we decided to adopt a simpler solution using independently trained 32-layers ResNets. The model was trained as described in Algorithm 2 and classes where predicted as described in Algorithm 3.

Algorithm 2: Task-specific networks training

Input X^g
// training examples for batch of classes
Input K
// memory size
Require Θ
// current model parameters
Require $P = (P_1, ..., P_{g-1})$
// current exemplars set
Require $T = (T_1, ..., T_{g-1})$
$G \leftarrow \{(x_i, g) : \forall (x_i, y_i) \in X^g\}$
$\Theta \leftarrow$UPDATEREPRESENTATION(G;P,Θ)
$T_g \leftarrow$TRAINRESNET(X^g)
$m \leftarrow K/g$
for $y = 1, ..., g-1$
  $P_y \leftarrow$REDUCEEXEMPLARSSET(P_y,m)
end for
$P_g \leftarrow$CONSTRUCTEXEMPLARSSET(G,m,Θ)
$P \leftarrow (P_1, ..., P_g)$
$T \leftarrow (T_1, ..., T_g)$

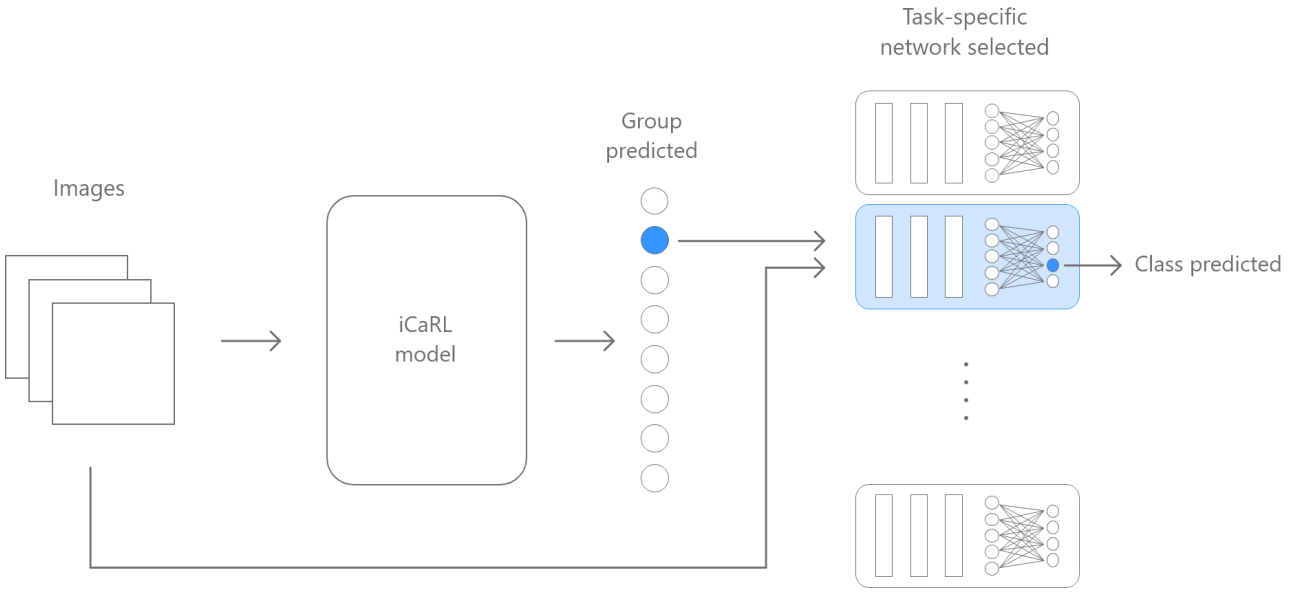

Figure 19. Task-specific network architecture

Algorithm 3: Task-specific networks classify

Input x
// training examples for batch of classes
Require Θ
// current model parameters
Require $P = (P_1, ..., P_n)$
// current exemplars set
Require $T = (T_1, ..., T_n)$
$g \leftarrow$ iCaRL.classify(x;P,Θ)
$y \leftarrow T_g(x)$
return y

4.2.1 Experiments

The model was trained using L_{BCE} both as L_{class} and L_{dist} for the embedded iCaRL model which was allowed to store K = 2000 exemplars. The task-specific networks T_g were 32-layers ResNet. Each training step consists of 70 epochs. The learning rate starts at 2.0 and is divided by 5 after 49 and 63 epochs. Different λ_0 were tested but the best result was obtained with λ_0 equal to 1.

4.2.2 Results

This architecture performs poorly compared to the original iCaRL as it can be seen in Figure 20. Despite of having a specific classifier for each group of images, the main problem was probably to correctly classify the group. Indeed it appears in the confusion matrix that the model kept the knowledge about old classes but wasn't able to learn new classes. So iCaRL didn't learned new groups.
Implementing the same patter with Progressive Neural Networks could be the object of a further study.
In the next section, we will try to solve the problem of unbalancing between number of new images and old images (exemplars) in the training phase.

4.3. Training samples and exemplars imbalance

In this section we aim to improve the limitation of iCaRL by which there's an imbalance between the number of images belonging to a new task, i.e. the last batch of classes considered, and the size of the data-set representing the old tasks, i.e the exemplars data-set.
Operatively, this section comprehends two updates with respect to the iCaRL management of exemplars.

First, we try to reduce progressively the samples composing the current training set in order to verify if the net-

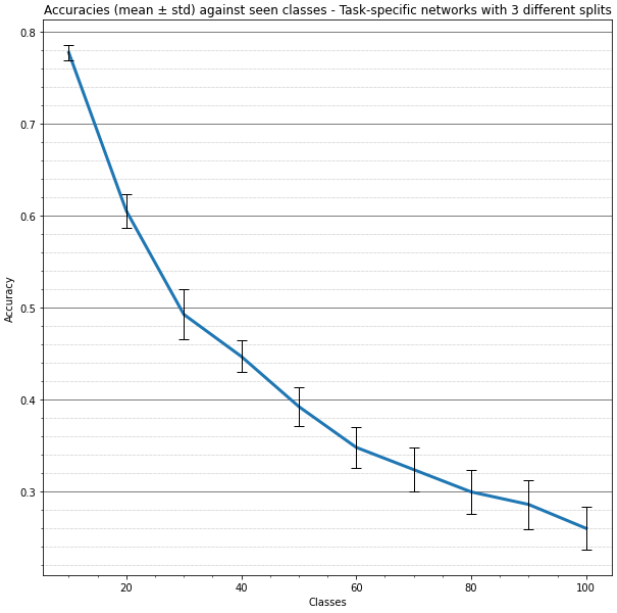

Figure 20. Accuracy against Classes (3 seeds) - Task-specific networks setting.

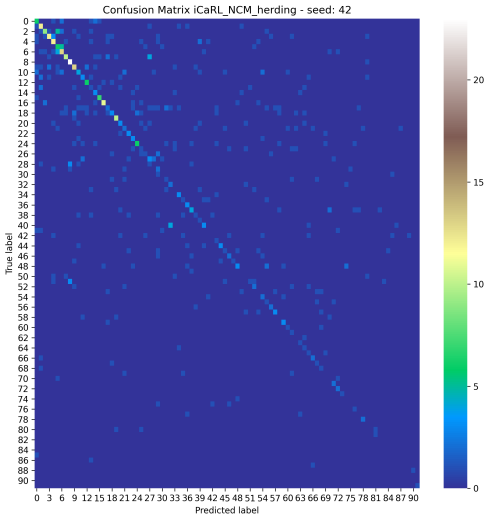

Figure 21. Confusion Matrix - Task-specific networks setting, iCaRL classifier NCM, seed 42

work is still able to learn and, if so, reduce the initial size of 5000 samples per task which is very unbalanced with respect to the data-set representing the old-tasks, i.e the exemplars data-set. In order to address this point we make use of Equation 6.

$$new\_dataset\_size = dataset\_size - (450 * (observed\_classes/10)) \quad (6)$$

The constant 450 has been found empirically and allow to reduce progressively the number of new samples considered during training, from 5000 on the first task down to 500 on the last one. Plase note that the cut of the samples data-set is not performed on the first batch of observed classes.
It is important to point out that the new data-set of samples is selected randomly without replacement and it aims to maintain the distribution of the different classes.

At the same time, in order to reduce the imbalance between the new and the old tasks in terms of data-set size, we performed a Data Augmentation on the exemplars data-set with the aim of improving the network's performances on the old tasks.
Since increasing the cardinality of the exemplars set, at training time, by doubling some of the images is considered a waste of limited resources, we addressed the problem by considering transformations of the current exemplars.
It is important to note that the Data Augmentation is performed at training time and the memory reserved to store exemplars, K, has not been increased since it would have resulted in an easy and meaningless boost of the iCaRL performances.
The Data Augmentations that have been performed are reported in the following Table 6, during each experiment the probability of those transformations has been kept between 0.7 and 1.

Data Augmentation trials	
Trial	Experiment
1	Horizontal Flip
2	Vertical Flip and GreyScale
3	GreyScale

Table 6. Experiments regarding exemplars data-set data augmentation

4.3.1 Experiments

Again, the hyper-parameters have been kept fixed with respect to those detailed in Table 1.
The experiments have been carried out by following the iCaRL paradigm and by using two teachers nets as detailed in the previous section.
The accuracy trend of the different experiments is reported in Figure 22.

It is interesting to point out that different Data Augmentation transformations yielded similar performances in

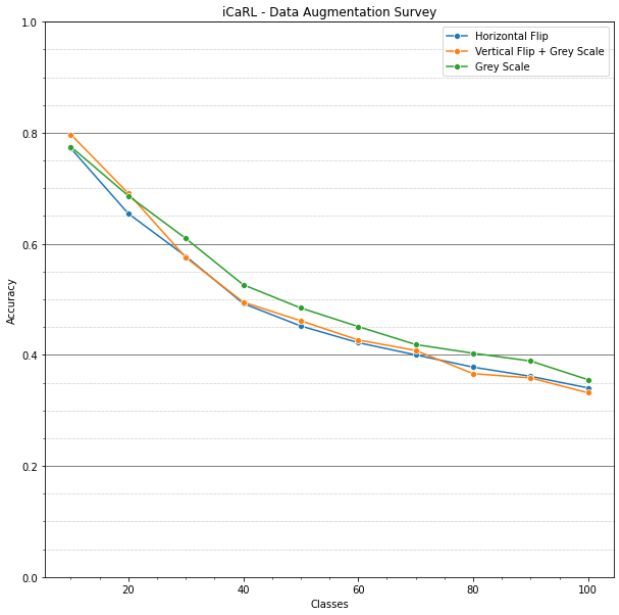

Figure 22. Accuracy against classes - Data Augmentation Survey

terms of accuracy on the test set. These performances are slightly worse than those achieved in the default iCaRL implementation, this might be due to interference produced by the implementation of the two teachers net used to perform these experiments.
Moreover, Data Augmentation could be weak in this specific problem since the size of the exemplars data-set, on which the transformations are performed, is quite small overall. Beside that, the size of each exemplar set, i.e. the set containing the exemplars related to a specific class, becomes smaller and smaller during the Incremental Learning task, so performing Data Augmentation of these very small sets is even weaker.
Further studies might focus on an ablation study regarding this point.

4.3.2 Results, limitations and further studies

Performing a reduction of the training samples corresponding to the new task together with a data augmentation carried out on the exemplars data-set did not help to obtain big leaps forward in the iCaRL performances, which have remained comparable.
These experiments, however, give us some interesting insights, for instance that the network is able to learn and maintain the acquired knowledge even if the samples of the new task are reduced.
Further studies might develop these ideas deeper for example by enhancing the exemplars data-set using GANs instead of Data Augmentation or by performing a study in which it is demonstrated how many samples of the new task are needed to make the network learn. Moreover, it could be investigated how to choose which samples to consider, based on certain characteristics.

5. Conclusions

In this project we have addressed an Incremental Learning task by adopting different approaches.
Firstly, we have performed a Joint Training in order to have an upper bound for the task, in terms of accuracy.
Then, we have implemented the state of art methodology in batch learning, Fine Tuning, ending up in catastrophic forgetting.
Furthermore, we have exploited new techniques such as knowledge distillation and exemplars management in order to mitigate the catastrophic forgetting phenomena, by implementing LwF and iCaRL.
In order to deeply understand iCaRL and its limitations we have performed an ablation study in which we have isolated each of its components in order to evaluate its contributions to the performances enhancement and exchanged it with different alternatives to evaluate new combinations of components.
Lastly, we tried to overcome two of the iCaRL limitations that are maintaining the knowledge on old tasks and imbalance between the size of the images regarding the old and new task. Our trials haven't been successful in terms of numbers but are, we think, a stepping stone and a series of experiments in a very active research area.
Further studies might specifically tackle the two problems we underlined previously, by respectively expanding the concept of Progressive Neural Networks or making use of GANs to increase the exemplars data-set.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[2] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. pages 831–839, 2019.

[3] Z. Li and D. Hoiem. Learning without forgetting. *ECCV*, 2016.

[4] F. Robert. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 1999.

[5] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks, 2016.

[6] A. K. Sylvestre-Alvise Rebuffi and C. H. Lampert. icarl: Incremental classifier and representation learning. *CVPR*, 2017.

[7] C. Wu, L. Herranz, X. Liu, Y. Wang, J. van de Weijer, and B. Raducanu. Memory replay gans: learning to generate images from new categories without forgetting, 2018.

[8] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning, 2019.