Incremental Learning in Image Classification

Manuele Macchia
s277309@studenti.polito.it

Francesco Montagna
s277596@studenti.polito.it

Giacomo Zema
s269614@studenti.polito.it

Abstract

Extending the knowledge of a model is an open problem in deep learning. A central issue in incremental learning is catastrophic forgetting, resulting in degradation of previous knowledge when gradually learning new information.

The scope of the described implementation is to reproduce some existing baselines that address the difficulties posed by incremental learning, to propose variations to the existing frameworks in order to gain a deeper knowledge of their components and in-depth insights, and finally to define new approaches to overcome existing limitations.

1. Introduction

Incremental learning is a paradigm that allows extending the knowledge of an existing model, gradually incorporating new information, as opposed to training the model on a complete dataset. Training a system on a dataset that includes all classes is unfeasible in many real-world scenarios, *e.g.*, where data comes in a stream or where previously collected data is not available anymore. Therefore, the possibility of incrementally learning new data while maintaining existing knowledge has great importance for training more flexible and dynamic systems.

A central issue in incremental learning is *catastrophic forgetting* or *catastrophic interference* [8], *i.e.*, training a model with new data interferes with previously acquired knowledge. This leads to a performance decrease or, in the worst case, to the old knowledge being overwritten.

Considerable research has been devoted to limiting the effects of catastrophic forgetting. We mainly consider two incremental learning baselines in computer vision, Learning without Forgetting [7] and iCaRL [9]. These methods mitigate catastrophic forgetting by adopting techniques such as knowledge distillation and prototype rehearsal, which we further explore in our study.

In this work, we implement the existing incremental learning baselines of Learning without Forgetting and iCaRL to gain a deeper knowledge of the problem and the proposed techniques. We explore the effect of each component of the baselines on the performance of the model, and study the effectiveness of alternative components, such as different loss functions and classifiers. Finally, we propose novel approaches to overcome existing limitations.

2. Method

In this section we describe the implementation details of our work, the underlying convolutional neural network and the dataset partitioning for incremental learning.

2.1. Network

We carry out all experiments using a 32-layers ResNet [3], a deep convolutional neural network. Residual networks address the vanishing gradient problem with shortcut connections between layers, allowing for deeper architectures. From now on, we treat the network up to the second to last layer as a feature extractor $\varphi : \chi \to \mathbb{R}^d$ and the last fully connected layer as a classifier. This choice is in line with the implementation of iCaRL.

In our incremental setting, we initialize the last fully connected layer to ten output nodes, and gradually increment this number as the network learns new classes. This approach is coherent with the assumption that the total number of classes is not known a priori. We use the Xavier normal initializer [2] to initialize the network weights. In our experiments, we find that this approach leads to better accuracy.

2.2. Data

We execute our experiments on the CIFAR-100 dataset [6]. It is a popular dataset for incremental learning, and both our baselines [7, 9] use it to test their strategies. It comprises 60000 tiny images belonging to 100 classes.

CIFAR-100 provides 500 training images and 100 testing images per class. We hold out 50 images per class from the training set to form a validation set. We use this data to validate our models in some of our studies and to monitor the performance of the network during training.

We perform data augmentation on the original dataset. This technique may help in improving the results and avoiding overfitting. We apply the same data preprocessing that the iCaRL authors use in the official code[1], which consists

[1] https://github.com/srebuffi/iCaRL

in applying a random horizontal flip with 50% probability and a 32 by 32 random crop with 4 pixels of padding to the training set. These transformations are actually effective, increasing the accuracy of the model on the first incremental step by 10%.

All three RGB channels are normalized with mean 0.5 and standard deviation 0.5. Using the true mean and standard deviation of the dataset would violate the incremental learning assumptions.

2.3. Incremental approach

Our incremental approach is consistent with the benchmark protocol of iCaRL. The dataset is split into ten batches of ten classes each, with classes randomly assigned to a batch. At each learning step, the network is trained on a batch of data and evaluated on a test set that contains all known classes up to that point. To obtain statistically significant results, we fix three class orders and run all experiments three times, as different class splits lead to very different accuracy scores. We present the results in terms of mean and standard deviation of classification accuracies after each batch of classes, using an error bar plot or reporting the average of the mean accuracies, called *average incremental accuracy*.

This approach does not eliminate all sources of randomness, which are still present in the initialization of the network weights. We do not explore this aspect due to time and computational resources limitations.

2.4. Settings

We train the model using stochastic gradient descent with momentum as optimization algorithm, with mini-batch size set to 64 and weight decay equal to 0.00001. The initial learning rate is set to 2.0 and is divided by 5 after 49 and 63 epochs. For prototype rehearsal, we store a maximum of $K = 2000$ exemplars. These are the same hyper-parameters used by the authors of iCaRL. The only exception is the size of the mini-batches, which we lower to 64 to obtain results comparable to the ones reported in their study.

This is the default hyper-parameter set-up. All further variations that we apply to these settings are reported in the description of the experiments.

2.5. Code

For our experiments we rely on the *PyTorch* framework. Our source code is publicly available at `https://github.com/manuelemacchia/incremental-learning-image-classification`.

3. Experiments

In this section we describe our implementation of the existing incremental learning baselines, present the results and provide an upper bound for the accuracy of incremental learning strategies.

3.1. Joint Training

Our first experiment violates the incremental learning protocol. At each learning step, the network is trained not only on the current batch of classes, but on the union of all batches seen until that point. Therefore, we get an upper bound for the classification accuracy at each incremental step.

3.2. Fine-tuning

The first incremental learning baseline is a naive fine-tuning approach. This method is useful to understand how catastrophic forgetting affects the accuracy of a model in the incremental learning setting. We use the validation set to choose the network that minimizes the validation loss. This network is then used to evaluate the model on the current test set.

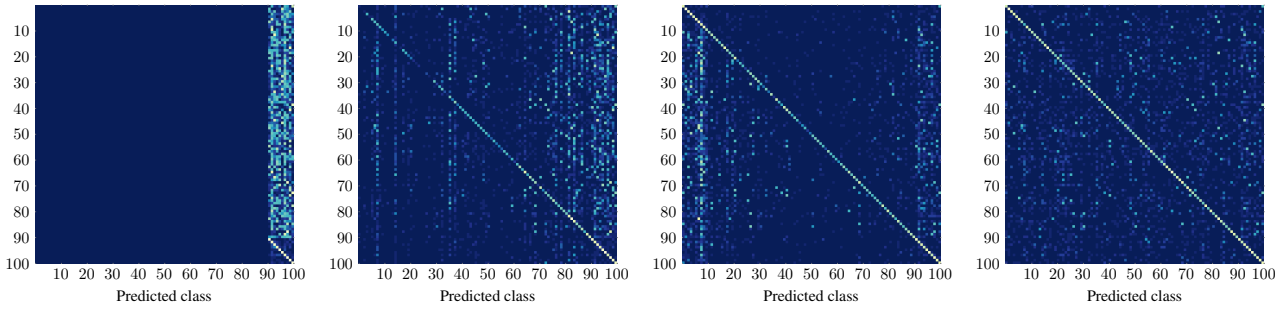
We use the binary cross entropy (BCE) loss function with one-hot encoded targets as the only contribution to the loss function. At each learning step, we optimize our network for the new classification task. The weights learned at the previous step are effectively the initialization for the new task network. As there is no mechanism to preserve previous knowledge, the optimization function updates the weights with the only goal of correctly classifying the current batch. In doing so, it makes the current weights less suitable for classifying old classes. This is how catastrophic forgetting occurs.

Figure 2 shows that fine-tuning is the least effective method among the baselines. The classification accuracy drops significantly at the end of the second learning step, and keeps decreasing as the number of classes grow. The accuracy of the model at the last batch hovers around 10%, which suggests it is able to properly classify only the last batch of classes. The confusion matrix in Figure 1 confirms this observation.

3.3. Learning without Forgetting

As a first step to mitigate the effects of catastrophic forgetting, we implement the Learning without Forgetting [7] framework. The novelty of this approach is the addition of the distillation loss, as presented in [4]. Knowledge distillation consists in transferring knowledge from the previous network to the current one. We copy and freeze the previous network and then train on the new batch of data. We use this training data to obtain outputs from the previous network and we use these outputs as targets for the training network. The network is encouraged to reproduce the probability scores of the old network by a distillation loss.

Differently from the Learning without Forgetting paper, we use the BCE loss for both the distillation and classifi-



(a) fine-tuning (b) LwF (c) hybrid1 (d) iCaRL

Figure 1: Confusion matrices of different methods on the incremental version of CIFAR-100. The predictions of iCaRL are evenly distributed across all classes, whereas Learning without Forgetting tends to classify samples from recent batches more frequently. The predictions of hybrid1 are skewed towards the first and the last batches, while fine-tuning exclusively predicts classes from the last batch.

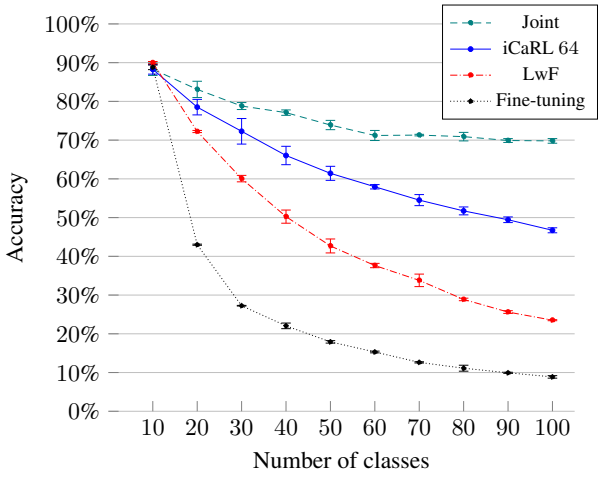


Figure 2: Accuracy comparison of the implemented baselines and upper bound. iCaRL outperforms both Learning without Forgetting and fine-tuning. The gap between the methods increases and the curves become flatter as more incremental steps are taken.

cation contributions. We are favouring the loss function of iCaRL to achieve better performance, as we demonstrate with our experiments on loss functions in Section 4.

Figure 2 shows a noteworthy increase in classification accuracy. By introducing a distillation loss contribution, Learning without Forgetting reaches an average incremental accuracy of 46.4%, while fine-tuning only scores 25.2%, highlighting the effectiveness of knowledge distillation. This is confirmed by the confusion matrix in Figure 1.

3.4. iCaRL

The final baseline for incremental learning consists in the iCaRL [9] algorithm. Besides using knowledge distillation, iCaRL introduces two novel components to address the shortcomings of the previous approaches. Firstly, it stores in memory K exemplars from old classes, augmenting the training dataset. It uses this additional training data to rehearse previously gained knowledge, limiting the effects of catastrophic forgetting. The size of the exemplars set K is fixed, so the memory footprint of the algorithm remains bounded. Secondly, it introduces a *nearest-mean-of-exemplars* (NME) classifier. In this setting, the network is treated as a feature extractor up to the second to last layer. To classify an unlabeled sample, iCaRL computes a prototype vector for each class observed so far and the feature vector of the image to classify, and assigns the class label of the most similar prototype.

The NME classifier is a valuable addition that results in a significant increase in classification accuracy. To understand the impact of this component, we reproduced the differential analysis of the iCaRL paper by creating a hybrid setup that learns a representation in the same way as iCaRL, but uses the outputs of the last fully connected layer for classification instead of the NME classifier. This is referred to *hybrid1* in the original paper and in our study as well. The results of our experiments show that NME increases the average incremental accuracy by 4.7%, as reported in Table 1.

The loss function used by iCaRL presents a distillation term that encourages the network to reproduce the scores of the old network for previous classes, and a classification term to output the correct class label for new classes.

In our implementation of iCaRL, we do not apply any data augmentation to the exemplars set in order to better preserve the original mean when computing the class prototype. We find that this marginally improves the classification performance of NME. Furthermore, we do not perform herding, *i.e.*, prioritized exemplars selection, but randomly sample exemplars from the current batch of data without

Method	Avg.
Fine-tuning	25.7%
LwF	46.4%
iCaRL	62.7%
hybrid1	58.0%

Table 1: Comparison between results with fine-tuning, Learning without Forgetting, our implementation of iCaRL and hybrid1 in terms of average incremental accuracy.

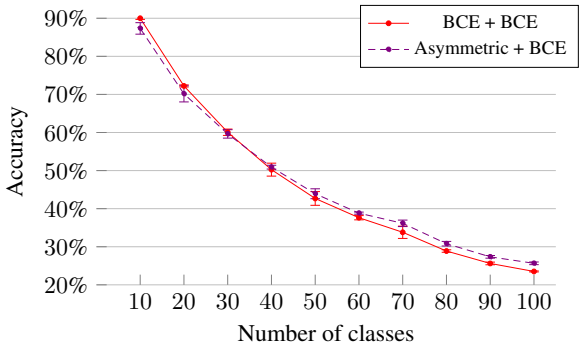


Figure 3: Different accuracy reached in the Learning without Forgetting framework using our custom asymmetric loss for classification against a network that uses the BCE loss. The distillation loss is BCE for both methods.

replacement. This approach is computationally less expensive than herding and improves the average incremental accuracy by 1.8%, compared to using the original iCaRL approach. We do not modify any other part of the exemplars handling algorithm.

Note that our results are 1.5% lower than those reported by the authors of iCaRL. This difference may be due to the fact that the authors implement their algorithm using TensorFlow and Theano, while our code is based on PyTorch. This may also explain why we obtain better results by halving the original mini-batch size.

4. Loss Ablation Study

In this section we mainly focus on loss functions. We want to observe the behaviour of the network as we replace the classification and distillation losses with different combinations. We mostly use the iCaRL framework as a baseline for our experiments, as it builds on the insights of Learning without Forgetting to increase the classification performance. In fact, it is the best performing method among all the evaluated incremental learning baselines.

This class of experiments is mainly devoted to understanding the behaviour and limitations of the existing frameworks, rather than trying to increase the accuracy of the baselines.

4.1. Learning without Forgetting

We perform only one experiment with the Learning without Forgetting framework. We define a custom loss that addresses a criticality in our setting. Whenever we train a model, after a finite number of epochs, the non-zero probability scores assigned to classes other than the target bring valuable information related to the similarity between the target class and other classes. The usage of the BCE loss function for classification implies that penalization does not only occur for misclassification errors, *i.e.*, when predictions differ from targets, but also when probabilities differ from exactly 0 on all remaining classes, implying also penalization of some informative scores.

To clarify our reasoning, let us consider a toy classification problem with three classes: cat, dog and car. The cat class is more similar to dog than car. During training, whenever the model encounters a sample of cat, valuable similarity information is encoded in the probability ratios between cat and the other classes, as we expect the model to give a higher probability score to dog than to car.

To verify this intuition, we design an asymmetric classification loss function with different penalization assigned to targets 0 and 1:

$$\mathcal{L}_{asym} = \sum_{y=s}^{t} -y \log g(x) + (1-y)\ (g(x))^2 \qquad (1)$$

where $s, \ldots, t$ are the new classes and $g(x)$ is the output of the sigmoid layer. Targets are passed to the function with one-hot encoding.

While BCE penalizes all targets 0 logarithmically, the cross entropy (CE) loss considers only the single output node of the target class. Our classification loss (1) stands between these two, with a quadratic penalization of 0 targets. We think that this formulation has an advantage over CE in the Learning without Forgetting framework, as it addresses the imbalance in contributions that occurs when the number of old classes grows. The classification loss becomes less and less important than the distillation part as more incremental learning steps are taken, because the distillation contribution involves an increasing number of output nodes. When this disproportion is high, the classification loss is not able to adequately learn to classify new classes and the performance of the network on the new batch decreases.

In this setting, using a sigmoid layer instead of a softmax function to transform the outputs of the last fully connected layer to a probability distribution is fundamental. The softmax function normalizes the outputs in such a way to guarantee that the sum of all outputs is 1. On the other hand, the sigmoid layer simply maps the scores to the $[0, 1]$ range, as-

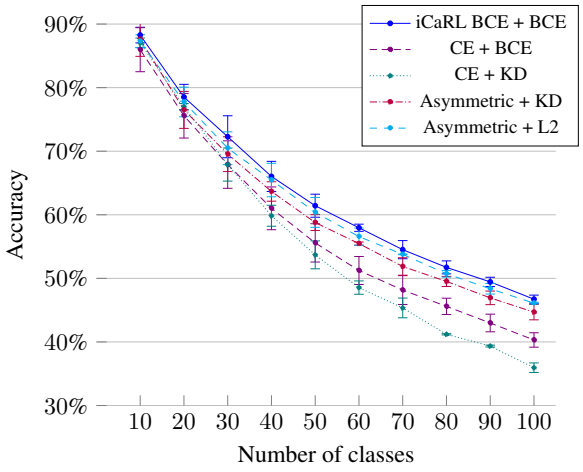


Figure 4: Representation of test accuracy of the iCaRL framework with different loss function pairings. The choice of the loss function is not negligible, resulting in about 10% difference of average incremental accuracy on the last split between BCE + BCE and CE + KD. We also see how the CE loss for classification is always giving the worst performance.

signing a probabilistic meaning to each output node activation independently from other ones. We present the results in Figure 3.

4.2. iCaRL

4.2.1 CE + BCE

The first combination of classification and distillation loss functions that we implement in the iCaRL framework is CE and BCE. In Section 4.1 we explain how these two contributions are unbalanced. In particular, we show how the CE contribute loses of importance as more learning steps are taken, due to the increasing number of output nodes for old classes. The inability to adequately learn new classes exhibits as an accuracy drop with respect to the baseline, as we show in Figure 4.

4.2.2 CE + Knowledge Distillation

The second experiment that we perform involves the knowledge distillation loss introduced by [4]. The combination of this distillation loss with CE for classification is in line with the original implementation of the LWF framework. We want to understand whether the use of BCE for both classification and distillation is sensible by performing this experiment, also given the conclusions drawn in Section 4.2.1.

The knowledge distillation loss presented by [4] can be

Method	Avg.
Asymmetric + BCE (LwF)	47.1%
CE + BCE	57.4%
CE + KD	55.6%
Asymmetric + BCE	60.5%
Asymmetric + L2	61.7%

Table 2: Comparison of the average incremental accuracy of different combinations of loss functions. The first combination refers to the Learning without Forgetting algorithm, while all the other ones are implemented using iCaRL.

interpreted as a modified CE loss:

$$\mathcal{L}_{old}(y_o, \hat{y}_o) = -H(y'_o, \hat{y}'_o) = -\sum_{i=1}^{l} y'^{(i)}_o \log \hat{y}'^{(i)}_o \qquad (2)$$

where l is the number of old classes and $y'^{(i)}_o, \hat{y}'^{(i)}_o$ are the modified versions of old and current probabilities $y^{(i)}_o, \hat{y}^{(i)}_o$:

$$y'^{(i)}_o = \frac{(y^{(i)}_o)^{1/T}}{\sum_j (y^{(j)}_o)^{1/T}}, \qquad \hat{y}'^{(i)}_o = \frac{(\hat{y}^{(i)}_o)^{1/T}}{\sum_j (\hat{y}^{(j)}_o)^{1/T}}. \qquad (3)$$

T is an hyper-parameter called temperature, which is set to 1 in the standard softmax function. We use $T > 1$ to produce *soft targets*, *i.e.*, a probability distribution more equally spread among the old classes. This is in accordance with [4], as to distill knowledge a set of soft targets with high entropy provide much more information than hard targets.

We set $T = 2$ as in the official Learning without Forgetting implementation. In Figure 4 we compare results given by our combination of losses and BCE for both classification and distillation. Table 2 shows a 7% drop of average incremental accuracy. This difference may be explained by our reasoning on the limits of CE as a classification loss for the incremental learning task. Moreover, using a sigmoid function naturally produces softer targets, since the sum of probabilities between all classes is not constrained to 1 as it is for softmax.

4.2.3 Asymmetric + BCE

Our next study consists in substituting the classification loss of iCaRL with our custom asymmetric loss (1), while keeping BCE as our distillation loss. We aim at mitigating the limitations of the CE loss (see Section 4.1) for the incremental learning task and evaluating the behaviour of the network with this intermediate approach. We also investigate whether this classification loss is able to outperform the baseline iCaRL implementation, as it did with the Learning without Forgetting algorithm.

In Figure 4 we show that the current combination of loss functions results in weaker performance than the baseline, losing 2.2% average incremental accuracy. We point out two main reasons to explain this result. The first reason is the imbalance between the classification and distillation loss contributions, that causes difficulties in learning new classes as more learning steps are taken. This imbalance is due to the fact that (1) presents a quadratic term, while BCE has a logarithmic term. The second reason is that the intuition behind the design of our custom asymmetric loss may not be valid anymore. With an increasing number of known classes, the probabilities associated to wrong labels may lose the similarity meaning that we pointed out. Therefore, they need a harsher penalization.

4.2.4 Asymmetric + L2

The last experiment with loss functions consists in our custom asymmetric loss function (1) as the classification loss, and L2 as the distillation loss. One of the issues identified in Section 4.2.3 is the penalization imbalance between BCE and (1), in favour of the former. By modifying the distillation loss, we hope to strike a good balance between the two loss components.

In Figure 4 we compare the results given by the current pair of loss functions and the combination of two BCE losses. We lose only 0.9% average incremental accuracy on the evaluation set. In fact, the plot shows that the results given by these two loss combinations are very similar.

At the end of these ablation studies, having evaluated different classification and distillation loss combinations, we acknowledge that a pair of BCE losses appears to be the most suitable choice for the iCaRL algorithm.

5. Classifier Ablation Study

We shift our focus to classifiers and evaluate several classification strategies. Our experiments aim at evaluating the performance of different classifiers and possibly finding a better alternative to the NME proposed by the authors of iCaRL. Therefore, we use NME as a baseline against which we compare all classifiers. The strategy we employ is to treat the network up to the second to last layer as a feature extractor $\varphi : \chi \to \mathbb{R}^d$, and use these features as inputs for the classifier. This is also how the authors of iCaRL implemented the NME classifier.

5.1. K-Nearest Neighbors

The first classifier we evaluate is the k-nearest neighbors (k-NN) algorithm. It is an instance-based learning method that computes the class of unlabeled samples by the majority label among its k nearest neighbors in the training set.

Both the k-NN algorithm and NME adopt a distance-based approach to classification. NME computes the mean of each class observed so far and then measures the distance between the new image and the class means, assigning the label of the nearest class mean. In contrast, k-NN measures the distance between the new sample and all available labeled samples without computing any mean. To ensure a fair comparison, we use the L2 distance for k-NN.

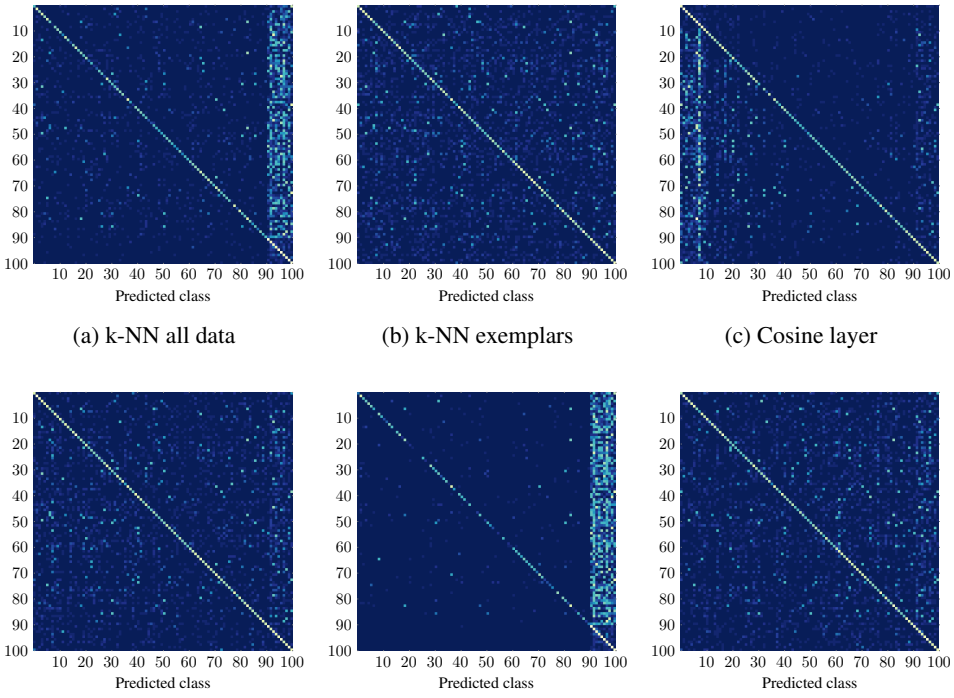
The k-NN algorithm requires a normalized and balanced dataset to perform at its best. We scale the feature set to the range $[0, 1]$ in order for each feature to be of equal importance when computing the L2 distance. Moreover, we discard the additional samples available during training and fit the classifier on the exemplars dataset only. In this case, using a smaller but well balanced dataset is better than using a dataset with more samples with a distribution heavily skewed towards new classes, as we demonstrate in the results section. This problem does not arise in NME, as it computes only one prototype per class regardless of the number of samples per class.

Implementation. After each incremental step, we extract features from the exemplar set and we fit the k-NN model on this data. We carry out a grid search to choose the best K and weight function at each incremental step, using the validation set to measure the accuracy of the models. We evaluate models with K equal to 3, 5, 7 and 9 and weight functions *uniform* and *distance*. The uniform function weights all points in each neighborhood equally, while the distance function weights points by the inverse of their distance, therefore points closer to the sample have a greater influence on classification.

We use the k-NN implementation provided by the *scikit-learn* library.

Results. We find that using k-NN for classification results in slightly worse accuracy than NME. Nevertheless, k-NN mitigates the effects of catastrophic forgetting reasonably well. We compare k-NN to other classifiers in Figure 6. The difference in accuracy between k-NN and NME may be due to the fact that computing a class mean prototype reduces the detrimental effects of noise in the data.

As we can see by comparing the confusion matrices of k-NN in Figure 5, training the classifier on the union of exemplars and training data results in more test samples to be misclassified as belonging to one of the last ten classes. In fact, the probability that the K nearest neighbors of any sample belong to the majority classes becomes higher due to their higher density in the space, resulting in worse classification performance. Using an unbalanced dataset results in an average incremental accuracy of 54.1%, which is 5.3% less than the accuracy of the model trained exclusively on the exemplars set.



(a) k-NN all data (b) k-NN exemplars (c) Cosine layer

(d) Cosine similarity (e) RF all data (f) RF exemplars

Figure 5: Confusion matrices of different classifiers on the incremental version of CIFAR-100. Training k-NN on the union of the exemplars dataset and the current training data results in more misclassification errors due to the higher number of data points belonging to the last ten classes. The cosine layer produces a confusion matrix similar to hybrid1, while the modified NME with cosine similarity retains a very good classification performance on old classes. Similarly to k-NN, the random forest classifier performs well when trained on a balanced dataset, while heavily skewing predictions towards the last ten classes when trained on all available training data.

5.2. Cosine Similarity

The second classifier we evaluate is a cosine normalization layer as described in [5] paired with a modified NME that uses cosine similarity to compute distances. The study carried out in [5] reveals that the imbalance between previous classes and new data is a crucial cause to catastrophic forgetting, and goes on to propose some solutions. We mainly focus on the cosine normalization layer and do not explore the effect of other proposed components, as they are outside the scope of the classifier ablation studies.

The authors observe that the magnitudes of the weight vectors and biases of new classes are higher than those of old classes. We are able to reproduce this harmful behaviour in iCaRL. The cosine layer solves this problem by enforcing balanced magnitudes across all classes, as illustrated in Figure 7. It replaces the last fully connected linear layer of the network, which computes the activations of its output neurons as described in (4), whereas the cosine layer measures the cosine similarity between two normalized vectors, computing the activations as (5):

$$\theta_i^T \varphi(x) + b_i \tag{4}$$

$$\eta \langle \bar{\theta}_i^T, \bar{\varphi}(x) \rangle \tag{5}$$

where $\bar{v} = v/\|v\|_2$ denotes the L2 normalized vector and $\langle v_1, v_2 \rangle = v_1^T v_2$ measures the cosine similarity between two normalized vectors.

A natural choice for the classification strategy is to use the output of the network as is. However, we may improve this method by adopting an approach similar to iCaRL. We use a modified version of the NME that takes into account the different nature of the features by using the maximum cosine similarity instead of the minimum L-2 distance between the sample and the class prototypes. In other words, we change the computation of the nearest prototype in Algorithm 1 of iCaRL to (6).

$$y^* \leftarrow \underset{y=1,\ldots,t}{\operatorname{argmax}} \langle \bar{\varphi}(x), \bar{\mu}_y \rangle \tag{6}$$

where y^* is the predicted label, t is the total number of

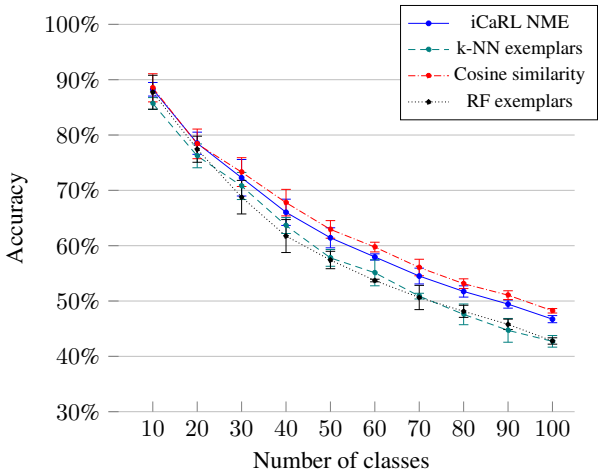


Figure 6: Comparison of the iCaRL NME baseline against other classifiers. The only alternative classifier that outperforms iCaRL is the cosine layer with a modified NME with cosine similarity. Both k-NN and random forest perform worse.

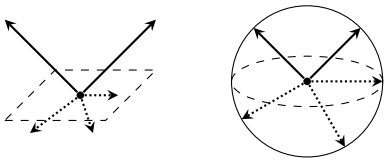


Figure 7: Illustration of the magnitude imbalance between previous classes and new data, which is a crucial cause to catastrophic forgetting, and how the cosine linear layer tackles this problem. Dotted lines represent old class embeddings and solid lines represent new ones.

classes observed so far and μ_y is the prototype vector of the y-th class.

Implementation. Firstly, we classify samples using the outputs of the cosine linear layer as they are. Then, we attempt to improve this method. Similarly to iCaRL, we extract features from the set of exemplars and compute a prototype per class. Then, we classify samples by a modified NME rule, calculating the cosine similarity between new images and prototypes.

Results. We find that using the outputs of the cosine normalization layer produces results comparable to hybrid1, only marginally higher towards the last incremental steps, reaching an average incremental accuracy of 58.4%. Figure 5 illustrates the similarities of the confusion matrices as well, showing that new classes are frequently misclassified as old classes. The relatively low accuracy of the cosine layer compared to other classifiers may be due to not having implemented the additional components proposed in [5], which are instrumental to obtain higher performance.

Nevertheless, we find that NME with cosine similarity outperforms iCaRL, as shown in Figure 6. This confirms that unbalanced magnitudes of the weight vectors between old and new classes play a role in catastrophic forgetting, and that the cosine layer effectively eliminates the bias caused by the difference in magnitudes, increasing the classification accuracy. The average incremental accuracy of the cosine similarity model is 63.9%, which is 1.2% above iCaRL.

5.3. Random Forest

The last classifier we test is a random forest [1], an ensemble learning method based on decision trees. A random forest exploits the insight that an ensemble with many relatively uncorrelated models has better predictive capability than any of its components. This set of models is built by introducing randomness in their construction through bootstrap aggregation (*bagging*) and feature selection, effectively reducing the variance of the estimator. The prediction is given by an average across all classifiers. We are interested in seeing how this model differs from the other distance-based classification approaches.

The decision trees in the ensemble are built by repeatedly sampling with replacement the training data, *i.e.*, extracting bootstrap samples from the current batch of classes. Moreover, each decision tree is built using a random subset of $\sqrt{d}$ features, with d the total number of features. This number empirically gives good results for classification tasks. The resulting decision trees are relatively uncorrelated, reducing the overall variance of the ensemble. As we only use a subset of training data for each decision tree, bias is likely to increase. However, the concurrent decrease in variance usually results in better classification accuracy, good generalization ability and robustness with respect to noise.

Decision trees are very sensitive to imbalanced training data, so we expect a random forest to perform poorly if trained on the union of the exemplars set and the current training data. Therefore, we also build a random forest trained only on the exemplars set.

Implementation. After each incremental step, we fit the ensemble on features extracted from the exemplars set. We carry out a grid search to choose the best number of estimators among 100, 200, 500 and 1000, and the minimum number of samples required to split an internal node among 10, 20 and 50. The latter parameter is useful to limit overfitting in the decision trees.

We use the random forest classifier implementation provided by the *scikit-learn* library.

Results. We find that the random forest trained exclusively on the exemplars set results in worse accuracy than NME. The average incremental accuracy is 59.4%, which is the lowest score among all classifiers.

As we expected, training a random forest on the union of exemplars and training data results in a higher misclassification error, with predictions skewed towards the last ten classes as we can see in Figure 5. The average incremental accuracy of this model suffers from a significant drop to 44.3%, which is 15.1% less than the one trained only on the exemplars set.

6. Beyond the Baselines

We dedicate this last section to studying more deeply some of the existing limitations in the incremental learning frameworks, and we propose some measures to attempt to mitigate them.

6.1. Distillation Targets Study

Every time the network receives a new batch of classes, we take a further incremental training step. In our setting, we need ten steps for our final model to learn all the classes contained in the original dataset. Targets for the distillation loss at step N are obtained from the network trained at step $N-1$, *i.e.*, on the previous batch.

Consider a prediction node in the last fully connected layer, related to a class associated to batch $M < N-1$, where M refers to any step older than the previous one. We hypothesize a significant difference between probabilities assigned by the network $N-1$ and the ones given by the network M. In order to prove this, we save the trained network at each incremental step, which is used in the next task to produce distillation targets only for the classes it was specifically trained on, *i.e.*, excluding the exemplars set. The loss function compares these scores with the one produced by the network at step $N-1$.

In Figure 8, we show the difference between distillation targets produced by the baseline framework versus the ones given by our new policy, which we refer to as *multi networks*. The scores are related to the ninth split, and all predictions are taken from a sample with label 0. We may expect that both policies can effectively recognize the correct label of the sample. However, we can observe a more spread distribution with respect to the one given by the last network. Since each model is an expert in recognizing data it was trained on, its predictions are more confident and therefore closer to 0. In some cases, the target gap between the policies can reach 6 order of magnitudes.

Using this approach, we record the scores given by the modified algorithm with respect to the ones given by iCaRL. We find that iCaRL outperforms our procedure by 1.2% average incremental accuracy over the test set. A possible cause is that new distillation targets are characterized by very large variance with extremely low values, with probability distribution unevenly spreaded. The authors of [4] already studied this problem and its solution, which we applied to our case by adding a T hyper-parameter in the sigmoid function (7), producing a soften targets distribution.

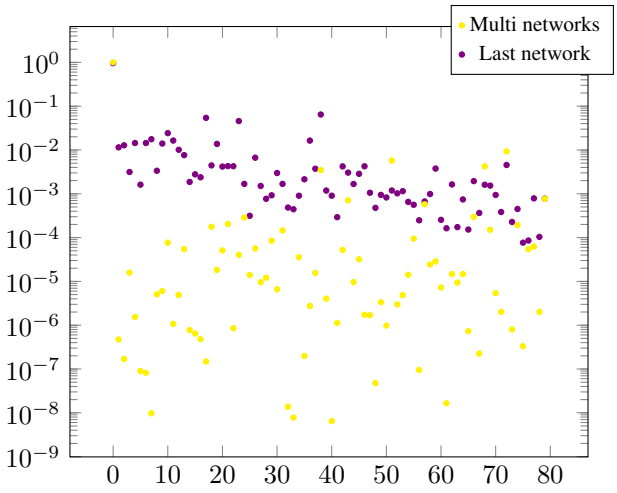


Figure 8: Distillation scores given by the two policies. They are presented on a logarithmic scale for better visualization.

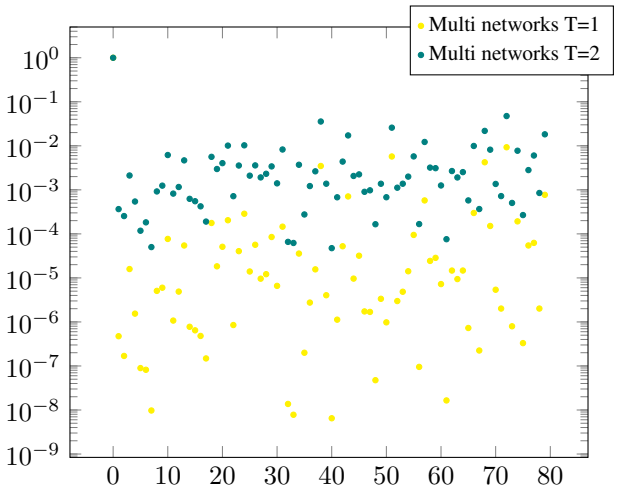


Figure 9: The application of T to the sigmoid function effectively reduces the amount of variance in the distribution of the multi networks targets.

$$\sigma(x) = \frac{1}{1 + \exp(-x/T)} \tag{7}$$

This is a modified sigmoid function which results in targets characterized by smaller deviation, as we show in Figure 9. We also jointly finetune the temperature T and the learning rate, which we find to give better results at $T = 2$ and learning rate equal to 3.

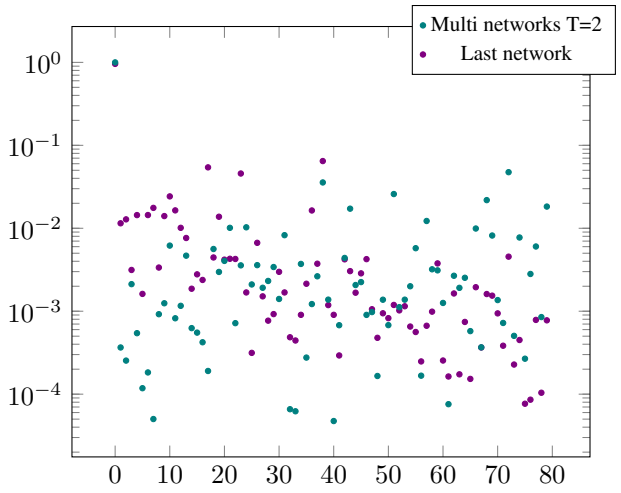


Figure 10: Soften targets distribution still results more irregular than the one provided by last net policy, as these targets remains more precise.

Method	Avg.
Last network (iCaRL)	62.7%
Multi networks $T = 1$	61.1%
Multi networks $T = 2$	63.2%

Table 3: Comparison of iCaRL accuracy scores with the modified algorithm using old nets distillation targets and temperature parameters in the sigmoid layer.

Accuracy scores on the test set are presented in Table 3. Figure 11 shows a comparison between our multi networks policy and the iCaRL baseline. We can observe a slight increase in accuracy due to the fact that we are giving more precise and meaningful targets to the distillation function. Moreover, the introduction of a temperature parameter T in the sigmoid layer applied to old classes leads to an improvement.

6.2. Features Representation Drift Study

In the incremental learning protocol, we assume that only a fixed amount of data from the old classes is stored, in order to maintain the memory footprint bounded. As a consequence, training steps are performed over unbalanced class distributions. One way to address this problem is to use a knowledge distillation loss. We think that another interesting way of tackling this issue is to try to maintain as much as possible a similar feature representation between the network at step $N-1$ and the one at step N. In order to do this, we shift our focus to the outputs of the last layer of the feature extractor, which is used to classify images at test time. Firstly, we demostrate the existence of a residual

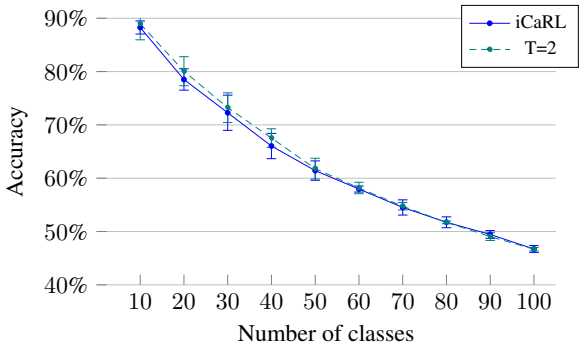


Figure 11: Incremental accuracy over the test with distillation targets obtained with multi net approach. T is set to 2 while the learning rate is equal to 2. Our new proposed approach outperforms iCaRL by 0.5% of incremental average accuracy.

features representation drift. Then, we propose an approach that tries to limit it.

To classify unseen data, iCaRL computes the mean of exemplars for all known classes. At the end of each incremental learning step, we store the class prototypes, which we suppose are representative of the features learned by the network. At step N we compare the prototypes of the network with those learned at step $N-1$. The difference defining the drift is computed as mean square error (MSE) between all 64 entries of the mean of exemplars vectors. These 64 entries correspond to the output nodes of our feature extractor. All entries are weighted by a coefficient, defined for each label, which states the importance of a feature for that specific class, since we are only interested in avoiding drift of meaningful features (see Appendix A).

In Figure 12 we show the drift between the representation learned at step 0 versus those learned at step 1 and 9. We can observe significant error between two consecutive steps, and we can also see that it propagates and increases over the networks.

To mitigate the drift, we add a contribution to the loss which tries to minimize the distance between important features. We choose a smooth L1 loss function (8) weighted by a parameter α which defines the importance of this contribution:

$$\text{loss}(x, y) = \frac{1}{n} \sum_i z_i * \alpha \tag{8}$$

$$z_i = \begin{cases} 0.5(x_i - y_l)^2, & \text{if } |x_i - y_l| < 1 \\ |x_i - y_l| - 0.5, & \text{otherwise} \end{cases}$$

where x_i is L2 normalized feature representation of image i of the trained network, while target y_l is the last network prototype corresponding to label l. α is an hyper-parameter to be tuned along with the learning rate. This loss behaves as L2 if the it falls below 1, and as L1 otherwise.

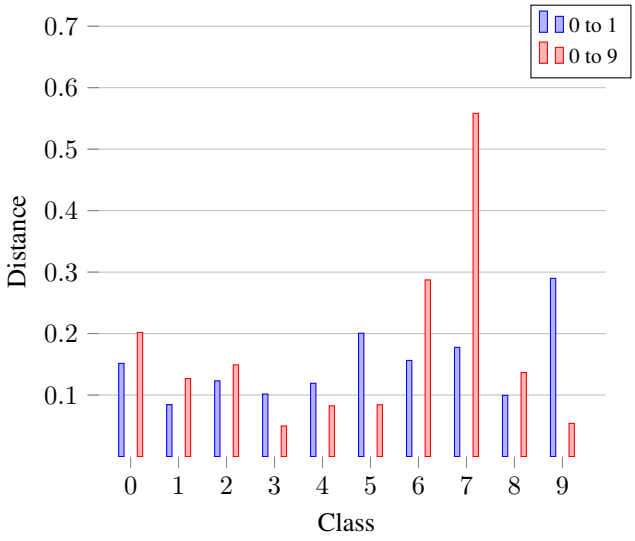


Figure 12: Weighted MSE drift from step 0 to 1 and 0 to 9. Inspecting the results, we confirm the features representation drift hypothesis. Moreover, we observe how the drift not only propagates, but also amplifies from step 1 to 9.

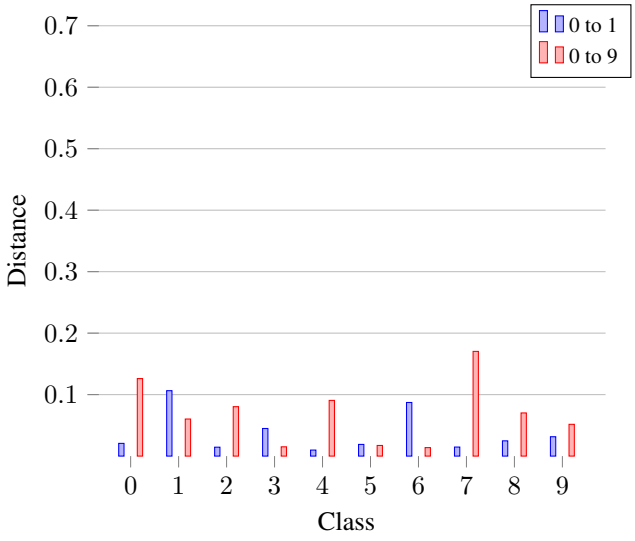


Figure 13: Drift reduction from net 0 to 1 and 0 to 9, comparing features representation with and without smooth L1 loss contribute over the last layer of the feature extractor.

We jointly tune α and the initial learning rate, which we set to 10^{-4} and 0.3 respectively. We train the network for 80 epochs, and decrease the initial learning rate by a factor of 5 after 75 epochs.

In Figure 13 we can quantitatively observe how the loss contribution that we introduced is able to mitigate drift over

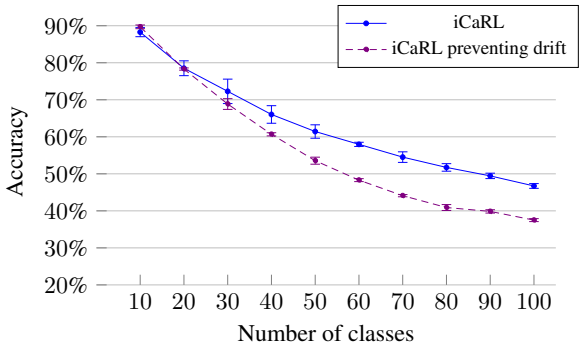


Figure 14: iCaRL scores with feature drift distillaton loss term. Compared to the plain iCaRL framework, our implementation leads to worse incremental accuracy. This is because of the additional loss contribute that, while preserving the old representation, also prevents the network from adequately learning the new task.

features. The adopted measure is effective, but results in a drop of accuracy over the incremental training steps, as we can see in Figure 14. This can be explained by the fact that without properly balancing the selected hyperparameters, preventing the drift also means preventing an effective learning. Nevertheless, we consider this a promising procedure, since the model may produce a more general feature representation, trying to balance with the learning and distillation needs. Despite this, due to lack of time and computational resources, we are not able to explore the problem any further.

7. Conclusions

This work aims at implementing some of the existing frameworks developed for multi-class incremental learning, such as Learning without Forgetting and iCaRL, comparing them to other baselines. Moreover, it studies the effects of different combinations of classification and distillation loss functions and alternative classifiers, comparing distance-based algorithms such as k-NN to more complex models such as random forests.

This paper analyzes some new problems beyond those addressed by the baseline algorithms, with both a theoretical and practical approach. Firstly, we show how the distillation targets change over different incremental learning steps. Then, we try to mitigate this behaviour by giving more precise soften targets from multiple networks.

Finally, this study shows how distillation loss does not fully address the features representation drift, and tries to reduce the residual drift by means of an additional loss contribution on the last layer of the feature extractor. Unfortunately, we could not obtain an improvement in average incremental accuracy as we have not yet found the

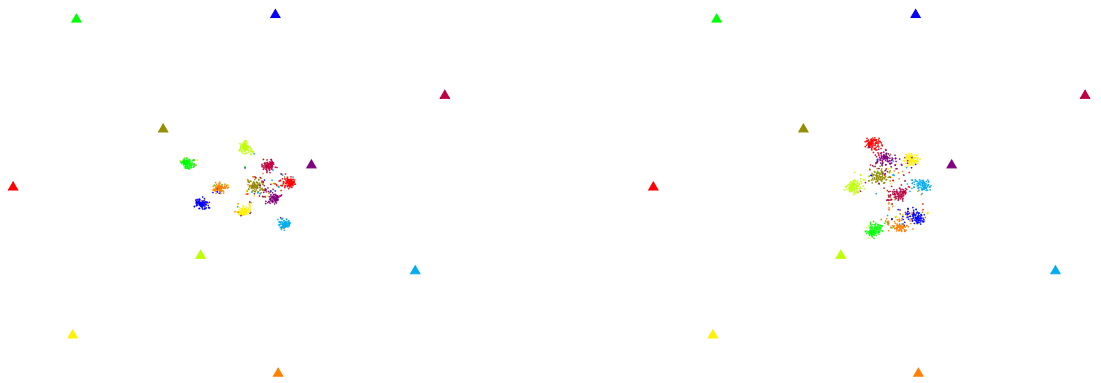


Figure 15: 2-dimensional t-SNE features drift representation. On the left we have feature representations given by network trained at step 2, while on the right features of the network trained at step 10. Dimensionality reduction is applied in order to plot the points in a 2-dimensional space. In both plots, each representation is compared to its corresponding class protoypes, produced by the network trained at step 1. We represent class prototypes with triangles.

right balance between distilling knowledge and learning new classes. Notwithstanding its limitations, this work proposes a promising approach for effectively tackling incremental learning problems.

A. Smooth L1 Loss and Features Coefficients

In Section 6.2, we explain how we try to mitigate representation drift by adding a loss contribuion over the outputs of the features extractor. We use a smooth L1 loss function (8) to achieve this goal. We choose a linear loss since we want to penalize less this type of errors. We experimented that with a quadratic function such as MSE, in order to avoid divergence, the learning rate needed to be too low to effectively learn new tasks. The quadratic contribution between 0 and 1 is necessary to have lower gradients in correspondence of the minimum, again with the same goal of avoiding divergence.

Another logic we decided to implement is to define some weighting coefficients for the penalized features scores. For each image with label l, we use as target the corresponding prototype produced by the last trained network. Given prototype vector μ_l, coefficients for correspondent images are defined as:

$$64 * \frac{\mu_l}{\sum_i (\mu_l)_i} \tag{9}$$

where 64 is the total number of entries, and $\sum_i (\mu_l)_i$ is the sum of all entries of the prototype. In this way, we are able to give a coefficient, representing importance, to each feature based on its label.

References

[1] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE.

[4] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531*, Mar. 2015. arXiv: 1503.02531.

[5] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a Unified Classifier Incrementally via Rebalancing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 831–839, Long Beach, CA, USA, June 2019. IEEE.

[6] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. page 60, 2009.

[7] Z. Li and D. Hoiem. Learning without Forgetting. *European Conference on Computer Vision (ECCV)*, 2016.

[8] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, May 2019.

[9] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, Honolulu, HI, July 2017. IEEE.