Incremental Learning in Image Classification

TA: Fabio Cermelli (fabio.cermelli@polito.it)

OVERVIEW

The ability of neural networks to learn new concepts over time is still very limited. Recently, different works explored how to add new classes to an already trained model and one of the most effective approaches is knowledge distillation. In this project, you will start by implementing and studying the knowledge distillation strategy, analyzing the different choices that can be made and assessing which are the most effective. After having familiarized with these algorithms, it will be your turn to propose some modification, based on your findings, to improve the model.

GOALS

1. To get familiar with Incremental Learning and Knowledge Distillation strategies.
2. To replicate the experiments proposed in ICaRL on the CIFAR-100 dataset, in the setting where every training step is made of 10 classes.
3. To make an experimental study regarding the performance of different losses and classifiers.
4. To propose, implement and test a variation for the project.

REQUIREMENTS

For this project, we will make the following simplification: you will use a model made by two components: a feature extractor *f* and a classifier *g*. Commonly, the feature extractor is a Convolutional Neural Network (in this project we will use the Resnet-32, as described in ICaRL). For the classifier, instead, we focus on two choices: a fully-connected layer or a non-parametric classifier, such as Nearest Class Mean or Nearest Mean of Exemplar (see, again, ICaRL).

The model can be trained using different losses. A common choice is to use a *classification loss* to learn the new classes, plus a *regularization term* to prevent the forgetting of previous knowledge, which in our case will be the distillation loss. Different choices for the losses can be made and each one provide its own advantages and disadvantages.

The experiments should be run on CIFAR-100, by using 10 classes per learning step. That is, you should randomly divide the CIFAR dataset on 10 set of classes, which are then learned on different training steps. Defining the validation set is forbidden to use samples from previous classes (unless it is part of the method). In general, the validation set should contain only samples of the classes belonging to the current learning step. The test set should include all the classes seen in current and previous training steps (e.g. after step 4, you will test on the 40 known classes).

To have a fair benchmark you should run every method on the three different random splits (division of the dataset on 10 sets of 10 classes) of the 100 classes. Different splits give very different results, so be aware that results would be meaningless without using the same splits across different methods.

The steps required for this projects are:

1. Implement the fine-tuning baseline in PyTorch (don't use any distillation loss, but only the classification one). This will be the starting point.
2. Implement Learning Without Forgetting and ICaRL as detailed in ICaRL paper. Follow the paper for all the implementation details. ICaRL will require to store and replay some samples (called *exemplars*) of previous classes. Since they are often used in recent works, in the following use the exemplars for your study. You can start with the hyperparameters declared in the paper, and optimize them further to get better results.
3. Make an ablation study considering at least three different combinations for classification and distillation losses. For a combination I mean the couple of a classification and a distillation loss, e.g. ICaRL is a combination made of the BCE loss and the ICaRL distillation. Details of the possible choices are described in the VARIATIONS section. Remember to select properly the hyperparameters (hint: start from the parameters obtained in the previous step and optimize them for the new losses)
4. Make an ablation study considering three different choices for the classifier. As for step 3, different choices are detailed in the VARIATIONS section. For this step you can use your preferred training losses (but remember to motivate your choice). Changing the classifier requires probably to also tune the hyperparameters.
5. Propose your own modification. If you are stuck and you don't have any idea, you can take inspiration from some recent works that you can find in the LITERATURE section.

The deliverable of the project are:

- Working pytorch scripts for all the required steps.
- Write a complete pdf report. The report must contain: a brief introduction, a related works section, a methodological section for describing the algorithms you are going to use, an

experimental section with all the results and discussion. End the report with a brief conclusion.

VARIATIONS

You can refer to this section for ideas of possible variations of the project.

Classification and Distillation losses

In literature we can find mostly two classification losses: Binary Cross Entropy (BCE) and Cross Entropy (CE) losses. Since they are widely accepted I suggest to try out these two losses. However, if you find any other classification loss that you want to try, feel free to use it and provide the reason of your choice in the report.

Differently from the classification loss, in the last few years a lot of distillation losses has been proposed. In the following the most common choices will be detailed.

- Softmax with temperature distillation. This loss has been proposed by Hinton et al. and, in the following, Li and Hoiem have been the first to apply it in incremental learning. The loss is very simple since it is similar to the cross entropy loss but it is computed among the probability obtained by the response of a network frozen after the previous training step (soft target) and the probability obtained by the new network. Take a look to the cited papers for the equations and the rationale behind them.
- Less Forget Constraint. The loss has been proposed in Hou at al. and minimizes the cosine divergence of the features extracted by the old model and the new one.
- L2 loss applied among features obtained from the old network and the new one.
- L1 loss applied among features obtained from the old network and the new one.
- Kullback-Leibler divergence applied among the probabilities obtained from the old network and the new one.

Classifiers

There are a lot of classifiers you can try in this project. The most simple choice is to use a linear classifier (a fully-connected layer) which is also the most common in literature. However, an alternative exploited in ICaRL is NME, where you take the mean of the features for the samples of a class, and you predict the class as minimum Mean Squared Error between the feature of the query sample and the class means. Other possible choices are:

- An SVM trained on the features. Training an SVM after a neural network is not straightforward, but can improve the performances if it is done properly.

- A cosine linear layer (as proposed by Hou at al.). This is similar to a fully connected layer, but the weights and the features are normalized before computing the dot product. In this way, we compute the cosine similarity among the sample and the weights.
- Interestingly, Wu et al. proposed a Bias Correction Layer, where the scores are normalized to remove the bias towards the new classes.
- E. Belouadah et al. proposed an alternative mechanism to remove the classification bias which is based on normalizing the output using the statistics of the old network and of the classes.

The list is not exhaustive, thus, as with the losses, feel free to propose your own classifier but remember to motivate it in the report (you can point out a paper which uses it or explain the intuition you had to use it).

LITERATURE

In this section, I make a list of **interesting paper** that came out in the last few years. Use this list only as a way to get new ideas (this is intended as a survey of existing methods).

- Continual learning with Deep Generative Replay, Hanul Shin, Jung Kwon Lee, Jaehong Kim, Jiwon Kim (NIPS 17)
- Dreaming to Distill: Data-free Knowledge Transfer via DeepInversion, Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, Jan Kautz
- End-to-End Incremental Learning, Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, Karteek Alahari (ECCV 18)
- Semantic Drift Compensation for Class-Incremental Learning, Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, Joost van de Weijer (CVPR 20)
- Learning without Memorizing, Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, Rama Chellappa (CVPR 19)
- Overcoming Catastrophic Forgetting with Unlabeled Data in the Wild Kibok Lee, Kimin Lee, Jinwoo Shin, Honglak Lee (ICCV19)
- Progressive Neural Networks, Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, Raia Hadsell
- Adversarial Continual Learning, Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, Marcus Rohrbach
- Conditional Channel Gated Networks for Task-Aware Continual Learning, Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, Babak Ehteshami Bejnordi (CVPR 20)