Object Recognition and Computer Vision - TP3

Alain Riou
Ecole Normale Supérieure de Cachan
alain.riou@ens-paris-saclay.fr

This document details the methods used to perform the task of classifying birds for assignment 3 of Object Recognition and Computer Vision course. The whole code is available on GitHub [1].

1. Model

The model I used is ResNext101 [2]. In fact, using this model instead of creating my own architecture has multiple good points:

- The network is fully implemented in PyTorch.
- Exists pretrained models on PyTorch, so I did not have to spend days training the whole network.
- The network is known to achieve respectable results on ImageNet (84.3% top 1 accuracy).

In practice, I did not train the whole network myself, but only the last fully-connected layer. This enabled to prevent prohibitive time-consuming computing, as it is quite fast to train only one layer. Moreover, it would have been impossible to store all the parameters and gradients in memory.

All the models have been trained using Adam, which provided better results than Stochastic Gradient Descent.

2. Improvements

The main problem with the used model was that accuracy achieved was much higher on validation set than on test set. My hypothesis was that maybe the validation set was not general enough, and too close to the training set. As this unexpected behavior seemed to be overfitting in a sense, I decided to implement different methods to prevent it.

2.1. Data augmentation

Data augmentation is a classical method to prevent overfitting. In practice, the idea is to add noise to the images the network is trained on, in order to make it more robust. First, input images are resized to 320px, then randomly cropped to a square of 288×288. The resulting images are then eventually flipped horizontally. Finally, to increase robustness to occlusion, a random rectangle from the image is erased. All this transforms are available in `torchvision.transforms`.

2.2. Learning rate scheduling

I used a learning rate of 0.1 for my experiments, as it enables the network to converge fast. However, it provides quite unstable results. Therefore I used a learning rate scheduler, which divides the learning rate by 10 when the accuracy on validation set does not increase during 10 epochs.

3. Noisy student

I implemented a semi-supervised technique used by the current best classifier on ImageNet: self-learning [1]. It consists in periodically adding unlabeled data to the original dataset during training, with network's predictions as labels for the new incoming data. Supplementary data is taken from the provided test set, and chosen by increasing uncertainty. The uncertainty measure used is $\frac{score_{2nd_{best}}}{score_{best}}$.

4. Results

Table below shows the accuracy obtained by my model, and details the influence of the different implemented improvements.

Improvements	Accuracy (%)
None	61.9
DA	69.0
DA + scheduler	77.4
DA + scheduler + noisy student	**78.1**

References

[1] Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with Noisy Student improves ImageNet classification. nov 2019.

[2] I. Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. may 2019.

[1] https://github.com/aRI0U/recvis19_a3