Object Recognition and Computer Vision : Assignment 3

Thomas Fauré
Master MVA
ENS Cachan
thomasfaure87@gmail.com

Abstract

We want to train a model in order to classify 20 species of birds. The training base is relatively small about a thousand pictures and the validation base about a hundred.

1. Introduction

Due to the small number of data, a hand made convolutional neural network does not perform very well. We will try to improve the performance of the classification.

1.1. Preprocessing

Our first idea was to increase our training base by transforming the data randomly (withe Horizontal/VerticalFlip, rotation, affine, etc, ...) and increasing the number of epochs in order to spread our transformations. After looking at the pictures shows us that the datas are of different sizes, sometimes the background is similar while the birds are different or the birds are in different positions or hidden by something. We have resized our pictures. The transformations applied to our training base helped our CNN to improve its performances. Then we tried to reduce the impact of the background in our classification (two different birds can have a similar background and our network may classify them together). We decided to segment our pictures using a mask RCNN. We tried this method too lately and we did not have the time to implement it.

1.2. First idea : hand-made CNN

The documentation encouraged us to increase the width and the depth of our network. Despite the large number of layers and hidden units the result on our validation test is not very good and we did not even try to submit it on kaggle.

1.3. Transfer learning

We decided to move to pretrained models. We choosed three of them: vgg16 with batch normalization, googlenet, residual network (resnet152). Those models have been trained on a database different from our database. We first froze a part of the layers according to the documentation, and we modify the last linear layer in order to match with our twenty classes. we obtained:

Network	validation	Kaggle
vgg16	82%	65 %
googlenet	79 %	65 %
resnet152	88 %	69 %

Then we thought that freezing layers does not helps our models to learn well our data (indeed it has been trained on an other database the weights are not well-adapted to our case). We decided to unfreeze the layers of resnet152. We obtained:

Network	validation	Kaggle
resnet152	91 %	72,9 %

1.4. Concatenate models

A last attempt to improve our results was to concatenate a resnet152 and an inceptionv3. The idea was to freeze one of the model (inception) and not the other one in order to apply pretrained weights (trained on ImageNet) and make them more specify for our example. We obtained :

Network	validation	Kaggle
mixed model	90 %	71,6 %

1.5. Scheduler and Optimizer

We used a learning rate schedule which tried to adapt the learning rate during the training by reducing the learning rate according to our schedule. We tried different optimizers like Adam, Adagrad or SGD. We have the best results for SGD with a momentum to 0.9 (But most of the documentation stated that Adam is better).