Assignment 3: Bird recognition

Raphael Reme

Abstract

The goal of the assignment is to classify bird images onto twenty different classes. Students can rely on any algorithm they can think of. The provided dataset is composed of images of 20 different birds. Using other data is allowed as soon as it is not labelled data on several classes of birds. The use of pretrained model on forbidden data is also forbidden.

1. Introduction

In order to reach quickly good performance in this task, it's important to see what is the baseline in the domain. We have here only a classification problem over images. It seems that CNNs are therefore recommended. Using pytorch we can access the recent successful architectures of CNNs like ResNets, GoogLeNets and so on...

As the dataset is quite small (only 1200 images labelled) big architectures are likely to overfit the train set. There is some few things that we could mix to prevent overfitting. We can first choose smaller architecture, then we could try to augment the dataset and we could rely on pretrained architectures on a much bigger dataset. (We could also use self-supervised methods to augment our dataset. But I did not have the time to try it.)

2. Architecture

I tried several architectures. Finally it seems that pretrained architectures work better. The ResNet50 architecture seems also to performs better than others. It overfits less than the bigger ResNet architectures. The best performances I got was with a ResNet50 archictecture pretrained over ImageNet where the last fully connected layer was reset to predict 20 classes and with its 2 last main layers learnable(layer3 and layer4 in the pytorch model).

3. Data preprocessing

In order to do batch learning images needed to be all of the same size. And in order to use the ResNets architectures (or others), this size had to be bigger than (224, 244). Moreover as the pretrained models were trained with a data normalization (From Imagenet), we ought to do the same normalization. This could be the default image preprocessing.

But in fact it's not at all sufficient because the testing set is much more messy than the labelled set (training + validation). Thus we should normalize the images before training and testing.

One can see that the images doesn't have the same ratio at all. I try to do a ratio normalization which would basically center crop the image in their bigger axis and normalize the image ratio. (This make the assumption that the bird is never on the edges of the image.) This improves a lot the performances.

But it seems that we could do better. Birds size in the image depends a lot from an image to another. I therefore try to use a bird detector. I take a pretrained fasterRCNN on the Coco dataset to find the bounding box of the bird in the image. From that bounding box I choose to crop a surronding rectangle (almost squarred in order to keep a normalized ratio.) Even if on some images birds were wrongly (or even not) found it still improves performances!

I could have mixed things a little bit but I hadn't have enough time to go further!

4. Data augmentation

I only use some basic data augmentation with on the fly random transformations for each batch.

I tried horizontal/vertical image flip, random resize/ratio and adding some gaussian noise. These augmentations has been effective (Except the gaussian noise). But I could have probably do some more!

5. Results

My best results were reached with a ResNet50 with the preprocessing described before. (I resample the train/validation set because the validation set seems to be a little bit too simple and to small) It reaches around 90 percent of accuracy on my validation set, and achieves around 80 percent of accuracy on the tested part of the test set.

There is still room for improvements: I could have handled better the wrong birds detection. And I would have like to mix some models in order to make a prediction.