Assignment 3 : Bird image classification competition

Justin Kossivi AYIVI
Université Paris Dauphine - PSL
kossivi-justin.ayivi@dauphine.eu

Abstract

Bird image classification competition is a kaggle challenge based on Caltech-UCSD Birds-200-2011 bird dataset. The objective is to produce a model that gives the highest possible accuracy on a test dataset containing the same categories. We achieved the performance of 81,29% on the mystery testing set.

1. Introduction

The dataset we were supplied with is a subset of the original Caltech-UCSD Birds-200-2011 dataset, reduced to 20 classes, instead of the original 200 bird species. There are 1082 training images and 103 validation images. This paper presents the methods we used to classify it into one of the 20 species of the subset.

2. Preprocessing

Unlike Image Net, the images have a high intra-class variance and a low inter-class variance. This makes prediction difficult. The images have different shapes, and within the same species, the position and size of the bird varies. The illumination and background are not as consistent within the different pictures. In some pictures we see that the birds are hidden behind twigs or that they are making a specific movement (a bird catching a snake).

Segmentation of the image is necessary because in most images the birds are always in the center of the image while in others, other objects distinct from the birds are in the foreground. Although bounding boxes are available in the original dataset, they were not provided for the competition. To address this issue, we had considered a pre-trained Mask R-CNN model but did not have time to implement it.

We finally focused on deep learning methods. We need the size of the dataset be very large to make deeper methods works very well. According to this, we had used data augmentation techniques.

3. Data Augmentation

We simply used the transformations RandomRotation (45 degrees), RandomHorizontalFlip and RandomResizeCrop (resize an image size randomly to 224 $\times$ 224) available in the torchvision.tranforms module in Pytorch to make our networks more robust. Finally, we used regularization techniques to keep the model robust and avoid overfitting.

4. Transfer learning

Using pretrained models is a good way to yield good performance. The first network that I used was ResNet34 to which I added one FC Layer. I trained this model, using 5 epochs and a Stochastic Gradient Descent Optimizer with 0.9 value of momentum and 10^{-4} to 10^{-3} value of learning rate according to faster or slower convergence of the optimization algorithm. I obtained accuracy turn around 65,16%. Intuitively one could imagine that the deeper models will allow to have better results. In a second step, I used ResNet 152 and DenseNet151 to which I added different layers at the end: one FC layer on both models and two dropout layers only on the DenseNet (with a rate of 0.9 for the first and 0.5 for the second) to avoid overfitting. I get 67,09 % of accuracy with Densenet and 81,29% of accuracy with Resnet152 on the mystery test set which is an improvement of the first Resnet Accuracy. That's shall good better. I summarize the results in the table below after training these models.

	Training	Validation	Kaggle
Resnet-152	$85,21\%$	$90,29\%$	$81,29\%$
Densenet-161	$85,33\%$	$79,37\%$	$67,09\%$
Resnet-34	$80,00\%$	$76,32\%$	$65,16\%$

Table 1: Accuracy using transfer learning

These models could be stacked to get a model with better accuracy and less variance, but I did not have time to implement this.