Object recognition and computer vision 2020-2021
Assignment 3 : Image classification

Avigail Zerrad
Paris Dauphine & ENS
avigail.zerrad@dauphine.eu

Abstract

This report will give an overview of my work for the Kaggle challenge on the Caltech-UCSD Birds-200-2011 bird dataset. The task was to produce a model that improves the accuracy on a test dataset with the same categories.

1. Introduction

The training set only contains 1087 images, which is not enough for the CNN to learn to high accuracy. Since the size is not sufficient to train an entire CNN from scratch, we will use *Transfer learning*. The idea is to use a model pretrained on a very large dataset (e.g. ImageNet) and to *transfer* its knowledge to our smaller dataset.

2. Data preprocessing

We first apply some transformations on the data.

- Data augmentation : we create new artificial data by applying random transformations to the **training** data. This technique is useful to avoid overfitting and help the network to be invariant to translation, viewpoint, size or illumination.

- Resizing the input : since we use pretrained models, we have to modify the size of the images. We set the input size to 224x224.

3. Choice of the pretrained model

There are a lot of models pretrained on ImageNet, but I decided to focus on Resnet101.The architecture of ResNet is the following : four main stages of convolution and a fully connected layer. After importing the model, we need to reshape the last layer to have the same number of outputs as the number of classes in the dataset (20 classes in our case).

4. Applying transfer learning

There are two main scenarios in Transfer learning : feature extraction and finetuning. The first one consists in updating only the final layer weights and the second one consists in finetuning model's parameters somewhere earlier in the network. This works because we observed that the first layers of a CNN contain generic features that are not specific to a particular task.

The main question is then what strategy to choose, and in the case of finetuning, what layers to update.

After a few tries, I decided to unfreeze 'layer3', 'layer4' and 'fc' and to update only the weights of these layers, using SGD with a learning rate lr=0.01. I trained my model on 20 epochs, with a batchsize of 20.

5. Obtained results

I obtained an accuracy of 92% on the validation data when I trained the model that I exposed above. For the test dataset, I have an accuracy of 74,19%, which is much better than the accuracy we had with the simple network of the beginning.

6. Conclusion

With transfer learning, we trained a good CNN in reasonable time. The choice of ResNet101 as pretrained model was motivated by the high accuracy we could obtain thanks to the large number of layers and without the vanishing gradient problem.

References

[1] Pytorch tutorial, Finetuning torchvision models, 2017

[2] Will Koehrsen, Transfer Learning with CNN in Pytorch, 2018