Kaggle competition: Image classification

Joséphine Raugel
Mines Paristech
MVA
josephine.raugel@mines-paristech.fr

Abstract

In this paper I present the construction of a computer vision model designed to label the different kinds of birds included in the Caltech-UCSD Birds-200-2011 dataset, in the context of the following kaggle competition: https://www.kaggle.com/c/mva-recvis-2021/.

1. The transfer learning model

Considering the relatively small size of the training/testing/validation datasets, I opted for transfer learning rather than training from zero a model. I selected and fine-tuned several models pre-trained on ImageNet, such as VGG16, GoogleNet, Resnet50, Resnet101, Resnet152, CoatNet and ResNext10132x8d. Once fine-tuned, I found the most accurate model for my dataset to be ResNext10132x8d, which might come from its advanced architecture: the network is inherited from Resnet's strategies and is constructed by repeating a building block that aggregates a set of transformations with the same topology [5].

I replaced the final fully connected layer of my pre-trained model by alternatively 1, 2 or 3 linear transformation layers. The adding of only 1 linear transformation resulted in the most accurate model once trained, which might be due to the small size of the dataset, not suitable for the training of too many parameters.

As a result of hyperparametrization, I chose a Stochastic Gradient Descent optimizer with a momentum of 0.5, which allowed better training than momentum with higher values, as the batch size I used was quite large (ranging between 20 and 64).

2. No Layer freezing

After trying several levels of layer freezing, I found out that training the pre-trained model without freezing any layer resulted in a slighlty yet consistently more accurate fine-tuned model.

3. Data Augmentation

I performed data augmentation, in order to increase the variety of data, thus getting a more comprehensive set of available information to train my model and avoid overfitting. I carried out transformations in the form of randomized cropping, randomized rotation up to 10 degrees, randomized horizontal flip and contrast, color, saturation and brightness variations[4].

4. Regularization

In order to avoid overfitting, I used random dropout, which was proven effective by previous research [1]. I also added a weight decay to penalize complexity and improve generalization [2]. Finally, two final ways I used to prevent overfitting were learning rate adaptation and early stopping, based on the work of Sovit Ranjan Rath [3].

References

[1] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. 1

[2] A. Krogh and J. Hertz. A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1992. 1

[3] S. R. Rath. https://colab.research.google.com/drive/1krrz-vvfpxushk3jfcjbtyuiuowl8vw0?usp=sharingscrollto=lb9cxgakk-jg. 2021. 1

[4] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *J. Big Data*, 6:60, 2019. 1

[5] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. 1