MVA - RecVis - TD3 report

Timothée Darcet
timothee.darcet@gmail.com

Abstract

I have implemented models for the bird type recognition task. Although I did not achieve a very good test accuracy, I acquired some precious experience.

Introduction

My approach consisted of many different concepts put together: - cropping - Transfer learning - distributed training - data augmentation - dropout - feature visualisation for model understanding - ensemble methods - test-time augmentation

1. Transfer learning

A very common technique in this kind of challenge, I used ResNet50, ResNet152, VGG16, denseNet121, GoogleNet and ResNeXt101. In addition to classical transfer learning, I occasionally trained the two last layers of the model instead of only the last one.

2. Cropping

I used an ImageNet-pretrained Mask-RCNN to detect bird bounding boxes, then cropped to the minimum bounding box encompassing all bird detection with more than 0.8 confidence + the most confident detection (because on some image the bird was detected, but with low confidence).

3. Distributed training

I took advantage of the fact that my university (École polytechnique) gives its students an ssh access to the computer in the lab rooms to learn about distributed training with Pytorch. I could access 200 computers with Quadro P2000 GPU, and I used at most 100, which gave me a lot of computing power. Of course, this challenge did not require computing power, but it was still a very interesting exercise, and it allowed me to test a few things. In particular it allowed me to train with the full dataset as batch size, which made the process very stable. I would have liked to use this power to do unsupervised training (contrastive [1], pseudo-labeling, image autoencoding), but distributed learning was full of technicalities and setting up the pipeline took most of my time.

4. Data augmentation and dropout

These are well-known processes that are very useful for small datasets to reduce overfitting. As my training was very stable, I used a lot of these. I used horizontal flips, rotations, scalings, shearings, translations, gaussian blurs, color jittering and random erasing [2]. For dropout [3], I usually used high values between 0.5 and 0.7.

5. Feature visualisation

I used PCA and t-SNE to visualise the features (second-to-last layer activations) of my model to better understand what was going on.

6. Ensemble and test-time augmentation

I combined a ResNeXt101, a DenseNet121 and a ResNet151 in my final model, and used a soft voting strategy (each output was applied a softmax before summing). I also tried without softmax, and I also tried concatenating the second-to-last layer activations and feeding into a densely connected layer. I also tried generating votes using test-time augmentation, but my model did not fit on my GPU any more so I could not use that before the deadline.

Conclusion

My best model was: ResNeXt101, 0.6 dropout, SGD(lr=0.01,momentum=0.5), 230 epochs. I was disappointed that the more complex methods underperformed, but that may be because I could not thoroughly test them due to a lack of time. I believe that with good hyperparameters, simple transferred models can reach 0.8 test accuracy, and that ensemble methods might reach 0.9. I would have also liked to try training more my models on some classes, since my model's accuracy was very uneven among classes (0.9 at best vs 0.2 at worst).

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 1

[2] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation.," 1

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," 1