Writing Scientific Papers and Software

Cheng Soon Ong
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—A critical part of scientific discovery is the communication of research findings to peers or the general public. Mastery of the process of scientific communication improves the visibility and impact of research. While this guide is a necessary tool for learning how to write in a manner suitable for publication at a scientific venue, it is by no means sufficient, on its own, to make its reader an accomplished writer. This guide should be a starting point for further development of writing skills.**

I. Introduction

The aim of writing a paper is to infect the mind of your reader with the brilliance of your idea [1]. The hope is that after reading your paper, the audience will be convinced to try out your idea. In other words, it is the medium to transport the idea from your head to your reader's head. In the following section, we show a common structure of scientific papers and briefly outline some tips for writing good papers in Section **??**.

At that point, it is important that the reader is able to reproduce your work [2], [3], [4]. This is why it is also important that if the work has a computational component, the software associated with producing the results are also made available in a useful form. Several guidelines for making your user's experience with your software as painless as possible is given in Section **??**.

This brief guide is by no means sufficient, on its own, to make its reader an accomplished writer. The reader is urged to use the references to further improve his or her writing skills.

II. Models and Methods

Describe your idea and how it was implemented to solve the problem. Survey the related work, giving credit where credit is due.

III. Results

Show evidence to support your claims made in the introduction.

IV. Discussion

Discuss the strengths and weaknesses of your approach, based on the results. Point out the implications of your novel idea on the application concerned.

V. Summary

Summarize your contributions in light of the new results.

VI. Obtaining Heat-Maps

In this section we are going to describe the first component of our framework which is a method to transform a given input image into (how we call it) a *Road Heat Map*. More precisely suppose an image I is given of dimensions $n \times n$, then the output of our method is an array $x \in [0,1]^{n \times n}$ with the intent that $x(i,j)$ corresponds to the probability that the pixel (i,j) is part of a road. Clearly this is the essential part of the task and the accuracy here crucially affects the final accuracy of our method, no matter how we proceed after that.

A. Basic CNN

The provided sample code *tf_aerial_images.py* of a Convolutional Neural Network for solving this task serves as good starting point for our final approach. Let us describe it briefly here. To compute a prediction for a given input picture of dimension $n \times n$, the picture is first divided into $\frac{n}{16} \times \frac{n}{16}$ chunks of size 16×16 each. Subsequently, every such chunk C is being input into a neural network $\mathcal{N}$, which outputs a number $p(C) \in [0,1]$ which is then used as the road heat map value for the corresponding 16×16 chunk. What is left to explain is how is $\mathcal{N}$ constructed and how is it trained. The network $\mathcal{N}$ has the following layers: (*Convolutional, RELU, Max Pooling*) repeated twice in this order. To calculate losses, the cross entropy function is used. The network is trained using training pairs (x,y) (input and output) of the form: x is a 16×16 chunk, part of a training picture and $y \in \{0,1\}$ is a suitably rounded mean of the corresponding ground truth chunk.

To deal with the network $\mathcal{N}$ efficiently, the *TensorFlow* library is used. In particular, crucially, this provides us with a ready-to-apply optimization primitive *MomentumOptimizer*. We do not go into details on the choice of parameters on which the method is invoked.

B. Big CNN

Our way of obtaining Heat-Maps builds upon the approach described above, but includes several new insights and modifications which we are going to describe below and briefly discuss how do they affect the results and accuracy.

The first modification we employ concerns the general philosophy how the heat map is obtained. Note that in the basic approach above, the heat map is "discrete" in the sense that it is constant on 16×16 chunks. Anotr disadvantage is that the result for a single chunk depends *only* on the

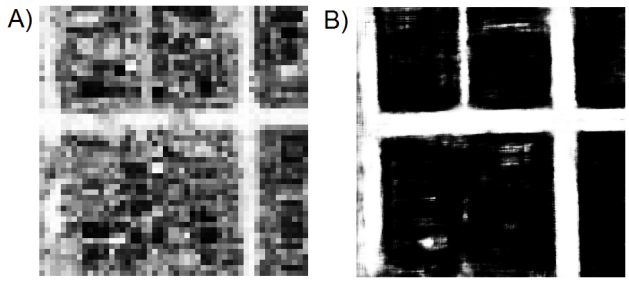

Figure 1. The difference in quality between heat maps computed using the Basic Neural Network A) and the Big Neural Network B).

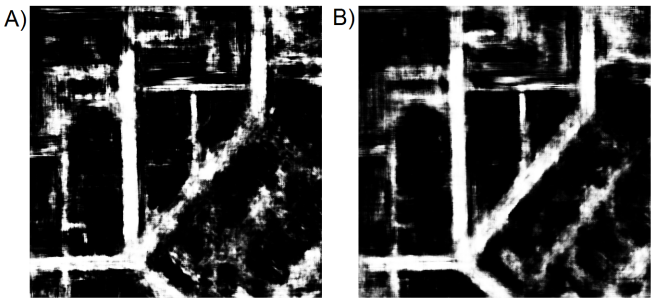

Figure 2. The results of predictions when trained on just the training data A) and after adding rotated images B).

pixels within it, not at all on the neighboring ones, whereas in reality such a prediction cannot be performed locally.

To fix these issues we propose and implement a different method for computing the Heat Map. For every pixel (i, j) in the image we take its square neighborhood of size 48×48 and feed it into a new neural network $\mathcal{N}'$ to get a single value $p' \in [0, 1]$ which determines the value of the corresponding entry (i, j) in the heat map. By this design we get a much "smoother" heat map, which also does not suffer so much from the "locality" issue, as the value of a pixel is determined by looking at a quite large neighborhood of it. To see a sample on how much improvement does it yield, refer to Figure 1. Note that such a change in the model introduces some issues with efficiency. Indeed the network $\mathcal{N}'$ is significantly larger than $\mathcal{N}$ and also the number of predictions which needs to be performed increases roughly $16^2 = 256$ times, when compared to the previous model. Further, also training such a network becomes computationally quite demanding. We obtain reasonable running times by taking advantage of a workstation with 20 cores and using the parallelization power *TensorFlow* provides. In particular, for the latter, we coded the predictions to run in batches of 32, which allows them to be executed in parallel at multiple cores. It turns out that 32 is essentially the optimal number here, since increasing it to 64 requires enormous amounts of memory and reducing it to 16 makes the process a little slower.

C. Diagonal Roads

One of the issues which appears after training our modelon the training data only is that the roads, which are diagonal in the testing data are hardly recognized. This might be before the typical road in the training data is either vertical or horizontal, which is then "learned" by the network as correct. For an example see Figure 2. To get around this problem we add several random rotations of the training images to the training data. This has the desired effect of making the "confidence" of diagonal and non-diagonal predictions equal. Indeed without rotated pictures in the training data, the vertical and horizontal roads were typically detected with a (too) large confidence while the diagonal ones were either very blurry or nonexistent. After this manipulation there is symmetry among different directions of roads in terms of intensity in the heat map.

D. Overfitting

To control overfitting we were performing validation over 30% of the training data. Indeed we encountered significant overfitting when using different parameters for the model of our Neural Network. When using chunks of size 64 (instead of 48 as we use in the final version) the difference between the average prediction scores for instances used for training and for instances outside of the training set reached roughly 0.1. In the current version it is no more than 0.02.

The weak form of validation we use is mainly because of efficiency reasons. While for the Basic Neural Network we were able to run full $K-$fold cross validation for $K = 4$, it is no more feasible for the Big Network (because training is very time consuming). Since the results we were obtaining were much better for the Big Network, we decided to stick to it and relax the validation method to a weaker one.

VII. The Post-Processing Phase

ACKNOWLEDGEMENTS

REFERENCES

[1] S. P. Jones, "How to write a great research paper," 2008, microsoft Research Cambridge.

[2] M. Schwab, M. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science and Engg.*, vol. 2, no. 6, pp. 61–67, 2000.

[3] J. B. Buckheit and D. L. Donoho, "Wavelab and reproducible research," Stanford University, Tech. Rep., 2009.

[4] R. Gentleman, "Reproducible research: A bioinformatics case study," *Statistical Applications in Genetics and Molecular Biology*, vol. 4, no. 1, 2005. [Online]. Available: http://www.bepress.com/sagmb/vol4/iss1/art2