How to Put Algorithms into Neural Networks

Anton Osokin[1] [0000−0002−8807−5132]

National Research University Higher School of Economics, Russia
aosokin@hse.ru

1 Introduction

Recently, neural networks have achieved remarkable success in many fields [4]. Many practical systems for fundamental tasks are built with neural networks. For example, in computer vision, it is image classification,, object detection and image segmentation; in natural language processing, it is language modeling and automatic translation; in audio processing, both speech recognition and synthesis. Many approaches have become an industrial standard, and companies around the world are building products based on this technology.

Successful algorithms for various tasks are very different from each other and required years of research to arrive at the current level of performance. Constructing a good algorithm for a new task is often a non-trivial challenge. It also turns out that networks can not just learn from data without exploiting some domain knowledge. This knowledge is usually encoded at least in the architecture itself. For example, convolutional neural networks [2] exploit intuition that translation of the object does not change the object itself, i.e., a cat does not stop being a cat if moved left.

At the same time, in many domains we already have powerful algorithms that do a decent job. It is a very natural idea to exploit those to construct better networks. We can look at this from two sides. From one side, this means constructing new layers or blocks of layers for networks. From another side, this means making trainable algorithms. In any case, the attempt is to take best of both worlds. This direction has been around since 90s [1,3,5], but for long time was not getting significant attention (together with neural networks).

In this talk, we will review three ways to combine algorithms and networks (see Fig. 1):

1. structured pooling: an algorithm is used to select active features (similarly to max pooling);
2. unrolling iterations into layers: an algorithm simply becomes a part of the network;
3. analytical derivative w.r.t. the algorithm input, i.e., building a layer with a special backward operator.

To illustrate all the approaches, we will use a running example of a simplified task of handwriting recognition: recognize a word given a sequence of images where each image shows exactly one letter.

Acknowledgments

Supported by RSF project 19-71-30020.

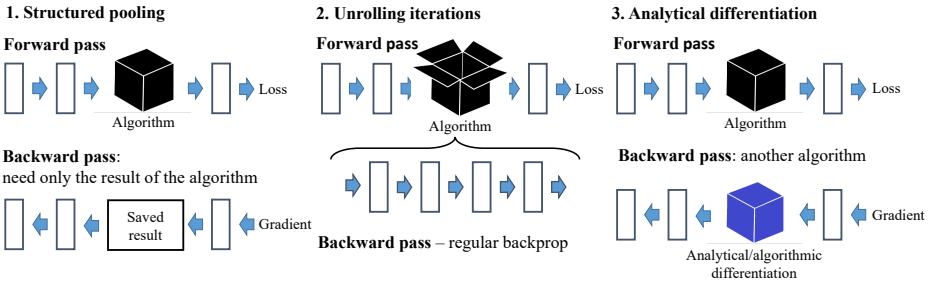

Fig. 1. Thee approaches to combine an algorithm and a neural network.

References

1. Bottou, L., Le Cun, Y., Bengio, and Y.: Global training of document processing systems using graph transformer networks. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (1997).
2. Goodfellow, I., Bengio, Y., and Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
3. Le Cun, Y., Bottou, L., Bengio, Y., and Haffner, P.: Gradient based learning applied to document recognition. Proceedings of IEEE **86** (11), 2278–2324 (1998).
4. LeCun, Y., Bengio, Y., and Hinton, G.: Deep learning. Nature **521** (7553), 436–444 (2015).
5. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F.: A tutorial on energy-based learning. Predicting structured data **1** (0) (2006).