NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

Assignment 2

TDT4173: Machine Learning and Case-Based Reasoning

Submitted by
Vittorio Triassi

September 2019

1 Essay

Authors
He, K., Zhang, X., Ren, S., and Sun, J. (2016)

Deep Residual Learning for Image Recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770-778.

1.1 Introduction

In the following paper it has been shown that when dealing with deep neural networks, a few side effects might occur. The reason why we usually stack several layers is to increase the non-linearity in order to better recognize the desired classes. Unfortunately, it has been pointed out that deeper neural networks can easily run into the *degradation* problem. In fact, if on one side we might think that deep networks always perform better than shallow ones, on the other side we have noticed that these can also have higher training errors. At this point, we could put this on vanishing gradients or overfitting but it has been proven that degradation is not related to these problems because it usually occurs when the model starts converging, and the deeper it goes, the sooner the accuracy gets saturated. This paper aims at proposing a new model that allows deep networks to prevent degradation without burdening on the number of parameters and on the complexity of the architecture.

1.2 Proposed Method

The model introduced by the authors is called *deep residual learning* which is a different approach we can use when dealing with deep convolutional neural networks. As it has previously stated, degradation occurs when we stack many layers in our network. The basic idea in neural networks is that they have to learn a mapping, and the deeper we go, the harder such mapping can be to learn. To ease this process

we can recast our architecture in such a way that we start from a (smaller) network we already know to behave well, and stack other layers by adding identity functions. What these identity functions do is to skip every time a number of layers. They let us to go from one layer to another without altering the results. Recall that an identity function returns the same value that was used as its argument [9]. By doing so, we do not need to learn new parameters and learning the identity functions costs zero. The authors think that it is easier to optimize something called *residual mapping* rather than an unreferenced mapping. To prove that, they start comparing two plain networks. The first one with 20 layers and the second one with 56 layers. It turns out that the second network performs worse than the first one, even if it is supposed to be the other way around considering its complexity. Then, they focus their attention on how to learn the residual mappings. If $\mathcal{H}(x)$ is the true mapping we would like to learn, we define another mapping $\mathcal{F}(x)$ and learn it instead of $\mathcal{H}(x)$. The new mapping is defined as $\mathcal{F}(x) := \mathcal{H}(x) - x$ so that the original function becomes $\mathcal{F}(x) + x$. It turns out that if the identity was optimal, it would be easier to set the weights as 0 and if the optimal mapping is closer to the identity, it is easier to find small fluctuations. To better understand the whole process, an illustration of the residual block is provided in Figure 1.1.

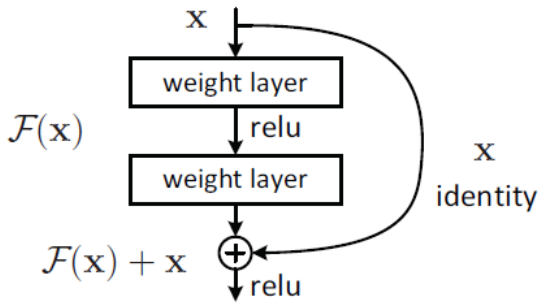

Figure 1.1: Residual learning: a building block [3]

The question that may arise now is:"why does residual learning work?". We can think of the problem as follows. Let us start considering a deep network with two extra layers and let us use the ReLU as activation function. Let $a^{[l+2]}$ be the activation for the last layer. If we connect (add) the input ($a^{[l]}$) to the output of the last layer, then $a^{[l+2]}$ would be defined as:

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

Expanding the activation, we should compute:

$$a^{[l+2]} = g(w^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

If we apply weight decay, then the value of $w^{[l+2]}$ and the bias $b^{[l+2]}$ can be considered equal to zero and we end up with:

$$a^{[l+2]} = g(a^{[l]}) = a^{[l]}$$

That is because using the ReLU, all the activations can not be negative. So, $g(a^{[l]})$ would be the result of the ReLU applied to a non negative quantity. This proves how

easy is for a residual block to learn the identity function. Moreover, adding the two extra layers did not hurt the performance of the network. Of course in our case, we would like to improve the performance and not just getting the same. If the hidden neurons of the new layers could learn something useful, it would be even better than simply learning the identity function. There is another aspect that is important to mention. We assumed that $z^{[l+2]}$ and $a^{[l]}$ have the same dimension. If the input and the output have different dimension, what we do is add an extra matrix called W_s such that:

$$a^{[l+2]} = g(w^{[l+2]}a^{[l+1]} + b^{[l+2]} + W_s a^{[l]})$$

This allows us to perform the computation without any error on the dimension of the quantities. The final intuition we might have after this demonstration is that the network chooses to change things when changing is the best thing to do, otherwise everything is the same.

1.3 Experimental Results

The authors evaluate their method on two datasets: ImageNet 2012 [6] and CIFAR-10 [4]. ImageNet is a dataset used for classification tasks and it consists of 1000 classes. The dataset has been split as follows: almost 1.3 million training examples for the training set, 50k for the validation set and 100k for the test set. First, two plain networks have been evaluated, respectively made of 18 and 34 layers. It turns out that the 34-layer network has higher error than the 18-layer one. Then, they evaluate 18-layer and 34-layer residual networks. In this case, the 34-layer ResNet performs better than the shallower one. What is interesting is that the deeper network shows lower training error and generalizes better. As far as the 18-layer ResNet, it converges faster than its plain version. After that, the authors decide to go even deeper. That is why they propose two different types of residual building blocks (Figure 1.2).

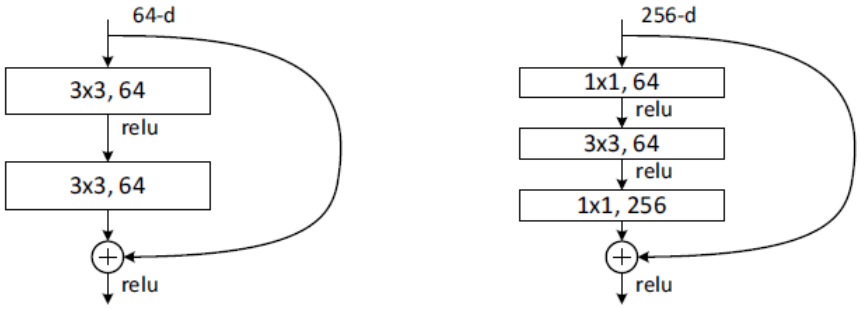

Figure 1.2: Left: a building block for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152. [3]

In the 34-layer ResNet, we skip 2 layers every time (thanks to the short cut connections). In deeper architectures, 3 layers are skipped instead. They call the following block, a "bottleneck" building block. In fact, the three layers make it so the dimensions are reduced and restored through the different sizes of the convolutions:

respectively 1×1, 3×3 and 1×1. The bottleneck block is used in three deeper architectures that are respectively made of 50, 101 and 152 layers. The 152-layer ResNet performs better than VGG-16/19 [7] (that has more parameters) and allowed the authors to win the 1st place in ILSVRC 2015 [6].

The second dataset that has been used is CIFAR-10 [4]. The following dataset contains 60k images that are split in 50k training examples, 10k test examples and it aims at classifying 10 classes. In this case, the main goal was to carry out experiments on very deep networks. They try two architectures: 110-layer ResNet and 1202-layer ResNet. The 110-layer ResNet performs quite well, converging faster than plain networks. The second approach, which involves 1202 layers, shows how overfitting impacts on the performance of the network. In fact, even if the training errors of both networks are comparable, the 1202-layer ResNet does not seem to be able to correctly generalize. Probably this depends on the size of the dataset that does not have enough examples for such a deep architecture.

In addition to that, the authors address other two tasks: *object detection* and *object localization* using PASCAL VOC 2007 [1], PASCAL VOC 2012 [2] and COCO [5] datasets. In this case, they use a detection method called Faster R-CNN in which, rather than selecting a huge number of regions as it happens in CNNs, just 2000 of them are extracted and used to perform the classification task. With the following method they aim at improving performance by using ResNet-101 rather than VGG-16. It turns out that with ResNet, they get better results than Faster R-CNN + VGG-16 ending up winning the 1st place for (1) ImageNet detection and localization and (2) COCO detection and segmentation.

1.4 Conclusion

The presented paper has been a breakthrough for Deep Neural Networks and more specifically for Convolutional Neural Networks. In fact, it has been used as a starting point for several other papers. The idea of short cut connections is quite simple but at the same time very effective because it allows the network to learn residual functions without requiring additional parameters. In the paper, only skips of two and three layers are shown. It would be interesting to see how the network performs with bigger skips. The authors have mainly pointed out the strengths of their method without caring too much about potential weaknesses. Since their goal was to mitigate the degradation problem, we can actually say that they did quite well. Moreover, we noticed that residual learning resembles in some ways an older approach called Highway networks [8], in which gated short cut connections were introduced. These gates could decide how much information could actually flow. Experiments show that ResNets perform better than Highway networks, that add more parameters instead. Reading other papers, I have personally noticed that very often the proposed works make use of a lot of GPUs, ending up with very complex and deep architectures without really introducing anything conceptually new. Something I personally appreciated for the following work has been that the authors have focused their attention more on the

single building block, in this case the *residual block*, rather than just unnecessarily complicating the structure of the whole network. In fact, when they only increased the number of the layers (1202-layer ResNet), they ran into overfitting. That suggested that the model was too complex for such a small dataset and other optimizations were required but that was not the goal of this paper.

References

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[9] Wikipedia. *Identity function*. https://en.wikipedia.org/wiki/Identity_function.