UNDERSTANDING VERY DEEP NETWORKS VIA VOLUME CONSERVATION

Thomas Unterthiner & Sepp Hochreiter
Institute of Bioinformatics
Johannes Kepler University, Linz, Austria
{unterthiner,hochreit}@bioinf.jku.at

ABSTRACT

Recently, very deep neural networks set new records across many application domains, like Residual Networks at the ImageNet challenge and Highway Networks at language processing tasks. We expect further excellent performance improvements in different fields from these very deep networks. However these networks are still poorly understood, especially since they rely on non-standard architectures.

In this contribution we analyze the learning dynamics which are required for successfully training very deep neural networks. For the analysis we use a symplectic network architecture which inherently conserves volume when mapping a representation from one to the next layer. Therefore it avoids the vanishing gradient problem, which in turn allows to effectively train thousands of layers. We consider highway and residual networks as well as the LSTM model, all of which have approximately volume conserving mappings.

We identified two important factors for making deep architectures working: (1) (near) volume conserving mappings through $x = x + f(x)$ or similar (cf. avoiding the vanishing gradient); (2) Controlling the drift effect, which increases/decreases x during propagation toward the output (cf. avoiding bias shifts);

1 INTRODUCTION

Since its inception, the field of deep learning has sought out architectures that can effectively learn many layers of hidden representation. In doing so, it has pushed the state of the art in many application domains. It seems like an increase in performance often goes hand in hand with an increase in the amount of layers that networks have. Within the field of computer vision for example, the prestigious ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) has been won by deep convolutional networks every year since 2012. While the AlexNet architecture of 2012 featured 8 hidden layers (Krizhevsky et al., 2012), the winning network of 2014 had 22 layers (Simonyan & Zisserman, 2015) and the ILSVRC 2015 was eventually won by a 152 layer network (He et al., 2015). In a similar vain, the recently introduced Highway Networks (Srivastava et al., 2015) show a lot of promise, not only in vision but also in language tasks (Jozefowicz et al., 2016).

2 COMMON FEATURES OF VERY DEEP NEURAL NETWORKS

Deeper networks can in principle be constructed without changing the network architecture at all, by using very careful initialization schemes (Saxe et al., 2014) or by learning to mimic a shallower teacher network (Romero et al., 2015). However the most successful very deep networks all used specialized architectures to obtain state of the art results, with Highway and Residual Networks being two of the most prominent examples of this.

Highway networks (Srivastava et al., 2015) employ a gating mechanism and apply the following transformation in each layer

$$\begin{aligned} y &= (1 - g(x)) \cdot x + g(x) \cdot f(x) \\ &\approx x + f(x) \end{aligned}$$

where g is a gating function that decides whether the network transforms the inputs x via a non-linearity f or just copies it to produce the output y. The second line is an approximation that holds true at the beginning of learning if the recommended initialization procedure is followed.

ResNet are based on the idea that layers can be grouped into blocks which should learn a modification that is added to the block's input, instead of fully replacing that input. Thus, the transformation performed by such a block is

$$y = x + f(x)$$

The f used in a ResNet is typically a two- or more hidden layer neural network, typically employed in a convolutional setting.

Additionally, recurrent networks such as the Long Short-Term Memory network (LSTM) Hochreiter & Schmidhuber (1997) result in very deep neural networks when unrolled in time. LSTMs are networks where each neuron can store information. Read, write and delete access to this information is controlled via gates. The information at time-step t is calculated as

$$x_t = x_{t-1} + f(z_t)$$

where the subscript denotes time-steps and z is the input to the neuron after it has passed the input gate. This information could later potentially be modified via a delete gate.

What all of these architectures have in common is that the function to transform the layer input x to an output y is a variant of the equation

$$y = x + f(x) \tag{1}$$

In the following we will see how, with an appropriate choice of f, this yields a volume conserving transformation in subsection 2.1 which avoids the vanishing gradient (Hochreiter, 1991; Hochreiter et al., 2001). To analyze the behavior of these networks, we use a network architecture first introduced by Deco & Brauer (1995) which guarantees volume conservation (introduced in the same subsection). However, Equation (1) can also lead to unwanted drift effects as described in subsection 2.2.

2.1 Volume Conservation

Training very deep networks is not a trivial task. It has been known for a long time that vanishing gradients makes training these networks difficult. It can be shown that the gradient flow through a neural network diminishes as the number of layers increases. One interesting way of looking at the vanishing gradient problem is from an information-theoretic point of view (Hochreiter, 1998): let h be a (bijective) mapping that transforms a layer's input x into an output y. If we consider Y and X as random variables that are connected via

$$Y = h(X)$$

Then we know from basic information theory that the entropy $\mathcal{H}$ between these variables behaves as

$$\mathcal{H}(Y) \leq \mathcal{H}(X) + \mathbb{E}\left(\log\left|\frac{\partial Y}{\partial X}\right|\right)$$

Where $\mathbb{E}$ denotes the expectation of a random variable and $\left|\frac{\partial Y}{\partial X}\right|$ is the determinant of the Jacobian of the mapping h. Whenever this determinant is smaller than 1, information will be lost in the transformation, and we will observe vanishing gradients. Thus, the key to make learning feasible over many layers is to ensure that the mapping is volume conserving/symplectic, i.e., that

$$\left|\frac{\partial Y}{\partial X}\right| = 1$$

This insight was exploited by Deco & Brauer (1995) to build a network for Independent Component Analysis that is volume conserving by design. By constructing a Jacobian $\frac{\partial Y}{\partial X}$ that has upper (or lower) triagonal form with ones on the diagonal. Concretely, the ith node of a layer in the Deco & Brauer architecture calculates

$$y_i = x_i + f(x_i) = x_i + f\left(\sum_{j \in \mathcal{I}(i)} w_{ij} x_j + b_i\right)$$

Where w_{ij} is the connection strength from input x_j to output y_i and b_i is a bias unit.

The crucial trick in this design is that the summation index j only goes over all indices that are larger (smaller) than i, depending on whether the Jacobian should be upper or lower diagonal. That is to say

$$\mathcal{I}(i) = \begin{cases} \{j | j < i\} & \text{for layers with upper triangular Jacobians} \\ \{j | j > i\} & \text{for layers with lower triangular Jacobians} \end{cases}$$

Note that this architecture implicitly assumes that each layer has the same number of hidden units.

If we stack multiple of these layers and connect them with a softmax, sigmoid or linear output unit on top, we obtain what we will refer to as a *Deep Volume Conserving Neural Network* (DVCNN). An example of such an architecture can be seen in Figure 4. These networks can be trained to perform classification or regression using conventional neural network optimization techniques such as stochastic gradient descent.

Srivastava et al. (2015) suggests to initialize Highway Networks in a way that all gates start out in a closed state, setting the layers to copy their input. In that case, the mapping function becomes $y = x$, and thus $|\frac{\partial f}{\partial x}| = 1$ trivially holds. This may later change during training, but it ensures that gradients can flow at the beginning of training, where avoiding vanishing gradients is most crucial.

Just like DVCNNs, ResNets mapping has the form $y = x + f(x)$. While it cannot be guaranteed that a ResNet is fully volume conserving, it is very close to that. This is because the convolutional structure of f implies that its Jacobian only has very few non-diagonal entries (i.e., it is sparse), even though not completely triagonal. This means that

$$\frac{\partial y}{\partial x} = I + \frac{\partial f}{\partial x}$$

Assuming the weights are initialized sensibly, we can assume that $\frac{\partial f}{\partial x}$ is sparse, its eigenvalues are relatively small as well, since a sparse matrix has a smaller frobenius norm than a dense matrix[1]. As the Frobenius norm is an upper bound to the spectral norm, this guarantees that the largest eigenvalue is relatively small, and thus the determinant of the Jacobian is close to 1.

2.2 Drift Effects

The non-linearities employed in DVCNNs are $h(x) = x + f(x)$. If f is a non-negative function, as is the case with sigmoids and ReLUs, the activations will increase throughout the layers, leaving the upper layers at very high activations. While this is not a big problem when there are only a small number of layers, this severely affects deeper networks. Even with a small number of layers this can slow down learning due to a bias shift (Clevert et al., 2015). When using an unbounded function like the ReLU, drift effects can become very severe and make learning highly unstable. With bounded function such as a sigmoid, it will result in activations that are saturated in the higher layers, which makes learning in those layers more difficult.

This problem was first described as "internal state drift" in the original LSTM publication (Hochreiter & Schmidhuber, 1997): each time-step the state inside each LSTM cell is updated by an equation of the form $x_t = x_{t-1} + f(z_t)$, which can lead to cells that drift away over time. This effect was ameliorated with the introduction of forget gates, but is still observable under certain conditions. For highway networks, this is not an issue, since the output of a layer is always a convex combination of the input and the transformation, making it unlikely that the end result is bigger than

[1] assuming entries of similar size

the input. ResNets make use of Batch-Normalization (Ioffe & Szegedy, 2015) to make sure that the state does not drift over time. For DVCNNs, the drift effect leads to interesting dynamics, which are explored more deeply in section 3.

3 Experimental Results

The DVCNN is a simple model that can easily encode hundreds of layers. This makes it a suitable model to study the effects of very deep networks. We trained the DVCNN on the *MNIST-bg-img* variation of the MNIST digit recognition data-set from Larochelle et al. (2007), where the black background was exchanged with image patches. This data-set consists of 10k training set images, 2k validation set images and 50k test set images, spread across 10 digit classes. Each image has a resolution of 28x28 pixels. As the architecture requires that each hidden layer has as many units as there are input features, each hidden layer had 784 hidden units. Training was done using stochastic gradient descent on a GPU.

3.1 Vanishing Gradients

As expected, a DVCNN can easily be trained even with hundreds or thousands of layers[2], showing no signs of vanishing gradients at all, as can be seen in Figure 1. In fact, deeper networks seemed to reach a lower training error, which is likely due to their increased capacity.

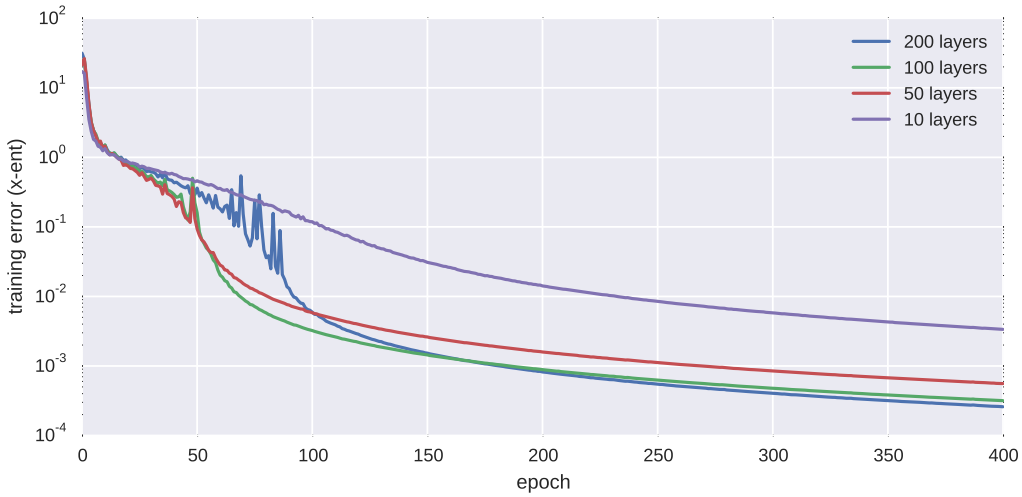

Figure 1: Training error of DVCNNs of different depth on MNIST bg-img. Even the very deep networks are able to effectively minimize the training objective without running into vanishing gradient issues. All nets were trained using the same hyperparameters.

3.2 Drift Effects

Using DVCNNs with ReLU activation functions immediately leads to numerical problems, and thus could not be explored further. However, both sigmoid and $\tanh$ activations worked nicely. One interesting observation is that a DVCNN would yield the best generalization performance when these non-linearities were initialized in their saturating regime. This was done by initializing the bias units to values such as ± 2.

Figure 2 shows the average activation value across all units of a DVCNN with 100 layers that uses a sigmoid activation function, once for an untrained network, and once after the network has converged. One can immediately see that the untrained network suffers heavily from the drift effect,

[2]Our implementation ran out of GPU memory with more than 2 000 layers.

with the average value increasing from 0 in the initial layer up to over 70 at the highest layer. After the network has trained, the drift effect is still noticeable, but much less pronounced. The reason for this is that the network has learned to push most non-linearities to zero, only activating certain non-linearities when a given signal is presented, as shown in Figure 3.

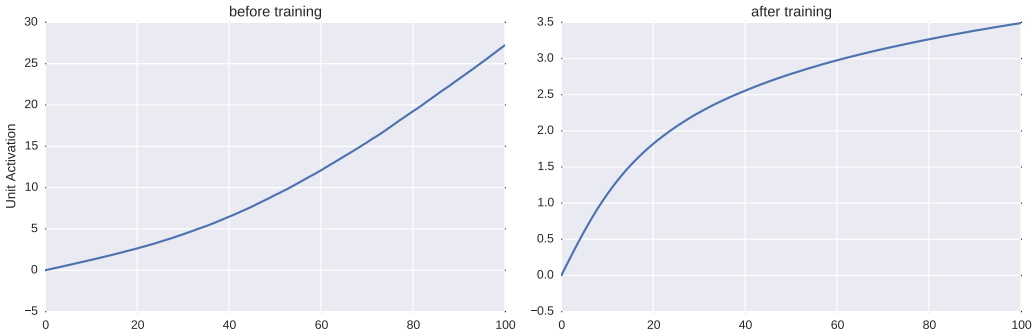

Figure 2: Average value of the non-linearity f across all units of each layers of a DVCNN using a sigmoid function with the bias set to -2. **Left:** before learning has begun, **Right:** after the network has converged.

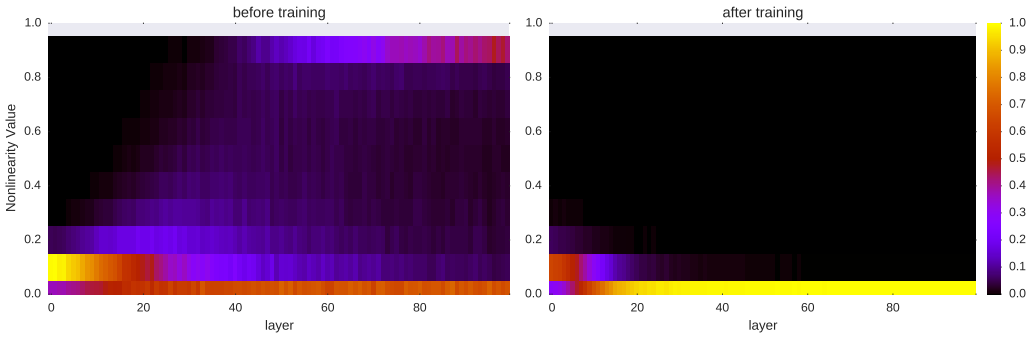

Figure 3: Distribution of the value of the non-linearity f across all units of each layers of a DVCNN using a sigmoid function with the bias set to -2. **Left:** before learning has begun, **Right:** after the network has converged.

4 Discussion and Conclusion

This work analyzed the properties of very deep networks, using a deep volume conserving neural network model as a basis for our analysis. Thanks to their perfect volume conservation property, this model allows us to analyze the behavior of very deep networks in the absence of vanishing gradients. For example, it also allows us to very clearly observe the drift effect which is otherwise hard to uncover in models like the LSTM or Residual Networks.

In the future, we intend to confirm our results using more advanced models like Highway or Residual Networks. We would also like to refine the DVCNN model to make its performance more competitive with regards to generalization performance. Currently, due to their superior learning behavior, DVCNNs tend to overfit on the training data very quickly. One easy fix would be to restrict the connectivity of the network: instead of connecting a unit to all inputs on its left/right, we connect it only to the n nearest units on the left/right. This effectively reduces the complexity of the network to fight overfitting. We also experimented with other forms of regularization such as dropout, but first experiments were not as encouraging. To fight the effect of the drift effect, Batch Normalization could be employed.

ACKNOWLEDGMENTS

The authors would like to thank Rupesh Kumar Srivastava and Klaus Greff for helpful discussions about highway networks, and gratefully acknowledge the NVIDIA Corporation for supporting this research through a hardware donation.

REFERENCES

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *CoRR*, abs/1511.07289, 2015.

G. Deco and W. Brauer. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525 – 535, 1995. ISSN 0893-6080.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *arXiv prepring arXiv:1506.01497*, 2015.

S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Master's thesis, Technische Universität München, Institut für Informatik, 1991.

S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, April 1998. ISSN 0218-4885.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer and Kolen (eds.), *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Journal of Machine Learning Research*, 37:448–456, 2015. Proceedings of the 32nd International Conference on Machine Learning (ICML15).

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the Limits of Language Modeling. *ArXiv e-prints*, February 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 473–480, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.

A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *International Conference of Learning Representations (ICLR 2015)*, 2015.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

A.M. Saxe, J.L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference of Learning Representations (ICLR 2014)*, 2014.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference of Learning Representations (ICLR 2015)*, 2015.

R. K Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2368–2376. Curran Associates, Inc., 2015.

A NETWORK STRUCTURE

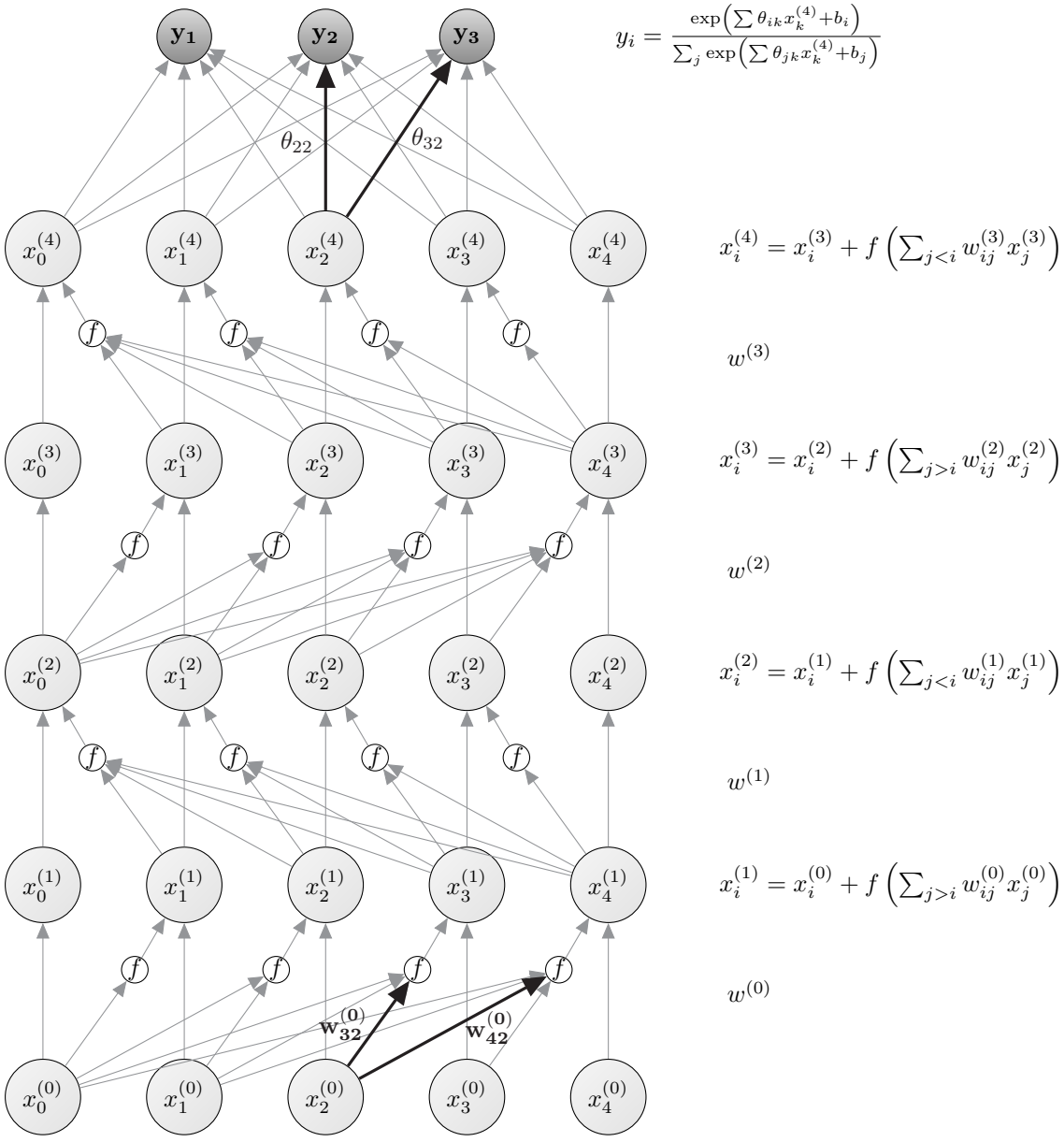

Figure 4: An example of a Deep Volume Conserving Network with four hidden layers and a softmax output layer.