3 - ResNets

Scribed by Manikanta Srikar Yellapragada, Luca Venturi

1 ResNets

In standard (feedforward/convolutional) neural networks, each layer process the input $\mathbf{x}_k$ (which is the output of the k-th layer), by applying an affine transformation defined by a weight-bias couple $(\mathbf{W}_k, \mathbf{b}_k)$ and a non-linear function ρ (tipically applied component-wise), as follows:

$$\begin{aligned}\mathbf{x}_{k+1} &= \rho(\mathbf{W}_k \mathbf{x}_k + \mathbf{b}_k) \\ \mathbf{x}_{k+2} &= \rho(\mathbf{W}_{k+1} \mathbf{x}_{k+1} + \mathbf{b}_{k+1}) \\ &\dots\end{aligned}$$

In a Residual Network (ResNet), skip-connections are introduced (cfr. (He et al., 2016a)):

$$\mathbf{x}_{k+2} = \mathbf{x}_k + \rho(\mathbf{W}_{k+1} \mathbf{x}_{k+1} + \mathbf{b}_{k+1})$$

This consists in nothing but adding the (unprocessed) output from a previous layer to the output of the current one. Of course, one can think at many variants of such architectures and many have been proposed. In general we could think to ResNets as networks of the form

$$\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \mathbf{f}(\mathbf{x}_k) + \mathcal{F}(\mathbf{W}_k, \mathbf{x}_k) \\ \mathbf{x}_{k+1} &= \mathbf{h}(\hat{\mathbf{x}}_{k+1})\end{aligned}$$

In a simple ResNet $\mathbf{f}$ is taken to be the identity; in standard neural networks, $\mathbf{f} \equiv \mathbf{0}$. ResNet aroused from the difficulty of training very deep networks. Indeed, while from an approximation point of view deeper networks perform not-worse than their shallower counterpart, in practice, it has been found difficult to train very deep networks (without residual connections) and get better performances.

Question: Show that a k-layer (feedforward) network can be written as $(k+1)$-layer network, if the activation function is non-negative.

Notice that this is very easy to show for a ResNet: indeed, by taking $\mathbf{W}_k \equiv \mathbf{0}$ (and $\mathbf{f} = \mathbf{h} = \mathrm{id}$), one gets $\mathbf{x}_{k+1} = \mathbf{x}_k$. Therefore, in some sense, it is easier to approximate the identity transformation with a ResNet w.r.t. a standard neural net. While it is unlikely that identity mappings are going to be optimal, the authors who first introduced residual architectures suggested that this could be a reason why such reformulation may help to

precondition the training problem. The authors of (He et al., 2016b) also noticed that, by taking a neural net of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathcal{F}(\mathbf{W}_k, \mathbf{x}_k) = \cdots = \mathbf{x}_{k-l} + \sum_{i=k-l}^{k} \mathcal{F}(\mathbf{W}_i, \mathbf{x}_i)$$

and with L layers, the derivative of a loss function $\mathcal{E} = \mathcal{E}(\mathbf{x}_N)$ w.r.t. the output of the k-th layer is given by

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \mathbf{x}_k} &= \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_k} \\ &= \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(1 + \frac{\partial}{\partial \mathbf{x}_k}\sum_{i=k}^{L} \mathcal{F}(\mathbf{W}_i, \mathbf{x}_i)\right)\end{aligned}$$

Therefore, as noted in (He et al., 2016b), *the gradient $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_k}$ can be decomposed into two additive terms: a term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}$ that propagates information directly without concerning any weight layers and another of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial}{\partial \mathbf{x}_k}\sum_{i=k}^{L} \mathcal{F}(\mathbf{W}_i, \mathbf{x}_i)$ that propagates through the weight layers. The additive term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}$ ensures that information directly propagated back to any shallower unit k. It also suggests that it is unlikely for the gradient $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_k}$ to be canceled out for a mini-batch* and therefore that the ResNets formulation alleviates the problem of vanishing gradients.

Of course, these are some very heuristic explanations, and while many other justifications have been proposed, we are still far from a full understanding of why this should help.

1.1 Lesion study and ResNets as an ensemble

The authors of the paper (Veit et al., 2016) proposed an unravelled view of residual networks. The input data can be seen as flowing along many paths from input to output, where each path is a unique configuration of which residual module to enter and which to skip. Consider a graph on $G = (V, E)$, where the vertices $V = [\![0, L]\!]$ correspond to the layers (where L is the number of layers) and two edges e_1^i, e_2^i go from layer $i-1$ to layer i; edge e_1^i correspond to the input flowing through residual module $\mathcal{F}(\mathbf{W}_i, \cdot)$ and edge e_2^i correspond to skipping the residual module. The final output of the residual network can be therefore seen as a sum of an exponential (2^L) number of nested terms.

The authors of (Veit et al., 2016), inspired by this view of residual networks, conducted lesion experiments to understand whether such paths are strictly dependent or rather behave as an ensemble. After training the network, they looked at the increase in testing error when removing or reordering some layers during test time. Surprisingly the error shows very little increase when changing only a few connections; moreover, the error that correlates smoothly with the amount of corruption of the architecture. This could seen as a validation of the idea that ResNets behave more as ensemble, even when trained jointly. The number of paths, when removing one connection, decreases from 2^L to 2^{L-1}, leaving half of the paths valid, and could explain the small increase in testing error. Different lesion studies were also conducted in the papers (Chang et al., 2017).

2 ResNets as Explicit Euler steps and stability

Consider a simple residual net where the layer outputs are propagated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}(\mathbf{x}_k, \boldsymbol{\theta}_k)$$

If we multiply $\mathbf{f}$ by some (small) constant $h > 0$, we get

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\mathbf{f}(\mathbf{x}_k, \boldsymbol{\theta}_k)$$

which looks exactly as an Explicit Euler step for an ODE of the form

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t, \boldsymbol{\theta}_t) \tag{1}$$

for some map $\boldsymbol{\theta}_t$. This simple observation suggested new residual architectures motivated by properties of dynamical system. For example, a simple desirable property could be for the solutions of the ODE not to *explode* (i.e. diverge) as $t \to \infty$, but instead belonging to a bounded set. This is ensured by (cfr. (Haber and Ruthotto, 2017)) $\boldsymbol{\theta}_t$ changing sufficiently slow and by the Jacobian $\mathbf{J}_t \doteq \nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x}_t, \boldsymbol{\theta}_t)$ satisfying

$$\max_i \mathrm{Re}(\lambda_i(\mathbf{J}_t)) \leq 0$$

We saw in the second lecture how the corresponding stability property for the explicit Euler's method is ensured by

$$\max_i |1 + h\lambda_i(\mathbf{J}_k)| \leq 1$$

where $\mathbf{J}_k = \nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x}_k, \boldsymbol{\theta}_k)$. Equivalently, a network with $\mathrm{Re}(\lambda(\mathbf{J})) > 0$ corresponds to a network which amplifies a signal, that, for long time propagation, would diverge; on the other hand, a network with $\mathrm{Re}(\lambda(\mathbf{J})) \ll 0$ would decay the signal exponentially; this could be useful in some part to decay high order oscillation, but having too much signal loss may be harmful. In the papers (Chang et al., 2018; Haber and Ruthotto, 2017) the authors look therefore for architectures where the condition $\mathrm{Re}(\lambda(\mathbf{J})) \simeq 0$ holds. Consider the case of transfomations

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}) = \rho(\mathbf{W}\mathbf{x} + \mathbf{b})$$

(where $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$ and $\rho(x)$ is a non-decreasing activation function), then

$$\mathbf{J}(\mathbf{x}) = \nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}) = \mathrm{diag}(\dot{\rho}(\mathbf{W}\mathbf{x} + \mathbf{b}))\mathbf{W}$$

Notice that, if $\mathbf{W}$ is anti-symmetric

$$\mathbf{W}^T = -\mathbf{W}$$

then it holds $\mathrm{Re}(\lambda(\mathbf{W})) = 0$, and the same thing hold for $\mathbf{J}$, since $\dot{\rho} \geq 0$. Such a condition can be easily imposed by taking $\mathbf{W}_k = \mathbf{K}_k - \mathbf{K}_k^T$, where $\mathbf{K}_k$ is the trainable parameter. Another possibility is to take

$$\begin{pmatrix} \mathbf{x}_{k+1} \\ \mathbf{z}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_k \\ \mathbf{z}_k \end{pmatrix} + h\rho\left(\begin{pmatrix} \mathbf{0} & \mathbf{K}_k \\ -\mathbf{K}_k^T & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{x}_k \\ \mathbf{z}_k \end{pmatrix} + \mathbf{b}_k\right)$$

where $\mathbf{z}_k$ is an additional propagating variable (with $\mathbf{z}_0 = \mathbf{0}$). Other variants include considering different numerical schemes to approximate the ODE (1) or architectures inspired by Hamiltonian systems (see (Haber and Ruthotto, 2017; Chang et al., 2018)).

2.1 Reversible networks

Some residual networks are *reversible*: the output at layer $k+1$ can be used to evaluate the output at layer k. While this is a property that could be useful e.g. in generative modeling, such property was noticed in (Gomez et al., 2017), where the authors used this idea to improve (memory) efficiency. Indeed, if the network is reversible, outputs of intermediate layers need not be stored for backpropagation. Assume for example that the output of the network at layer $k+1$ is given by $(\mathbf{x}_{k+1}, \mathbf{z}_{k+1})$, where

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \mathcal{F}(\mathbf{z}_k)\\ \mathbf{z}_{k+1} &= \mathbf{z}_k + \mathcal{G}(\mathbf{x}_{k+1})\end{aligned}$$

Then we can get $(\mathbf{x}_k, \mathbf{z}_k)$ given $(\mathbf{x}_{k+1}, \mathbf{z}_{k+1})$ as

$$\begin{aligned}\mathbf{z}_k &= \mathbf{z}_{k+1} - \mathcal{G}(\mathbf{x}_{k+1})\\ \mathbf{x}_k &= \mathbf{x}_{k+1} - \mathcal{F}(\mathbf{z}_k)\end{aligned}$$

Question: Consider the network above with $\mathcal{F}(\mathbf{z}) = \alpha\mathbf{z}$ and $\mathcal{G}(\mathbf{x}) = \beta\mathbf{x}$ (as in (Chang et al., 2018)). Is this network stable (in the sense of Section 2)?

We have that

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha\mathbf{z}_k\\ \mathbf{z}_{k+1} &= \mathbf{z}_k + \beta\mathbf{x}_{k+1}\\ \mathbf{x}_k &= \mathbf{x}_{k-1} + \alpha\mathbf{z}_{k-1}\\ \mathbf{z}_k &= \mathbf{z}_{k-1} + \beta\mathbf{x}_k\end{aligned}$$

Subtracting the third equation from the first equation above, we have

$$\begin{aligned}\mathbf{x}_{k+1} - \mathbf{x}_k &= \mathbf{x}_k - \mathbf{x}_{k-1} + \alpha\mathbf{z}_k - \alpha\mathbf{z}_{k-1}\\ \mathbf{x}_{k+1} - \mathbf{x}_k &= \mathbf{x}_k - \mathbf{x}_{k-1} + \alpha\beta\mathbf{x}_k\\ \mathbf{x}_{k+1} - (2+\alpha\beta)\mathbf{x}_k - \mathbf{x}_{k-1} &= 0\end{aligned}$$

As we saw in the second lecture, for the above propagation to be stable we need the roots of the polynomial $p(x) = x^2 - (2+\alpha\beta)x - 1$ to be bounded (in absolute value) by 1. This simple example already sheds lights on the difficulties of designing architectures with a set of desired properties.

References

Chang, B., L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham
2018. Reversible architectures for arbitrarily deep residual neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chang, B., L. Meng, E. Haber, F. Tung, and D. Begert
2017. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*.

Gomez, A. N., M. Ren, R. Urtasun, and R. B. Grosse
2017. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*, Pp. 2214–2224.

Haber, E. and L. Ruthotto
2017. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004.

He, K., X. Zhang, S. Ren, and J. Sun
2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pp. 770–778.

He, K., X. Zhang, S. Ren, and J. Sun
2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, Pp. 630–645. Springer.

Veit, A., M. J. Wilber, and S. Belongie
2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, Pp. 550–558.