Nonlinear Acceleration of CNNs

Damien Scieur, Edouard Oyallon, Alexandre d'Aspremont, Francis Bach

NONLINEAR ACCELERATION OF CNNS

Damien Scieur*, **Edouard Oyallon**†, **Alexandre d'Aspremont*** **and Francis Bach***
*DI, Ecole Normale Supérieure, Université PSL; Sierra team, INRIA Paris
†CVN, CentraleSupélec, Université Paris-Saclay; Galen team, INRIA Saclay

ABSTRACT

The Regularized Nonlinear Acceleration (RNA) algorithm is an acceleration method capable of improving the rate of convergence of many optimization schemes such as gradient descend, SAGA or SVRG. Until now, its analysis is limited to convex problems, but empirical observations shows that RNA may be extended to wider settings. In this paper, we investigate further the benefits of RNA when applied to neural networks, in particular for the task of image recognition on CIFAR10 and ImageNet. With very few modifications of exiting frameworks, RNA improves slightly the optimization process of CNNs, *after training*.

1 INTRODUCTION

Successful Deep Convolutional Neural Networks(CNNs) for large-scale classification are typically optimized through a SGD algorithm (Krizhevsky et al., 2012). Refining this optimization scheme is a complicated task because it requires a lot of engineering that is not well understood mathematically (Mallat, 2016; Bach, 2014). Instead, we propose to wrap an adhoc acceleration technique applied at regular interval, to which we refer as Regularized Nonlinear Acceleration algorithm (RNA) (Scieur et al., 2016). It is generic as it does not depend on the optimization algorithm, but simply requires several successive steps of the flow of gradient which is a minimal adaptation in many frameworks. This meta-algorithm has been applied successfully to the gradient descend in the smooth and strongly convex cases, with convergence and rate guarantees (Scieur et al., 2016). Recent works Scieur et al. (2017) show in addition that it improves standard optimization scheme such as SAGA or SVRG (Defazio et al., 2014; Johnson & Zhang, 2013), which indicates it is a strong candidate as an accelerated method in stochastic non convex cases.

RNA is an ideal meta-learning algorithm for deep CNNs, because contrary to many acceleration methods **?**Güler (1992), the optimization can be performed off-line and does not involve any extra-learning process that can be potentially expensive. It means one can study the optimization *a posteriori*. Its computations are not expensive, because it consists in a well-chosen linear combinations of several optimization steps. It requires roughly to invert a matrix which is small, in comparison with the number of parameters.

In this work, we study its application to several recent architectures, like a ResNet (He et al., 2016), applied to challenging standard datasets, like CIFAR10 or ImageNet. Our contribution are as follow: first we demonstrate that it is possible to achieve an accuracy similar to the final epoch two times earlier; secondly, we show that RNA improves slightly the final classification performances, at no cost. We provide an implementation that can be incorporated with only few lines of code in many python deep learning frameworks, like PyTorch[1].

2 ACCELERATING WITH REGULARIZED NONLINEAR ACCELERATION

This section intuitively describe the derivation of RNA procedure and we refer the reader to (Scieur et al., 2016) for more extensive explanations and theoretical guarantees. For the sake of simplicity, we consider a sequence $\{\theta_k\}_{0\leq k\leq K}$ of $K+1$ elements of $\mathbb{R}^d$ resulting from the different successive steps of an iterative optimization algorithm. For example, each θ_k could correspond to the parameters of a neural network at epoch i trained via a gradient descent algorithm, i.e.:

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k) \ ,$$

[1]Code can be found ???

with η the step size (or learning rate) of the algorithm. Local minimization of f is naturally achieved by θ^* where $\nabla f(\theta^*) = 0$. RNA aims to linearly combine the parameters $\{\theta_k\}_k$ into $\hat{\theta}$:

$$\hat{\theta} = \sum_{k \leq K} c_k \theta_k \text{ s.t. } \sum_{k \leq K} c_k = 1 \,, \tag{1}$$

so that $\nabla f(\hat{\theta})$ becomes smaller. In other terms, RNA output $\hat{\theta}$ which solves approximatively

$$\min_c \left\| \nabla f \left(\sum_{k \leq K} c_k \theta_k \right) \right\|^2 \quad \text{subject to } \sum_{k \leq K} c_k = 1. \tag{2}$$

In the next subsection, we describe the algorithm and intuitively explain the acceleration mechanism when using the optimization method (1), because this restrictive setting makes the analysis simpler.

2.1 Regularized Nonlinear Acceleration Algorithm

In practice, (2) corresponds to a difficult task. Instead, we will assume that the function f is *approximately* quadratic in the neighbourhood of $\{\theta_k\}_{k \leq K}$. This approximation is common in optimization for the design of (quasi-)second order methods, such as the Newton's method or BFGS. Thus, ∇f can be considered almost as a linear function, which means:

$$-\nabla f \left(\sum_{k \leq K} c_k \theta_k \right) \approx \sum_{k \leq K} c_k \nabla f \left(\theta_k \right). \tag{3}$$

From a finite difference scheme, one can easily recover $\{\nabla f(\theta_k)\}_k$ from the iterates in (1), because for any k, $-\eta \nabla f(\theta_k) = (\theta_{k+1} - \theta_k)$. As linearized iterates of a flow tends to be aligned, minimizing the ℓ^2-norm of (3) requires to incorporate some regularization to avoid ill-conditionning effects:

$$\min_c \|Rc\|^2 + \lambda \|c\|^2 \quad \text{subject to } \sum_{k \leq K} c_k = 1.$$

Where $R = [\theta_1 - \theta_0, \ldots, \theta_K - \theta_{K-1}]$. This corresponds exactly to the combination of steps 2 and 3 of Algorithm 1. Similar ideas hold is the stochastic case Scieur et al. (2017), under limited assumption on the signal to noise ratio.

Algorithm 1 Regularized Nonlinear Acceleration (**RNA**), computational complexity of each step in paranthesis.

Input: Sequence of vectors $\{\theta_0, \theta_1, \ldots, \theta_k\} \in \mathbb{R}^d$, regularization parameter $\lambda > 0$.

1: Compute $R = [\theta_1 - \theta_0, \ldots, \theta_K - \theta_{K-1}]$ $O(K)$

2: Solve $(R^T R + \lambda I) z = \mathbf{1}$. $O(K^2 d + K^3)$

3: Normalize $c = z / (\sum_{k \leq K} z_k)$. $O(K)$

Output: $\hat{\theta} = \sum_{k \leq K} c_k \theta_k$. $O(Kd)$

2.2 Practical Usage

We release a software based on PyTorch that consists in minimal modifications of existing standard procedures. As claimed, the RNA procedure does **not** require any access to the data, yet simply to store regularly some parameters in a buffer. An on the fly acceleration on CPU is achievable, because all the considered matrix are small and storage as well does not consume resources..

3 Applications to CNNs

3.1 Classification pipelines

Because the RNA algorithm is generic, it can be easily applied on many different existing training codes. We used the method with various CNN on CIFAR10 and ImageNet; the first dataset consists of $50k$ RGB images of size 32^2 whereas the latter is more challenging with $1.2M$ images of size 224^2. Data augmentation via random translation is applied. In both cases, we trained our CNN via a SGD of momentum 0.9 and a weight decay of 10^{-5}, until convergence. The initial learning rate is 0.1, and is decreased by 10 at epoch $150, 250$ and $30, 60, 90$ respectively for CIFAR and ImageNet.

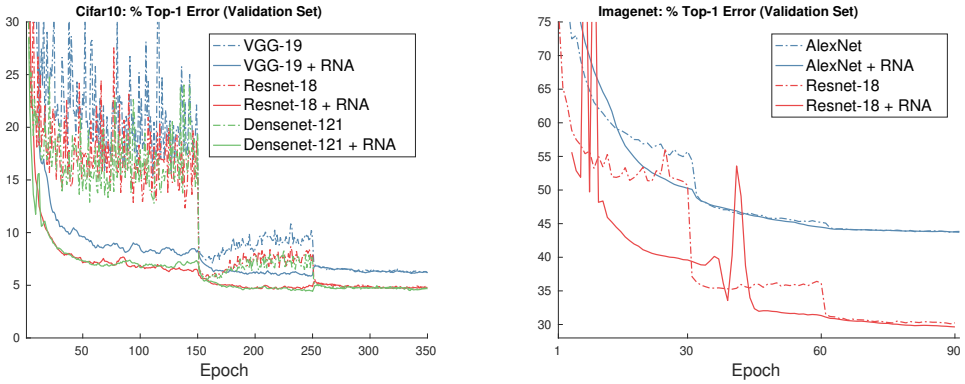

Figure 1: Comparison of Top-1 error between vanilla and extrapolated network.

Network	Vanilla	Extrapolated
VGG	6.18%	5.86%
Resnet18	4.71%	4.64%
Densenet 121	4.50%	4.42%

Network	Vanilla	Extrapolated
AlexNet	43.72%	43.72%
Resnet18	30.11%	29.64%

Table 1: Lowest Top-1 error on CIFAR10 (Left) and ImageNet (Right)

For ImageNet, we used the fame AlexNet (Krizhevsky et al., 2012) and ResNet (He et al., 2016) because they are standard architectures on vision. For CIFAR dataset, we employed the standard VGG, ResNet and DenseNet (Huang et al., 2017). The AlexNet is trained with drop-out (Srivastava et al., 2014) on its fully connected layers, whereas the others CNNs are trained with Batch-normalization (Ioffe & Szegedy, 2015).

The RNA was used off-line using the parameters $K = 10$ and $\lambda = 10^{-8}$, using the parameters produced by the final iteration of each epochs. We then reported the new classification performances at each epoch.

3.2 Numerical results

Figure 1 corresponds to the performances on the *validation set*, of the vanilla and extrapolated CNN via RNA, at each epoch. Observe that the RNA accuracies are smoother than the original CNNs which shows an effective variance reduction. In addition, we observe the impact of acceleration: quickly, the extrapolated network presents good generalization performance, even competitive with the best one. Several iterations after a learning rate drop are necessary to obtain an acceleration because it corresponds to a brutal change in the optimization. Furthermore, selecting the λ hyper parameters can be tricky: for example, a larger λ at epoch 40 removes the outlier validation performance of Figure 1, for the ResNet-18. This is a direct implication of our decision to use generic parameters to make the comparison as fair as possible, but adaptive strategy have been discussed in Scieur et al. (2017).

The Tables 3.2 reports the lowest validation error of the vanilla architectures compared to their extrapolated counterpart. In every setting, the off-line optimization performed by the RNA has only slightly improved the final accuracy of the CNN. But we remind that these improvement have been obtained *after* the training procedure, in an offline fashion, without extra-learning.

4 Future work

EDOUARD: possible de retirer, j'anticipe les reviewers

Several questions are opened: is it possible to keep performances while restarting from the extrapolated CNN? Can we improve the RNA performances by feeding more succesive iterations to the algorithm? Can we extend current proofs to deep pipelines?

ACKNOWLEDGMENTS

REFERENCES

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *arXiv preprint arXiv:1412.8690*, 2014.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pp. 1646–1654, 2014.

Osman Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

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*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, pp. 3, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065): 20150203, 2016.

Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Advances In Neural Information Processing Systems*, pp. 712–720, 2016.

Damien Scieur, Francis Bach, and Alexandre d'Aspremont. Nonlinear acceleration of stochastic algorithms. In *Advances in Neural Information Processing Systems*, pp. 3985–3994, 2017.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.