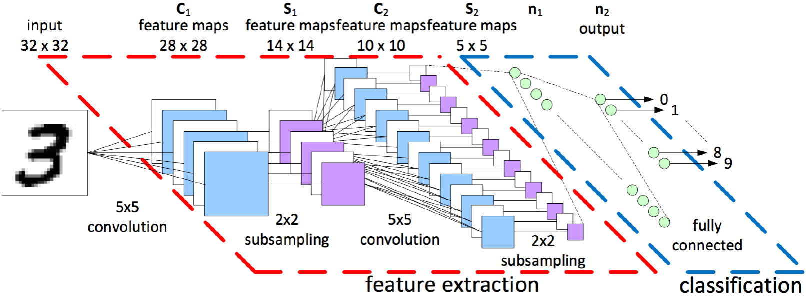
Momentum in Stochastic Gradient Descent and Neural Architecture Design

Bao Wang
Department of Mathematics, UCLA

Joint work with Tan Nguyen, Rich Baraniuk, Andrea Bertozzi, and Stan Osher

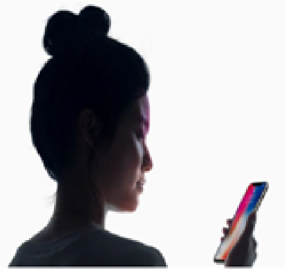
Deep Learning (DL)



DL = Big Data + Deep Nets + SGD + HPC

Deep Learning: Revolution in Technology

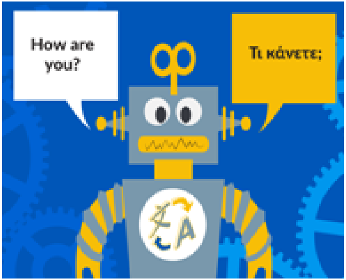
Face ID



Autonomous Cars

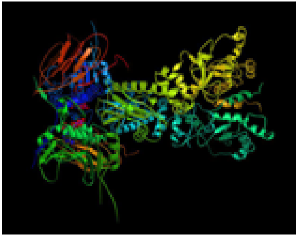
Alpha Go

Machine Translation

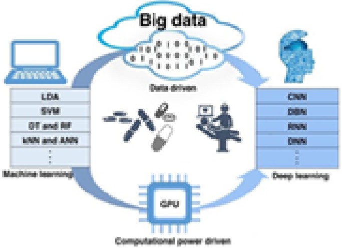


Deep Learning: Revolution in Science

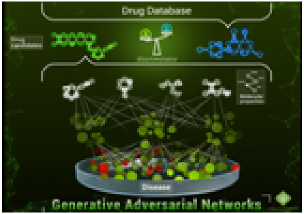
Protein Structure Prediction



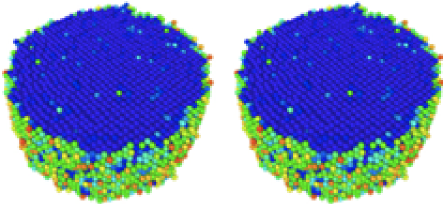
Drug Discovery



Molecular Generation



Material Design



Momentum in SGD

Accelerate convergence

Better generalizable models

Momentum in Deep Nets Design

Principled approach

Easier to train & Better generalizability

Momentum in Optimization

Machine Learning – Empirical Risk Minimization (ERM)

Consider training a machine learning model:

$$y = g(\mathbf{x}, \mathbf{w}), \mathbf{w} \in \mathbb{R}^d,$$

which is parameterized by $\mathbf{w}$, $(\mathbf{x}, y)$ is an input data-predicted label pair.

$\mathbf{w}$ is usually learned by solving the following ERM problem:

$$\min_{\mathbf{w}} f(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(g(\mathbf{x}_i, \mathbf{w}), y_i),$$

where $\mathcal{L}$ is the loss between the predicted label $\hat{y}_i$ and the ground-truth label y_i.

For classification, typically we have:

$$\mathcal{L}(\hat{y}_i, y_i) = -\sum_{j=1}^{M} y_i^j \log(p_i^j),$$

where p_i^j is the predicted probability that y_i is belong to y_i^j's class.

$f(\mathbf{w})$ can be highly nonconvex, and $d \sim 10^{10}$, $N \sim 10^9$.

Convex Optimization – Gradient Descent

Suppose $f(\mathbf{w})$ is convex and L-smooth. Start from $\mathbf{w_0}$, and perform:

$$\mathbf{w}_k = \mathbf{w}_{k-1} - s\nabla f(\mathbf{w}_{k-1}), \text{ and } s = 1/L.$$

Gradient descent has a convergence rate $O(1/k)$ for convex smooth functions.

Consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{L}\mathbf{w} - \mathbf{w}^T\mathbf{b},$$

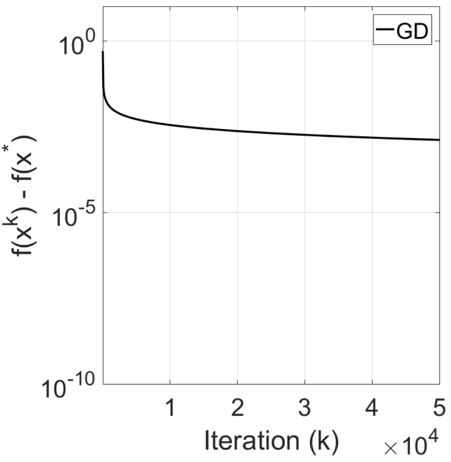
where

$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}_{1000\times 1000},$$

and $\mathbf{b}$ is a 1000-dim vector whose first entry is 1 and all the other entries are 0.

A. Cauchy, 1847

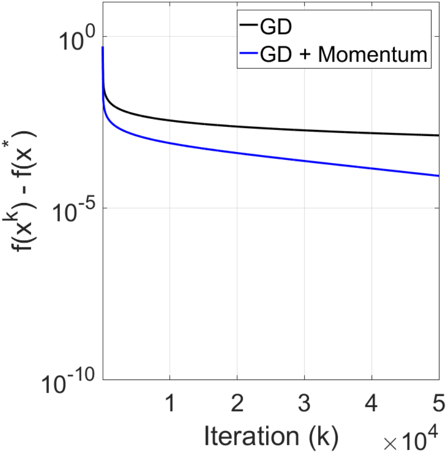
Convex Optimization – Gradient Descent



Convex Optimization – Gradient Descent + Momentum

$$\mathbf{v}_k = \mathbf{w}_{k-1} - s\nabla f(\mathbf{w}_{k-1}),$$
$$\mathbf{w}_k = \mathbf{v}_k + \mu(\mathbf{v}_k - \mathbf{v}_{k-1}).$$

$O(1/k)$ convergence rate!



Heavy Ball

$$\mathbf{w}_k = \mathbf{w}_{k-1} - s\nabla f(\mathbf{w}_{k-1}) + \mu(\mathbf{w}_{k-1} - \mathbf{w}_{k-2}),$$

$O(1/k)$ convergence rate!

B. Polyak, 1964

Why momentum works

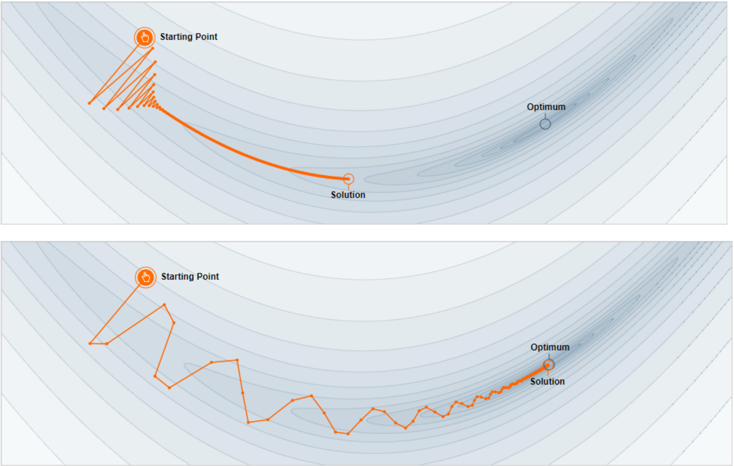


Figure: Top: no momentum; Bottom: with momentum.

G. Goh, Why momentum really works. Distill, 2017

Other Variants (Look-ahead)

Let $\mathbf{u}_k := \mathbf{v}_k - \mathbf{v}_{k-1}$ and $\mathbf{g}_k := \nabla f(\mathbf{v}_{k-1} + s\mathbf{u}_{k-1})$.

Sutskever et al. 2013

$$\begin{cases} \mathbf{v}_k = \mu \mathbf{v}_{k-1} + s\mathbf{g}_k, \\ \mathbf{w}_k = \mathbf{w}_{k-1} - \mathbf{v}_k. \end{cases}$$

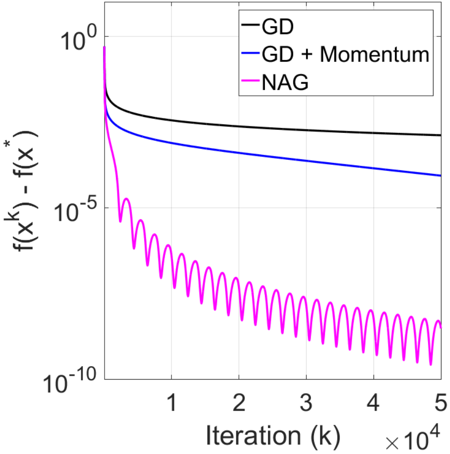
PyTorch

$$\begin{cases} \mathbf{v}_k = \mu \mathbf{v}_{k-1} + \mathbf{g}_k, \\ \mathbf{w}_k = \mathbf{w}_{k-1} - s\mathbf{v}_k. \end{cases}$$

Convex Optimization – Nesterov Accelerated Gradient (NAG)

$$\mathbf{v}_k = \mathbf{w}_{k-1} - s\nabla f(\mathbf{w}_{k-1}),$$
$$\mathbf{w}_k = \mathbf{v}_k + \frac{k-1}{k+2}(\mathbf{v}_k - \mathbf{v}_{k-1}).$$

$O(1/k^2)$ convergence rate!



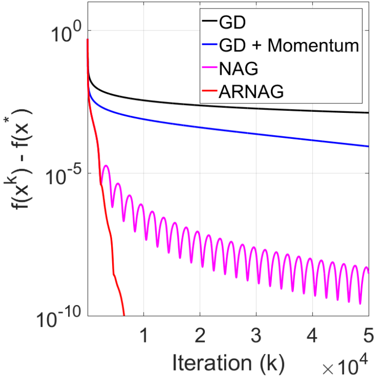
Convex Optimization – Adaptive Restart NAG (ARNAG)

$$\mathbf{v}_k = \mathbf{w}_{k-1} - s\nabla f(\mathbf{w}_{k-1}),$$

$$\mathbf{w}_k = \mathbf{v}_k + \frac{t-1}{t+2}(\mathbf{v}_k - \mathbf{v}_{k-1}),\ t \text{ increase by1 if objective decay, else } t = 1.$$

$O(e^{-\alpha k})$ convergence rate with an extra sharpness assumption!

Sharpness: $\frac{\mu}{r} d(x, X^*)^r < f(x) - f^*, \quad \mu > 0, r > 1.$



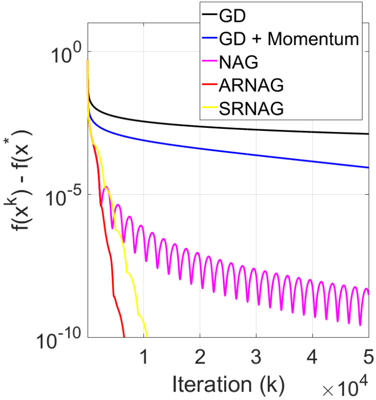
V. Roulet et al. NIPS 2017

Convex Optimization – Scheduled Restart NAG (SRNAG)

$$\mathbf{v}_k = \mathbf{w}_{k-1} - s\nabla f(\mathbf{w}_{k-1}),$$

$$\mathbf{w}_k = \mathbf{v}_k + \frac{t-1}{t+2}(\mathbf{v}_k - \mathbf{v}_{k-1}),\ t \text{ increase by1 within some iterations, else } t = 1.$$

$O(e^{-\beta k})$ convergence rate with an extra sharpness assumption!



V. Roulet et al. NIPS 2017

What If We Do Not Have Exact Gradient?

In ERM,

$$\min_{\mathbf{w}} f(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{w}) := \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(g(\mathbf{x}_i, \mathbf{w}), y_i),$$

when $N \gg 1$, compute $\nabla f(\mathbf{w})$ will be very expensive.

Stochastic Gradient:

$$\nabla f(\mathbf{w}) \approx \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}), \text{ with } [n] \subset [N] \text{ and } n \ll N.$$

Can NAG still accelerate convergence with Stochastic Gradient?

A Motivating Example – Gaussian Noise Corrupted Gradient – Case I

Consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{L}\mathbf{w} - \mathbf{w}^T\mathbf{b},$$

where

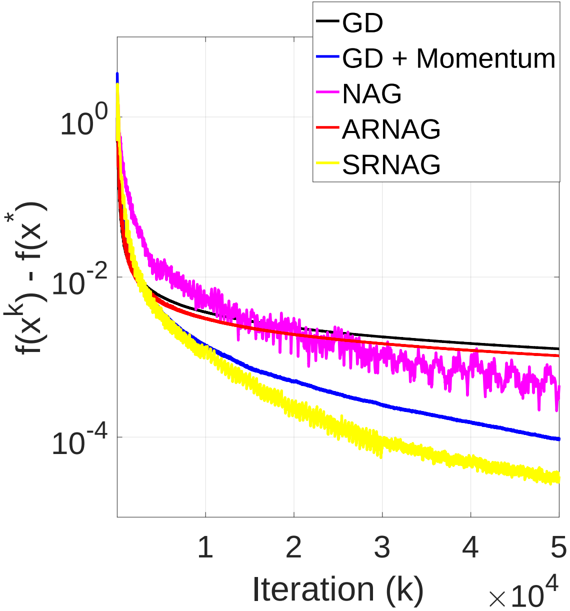
$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}_{1000\times 1000},$$

and $\mathbf{b}$ is a 1000-dim vector whose first entry is 1 and all the other entries are 0.

Gaussian Noise Corrupted Gradient:

$$\nabla f(\mathbf{w}) = \mathbf{L}\mathbf{w} - \mathbf{b} + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(0, (\frac{0.1}{\lfloor k/100 \rfloor + 1})^2).$$

A Motivating Example – Gaussian Noise Corrupted Gradient – Case I



A Motivating Example – Gaussian Noise Corrupted Gradient – Case II

Consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{L}\mathbf{w} - \mathbf{w}^T\mathbf{b},$$

where

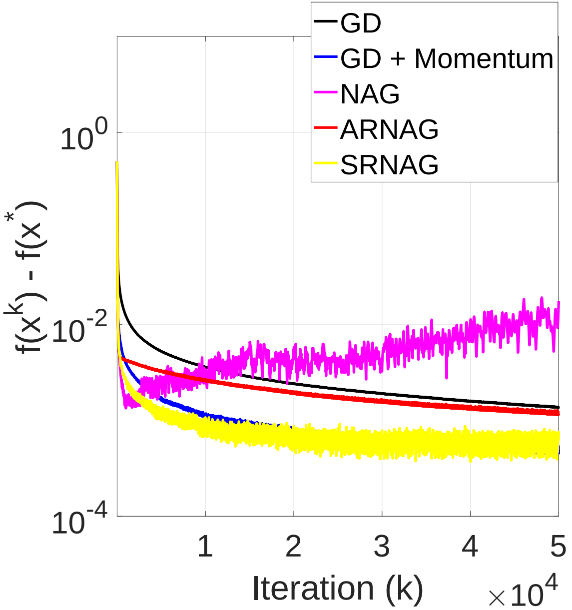
$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}_{1000\times1000},$$

and $\mathbf{b}$ is a 1000-dim vector whose first entry is 1 and all the other entries are 0.

Gaussian Noise Corrupted Gradient:

$$\nabla f(\mathbf{w}) = \mathbf{L}\mathbf{w} - \mathbf{b} + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(0, 0.001^2).$$

A Motivating Example – Gaussian Noise Corrupted Gradient – Case II



A Motivating Example – Logistic Regression – Case III

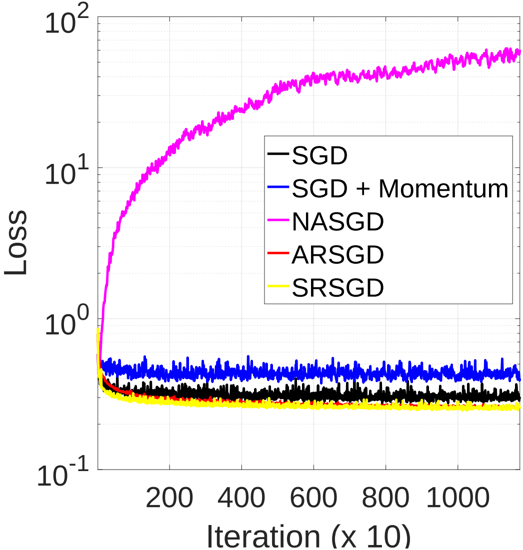


Figure: Training loss of logistic regression for MNIST classification.

Analysis

NAG:

Given a convex function $f : \mathbb{R}^d \to \mathbb{R}$ with smoothness parameter L. Given any point $\mathbf{w} \in \mathbb{R}^d$ an exact first order oracle returns a pair $(f_L(\mathbf{x}), g_L(\mathbf{w})) \in \mathbb{R} \times \mathbb{R}^d$ so that for all $\mathbf{u} \in \mathbb{R}^d$, we have

$$0 \leq f(\mathbf{u}) - (f_L(\mathbf{w}) + \langle g_L(\mathbf{w}), \mathbf{u} - \mathbf{w} \rangle) \leq \frac{L}{2} \|\mathbf{u} - \mathbf{w}\|^2.$$

The first inequality follows from convexity and the second from the smoothness condition.
An inexact oracle returns for any given point $\mathbf{w} \in \mathbb{R}^d$ a pair $(f_{\delta,L}(\mathbf{x}), g_{\delta,L}(\mathbf{w})) \in \mathbb{R} \times \mathbb{R}^d$ so that for all $\mathbf{u} \in \mathbb{R}^d$ we have

$$0 \leq f(\mathbf{u}) - (f_{\delta,L}(\mathbf{w}) + \langle g_{\delta,L}(\mathbf{w}), \mathbf{u} - \mathbf{w} \rangle) \leq \frac{L}{2} \|\mathbf{u} - \mathbf{w}\|^2 + \delta.$$

It has the same picture as before except now there is some δ slack between the linear and parabola approximations.

Theorem Given access to δ-inexact first-order oracle GD spits out a point $\mathbf{w}^k$ after k steps so that

$$f(\mathbf{w}^k) - f(\mathbf{w}^*) \leq O(L/k) + \delta.$$

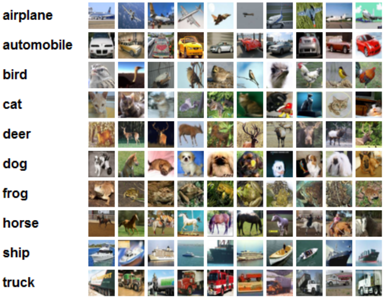
NAG on the other hand gives

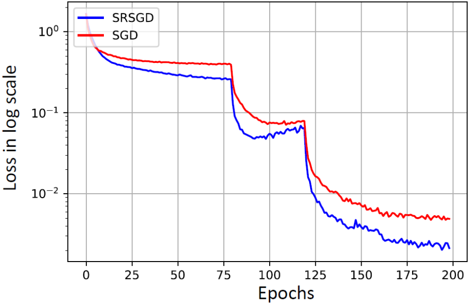
$$f(\mathbf{w}^k) - f(\mathbf{w}^*) \leq O(L/k^2) + O(k\delta).$$

Adaptive Restart NAG: restart too often, degenerates to GD.

O. Devolder, F. Glineur, and Y. Nesterov, First-Order Methods of Smooth Convex Optimization With Inexact Oracle. Math Prog A, 2014.

SRNAG for Deep Learning – ResNet for CIFAR10 and CIFAR100 Classification





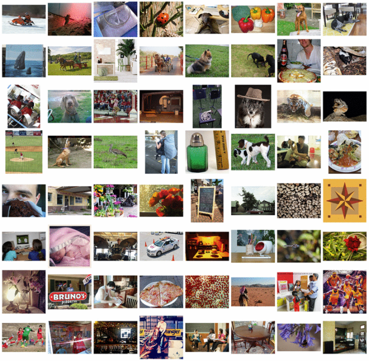
SRNAG for Deep Learning – ResNet for CIFAR10 and CIFAR100 Classification

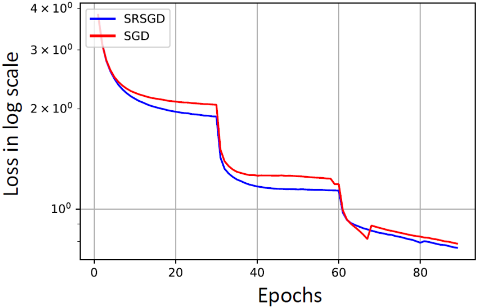
Network	# Params	SGD (baseline)	SRSGD (linear)	SRSGD (exponential)	Improvement(linear/exponential)
PreResNet-110	1.1M	94.75 ± 0.14 (93.63 (He et al., 2016))	95.07 ± 0.13	95.00 ± 0.47	0.32/0.25
PreResNet-290	3.0M	94.95 ± 0.23	95.63 ± 0.15	95.50 ± 0.18	0.68/0.55
PreResNet-470	4.9M	95.08 ± 0.10	95.82 ± 0.09	95.51 ± 0.19	0.74/0.43
PreResNet-650	6.7M	95.13 ± 0.14	96.00 ± 0.07	95.60 ± 0.13	0.87/0.47
PreResNet-830	8.6M	95.00 ± 0.23	95.91 ± 0.10	95.71 ± 0.13	0.91/0.71
PreResNet-1001	10.3M	95.16 ± 0.19 (95.08 (He et al., 2016))	96.13 ± 0.07	95.87 ± 0.10	0.97/0.71

Network	# Params	SGD (baseline)	SRSGD (linear)	SRSGD (exponential)	Improvement(linear/exponential)
PreResNet-110	1.2M	76.25 ± 0.20	76.51 ± 0.23	76.50 ± 0.39	0.26/0.25
PreResNet-290	3.0M	78.22 ± 0.21	78.51 ± 0.27	78.42 ± 0.20	0.29/0.20
PreResNet-470	4.9M	78.57 ± 0.30	79.29 ± 0.32	79.36 ± 0.18	0.72/0.79
PreResNet-650	6.7M	78.73 ± 0.14	79.64 ± 0.25	79.59 ± 0.21	0.91/0.86
PreResNet-830	8.6M	78.81 ± 0.22	79.78 ± 0.23	79.80 ± 0.25	0.97/0.99
PreResNet-1001	10.4M	79.13 ± 0.20 (77.29 (He et al., 2016))	80.25 ± 0.11	80.47 ± 0.19	1.12/1.34

Figure: Top: CIFAR10; Bottom: CIFAR100

SRNAG for Deep Learning – ResNet for ImageNet Classification





ImageNet Classification

Network	# Params	SGD	SRSGD	Improvement
ResNet-50	25.56M	75.83	76.24	0.41
ResNet-101	44.55M	77.6	78.06	0.44
ResNet-152	60.19M	78.10	78.71	0.61
ResNet-200	64.67M	77.82	78.92	1.10

Image Classification

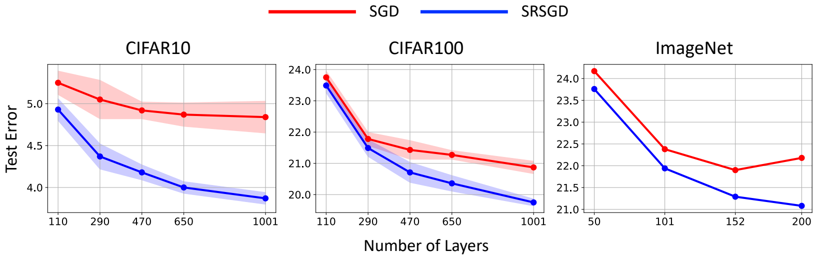


Figure: Error vs. depth of ResNet.

Momentum in Deep Neural Nets

Rethinking Deep Residual Nets (ResNets)

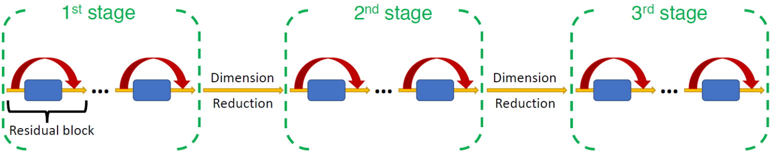
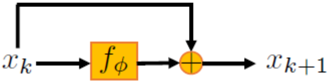


Figure: ResNet

Residual mapping: $x_{k+1} = x_k + f(x_k, w_k)$.

Residual block



How about $x_{k+1} = x_k - \mu f(x_k, w_k)$?

What if

$$y_{k+1} = x_k - \mu f(x_k, w_k),$$

$$x_{k+1} = y_{k+1} + \nu(y_{k+1} - y_k)?$$

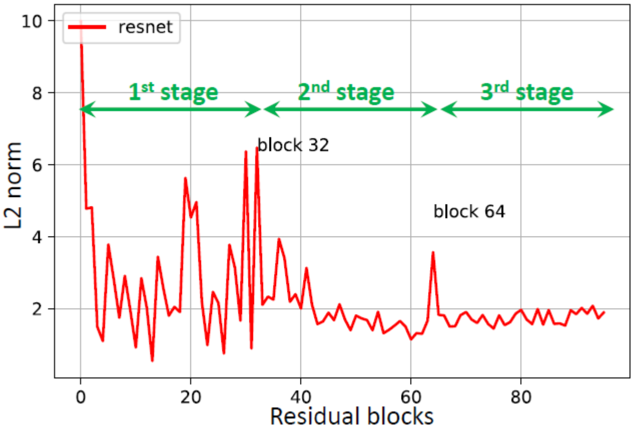
ResNet vs. Gradient Descent

Gradient descent

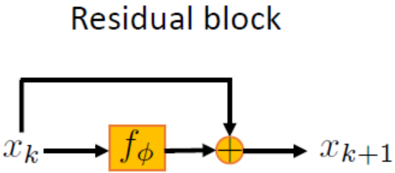
$$x_{k+1} = x_k - \mu \nabla f(x_k),$$

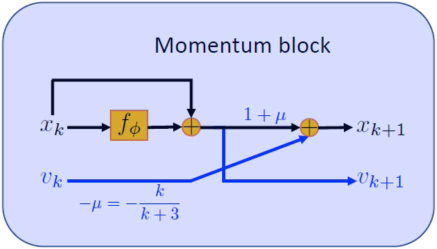
and we have

$$\lim_{k \to \infty} \|\nabla f(x_k)\|_2 = 0.$$

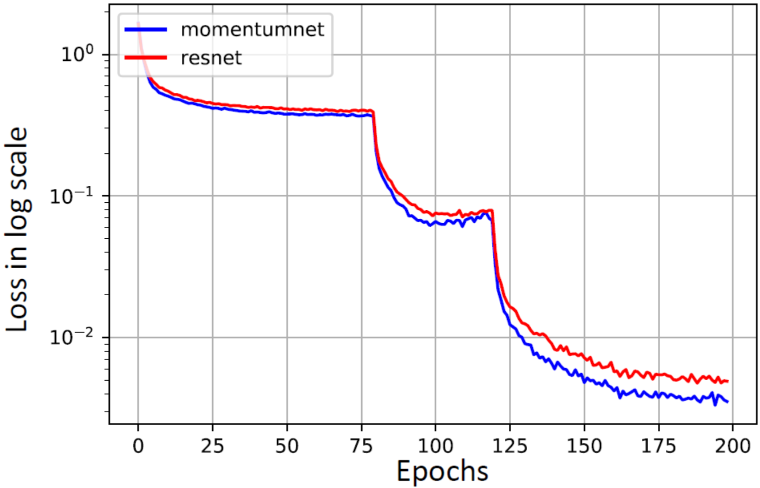


Deep Momentum Nets





Deep Momentum Nets Versus Deep Residual Nets



CIFAR10 Classification

Network	# Params	ResNet (baseline)	MomentumNet (linear)	Improvement(linear/exponential)
PreResNet-110	1.1M	94.75 ± 0.14 (93.63 (He et al., 2016))	94.69 ± 0.12	-0.06
PreResNet-290	3.0M	94.95 ± 0.23	95.09 ± 0.11	0.14
PreResNet-470	4.9M	95.08 ± 0.10	95.09 ± 0.12	0.01
PreResNet-650	6.7M	95.13 ± 0.14	95.21 ± 0.13	0.08
PreResNet-830	8.6M	95.00 ± 0.23	95.31 ± 0.22	0.31
PreResNet-1001	10.3M	95.16 ± 0.19 (95.08 (He et al., 2016))	95.35 ± 0.15	0.19

ODE Models

$$\dot{X} + \nabla f(X) = 0,$$

vs.

$$\ddot{X} + \frac{3}{t}\dot{X} + \nabla f(X) = 0.$$

Su, Boyd, Candes, NIPS, 2014

Future Directions

Optimal restart scheduling in SRSGD.

How to generalize SRSGD to adaptive step size settings.

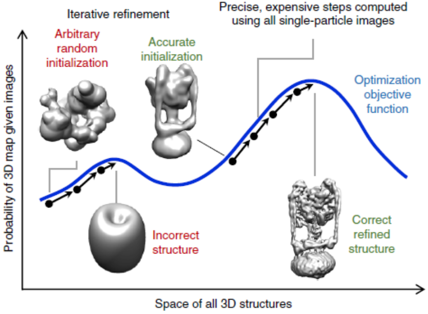
Neural architecture search with momentum units.

Microscopic Imaging (Cryo-EM)

RELION: Bayesian likelihood framework for cryo-EM structure determination:

$$\arg\max_{V_{1,\cdots,K}} \log p(V_{1,\cdots,K}|X_{1,\cdots,N}) =$$
$$\arg\max_{V_{1,\cdots,K}} \sum_{i=1}^{N} \log \sum_{j=1}^{K} \frac{1}{K} \int p(X_i, \phi_i | V_j) d\phi_i + \log p(V_{1,\cdots,K}).$$

The aim of the optimization is to find the 3D structures (V_1 to V_K) that best explain the observed images (X_1 to X_N) by marginalizing over class assignment (j) and the unknown pose variable (ϕ_i) which describes a 3D rotation and a 2D translation for each single-particle image.



Stochastic Gradient Descent!

A. Punjani, J. Rubinstein, D. Fleet, and M. Brubaker, Nature Methods, 2017.

Thank You

I. Scheduled Restart NAG Momentum

- I.1 Accelerate convergence
- I.2 Better generalization accuracy

II. Momentum in Neural Architecture Design

- II.1 Speed-up training
- II.2 Better generalization accuracy
- II.3 Mathematically mechanistic design

References:

1. B. Wang*, T. Nguyen*, A. Bertozzi[#], R. Baraniuk[#], and S. Osher[#], Scheduled Restart Momentum for Accelerated Stochastic Gradient Descent, Submitted, 2020.

2. T. Nguyen*, B. Wang*, A. Bertozzi[#], R. Baraniuk[#], and S. Osher[#], Deep Momentum Networks, Submitted, 2020.