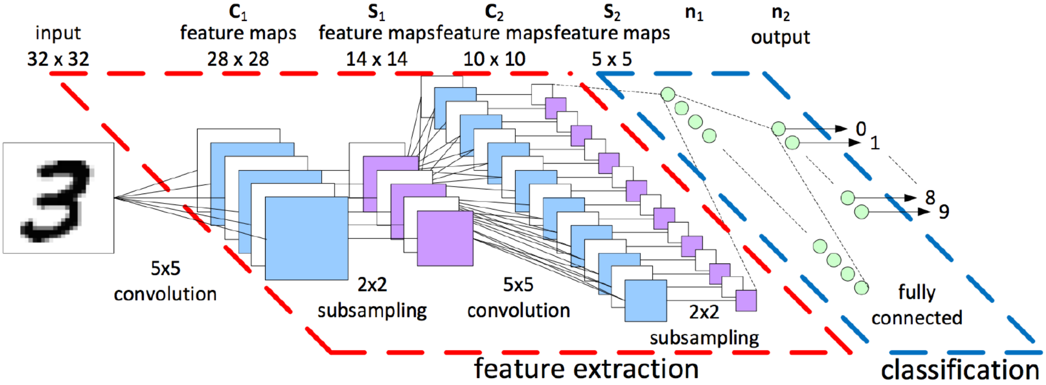
PDEs Principled Trustworthy Deep Learning

Bao Wang
Department of Mathematics
Scientific Computing and Imaging Institute University of Utah

Deep Learning (DL)



DL = Big Data + Deep Nets + SGD + HPC

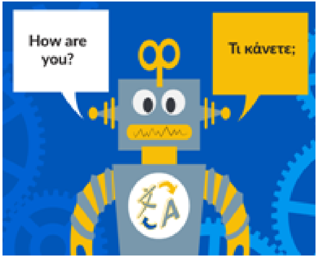
Deep Learning: Revolution in Technology

Face ID

Autonomous Cars

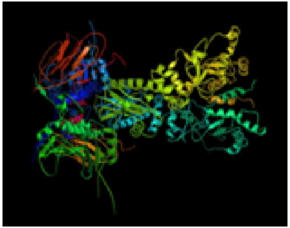
Alpha Go

Machine Translation

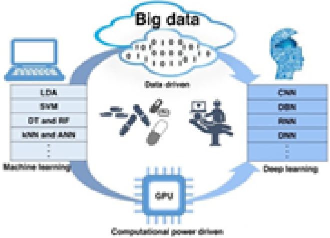


Deep Learning: Revolution in Science

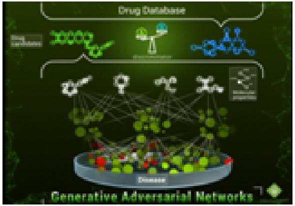
Protein Structure Prediction



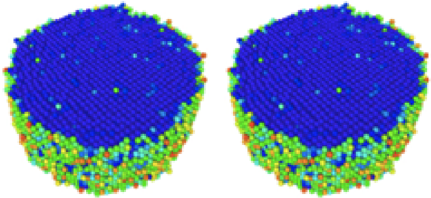
Drug Discovery



Molecular Generation



Material Design



However, Deep Learning is Not Trustworthy!

Trustworthy deep learning:

1. Robust deep learning
2. Accurate deep learning
3. Efficient deep learning
4. Private deep learning

...

with theoretical guarantees!

Adversarial Vulnerability of Deep Neural Nets

Original Inputs

Modified Inputs

Wrong ML Detection

Evtimov et al., CVPR, 2018

Deep Learning is Very Expensive

AlphaGO	**Lee Se-dol**
1202 CPUs, 176 GPUs, 100+ Scientists.	1 Human Brain, 1 Coffee.

Break Privacy of the Face Recognition System

Figure: Recovered (Left), Original (Right)

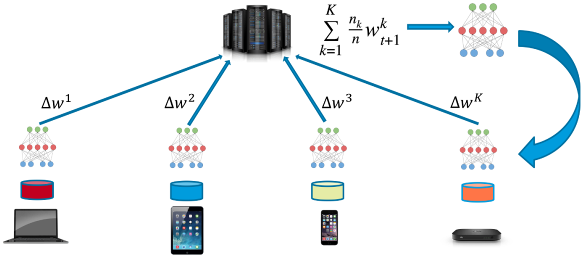
Membership Attack: determine if a record is in the training set.

Model Inversion Attack: recover the photo of a person given his name in face recognition task.

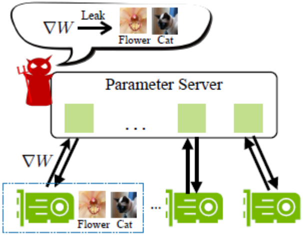
Other Data Abuse: Netflix Recommendation Competition, Privacy of the Genome Data, ...

Fredrikson et al., Proc. CCS, 2016
R. Shokri et al., Proc. SSP, 2017

Federated Learning is Not Private



Federated Learning



Gradient Leakage

Federated Learning: train a centralized model, w, while training data is distributed over many clients. In each communication-round, clients update their local models with their own private data. The center server then aggregates these local models, and sends the updated model to clients.

Gradient Leakage: update of the local model encodes private data. Gradient is not an encryption of private data.

L. Zhu, Z. Liu, and S. Han, NeurIPS, 2019.

Our Efforts Towards the Trustworthy Deep Learning

1. Robust deep learning
   Adversarial defense & Verification
2. Accurate deep learning
   Optimization & Neural Architecture Design
3. Efficient deep learning
   Acceleration & Compression
4. Private deep learning
   Federated Learning & Differential Privacy

...

with theoretical guarantees!

Simple and principled approaches converge with working machine learning algorithms!

A few examples:

Accelerate Deep Learning I

Adversarial Robust Deep Learning II.1

Deep Nets Compression II.2

Privacy-Preserving Machine Learning III.1 & III.2

I. Scheduled Restart Momentum for Accelerated Stochastic Gradient Descent

B. Wang*, T. Nguyen*, T. Sun, A. Bertozzi, R. Baraniuk, and S. Osher, Scheduled Restart Momentum for Stochastic Gradient Descent, arXiv:2002.10583, 2020.
Code: https://github.com/minhtannguyen/SRSGD

Blog: http://almostconvergent.blogs.rice.edu/2020/02/21/srsgd/

Empirical Risk Minimization (ERM)

Consider training a machine learning model

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

Empirical Risk Minimization (ERM)

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

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

Classification: cross-entropy loss $\mathcal{L}(\hat{y}_i, y_i) = -\sum_{j=1}^{c} y_i^j \log(p_i^j)$. where p_i^j is the predicted probability that y_i is belong to j-th class.

Regression: mean squared error $\mathcal{L}(\hat{y}_i, y_i) = (y_i - \hat{y}_i)^2$.

Challenges: $d \sim 10^{10}$, $N \sim 10^{10}$, and $f(\mathbf{w})$ is nonconvex.

Gradient Descent

Suppose $f(\mathbf{w})$ is L-smooth, i.e., $\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{v})\|_2 \leq L\|\mathbf{w} - \mathbf{v}\|_2$.

Start from $\mathbf{w}_0$, gradient descent performs the following iteration

$$\mathbf{w_k} = \mathbf{w_{k-1}} - \mathbf{s}\nabla \mathbf{f}(\mathbf{w_{k-1}}).$$

1. $f(\mathbf{w})$ is μ-strongly convex (bounded below by a quadratic function), let $s = 2/(\mu + L)$, we have

$$\|\mathbf{w}_k - \mathbf{w}_*\|_2 \leq \left(\frac{L/\mu - 1}{L/\mu + 1}\right)^k \|\mathbf{w}_0 - \mathbf{w}_*\|_2, \quad \mathbf{w}_* \text{ is the minimum.}$$

2. $f(\mathbf{w})$ is convex, let $s = 1/L$, we have

$$f(\mathbf{w}_k) - f(\mathbf{w}_*) \leq \frac{2L\|\mathbf{w}_0 - \mathbf{w}_*\|_2^2}{k}.$$

3. $f(\mathbf{w})$ is nonconvex, let $s = 1/L$, we have

$$\|\nabla f(\mathbf{w}_k)\|_2 \leq \sqrt{\frac{2L(f(\mathbf{w}_0) - f(\mathbf{w}_*))}{k}}.$$

A. Cauchy, 1847

Gradient Descent

Consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T \mathbf{L}\mathbf{w} - \mathbf{w}^T \mathbf{e}_1,$$

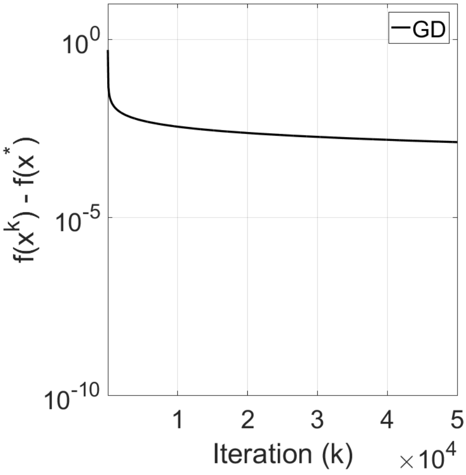
where

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

and $\mathbf{e}_1$ is a 1000-dim vector whose first entry is 1 and all the other entries are 0.

A. Nemirovski et al, 1985

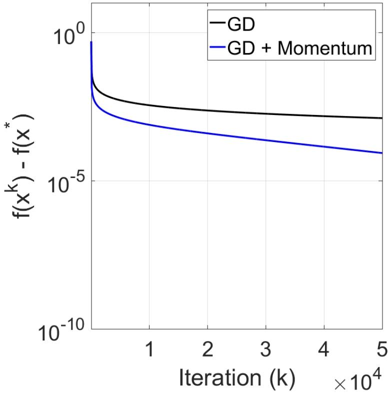
Gradient Descent



$O(1/k)$ convergence rate! Very slow!

Gradient Descent + (Lookahead/Nesterov) 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

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

B. Polyak, 1964

Why momentum works

High dimensional problem is usually ill-conditioned!

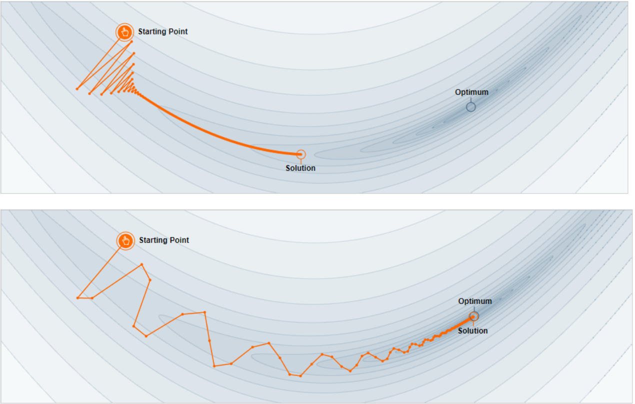
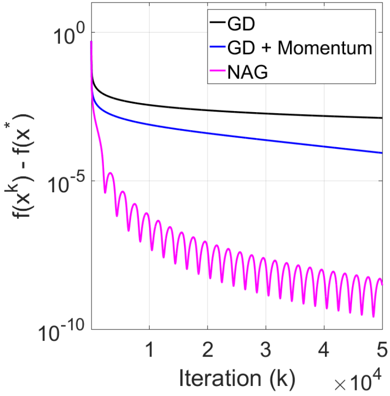


Figure: Top: no momentum; Bottom: with momentum.

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

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!

Nesterov Accelerated Gradient (NAG)

One of the most beautiful and mysterious results in optimization!

Not a descent method! (ripples/bumps in the traces of cost values)

Continuous dynamics

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

which satisfies $f(X(t)) - f(X^*) \leq O\left(\frac{1}{t^2}\right)$.

We can prove the above result by considering the following Lyapunov function

$$\mathcal{E}(t) := t^2(f(X(t)) - f(X^*)) + 2\|X(t) + \frac{t}{2}\dot{X}(t) - X^*\|_2^2.$$

Can we further accelerate NAG? NAG is not monotonically converge!

Y. Nesterov, 1983.

Su, Boyd, and Candes, 2014.

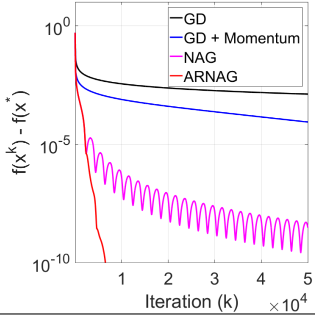
Adaptive Restart NAG (ARNAG)

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

$$\mathbf{w}_k = \mathbf{v}_k + \frac{m(k-1)-1}{m(k-1)+2}(\mathbf{v}_k - \mathbf{v}_{k-1}),$$

where

$$m(k) = \begin{cases} m(k-1)+1, & \text{if } f(\mathbf{w}_k) \le f(\mathbf{w}_{k-1}), \\ 1, & \text{otherwise.} \end{cases}$$



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

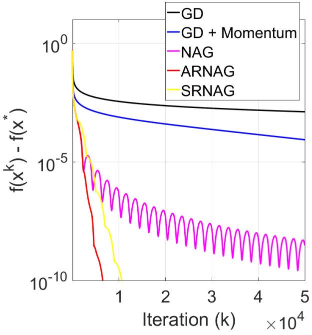
Sharpness: $\frac{\mu}{r} d(\mathbf{w}, \mathbf{w}_*)^r \le f(\mathbf{w}) - f(\mathbf{w}_*), \quad \mu > 0, r > 1.$

V. Roulet et al. NIPS 2017

Scheduled Restart NAG (SRNAG)

Let $(0, T] = \bigcup_{i=1}^m I_i = \bigcup_{i=1}^m (T_{i-1}, T_i]$. In each I_i, we restart the momentum after F_i iterations as follows:

$$\mathbf{v}_k = \mathbf{w}_{k-1} - s\nabla f(\mathbf{w}_{k-1}),$$
$$\mathbf{w}_k = \mathbf{v}_k + \frac{(k \bmod F_i)}{(k \bmod F_i) + 3}(\mathbf{v}_k - \mathbf{v}_{k-1}).$$



$O(e^{-\beta k})$ convergence with 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:

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

Can NAG still accelerate convergence with Stochastic Gradient?

A Motivating Example – Gaussian Noise Corrupted Gradient – Case I

Consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{L}\mathbf{w} - \mathbf{w}^T\mathbf{e}_1,$$

where

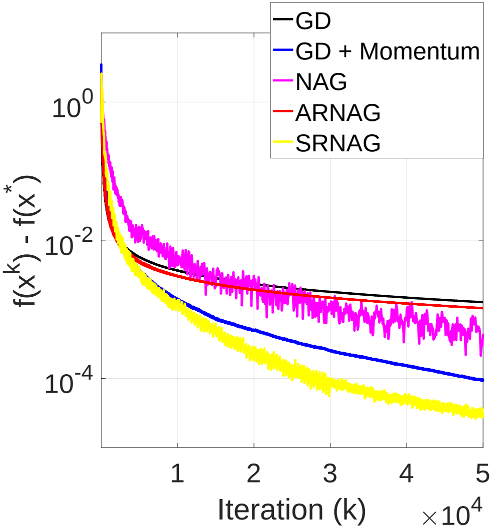
$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ -1 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}_{1000\times 1000},$$

and $\mathbf{e}_1$ is a 1000-dim vector whose first entry is 1 and all the other entries are 0.

Gaussian Noise Corrupted Gradient:

$$\nabla f(\mathbf{w}) = \mathbf{L}\mathbf{w} - \mathbf{e}_1 + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(0, (\frac{0.1}{\lfloor k/100 \rfloor + 1})^2).$$

A Motivating Example – Gaussian Noise Corrupted Gradient – Case I



A Motivating Example – Gaussian Noise Corrupted Gradient – Case II

Consider

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{L}\mathbf{w} - \mathbf{w}^T\mathbf{e}_1,$$

where

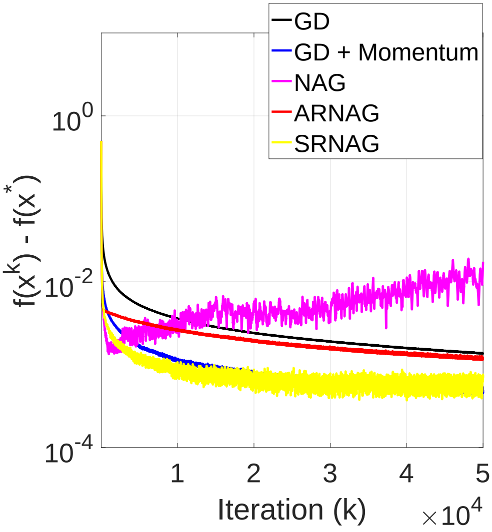
$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ -1 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}_{1000\times 1000},$$

and $\mathbf{e}_1$ is a 1000-dim vector whose first entry is 1 and all the other entries are 0.

Gaussian Noise Corrupted Gradient:

$$\nabla f(\mathbf{w}) = \mathbf{L}\mathbf{w} - \mathbf{e}_1 + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(0, 0.001^2).$$

A Motivating Example – Gaussian Noise Corrupted Gradient – Case II



A Motivating Example – Logistic Regression – Case III

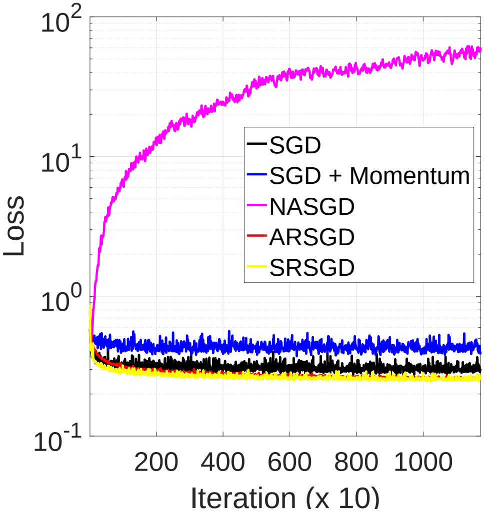


Figure: Training loss of logistic regression for MNIST classification.

Error Accumulation of NAG with Stochastic Gradient

Theorem Let $f(\mathbf{w})$ be a convex and L-smooth function. The sequence $\{\mathbf{w}^k\}_{k\geq 0}$ generated by NAG with mini-batch stochastic gradient using any constant step size $s \leq 1/L$, satisfies

$$\mathbb{E}\left(f(\mathbf{w}^k) - f(\mathbf{w}^*)\right) = O(k),$$

where $\mathbf{w}^*$ is the minimum of f, and the expectation is taken over the random mini-batch samples.

B. Wang*, T. Nguyen*, T. Sun, A. Bertozzi, R. Baraniuk, and S. Osher, 2020.

NAG with Restart (Inexact Oracle)

Adaptive Restart NAG with Inexact Oracle: restart too often, degenerates to GD without momentum.

Scheduled Restart NAG with Inexact Oracle: appropriate restart scheduling can lead to an optimal trade-off between convergence and error accumulation.

Scheduled Restart SGD (SRSGD)

$$
\begin{aligned}
\mathbf{v}^k &= \mathbf{w}^{k-1} - s\frac{1}{m}\sum_{j=1}^{m}\nabla f_{i_j}(\mathbf{w}^{k-1}),\\
\mathbf{w}^k &= \mathbf{v}^k + \frac{(k \bmod F_i)}{(k \bmod F_i)+3}(\mathbf{v}^k - \mathbf{v}^{k-1}).
\end{aligned}
$$

where m is the batch size.

B. Wang*, T. Nguyen*, T. Sun, A. Bertozzi, R. Baraniuk, and S. Osher, 2020.

Convergence of SRSGD

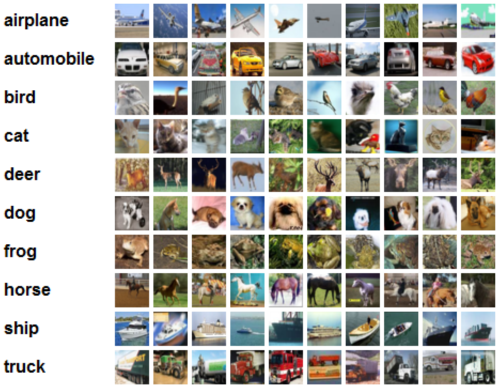
Theorem Suppose $f(\mathbf{w})$ is L-smooth. Consider the sequence $\{\mathbf{w}^k\}_{k\geq 0}$ generated by SRSGD with mini-batch stochastic gradient and any restart frequency F using any constant step size $s \leq 1/L$. Assume that the set $\mathcal{A} := \{k \in \mathbb{Z}^+ | \mathbb{E}f(\mathbf{w}^{k+1}) \geq \mathbb{E}f(\mathbf{w}^k)\}$ is finite, then we have

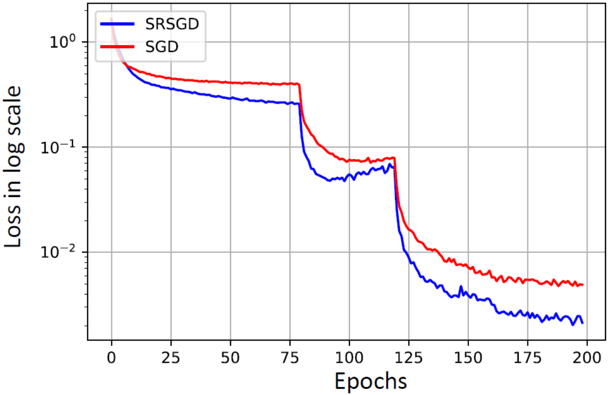
$$\min_{1\leq k\leq K} \{\mathbb{E}\|\nabla f(\mathbf{w}^k)\|_2^2\} = O(s + \frac{1}{sK}).$$

Therefore for $\forall\epsilon > 0$, to get ϵ error, we just need to set $s = O(\epsilon)$ and $K = O(1/\epsilon^2)$.

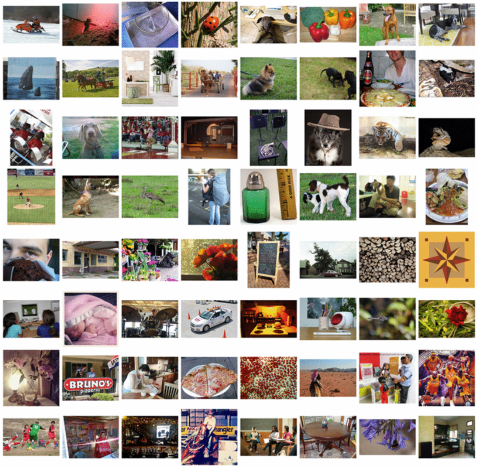
B. Wang*, T. Nguyen*, T. Sun, A. Bertozzi, R. Baraniuk, and S. Osher, 2020.

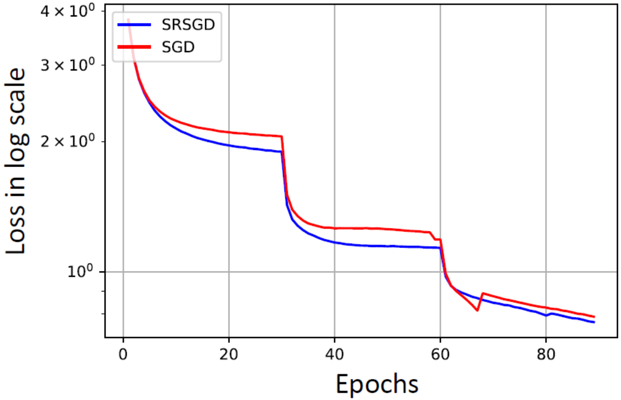
SRSGD for Deep Learning – CIFAR10/CIFAR100 Classification





SRSGD for Deep Learning – ImageNet Classification





Improving Testing Accuracy

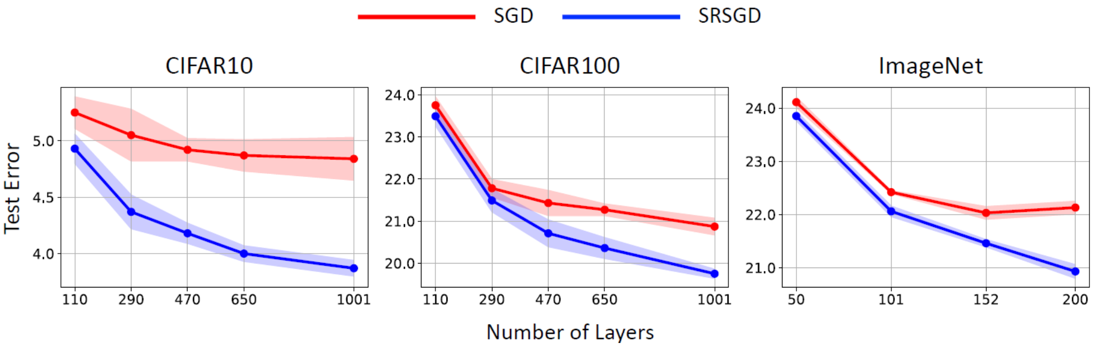
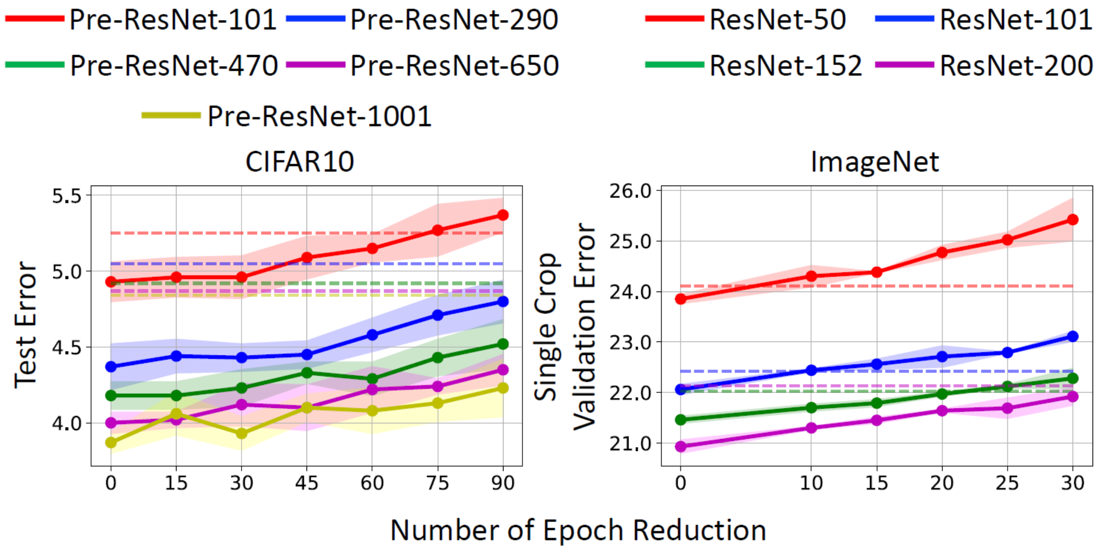


Figure: Error vs. depth of ResNet.

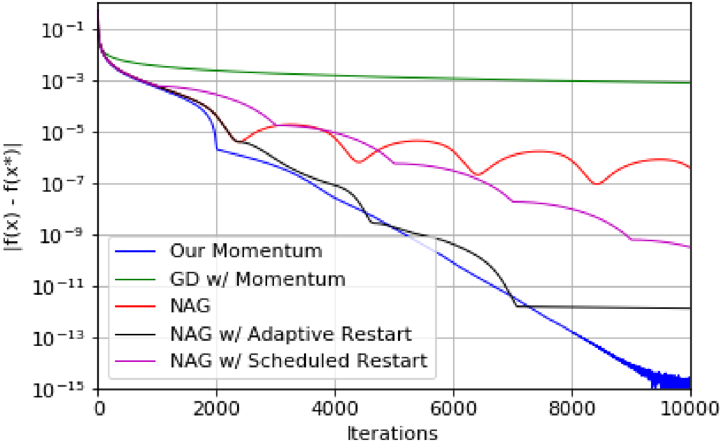
Reduce the Training Epochs



Is NAG-style Momentum Optimal?

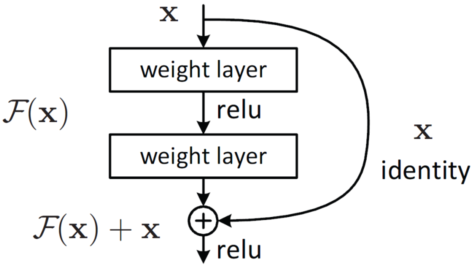
Theoretically, yes! Due to Nemirovski & Nesterov!

Empirically, not!



II. Transport Equation vs. Residual Learning

ResNet vs. Transport Equation



Plain Net: $\mathbf{x}_{l+1} = \mathcal{G}(\mathbf{x}_l)$
ResNet: $\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l)$

Forward propagation (FP) of ResNet for any data-label pair $(\hat{\mathbf{x}}, y)$

$$\begin{cases} \mathbf{x}(0) = \hat{\mathbf{x}}, \\ \mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \Delta t \cdot \overline{F}(\mathbf{x}(t_k), \mathbf{w}(t_k)), k = 1, 2, \cdots, L-1 \quad \text{with} \quad \overline{F} \doteq \frac{1}{\Delta t}\mathcal{F} \\ \hat{y} \doteq f(\mathbf{x}(1)) = \mathrm{softmax}(\mathbf{w}_{\mathrm{FC}} \cdot \mathbf{x}). \end{cases}$$

Continuum limit: $\frac{d\mathbf{x}(t)}{dt} = \overline{F}(\mathbf{x}(t), \mathbf{w}(t))$.

Transport equation (TE): $\frac{\partial u}{\partial t}(\mathbf{x}, t) + \overline{F}(\mathbf{x}, \mathbf{w}(t)) \cdot \nabla u(\mathbf{x}, t) = 0, \quad \mathbf{x} \in \mathbb{R}^d$.

He et al., CVPR, 2016.

Many related works ...

Forward and backward propagation

1. Let $u(\mathbf{x},1)=f(\mathbf{x})$, note $u(\hat{\mathbf{x}},0)=u(\mathbf{x}(1),1)=f(\mathbf{x}(1))$. Therefore, we model FP as computing $u(\hat{\mathbf{x}},0)$ along the characteristics of the following TE

$$\begin{cases} \frac{\partial u}{\partial t}(\mathbf{x},t)+\overline{F}(\mathbf{x},\mathbf{w}(t))\cdot\nabla u(\mathbf{x},t)=0, \quad \mathbf{x}\in\mathbb{R}^d, \\ u(\mathbf{x},1)=f(\mathbf{x}). \end{cases}$$

2. Backpropagation (BP): find $\mathbf{w}(t)$ for the following control problem

$$\begin{cases} \frac{\partial u}{\partial t}(\mathbf{x},t)+\overline{F}(\mathbf{x},\mathbf{w}(t))\cdot\nabla u(\mathbf{x},t)=0, \quad \mathbf{x}\in\mathbb{R}^d, \\ u(\mathbf{x},1)=f(\mathbf{x}), \\ u(\mathbf{x}_i,0)=y_i, \quad \mathbf{x}_i\in T, \text{ with } T \text{ being the training data.} \end{cases}$$

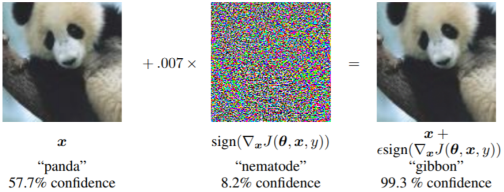
$\mathbf{x}(1)$ is the transport of $\hat{\mathbf{x}}$ along the characteristics.

II.1 Feynman-Kac Formalism Principled Adversarial Defense

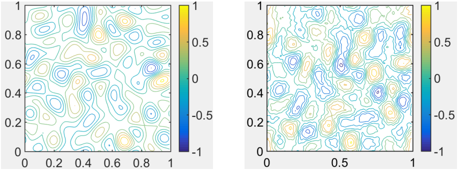
B. Wang, B. Yuan, Z. Shi, and S. Osher, ResNets Ensemble via the Feynman-Kac Formalism to Improve Natural and Robust Accuracies, NeurIPS, 2019.

Code: `https://github.com/BaoWangMath/EnResNet`

Why Adversarial Example Arise? – A PDE Interpretation



In the TE model, $u(\mathbf{x}, 0)$ serves as the decision function for classification.



(a) $u(\mathbf{x}, 1)$ (b) $u(\mathbf{x}, 0)$

The decision boundary is highly erratic, exposed to adversarial attacks!

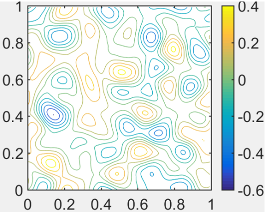
Goodfellow et al., ICLR, 2015.

Given input data distribution $\{x\}$, (a): softmax landscape; (b): deep learning classifier's landscape.

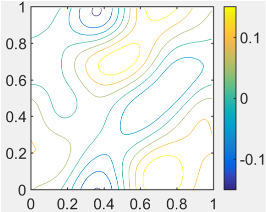
Improving Robustness via Diffusion

We use diffusion to regularize the decision function $u(x,0)$, which resulting in

$$\begin{cases} \frac{\partial u}{\partial t} + \overline{F}(\mathbf{x}, \mathbf{w}(t)) \cdot \nabla u + \frac{1}{2}\sigma^2 \Delta u = 0, \quad \mathbf{x} \in \mathbb{R}^d, \; t \in [0,1), \\ u(\mathbf{x}, 1) = f(\mathbf{x}). \end{cases}$$



(a) $u(\mathbf{x},0), \sigma = 0.01$



(b) $u(\mathbf{x},0), \sigma = 0.1$

Theorem (Stability) Let $\overline{F}(\mathbf{x}, t)$ be Lipschitz in both $\mathbf{x}$ and t, and $f(\mathbf{x})$ is bounded. For the above terminal value problem of convection-diffusion equation, $\sigma \neq 0$, we have

$$|u(\mathbf{x}+\delta, 0) - u(\mathbf{x}, 0)| \leq C \left(\frac{\|\delta\|_2}{\sigma} \right)^{\alpha}$$

for some constant $\alpha > 0$ if $\sigma \leq 1$. $C := C(d, \|f\|_\infty, \|\overline{F}\|_{L^\infty_{x,t}})$ is a constant.

O. Ladyzhenskaja and et al., Linear and Quasilinear Equations of Parabolic Type

Feynman-Kac Formula and Deep Nets Design

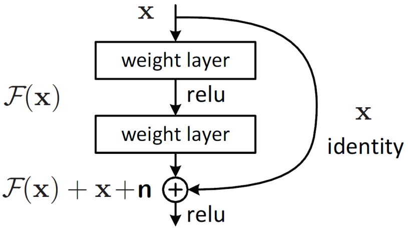
By Feynman-Kac formula, we have

$$u(\hat{\mathbf{x}}, 0) = \mathbb{E}\left[f(\mathbf{x}(1)) | \mathbf{x}(0) = \hat{\mathbf{x}}\right],$$

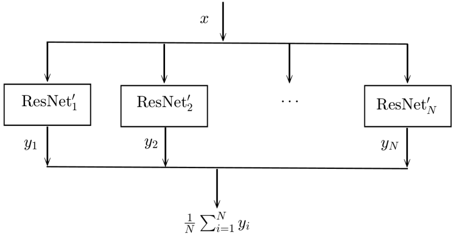
where $\mathbf{x}(t)$ is an Itô process,

$$d\mathbf{x}(t) = \overline{F}(\mathbf{x}(t), \mathbf{w}(t))dt + \sigma dB_t.$$

Deep Nets Design!



Residual mapping + Gaussian noise



Average multiple jointly trained ResNets

Empirical Adversarial Risk Minimization (Robust Training)

Adversarial training: $\min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\max_{\delta\in S} \mathcal{L}(f(\mathbf{w},\mathbf{x}+\delta),y)\right]$

Adversarial attacks:

FGSM

$$\mathbf{x}' = \mathbf{x} + \epsilon\,\mathrm{sign}\left(\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},y)\right).$$

IFGSM

$$\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \alpha\cdot\mathrm{sign}\left(\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^{(m-1)},y)\right), \quad m = 1,2,\cdots,M.$$

C&W

$$\min_{\delta} ||\delta||_2^2, \quad \text{subject to} \quad f(\mathbf{x}+\delta,\mathbf{w}) = t, \ \mathbf{x}+\delta\in[0,1]^n.$$

N. Carlini and D. Wagner, arXiv:1608.04644

C. Szegedy and et al., arXiv:1312.6199

Performance on CIFAR10 Classification

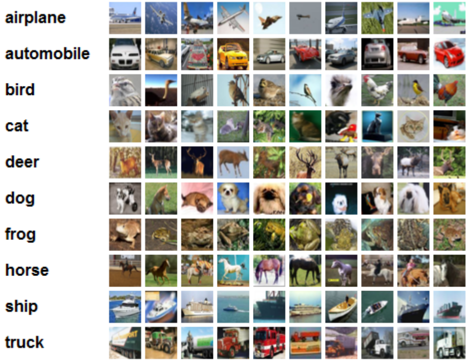


Table: Natural and robust acc of EnResNets on the CIFAR10. Unit: %.

Model	$\mathcal{A}_{\mathrm{nat}}$	$\mathcal{A}_{\mathrm{rob}}$ (FGSM)	$\mathcal{A}_{\mathrm{rob}}$ (IFGSM^{20})	$\mathcal{A}_{\mathrm{rob}}$ (C&W)
ResNet20	75.11	50.89	46.03	58.73
En$_1$ResNet20	77.21	55.35	49.06	65.69
En$_5$ResNet20	**82.52**	**58.92**	**51.48**	**67.73**

II.2 Deep Neural Nets Compression

Channel-Pruning for Adversarial Robust Deep Nets

T. Dinh*, B. Wang*, A. Bertozzi, S. Osher and J. Xin, Sparsity Meets Robustness: Channel pruning for the Feynman-Kac Formalism Principled Robust Neural Nets, Preprint, 2019.

Deep Nets Compression

Common approaches to improve inference efficiency of deep learning:

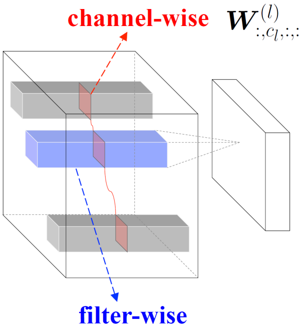
- **Sparse weights**
- **Quantized weights**

We focus on sparsifying deep nets (structured & unstructured)!

Neural architecture redesign!

$+$

Structured & Unstructured weights pruning!

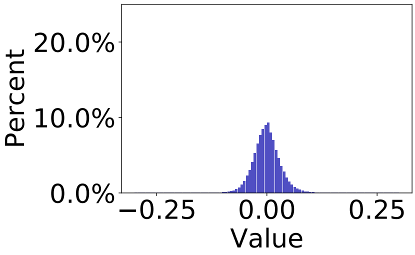


$$n_f \times n_c \times n_w \times n_h$$

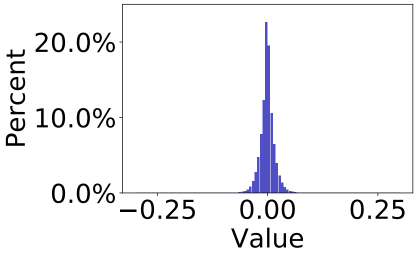
Remark. Structured sparsity can remarkably speed up inference.

Neural Architecture Redesign

Sparsity meets Robustness: ResNet20 vs. En_5ResNet20



(a) ResNet20 (AT) (3.64% ($\leq 10^{-3}$))



(b) En_5ResNet20 (AT) (11.57% ($\leq 10^{-3}$))

Table: Natural and robust acc of EnResNet on the CIFAR10. Unit: %.

Model	$\mathcal{A}_{\rm nat}$	$\mathcal{A}_{\rm rob}$ (FGSM)	$\mathcal{A}_{\rm rob}$ (IFGSM^{20})	$\mathcal{A}_{\rm rob}$ (C&W)
ResNet20	75.11	50.89	46.03	58.73
En_5ResNet20	**82.52**	**58.92**	**51.48**	**67.73**

Maximize Sparsity: Structured & Unstructured Sparsity

Adversarial training:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := \min_{\mathbf{w}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\max_{\delta\in S} \mathcal{L}(f(\mathbf{w}, \mathbf{x}+\delta), y) \right]$$

Augmented Lagrangian:

$$\mathcal{L}_\beta(\mathbf{w}, \mathbf{u}, \mathbf{z}) = \mathcal{L}(\mathbf{w}) + \lambda \|\mathbf{u}\|_1 + \langle \mathbf{z}, \mathbf{w} - \mathbf{u} \rangle + \frac{\beta}{2} \|\mathbf{w} - \mathbf{u}\|^2, \quad \lambda, \beta \geq 0$$

Unstructured sparsity: ℓ_1-penalty; Structured sparsity: group ℓ_1-penalty.

ADMM:

$$\begin{cases} \mathbf{w}^{t+1} \leftarrow \arg\min_{\mathbf{w}} \mathcal{L}_\beta(\mathbf{w}, \mathbf{u}^t, \mathbf{z}^t) \\ \mathbf{u}^{t+1} \leftarrow \arg\min_{\mathbf{u}} \mathcal{L}_\beta(\mathbf{w}^{t+1}, \mathbf{u}, \mathbf{z}^t) \\ \mathbf{z}^{t+1} \leftarrow \mathbf{z}^t + \beta(\mathbf{w}^{t+1} - \mathbf{u}^{t+1}) \end{cases}$$

Remark 1. One can improve the sparsity of the final learned weights by replacing $\|\mathbf{u}\|_1$ with $\|\mathbf{u}\|_0$; but $\|\cdot\|_0$ is not differentiable.

Remark 2. The Lagrange multiplier term, $\langle \mathbf{z}, \mathbf{w} - \mathbf{u} \rangle$, seeks to close the gap between $\mathbf{w}^t$ and $\mathbf{u}^t$, and this in turn reduces sparsity of $\mathbf{w}^t$.

Group the weights into: $\{\mathbf{w_1}, \mathbf{w_2}, \cdots, \mathbf{w}_G\}$, group Lasso: $\sum_{g=1}^{G} \|\mathbf{w}_g\|_2$; group ℓ_0: $\sum_{g=1}^{G} 1_{\|\mathbf{w}_g\|_2 \neq 0}$.

Relaxed Augmented Lagrangian:

$$\mathcal{L}_\beta(\mathbf{w},\mathbf{u}) = \mathcal{L}(\mathbf{w}) + \lambda\|\mathbf{u}\|_0 + \frac{\beta}{2}\|\mathbf{w}-\mathbf{u}\|^2.$$

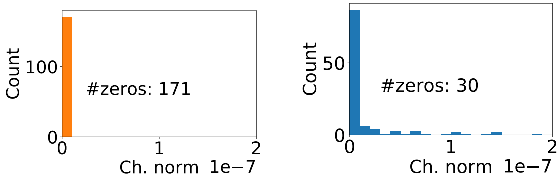
Remark 1. For a fixed $\mathbf{w}^t$, we have

$$\mathbf{u}^t = H_{\sqrt{2\lambda/\beta}}(\mathbf{w}^t) = (\mathbf{w}_1^t \chi_{\{|\mathbf{w}_1|>\sqrt{2\lambda/\beta}\}}, \dots, \mathbf{w}_d^t \chi_{\{|\mathbf{w}_d|>\sqrt{2\lambda/\beta}\}}),$$

where $H_\alpha(\cdot)$ is the hard-thresholding operator with parameter α.

Remark 2. Fixed $\mathbf{u}^t$, $\mathbf{w}^t$ can be updated by gradient descent.

Remark 3. $\mathbf{w}$ here is sparser than that in the augmented Lagrangian.



(a) Relaxed (Zoom in) (b) Original (Zoom in)

Figure: Channel norms of the adversarially trained ResNet20.

Convergence of the Relaxed Augmented Lagrangian

Theorem. Assume $\mathcal{L}_\beta$ is L-smooth in $\mathbf{w}$, then the relaxed augmented Lagrangian $\mathcal{L}_\beta(\mathbf{w}^t, \mathbf{u}^t)$ decreases monotonically and converges sub-sequentially to a limit point $(\bar{\mathbf{w}}, \bar{\mathbf{u}})$ provided the stepsize η such that $\eta < 2/(\beta + L)$

Sparsity vs. Accuracy & Robustness

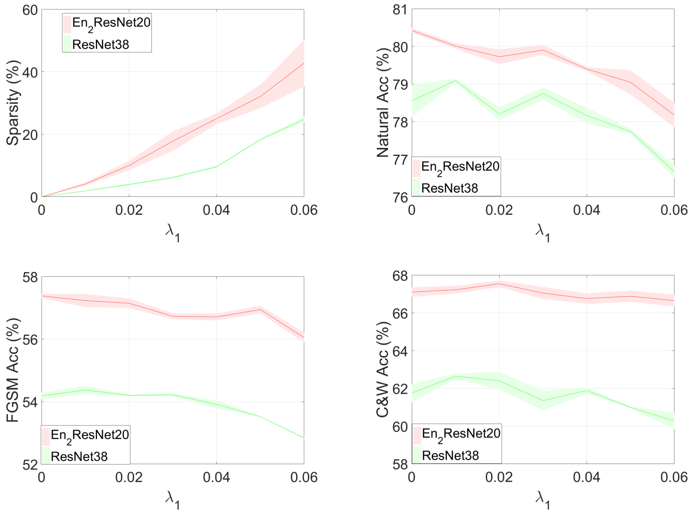


Figure: En_2ResNet20 vs. ResNet38 under different λ_1. (5 runs, $\beta = 1$).

III. Privacy-Preserving Machine Learning with Laplacian Smoothing

III.1 Privacy-Preserving Empirical Risk Minimization (ERM)

B. Wang, Q. Gu, M. Boedihardjo, F. Barekat, and S. Osher. DP-LSSGD: A Stochastic Optimization Method to Lift the Utility in Privacy-Preserving ERM, ArXiv:1906.12056, 2019

Code: `https://github.com/BaoWangMath/DP-LSSGD`

Differential Privacy

Figure: Recovered (Left), Original (Right)

Differential privacy (DP) is a successful countermeasure to adversaries that try to break the privacy of machine learning.

Add differential privacy constraint in training machine learning models!

F. McSherry and I. Mironov, Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders, KDD, 2009.

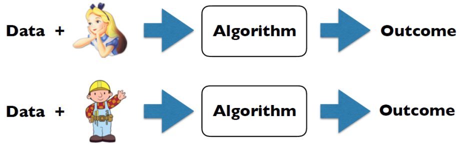
M. Fredrikson, S. Jha, T. Ristenpart, Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, CCS, 2015.

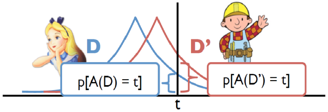
Differential Privacy

Definition. A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for any two neighboring datasets D, D' that differ in only one entry and for all events S in the output space of $\mathcal{A}$, we have

$$Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon} Pr(\mathcal{A}(D') \in S) + \delta.$$

DP promises to protect individuals from any additional harm that they might face due to their data being in the private database **x** that they would not have faced had their data not been part of **x**.





For all D, D' that differ in one person, if $\mathcal{A}$ is (ϵ, δ)-DP, then:

$$\Pr\left[\left| \ln \left(\frac{\Pr[\mathcal{A}(D) \in S]}{\Pr[\mathcal{A}(D') \in S]} \right) \right| \geq \epsilon \right] \leq \delta$$

C. Dwork and A. Roth, The Algorithmic Foundation of Differential Privacy, 2014.

Figures courtesy of K. Chaudhuri

Privacy-Preserving Empirical Risk Minimization

Empirical risk minimization (ERM):

$$\min F(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\mathbf{w}, \mathbf{x}_i, y_i).$$

Differentially private SGD (DP-SGD)

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta_k \left(\frac{1}{m}\sum_{k=1}^{m} \nabla f_{i_k}(\mathbf{w}^k) + \mathbf{n} \right),\ \mathbf{n} \sim \mathcal{N}(0, \nu^2 I_{d\times d}), \{i_k\}_{k=1}^{m} \subset [n]$$

How to quantify n to guarantee (ϵ, δ)-DP?

Major difficulty: quantifying privacy loss aggregation during SGD.

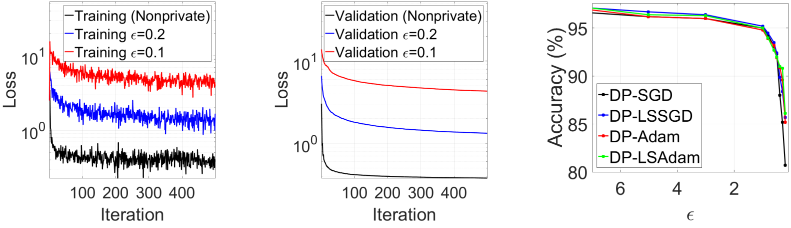
K. Chaudhuri, C. Monteleoni, and A. Sarwate, Differentially Private ERM, JMLR, 2011.

M. Abadi, and et al,, Deep Learning with Differential Privacy, arXiv:1607.00133, 2016.

Theorem (Privacy Budget) Suppose that each f_i is L-Lipschitz. Given the number of iterations T, for any $(\epsilon, \delta > 0)$, DP-SGD, with injected Gaussian noise $\mathcal{N}(0, \nu^2 I)$, satisfies (ϵ, δ)-DP with $\nu^2 = 20T\alpha G^2/(\mu n^2 \epsilon)$, where $\alpha = \log(1/\delta)/\big((1-\mu)\epsilon\big) + 1$, if $\exists \mu \in (0,1)$ s.t. $\alpha \leq \log\big(\mu n^3 \epsilon/(5b^3 T\alpha + \mu b n^2 \epsilon)\big)$ and $5b^2 T\alpha/(\mu n^2 \epsilon) \geq 1.5$.

B. Wang, et al., arXiv:1906.12056, 2019.

SGD vs. DP-SGD



Training Loss — Validation Loss — Validation Acc

Figure: Logistic regression on the MNIST trained by DP-SGD with $(\epsilon, 10^{-5})$ -DP guarantee (left & middle). LeNet on the MNIST trained by DP-SGD with $(\epsilon, 10^{-5})$-DP guarantee (right).

DP-SGD reduces the utility of the trained model severely.

Question: Can we do better than DP-SGD with negligible extra computation and memory costs?

DP-SGD with Laplacian Smoothing (DP-LSSGD)

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta_k A_\sigma^{-1}\left(\frac{1}{m}\sum_{k=1}^{m}\nabla f_{i_k}(\mathbf{w}^k) + \mathbf{n}\right).$$

where

$$A_\sigma = (I - \sigma L) = \begin{bmatrix} 1+2\sigma & -\sigma & 0 & \dots & 0 & -\sigma \\ -\sigma & 1+2\sigma & -\sigma & \dots & 0 & 0 \\ 0 & -\sigma & 1+2\sigma & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -\sigma & 0 & 0 & \dots & -\sigma & 1+2\sigma \end{bmatrix}_{d\times d}$$

A_σ^{-1} p.s.d with condition number $1+4\sigma$; FFT Implementation

DP-LSSGD has the same privacy budget as DP-SGD!

Proposition (Post-processing) Let $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ be a randomized algorithm that is (ϵ, δ)-DP. Let $f : R \to R'$ be an arbitrary mapping. Then $f \circ \mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R'$ is (ϵ, δ)-DP.

S. Osher, B. Wang, P. Yin, X. Luo, F. Barekat, M. Pham, and A. Lin, arXiv:1806.06317, 2018

Code: `https://github.com/BaoWangMath/LaplacianSmoothing-GradientDescent`

For any pair of $\|\mathbf{x}-\mathbf{y}\|_1 \leq 1$, and any $S \subset R'$, let $T = \{r \in R : f(r) \in S\}$, we have

$$Pr[f(\mathcal{M}(\mathbf{x})) \in S] = Pr[\mathcal{M}(\mathbf{x}) \in T] \leq \exp(\epsilon)Pr[M(\mathbf{y}) \in T] + \delta = \exp(\epsilon)Pr[f(\mathcal{M}(\mathbf{x})) \in S] + \delta$$

Laplacian Smoothing as a Denoiser

Consider the following diffusion equation with the Neumann BC

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad (x,t) \in [0,1] \times [0,+\infty), \\ \frac{\partial u(0,t)}{\partial x} = \frac{\partial u(1,t)}{\partial x} = 0, \ \ t \in [0,+\infty) \\ u(x,0) = f(x), \quad x \in [0,1] \end{cases}$$

Backward Euler in time and central finite difference in space with v^0 being a discretization of $f(x)$.
Unconditionally stable!

$$\mathbf{v}^{\Delta t} - \mathbf{v}^0 = \Delta t \mathbf{L} \mathbf{v}^{\Delta t} \Rightarrow \mathbf{v}^{\Delta t} = (I - \Delta t \mathbf{L})^{-1} \mathbf{v}^0 \quad (\sigma = \Delta t)$$

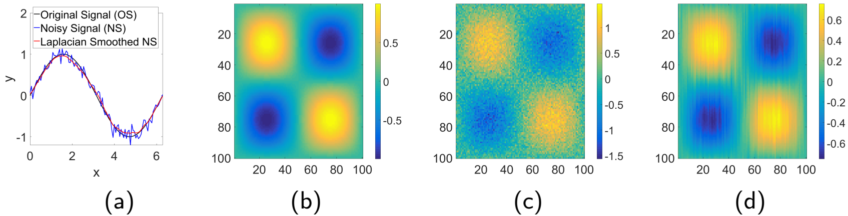


Figure: Illustration of LS ($\sigma = 10$ for $\mathbf{v}_1$ and $\sigma = 100$ for $\mathbf{v}_2$). (a): 1D signal sampled uniformly from $\sin(x)$ for $x \in [0, 2\pi]$. (b), (c), (d): 2D original, noisy, and denoised signals sampled from $\sin(x)\sin(y)$ for $(x, y) \in [0, 2\pi] \times [0, 2\pi]$.

DP-LSSGD Improves Utility of Logistic Regression over DP-SGD (MNIST)

$$\min_{\mathbf{w}} F(\mathbf{w}) = \min_{\mathbf{w}} \left\{ \frac{1}{n} \sum_{i=1}^{n} -\log \left(\frac{\exp < \mathbf{w}, \mathbf{x}_i >_{y_i}}{\sum_j \exp < \mathbf{w}, \mathbf{x}_i >_j} \right) + \lambda \|\mathbf{w}\|_2 \right\}, \quad \lambda = 1 \times 10^{-4}.$$

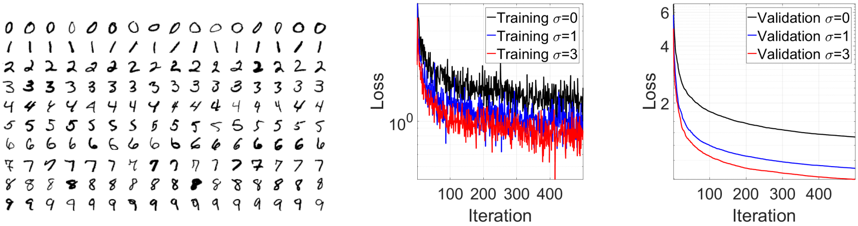


Figure: $(0.2, 10^{-5})$-DP guarantee. Step size: $1/t$.

Table: Acc of logistic regression with $(\epsilon, \delta = 10^{-5})$-DP guarantee.

ϵ	0.25	0.20	0.15	0.10
$\sigma = 0$	81.45 ± 1.59	78.92 ± 1.14	77.03 ± 0.69	73.49 ± 1.60
$\sigma = 1$	83.27 ± 0.35	81.56 ± 0.79	79.46 ± 1.33	76.29 ± 0.53
$\sigma = 2$	$\mathbf{83.65 \pm 0.76}$	$\mathbf{82.15 \pm 0.59}$	$\mathbf{80.77 \pm 1.26}$	$\mathbf{76.31 \pm 0.93}$

Utility Guarantees

Algorithm	Privacy	Assumption	Utility	Measurement
DP-SGD	(ϵ,δ)	convex	$\tilde{\mathcal{O}}\left(\sqrt{(D_0+G^2)d}/(\epsilon n)\right)$	optimality gap
DP-SGD	(ϵ,δ)	nonconvex	$\tilde{\mathcal{O}}\left(\sqrt{d}/(\epsilon n)\right)$	ℓ_2-norm of gradient
DP-LSSGD	(ϵ,δ)	convex	$\tilde{\mathcal{O}}\left(\sqrt{\gamma(D_\sigma+G^2)d}/(\epsilon n)\right)$	optimality gap
DP-LSSGD	(ϵ,δ)	nonconvex	$\tilde{\mathcal{O}}\left(\sqrt{\beta d}/(\epsilon n)\right)$ [1]	ℓ_2-norm of gradient

[1] Measured in the norm induced by $\mathbf{A}_\sigma^{-1}$.

$D_\sigma = \|\mathbf{w}^0 - \mathbf{w}^*\|_{\mathbf{A}_\sigma}^2$ and $\mathbf{w}^*$ is the global minimizer.

$$\gamma = \frac{1}{d}\sum_{i=1}^{d}\frac{1}{1+2\sigma-2\sigma\cos(\frac{2\pi}{d})} = \frac{1+\alpha^d}{(1-\alpha^d)\sqrt{4\sigma+1}}, \quad \text{with} \quad \alpha = \frac{2\sigma+1-\sqrt{4\sigma+1}}{2\sigma}.$$

$$\beta = \frac{2\alpha^{2d+1}-\xi\alpha^{2d}+2\xi d\alpha^d-2\alpha+\xi}{\sigma^2\xi^3(1-\alpha^d)^2},$$

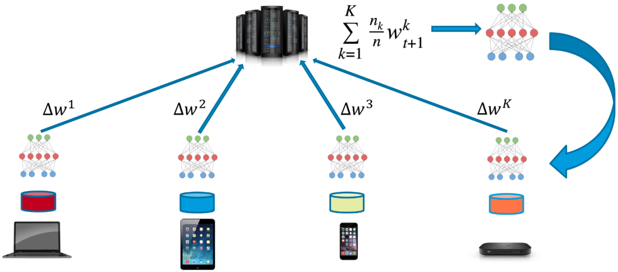
where

$$\alpha = \frac{2\sigma+1-\sqrt{4\sigma+1}}{2\sigma}, \quad \text{and} \quad \xi = -\frac{\sqrt{1+4\sigma}}{\sigma}.$$

II.2 Differentially Private Federated Learning

Z. Liang, B. Wang, Q. Gu, S. Osher, and Y. Yao. Differentially Private Federated Learning with Laplacian Smoothing, Preprint.

Federated Learning (FL)



Train a centralized model ($\mathbf{w}$) while training data is distributed over many clients. In communication-round t, the server distributes the current model $\mathbf{w}_t$ to a subset M_t of m clients. These clients update the model based on their local data. Let the updated local models be $\mathbf{w}_1^t$, $\mathbf{w}_2^t$, $\cdots$, $\mathbf{w}_{n_t}^t$, so the update is

$$H_i^t \doteq \mathbf{w}_i^t - \mathbf{w}^t, \quad \text{for} \quad i \in M_t.$$

These updates could be a single gradient computed on the client. Then the server collects these updates to update the global model

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta^t H^t, \quad H^t \doteq \frac{1}{m} \sum_{i \in M_t} H_i^t.$$

How to protect the privacy of clients' data?

Algorithm Differentially-Private Federated Learning with Laplacian Smoothing (DP-Fed-LS)

Server executes:

initialize $\mathbf{w}^0$

for each round $t = 1, 2, \cdots, T$ **do**

$m \leftarrow \max(\tau \cdot K, 1)$ where $0 < C \leq 1$

$M_t \leftarrow$ (random set of m clients)

for each client $j \in M_t$ **in parallel do**

$\mathbf{w}_j^t \leftarrow \mathbf{w}^{t-1}$

$\mathcal{B} \leftarrow$ (split local data set into batches of size B)

for each local epoch $i = 1, 2, \cdots, E$ **do**

for batch $b \in \mathcal{B}$ **do**

$\mathbf{w}_j^t \leftarrow \mathbf{w}_j^t - \eta_t \cdot \frac{1}{B} \sum_{i \in b} \nabla \ell(\mathbf{w}_j^t; b_i)$

$\mathbf{w}_j^t \leftarrow \mathbf{w}^{t-1} + \mathrm{clip}(\mathbf{w}_j^t - \mathbf{w}^{t-1})$, where $\mathrm{clip}(\mathbf{v}) \leftarrow \mathbf{v} / \max(1, \|\mathbf{v}\|_2 / G)$

$\Delta_j^t \leftarrow \mathbf{w}_j^t - \mathbf{w}^{t-1}$

$\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} + \frac{1}{m} \mathbf{A}_\sigma^{-1} \left(\sum_{j=1}^m \Delta_j^t + \mathbf{n} \right)$, where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \nu^2 \mathbf{I})$

Output $\mathbf{w}^T$

Privacy Budget for DP-Fed-LS

Theorem (Privacy Budget for DP-Fed-LS) For any $\delta \in (0, 1)$, and ϵ satisfying

$$(2\log(1/\delta) + (1+\tau)\epsilon)^2 \leq \frac{3(1-\tau)\epsilon^3}{8\tau^2 T} \quad \text{and} \quad \epsilon \leq \tau\sqrt{\frac{8T}{3}\log(\frac{1}{\delta})}$$

the DP-Fed-LS algorithm (with or without Laplacian smoothing), satisfies (ϵ, δ)-DP if its injected Gaussian noise $\mathcal{N}(0, \nu^2 I)$ is chosen to be

$$\nu \geq (4\tau G)/\epsilon$$

where G is the ℓ_2-bound of clipped gradient, $\tau := m/K$ is the subsampling ratio of active clients, T is the total number of communication rounds.

DP-Fed-LS Improves Utility of Logistic Regression over DP-Fed (MNIST)

Table: Testing accuracy of logistic regression trained by DP-Fed ($\sigma = 0$) and DP-Fed-LS ($\sigma = 1, 2, 3$) on MNIST with $(\epsilon, 1/K^{1.1})$-DP guarantee with $K = 2000$ be the number of clients.

ϵ	2	3	4	5
$\sigma = 0$	60.23 $\pm$ 2.7	73.50 $\pm$ 1.0	80.72 $\pm$ 0.49	82.24 $\pm$ 0.27
$\sigma = 1$	66.11 $\pm$ 2.7	76.98 $\pm$ 0.68	82.85 $\pm$ 0.26	84.09 $\pm$ 1.1
$\sigma = 2$	67.84 $\pm$ 2.1	79.57 $\pm$ 1.1	**82.88 $\pm$ 0.19**	**84.85 $\pm$ 0.80**
$\sigma = 3$	**68.52 $\pm$ 1.1**	**80.60 $\pm$ 0.84**	82.54 $\pm$ 0.12	84.51 $\pm$ 0.44

Batch size: 128; local epoch: 20; sensitivity: 0.15; Communication round: 15.

Thank You

I. Scheduled Restart NAG Momentum
  I.1 Accelerate convergence
  I.2 Better generalization accuracy
I. TE modeling of DNN
  I.1 Feynman-Kac formalism for robust and efficient DL
  I.2 Channel-pruning for the Feynman-Kac formula principled deep nets
II. Laplacian smoothing
  II.1 Differentially-private ERM
  II.2 Differentially-private federated learning

1. B. Wang, T. Ngyuen, T. Sun, A. Bertozzi, R. Baraniuk, and S. Osher, 2020.
2. B. Wang, X. Luo, W. Zhu, Z. Li, Z. Shi, and S. Osher, NeurIPS, 2018.
3. B. Wang, B. Yuan, Z. Shi, and S. Osher, NeurIPS, 2019.
4. T Dinh*, B. Wang*, A. Bertozzi, S. Osher, and J. Xin, Preprint, 2019.
5. B. Wang, Q. Gu, M. Boedihardjo, and S. Osher, arXiv:1906.12056, 2019.
6. Z. Liang, B. Wang, Q. Gu, S. Osher. and Y. Yao, Preprint, 2019.

Website: `https://www.math.ucla.edu/~wangbao/`

Code: `https://github.com/BaoWangMath`