PDE Approaches for Deep Learning

Stan Osher
Department of Mathematics, UCLA

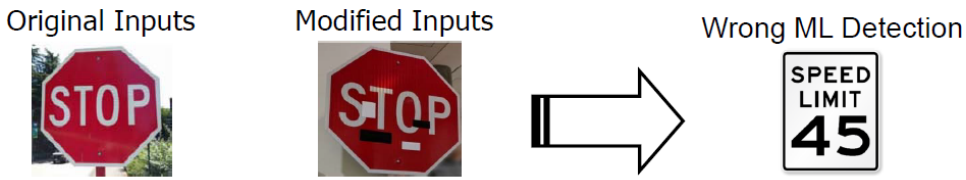
Deep Learning Advances
Science, Technology, Engineering, and Mathematics

While Deep Learning is Not Perfect!

Adversarial Vulnerability of Deep Neural Nets

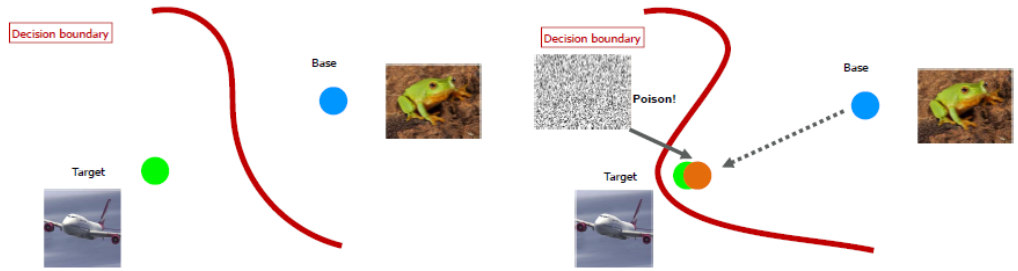
Deep neural nets are vulnerable to the following attacks:

- Physical Attack (Attack during data acquisition)

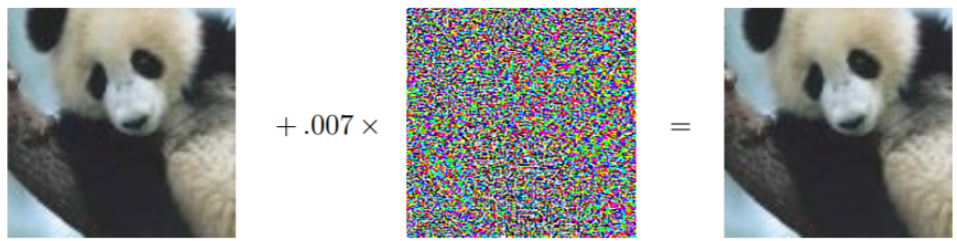


- Poisoning Attack (Attack during training)

  Add poisoning data will change the decision boundary!



- Inference Attack (Attack during testing)



$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Break Privacy of the Face Recognition System

Figure: An image recovered using model inversion attack (left) and a training set image of the victim (right).

We can recover the sensitive training data by using the model-inversion attack on the released deep learning based systems!

Fredrikson et al., Proc. CCS, 2016

Our Goal

We leverage ideas from PDEs to improve deep neural nets from the following aspects:

- **Adversarial Robustness**
- **Data Privacy**
- **Generalizability**
- **Nonconvex Optimization**

Laplacian Smoothing (Stochastic) Gradient Descent

Pros of SGD

- Use training data efficiently.

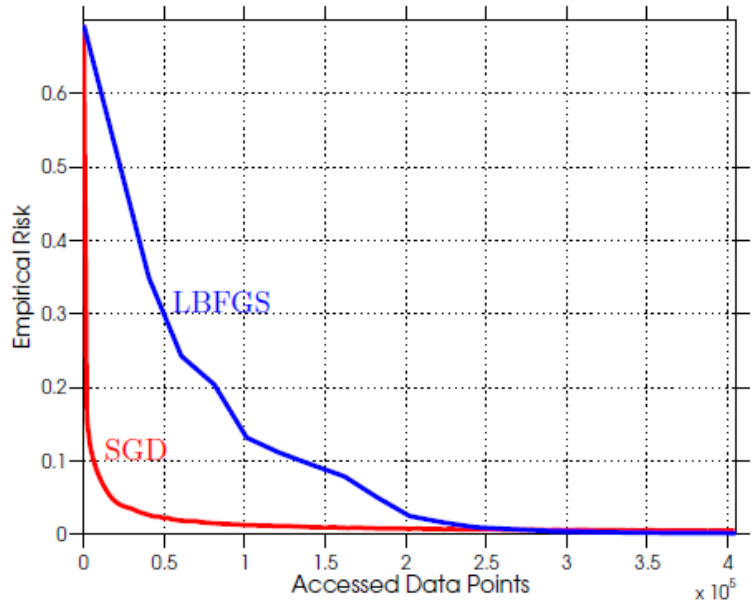


Figure: Training Logistic regression model on RCV1 data. Courtesy of Bottou, Curtis, and Nocedal.

- Make training deep neural nets possible.
- Computational efficiency.

Cons of SGD

- Weaker theoretical guarantee mainly dues to the variance of SGD.
- Cannot protect the privacy of the training data, at least not a good trade-off between privacy and utility.

Laplacian Smoothing Stochastic Gradient Descent

For any differentiable function $F(\mathbf{w})$, consider

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \gamma(I - \sigma L)^{-1}\nabla F(\mathbf{w}^k).$$

Laplacian Smoothing Gradient Descent (LS-GD)

Note the condition number of $(I - \sigma L)^{-1}$ is $1 + 4\sigma$.

High order schemes

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \gamma(I + (-1)^n \sigma L^n)^{-1}\nabla F(\mathbf{w}^k).$$

Osher, Wang, Yin, Luo, Barekat, Pham, and Lin, arXiv:1806.06317, 2018

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

Implementation

Discrete form of $(I - \sigma L)^{-1}$ with periodic boundary condition

$$\textbf{inv}\left(\begin{bmatrix} 1+2\sigma & -\sigma & 0 & \dots & 0 & -\sigma \\ -\sigma & 1+2\sigma & -\sigma & \dots & 0 & 0 \\ 0 & -\sigma & 1+2\sigma & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -\sigma & 0 & 0 & \dots & -\sigma & 1+2\sigma \end{bmatrix}\right)$$

- Thomas algorithm + Sherman Morrison formula.
- **FFT:** Given a vector $\mathbf{a}$, a smoothed vector $\mathbf{b}$ can be obtained by solving $(\mathcal{I} - \sigma L)^{-1}\mathbf{a} = \mathbf{b}$. This is equivalent to $\mathbf{a} = \mathbf{b} - \sigma L \cdot \mathbf{b}$, or $\mathbf{a} = \mathbf{b} - \sigma \mathbf{v} * \mathbf{b}$, where $\mathbf{v} = (-2, 1, 0, \cdots, 0, 1)$ and $*$ is the convolutional operator. Hence, we have

$$\mathbf{b} = \text{ifft}\left(\frac{\text{fft}(\mathbf{a})}{1 - \sigma \cdot \text{fft}(\mathbf{v})}\right),$$

LS(S)GD Allows to Take a Larger Step Size

- LSGD $\mathbf{w}^{k+1} = \mathbf{w}^k - \gamma \mathbf{A}_\sigma^{-1} \nabla F(\mathbf{w}^k)$, is equivalent to

$$\mathbf{v}^{k+1} = \mathbf{v}^k - \eta_k \mathbf{H}_\sigma^{1/2} \nabla F(\mathbf{H}_\sigma^{1/2} \mathbf{v}^k),$$

  where $\mathbf{v}^k = \mathbf{H}_\sigma^{-1/2} \mathbf{w}^k$ and $\mathbf{H}_\sigma = \mathbf{A}_\sigma^{-1}$.

- Let $\mathbf{v}$ be uniformly distributed in the unit ball of the m dimensional ℓ_2 space. Then for any $\alpha > \frac{\sqrt{\beta}}{1 - \frac{\pi}{\sqrt{m}}}$, we have

$$\mathbb{P}\left(\|\mathbf{H}_\sigma^{1/2} \mathbf{v}\| \geq \alpha \|\mathbf{v}\| \right) \leq 2 \exp\left(-\frac{2}{\pi^2} m \left(\frac{\alpha - \alpha \frac{\pi}{\sqrt{m}} - \sqrt{\beta}}{\alpha + 1} \right)^2 \right),$$

  where $\beta = \frac{1}{m} \sum_{j=1}^m \frac{1}{1 + 2\sigma - \sigma z_j - \sigma \bar{z}_j} = \frac{1 + \gamma^m}{(1 - \gamma^m)\sqrt{4\sigma + 1}}$, z_j being the j-th root of unity, and $\gamma = \frac{2\sigma + 1 - \sqrt{4\sigma + 1}}{2\sigma}$.

- Suppose F is L-Lipschitz, the largest step size for GD is $1/L$. While, as m is large enough, the largest step size for LSGD is $\frac{1}{\sqrt{\beta} L}$ with high prob.

LSSGD Reduces Variance of the Stochastic Gradient

Assume the noise in SGD is Gaussian, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, then we have

Proposition Let κ denote the condition number of the covariance matrix Σ. Then, for m dimensional Gaussian random vector $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, we have

$$\frac{\sum_{i=1}^{m} \mathrm{Var}[((\mathbf{A}_\sigma^n)^{-1}\mathbf{n})_i]}{\sum_{i=1}^{m} \mathrm{Var}[(\mathbf{n})_i]} \leq 1 - \frac{1}{\kappa} + \frac{1}{\kappa m} \sum_{j=0}^{m} \frac{1}{[1 + 4^n \sigma \sin^{2n}(\pi j/m)]^2}.$$

Table: Theoretical upper bound of $\frac{\sum_{i=1}^{m} \mathrm{Var}[((\mathbf{A}_\sigma^n)^{-1}\mathbf{n})_i]}{\sum_{i=1}^{m} \mathrm{Var}[(\mathbf{n})_i]}$ where n is the order of smoothing (1: Laplacian smoothing; 2: biharmonic smoothing, etc.), and $\mathbf{n}$ is an m-dimensional standard normal vector with $m \geq 10000$.

σ	1	2	3	4	5
$n = 1$	0.268	0.185	0.149	0.129	0.114
$n = 2$	0.279	0.231	0.207	0.192	0.181
$n = 3$	0.290	0.256	0.238	0.226	0.218

In practice, LS can reduce variance without Gaussian noise assumption.

We will show numerical results on variance reduction later!

Maximum/Minimum Principle and Preserving the Sum

Proposition For any $\mathbf{g} \in \mathbb{R}^m$, let $\mathbf{d} = (I + (-1)^n \sigma L^n)^{-1} \mathbf{g}$, we have

$$\|\mathbf{d}\|_2^2 = \|\mathbf{g}\|_2^2 - 2\sigma \|D_+^n \mathbf{d}\|_2^2 - \sigma^2 \|L^n \mathbf{d}\|_2^2$$

Proposition For any vector $\mathbf{g} \in \mathbb{R}^m$, $\mathbf{d} = \mathbf{A}_\sigma^{-1}\mathbf{g}$, let $j_{\max} = \arg\max_i d_i$ and $j_{\min} = \arg\min_i d_i$. We have $\max_i d_i = d_{j_{\max}} \leq g_{j_{\max}} \leq \max_i g_i$ and $\min_i d_i = d_{j_{\min}} \geq g_{j_{\min}} \geq \min_i g_i$.

Proposition The operator $\mathbf{A}_\sigma^{-1}$ preserves the sum of components. For any $\mathbf{g} \in \mathbb{R}^m$ and $\mathbf{d} = \mathbf{A}_\sigma^{-1}\mathbf{g}$, we have $\sum_j d_j = \sum_j g_j$, or equivalently, $\mathbf{1}^\top \mathbf{d} = \mathbf{1}^\top \mathbf{g}$.

Proposition Given vectors $\mathbf{g}$ and $\mathbf{d} = \mathbf{A}_\sigma^{-1}\mathbf{g}$, for any $p \in \mathbb{N}$, it holds that $\|\mathbf{D}^p \mathbf{d}\|_1 \leq \|\mathbf{D}^p \mathbf{g}\|_1$. The inequality is strict unless $\mathbf{D}^p \mathbf{g}$ is a constant vector.

SGD for Convex Optimization

Consider quadratic function:

$f(x_1, x_2, \cdots x_{10}) = \sum_{i=1}^{10} \frac{x_i^2}{a_i^2}$, $a_i = 1$ if i is odd; 10 otherwise.

Add Gaussian noise to gradient to simulate SGD, i.e., $\nabla^n f = \nabla f + \epsilon N(0, \mathbf{I})$.

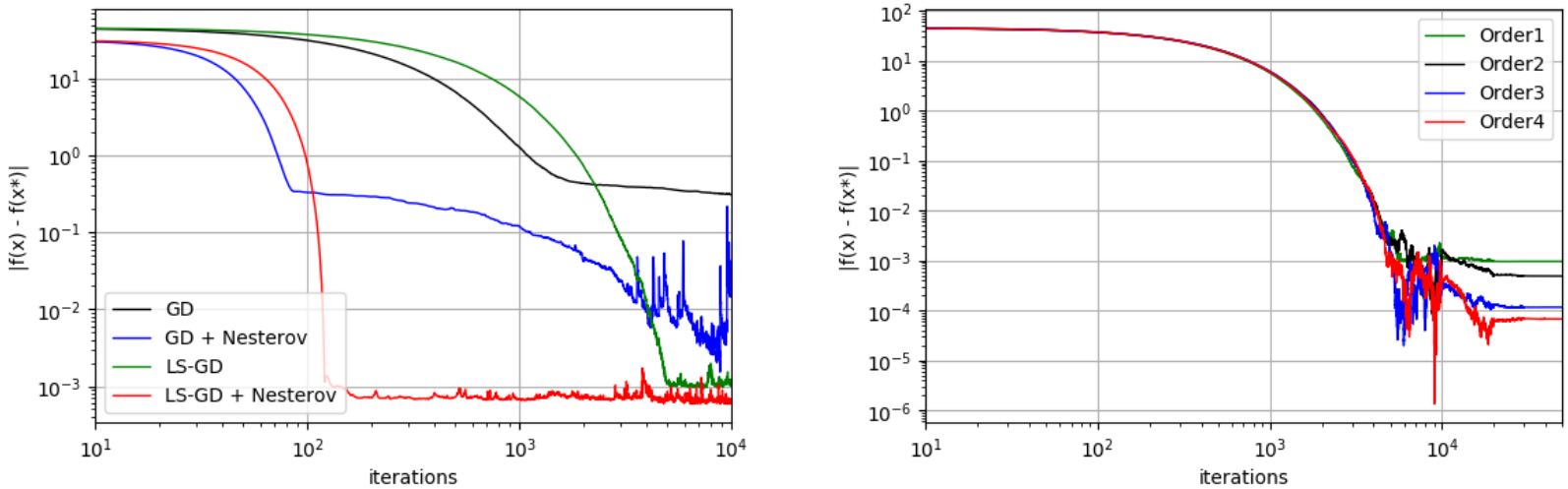


Figure: Solve quadratic function with different stochastic gradient descent methods. ($\mathbf{x}_0 = -(1, 1, \cdots, 1)$, $\epsilon = 1$, step size: 1e-3 (GD), 1.8e-3 (LSGD).)

LS helps to converge faster with/without Nesterov momentum and reduces the optimality gap!

It is not real SGD! Next, we show the results of real SGD.

Minibatch LS-SGD v.s. SGD

We consider the following finite-sum optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^{50}} \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}-\mathbf{d}_i)^T\cdot(\mathbf{x}-\mathbf{d}_i)$$

where $N = 20000$, and we use batch size 20 for both SGD and LS-SGD to find the center $\mathbf{x}^1$.

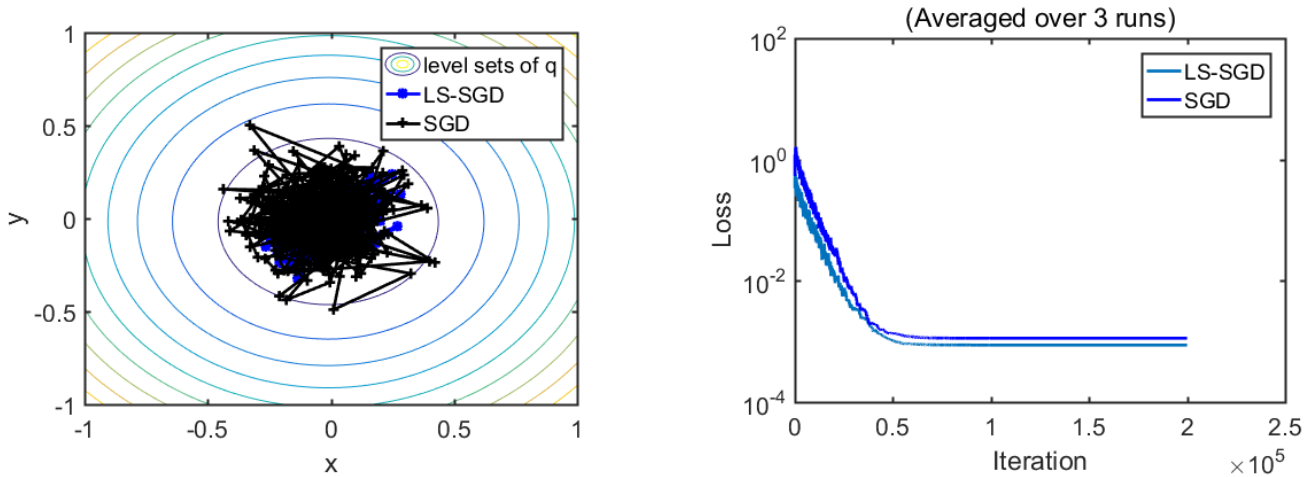
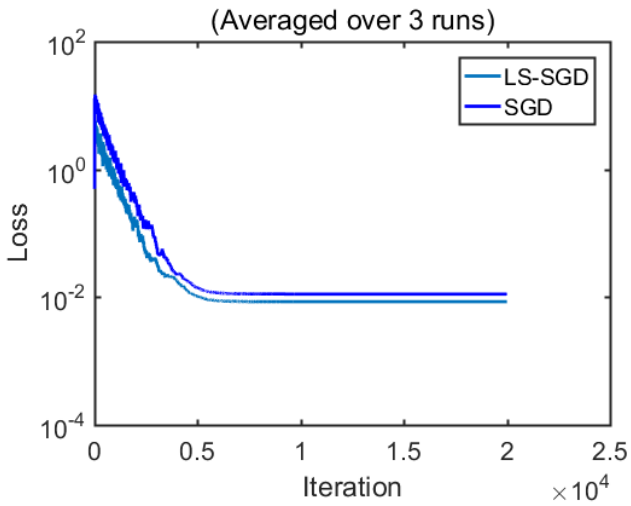


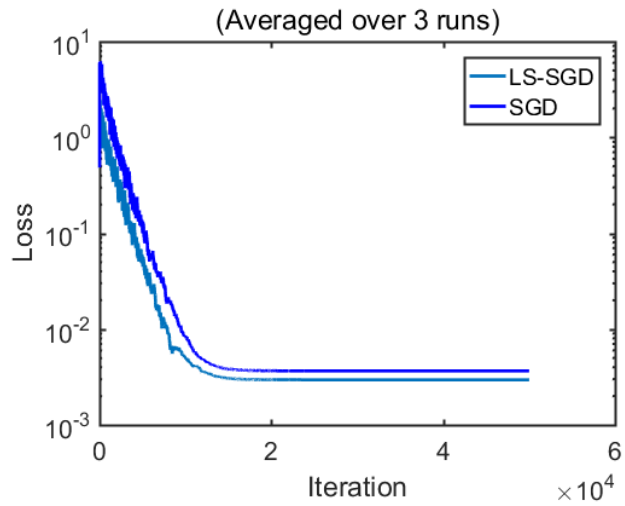
Figure: Left: trajectories of SGD and LS-SGD (2D cross section). Right: Iteration v.s. Loss for SGD and LS-SGD. Step size: 1e-3 (SGD), 1.2e-3 (LS-SGD)

[1] Due to professor Adam Oberman

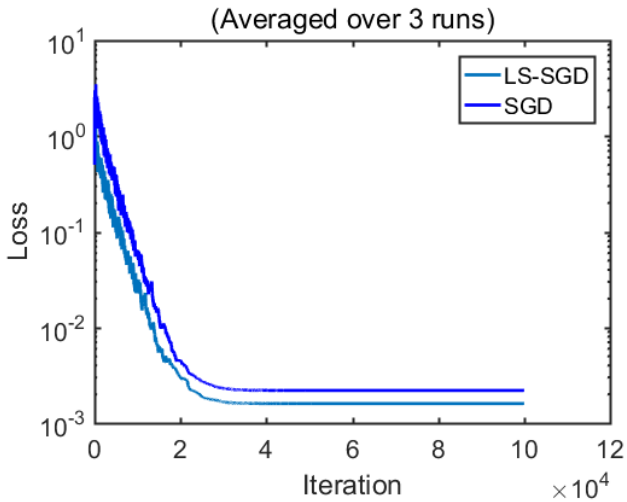
Different Batch Size



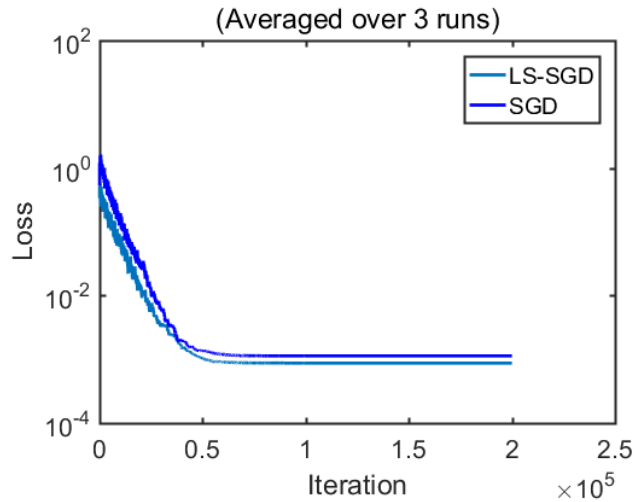
Batch Size 2



Batch Size 5



Batch Size 10



Batch Size 20

We apply LS-GD and LS-SGD with different σ and batch sizes to compute the full batch gradient $\nabla\mathcal{L}$ and stochastic gradient $\tilde{\nabla}\mathcal{L}$ along the descent paths. And then compute the maximum of the coordinate-wise variance among 100 independent experiments.

Table: The maximum variance of the stochastic gradient generated by LS-SGD with different σ and batch size.

Batch Size	2	5	10	20	50
$\sigma = 0$	1.50E-1	5.49E-2	2.37E-2	1.01E-2	4.40E-3
$\sigma = 1$	3.40E-3	1.30E-3	5.45E-4	2.32E-4	9.02E-5
$\sigma = 2$	2.00E-3	7.17E-4	3.46E-4	1.57E-4	5.46E-5
$\sigma = 3$	1.40E-3	4.98E-4	2.56E-4	1.17E-4	3.97E-5

The numerical result is much better than theoretical upper bound!

SGD v.s. LS-SGD - Softmax Regression (Generalization)

Consider the MNIST hand written digits recognition by using the Softmax regression. The models are trained by running 100 epochs of SGD and LS-SGD respectively on the 60000 training instances with batch size 100 and learning rate 0.05.

$\sigma = 0$ $\sigma = 0.2$ $\sigma = 0.5$ $\sigma = 0.8$

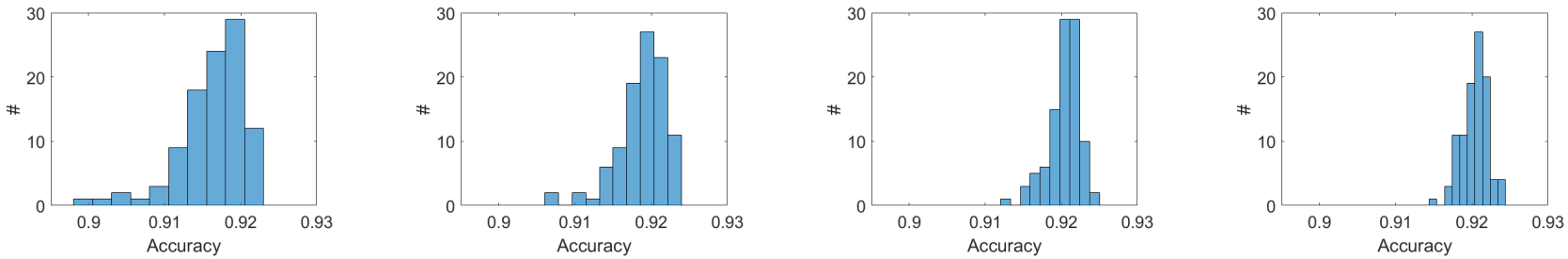


Figure: The histogram of generalization accuracies of the softmax regression model trained with LS-SGD over 100 independent experiments by using different σ.

Training LeNet with Small Batch Size

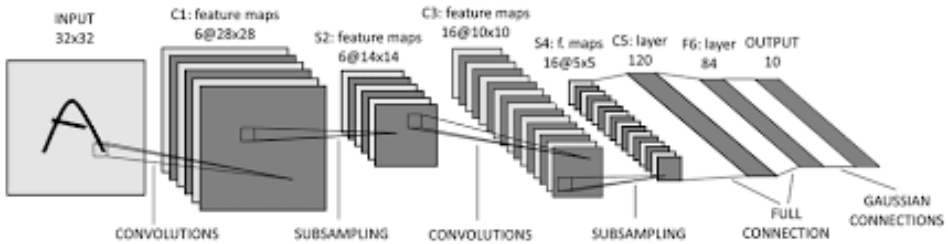


Figure: Architecture of LeNet5.

LeCun et al., Proc. IEEE, 1998

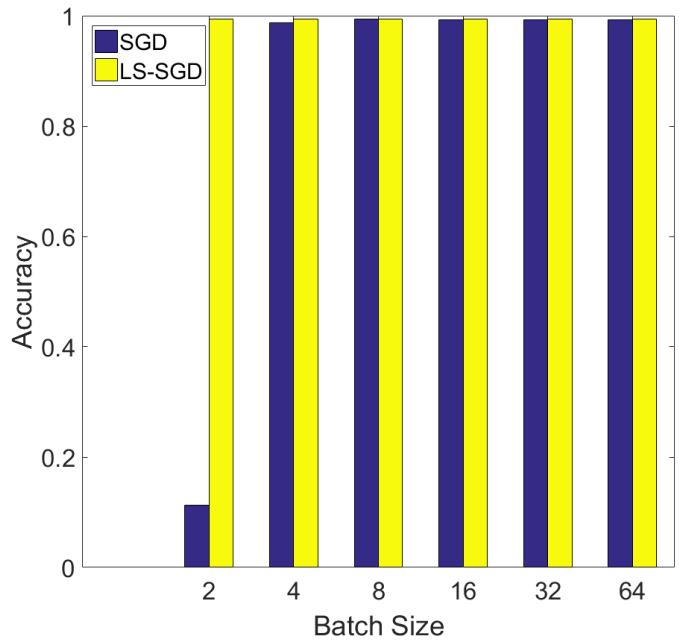
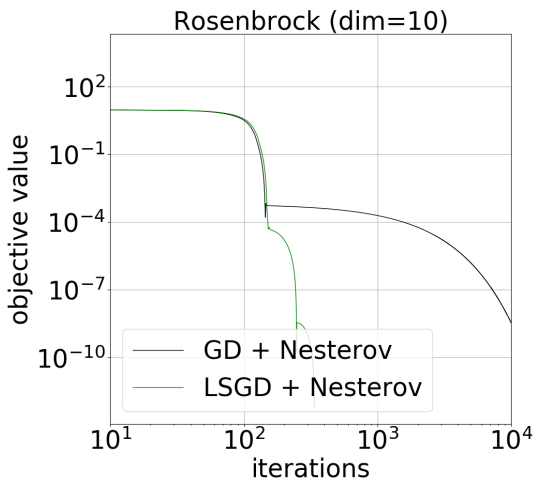


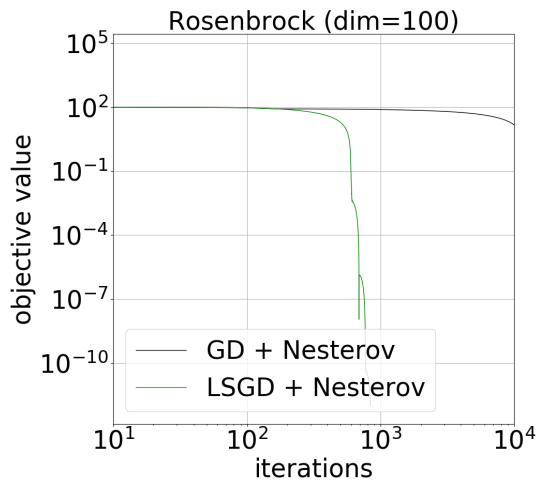
Figure: Generalization accuracy of LeNet5 trained with different batch sizes by SGD and LS-SGD.

Nonconvex Optimization – Rosenbrock Function

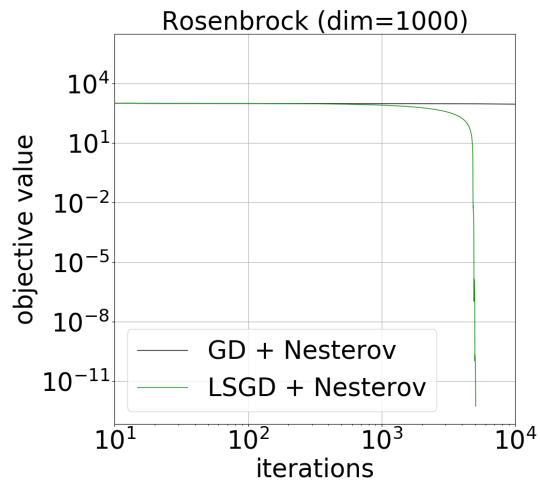
$$f(\mathbf{x}) = f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n/2} [100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2]$$



10D



100D



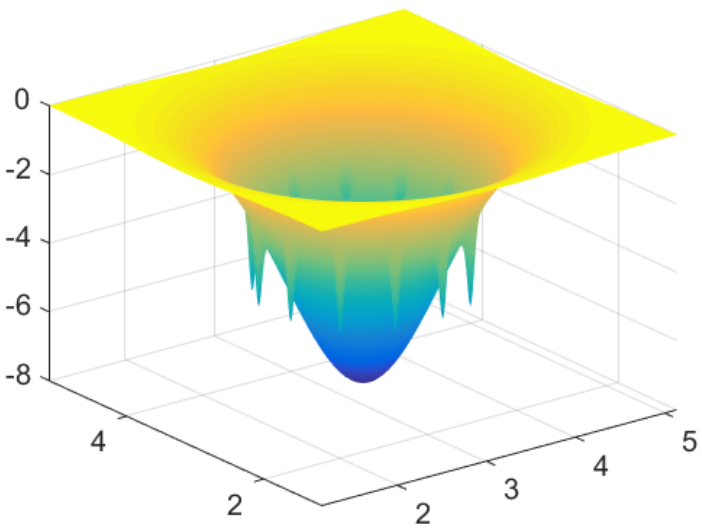
1000D

Circumvent Local Minima - An Illustration

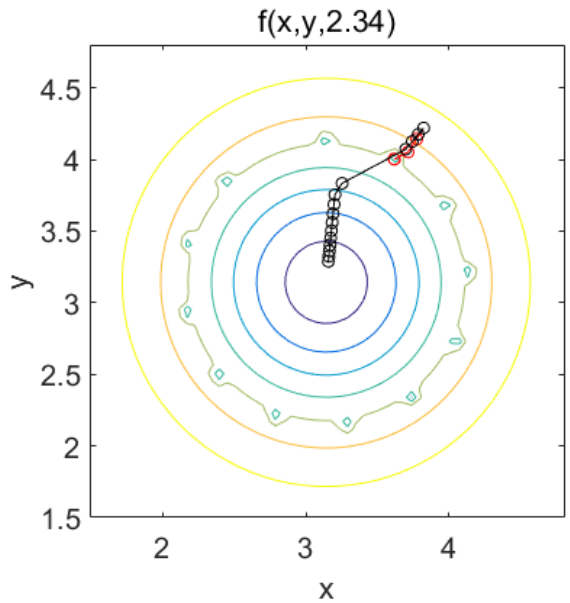
Consider

$$f(x,y,z) = -4e^{-\left((x-\pi)^2+(y-\pi)^2+(z-\pi)^2\right)} - 4\sum_{i/2}\cos(x)\cos(y)e^{-\beta\left((x-r\sin(i/2)-\pi)^2+(y-r\cos(i/2)-\pi)^2\right)} \tag{1}$$

summation over $\{i \in \mathbb{N} | 0 \leq i < 4\pi\}$, $r = 1$, $\beta = \frac{1}{\sqrt{500}}$.



(a)



(b)

Figure: Demo of gradient descent with raw and Laplacian smoothed gradients. Panel (a) depicts a slice of the function given by Eq.(1); panel (b) shows the paths by using two different gradients, where red and black dots are the points on the paths with raw and smoothed gradients, respectively. The learn rate in the gradient descent is set to be 0.02 and the smooth parameter $\sigma = 1.0$.

Related Work and Future Directions

- **Related Work:** LS-SGD is much simpler than the other variance reduction methods, e.g., SAGA, SDCA, SVRG. And LS-SGD is applicable to train deep neural nets with negligible extra cost.
  - LS-SGD is a purely-stochastic method.
  - The others are semi-stochastic methods, which require to compute the full batch gradient or similar stuff.
- **Future Direction:** Analyze the convergence rate of LS-SGD for strongly convex, convex, and smoothing functions.

DP-LSSGD: A Stochastic Optimization Method to Lift the Utility in Privacy-Preserving ERM

Joint work with Bao Wang, Quanquan Gu, March Boedihardjo, and Farzin Barekat

Machine Learning and Data Privacy

- Machine learning should keep the privacy of the training data. Individuals are not generally willing to allow their personal data to be used without control on how it will be used and how much privacy loss they will incur.

- However, the privacy principle is often violated in machine learning.

Recap

Figure: An image recovered using a model inversion attack (left) and a training set image of the victim (right). The attacker is given only the persons name and access to a deep learning based facial recognition system that returns a class confidence score.

Deep neural nets memorise their training data as part of the standard training.

Fredrikson et al., Proc. CCS, 2016

Netflix Challenge

- Netflix released anonymized movie rating data for its Netflix challenge, with data and value of movie ratings.
- Knowing 6-8 approximate movie ratings and dates is able to uniquely identify a record with over 90% probability. A simple way to achieve this is by correlating Netflix releases with IMDB database (public).
- Netflix cancels recommendation contest after privacy lawsuit.

Anonymization cannot Guarantee Privacy!

A. Narayanan and V. Shmatikov, How to break anonymity of the Netflix Prize Dataset, arXiv:cs/0610105, 2006.

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

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

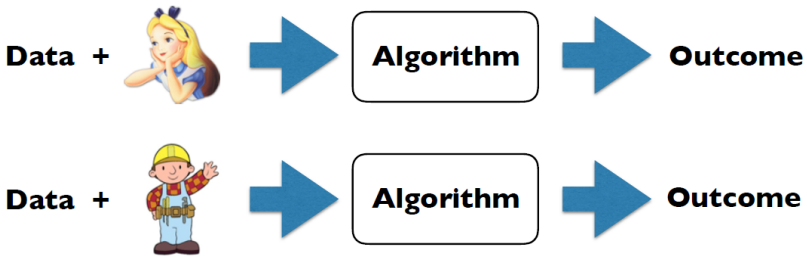
$$Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon} Pr(\mathcal{A}(D') \in S) + \delta,$$

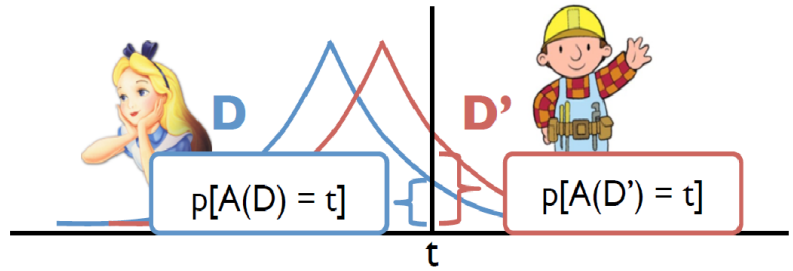
when $\delta = 0$ and $\mathcal{A}$ is ϵ-differentially private .

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

What DP Guarantees?

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.





Participation of a person does not change the outcome much! We cannot infer whether the boy or the girl is participated in the dataset or not based on the outcome. Therefore the privacy of the participant is guaranteed.

Figures courtesy of K. Chaudhuri

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

$$\max_{S, Pr(\mathcal{A}(D) \in S) > \delta} \left\{ \log \frac{Pr(\mathcal{A}(D) \in S) - \delta}{Pr(\mathcal{A}(D')) \in S} \right\} \leq \epsilon$$

Other Notions of Privacy

- **Local DP**

  J. Duchi, M. Jordan, and M. Wainwright. Privacy aware learning. JACM, 61(6), 2014

- **Concentrated DP**

  C. Dwork and G. Rothblum. Concentrated diffentially privacy. arXiv:1603.01887, 2016

- **Zero-concentrated DP**

  M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. arXiv:1605.02065, 2016.

- **Rényi DP**

  I. Mironov. Renyi differential privacy. In Computer Security Foundations Symposium (CSF), IEEE 30th, pp. 263275. IEEE, 2017.

Laplace Mechanism

Definition (ℓ_1-Sensitivity) For any given function $f(\mathbf{x})$, the ℓ_1-sensitivity of f is:

$$\Delta_1(f) = \max_{\|\mathbf{x}-\mathbf{x}'\|_1=1} \|f(\mathbf{x}) - f(\mathbf{x}'))\|_1,$$

where $\|\mathbf{x} - \mathbf{x}'\|_1$ means the data sets $\{\mathbf{x}\}$ and $\{\mathbf{x}'\}$ differ in only one entry.

Remark The privacy of a model f that has a larger sensitivity is easier to break.

Theorem (Laplace Mechanism) Given any function $f : \mathbf{x} \to \mathbf{y}$, the following Laplace mechanism guarantees $(\epsilon, 0)$-differential privacy.

$$\mathcal{M}_L(\mathbf{x}, f(\mathbf{x}), \epsilon) = f(\mathbf{x}) + \mathbf{Y},$$

where $\mathbf{Y}$ is the Laplace noise with the same dimension as $\mathbf{y}$ and each coordinate is sampled i.i.d. from $\mathrm{Lap}(\frac{\Delta_1(f)}{\epsilon}) \sim \frac{1}{\Delta_1(f)/\epsilon} \exp\left(-\frac{|x|}{\Delta_1(f)/\epsilon}\right)$.

Laplace Mechanism – A Simple Example

Suppose we want to study average weight μ of a group of 25 students whose weights are in the interval $[30, 60]$kg. Assume student A wants to keep his weight private.

- Privacy Mechanism I: Allow release only for averages of more than 20 people.
  We only need two queries to break the privacy! Average with and without student A.
- Privacy Mechanism II: Apply the Laplace Mechanism.
  The ℓ_1-sensitivity of the mean over N students, $f(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} x_i$, is $\Delta_1(f) = \max \|f(\mathcal{D}) - f(\mathcal{D}')\|_1 = \frac{30}{N}$ kg.
  Suppose the exact mean is 43.78, and we want to get the privacy with $\epsilon = 1.0$, the five possible private means:

$$43.62,\ 44.19,\ 44.57,\ 45.77,\ 44.52$$

Gaussian Mechanism

Definition (ℓ_2-Sensitivity) For any given function $f(\mathbf{x})$, the ℓ_2-sensitivity of f is:

$$\Delta_2(f) = \max_{\|\mathbf{x}-\mathbf{x}'\|_1=1} \|f(\mathbf{x}) - f(\mathbf{x}'))\|_2,$$

where $\|\mathbf{x} - \mathbf{x}'\|_1$ means the data sets $\{\mathbf{x}\}$ and $\{\mathbf{x}'\}$ differ in only one entry.

Theorem (Gaussian Mechanism) Given any function $f : \mathbf{x} \to \mathbf{y}$, the following Gaussian mechanism guarantees (ϵ, δ)-differential privacy $(0 < \epsilon < 1)$.

$$\mathcal{M}_G(\mathbf{x}, f(\mathbf{x}), \epsilon) = f(\mathbf{x}) + \mathbf{Y},$$

where $\mathbf{Y}$ is the Gaussian noise with the same dimension as $\mathbf{y}$ and each coordinate is sampled i.i.d. from $\mathcal{N}(0, \sigma^2)$ with $\sigma \geq \frac{c\Delta_2(f)}{\epsilon}$ for $\forall\ c^2 > 2\ln\left(\frac{1.25}{\delta}\right)$.

Privacy-Preserving ERM

To obtain the DP solution of the empirical risk minimization,

$$\min \mathcal{L}(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^N) := \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathbf{w}, \{\mathbf{x}_i, y_i\}),$$

we can do:

- **Output Perturbation:**

$$\mathbf{w}^{\text{priv}} = \text{argmin}\mathcal{L}(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^n) + \mathbf{n}_{\text{priv}}^{\alpha}.$$

- **Objective Perturbation:**

$$\mathcal{L}_{\text{priv}}(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^n) = \mathcal{L}(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^n) + \frac{1}{n} < \mathbf{n}_{\text{priv}}^{\beta}, \mathbf{w} > .$$

- **Gradient Perturbation:**

$$\nabla\mathcal{L}_{\text{priv}}(\mathbf{w}^k, \{\mathbf{x}_i, y_i\}_{i=1}^n) = \nabla\mathcal{L}(\mathbf{w}^k, \{\mathbf{x}_i, y_i\}_{i=1}^n) + \mathbf{n}_{\text{priv}}^{\gamma}.$$

Laplace Mechanism: the noise $\mathbf{n}_{\text{priv}}$ is sampled from Laplace distribution, which typically guarantees ϵ-DP.

Gaussian Mechanism: the noise $\mathbf{n}_{\text{priv}}$ is sampled from the Gaussian distribution, which typically guarantees (ϵ, δ)-DP.

DP-SGD

We consider (ϵ, δ)-DP guarantee for SGD:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta \left(\nabla \mathcal{L}(\mathbf{w}^k, \{\mathbf{x}_{i_k}, y_{i_k}\}) + \mathbf{n} \right),$$

where $\mathbf{n} \sim N(0, \nu^2 \mathbf{I})$.

How to quantify the noise **n**?
Composition theorem: if the i-th step is (ϵ_i, δ_i)-DP, then the model trained by using T steps of SGD is $(\sum_{i=1}^{T} \epsilon_i, \sum_{i=1}^{T} \delta_i)$-DP.
However, composition does not give the optimal DP-bound.
Instead, we use the notion of Rényi differential privacy to quantify the noise **n**.

DP-SGD

Definition (RDP) For $\alpha > 1$, $\rho > 0$, a randomized mechanism $\mathcal{M} : \mathcal{S}^n \to \mathcal{R}$ satisfies (α, ρ)-Rényi differential privacy, i.e., (α, ρ)-RDP, if for all adjacent datasets $S, S' \in \mathcal{S}^n$ differing by one element, we have

$$D_\alpha\left(\mathcal{M}(S)||\mathcal{M}(S')\right) := \frac{1}{\alpha - 1} \log \mathbb{E}\left(\frac{\mathcal{M}(S)}{\mathcal{M}(S')}\right)^\alpha \leq \rho,$$

where the expectation is taken over $\mathcal{M}(S')$.

Lemma Given a function $q : \mathcal{S}^n \to \mathcal{R}$, the Gaussian Mechanism $\mathcal{M} = q(S) + \mathbf{n}$, where $\mathbf{n} \sim N(0, \nu^2\mathbf{I})$, satisfies $(\alpha, \alpha\Delta^2(q)/(2\nu^2))$-RDP. In addition, if we apply the mechanism $\mathcal{M}$ to a subset of samples using uniform sampling without replacement, $\mathcal{M}$ satisfies $(\alpha, \tau^2\Delta_2^2(q)\alpha/\nu^2)$-RDP given $\nu^2 \geq 1/1.25$, where τ is the subsample rate.

Lemma If k randomized mechanisms $\mathcal{M}_i : \mathcal{S}^n \to \mathcal{R}$ for $i \in [k]$, satisfy (α, ρ_i)-RDP, then their composition $\left(\mathcal{M}_1(S), \ldots, \mathcal{M}_k(S)\right)$ satisfies $(\alpha, \sum_{i=1}^k \rho_i)$-RDP. Moreover, the input of the i-th mechanism can base on the outputs of previous $(i-1)$ mechanisms.

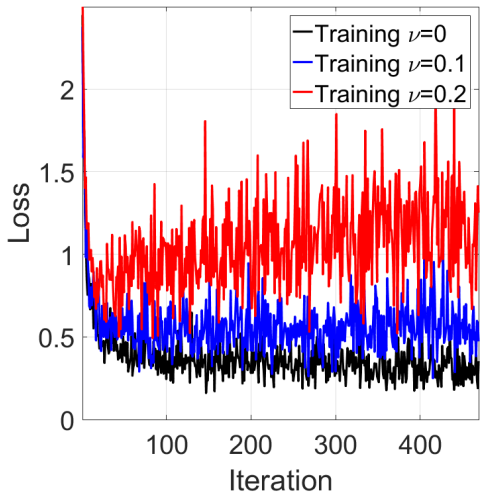
Lemma If a randomized mechanism $\mathcal{M} : \mathcal{S}^n \to \mathcal{R}$ satisfies (α, ρ)-RDP, then $\mathcal{M}$ satisfies $(\rho + \log(1/\delta)/(\alpha - 1), \delta)$-DP for all $\delta \in (0, 1)$.

Theorem (Privacy Guarantee for DPSGD) Suppose $\mathcal{L}$ is G-Lipschitz. Given the total number of iterations T, for any $\delta > 0$ and the privacy budget $\epsilon^2 \leq 20T\log(1/\delta)G^2/n^2$, DP-SGD with injected Gaussian noise $\mathcal{N}(0, \nu^2)$ for each coordinate satisfies (ϵ, δ)-differential privacy with $\nu^2 = 8T\alpha G^2/(n^2\epsilon)$, where $\alpha = 2\log(1/\delta)/\epsilon + 1$.

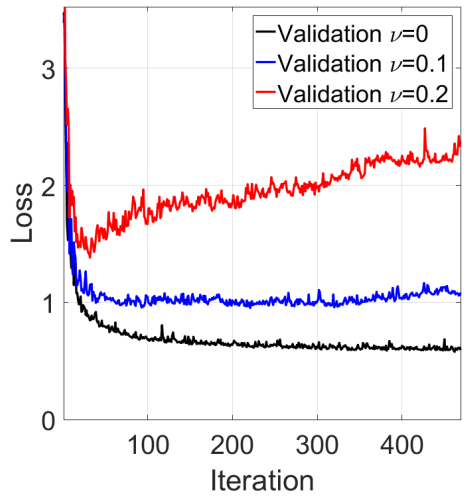
Y. Wang, B. Balle, and S. Kasiviswanathan. arXiv:1808.00087, 2018

I. Mironov. CSF, 2017.

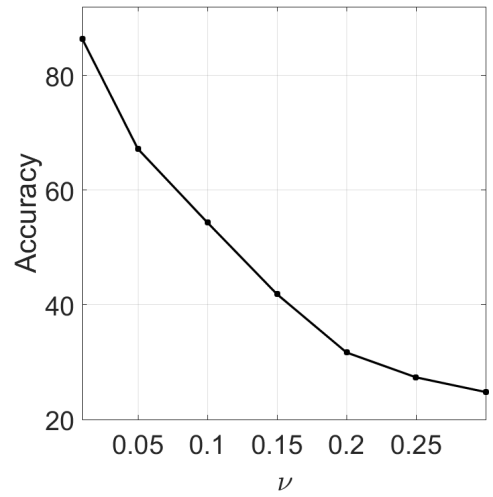
SGD v.s. DP-SGD



Logistic Regression (Training)



Logistic Regression (Testing)



ResNet20 (Testing Accuracy)

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

Question: Can we improve DP-SGD with negligible extra computational and memory costs to improve utility of the trained model without loss of privacy guarantee?

Laplacian Smoothing SGD

For any differentiable function $\mathcal{L}(\mathbf{w})$, consider

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \gamma(I - \sigma L)^{-1}\nabla\mathcal{L}(\mathbf{w}^k).$$

Discrete form of $(I - \sigma L)^{-1}$ with periodic boundary condition

$$\textbf{inv}\left(\begin{bmatrix} 1+2\sigma & -\sigma & 0 & \dots & 0 & -\sigma \\ -\sigma & 1+2\sigma & -\sigma & \dots & 0 & 0 \\ 0 & -\sigma & 1+2\sigma & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -\sigma & 0 & 0 & \dots & -\sigma & 1+2\sigma \end{bmatrix}\right)$$

- **FFT:** Given a vector $\mathbf{a}$, a smoothed vector $\mathbf{b}$ can be obtained by solving $(\mathcal{I} - \sigma L)^{-1}\mathbf{a} = \mathbf{b}$. This is equivalent to $\mathbf{a} = \mathbf{b} - \sigma L \cdot \mathbf{b}$, or $\mathbf{a} = \mathbf{b} - \sigma\mathbf{v} * \mathbf{b}$, where $\mathbf{v} = (-2, 1, 0, \cdots, 0, 1)$ and $*$ is the convolutional operator. Hence, we have

$$\mathbf{b} = \text{ifft}\left(\frac{\text{fft}(\mathbf{a})}{1 - \sigma \cdot \text{fft}(\mathbf{v})}\right).$$

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

DP-LSSGD – Recap

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta \mathbf{A}_\sigma^{-1} \left(\nabla \mathcal{L}(\mathbf{w}^k) + \mathbf{n} \right),$$

where $\mathbf{A}_\sigma := (\mathbf{I} - \sigma \mathbf{L})$.

DP-LSSGD – Privacy Guarantee

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.

For any fixed pair of neighboring databases $\mathbf{x}$, $\mathbf{y}$ with $\|\mathbf{x} - \mathbf{y}\|_1 \leq 1$, and fix any event $S \subset R'$. Let $T = \{r \in R : f(r) \in S\}$. We then have:

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

DP-LSSGD – Privacy Guarantee

Alternatively, we can consider the privacy loss!

Definition (Privacy Loss) For any given randomize mechanism $\mathcal{M}$, we call the following quantity

$$\mathcal{L}^{(\xi)}_{\mathcal{M}(S)||\mathcal{M}(S')} = \ln\left(\frac{\mathbb{P}[\mathcal{M}(S)=\xi]}{\mathbb{P}[\mathcal{M}(S')=\xi]}\right) \tag{2}$$

the privacy loss incurred by observing $\xi \subset \mathrm{Range}(\mathcal{M})$ for any two adjacent databases S and S'.

Remark If the randomized algorithm $\mathcal{M}$ is (ϵ, δ)-DP, then we have

$$\mathbb{P}[\mathcal{L}^{(\xi)}_{\mathcal{M}(S)||\mathcal{M}(S')} \geq \epsilon] = \mathbb{P}[\ln\left(\frac{\mathbb{P}[\mathcal{M}(S)=\xi]}{\mathbb{P}[\mathcal{M}(S')=\xi]}\right) \geq \epsilon] \leq \delta, \tag{3}$$

for any $\xi \subset \mathrm{Range}(\mathcal{M})$ and any two adjacent databases S and S'.

Lemma At any iteration i, the privacy loss of DP-LSSGD is no more than that of DP-SGD.

From privacy loss point of view, it is easy to show that DP-LSSGD is at least (ϵ_1, δ_1)-DP. Note that

$$\mathbb{P}[\ln\left(\frac{\mathbf{A}_\sigma^{-1}(\nabla f(S)+\mathbf{n})=\xi}{\mathbf{A}_\sigma^{-1}(\nabla f(S')+\mathbf{n})=\xi}\right) \geq \epsilon_i] = \mathbb{P}[\ln\left(\frac{(\nabla f(S)+\mathbf{n})=\mathbf{A}_\sigma\xi}{(\nabla f(S')+\mathbf{n})=\mathbf{A}_\sigma\xi}\right) \geq \epsilon_i] \leq \delta_i.$$

DP-LSSGD – Utility Guarantee (Convex)

Theorem (Utility Guarantee for convex optimization)

Suppose $\mathcal{L}$ is convex and each component function $\mathcal{L}_i$ is G-Lipschitz. Given any $\epsilon^2 \leq 20T\log(1/\delta)G^2/n^2, \delta > 0$, if we choose $\eta_k = 1/\sqrt{T}$ and $T = (D+G^2)n^2\epsilon^2/\big(24mG^2\log(1/\delta)\big)$, where $D = \|\mathbf{w}^k - \mathbf{w}^*\|^2_{\mathbf{A}_\sigma}$ and $\mathbf{w}^*$ is the global minimizer of $\mathcal{L}$, the DP-LSSGD output $\tilde{\mathbf{w}} = \sum_{k=0}^{T-1}\eta_k/\big(\sum_{i=0}^{T-1}\eta_i\big)\mathbf{w}^k$ satisfies the following utility

$$\mathbb{E}\big(F(\tilde{\mathbf{w}}) - F(\mathbf{w}^*)\big) \leq \frac{2G\sqrt{6\gamma(D+G^2)d\log(1/\delta)}}{n\epsilon},$$

where $\gamma = 1/m\sum_{i=1}^{m} 1/[1+2\sigma-2\sigma\cos(2\pi i/m)] = \frac{1+\alpha^m}{(1-\alpha^m)\sqrt{4\sigma+1}}$, m is the dimension of $\mathbf{w}$ and $\alpha = \frac{2\sigma+1-\sqrt{4\sigma+1}}{2\sigma}$.

Note, DP-LSSGD has a strictly better utility than DP-SGD by a factor of γ.

To prove the above result, we need to observe that

Lemma

Let $\mathbf{n} \in \mathbb{R}^d$ be the standard Gaussian random vector. Then

$$\mathbb{E}\|\mathbf{n}\|^2_{\mathbf{A}_\sigma^{-1}} = \sum_{i=1}^{m}\frac{1}{1+2\sigma-2\sigma\cos(2\pi i/d)} = \gamma,$$

where $\|\mathbf{n}\|^2_{\mathbf{A}_\sigma^{-1}} \doteq \langle \mathbf{n}, \mathbf{A}_\sigma^{-1}\mathbf{n}\rangle$ is the square of the induced norm of $\mathbf{n}$ by $\mathbf{A}_\sigma^{-1}$.

DP-LSSGD – Utility Guarantee (Nonconvex)

Theorem (Utility Guarantee for nonconvex optimization)

Suppose $\mathcal{L}$ is nonconvex and each component function $\mathcal{L}_i$ is G-Lipschitz and has L-Lispchitz continuous gradient. Given any $\epsilon^2 \leq 20T\log(1/\delta)G^2/n^2, \delta > 0$, if we choose $\eta = 1/\sqrt{T}$ and $T = (D_F + L\nu^2)n^2\epsilon^2/\big(12mLG^2\log(1/\delta)\big)$, where $D_F = F(\mathbf{w}^0) - F(\mathbf{w}^*)$ and $F(\mathbf{w}^*)$ is the global minimum of F, the DP-LSSGD output $\tilde{\mathbf{w}} = \sum_{k=0}^{T-1}\mathbf{w}^k/T$ satisfies the following utility

$$\mathbb{E}\|\nabla F(\tilde{\mathbf{w}})\|_2^2 \leq 4\zeta\frac{G\sqrt{6mL(2D_F + LG^2)\log(1/\delta)}}{n\epsilon}$$

where $\zeta = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\frac{(1+4\sigma)^2}{(1+2\sigma-2\sigma\cos(2\pi i/d))^2}} > 1.$

The utility of DP-SGD is when $\zeta = 1$, DP-LSSGD seems to have a worse utility bound than DP-SGD measured by the gradient norm.

Is Gradient Norm the Right Metric for Measuring Utility?

DP-LSSGD – Utility Guarantee (Nonconvex)

Consider the following simple nonconvex function

$$f(x,y) = \begin{cases} \frac{x^2}{4} + y^2, & \text{for } \frac{x^2}{4} + y^2 \leq 1 \\ \sin\left(\frac{\pi}{2}\left(\frac{x^2}{4} + y^2\right)\right), & \text{for } \frac{x^2}{4} + y^2 > 1 \end{cases}$$

For the two points $\mathbf{a}_1 = (2,0)$ and $\mathbf{a}_2 = (1, \frac{\sqrt{3}}{2})$: the distance to the local minima $\mathbf{a}^* = (0,0)$ are 2 and $\frac{\sqrt{7}}{2}$, while $\|\nabla f(\mathbf{a}_1)\|_2 = 1$ and $\|\nabla f(\mathbf{a}_2)\|_2 = \frac{\sqrt{13}}{2}$. Therefore, $\mathbf{a}_2$ is closer to the local minima $\mathbf{a}^*$ than $\mathbf{a}_1$ while its gradient has a larger ℓ_2-norm.

Logistic Regression DP-SGD v.s. DP-LSSGD (MNIST)

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^n) = \min_{\mathbf{w}} \left\{ \frac{1}{n} \sum_{i=1}^n - \log \left(\frac{\exp \langle \mathbf{w}, \mathbf{x}_i \rangle_{y_i}}{\sum_j \exp \langle \mathbf{w}, \mathbf{x}_i \rangle_j} \right) + \lambda \|\mathbf{w}\|_2 \right\}.$$

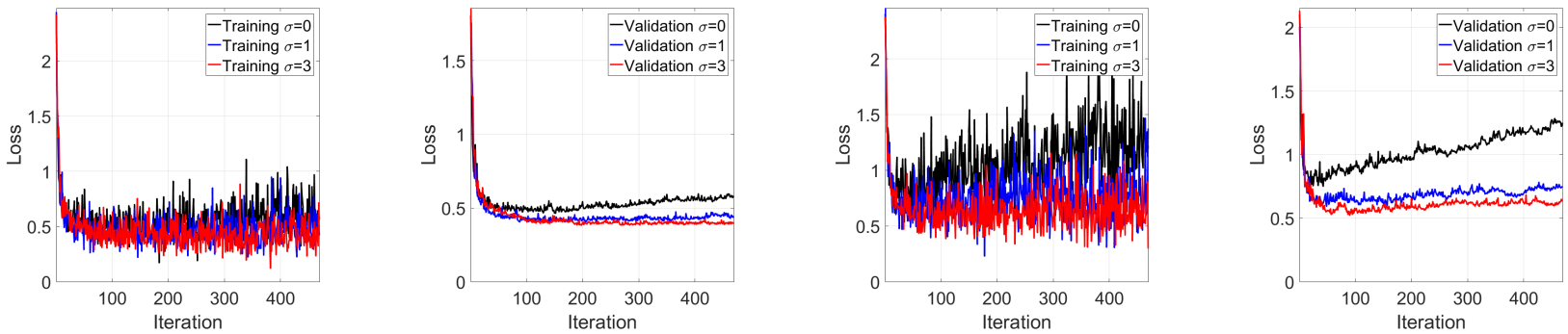
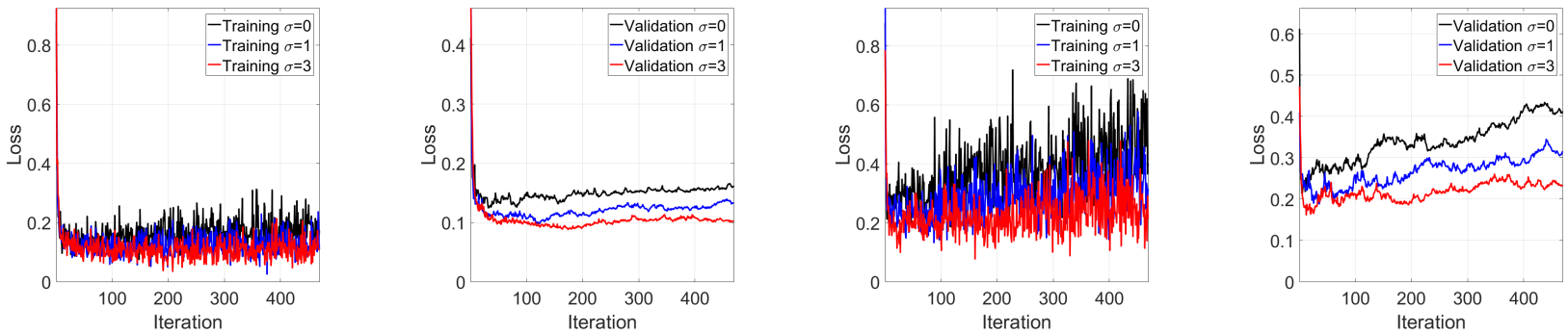


Figure: Training and testing losses of the softmax regression model trained by SGD with different noise injection. (a) and (b): training and testing curves with Gaussian noise injection ($\nu = 0.1$); (c) and (d): training and testing curves with Gaussian noise injection ($\nu = 0.2$).

DP-LSSGD makes training much more stable than DP-SGD and gives better validation loss.

Support Vector Machine DP-SGD v.s. DP-LSSGD (MNIST)

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \{\mathbf{x}_i, y_i\}_{i=1}^n) = \min_{\mathbf{w}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \sum_{j \neq y_i} \max\left(0, \langle \mathbf{w}, \mathbf{x}_i \rangle_j - \langle \mathbf{w}, \mathbf{x}_i \rangle_{y_i} + 1\right) + \lambda \|\mathbf{w}\|_2 \right\}. \tag{4}$$

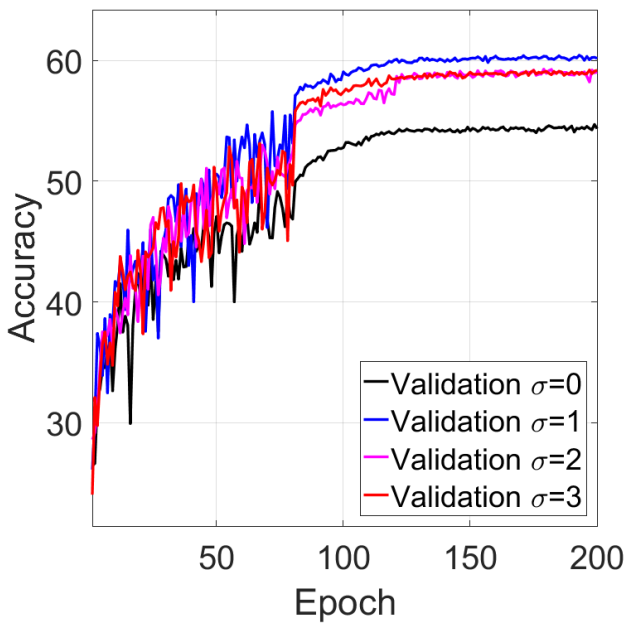


(a) (b) (c) (d)

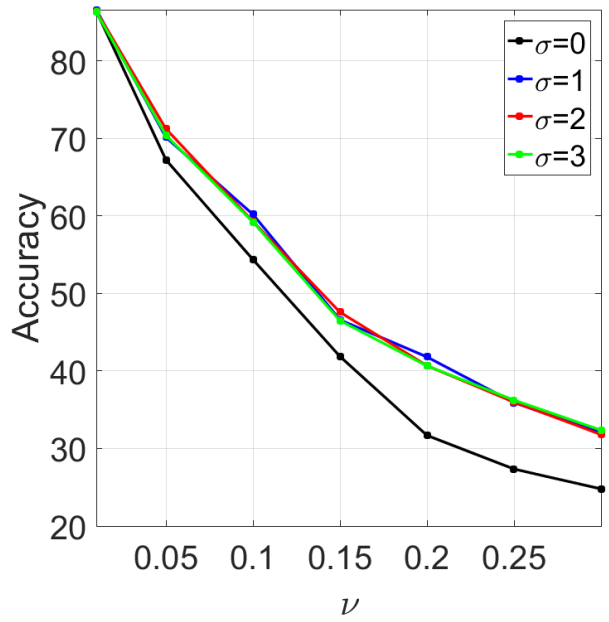
Figure: Training and testing losses of SVM trained by SGD with different noise injection. (a) and (b): training and testing curves with Gaussian noise injection ($\nu = 0.1$); (c) and (d): training and testing curves with Gaussian noise injection ($\nu = 0.2$).

DP-LSSGD makes training much more stable than DP-SGD and gives better validation loss.

Deep Learning DP-SGD v.s. DP-LSSGD (CIFAR10)



(a) (Validation Accuracy)



(b) (Testing Accuracy)

Figure: (a): epoch v.s. validation of the ResNet20 trained by DP-LSSGD with Gaussian noise of standard deviation $\nu = 0.1$ and different Laplacian smoothing parameter σ. (b): Testing accuracy of ResNet20 trained by DP-LSSGD with different Gaussian noise injection.

Deep Learning (Gradient norm is not a good measure for utility!)

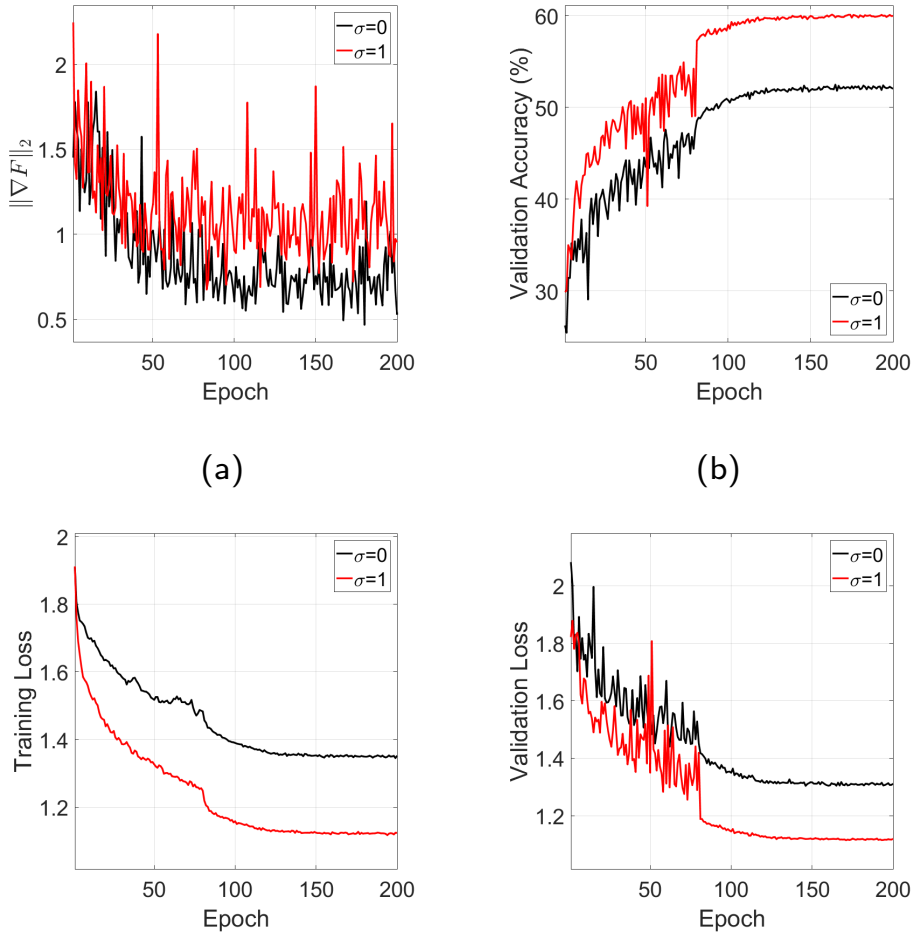


Figure: DP-SGD v.s. DP-LSSGD with $\sigma = 1$ on ResNet20 with Gaussian noise injection ($\nu = 0.1$). (a): epoch v.s. $\|\nabla F(\mathbf{w}^k)\|_2$. (b): epoch v.s. validation accuracy. (c): epoch v.s. training loss. (d): epoch v.s. validation loss.

DP-SGD has smaller gradient norm, but larger training and validation loss and worse validation accuracy! The loss and validation accuracy are better measures for nonconvex optimization.

Theorem

For the random vector $\mathbf{x} \in \mathbb{R}^n$ *that is uniformly distributed in a unit ball, we have the following estimates for* $\forall\ \alpha_1 > 0$ *and* $\alpha_2 > 0$*:*

- $$\mathbb{P}\left(\|\mathbf{A}_\sigma^{-1}\mathbf{x}\|_1 \le \alpha_1\|\mathbf{x}\|_1\right) > 1-\delta_1,$$
- $$\mathbb{P}\left(\|\mathbf{A}_\sigma^{-1}\mathbf{x}\|_2 \le \alpha_2\|\mathbf{x}\|_2\right) > 1-\delta_2,$$

$$\delta_1 = 2e^{-\frac{4}{\pi^3}n^2\left(\frac{\alpha_1-\sqrt{\beta}}{1+\alpha_1}\right)^2}, \quad \delta_2 = 2e^{-\frac{2}{\pi^2}n\frac{\alpha_2-\alpha_2\frac{\pi}{\sqrt{n}}-\sqrt{\beta}}{1+\alpha_2}},$$

with

$$\beta = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|1+2\sigma-\sigma z_i-\sigma\overline{z_i}|^2} \to \frac{1+2\sigma}{(1+4\sigma)^{3/2}},$$

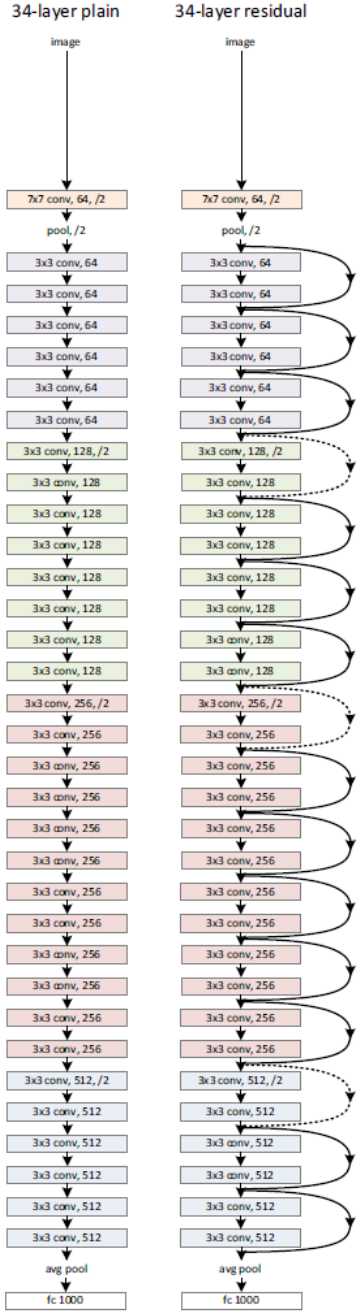
as $d \to \infty$*, where* $z_1, \ldots, z_n$ *are the* n*th roots of unity.*

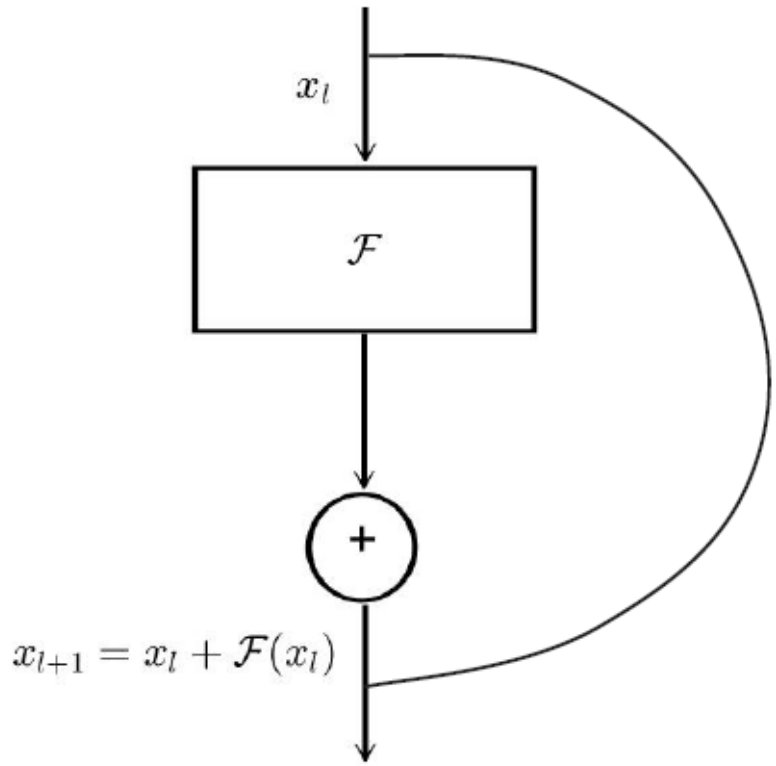
- We can prove better privacy guarantee of DP-LSSGD by using the above estimate.
- Another algorithm: $\mathbf{w}^{k+1} = \mathbf{w}^k - \eta\mathbf{A}_\sigma^{-1}\nabla\mathcal{L}(\mathbf{w}^k) + \mathbf{n}$

Ref: Bao Wang, Quanquan Gu, March Boedihardjo, Farzin Barekat, and Stanley Osher, Privacy-Preserving ERM by Laplacian Smoothing Stochastic Gradient Descent, UCLA-Computational and Applied Mathematics Reports (19-24), April 2019

ResNet and Transport Equation

ResNet





Plain Net: $x_{l+1} = \mathcal{G}(x_l)$

ResNet: $x_{l+1} = x_l + \mathcal{F}(x_l)$

Learning $\mathcal{F}$ is much easier than learning $\mathcal{G}$.

He et al., CVPR, 2016.

ResNet and Transport Equation

For $\forall \hat{x} \in T$ with label y, the forward propagation of ResNet is

$$\begin{cases} x_0 = \hat{x}, \\ x_{k+1} = x_k + \mathcal{F}(x_k, W_k), \quad k = 0, 1, \ldots, L-1, \\ \hat{y} \doteq f(x^L), \end{cases} \tag{5}$$

where $\hat{y}$ is the predicted label, f is the output activation, e.g.,

$$f(x) = \mathrm{softmax}(W_{\mathrm{FC}} \cdot x).$$

Let $t_k = \frac{k}{L}$, for $k = 0, 1, \cdots, L$ with $\Delta t = \frac{1}{L}$. Heuristically, without considering the dimensional consistency, we can regard x_k in Eq. (5) as $x(t_k)$, then Eq. (5) can be rewritten as

$$\begin{cases} x(0) = \hat{x}, \\ x(t_{k+1}) = x(t_k) + \Delta t \cdot \overline{F}(x(t_k), W(t_k)), \ k = 0, 1, \ldots, L-1, \\ \hat{y} \doteq f(x(1)), \end{cases} \tag{6}$$

where $\overline{F} \doteq \frac{1}{\Delta t}\mathcal{F}$.

ResNet and Transport Equation

From the numerical discretization point view, Eq. (6) is just the forward-Euler discretization of the following ODE

$$\frac{dx(t)}{dt} = \overline{F}(x(t), W(t)), \quad x(0) = \hat{x}. \tag{7}$$

For $\forall x \in T$, in the continuum limit ResNet can be viewed as

$$\begin{cases} \frac{dx(t)}{dt} = \overline{F}(x(t), W(t)), \\ x(0) = \hat{x}, \\ \hat{y} = f(x(1)). \end{cases} \tag{8}$$

ResNet and Transport Equation

Furthermore, Eq. (7) defines the characteristic curves of the following transport equation

$$\frac{\partial u}{\partial t}(x,t) + \overline{F}(x, W(t)) \cdot \nabla u(x,t) = 0, \quad x \in \mathbb{R}^d. \tag{9}$$

Along the characteristic curve determined by Eq. (7), we have

$$\frac{du(x(t),t)}{dt} = \frac{\partial u}{\partial t}(x(t),t) + \overline{F}(x(t), W(t)) \cdot \nabla u(x(t),t) = 0,$$

therefore,

$$u(\hat{x}, 0) = u(x(0), 0) = u(x(1), 1). \tag{10}$$

If we enforce the terminal condition at $t = 1$ for Eq. (9) to be

$$u(x, 1) = f(x).$$

According to Eq. (10), we have $u(\hat{x}, 0) = u(x(1), 1) = f(x(1))$.

ResNet and Transport Equation

The forward propagation of ResNet can be modeled as computing $u(\hat{x}, 0)$ along the characteristics of the following transport equation

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) + \overline{F}(x, W(t)) \cdot \nabla u(x,t) = 0, \quad x \in \mathbb{R}^d, \\ u(x,1) = f(x). \end{cases} \tag{11}$$

Meanwhile, the backpropagation in training ResNet can be modeled as finding the velocity field $W(t)$ with a given form of $\overline{F}(x(t), W(t))$, for the following control problem

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) + \overline{F}(x, W(t)) \cdot \nabla u(x,t) = 0, \quad x \in \mathbb{R}^d, \\ u(x,1) = f(x), \\ u(x_i, 0) = y_i, \quad x_i \in T, \end{cases} \tag{12}$$

where $u(x_i, 0) = y_i, \quad x_i \in T$ with y_i be the label of x_i, enforces the initial condition on the training data.

Improving Adversarial Robustness of Deep Neural Nets

Joint work with Bao Wang, Binjie Yuan, and Zuoqiang Shi

Controlling the Smoothness of Decision Boundary

In the transport equation model, $u(x, 0)$ serves as the decision function to classify data, which might be singular and not robust to adversarial attack. We add a viscosity term to smooth this decision function

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

The decision function $u(\hat{x}, 0)$ can be obtained by using the Feynman-Kac formula, which gives

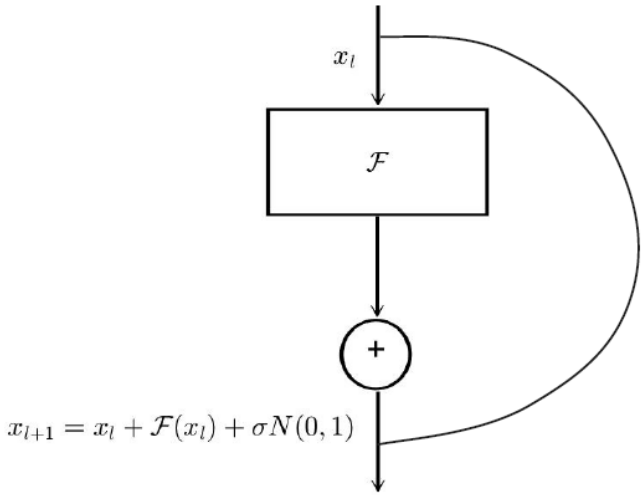
$$u(\hat{x}, 0) = \mathbb{E}\left[f(x(1))|x(0) = \hat{x}\right],$$

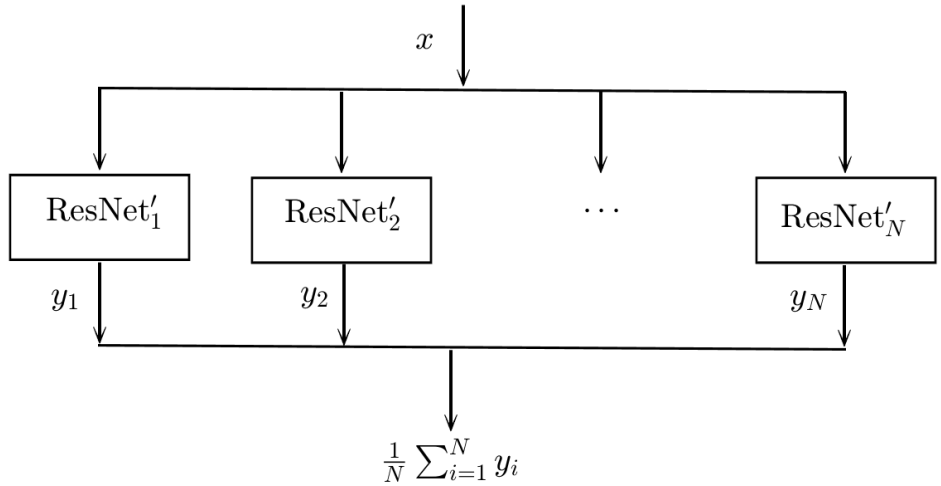
where $x(t)$ is an Itô process,

$$dx(t) = F(x(t), W(t))dt + \sigma dB_t,$$

and $u(\hat{x}, 0)$ is the conditional expectation of $f(x(1))$.

ResNet Approximation of the Feynman-Kac Formula





Add Gaussian noise to residual mapping.

Average over multiple jointly trained ResNet

Wang, Yuan, Shi, and Osher, arXiv:1811:10745, 2018.

Adversarial Robustness

For the given training data $\mathcal{D} \doteq \{(x, y)\}$ and loss function $\mathcal{L}(\theta, x, y)$, we can train an adversarially resistant model $f(x, \theta)$ by solving the following min-max formalism

$$\min_{\theta} \rho(\theta) = \min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta \in S} \mathcal{L}(\theta, x + \delta, y) \right], \tag{13}$$

where δ is the admissible adversarial perturbation. We solve the inner maximization by PGD adversarial attack.

Table: Robust testing accuracy of ResNet20 and En_2ResNet20 on the CIFAR10 benchmark under both white-box PGD and C&W attacks.

Model Under Attack	Attack	Distance	Accuracy
ResNet20	White-Box PGD	0.031 (ℓ_∞)	46.70%
ResNet110	White-Box PGD	0.031 (ℓ_∞)	50.28%
En_1ResNet20	White-Box PGD	0.031 (ℓ_∞)	49.28%
En_2ResNet20	White-Box PGD	0.031 (ℓ_∞)	52.10%
En_5ResNet20	White-Box PGD	0.031 (ℓ_∞)	**54.89%**
ResNet20	White-Box C&W	0.031 (ℓ_∞)	57.78%
ResNet110	White-Box C&W	0.031 (ℓ_∞)	62.92%
En_1ResNet20	White-Box C&W	0.031 (ℓ_∞)	64.97%
En_2ResNet20	White-Box C&W	0.031 (ℓ_∞)	66.41%
En_5ResNet20	White-Box C&W	0.031 (ℓ_∞)	**68.67%**

Improving Generalization of Deep Neural Nets

Joint work with Bao Wang, Xiyang Luo, Wei Zhu, Zhen Li, and Zuoqiang Shi

Training and Testing for DNNs with Softmax Activation

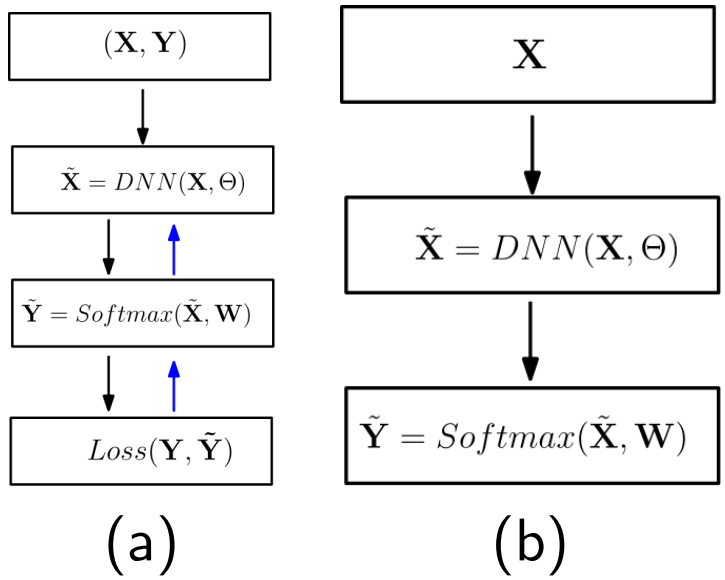


Figure: Training (a) and testing (b) of DNNs with softmax as output layer.

Forward propagation: Transform $\mathbf{X}$ into deep features by DNN block (ensemble of conv layers, nonlinearities and others), and then activated by softmax function to obtain the predicted labels $\tilde{\mathbf{Y}}$:

$$\tilde{\mathbf{Y}} = \mathrm{Softmax}(\mathrm{DNN}(\mathbf{X}, \Theta^{k-1}), \mathbf{W}^{k-1}).$$

Then compute loss (e.g., cross entropy) between $\mathbf{Y}$ and $\tilde{\mathbf{Y}}$: $\mathcal{L} = \mathrm{Loss}(\mathbf{Y}, \tilde{\mathbf{Y}})$.
Backpropagation: Update weights (Θ^{k-1}, $\mathbf{W}^{k-1}$) by gradient descent (learning rate γ):

$$\mathbf{W}^k = \mathbf{W}^{k-1} - \gamma \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \mathbf{W}}, \quad \Theta^k = \Theta^{k-1} - \gamma \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \tilde{\mathbf{X}}} \cdot \frac{\partial \tilde{\mathbf{X}}}{\partial \Theta}.$$

Data-Dependent Activation Function

We replace Softmax function, which is training data agnostic, to a high dimensional interpolating function by minimizing the Dirichlet energy:

$$\mathcal{E}(u) = \frac{1}{2} \sum_{\mathbf{x},\mathbf{y} \in \mathbf{X}} w(\mathbf{x},\mathbf{y}) \left(u(\mathbf{x}) - u(\mathbf{y})\right)^2, \tag{14}$$

with the boundary condition:

$$u(\mathbf{x}) = g(\mathbf{x}), \ \mathbf{x} \in \mathbf{X}^{\mathrm{te}},$$

where $w(\mathbf{x},\mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{\sigma^2})$ is the weight function. And $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ be a set of points in a high dimensional manifold $\mathcal{M} \subset \mathbb{R}^d$ and $\mathbf{X}^{\mathrm{te}} = \{\mathbf{x}_1^{\mathrm{te}}, \mathbf{x}_2^{\mathrm{te}}, \cdots, \mathbf{x}_m^{\mathrm{te}}\}$ be a labeled subset of $\mathbf{X}$ with label $g(\mathbf{x})$. Moreover, to resolve the issue of only tiny amount of data been labeled, we add a balance factor to the variational derivative to Eq. (14), which gives:

$$\begin{cases} \sum_{\mathbf{y} \in \mathbf{X}} \left(w(\mathbf{x},\mathbf{y}) + w(\mathbf{y},\mathbf{x})\right)\left(u(\mathbf{x}) - u(\mathbf{y})\right) + \\ \left(\frac{|\mathbf{X}|}{|\mathbf{X}^{\mathrm{te}}|} - 1\right) \sum_{\mathbf{y} \in \mathbf{X}^{\mathrm{te}}} w(\mathbf{y},\mathbf{x})\left(u(\mathbf{x}) - u(\mathbf{y})\right) = 0 & \mathbf{x} \in \mathbf{X}/\mathbf{X}^{\mathrm{te}} \\ u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in \mathbf{X}^{\mathrm{te}}. \end{cases} \tag{15}$$

We call Eq. (15) weighted nonlocal Laplacian (WNLL).

Training and Testing for WNLL Activated DNNs

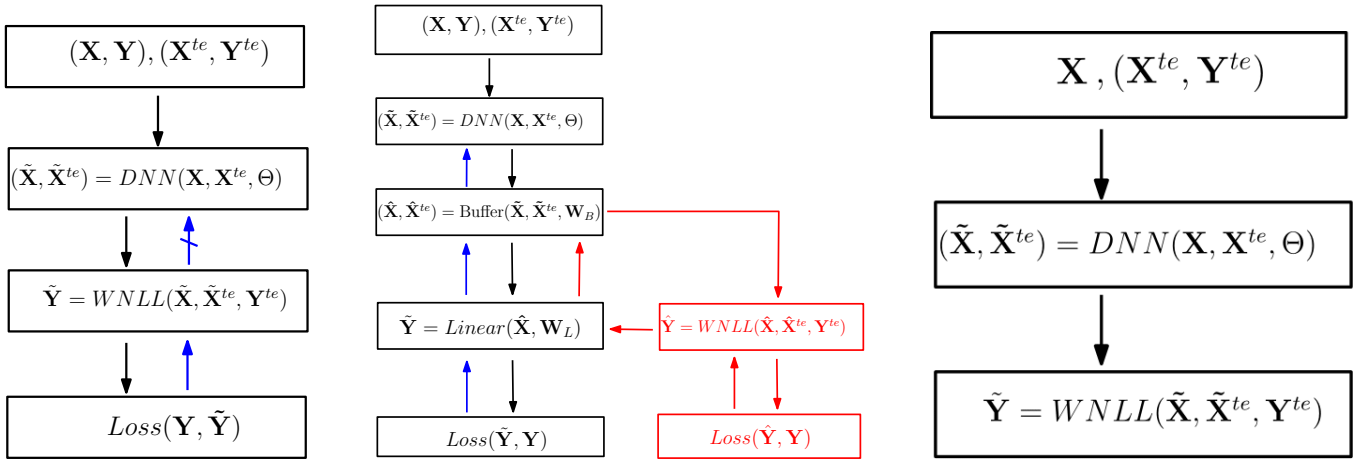


Figure: Training and testing procedure of the deep neural nets with WNLL as the last activation layer.(a): Direct replacement of the softmax by WNLL, (b): An alternating training procedure. (c): Testing.

Direct replacement of softmax to WNLL has the issue of backprop, instead we apply the training strategy in panel (b).

$$\mathbf{W}_B^k = \mathbf{W}_B^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{WNLL}}}{\partial \hat{\mathbf{Y}}} \cdot \frac{\partial \hat{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{W}_B} \approx \mathbf{W}_B^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{W}_B}.$$

Performance on Cifar-10

Table: Generalization error rates over the test set of vanilla DNNs, SVM and WNLL activated ones trained over the entire, the first 10000, and the first 1000 instances of training set of CIFAR10. (Median of 5 independent trials)

Network	Whole			10000		1000	
	Vanilla	**WNLL**	**SVM**	**Vanilla**	**WNLL**	**Vanilla**	**WNLL**
ResNet32	7.99%	**5.95%**	8.73%	11.18%	**8.15%**	33.41%	**28.78%**
ResNet44	7.31%	**5.70%**	8.67%	10.66%	**7.96%**	34.58%	**27.94%**
ResNet56	7.24%	**5.61%**	8.58%	9.83%	**7.61%**	37.83%	**28.18%**
ResNet110	6.41%	**4.98%**	8.06%	8.91%	**7.13%**	42.94%	**28.29%**
ResNet18	6.16%	**4.65%**	6.00%	8.26%	**6.29%**	27.02%	**22.48%**
ResNet34	5.93%	**4.26%**	6.32%	8.31%	**6.11%**	26.47%	**20.27%**
ResNet50	6.24%	**4.17%**	6.63%	9.64%	**6.49%**	29.69%	**20.19%**
PreActResNet18	6.21%	**4.74%**	6.38%	8.20%	**6.61%**	27.36%	**21.88%**
PreActResNet34	6.08%	**4.40%**	5.88%	8.52%	**6.34%**	23.56%	**19.02%**
PreActResNet50	6.05%	**4.27%**	5.91%	9.18%	**6.05%**	25.05%	**18.61%**

Wang, Luo, Li, Zhu, Shi, and Osher, NIPS, 2018

Feature's Geometry

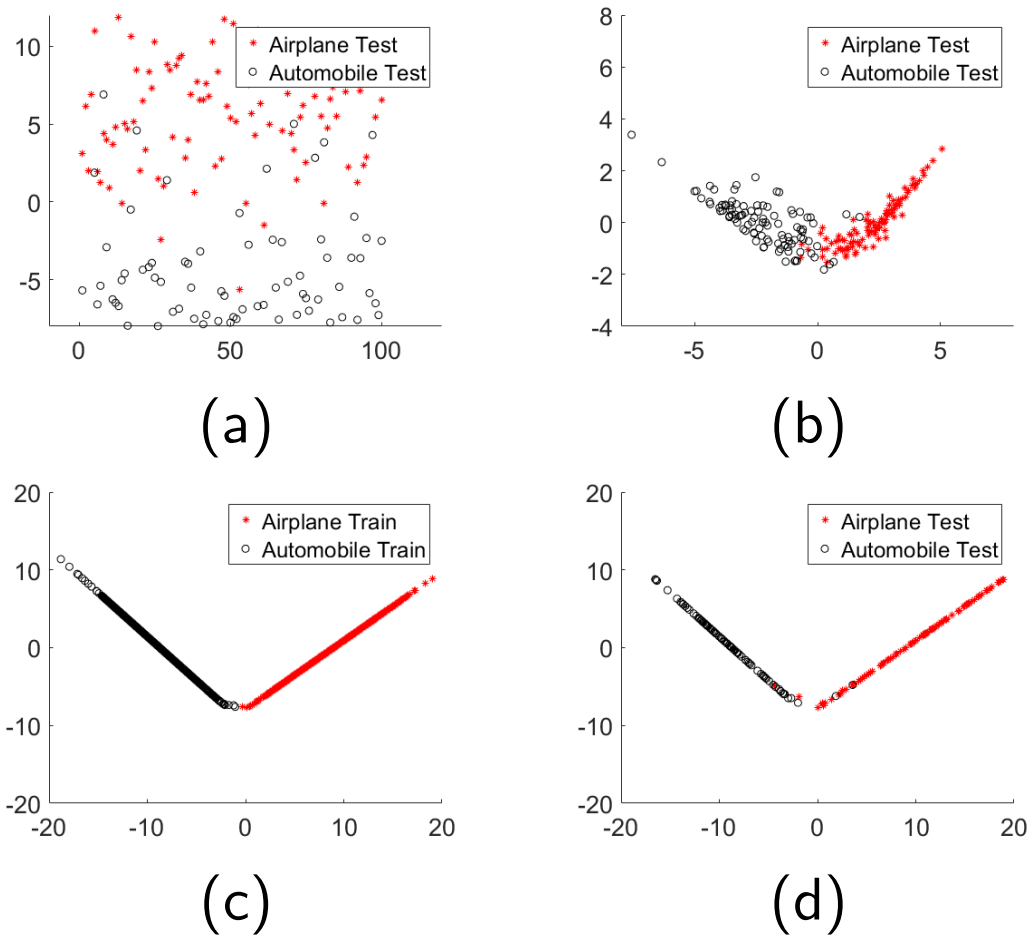


Figure: Visualization of the features learned by DNNs with softmax ((a), (b)) and WNLL ((c), (d)) activation functions. (a) and (b) plot the 2D features and the first two principle components of the 64D. (c) and (d) are the first two principle components of the 64D features for the training and testing images, respectively.

Summary

WNLL activated DNNs can improve generalization remarkably, especially when we do not have a large number of training data.

Code: `https://github.com/BaoWangMath/DNN-DataDependentActivation`

Thank you!

Special Thanks to professor Adam Oberman for stimulating discussions over the entire project!