Decentralized Optimization Algorithms for Large-Scale Deep Neural Network Training

Kun Yuan

DAMO Academy, Alibaba Group

Joint work with Yiming Chen, Pan Pan, Yinghui Xu, Wotao Yin (Alibaba), Bicheng Ying (UCLA), Hanbin Hu (UCSB), Xinmeng Huang (Upenn) , and Sulaiman A. Alghunaim (Kuwait University)

Aug 5, 2021, Zhejiang University

Contents in the lecture

Introduction to deep neural network (DNN) and various training modes (Part I)

- Stochastic gradient descent and single-node training
- Parallel/distributed training
- Decentralized training

Making decentralized algorithms practical for large-scale deep training (Part II)

- Exponential graphs
- Primal-dual decentralized methods
- Periodic global averaging

Other advanced topics and BlueFog (Part III)

- Large-batch deep training
- An open source decentralized deep training framework: BlueFog

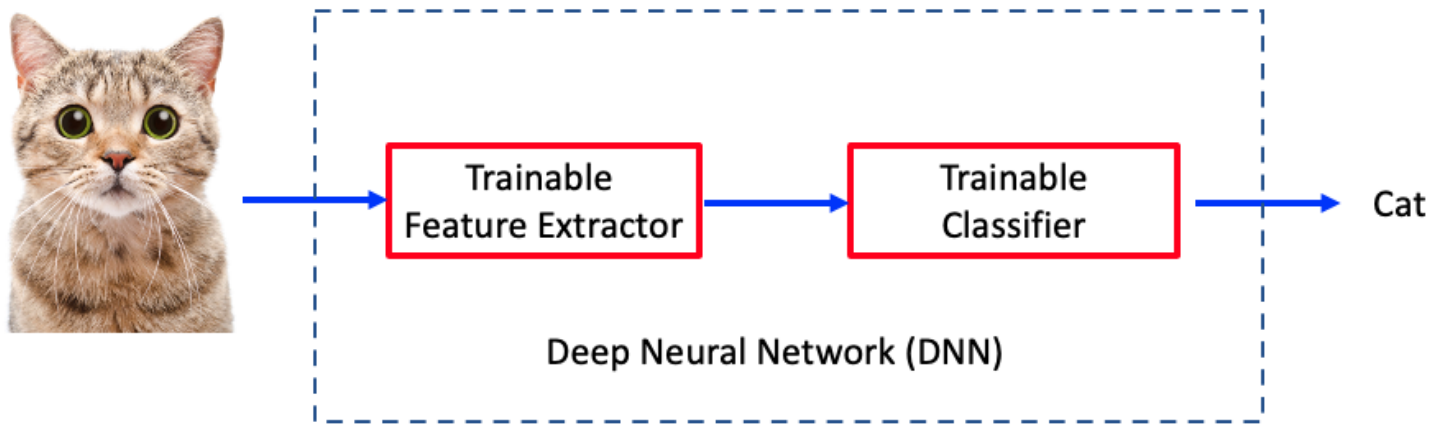
Part I: Deep Neural Network (DNN) Training Algorithms

Part I: Deep Neural Network (DNN) Training Algorithms

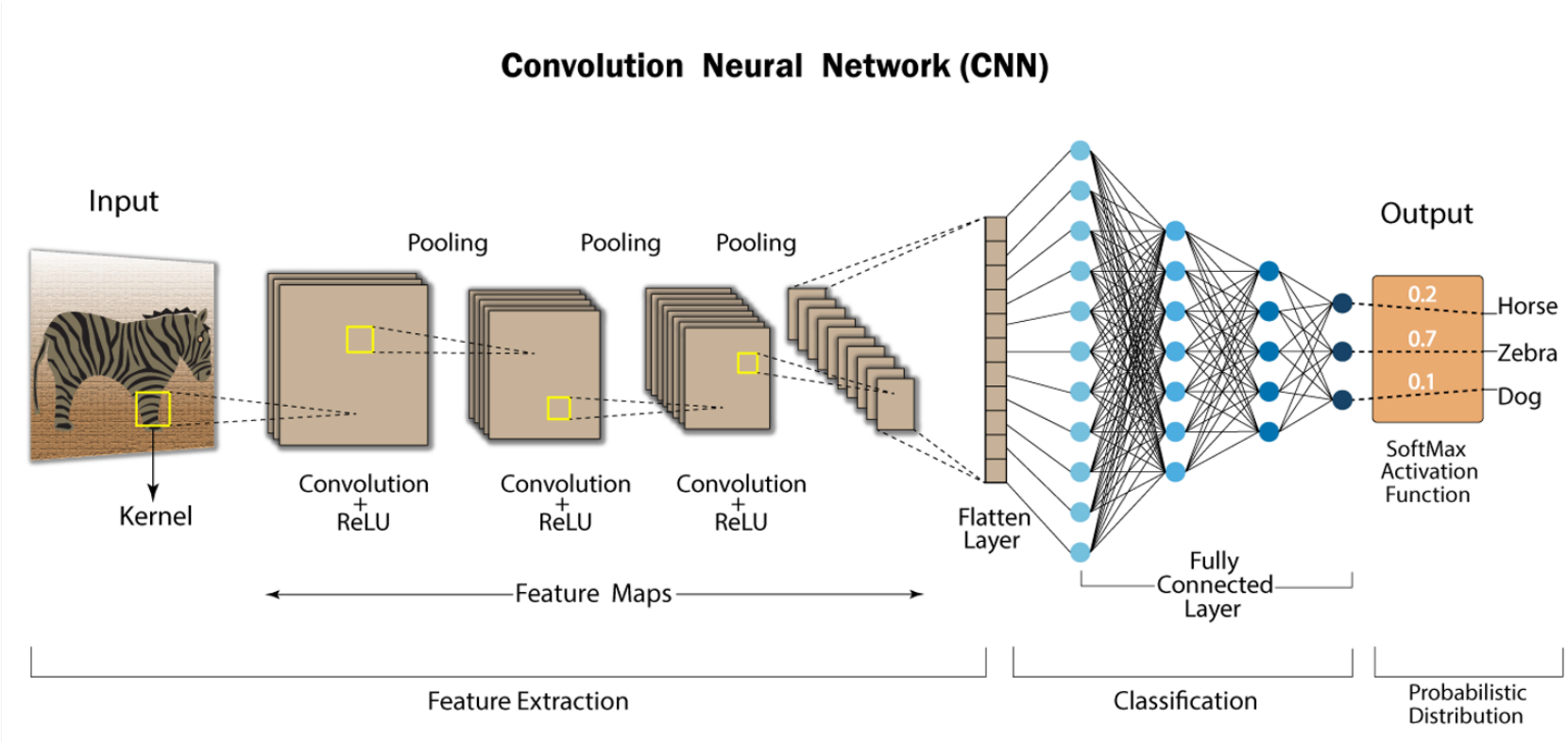
- Sec.1: Deep Neural Network Model
- Sec.2: Stochastic Gradient Descent and Single-Node Training
- Sec.3: Parallel/Distributed Training
- Sec.4: Decentralized Training

Deep Neural Network

- DNN is widely used in almost all AI applications
- A typical DNN model includes a feature extractor and a classifier
- Well-trained DNN can make precise predictions



A practical DNN example[1]

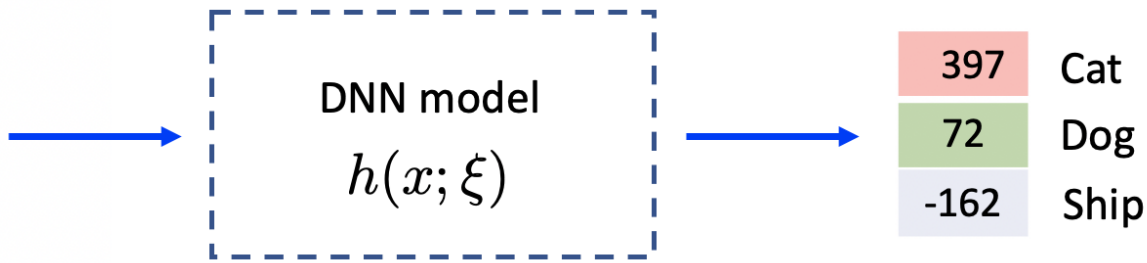


[1] Source: analyticsvidhya.com

DNN model

- We model DNN as $h(x;\xi) : \mathbb{R}^d \to \mathbb{R}^c$
  - $x \in \mathbb{R}^d$ is the DNN model parameter to be trained
  - ξ is the input data sample
  - c is the number of classes

- Given the model parameter x, DNN outputs prediction scores $\hat{y}_i$ for input ξ_i

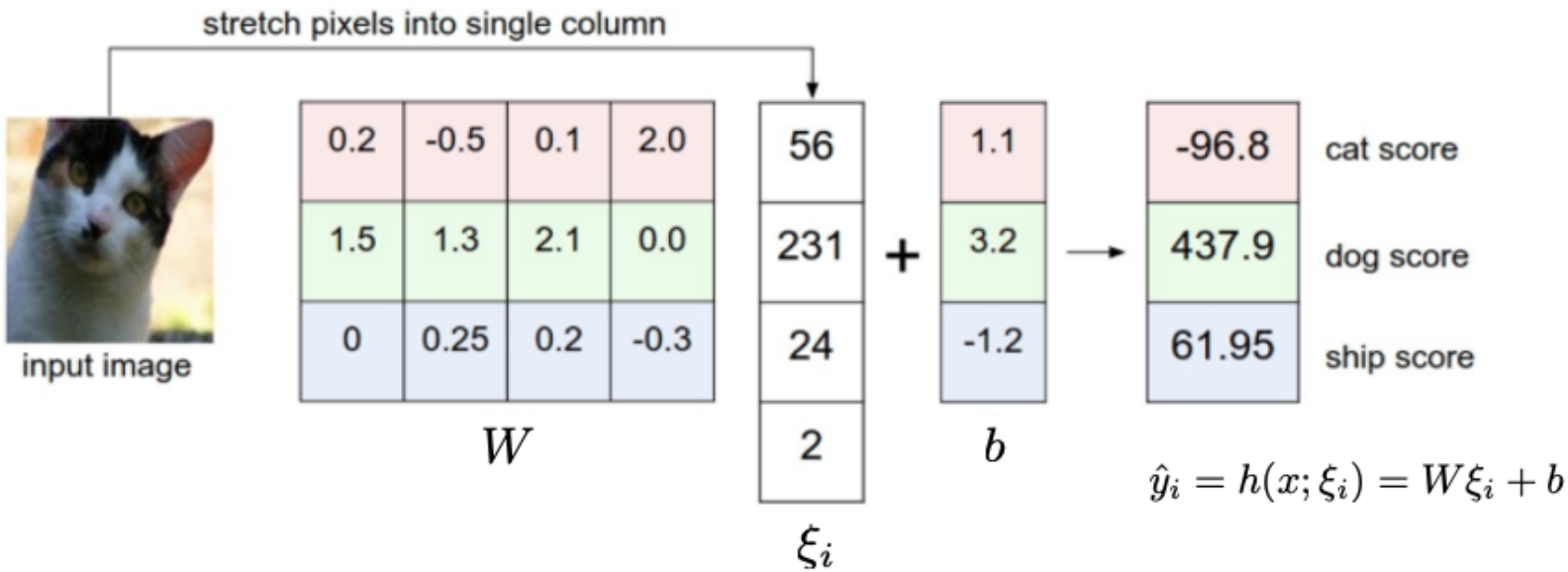


ξ_i

$\hat{y}_i = h(x;\xi_i) \in \mathbb{R}^c$

DNN model: a trivial example

- Given model parameter $x = [W; b]$, and a linear model $h(x; \xi) = W\xi + b$,
- An illustration of the trivial DNN model and its output is as follows[2]



[2] Source: https://cs231n.github.io/linear-classify/

How to train a DNN model?

- Given model parameter x, DNN $h(x;\xi)$ can make precise predictions
- But how to train/achieve the model parameter x ?
- Given a dataset $\{\xi_i, y_i\}_{i=1}^m$ where y_i is the ground-truth label for data ξ_i
- Define $L(\hat{y}_i, y_i) = L(h(x;\xi_i), y_i)$ as a loss function to measure the difference/mismatch between predictions and ground-truth labels
- DNN training is to find a model parameter x such that the mismatch (between pred and real) are minimized across the entire dataset:

$$x^\star = \arg\min_{x\in\mathbb{R}^d} \left\{ \frac{1}{m} \sum_{i=1}^{m} L(h(x;\xi_i), y_i) \right\}$$

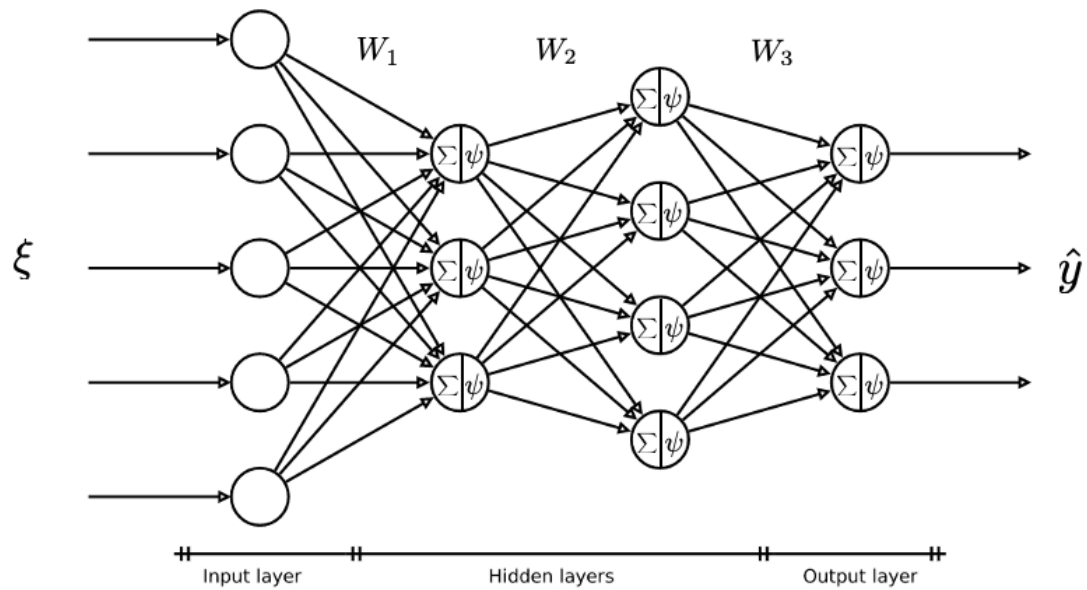
DNN model is notoriously difficult to train

- DNN model $L(h(x;\xi),y)$ is highly non-convex, and probably non-smooth

$$h(x;\xi) = \psi(\cdots\psi(W_2\cdot\psi(W_1\xi+b_1)+b_2)\cdots)$$

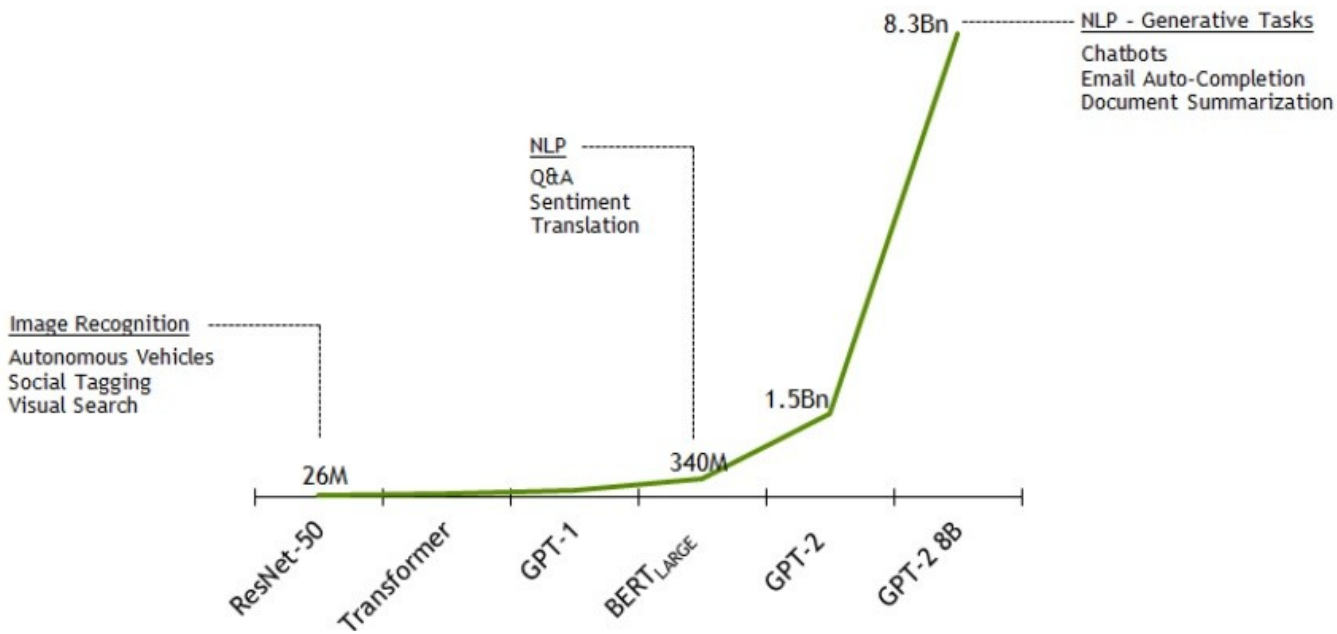
$$L(\hat{y};y) = \frac{1}{2}\|y-\hat{y}\|^2 \text{ or } -y\log(\hat{y}_i) \text{ or others}$$

where $x=\{W_i,b_i\}$ and $\psi(\cdot)$ is a non-linear activation function



DNN model is notoriously difficult to train

- Cannot find global minima; trapped into local minima and saddle points
- The dimension of model parameter $x = \{W_i, b_i\}$ (or model size) is huge[3]



[3] Image source: neowin.net

DNN model is notoriously difficult to train

- Cannot find global minima; trapped into local minima and saddle points
- The dimension of model parameter $x = \{W_j, b_j\}$ (or model size) is huge
- The size of the dataset $\{\xi_i, y_i\}_{i=1}^m$ is huge

DNN Trainig = Non-convexity training + Huge dimension + Huge dataset

Part I: Deep Neural Network (DNN) Training Algorithms

- Sec.1: Deep Neural Network Model
- Sec.2: Stochastic Gradient Descent and Single-Node Training
- Sec.3: Parallel/Distributed Training
- Sec.4: Decentralized Training

DNN model formulated as stochastic optimization

- Recall the DNN training problem

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m} L(h(x;\xi_i), y_i)$$

  - x is the model parameter to train; $\{\xi_i, y_i\}_{i=1}^m$ is the dataset
  - $h(x;\xi)$ is the DNN model; highly non-convex
  - $L(\hat{y}, y)$ is the loss function

- Let $\xi_i := \{\xi_i, y_i\}$ and $F(x;\xi_i) := L(h(x;\xi_i), y_i)$, the problem becomes

$$\min_{x\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m} F(x;\xi_i)$$

  which is a finite-sum empirical risk minimization (ERM) problem.

DNN model formulated as stochastic optimization

- When ξ follows distribution D, DNN training can also be formulated as

$$\min_{x \in \mathbb{R}^d} \; f(x) \quad \text{where} \quad f(x) = \mathbb{E}_{\xi \sim D} F(x; \xi)$$

  which is a stochastic optimization problem.

- ERM is a good approximation to the above problem, especially for large m

- In this lecture, we will focus on the above stochastic problem formulation.

Stochastic gradient descent

- D is unknown; no closed-form for $f(x)$; cannot use gradient descent
- The most popular algorithm is stochastic gradient descent (SGD) (Robbins and Monro, 1951; Bottou, 2010)
- Main idea: sample one (or one batch of) data sample and perform SGD

$$x^{(k+1)} = x^{(k)} - \gamma \nabla F(x^{(k)}; \xi^{(k)})$$

  - $\xi^{(k)}$ is the data sampled at iteration k
  - $\nabla F(x^{(k)}; \xi^{(k)})$ is a stochastic gradient associated with sample $\xi^{(k)}$
  - γ is the learning rate

Why does stochastic gradient descent work?

- If stochastic gradient is unbiased, i.e.,

$$\mathbb{E}_{\xi\sim D}\nabla F(x^{(k)};\xi) = \nabla\,\mathbb{E}_{\xi\sim D}[F(x^{(k)};\xi)] = \nabla f(x^{(k)}),$$

  the SGD recursion in expectation becomes

$$\begin{aligned}\mathbb{E}[x^{(k+1)}] &= \mathbb{E}[x^{(k)}] - \gamma\mathbb{E}[\nabla F(x^{(k)};\xi)]\\ &= \mathbb{E}[x^{(k)}] - \gamma\nabla f(x^{(k)}),\end{aligned}$$

  which reduces to a deterministic gradient descent.

- In other words, SGD is equivalent to GD in expectation. This intuitively explains why SGD works.

Stochastic gradient descent: convergence

Assumption

(A1) The loss function $F(x;\xi)$ is L-smooth in terms of x;

(A2) The stochastic gradient is unbiased, and has bounded variance σ^2.

Theorem

Under the above assumptions, and let $\gamma = O(1/\sqrt{T})$, we have

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{T}}\right)$$

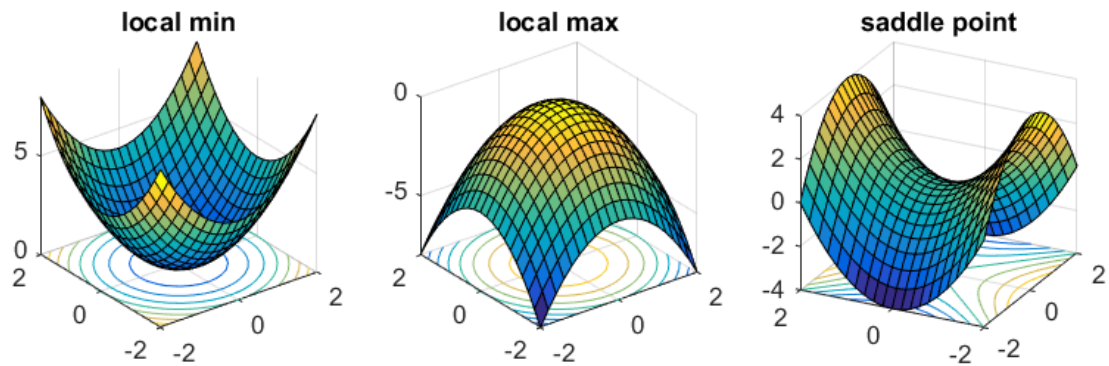
where $T \geq 1$ is the number of iterations

Note that we do no assume convexity for $f(x)$.

Stochastic gradient descent: convergence

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{T}}\right)$$

- When iteration $T \to \infty$, it holds that $\mathbb{E}\|\nabla f(x^{(k)})\|^2 \to 0$
- $\mathbb{E}\|\nabla f(x^{(k)})\|^2 \to 0$ implies SGD converges to a stationary solution
- A stationary solution can be local min, local max, or saddle point[4]

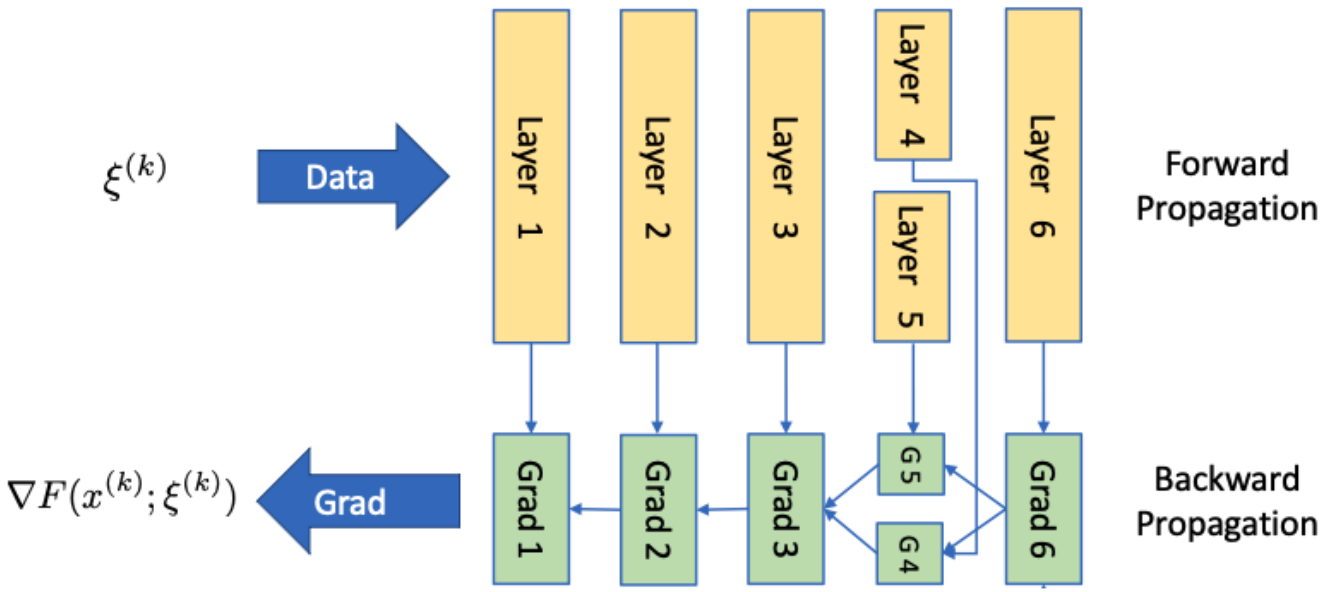


[4] Image source: from Prof. Rong Ge's online post

Stochastic gradient descent: convergence

- Generally speaking, approaching the stationary solution is the best result we can get for SGD; no guarantee to approach the global minimum
- Empirically, SGD performs extremely well when training DNN
- Recent advanced studies show SGD can escape local maximum, saddle point, and even "sharp" local minimum, see, e.g., (Ge et al., 2015; Sun et al., 2015; Jin et al., 2017; Du et al., 2018, 2019; Kleinberg et al., 2018) and references therein
- SGD can even find global minimum under certain conditions, e.g. the PL condition (Karimi et al., 2016)
- However, we will skip these exciting results in this lecture

Implementing SGD in DNN training



- Stochastic gradient can be calculated via forward-backward propagation
- Stochastic gradient can be achieved automatically via Pytorch/Tensorflow
- DNN training typically utilizes GPUs
- Momentum-SGD/ADAM are very useful to accelerate DNN training

Image Classification

- Cifar-10 dataset
- 50K training images
- 10K test images
- DNN model: ResNet-18
- GPU: Tesla V100

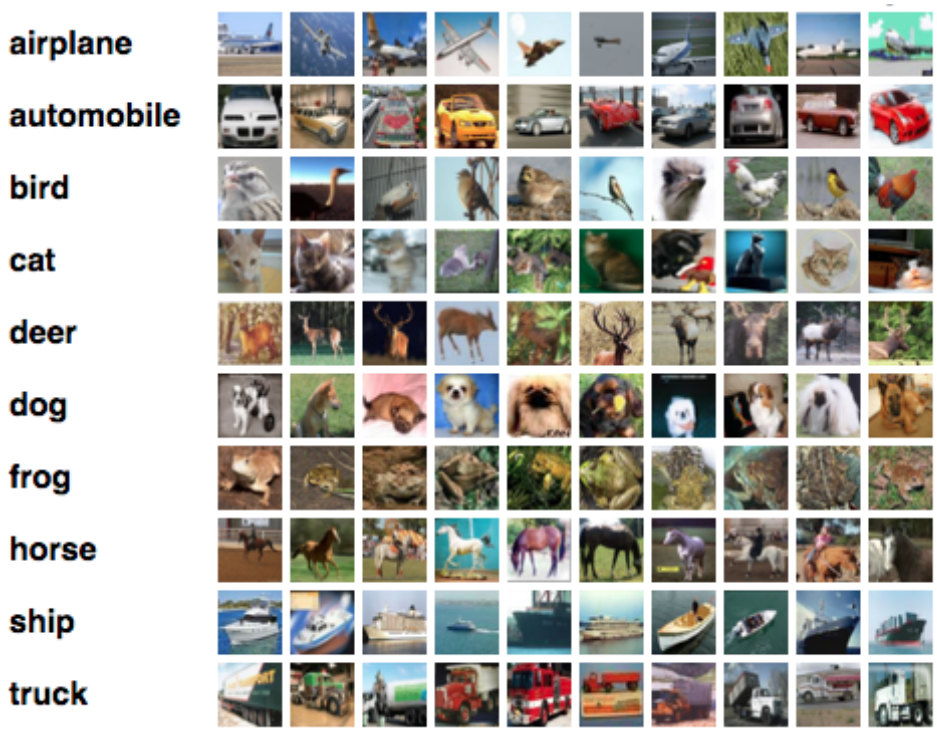
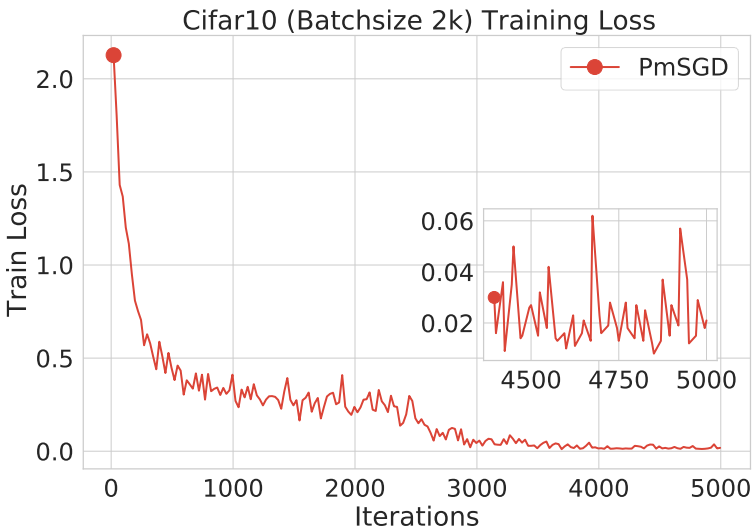
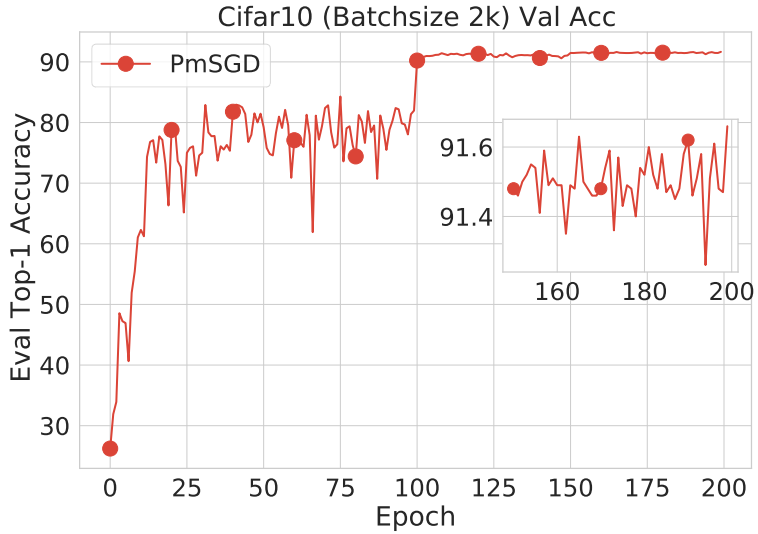


Image Classification





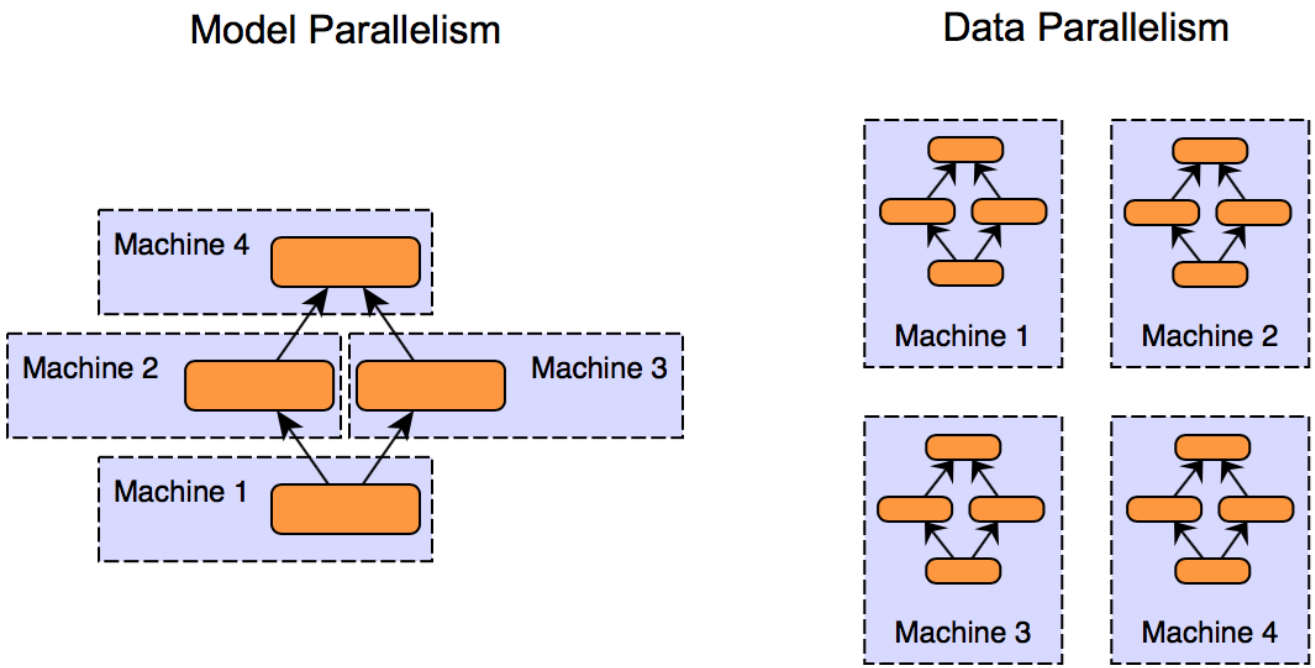
Part I: Deep Neural Network (DNN) Training Algorithms

- Sec.1: Deep Neural Network Model
- Sec.2: Stochastic Gradient Descent and Single-Node Training
- Sec.3: Parallel/Distributed Training
- Sec.4: Decentralized Training

Parallel/Distributed training is necessary in DNN

- Scale to larger models and bigger data
- Bring down training time from days to hours
- Different types of parallel training:
  - Data-parallel training: share the model; partition the data
  - Model-parallel training: share the data; partition the model
  - Data-parallel and model-parallel mixed training
- In this lecture, we will focus on data-parallel training

Data-parallel and model-parallel training[5]



[5] Image source: https://xiandong79.github.io/Intro-Distributed-Deep-Learning

DNN training formulated as distributed optimization

- A network of n nodes (GPUs) collaborate to solve the problem:

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} [f_i(x) = \mathbb{E}_{\xi_i \sim D_i} F(x; \xi_i)].$$

- Each component $f_i : \mathbb{R}^d \to \mathbb{R}$ is local and private to node i
- Random variable ξ_i denotes the local data that follows distribution D_i
- Each local distribution D_i may be different; data heterogeneity

DNN training formulated as distributed optimization

- We consider deep training within high-performance **data-center** clusters
  - all GPUs are connected with high-bandwidth channels
  - network topology can be fully controlled
  - communication is highly reliable; no occasional link failure
- Different from the mobile AI applications, or Federated Learning where
  - nodes are connected with low-bandwidth channels
  - network topology can not be controlled
  - communication is highly fragile; occasional link failures

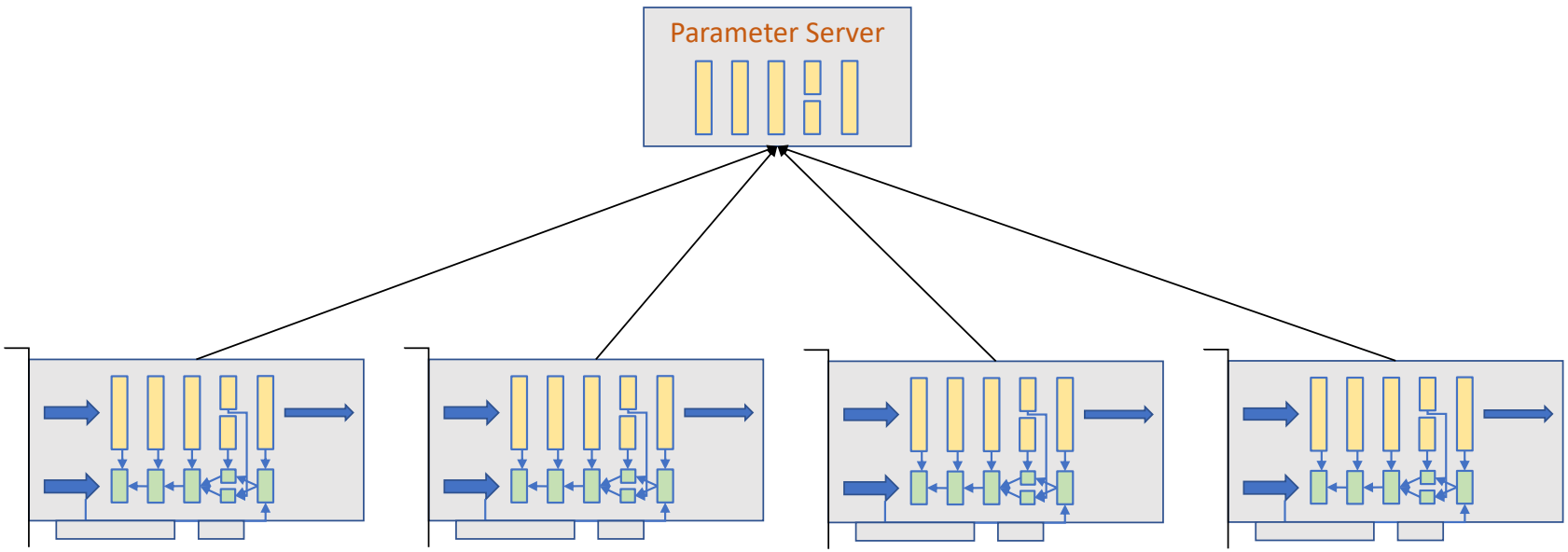
Parallel stochastic gradient descent (SGD)

$$g_i^{(k)} = \nabla F(x^{(k)}; \xi_i^{(k)}) \qquad \text{(Local compt.)}$$

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^{n} g_i^{(k)} \qquad \text{(Global comm.)}$$

- Each node i samples data $\xi_i^{(k)}$ and computes gradient $\nabla F(x^{(k)}; \xi_i^{(k)})$
- All nodes synchronize (i.e. global averaged) to update model x
- Global average incurs significant comm. cost; hinders training scalability

Global average via Parameter-Server (Li et al., 2014)



Global average via Ring-Allreduce (Patarasuk and Yuan, 2009)

Parallel SGD convergence

Assumption

(A1) Each local loss function $F(x;\xi_i)$ is L-smooth in terms of x;

(A2) Each local stochastic gradient is unbiased, and has bounded variance σ^2:

$$\mathbb{E}[g_i^{(k)}] = \nabla f_i(x^{(k)}), \quad \mathbb{E}\|g_i^{(k)} - \nabla f_i(x^{(k)})\|^2 \leq \sigma^2$$

(A3) Each local stochastic gradient $g_i^{(k)}$ is independent of each other

The variance of the globally averaged gradient is remarkably reduced:

$$\mathbb{E}\|\frac{1}{n}\sum_{i=1}^{n} g_i^{(k)} - \nabla f(x^{(k)})\|^2 = \frac{1}{n^2}\sum_{i=1}^{n} \mathbb{E}\|g_i^{(k)} - \nabla f_i(x^{(k)})\|^2 \leq \frac{\sigma^2}{n}$$

Parallel SGD (P-SGD) convergence

- Substituting the above inequality into the derivation, we achieve

Theorem (Parallel SGD convergence)

Under the above assumptions, and let $\gamma = O(1/\sqrt{T})$*, we have*

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}}\right)$$

where $T \geq 1$ *is the number of iterations,* n *is the number of nodes.*

- We achieve single-node SGD convergence when $n = 1$

Parallel SGD can achieve linear speedup

- Recall the SGD convergence rate:

$$\text{Single-node training:} \quad \frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{T}}\right)$$

$$n\text{-node parallel training:} \quad \frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}}\right)$$

- To achieve an ϵ-accurate solution, i.e., $\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x^{(k)})\|^2 \leq \epsilon$,

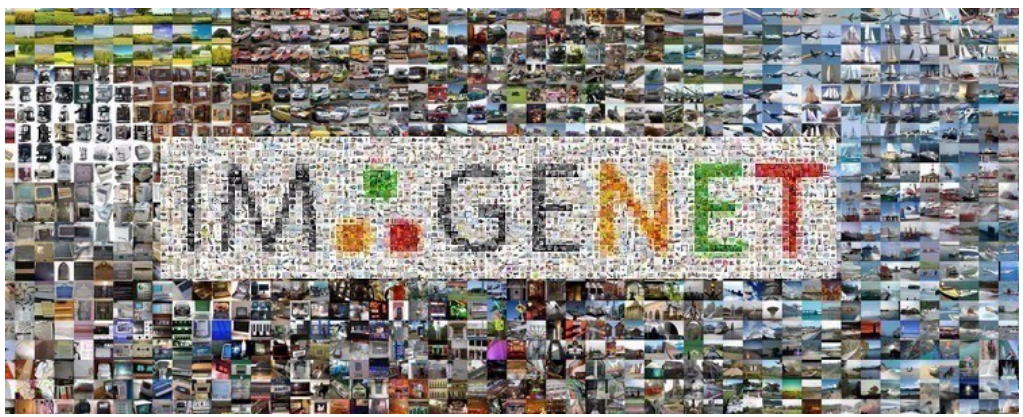
$$\text{Single-node training requires} \quad \frac{\sigma^2}{\epsilon^2} \quad \text{iterations}$$

$$n\text{-node parallel training requires} \quad \frac{\sigma^2}{n\epsilon^2} \quad \text{iterations}$$

- Iteration complexity is inversely proportional to n; P-SGD has linear speedup

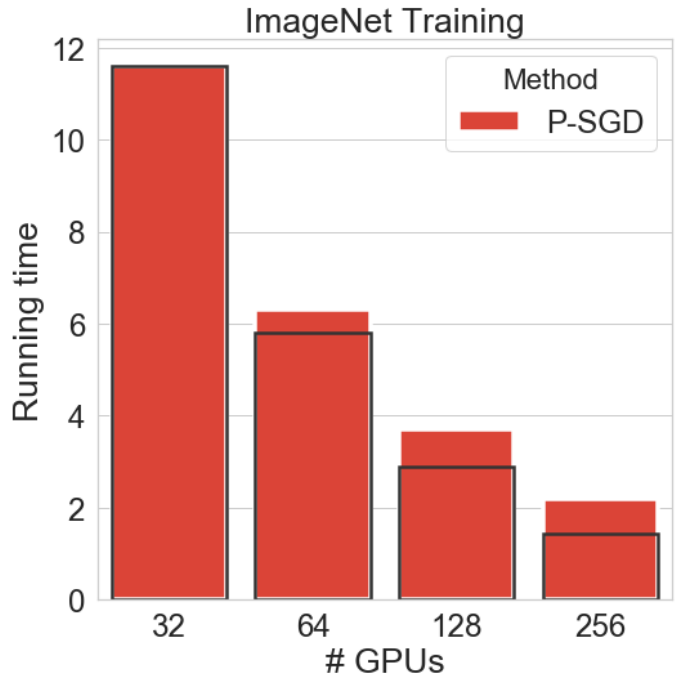
Image Classification

- ImageNet-1K dataset
- 1.3M training images
- 50K test images
- 1K classes
- DNN Model: ResNet-50 (∼25.5M parameters)
- GPU: Tesla V100 clusters
- Framework: Pytorch DDP



Parallel SGD has linear speedup in DNN training

- Wall-clock training time to achieve $> 76\%$ top-1 accuracy (black box indicates ideal running time linear speedup)



- Cannot achieve ideal linear speedup due to comm. cost
- Global average incurs significant comm. cost; hinders training scalability

Comm. overhead in global average

- A single communication includes **bandwidth cost** and **latency** (Ben-Nun and Hoefler, 2019)
- The single communication cost

	Bandwidth Cost	Latency	Total Cost
Parameter server	$\Omega(n)$	$\Omega(1)$	$\Omega(n+1)$
Ring allreduce	$\Omega(1)$	$\Omega(n)$	$\Omega(1+n)$

- In either approach, the cost is $\Omega(n)$, proportional to network size n.
- In deep training, the bandwidth cost is typically more severe; but latency cannot be ignored neither
- To approach the ideal linear speedup, comm. cost must be reduced

Approaches to saving communication cost

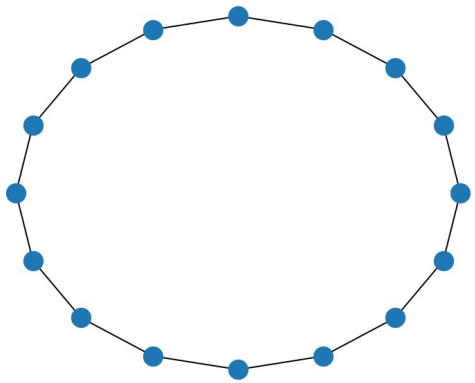
- Model/Gradient sparsification (Tang et al., 2019; Koloskova et al., 2019a,b; Wangni et al., 2017; Alistarh et al., 2018; Stich et al., 2018)
- Model/Gradient quantization (Das et al., 2018; Alistarh et al., 2017; Bernstein et al., 2018; Wen et al., 2017)
- Local SGD/lazy-communication (Chen et al., 2018; Liu et al., 2019; Chen et al., 2020; Zinkevich et al., 2010; Zhang et al., 2016; Stich, 2019; Yu et al., 2019a,b; Lin et al., 2018; McMahan et al., 2017; Li et al., 2019a)
- Decentralized communication (Lopes and Sayed, 2008; Nedic and Ozdaglar, 2009; Shi et al., 2015; Yuan et al., 2016; Assran et al., 2019; Yuan et al., 2019; Li et al., 2019b; Di Lorenzo and Scutari, 2016; Nedic et al., 2017; Qu and Li, 2018)

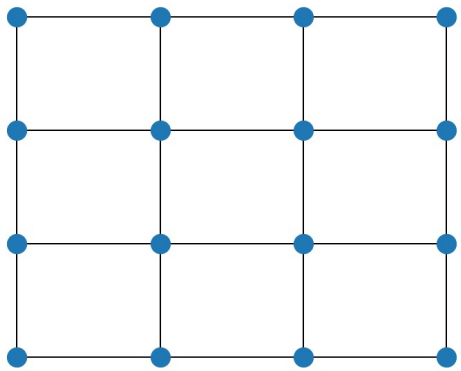
Part I: Deep Neural Network (DNN) Training Algorithms

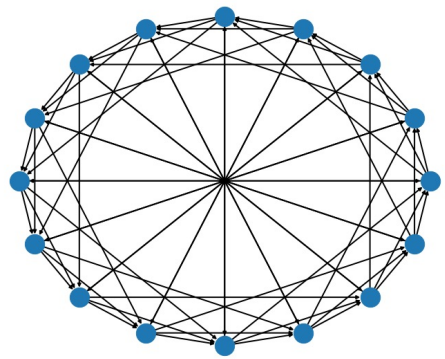
- Sec.1: Deep Neural Network Model
- Sec.2: Stochastic Gradient Descent and Single-Node Training
- Sec.3: Parallel/Distributed Training
- Sec.4: Decentralized Training

Decentralized SGD: topology

- Assume we connect all nodes with some topology (n=16)







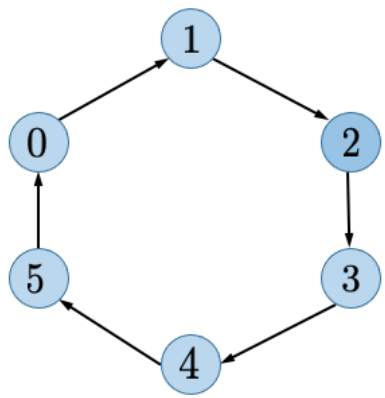
- Communication is only allowed between neighbors
- No global synchronization is allowed

Decentralized SGD: weight matrix

- The weight matrix associated with the topology is defined as

$$w_{ij} \begin{cases} > 0 & \text{if node } j \text{ is connected to } i, \text{ or } i = j; \\ = 0 & \text{otherwise.} \end{cases}$$

- Throughout the lecture we assume the row and column sums of W to be 1
- An example:



$$W = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

Figure: A directed ring topology and its associated combination matrix W.

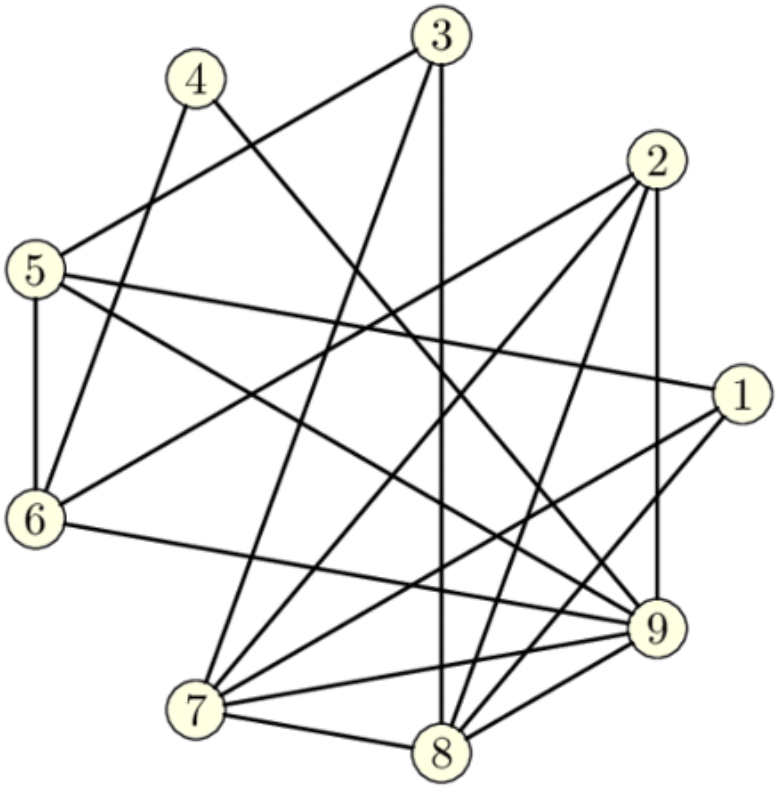
Decentralized SGD (D-SGD): partial averaging

- D-SGD is based on partial-averaging within neighborhood

$$\text{Partial averaging:} \quad x_i^+ \leftarrow \sum_{j \in \mathcal{N}_i} w_{ij} x_j. \quad \forall i \in [n]$$

- $\mathcal{N}_i$ is the set of neighbors of node i
- Each node only communicates with neighbors; no global sync
- Incurs $\Omega(d_{\max})$ comm. overhead ($d_{\max}$: maximum degree)

Maximum degree[6]



$$d_1 = 3$$
$$d_2 = 4$$
$$d_3 = 3$$
$$\vdots$$
$$d_9 = 6$$

$$d_{\max} = \max_i \{d_i\} = 6$$

[6] Image source:

Decentralized SGD (D-SGD): recursions

- D-SGD = local SGD update+ paritial averaging (Loizou and Richtárik, 2020; Nedic and Ozdaglar, 2009; Chen and Sayed, 2012)

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) \quad \text{(Local update)}$$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k+\frac{1}{2})} \quad \text{(Partial averaging)}$$

- Per-iteration communication: $\Omega(d_{\max}) \ll \Omega(n)$ when topology is sparse
- Incurs $\Omega(1)$ comm. overhead on sparse topology (ring or grid)

Decentralized SGD is more communication efficient

Model	Ring-Allreduce	Partial average
ResNet-50	278 ms	150 ms
Bert	1469 ms	567 ms

Table: Comparison of per-iter comm. in terms of runtime with 256 GPUs

- ResNet-50 has 25.5M parameters; Bert has 300M parameters
- Partial average saves more communication for larger model

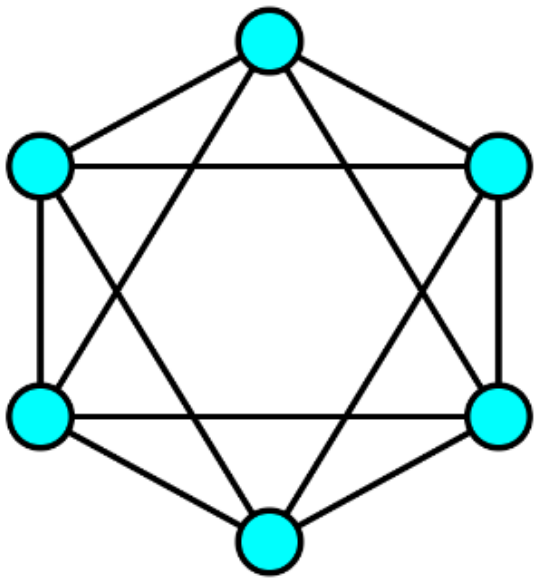
However, D-SGD has slower convergence

- The efficient communication comes with a cost: slow convergence
- Partial averaging is less effective to aggregate information
- The average effectiveness can be evaluated by spectral gap:

$$\rho = \|W - \frac{1}{n}\mathbb{1}\mathbb{1}^T\|_2$$

  - Assume W is doubly-stochastic, it holds that $\rho \in (0, 1)$.
  - Well-connected topology has $\rho \to 0$, e.g. fully-connected topology
  - Sparsely-connected topology has $\rho \to 1$, e.g., ring has $\rho = O(1 - \frac{1}{n^2})$

Weight-matrix of the fully-connected topology



$$W = \frac{1}{5}\mathbb{1}\mathbb{1}^T = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

Decentralized SGD convergence

Recall the assumptions of P-SGD:

Assumption

(A1) Each local loss function $F(x;\xi_i)$ is L-smooth in terms of x;

(A2) Each local stochastic gradient is unbiased, and has bounded variance σ^2:

$$\mathbb{E}[g_i^{(k)}] = \nabla f_i(x^{(k)}), \quad \mathbb{E}\|g_i^{(k)} - \nabla f_i(x^{(k)})\|^2 \leq \sigma^2$$

(A3) Each local stochastic gradient $g_i^{(k)}$ is independent of each other

We further introduce another data-heterogeneity assumption

Assumption

(A4) The data heterogeneity is bounded, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x) - \nabla f(x)\|^2 \leq b^2, \quad \forall x \in \mathbb{R}^d$$

When D_i is identical, we have $\nabla f_i(x) = \nabla f(x)$ for any i and hence $b^2 = 0$

Decentralized SGD convergence

- (Lian et al., 2017; Assran et al., 2019; Koloskova et al., 2020) show that

Theorem (Decentralized SGD convergence)

Under Assumptions (A1)-(A4), and let $\gamma = O(1/\sqrt{T})$*, we have*

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}\|\nabla f(x^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}} + \frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}} + \frac{\rho^{2/3}b^{2/3}}{T^{2/3}(1-\rho)^{2/3}}\right)$$

where $T \geq 1$ *is the number of iterations, and* n *is the number of nodes.*

- When topology is fully connected ($\rho = 0$), D-SGD reduces to P-SGD.
- When $\rho = 0$ and $n = 1$, D-SGD reduces to single-node SGD

Convergence rate: P-SGD v.s. D-SGD

- Convergence comparison (i.i.d data distribution, i.e., $b^2 = 0$):

$$\text{P-SGD}: \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\Big(\frac{\sigma}{\sqrt{nT}}\Big)$$

$$\text{D-SGD}: \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\Big(\frac{\sigma}{\sqrt{nT}} + \underbrace{\frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}}}_{\text{extra overhead}}\Big)$$

  where σ^2 is the gradient noise, and T is the number of iterations.

- D-SGD can asymptotically converge as fast as P-SGD when $T \to \infty$; the first term dominates; reach linear speedup asymptotically

- But it requires more iteration (i.e., T has to be large enough) to reach that stage due to the extra overhead caused by partial averaging

Transient iterations

- **Definition** (Pu et al., 2020): number of iterations before D-SGD achieves linear speedup
- Transient iterations measure the converg. gap between P-SGD and D-SGD
- Longer tran. iters. $\Longrightarrow$ slower convergence than P-SGD
- The transient iteration complexity of D-SGD is

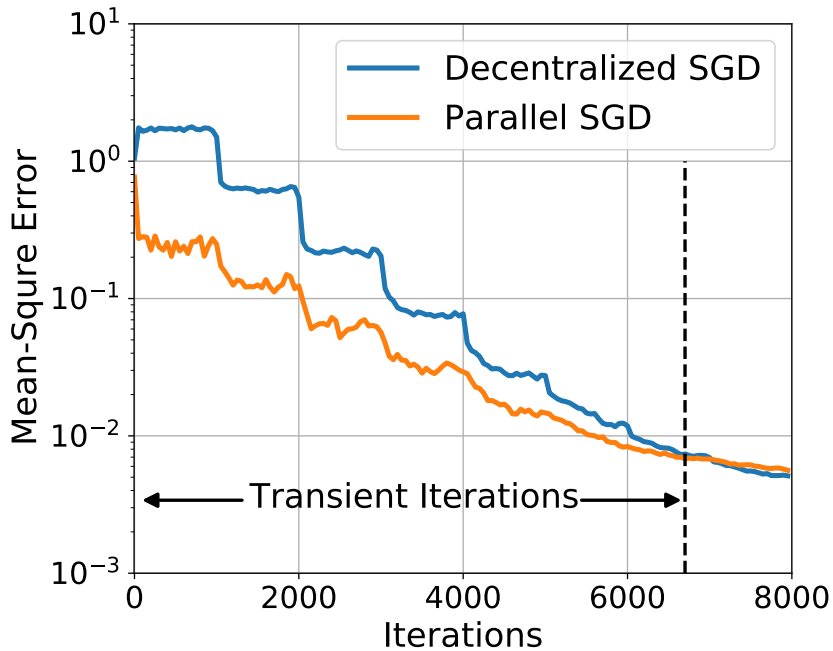
$$\text{iid data}: \quad \frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}} \leq \frac{\sigma}{\sqrt{nT}} \quad \Longrightarrow \quad T = \Omega\left(\frac{\rho^4 n^3}{(1-\rho)^2}\right)$$

$$\text{non-iid data}: \quad \frac{\rho^{2/3}b^{2/3}}{T^{2/3}(1-\rho)^{2/3}} \leq \frac{\sigma}{\sqrt{nT}} \quad \Longrightarrow \quad T = \Omega\left(\frac{\rho^4 n^3}{(1-\rho)^4}\right)$$

- Sparse topology ($\rho \to 1$) incurs large tran. iters. complexity

Transient iterations: illustration

Illustration of the tran. iters. on D-SGD over ring (logistic regression)



If the transient stage is too long, we may not be able to achieve linear speedup given the limited time/resource budget

Part I summary

- DNN training can be formulated as stochastic optimization
- SGD is the leading approach to train DNN
- Parallel SGD can achieve linear speedup theoretically; but the comm. cost incurred by global average hinders its empirical linear speedup performance
- Decentralized SGD utilizes partial averaging within neighborhood; reduce per-iter comm. cost from $\Omega(n)$ to $\Omega(d_{\max})$, and even $\Omega(1)$.
- D-SGD suffers from slower convergence; compensate its comm. efficiency.

In Part II, we will

Introduce several techniques to accelerate D-SGD and make it practically valuable for large-scale deep learning

References I

H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points—online stochastic gradient for tensor decomposition," in *Conference on learning theory*. PMLR, 2015, pp. 797–842.

J. Sun, Q. Qu, and J. Wright, "When are nonconvex problems not scary?" *arXiv preprint arXiv:1510.06096*, 2015.

C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1724–1732.

S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," *arXiv preprint arXiv:1810.02054*, 2018.

References II

S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1675–1685.

B. Kleinberg, Y. Li, and Y. Yuan, "An alternative view: When does sgd escape local minima?" in *International Conference on Machine Learning*. PMLR, 2018, pp. 2698–2707.

H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 795–811.

M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 583–598.

P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *Journal of Parallel and Distributed Computing*, vol. 69, no. 2, pp. 117–124, 2009.

References III

T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–43, 2019.

H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6155–6165.

A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *International Conference on Machine Learning*, 2019, pp. 3478–3487.

A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *International Conference on Learning Representations*, 2019.

J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," *arXiv preprint arXiv:1710.09854*, 2017.

References IV

D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," *arXiv preprint arXiv:1809.10505*, 2018.

S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *arXiv preprint arXiv:1809.07599*, 2018.

D. Das, N. Mellempudi, D. Mudigere, D. Kalamkar, S. Avancha, K. Banerjee, S. Sridharan, K. Vaidyanathan, B. Kaul, E. Georganas *et al.*, "Mixed precision training of convolutional neural networks using integer operations," *arXiv preprint arXiv:1802.00930*, 2018.

D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signsgd with majority vote is communication efficient and fault tolerant," *arXiv preprint arXiv:1810.05291*, 2018.

References V

W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *arXiv preprint arXiv:1705.07878*, 2017.

T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 5050–5060.

Y. Liu, W. Xu, G. Wu, Z. Tian, and Q. Ling, "Communication-censored admm for decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2565–2579, 2019.

T. Chen, Y. Sun, and W. Yin, "Lasg: Lazily aggregated stochastic gradients for communication-efficient distributed learning," *arXiv preprint arXiv:2002.11360*, 2020.

M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603.

J. Zhang, C. De Sa, I. Mitliagkas, and C. Ré, "Parallel sgd: When does averaging help?" *arXiv preprint arXiv:1606.07365*, 2016.

References VI

S. U. Stich, "Local sgd converges fast and communicates little," in *International Conference on Learning Representations (ICLR)*, 2019.

H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7184–7193.

H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5693–5700.

T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local sgd," *arXiv preprint arXiv:1808.07217*, 2018.

B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

References VII

X. Li, W. Yang, S. Wang, and Z. Zhang, "Communication efficient decentralized training with multiple local updates," *arXiv preprint arXiv:1910.09126*, 2019.

C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

References VIII

M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *International Conference on Machine Learning (ICML)*, 2019, pp. 344–353.

K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact dffusion for distributed optimization and learning – Part I: Algorithm development," *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 708 – 723, 2019.

Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Transactions on Signal Processing*, July 2019, early acces. Also available on arXiv:1704.07807.

P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

References IX

G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2018.

N. Loizou and P. Richtárik, "Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods," *Computational Optimization and Applications*, vol. 77, no. 3, pp. 653–710, 2020.

J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012.

X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. U. Stich, "A unified theory of decentralized sgd with changing topology and local updates," in *International Conference on Machine Learning (ICML)*, 2020, pp. 1–12.

References X

S. Pu, A. Olshevsky, and I. C. Paschalidis, "Asymptotic network independence in distributed stochastic optimization for machine learning: Examining distributed and centralized stochastic gradient descent," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 114–122, 2020.