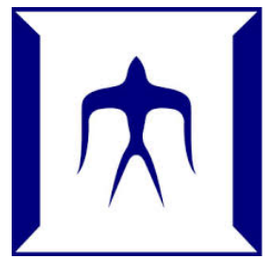
# Optimization Methods for Large Scale Distributed Deep Learning

Tokyo Institute of Technology
Rio Yokota

IPAM Workshop I: Big Data Meets Large-Scale Computing
September 24-28, 2018, UCLA

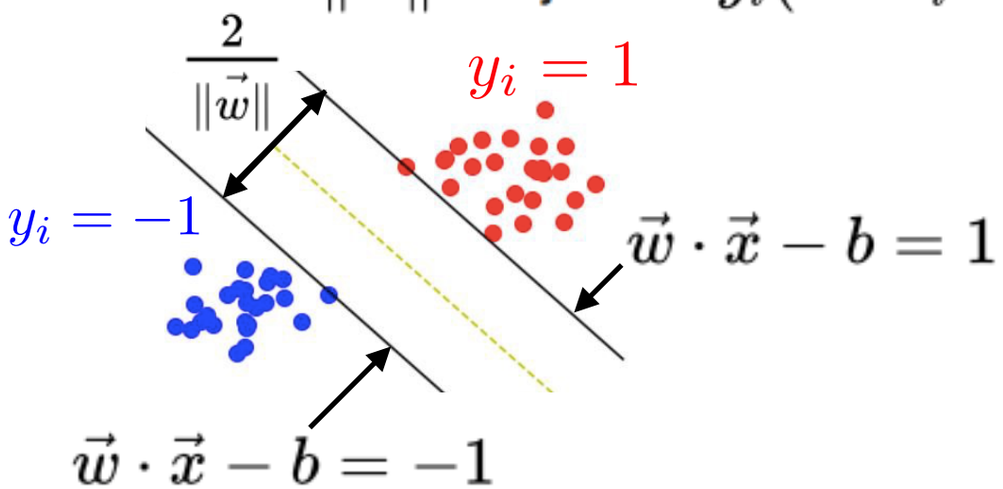



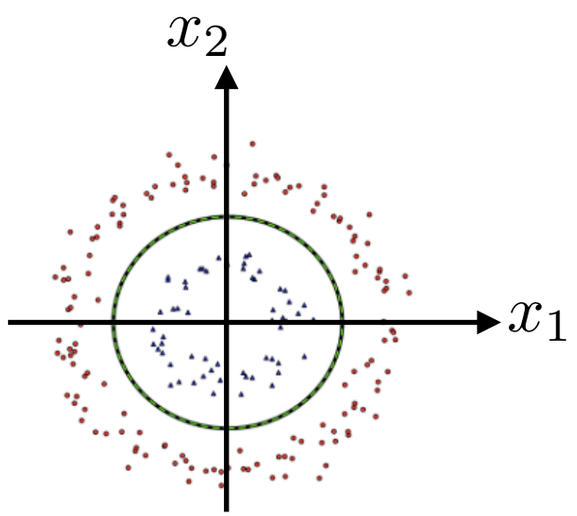
# HLRA for Kernel Methods

Hierarchical Low Rank Approximation

## Support Vector Machines

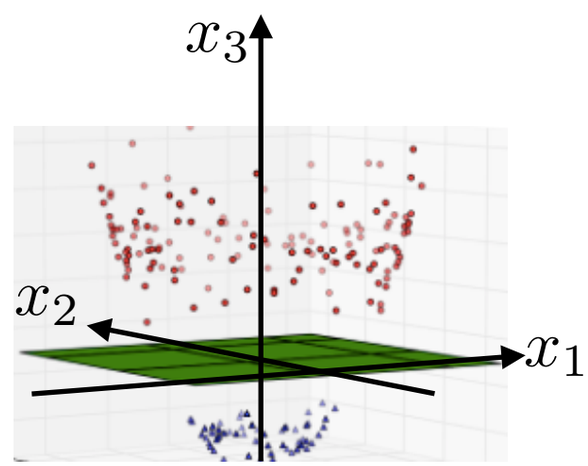
"Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1$, for $i = 1, \ldots, n$"




## Kernel Method

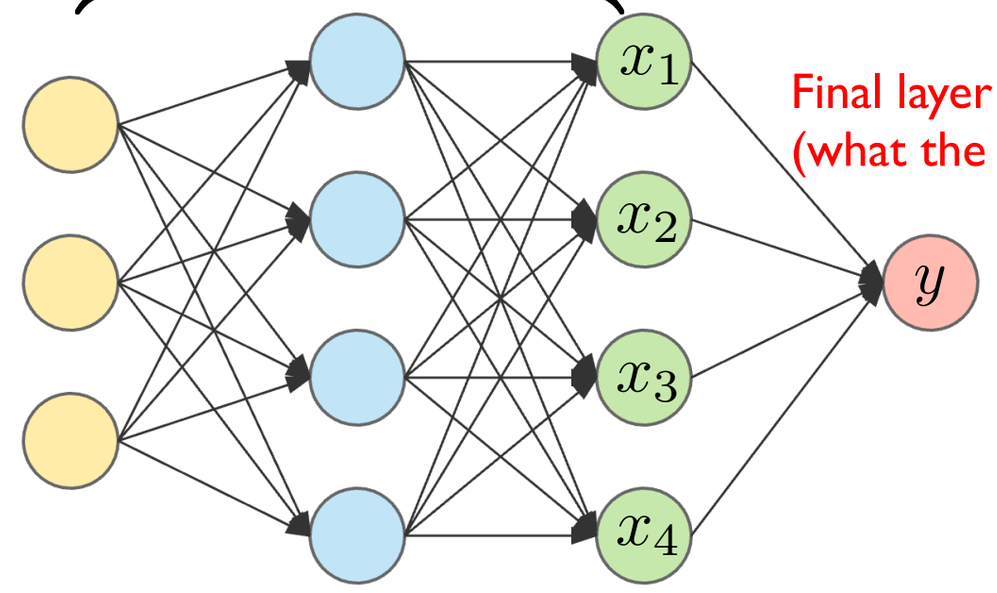
$x_3 = K(\vec{x}) = x_1^2 + x_2^2$



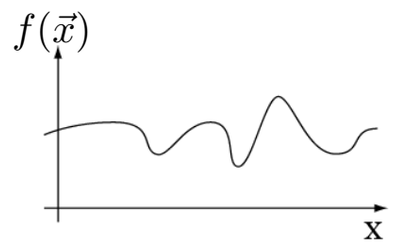

Not linearly separable in the $x_1$,$x_2$ plane, but separable if we add $x_3$

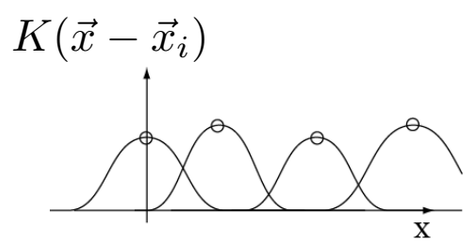
## Deep Neural Networks

$K(\vec{x})$ Learn the kernel/feature vectors (like $x_3$)



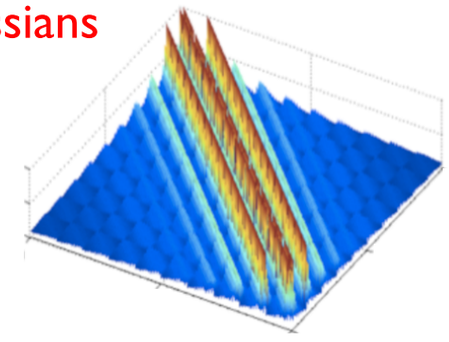

Final layer does the classification (what the SVM is doing)

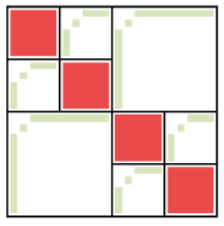



You can express an arbitrary function using a superposition of Gaussians

$$\begin{bmatrix} \ddots & \cdots & \cdots \\ \vdots & K(\vec{x}_i - \vec{x}_j) & \vdots \\ \vdots & \cdots & \ddots \end{bmatrix}$$



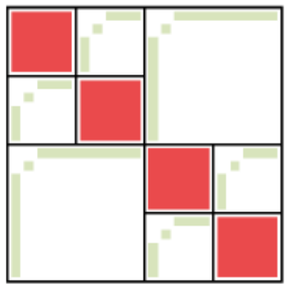


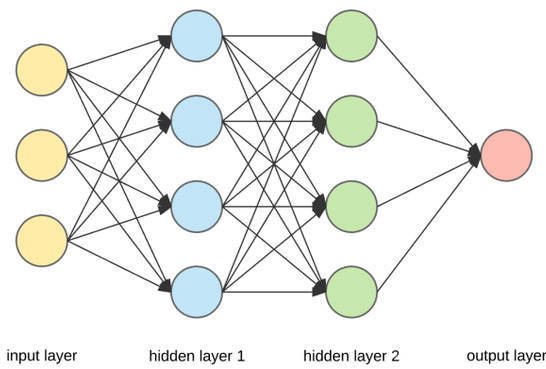
The K matrix has hierarchical low-rank structure

# HLRA for Deep Neural Networks?

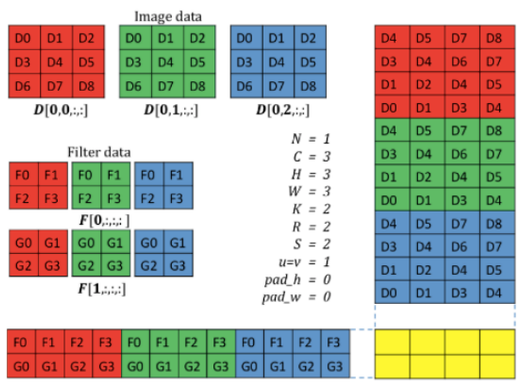
Deep neural networks are
a bunch of dense matrix operations

Dense matrix operations can be
accelerated by HLRA






Tensors in CNNs are turned into
dense matrix multiplications



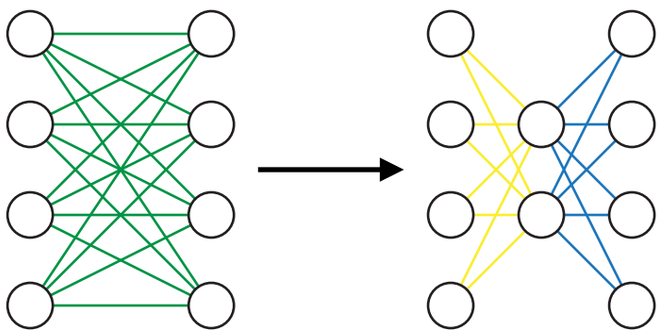

https://arxiv.org/abs/1410.0759

These matrices were way
too tall and skinny

Their numerical rank was
not very small either

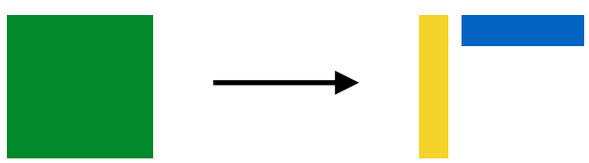
Compression cost can not be amortized
if the matrix keeps changing

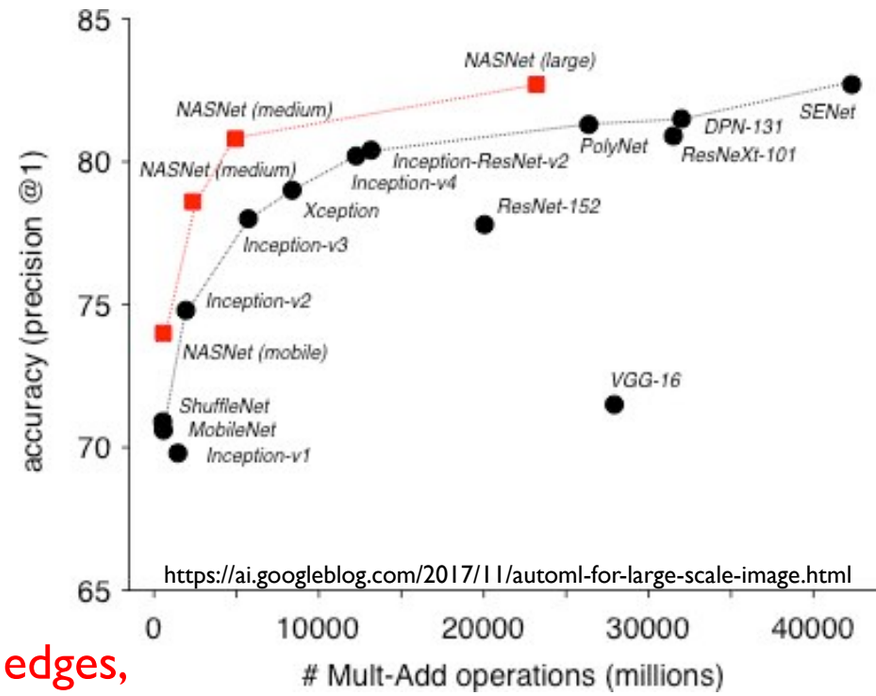
## Can we use low-rank approximation to design/prune the networks?



The objective is not to design an auto-encoder

The compressed network needs to retain
generalization capability and resilience
to adversarial attacks



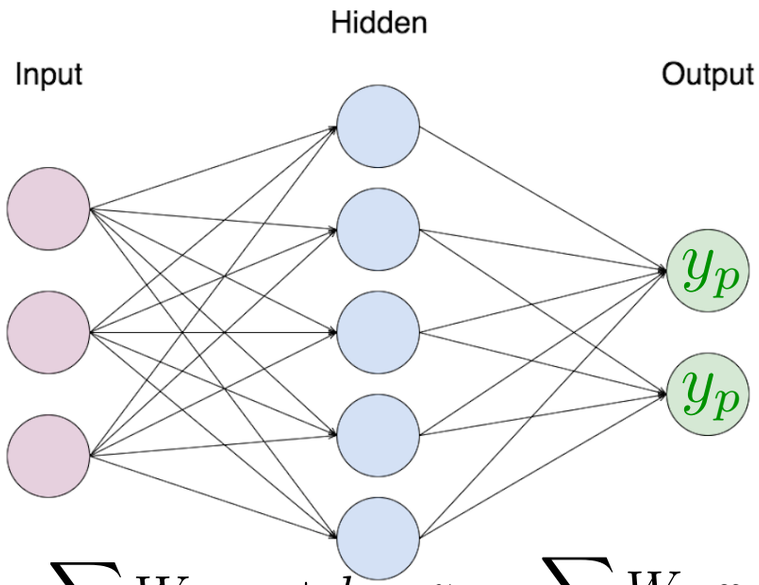



https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html

Modern neural networks are not just nodes and edges,
e.g. residual units in ResNet, memory units in LSTM, skip connections in DenseNet

AutoML can handle complicated units and still find optimal designs

# Second Order Optimization Methods




$y_q$ data (label)

$y_p$ model (output)

1

0.9

0

0.1

$$z_i = \sum_j W_{ij} x_j + b \qquad z_i = \sum_j W_{ij} x_j + b$$

$$x_i = \max(0, z_i)$$

ReLU

$$y_{pi} = \frac{exp(z)}{\sum_i \exp(z_i)}$$

softmax

Cross entropy loss function

$$\begin{aligned} J(\theta) &= E_q(-\log p_\theta(y|\mathbf{x})) \\ &= \sum_{(\mathbf{x},y)} -q(y|\mathbf{x}) \log p_\theta(y|\mathbf{x}) \\ &= \sum_{\mathbf{x}} \{-y_q \log y_p - (1-y_q)\log(1-y_p)\} \end{aligned}$$

Back propagation

$$\frac{\partial J}{\partial W_{ij}} = \frac{\partial J}{\partial y_{pi}} \frac{\partial y_{pi}}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}} \qquad \frac{\partial J}{\partial b} = \frac{\partial J}{\partial y_{pi}} \frac{\partial y_{pi}}{\partial z_i} \frac{\partial z_i}{\partial b}$$

---

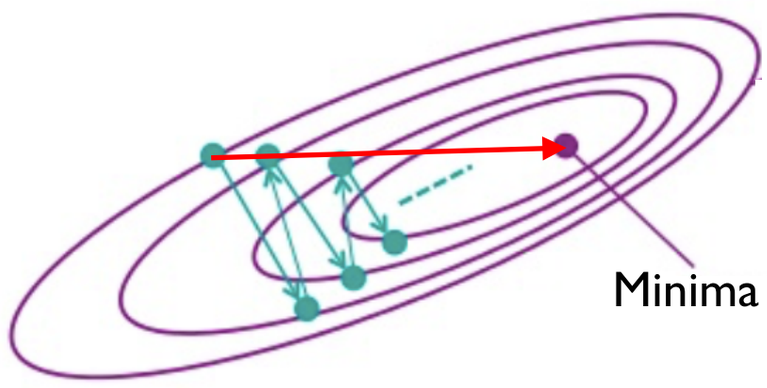
Newton's method

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left( \frac{\partial^2 J}{\partial \boldsymbol{\theta}^2} \right)^{-1} \frac{\partial J}{\partial \boldsymbol{\theta}}$$

Gauss-Newton method

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left( \frac{\partial J}{\partial \boldsymbol{\theta}}^T \frac{\partial J}{\partial \boldsymbol{\theta}} \right)^{-1} \frac{\partial J}{\partial \boldsymbol{\theta}}$$

If the loss function is convex




Gradient descent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial J}{\partial \boldsymbol{\theta}}$$

Mini-batch gradient descent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial J_{batch}}{\partial \boldsymbol{\theta}}$$

Stochastic gradient descent

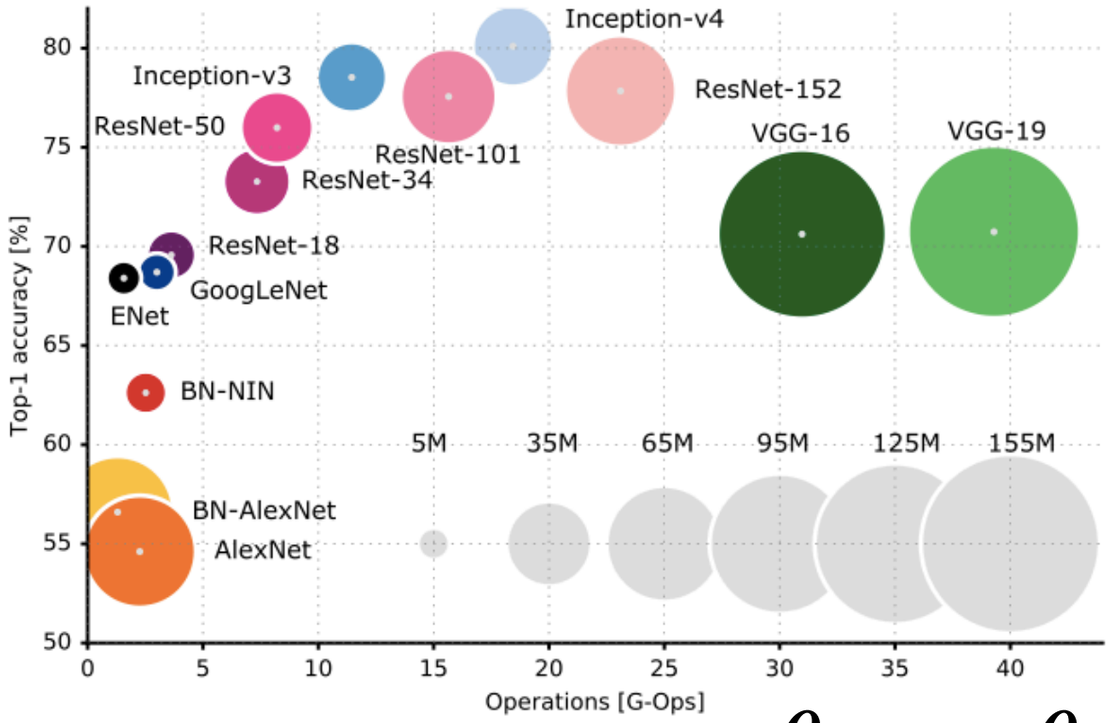
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial J_{single}}{\partial \boldsymbol{\theta}}$$

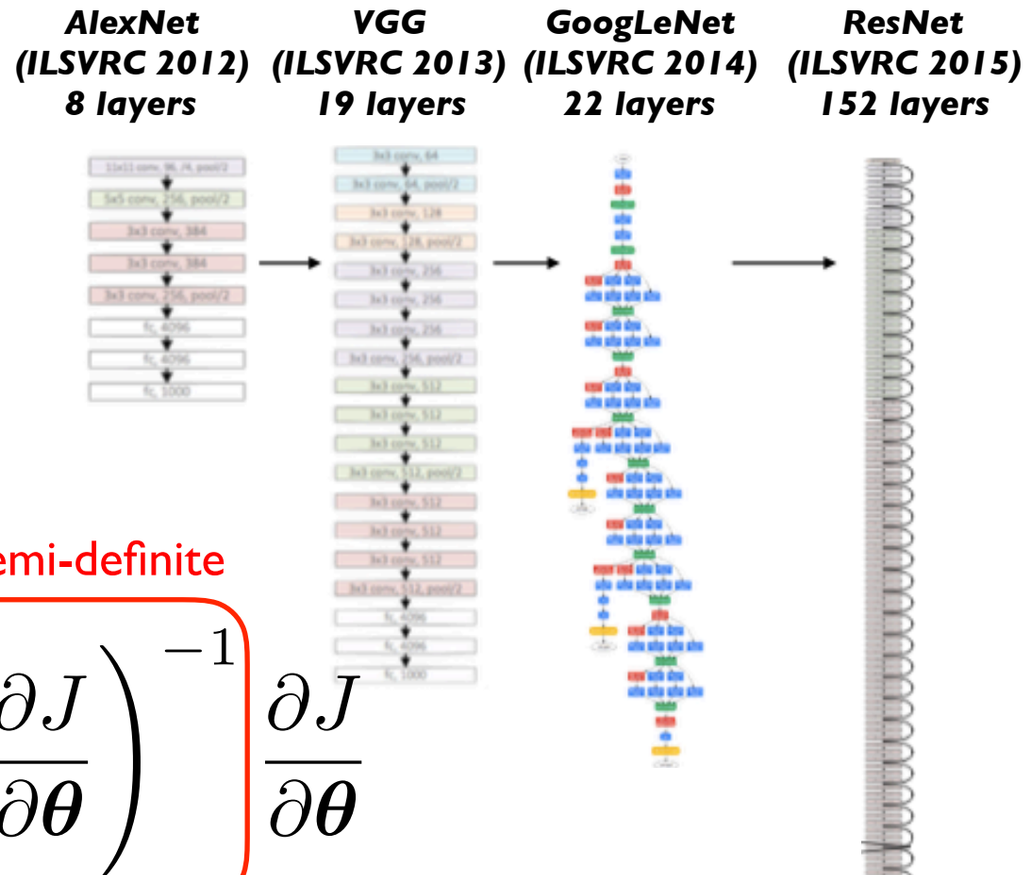
Difference between the Hessian and (empirical) Fisher information matrix

$$\frac{\partial^2 J}{\partial \boldsymbol{\theta}^2} = \frac{\partial J}{\partial \boldsymbol{\theta}}^T \frac{\partial J}{\partial \boldsymbol{\theta}} - E_q \left( \frac{\partial^2 p_\theta}{\partial \theta^2} \frac{1}{p_\theta} \right)$$

$$\lim_{p_\theta \to q} E_q \left( \frac{\partial^2 p_\theta}{\partial \theta^2} \frac{1}{p_\theta} \right) = 0$$

# The Fisher Matrix is a 100Mx100M Dense Matrix




Positive Semi-definite

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left( \frac{\partial J}{\partial \boldsymbol{\theta}}^T \frac{\partial J}{\partial \boldsymbol{\theta}} \right)^{-1} \frac{\partial J}{\partial \boldsymbol{\theta}}$$

<u>Approximate it</u>

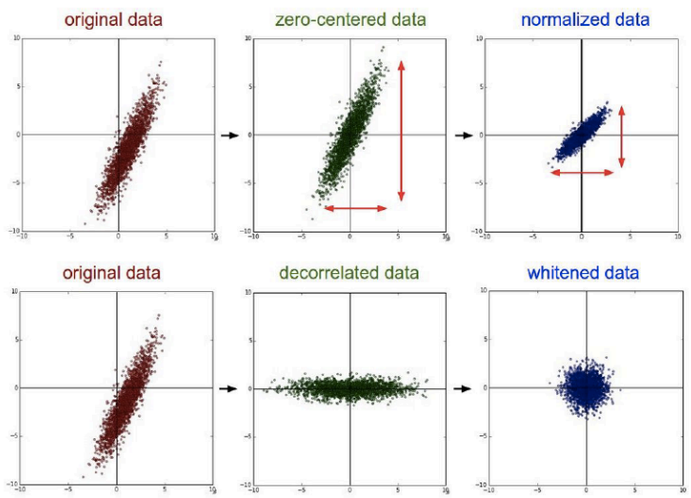
Use gradient from past updates (L-BFGS) [Liu et al. 1989]

Low rank approximation of the whole thing (TONGA) [Le Roux et al. 2012]

Use an iterative solver (Hessian-free) [Martens 2014]

Sparse approximation (FANG) [Grosse 2015]

Kronecker Factorization (K-FAC) [Martens 2016]




<u>Make the off-diagonals weaker (and use diagonal approximation)</u>

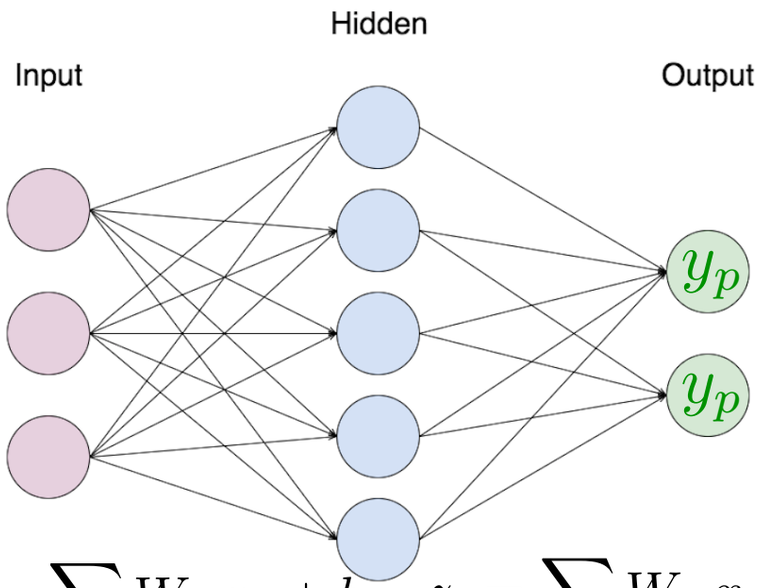
Path-wise data normalization [Neyshabur et al. 2015]

Layer-wise data normalization (PRONG) [Desjardins et al. 2015]

Weight normalization [Salimans et al. 2016]

Low rank weight normalization [Luo 2017]

# Second Order Optimization Methods




$y_q$ data (label)

$y_p$ model (output)

1

0.9

0

0.1

$$z_i = \sum_j W_{ij} x_j + b \qquad z_i = \sum_j W_{ij} x_j + b$$

$$x_i = \max(0, z_i)$$

ReLU

$$y_{pi} = \frac{exp(z)}{\sum_i \exp(z_i)}$$

softmax

Cross entropy loss function

$$\begin{aligned} J(\theta) &= E_q(-\log p_\theta(y|\mathbf{x})) \\ &= \sum_{(\mathbf{x},y)} -q(y|\mathbf{x}) \log p_\theta(y|\mathbf{x}) \\ &= \sum_{\mathbf{x}} \{-y_q \log y_p - (1-y_q)\log(1-y_p)\} \\ &= \sum_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \theta) \end{aligned}$$

Back propagation

$$\frac{\partial J}{\partial W_{ij}} = \frac{\partial J}{\partial y_{pi}} \frac{\partial y_{pi}}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}}$$

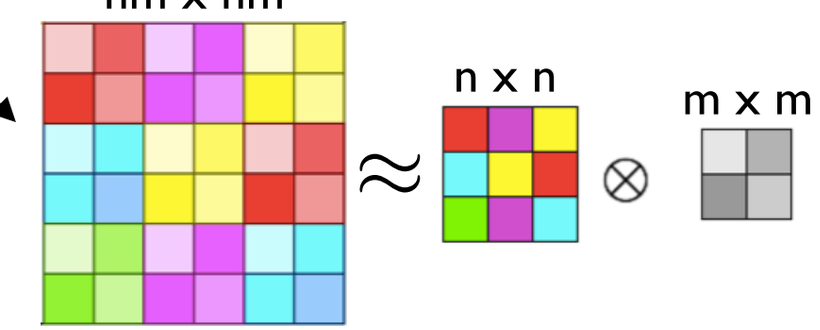
Kronecker Product

$$= \sum_{\mathbf{x}} \frac{\partial \mathcal{L}}{\partial W_{ij}} = \sum_{\mathbf{x}} \frac{\partial \mathcal{L}}{\partial z_i} \frac{z_i}{W_{ij}} = \sum_{\mathbf{x}} g_i a_j$$

Hierarchical Low-rank?

Fisher Matrix

$$\left[ \frac{\partial J}{\partial \boldsymbol{\theta}}^T \frac{\partial J}{\partial \boldsymbol{\theta}} \right]$$




Kronecker Factorization

$$\left( \frac{\partial J}{\partial \boldsymbol{\theta}}^T \frac{\partial J}{\partial \boldsymbol{\theta}} \right)^{-1} = \left\{ \left( \sum_{\mathbf{x}} \mathbf{g} \otimes \mathbf{a} \right)^T \left( \sum_{\mathbf{x}} \mathbf{g} \otimes \mathbf{a} \right) \right\}^{-1}$$

$$\approx \left( \sum_{\mathbf{x}} \mathbf{g}^T \mathbf{g} \right)^{-1} \otimes \left( \sum_{\mathbf{x}} \mathbf{a}^T \mathbf{a} \right)^{-1}$$

$$= G^{-1} \otimes A^{-1}$$

# Regularization

Eigenvalues of the Fisher Matrix

- Extremely long tail
- Very few are very large
- Many are close to 0

$$\left[ \frac{\partial J}{\partial \boldsymbol{\theta}}^T \frac{\partial J}{\partial \boldsymbol{\theta}} \right] + \lambda \mathbf{I}$$

What the inverse Hessian does

- Stops moving in the direction of largest eigenvalues (sharp)
- Moves a lot in the direction of small eigenvalues (flat)
- Regularization keeps the update from becoming too large in flat directions

Cross entropy loss function + L2 regularization

$$\tilde{J}(\theta) = J(\theta) + \frac{1}{2}\lambda ||\boldsymbol{\theta}||_2^2$$

Gradient: Weight decay

$$\frac{\partial \tilde{J}}{\partial \boldsymbol{\theta}} = \frac{\partial J}{\partial \boldsymbol{\theta}} + \lambda \boldsymbol{\theta}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial J}{\partial \boldsymbol{\theta}_t} - \eta \lambda \boldsymbol{\theta}_t$$

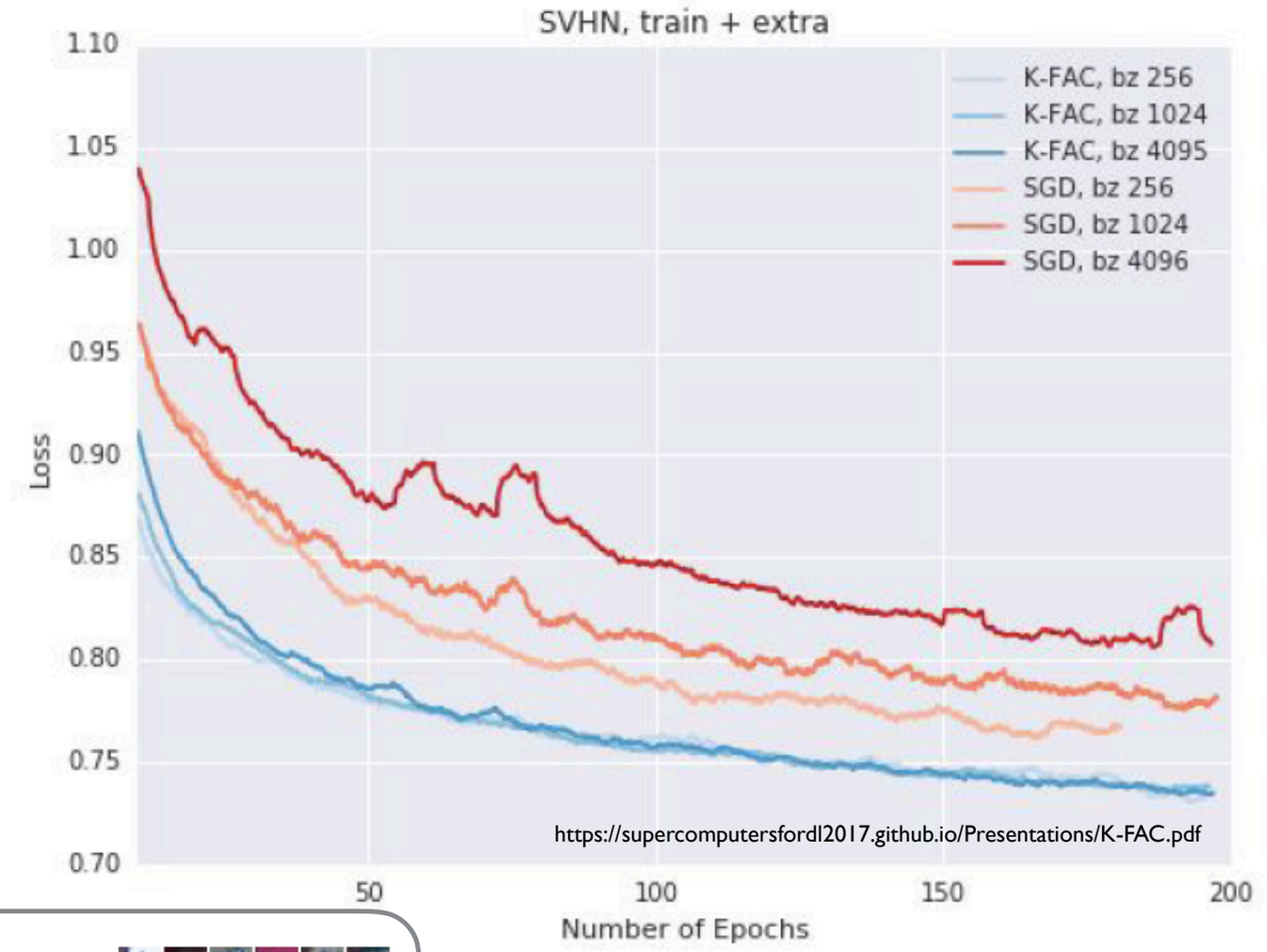
Hessian: Diagonal component

$$\frac{\partial^2 \tilde{J}}{\partial \boldsymbol{\theta}^2} = \frac{\partial^2 J}{\partial \boldsymbol{\theta}^2} + \lambda \mathbf{I}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left( \frac{\partial^2 J}{\partial \boldsymbol{\theta}_t^2} + \lambda \mathbf{I} \right)^{-1} \left( \frac{\partial J}{\partial \boldsymbol{\theta}_t} + \lambda \boldsymbol{\theta}_t \right)$$

# Convergence of K-FAC

**Training .oss**




K-FAC converges at the **same rate**, regardless of batch size!

<u>Data</u>
SVHN (Street View House Numbers)
32 x 32 images
10 digit classes
600,000 examples

<u>Model</u>
ResNet-50
50 Layers
25.5 M parameters
3.8 GFLOPs per inference

Google **Brain**
Google **Research**
DeepMind


James Martens


Roger Grosse


Jimmy Ba


James Keeling


Noah Siegel


Olga Wichrowska


Alok Aggarwal

# Distributed Deep Learning

## Data/model/pipeline parallelism

**Integrated Model, Batch and Domain Parallelism in Training Neural Networks**

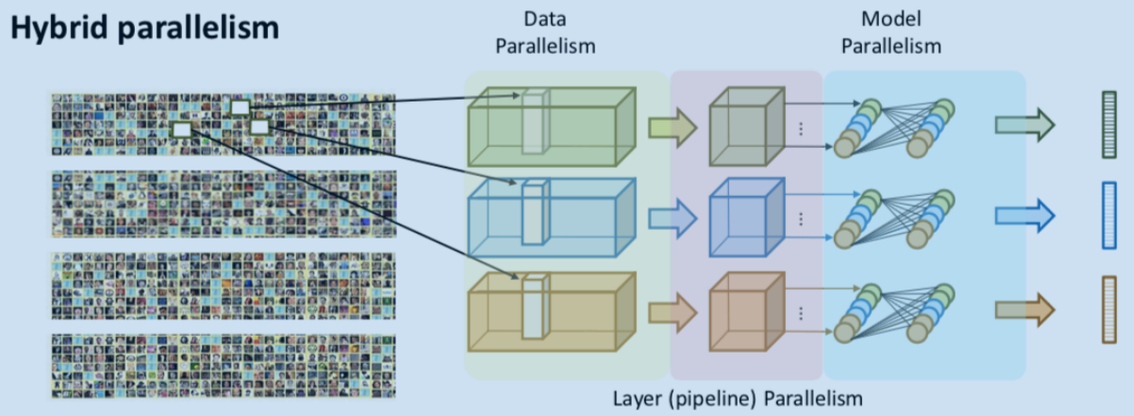

Amir Gholami
EECS Department, UC Berkeley
amirgh@eecs.berkeley.edu

Ariful Azad
CRD, Lawrence Berkeley Lab
azad@lbl.gov

Peter Jin
EECS Department, UC Berkeley
phj@eecs.berkeley.edu

Kurt Keutzer
EECS Department, UC Berkeley
keutzer@eecs.berkeley.edu

Aydın Buluç
CRD, Lawrence Berkeley Lab
abuluc@lbl.gov

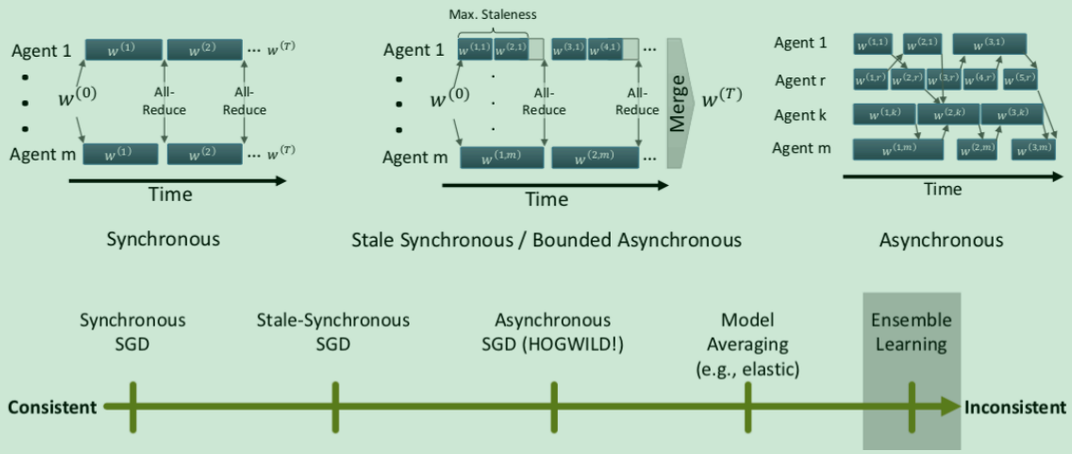



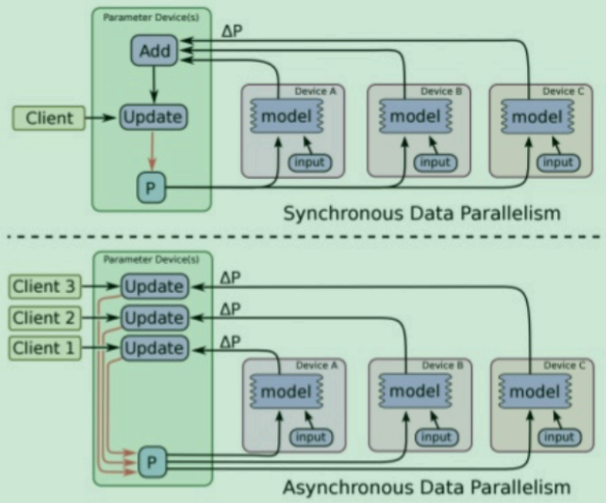

T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning

## Synchronous vs asynchronous parallelism

**HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent**


Feng Niu, Benjamin Recht, Christopher Ré and Stephen J. Wright
Computer Sciences Department, University of Wisconsin-Madison
1210 W Dayton St, Madison, WI 53706

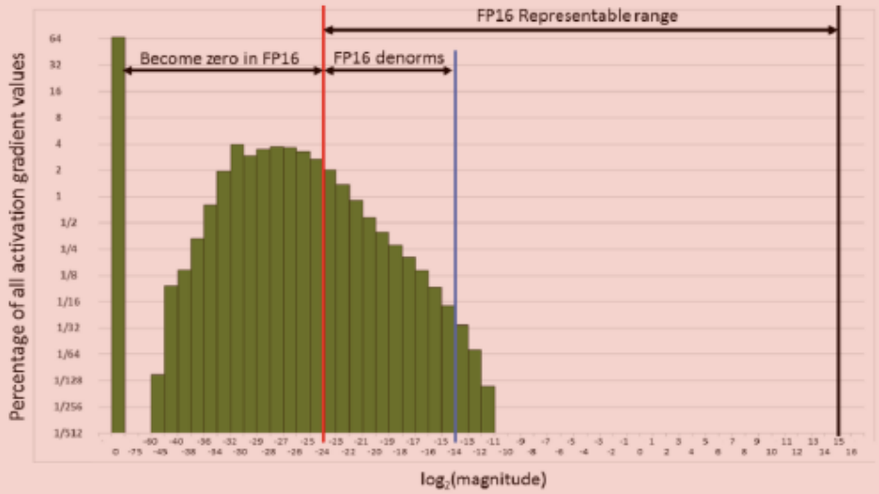

## Reduced precision communication/compression

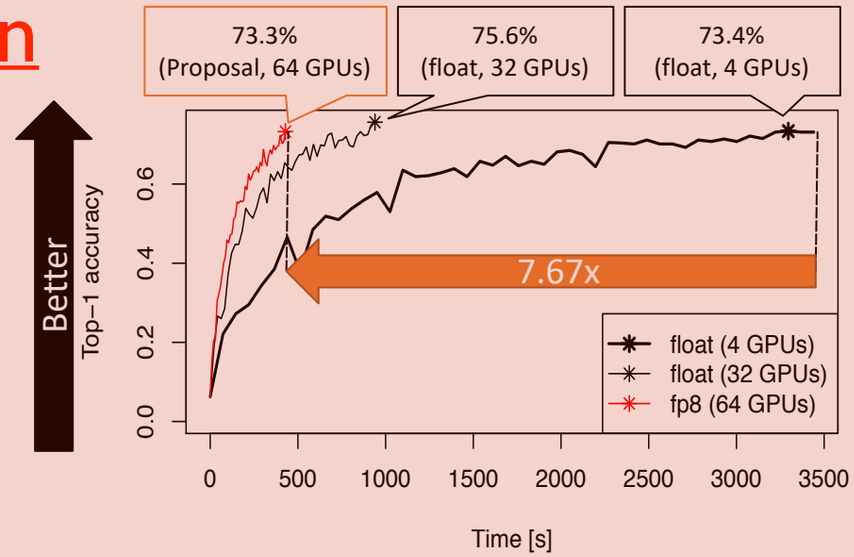
**DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING**


**Song Han**
Stanford University, Stanford, CA 94305, USA
songhan@stanford.edu

**Huizi Mao**
Tsinghua University, Beijing, 100084, China
mhz12@mails.tsinghua.edu.cn

**William J. Dally**
Stanford University, Stanford, CA 94305, USA
NVIDIA, Santa Clara, CA 95050, USA
dally@stanford.edu

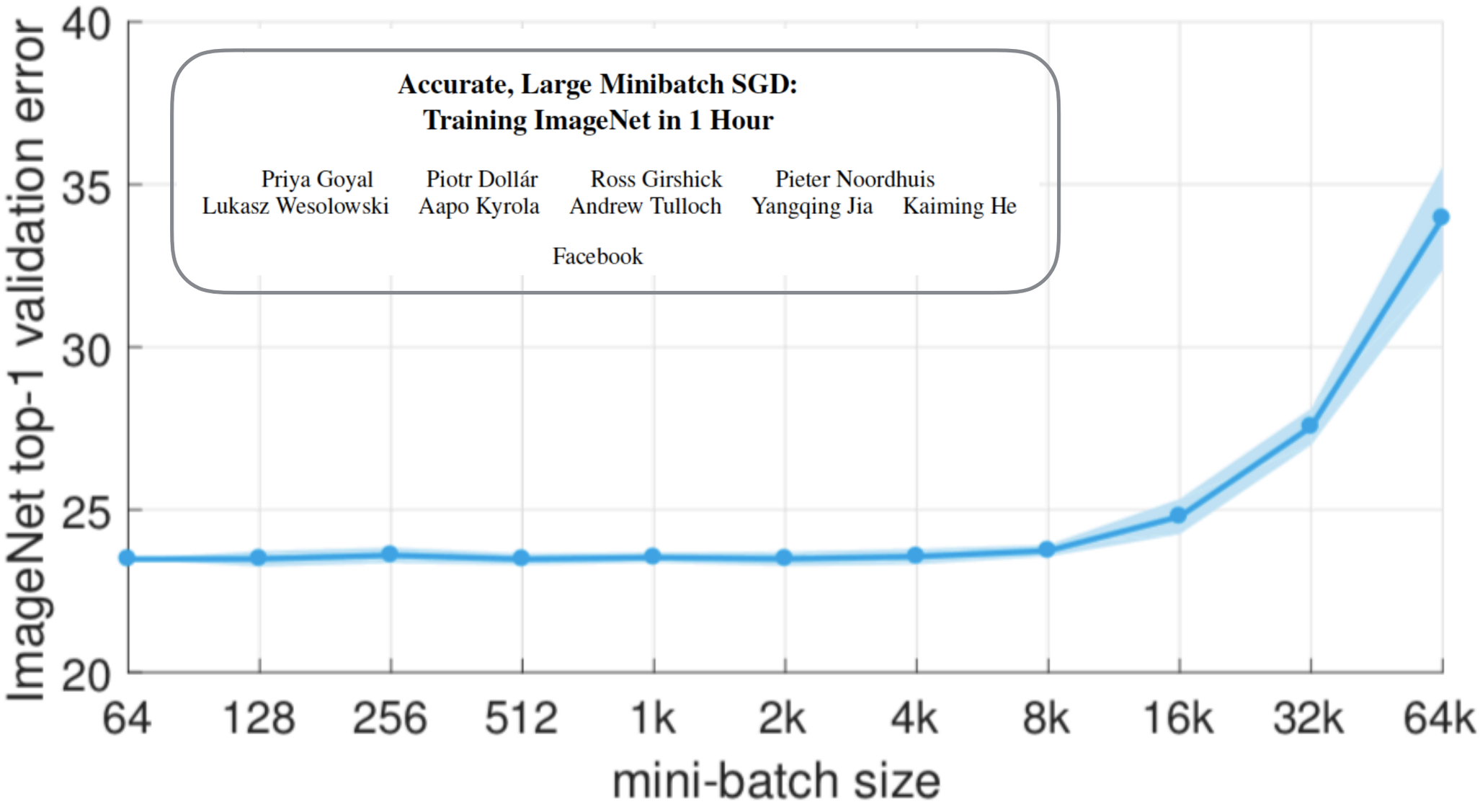

**Top-1 accuracy of GoogLeNet with various communiation types on TSUBAME-KFC/DL (Mini-batch size of 256)**

# Pushing the Limits of Large Mini-batch Learning

**Accurate, Large Minibatch SGD:**
**Training ImageNet in 1 Hour**

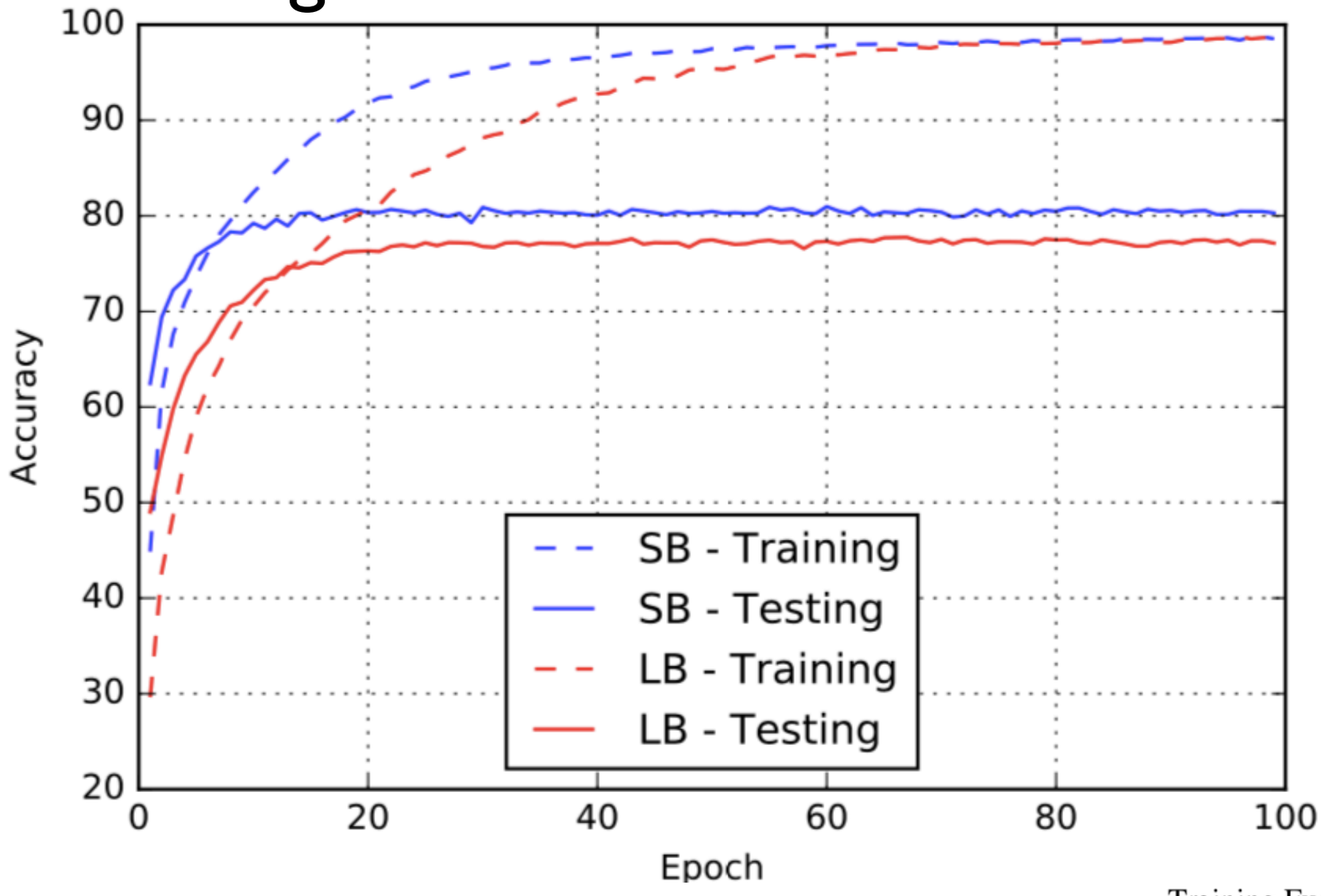
Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook




1. Increase learning rate proportional to the mini-batch size
2. But use a smaller learning rate for the first few epochs

# Why Does Large Mini-batch Have Lower Accuracy?




## On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima

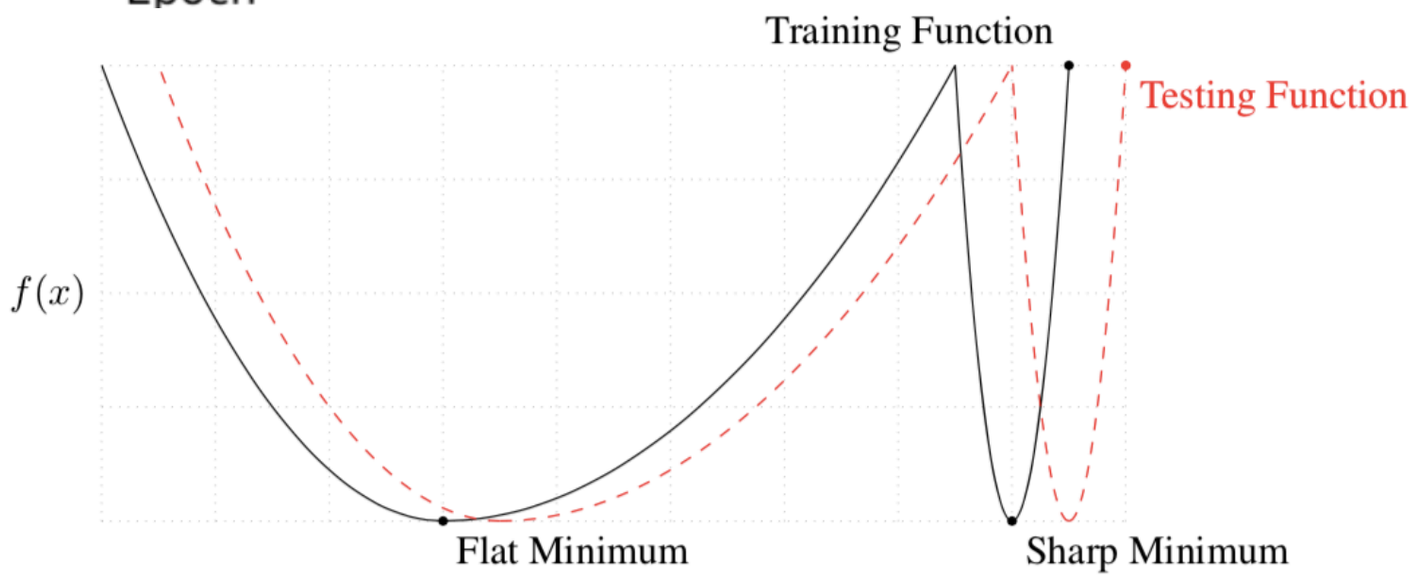
**Nitish Shirish Keskar***
Northwestern University
Evanston, IL 60208
keskar.nitish@u.northwestern.edu

**Dheevatsa Mudigere**
Intel Corporation
Bangalore, India
dheevatsa.mudigere@intel.com

**Jorge Nocedal**
Northwestern University
Evanston, IL 60208
j-nocedal@northwestern.edu

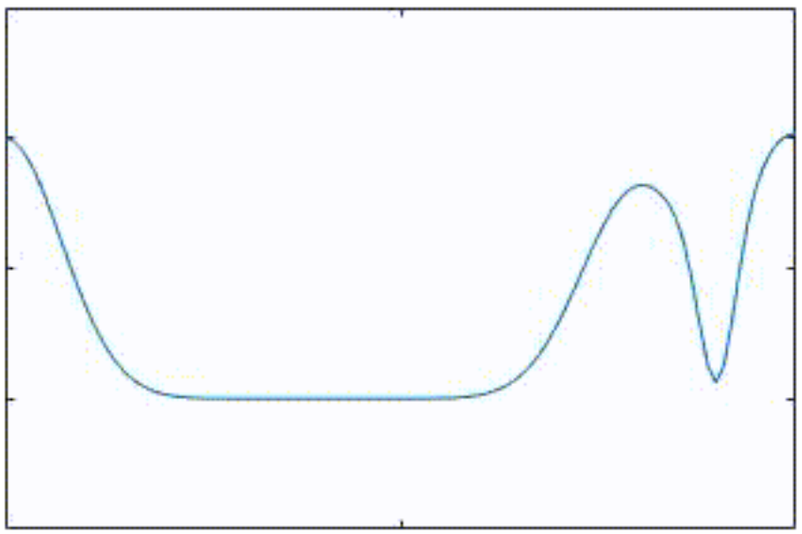
**Mikhail Smelyanskiy**
Intel Corporation
Santa Clara, CA 95054
mikhail.smelyanskiy@intel.com

**Ping Tak Peter Tang**
Intel Corporation
Santa Clara, CA 95054
peter.tang@intel.com

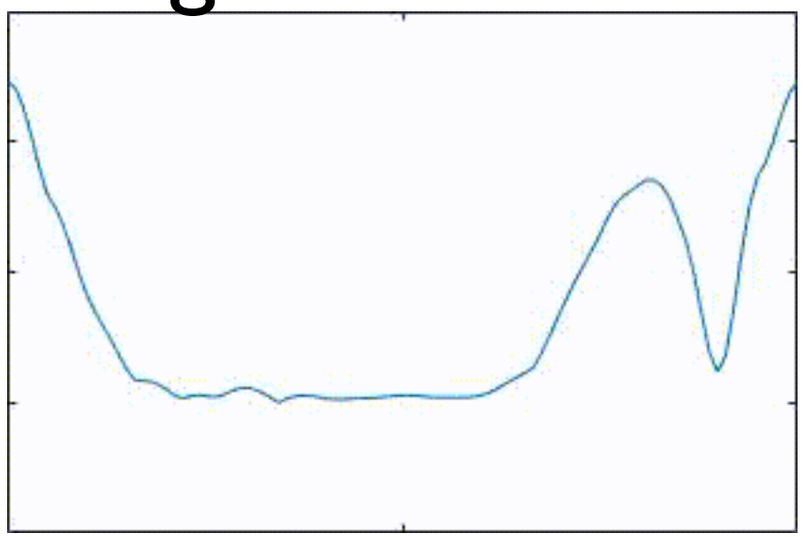


# Why Does Large Mini-batch Have Lower Accuracy?

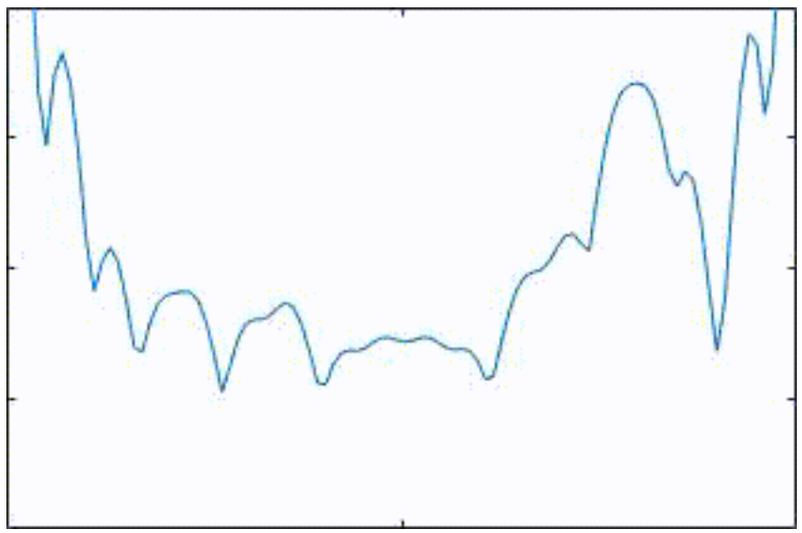
full batch



large mini-batch



small mini-batch



## On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima

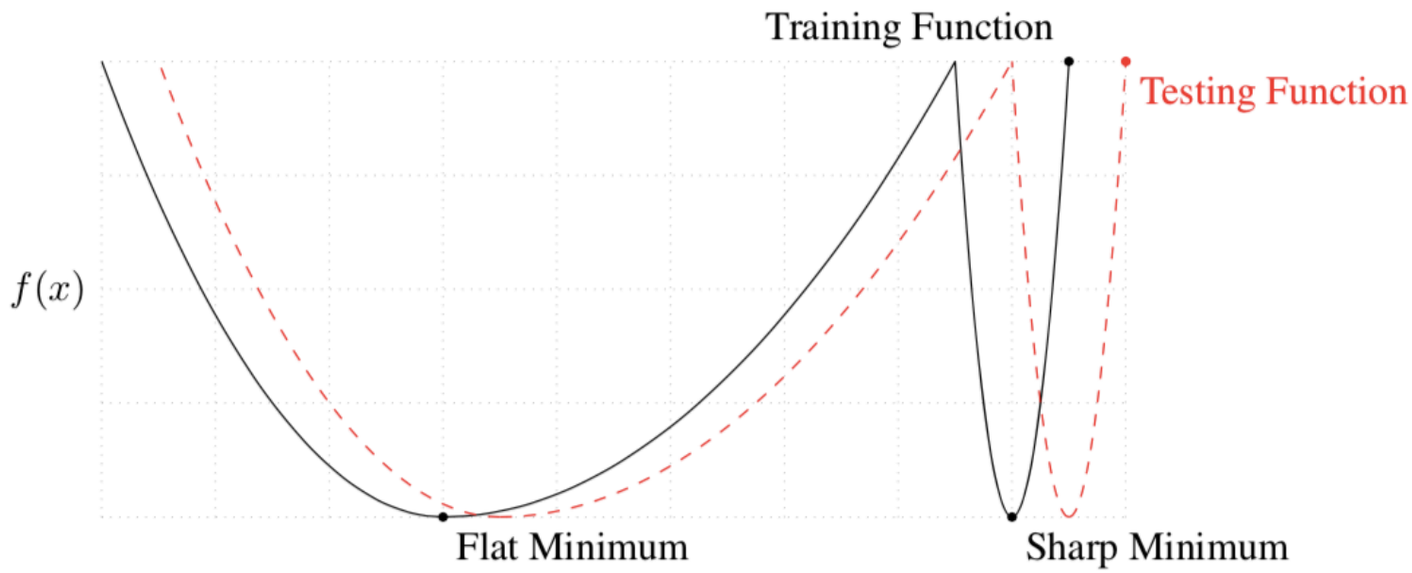

**Nitish Shirish Keskar***
Northwestern University
Evanston, IL 60208
keskar.nitish@u.northwestern.edu

**Dheevatsa Mudigere**
Intel Corporation
Bangalore, India
dheevatsa.mudigere@intel.com

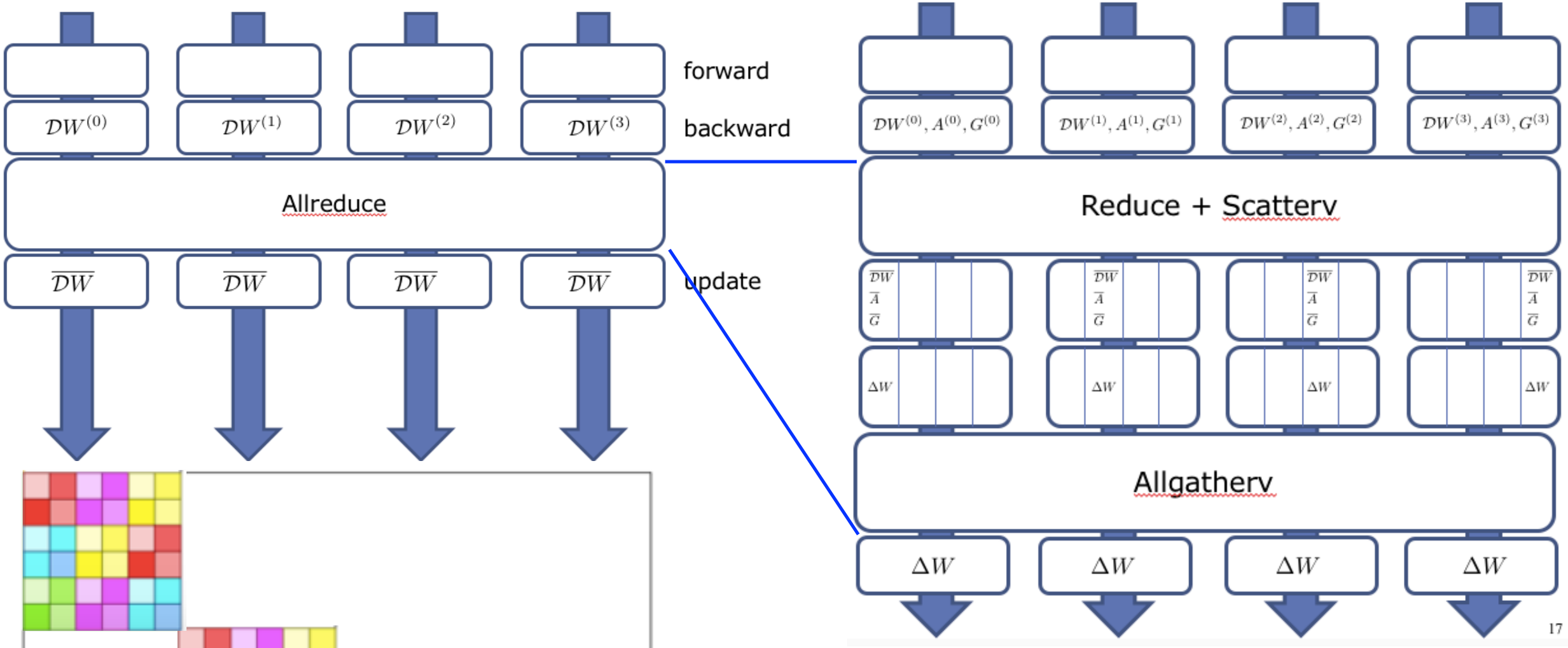
**Jorge Nocedal**
Northwestern University
Evanston, IL 60208
j-nocedal@northwestern.edu

**Mikhail Smelyanskiy**
Intel Corporation
Santa Clara, CA 95054
mikhail.smelyanskiy@intel.com

**Ping Tak Peter Tang**
Intel Corporation
Santa Clara, CA 95054
peter.tang@intel.com

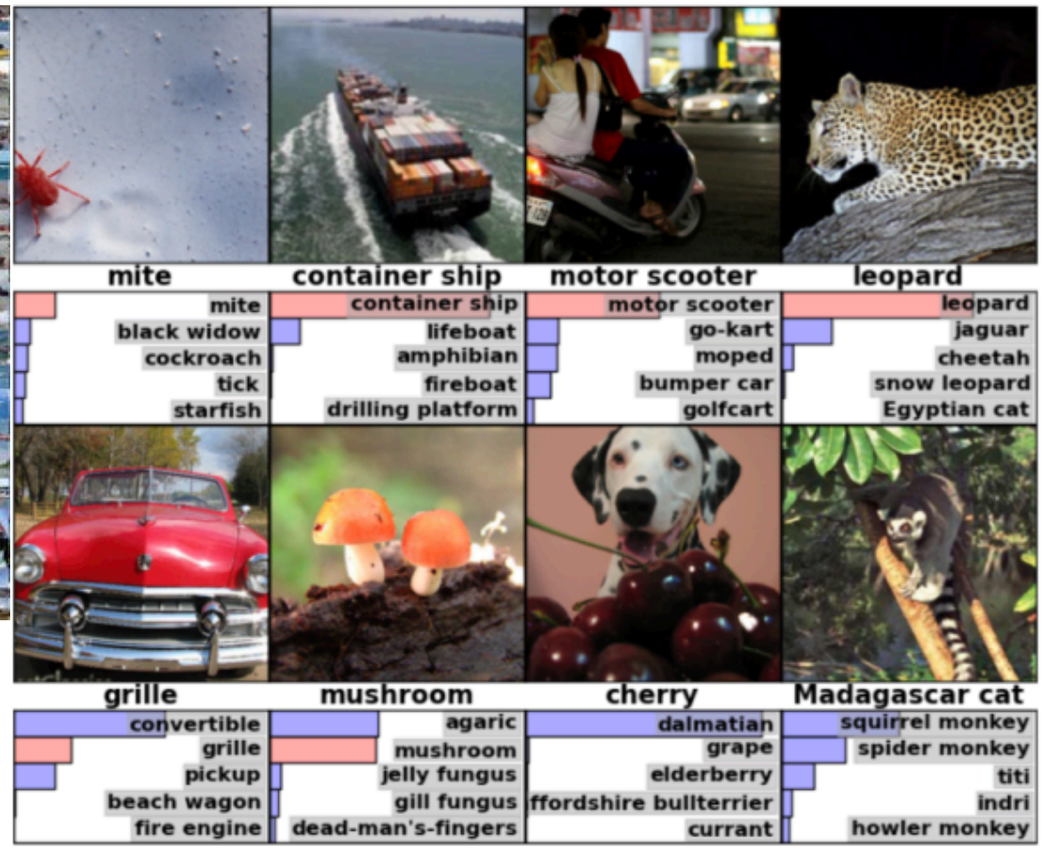
# K-FAC fits in Rabenseifner AllReduce




After the expectation and inverse are calculated
$F^{-1}\nabla J$ turns into a triple matrix multiplication

$$\approx \underbrace{E[\ \blacksquare\ ]^{-1}}_{\overline{A}} \otimes \underbrace{E[\ \blacksquare\ ]^{-1}}_{G}$$

# Training ImageNet in Minutes




**Accurate, Large Minibatch SGD:**
**Training ImageNet in 1 Hour**


Priya Goyal Piotr Dollár Ross Girshick Pieter Noordhuis
Lukasz Wesolowski Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He

Facebook


**ImageNet Training in Minutes**

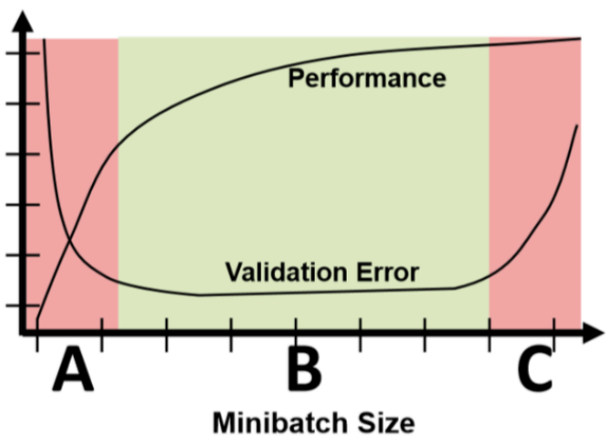

**Yang You[1], Zhao Zhang[2], Cho-Jui Hsieh[3], James Demmel[1], Kurt Keutzer[1]**


**Extremely Large Minibatch SGD:**
**Training ResNet-50 on ImageNet in 15 Minutes**


**Takuya Akiba**
Preferred Networks, Inc.
akiba@preferred.jp

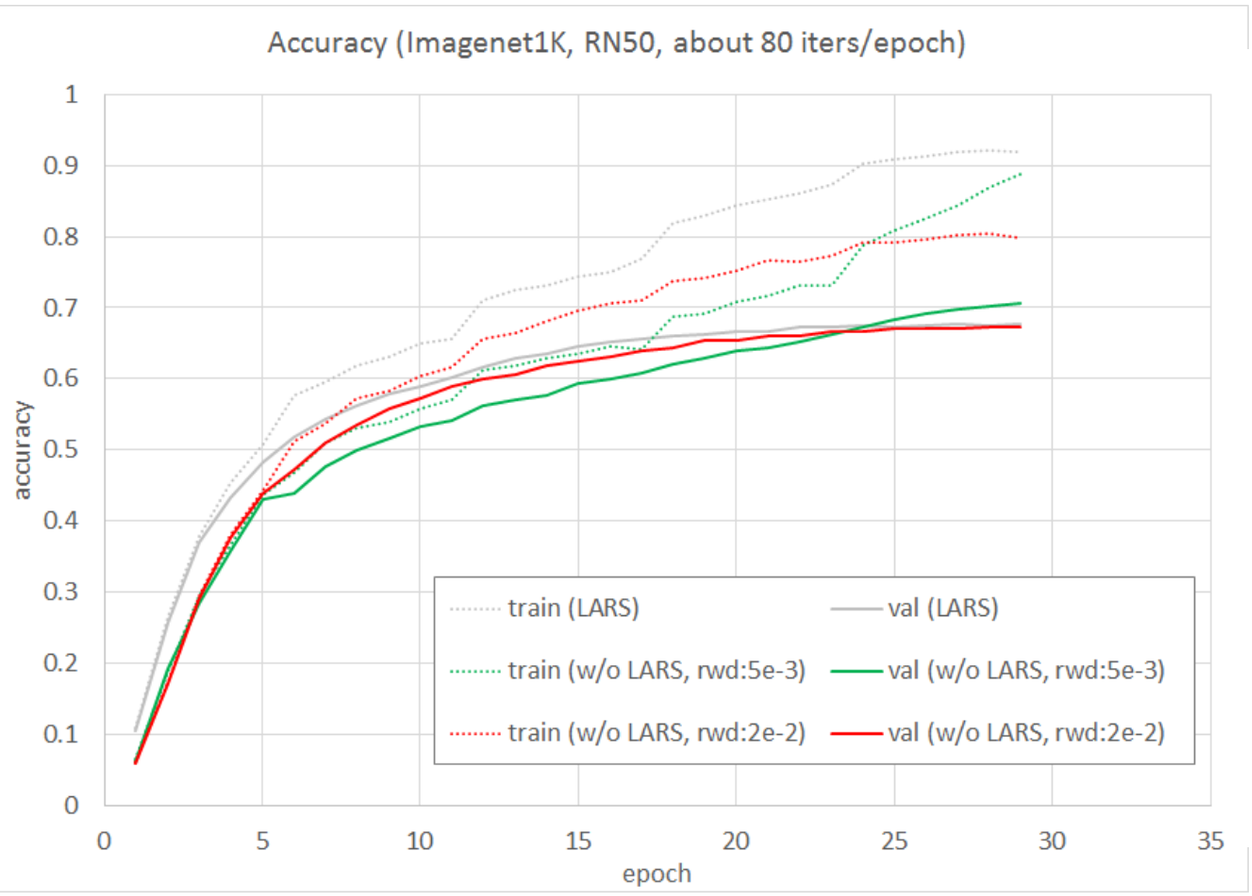
**Shuji Suzuki**
Preferred Networks, Inc.
ssuzuki@preferred.jp

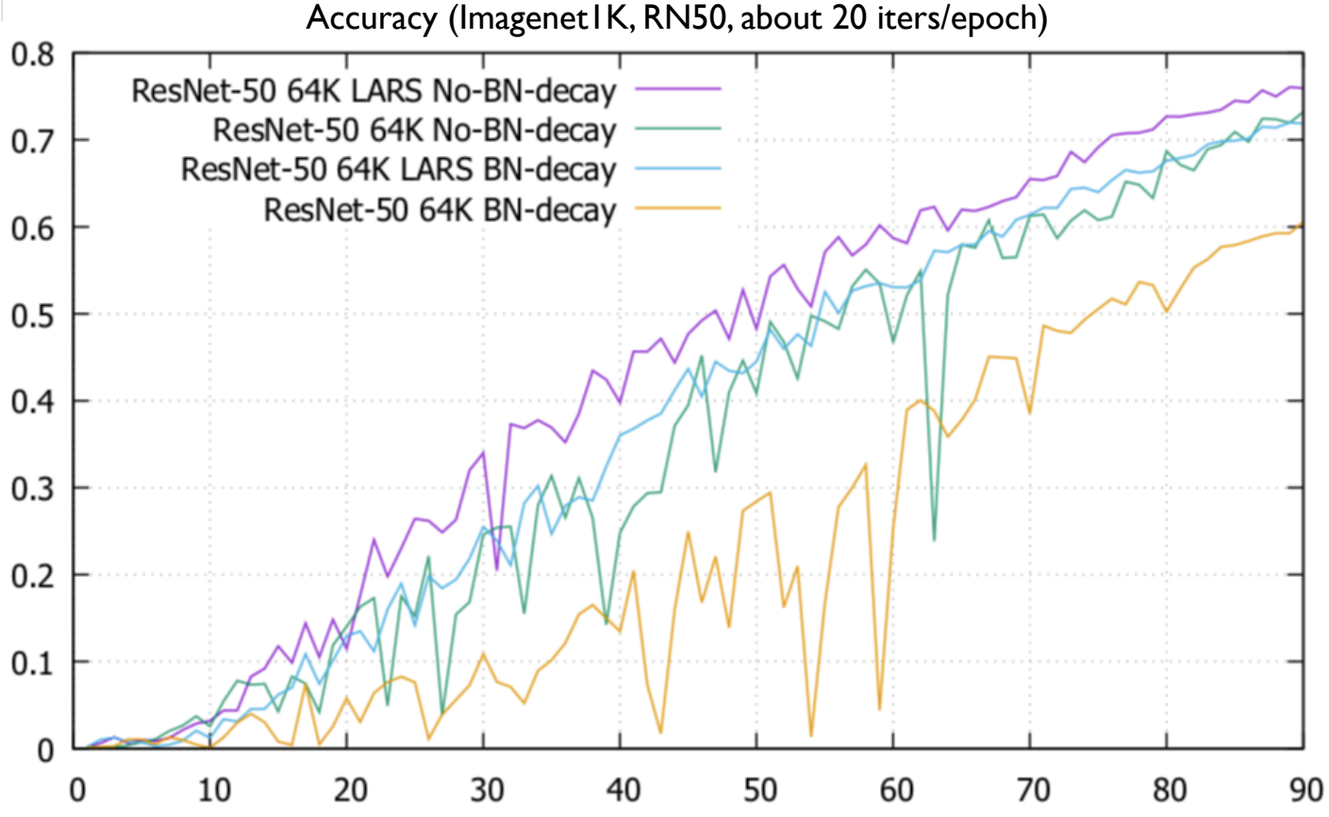
**Keisuke Fukuda**
Preferred Networks, Inc.
kfukuda@preferred.jp


| Team | Hardware | Software | Minibatch size | Time | Accuracy |
|---|---|---|---|---|---|
| He *et al.* [5] | Tesla P100 × 8 | Caffe | 256 | 29 hr | 75.3 % |
| Goyal *et al.* [4] | Tesla P100 × 256 | Caffe2 | 8,192 | 1 hr | 76.3 % |
| Codreanu *et al.* [3] | KNL 7250 × 720 | Intel Caffe | 11,520 | 62 min | 75.0 % |
| You *et al.* [10] | Xeon 8160 × 1600 | Intel Caffe | 16,000 | 31 min | 75.3 % |
| This work | Tesla P100 × 1024 | Chainer | 32,768 | 15 min | 74.9 % |

# Top-1 Validation Accuracy




Our results for 16K batch

LARS: Layerwise normalization

$$\Delta w_t^l = \gamma \cdot \eta \cdot \frac{\|w^l\|}{\|\nabla L(w^l))\|} \cdot \nabla L(w_t^l)$$

**Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes**


**Xianyan Jia[*1], Shutao Song[*1], Wei He[1], Yangzihao Wang[1], Haidong Rong[1], Feihu Zhou[1], Liqiang Xie[1], Zhenyu Guo[1], Yuanzhou Yang[1], Liwei Yu[1], Tiegang Chen[1], Guangxiao Hu[1], Shaohuai Shi[*2], Xiaowen Chu[2]**

Tencent Inc.[1], Hong Kong Baptist University[2]

{xianyanjia, sampsonsong, winsonwhe, slashwang, hudsonrong, hopezhou, felixxie, alexguo, jokeyang, leolwyu, steelchen, gethinhu}@tencent.com;

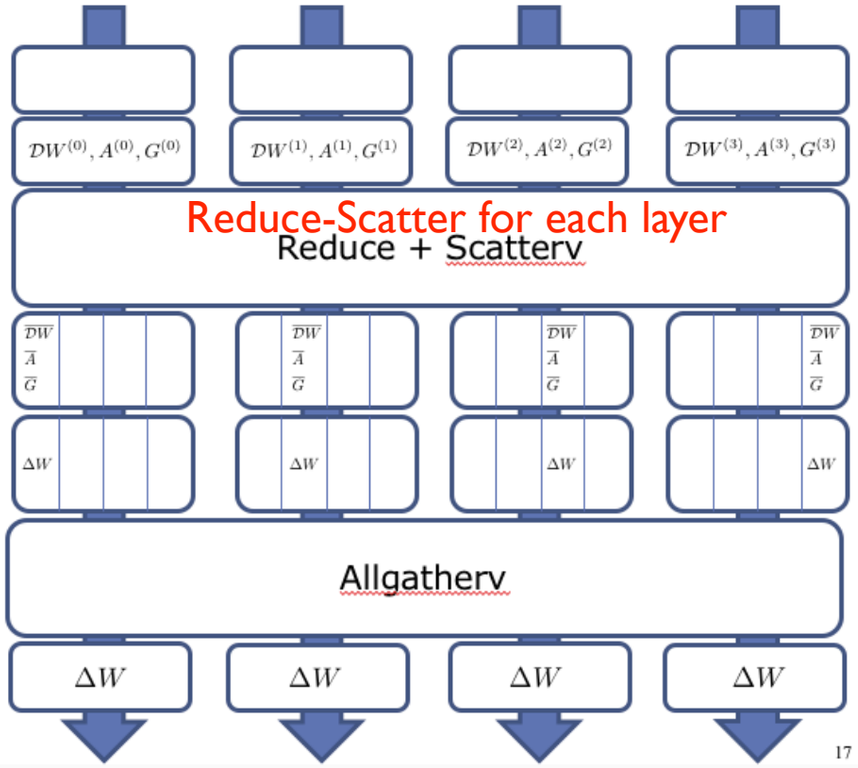
{csshshi, chxw}@comp.hkbu.edu.hk

*Authors contributed equally


No weight decay for Batch Norm layer

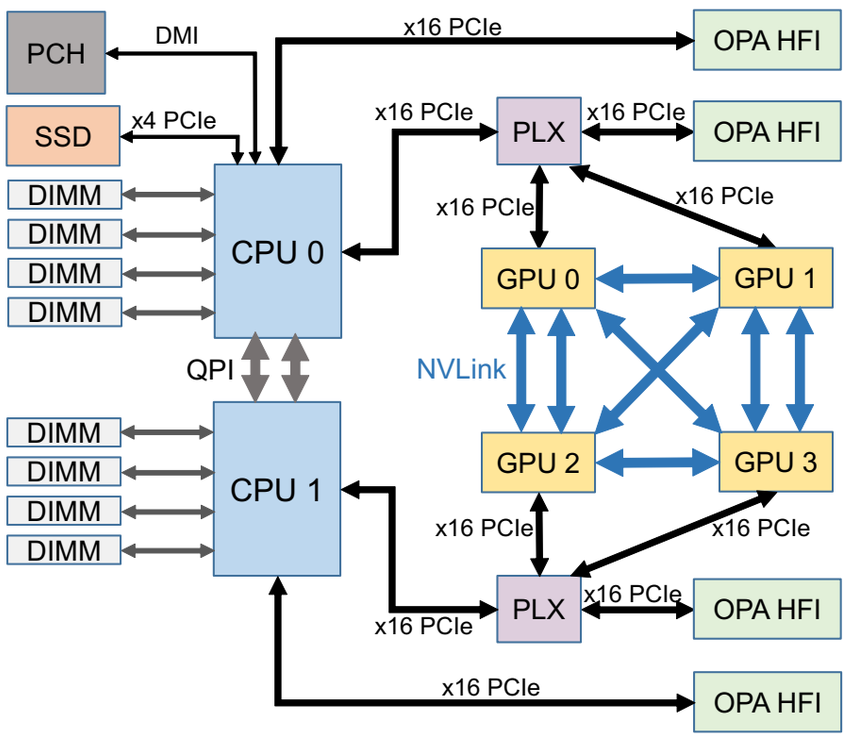
K-FAC is invariant to reparameterizations such as Batch Norm

# Parallel Scalability

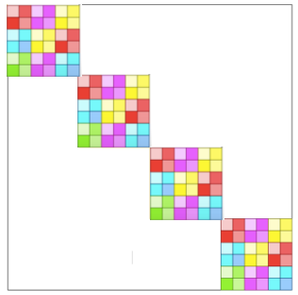



Inter-node Rabenseifner AllReduce

Intra-node Ring AllReduce



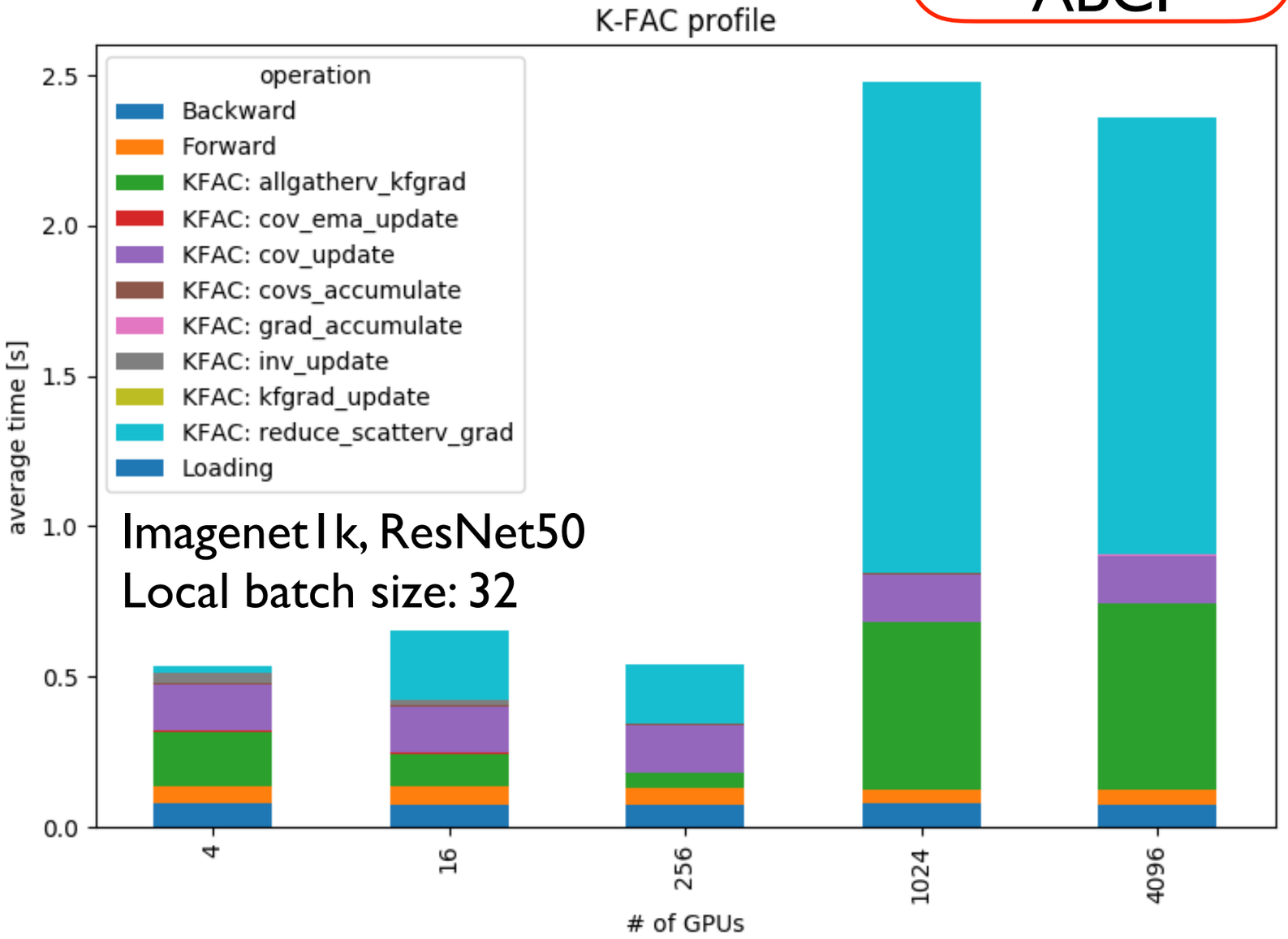

Cross-over from data-parallel to model-parallel



Cross-over from model-parallel to data-parallel

TSUBAME3.0

ABCI



# Further Possibilities for K-FAC

Large Scale Distributed Learning
Distributed Second-Order Optimization Using Kronecker-factored Approximations [Ba et al. 2017]

Recurrent Neural Networks
Konecker-factored Curvature Approximations for Recurrent Neural Networks [Martens et al. 2018]

Approximating Real-Time Recurrent Learning with Random Kronecker Factors [Mujika et al. 2018]

Kronecker Recurrent Units [Jose et al. 2018]

Reinforcement Learning
Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation [Wu et al. 2017]

An Empirical Analysis of Proximal Policy Optimization with Kronecker-factored Natural Gradients [Song & Wu 2018]

Variational Inference
Noisy Natural Gradient as Variational Inference [Zhang et al. 2017]

# Summary & Outlook

1. Large mini-batch and second order methods have the same problem
   - They overfit (fall into sharp minima)

2. Proper regularization is the only way to solve this problem
   - Once solved, second order methods become advantageous

3. Second order methods can take larger steps in the correct direction
   - For large-batch learning this is a huge advantage

# Acknowledgements


<u>Collaborators</u>
Koichi Shinoda
Tsuyoshi Murata
Satoshi Matsuoka
Hitoshi Sato
Akira Naruse

<u>Students</u>
Kazuki Osawa
Hiroki Naganuma
Shun Iwase
Hiroyuki Otomo
Yuji Kuwamura
Yuichiro Ueno
Yohei Tsuji