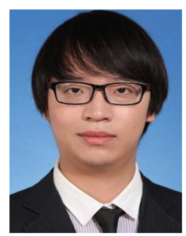
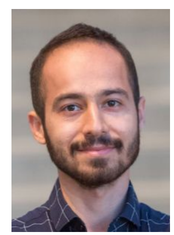
MENG Capstone Final Exam

Systematic Analysis of Second Order Optimization in Large-scale Neural Network Training

Linjian Ma*, Gabe Montague*, Jiayu Ye*,

Zhewei Yao, Amir Gholami, Kurt Keutzer, Michael W. Mahoney

University of California, Berkeley

Berkeley UNIVERSITY OF CALIFORNIA

riselab UC Berkeley

High-Level Outline

- Deep Learning and Large batch training
- Kronecker-Factored Approximate Curvature (K-FAC)
- Results: Level-playing-field performance comparison
- Results: Hyperparameter robustness behavior

Deep learning is the Trend

AlphaGo

Large Scale Deep Learning

Consuming data is slow

OPENAI FIVE

128,000 preemptible CPU cores on GCP	
256 P100 GPUs on GCP	
~180 years per day (~900 years per day counting each hero separately)	

Total: 10000 years

~2 realtime month

Large batch training saves time

- need fast training -> parallelization -> large batch

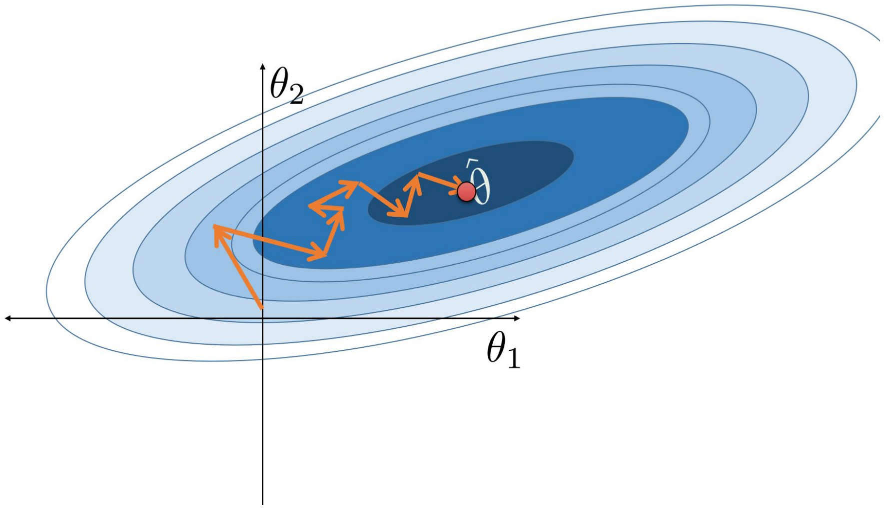
Objective function

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(x_i, y_i, \theta)$$

Update rule

$$\theta_{t+1} = \theta_t - \eta_t \frac{1}{|B|}\sum_{(x,y)\in B} \nabla_\theta l(x, y, \theta_t)$$

small $|B|$



large $|B|$

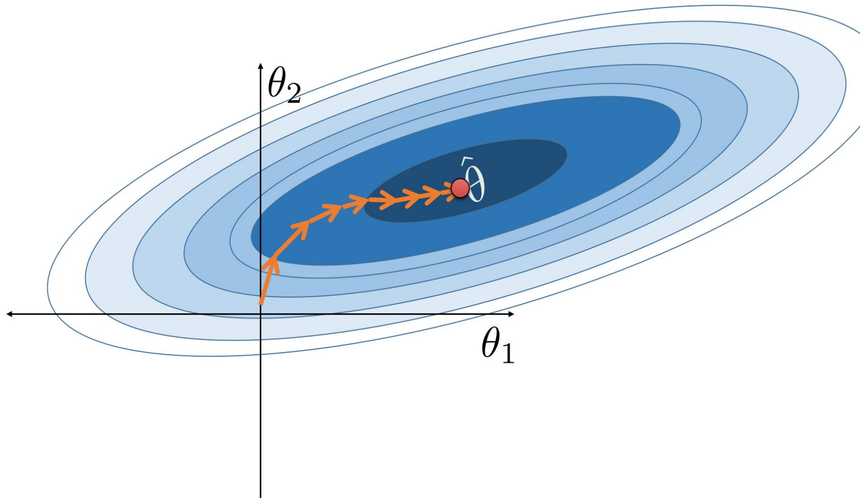
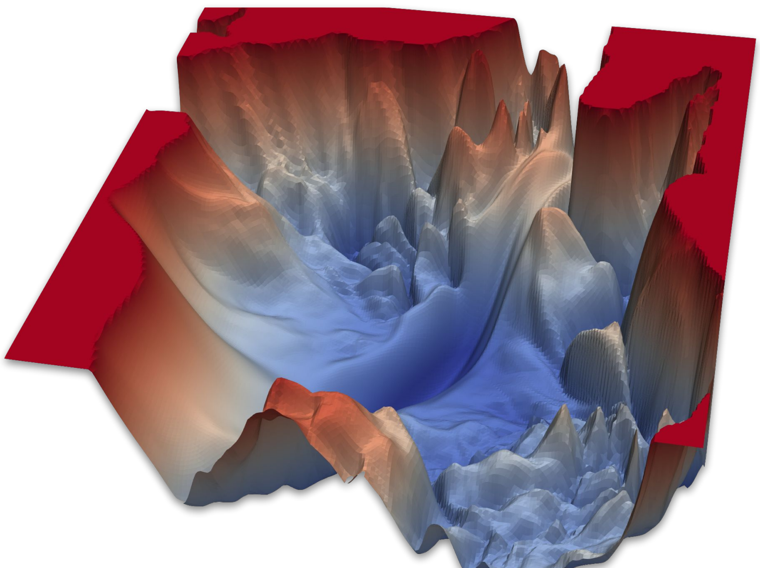
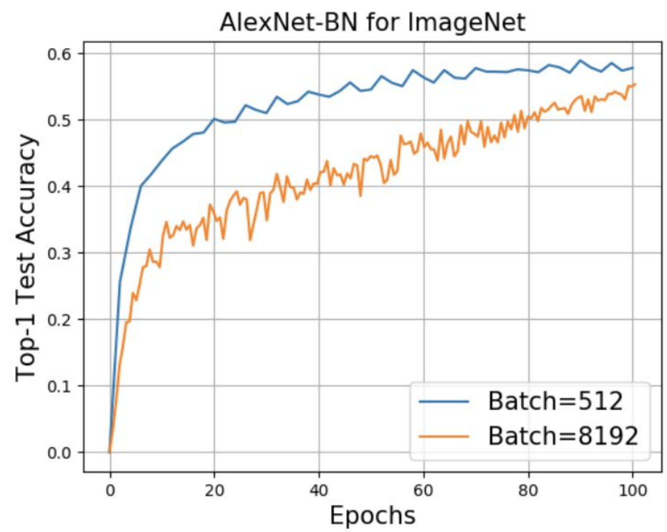


Image from https://www.textbook.ds100.org/ch/11/gradient_stochastic.html

Large batch SGD encounters problems

- **Large batch SGD may get "stuck":**
  - Sharp minima
  - Saddle points
- Testing **and** training performance affected



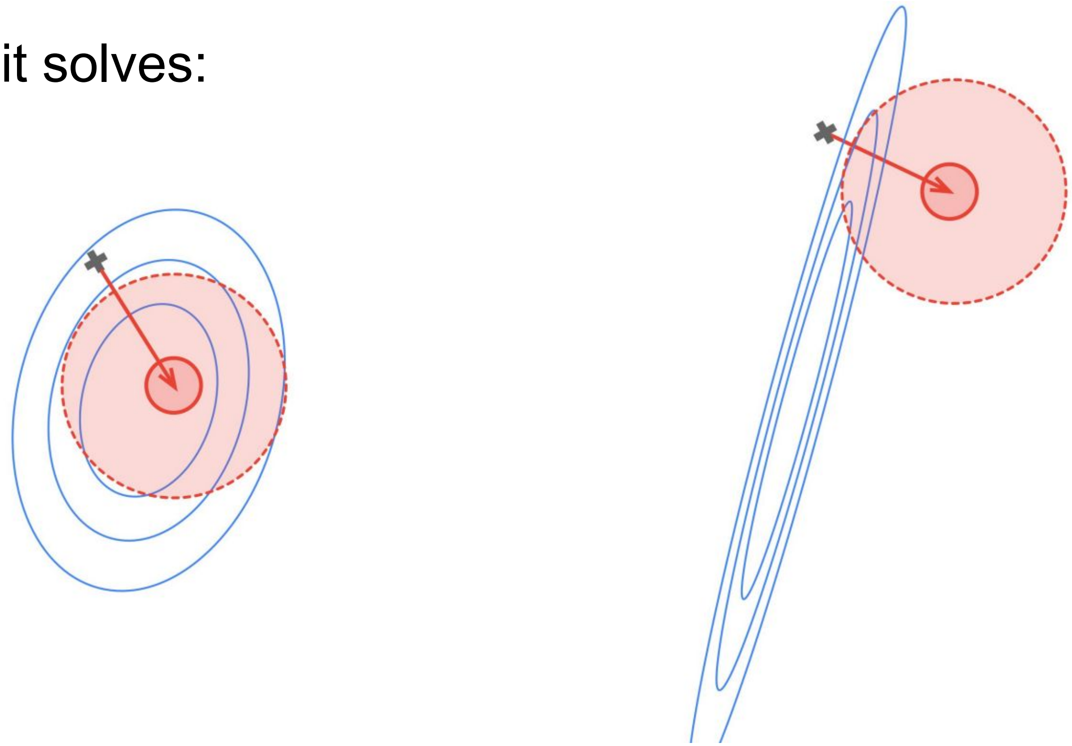


(a) Training without LARS

Loss landscape from https://www.cs.umd.edu/~tomg/projects/landscapes/
Ginsburg, Boris, Igor Gitman, and Yang You. "Large Batch Training of Convolutional Networks with Layer-wise Adaptive Rate Scaling." arxiv:1708.03888.

Second order method: K-FAC

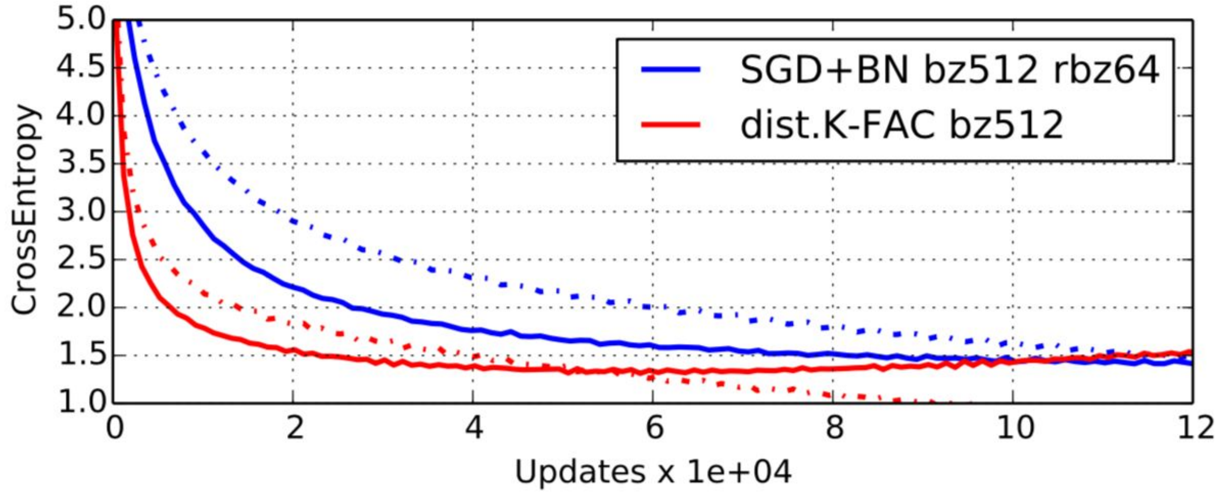
- Kronecker-Factored Approximate Curvature (K-FAC)
- Problem it solves:



- Uses heavily-approximated second-order information to compute parameter updates

https://supercomputersfordl2017.github.io/Presentations/K-FAC.pdf
James Martens and Roger B. Grosse. “Optimizing Neural Networks with Kronecker-factored Approximate Curvature”. In: *CoRR* abs/1503.05671 (2015). arXiv: 1503.05671.

Empirical studies of K-FAC vs. SGD



- Comparable performance to SGD
- Better for large batch training?

Jimmy Ba, Roger Grosse, and James Martens. “Distributed second-order optimization using Kronecker- factored approximations”. In: *International Conference on Learning Representations*. 2017.

Training budget selection

- Different assumptions exist in literature:

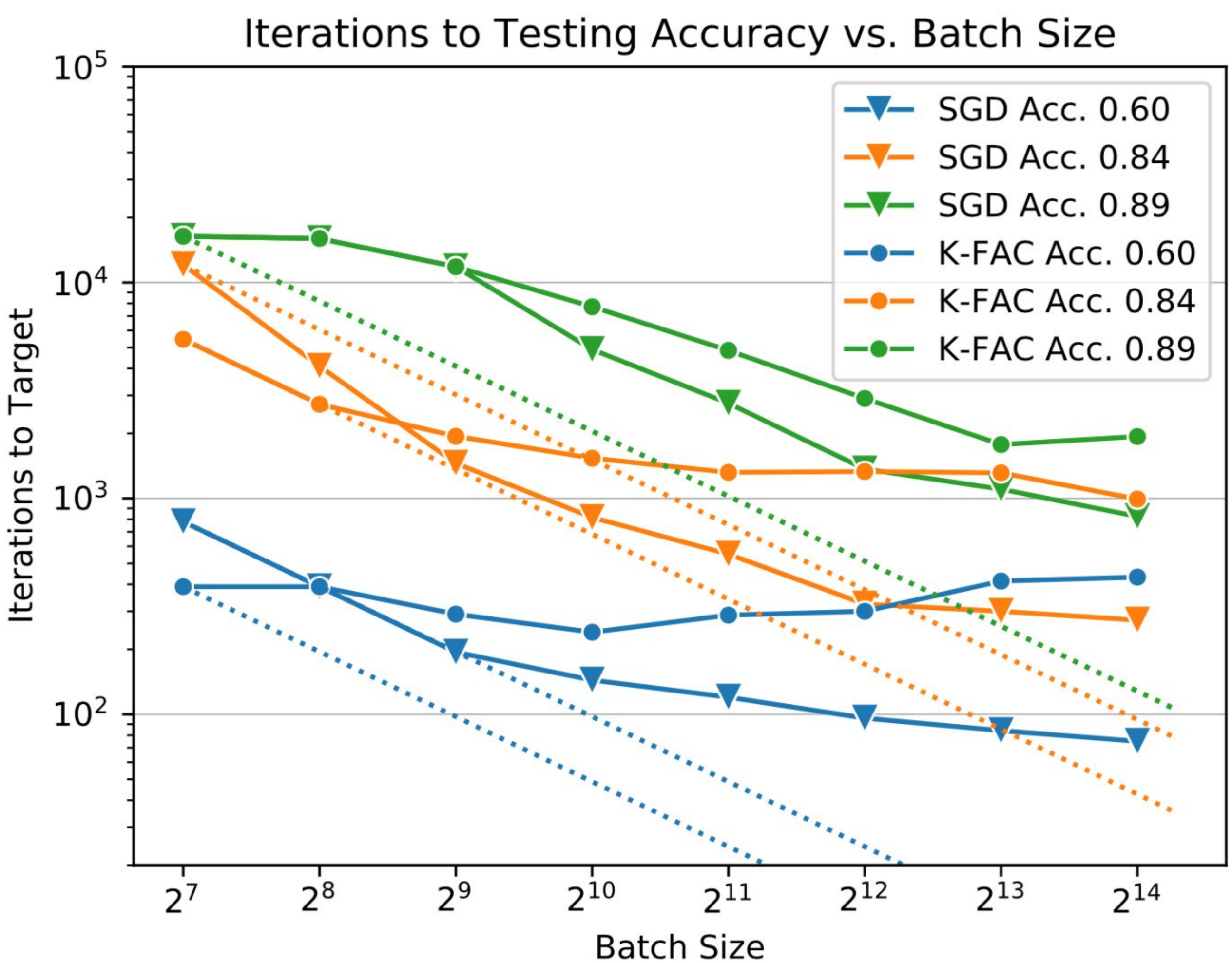
Limiting factor is **Time**	Limiting factor is **Money/Computation**
• Example: Owners of computing resources, e.g. Google • Easier hyperparameter tuning parallelization • **Stop training based on number of updates: "iterations"**	• Example: Renters of computing resources, e.g. AWS users • Paying for each hyperparameter tried • **Stop training based on number of epochs or training examples**

A level-playing-field performance comparison

Hyperparameters of **SGD**	Hyperparameters of **K-FAC**
◦ **Learning rate** ◦ **Momentum** ◦ Weight decay	◦ **Learning rate** ◦ **Damping** ◦ Momentum ◦ Weight decay ◦ Clipping magnitude ◦ Second-order update frequency ◦ Second-order update momentum

- No relationships assumed between hyperparameters
- We tune two hyperparameters equally - giving the same amount of training to both methods

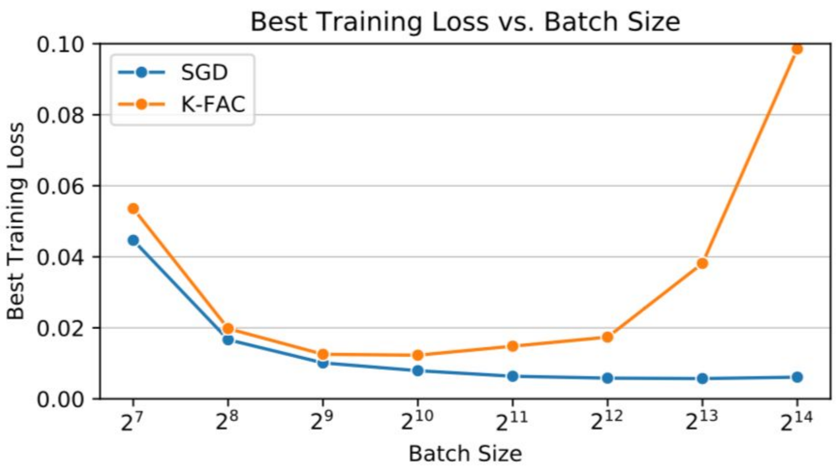
A level-playing-field performance comparison

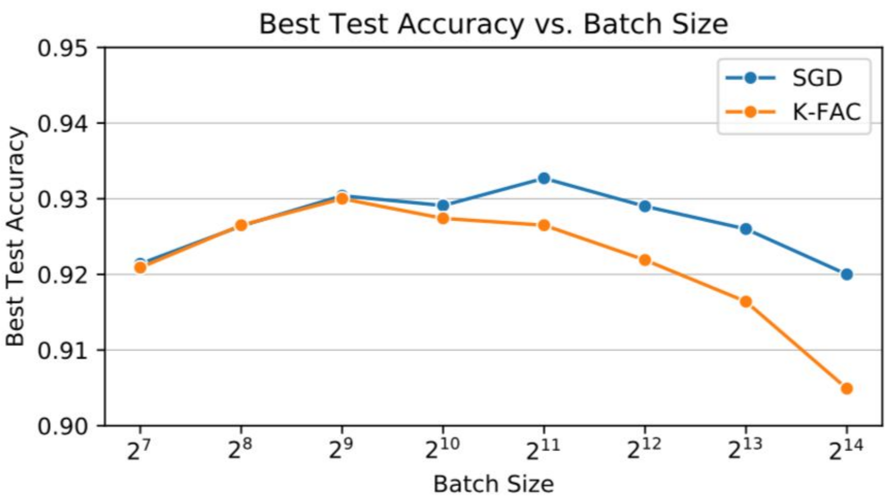


- We race SGD and K-FAC to 3 target **accuracies**
- **Dotted: Ideal hyperbola** (linear because log-log)

A level-playing-field performance comparison

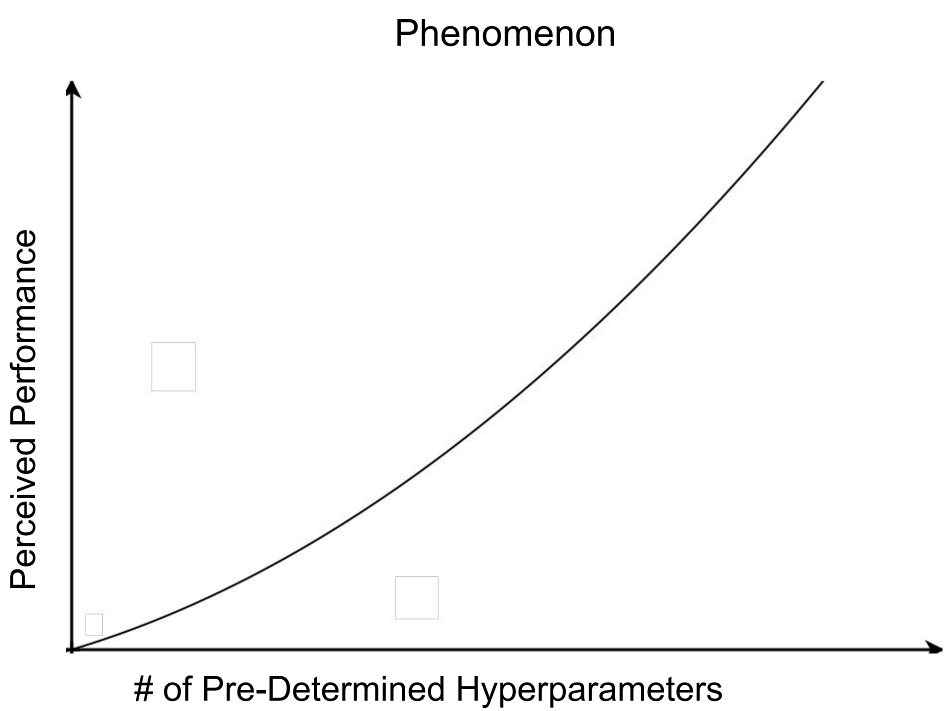
- **Consistently slower = consistently worse end-of training performance (regardless of stopping rule)**
- Our stopping rule:
  - Total training epochs = (log2(batch size/128) + 1) × 100
- End-of-training (SGD vs. K-FAC):





Discussions

Did we get our results because we didn't tune carefully enough for K-FAC?



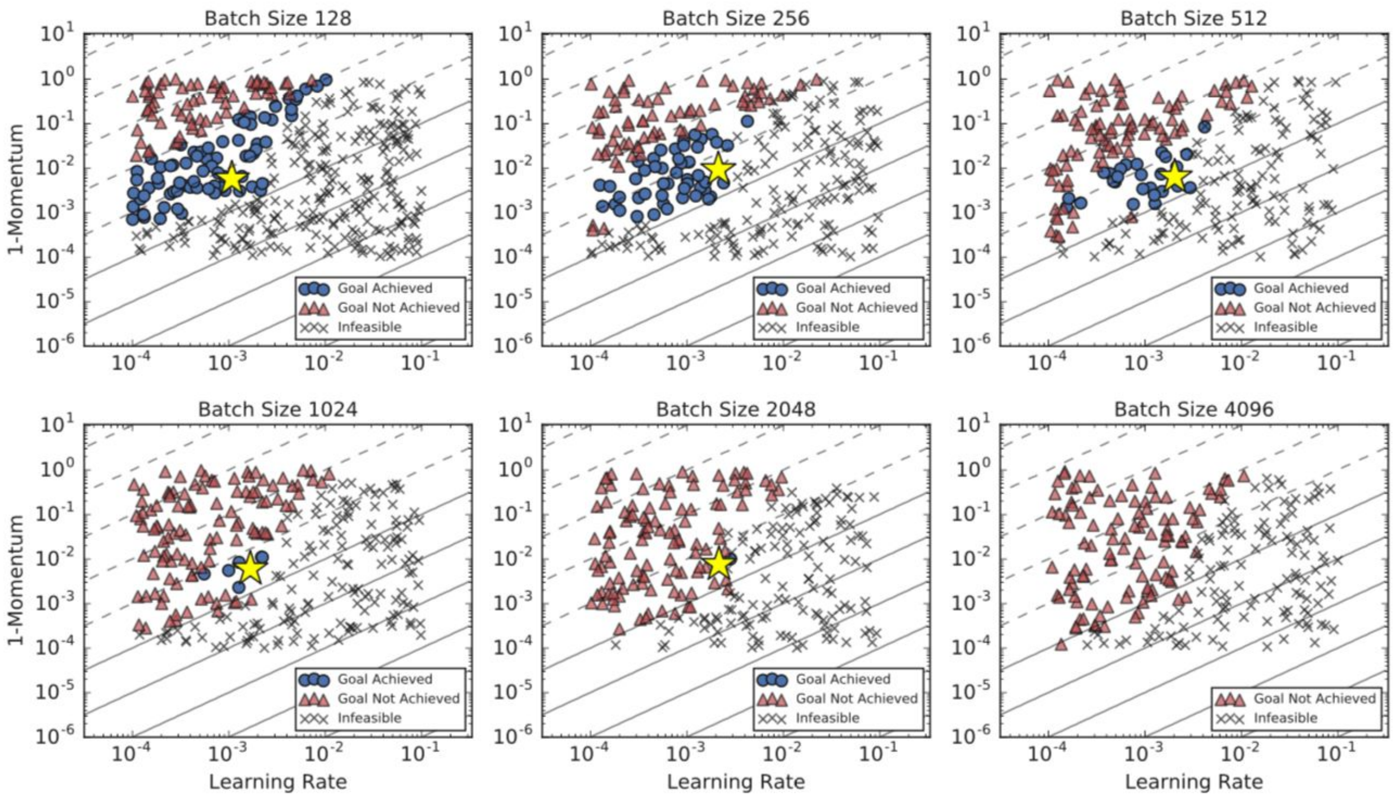
- **Hyperparameter tuning time is often ignored**
  - Justification: they are **"tune once" values** that **generalize**
- We used the "tune once" values for K-FAC and **it appears they did not generalize**

Conclusion: K-FAC hyperparameter tuning time = training time. We gave it the same amount as SGD

How do we measure hyperparameter robustness?

When limiting factor is money/computation (epochs):

Google SGD study:



Failure

Success

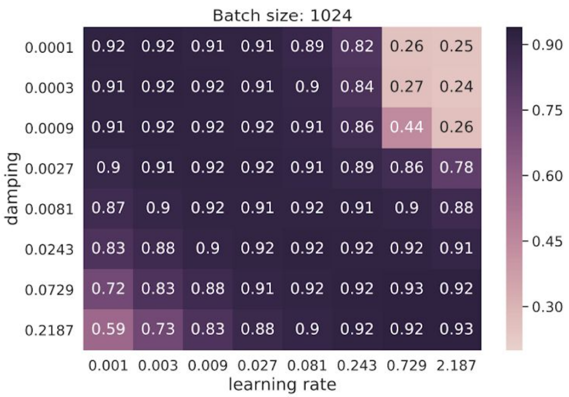
“Transformer on LM1B with a training budget of one epoch”

Christopher J. Shallue, Jaehoon Lee, Joseph M. An- tognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. “Measuring the Effects of Data Parallelism on Neural Network Training”. In: *CoRR* abs/1811.03600 (2018). arXiv: 1811.03600.

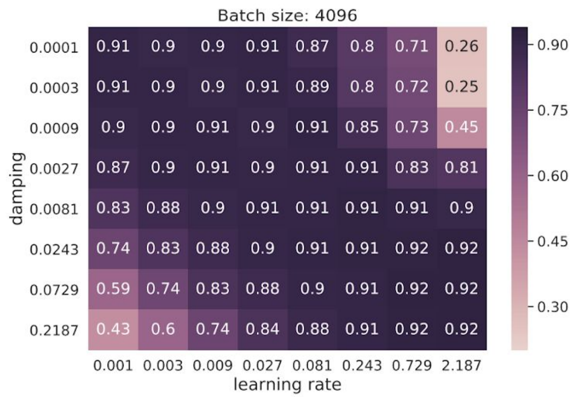
K-FAC Hyperparameter Robustness: Heatmaps

K-FAC Accuracy Heatmaps under our epoch-like budget:*

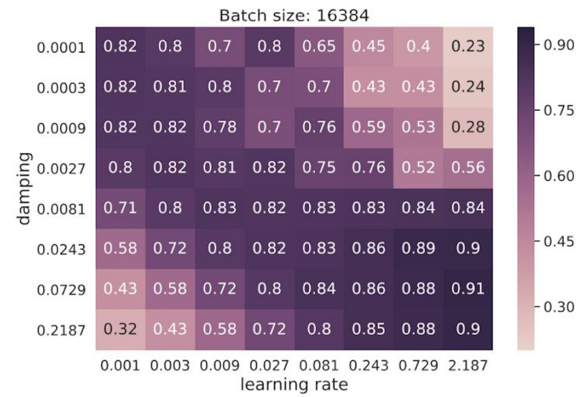
|B| = 1,024



|B| = 4,096



|B| = 16,384

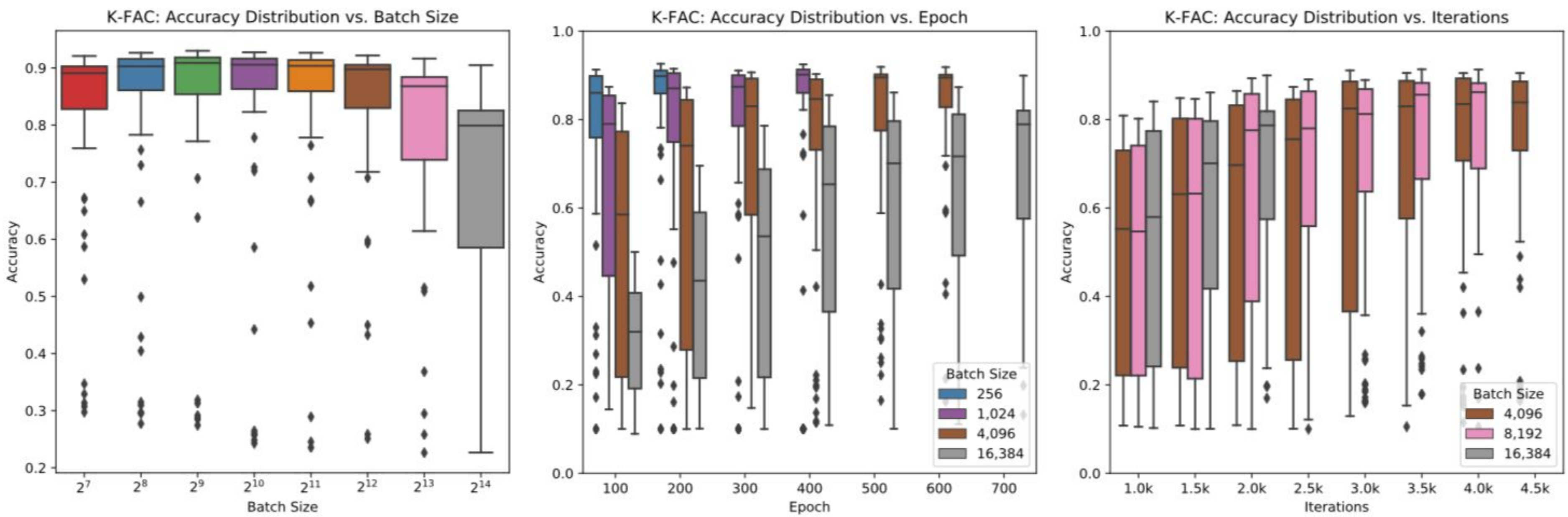


- **Shrinking** of the high-accuracy region with increasing batch size
- (Coincidental: positive correlation observed between damping and learning rate)

*Total training epochs = $(\log_2(\text{batch size}/128) + 1) \times 100$

K-FAC Hyperparameter Robustness: Box plots

K-FAC Accuracy Box Plots over time:



(a) (b) (c)

- **Growing** of the high-accuracy region with increasing batch size **with iteration budget** (not epoch buget)
- Aligns with Google finding for SGD
- Demonstrates competency of K-FAC algorithm

Christopher J. Shallue, Jaehoon Lee, Joseph M. An- tognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. “Measuring the Effects of Data Parallelism on Neural Network Training”. In: *CoRR* abs/1811.03600 (2018). arXiv: 1811.03600.

Summary of Empirical Results

- At small batch sizes, even with extensive hyperparameter tuning, K-FAC has **comparable**, but not superior, train/test performance to SGD.
- Increasing batch size for K-FAC results in **slower training** compared to SGD, when measured in terms of iterations.
- For fixed epochs, larger batch sizes result in **weaker** hyperparameter robustness.
- For fixed iterations, larger batch sizes result in **greater** hyperparameter robustness.

Summary of Findings

- Methods investigated: **K-FAC, TRCG (Trust Region Conjugate Gradient), Stochastic TRCG, SGD**
- *Similarities*:
  - General robustness patterns
  - Large-batch scalability problems
- *Differences:*
  - Large-batch performance, hyperparameter robustness