Large-Scale Distributed Second-Order Optimization

Robin Schmidt

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Motivation and Related Work

- **Optimization**: Any branch of ML has optimization problems (RL, Graphics & Vision, DL, etc.)

- **Problem**: Increasing Data sizes $\Rightarrow$ Faster Convergence $\Rightarrow$ Better Optimizers, Parallel computing

- **First-Order Optimization Methods**: *SGD* [RM51], *Adam* [KB14], *AdamW* [LH17], *AMSGrad* [RKK19], *AdaBound* [LXLS19], *AMSBound* [LXLS19], *RAdam* [LJH^{+}19], *LookAhead* [ZLHB19]

- **Second-Order-Optimization Methods**: *Gauss-Newton-Method* [Sch02], *Natural Gradient Descent* (NGD) [Ama98], *Kronecker-factored Approximate Curvature* (K-FAC) [MG15]

Motivation and Related Work: [SLA+18]

- **Problem in Parallel Optimization:** Increasing Mini-Batch Size decreases validation accuracy [SLA+18]

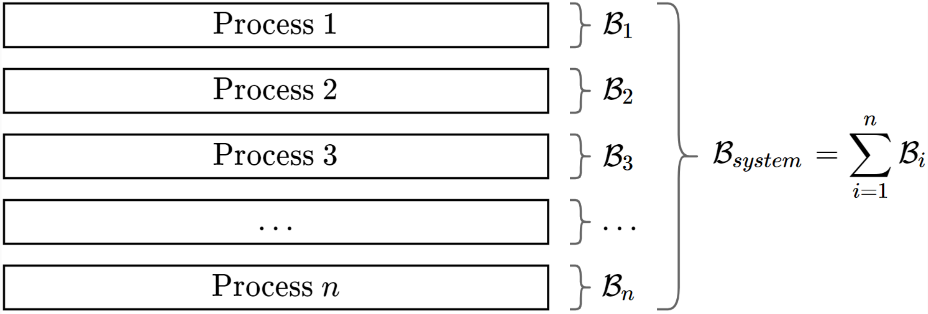


Figure: Increasing Mini-Batch size $\mathcal{B}_{system}$ for Parallel Computing

Update Rules Second-Order Optimization and FIM

Loss Term: $\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell\left(\hat{\mathbf{y}}_i, \mathbf{y}_i\right) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell\left(F\left(\mathbf{x}_i; \boldsymbol{\theta}\right), \mathbf{y}_i\right)$

SGD Update Rule:

NGD Update Rule:

Fisher Information matrix:

Figure: Introduction to Notation and Update Rules

Update Rules Second-Order Optimization and FIM

Loss Term: $\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell\left(\hat{\mathbf{y}}_i, \mathbf{y}_i\right) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell\left(F\left(\mathbf{x}_i; \boldsymbol{\theta}\right), \mathbf{y}_i\right)$

SGD Update Rule: $\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}; (\mathbf{x}_i, \mathbf{y}_i))$

NGD Update Rule:

Fisher Information matrix:

Figure: Introduction to Notation and Update Rules

Loss Term: $$\mathcal{L}(\boldsymbol{\theta};\mathcal{T}) = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{T}} \ell\left(\hat{\mathbf{y}}_i,\mathbf{y}_i\right) = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{T}} \ell\left(F\left(\mathbf{x}_i;\boldsymbol{\theta}\right),\mathbf{y}_i\right)$$

SGD Update Rule: $$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)};(\mathbf{x}_i,\mathbf{y}_i))$$

NGD Update Rule: $$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \mathbf{F}_{\boldsymbol{\theta}}^{-1} \cdot \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)};\cdot)$$

Fisher Information matrix:

Figure: Introduction to Notation and Update Rules

Loss Term: $\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell\left(\hat{\mathbf{y}}_i, \mathbf{y}_i\right) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell\left(F\left(\mathbf{x}_i; \boldsymbol{\theta}\right), \mathbf{y}_i\right)$

SGD Update Rule: $\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}; (\mathbf{x}_i, \mathbf{y}_i))$

NGD Update Rule: $\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \mathbf{F}_{\boldsymbol{\theta}}^{-1} \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}; \cdot)$

Fisher Information matrix: $\mathbf{F}_{\boldsymbol{\theta}} = \mathop{\mathbb{E}}_{p(\mathbf{x}, \mathbf{y})} \left[\nabla \log p(\mathbf{y} | \mathbf{x}; \boldsymbol{\theta}) \nabla \log p(\mathbf{y} | \mathbf{x}; \boldsymbol{\theta})^T \right]$

Figure: Introduction to Notation and Update Rules

Update Rules Second-Order Optimization and FIM

Loss Term: $$\mathcal{L}(\boldsymbol{\theta};\mathcal{T}) = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{T}} \ell\left(\hat{\mathbf{y}}_i,\mathbf{y}_i\right) = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{T}} \ell\left(F\left(\mathbf{x}_i;\boldsymbol{\theta}\right),\mathbf{y}_i\right)$$

SGD Update Rule: $$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta\cdot\ \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)};(\mathbf{x}_i,\mathbf{y}_i))$$

NGD Update Rule: $$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta\cdot\boxed{\boxed{\mathbf{F}_{\boldsymbol{\theta}}^{-1}}\cdot\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)};\cdot)}$$

Fisher Information matrix: $$\mathbf{F}_{\boldsymbol{\theta}} = \mathop{\mathbb{E}}_{p(\mathbf{x},\mathbf{y})}\left[\nabla\log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})\nabla\log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})^T\right]$$

Figure: Introduction to Notation and Update Rules

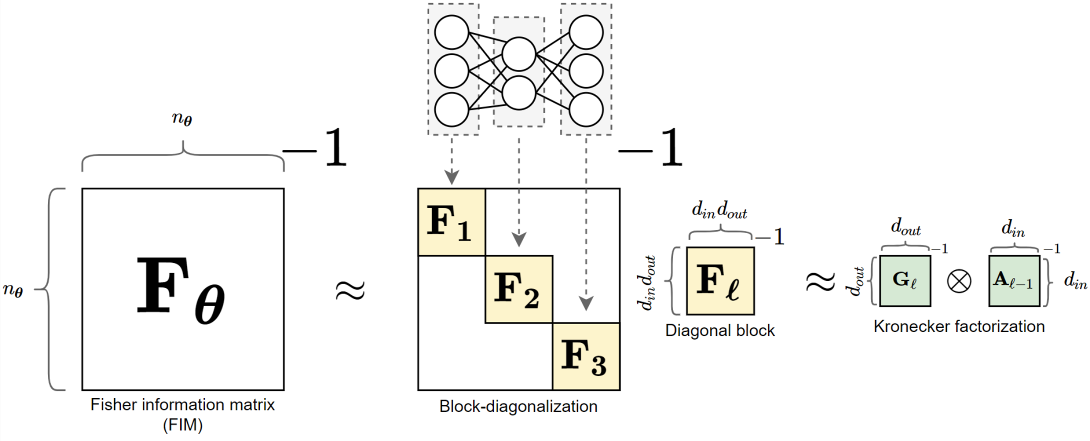


Figure: Approximation of the Fisher Information matrix alternated from: [Osa18]

Approximation of the Fisher Information matrix example: [KSH12, Osa18]

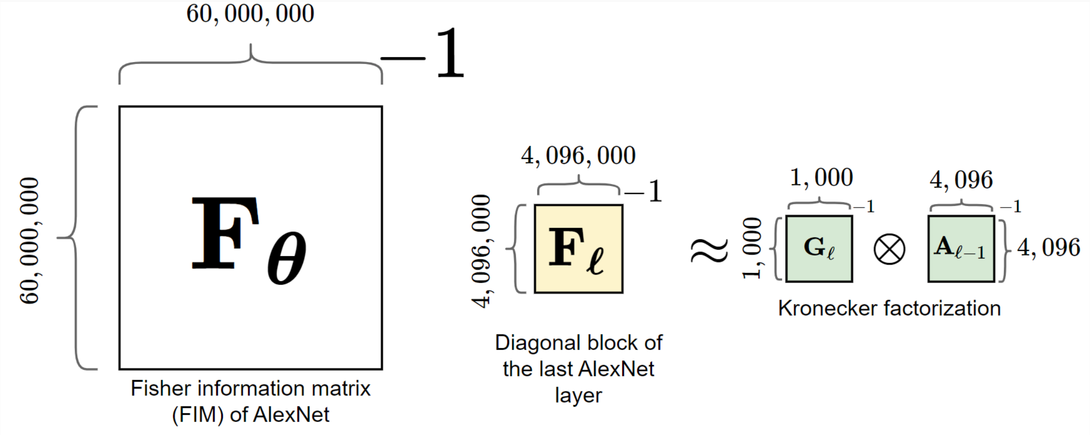
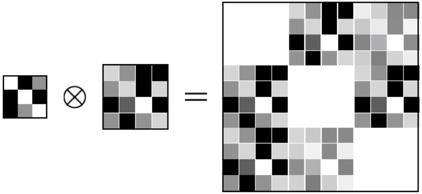


Figure: Approximation of the Fisher Information matrix for AlexNet alternated from: [Osa18]

Kronecker Product: [MG15]

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\,\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{m,n}\,\mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$

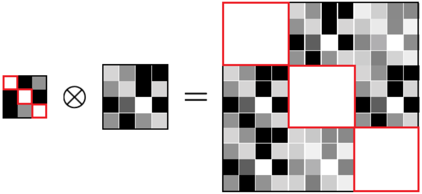


$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

Figure: Visualized Kronecker Product alternated from: [Osa18]

Kronecker Product: [MG15]

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\,\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{m,n}\,\mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$

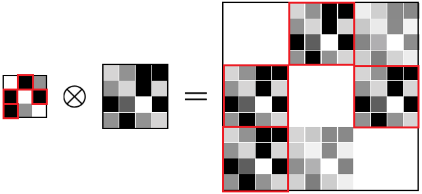


$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

Figure: Visualized Kronecker Product alternated from: [Osa18]

Kronecker Product: [MG15]

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\,\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{m,n}\,\mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$

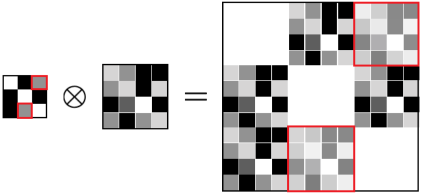


$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

Figure: Visualized Kronecker Product alternated from: [Osa18]

Kronecker Product: [MG15]

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\,\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1}\,\mathbf{B} & \cdots & [\mathbf{A}]_{m,n}\,\mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$



$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

Figure: Visualized Kronecker Product alternated from: [Osa18]

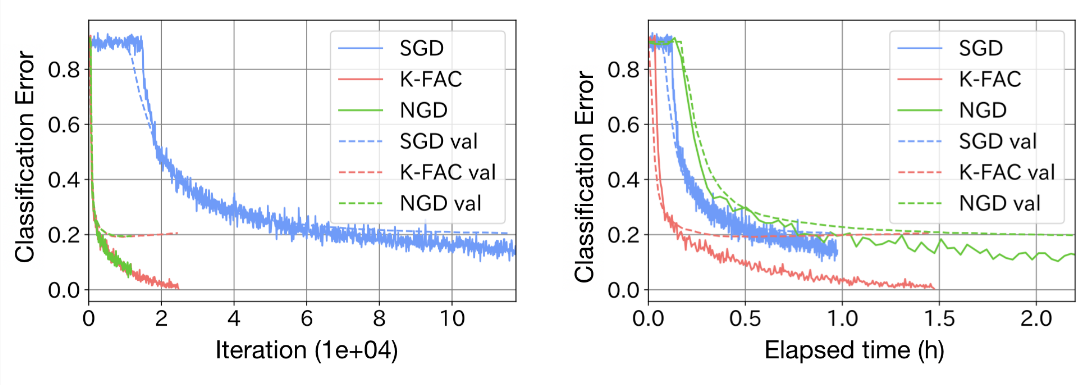


Figure: Comparison of training of ConvNet for CIFAR-10 dataset. Solid line - train, dashed line - validation: [Osa18]

Why is K-FAC a good choice?

It does a good job approximating the FIM and therefore is definitely way more efficient than other second-order techniques

Proposed Parallelized K-FAC Overview: [OTU+18]

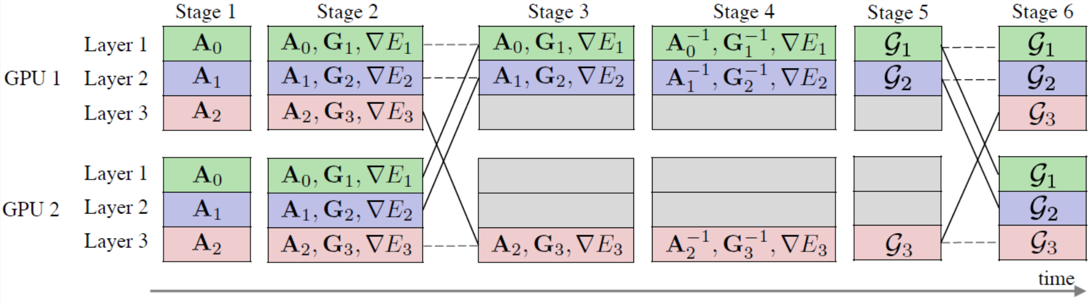


Figure: Proposed Parallelized K-FAC Overview: [OTU+18]

Results: [OTU+18]

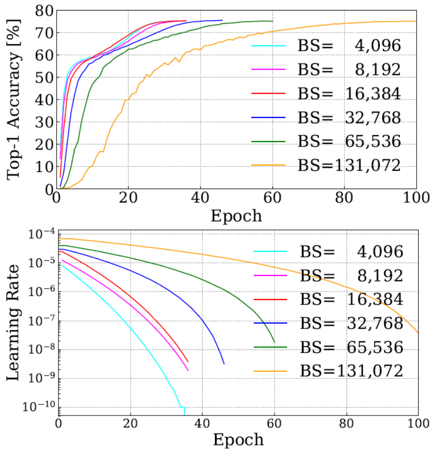


Figure: Accuracy & Learning rate of Parallelized K-FAC with different Batch sizes: [OTU+18]

Results: [OTU$^+$18]

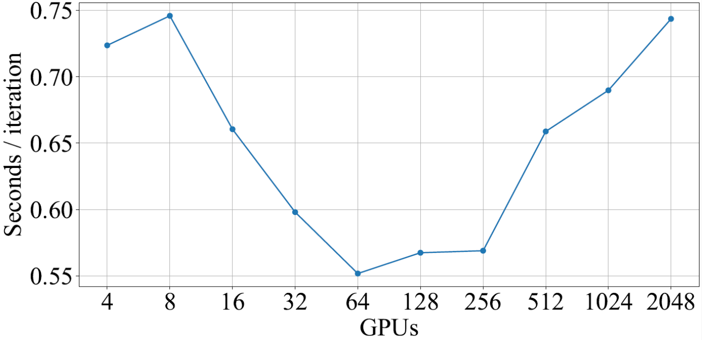


Figure: Iteration cost of Parallelized K-FAC with different amount of GPUs: [OTU$^+$18]

Results: [OTU$^+$18]

	Hardware	Software	Mini-batch size	Optimizer	Iteration	Time	Accuracy
Goyal *et al.* [9]	Tesla P100 × 256	Caffe2	8,192	SGD	14,076	1 hr	76.3%
You *et al.* [29]	KNL × 2048	Intel Caffe	32,768	SGD	3,519	20 min	75.4%
Akiba *et al.* [3]	Tesla P100 × 1024	Chainer	32,768	RMSprop/SGD	3,519	15 min	74.9%
You *et al.* [29]	KNL × 2048	Intel Caffe	32,768	SGD	2,503	14 min	74.9%
Jia *et al.* [15]	Tesla P40 × 2048	TensorFlow	65,536	SGD	1,800	6.6 min	75.8%
Ying *et al.* [28]	TPU v3 × 1024	TensorFlow	32,768	SGD	3,519	2.2 min	76.3%
Mikami *et al.* [22]	Tesla V100 × 3456	NNL	55,296	SGD	2,086	2.0 min	75.3%
This work (Sec. 5.4)	Tesla V100 × 1024	Chainer	32,768	K-FAC	1,760	10 min	74.9%
This work (Sec. 5.3)	-	Chainer	131,072	K-FAC	978	-	75.0%

Figure: Training iterations (time) and top-1 single-crop validation accuracy of ResNet-50 for ImageNet reported by related work: [OTU$^+$18]

References I

[Ama98] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, February 1998.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[LH17] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam, 2017.

[LJH+19] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2019.

[LXLS19] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate, 2019.

[MG15] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2015.

[Osa18] Kazuki Osawa. Introducing k-fac: A second-order optimization method for large-scale deep learning, 2018.

[OTU+18] Kazuki Osawa, Yohei Tsuji, Yuichiro Ueno, Akira Naruse, Rio Yokota, and Satoshi Matsuoka. Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks, 2018.

[RKK19] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[Sch02] Nicol Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14:1723–38, 08 2002.

[SLA+18] Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training, 2018.

[ZLHB19] Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back, 2019.