“KAISA: An Adaptive Second-Order Optimizer Framework for Deep Neural Networks”

HPC 2021 # 61

Nikola Janjušević, Feb 6th 2023

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

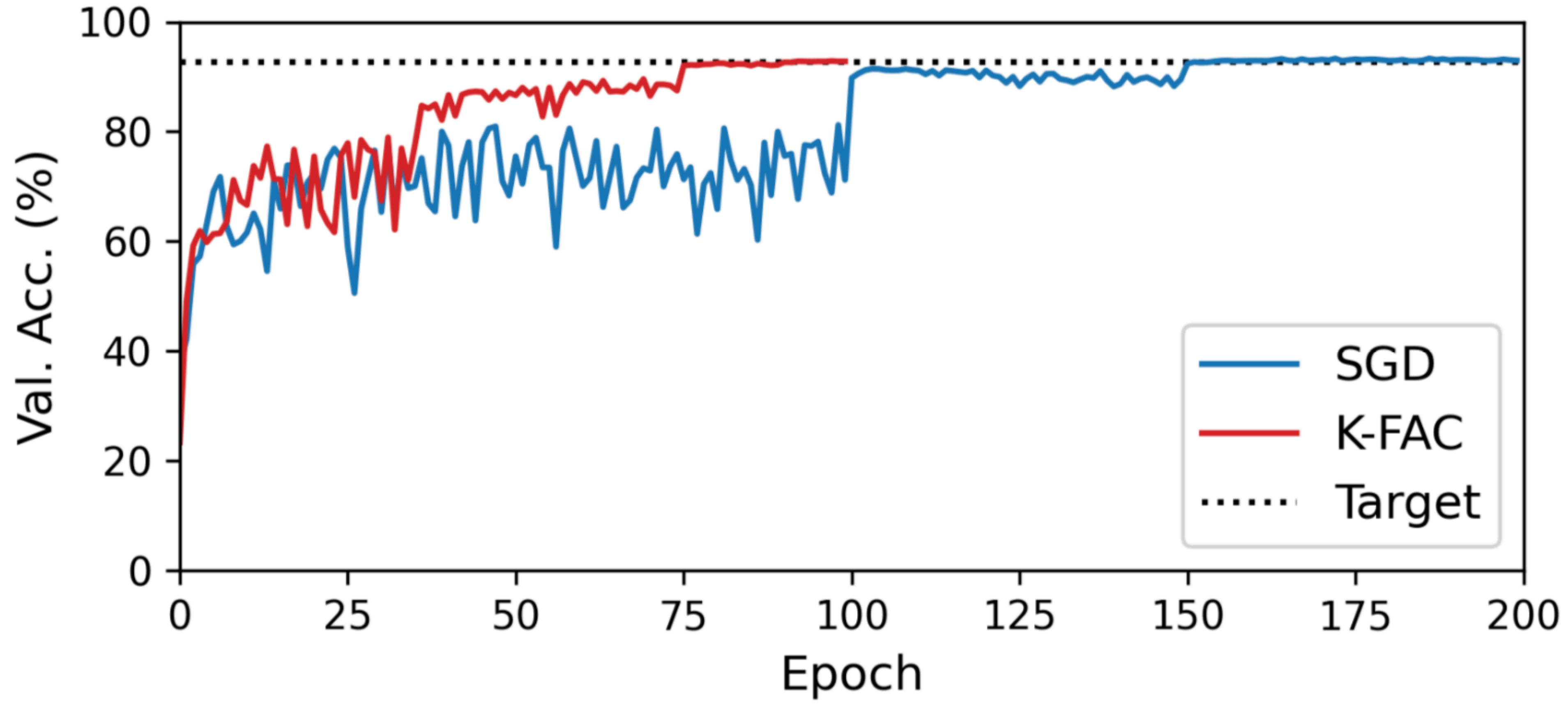


Figure 1: SGD vs. K-FAC training for ResNet-32 with the CIFAR-10 dataset. K-FAC reduces iterations needed for convergence.

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

$$\text{SGD: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)}}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{1}$$

$$\text{K-FAC: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)} F^{-1}(w^{(k)})}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{2}$$

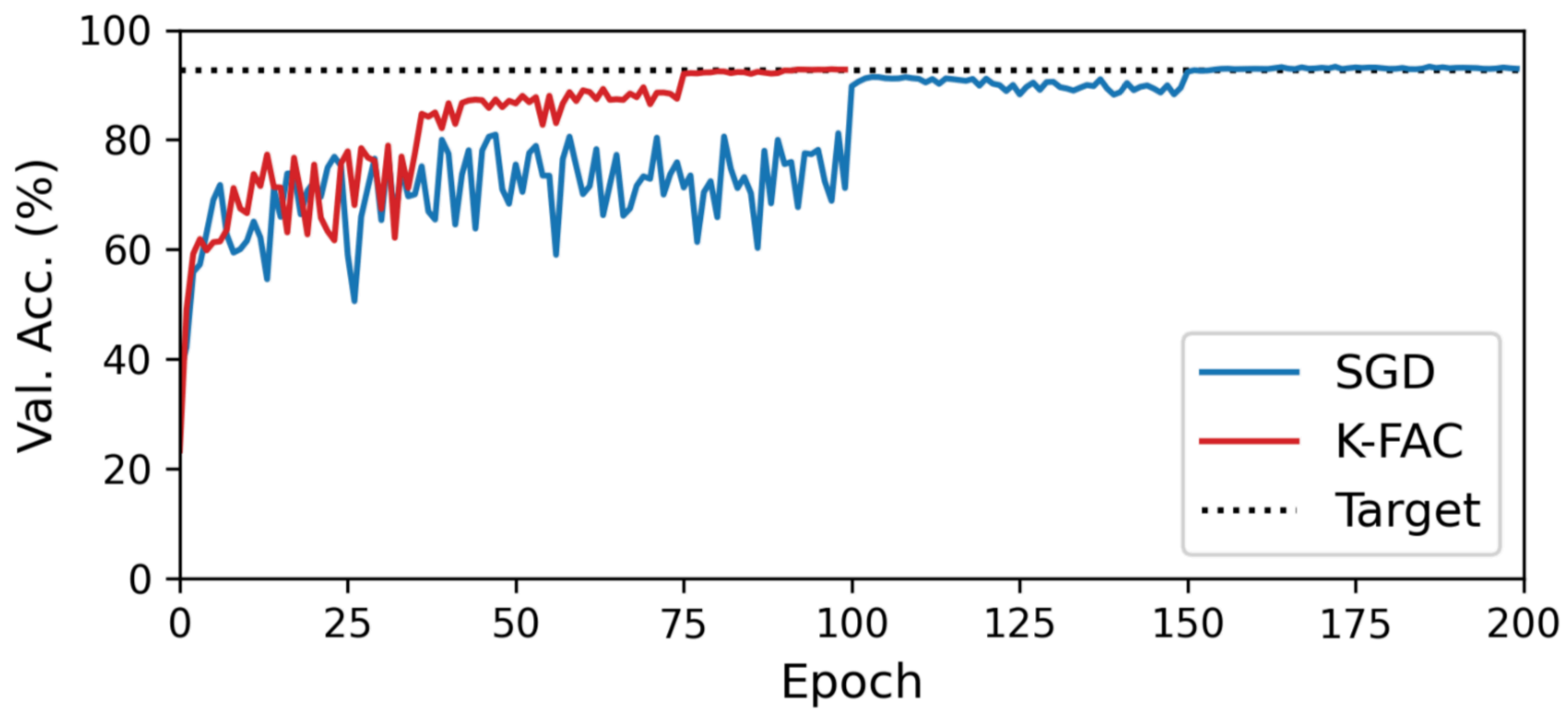


Figure 1: SGD vs. K-FAC training for ResNet-32 with the CIFAR-10 dataset. K-FAC reduces iterations needed for convergence.

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

$$\text{SGD: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)}}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{1}$$

$$\text{K-FAC: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)} F^{-1}(w^{(k)})}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{2}$$

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

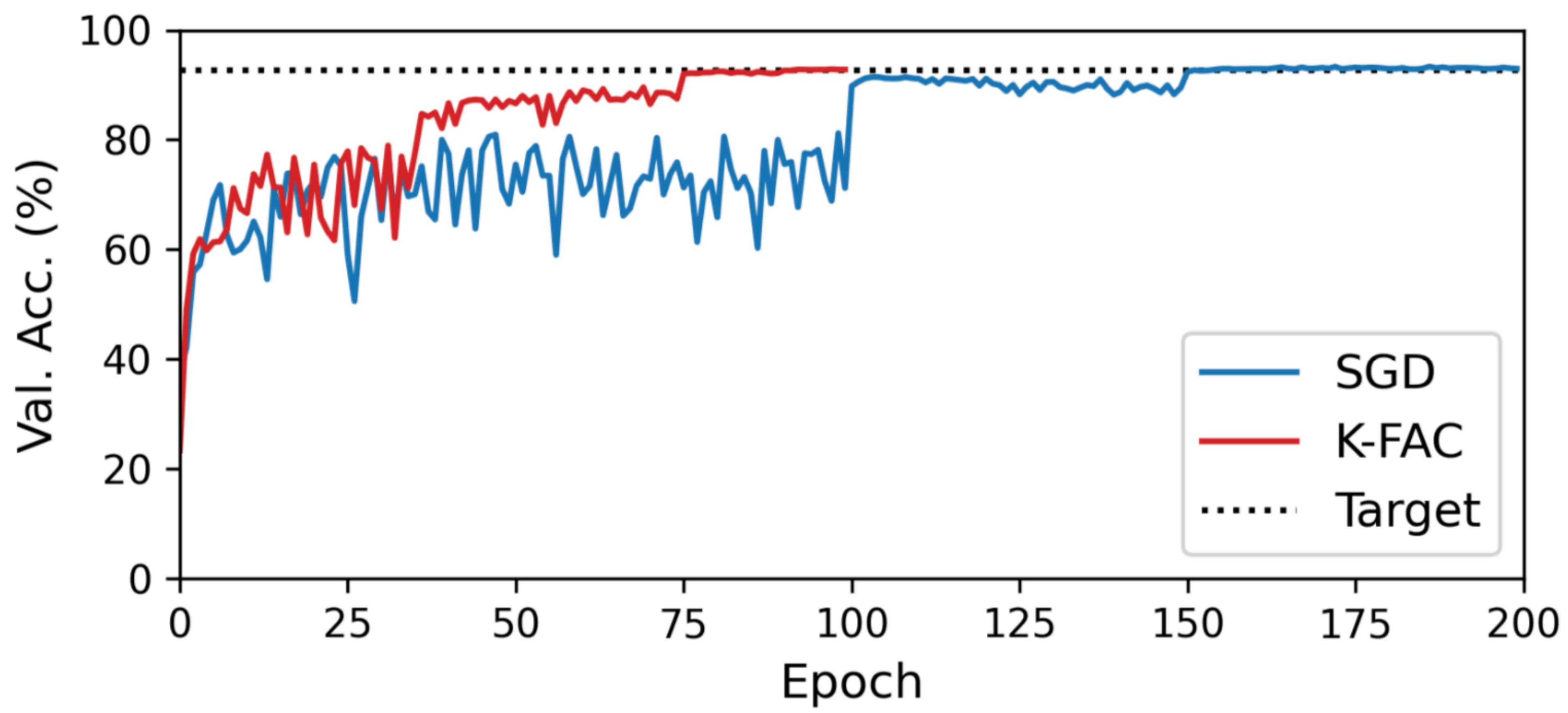


Figure 1: SGD vs. K-FAC training for ResNet-32 with the CIFAR-10 dataset. K-FAC reduces iterations needed for convergence.

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

$$\text{SGD: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)}}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{1}$$

$$\text{K-FAC: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)} F^{-1}(w^{(k)})}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{2}$$

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$$
$$(A \otimes B)\vec{c} = B^{\top}\vec{c}A.$$

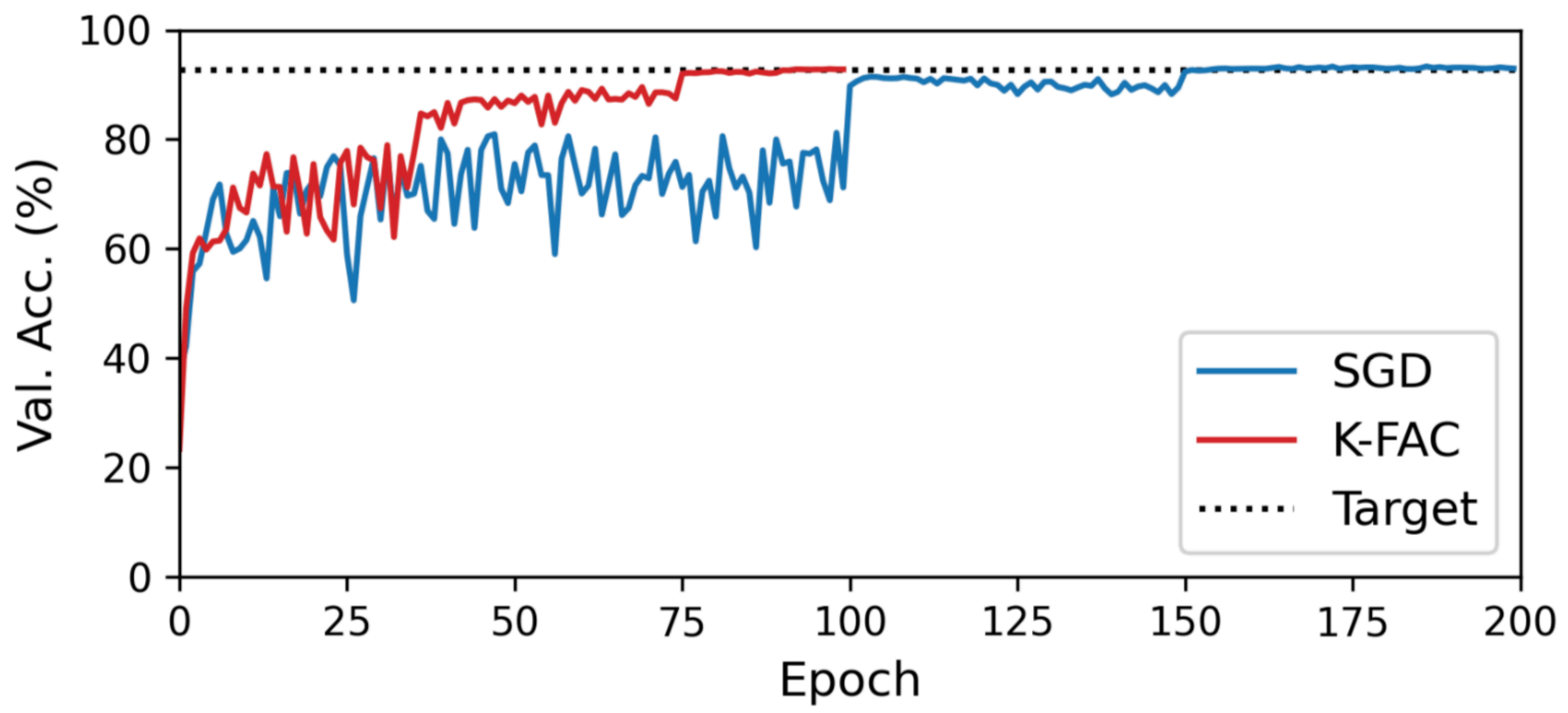


Figure 1: SGD vs. K-FAC training for ResNet-32 with the CIFAR-10 dataset. K-FAC reduces iterations needed for convergence.

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

$$\text{SGD: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)}}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{1}$$

$$\text{K-FAC: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)} F^{-1}(w^{(k)})}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{2}$$

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$$
$$(A \otimes B)\vec{c} = B^\top \vec{c} A.$$

$$F_{i,j} \approx a_{i-1} a_{j-1}^\top \otimes g_i g_j^\top.$$

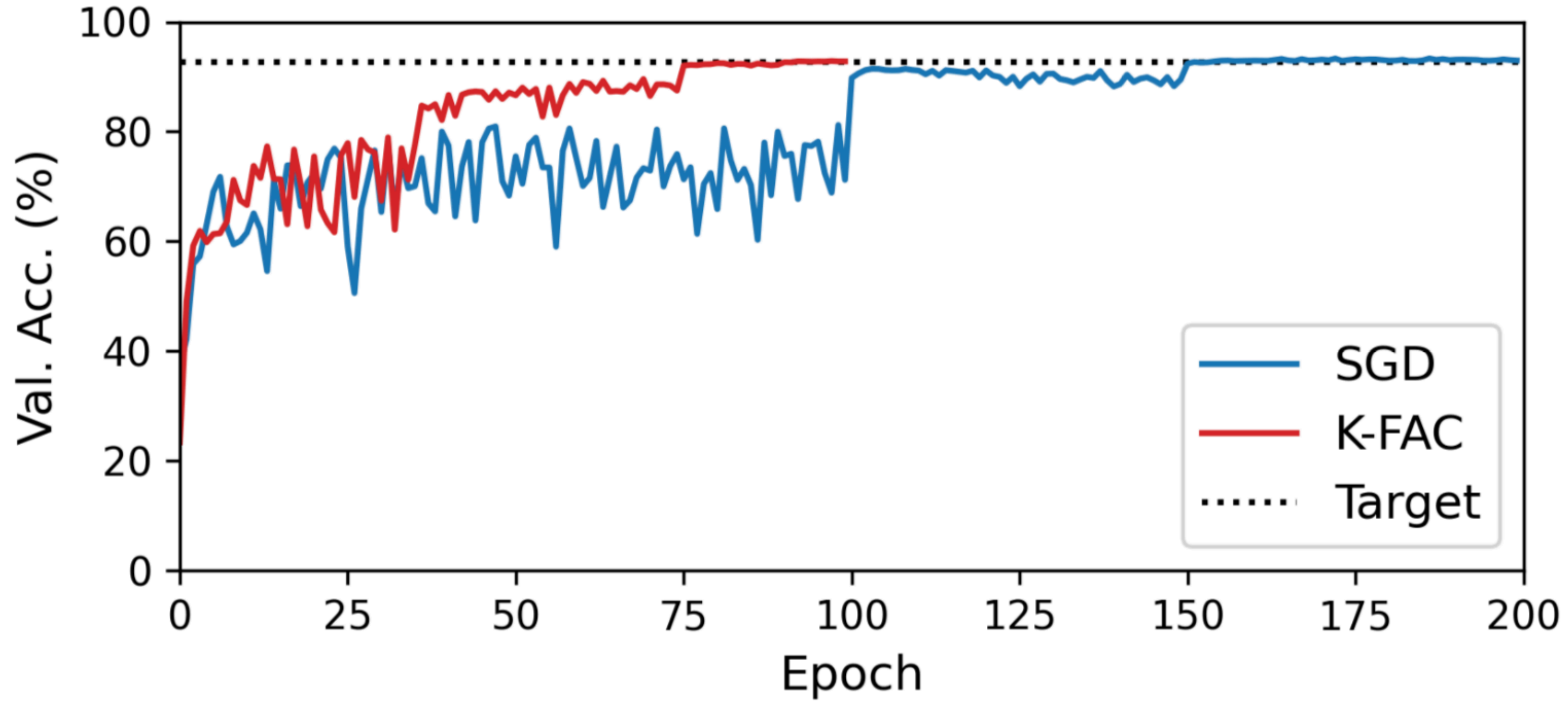


Figure 1: SGD vs. K-FAC training for ResNet-32 with the CIFAR-10 dataset. K-FAC reduces iterations needed for convergence.

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

$$\text{SGD: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)}}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{1}$$

$$\text{K-FAC: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)} F^{-1}(w^{(k)})}{n} \sum_{i=1}^{n} \nabla L_i(w^{(k)}) \tag{2}$$

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$$

$$(A \otimes B)\vec{c} = B^{\top}\vec{c}A.$$

$$F_{i,j} \approx a_{i-1}a_{j-1}^{\top} \otimes g_i g_j^{\top}.$$

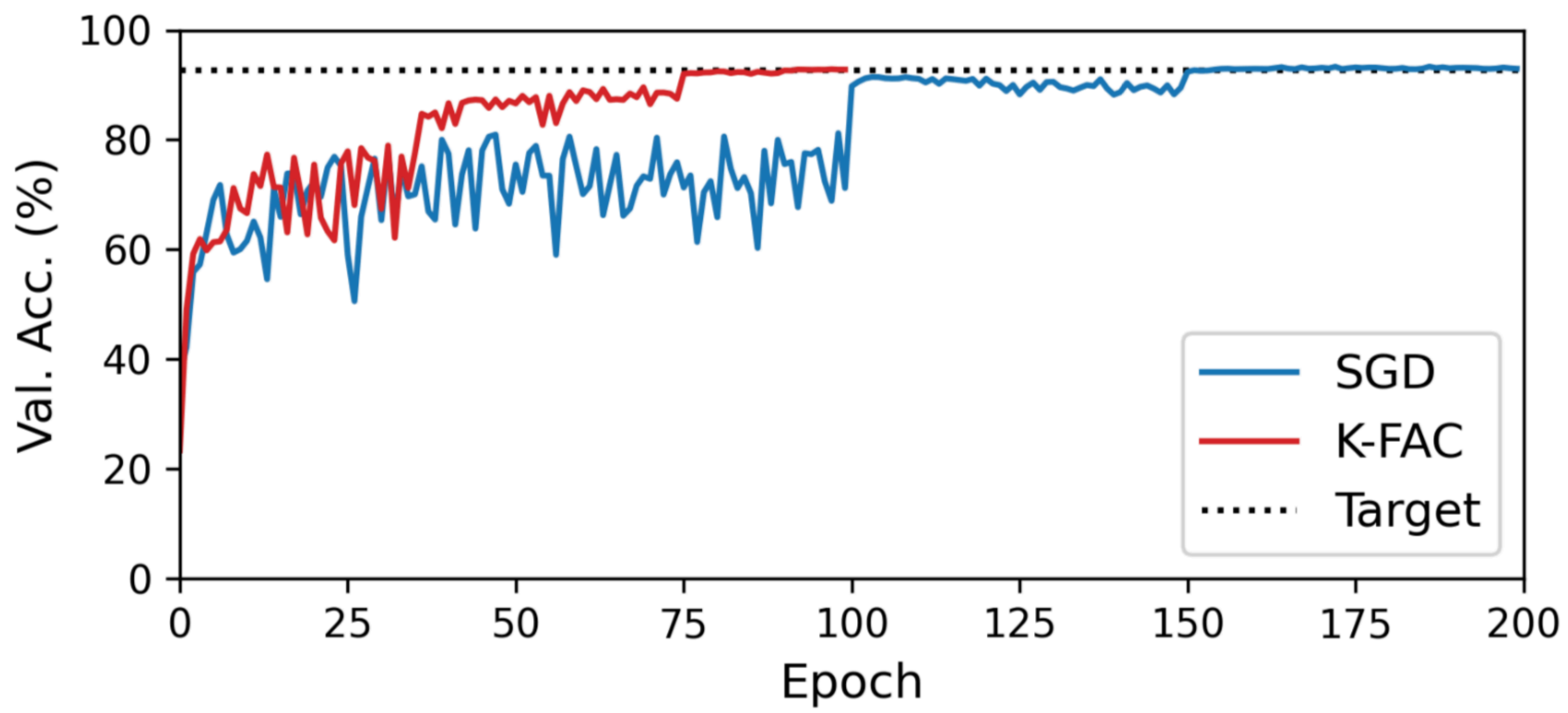


Figure 1: SGD vs. K-FAC training for ResNet-32 with the CIFAR-10 dataset. K-FAC reduces iterations needed for convergence.

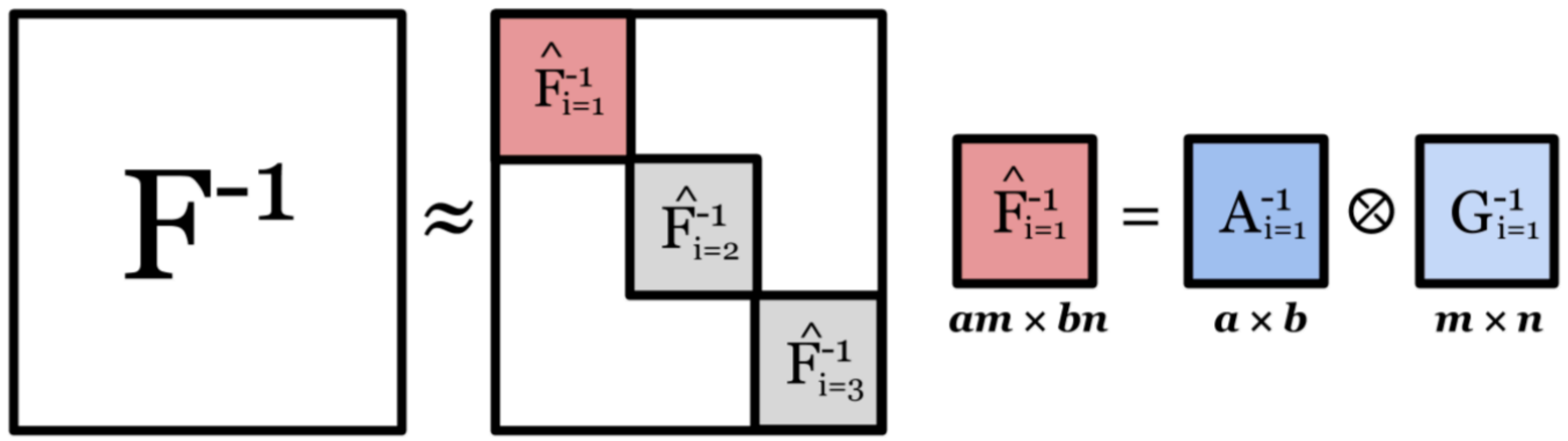


Figure 2: The K-FAC approximation of the Fisher information matrix. ⊗ is the Kronecker product [44].

Second-Order Optimization

Stochastic Kronecker-Factored Approximate Curvature (KFAC)

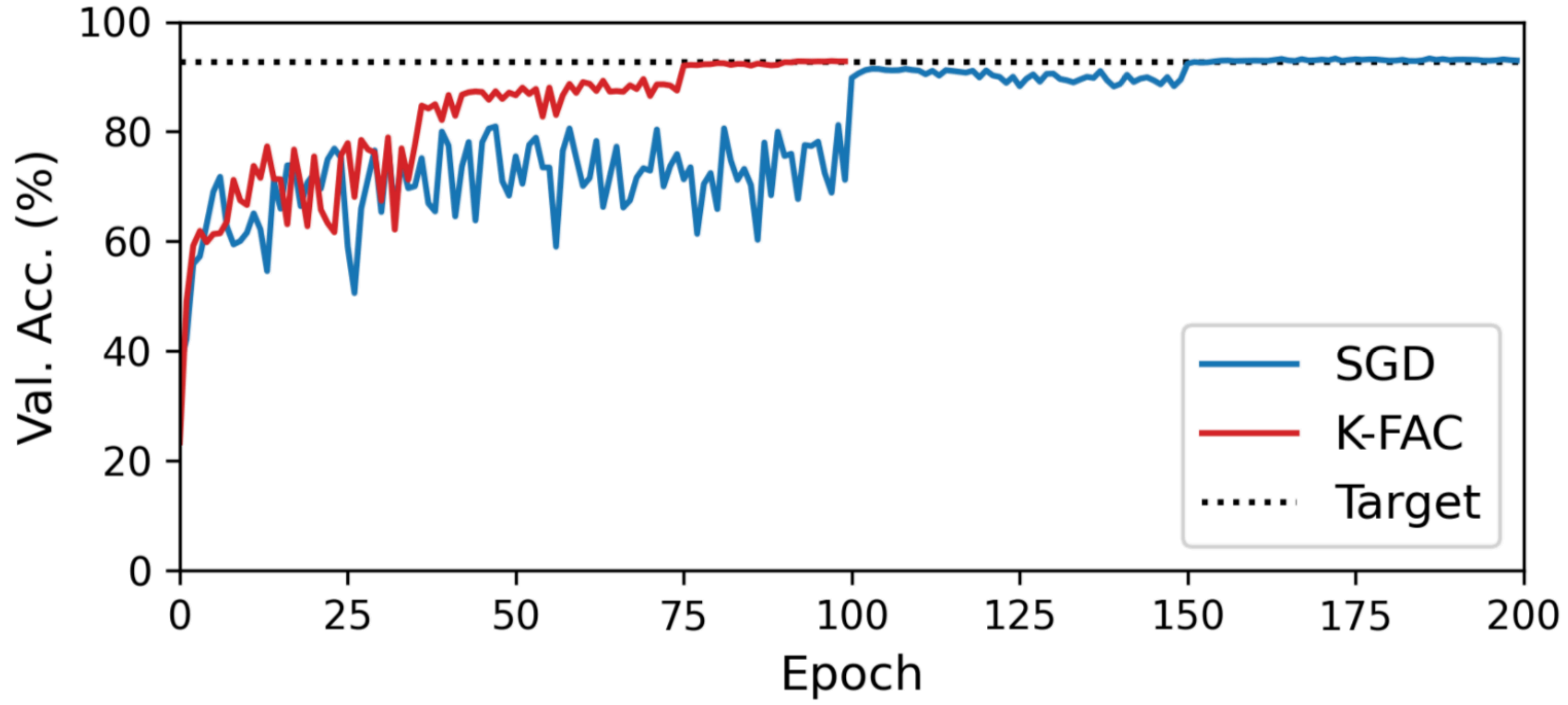


Figure 1: SGD vs. K-FAC training for ResNet-32 with the CIFAR-10 dataset. K-FAC reduces iterations needed for convergence.

$$\text{SGD: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)}}{n}\sum_{i=1}^{n}\nabla L_i(w^{(k)}) \tag{1}$$

$$\text{K-FAC: } w^{(k+1)} = w^{(k)} - \frac{\alpha^{(k)}F^{-1}(w^{(k)})}{n}\sum_{i=1}^{n}\nabla L_i(w^{(k)}) \tag{2}$$

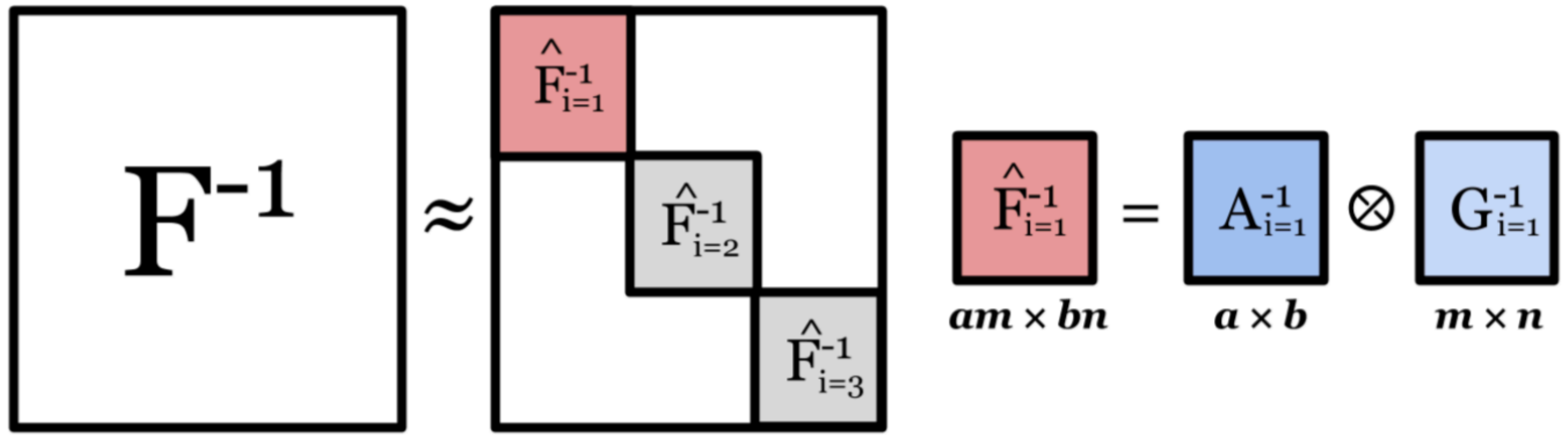


Figure 2: The K-FAC approximation of the Fisher information matrix. ⊗ is the Kronecker product [44].

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$$

$$(A \otimes B)\vec{c} = B^{\top}\vec{c}A.$$

$$F_{i,j} \approx a_{i-1}a_{j-1}^{\top} \otimes g_i g_j^{\top}.$$

$$w_i^{(k+1)} = w_i^{(k)} - \alpha^{(k)}(\hat{F}_i + \gamma I)^{-1}\nabla L_i(w_i^{(k)}) \tag{13}$$

$$= w_i^{(k)} - \alpha^{(k)}(G_i + \gamma I)^{-1}\nabla L_i(w_i^{(k)})(A_{i-1} + \gamma I)^{-1}. \tag{14}$$

Distributed Implementation

MEM, COMM, HYBRID - OPT

- Workers n=1,2,...,N
  - *each have copy of entire DNN*
- DNN layers i=1,2,...,M
- Gradient-Reduce:

$$\nabla L_n \rightarrow \nabla L = \frac{1}{N}\sum_{n=1}^{N} \nabla L_n$$

- Factor-Reduce:

$$A^n_{(i-1)}, G^n_i \rightarrow A_{(i-1)} = \sum_{n=1}^{N} A^n_{(i-1)}, G_i = \sum_{i=1}^{N} G^n_i$$

- PG Broadcast:
  - *compute (14) via eigen-decomp*

Distributed Implementation

MEM, COMM, HYBRID - OPT

- Workers n=1,2,...,N
  - *each have copy of entire DNN*
- DNN layers i=1,2,...,M
- Gradient-Reduce:

$$\nabla L_n \rightarrow \nabla L = \frac{1}{N}\sum_{n=1}^{N} \nabla L_n$$

- Factor-Reduce:

$$A^n_{(i-1)}, G^n_i \rightarrow A_{(i-1)} = \sum_{n=1}^{N} A^n_{(i-1)}, G_i = \sum_{i=1}^{N} G^n_i$$

- PG Broadcast:
  - *compute (14) via eigen-decomp*

- **MEM-OPT**: *worker assigned its own layers for eigen-decomps of* $A_{(i-1)}, G_i$.
  A. *Grad all-reduce*
  B. *Factor all-reduce*
  C. *Eigen-decomp & PG broadcast*

Distributed Implementation

MEM, COMM, HYBRID - OPT

- Workers n=1,2,…,N
  - *each have copy of entire DNN*
- DNN layers i=1,2,…,M
- Gradient-Reduce:

$$\nabla L_n \rightarrow \nabla L = \frac{1}{N}\sum_{n=1}^{N} \nabla L_n$$

- Factor-Reduce:

$$A^n_{(i-1)}, G^n_i \rightarrow A_{(i-1)} = \sum_{n=1}^{N} A^n_{(i-1)}, G_i = \sum_{i=1}^{N} G^n_i$$

- PG Broadcast:
  - *compute (14) via eigen-decomp*

- **MEM-OPT**: *worker assigned its own layers for eigen-decomps of $A_{(i-1)}$, G_i.*
  A. *Grad all-reduce*
  B. *Factor all-reduce*
  C. *Eigen-decomp & PG broadcast*

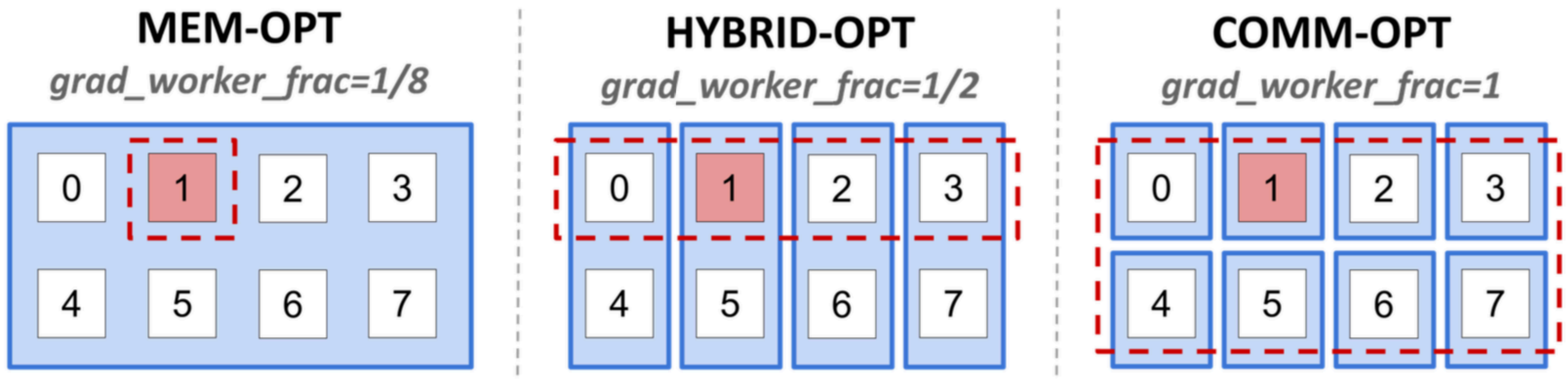
- **COMM-OPT**: $A_{(i-1)}$, G_i *i=1,…,M split across all workers, decomps stored on all workers.*
  A. *Grad all-reduce*
  B. *Factor all-reduce*
  C. *Eigen-decomp broadcast*
  D. *Precondition Gradient*

HYBRID-OPT & results

tunable memory-comm tradeoff

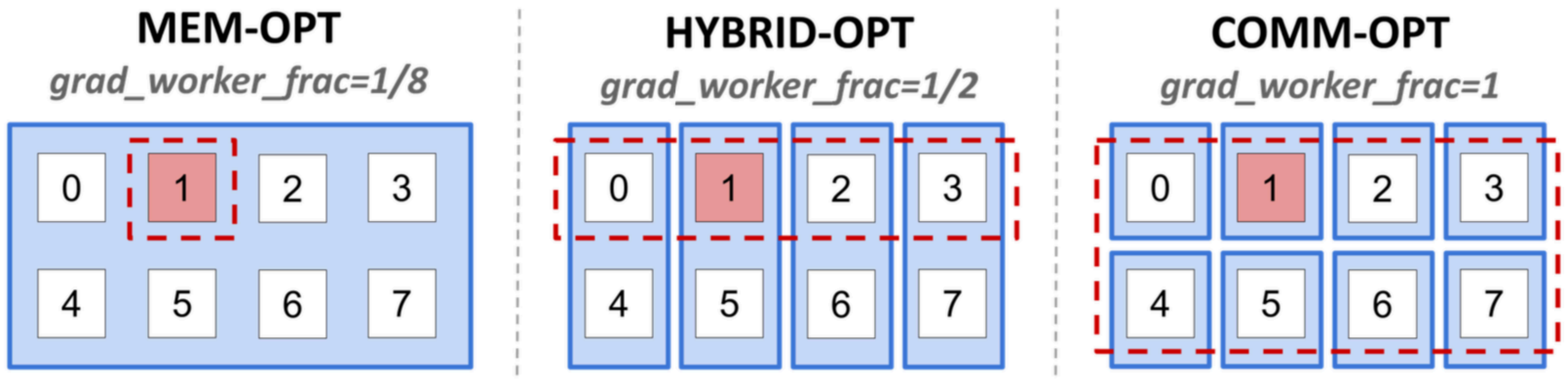
HYBRID-OPT & results

tunable memory-comm tradeoff



HYBRID-OPT & results

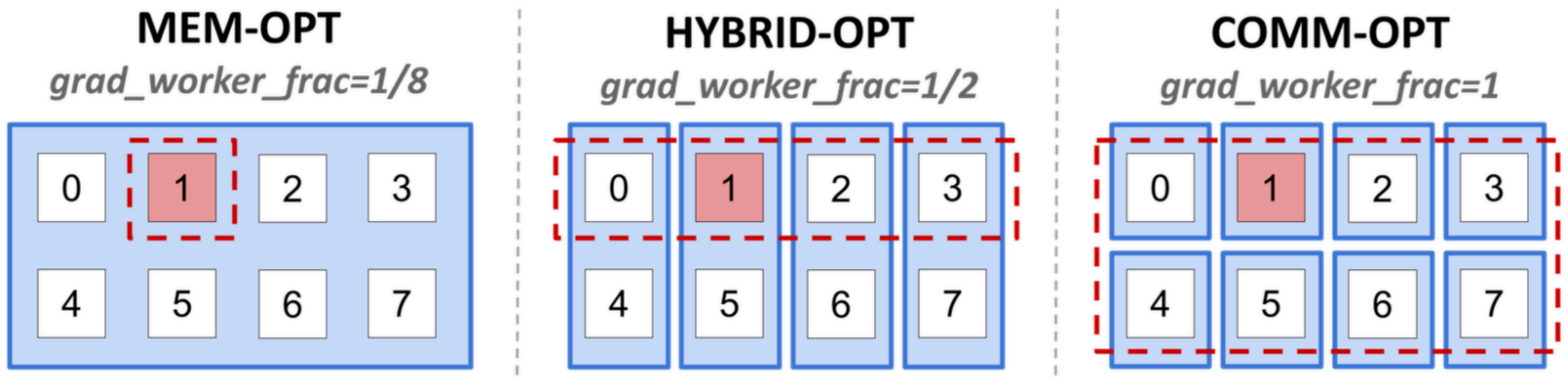
tunable memory-comm tradeoff



"I'll precondition the gradient for you"

HYBRID-OPT & results

tunable memory-comm tradeoff

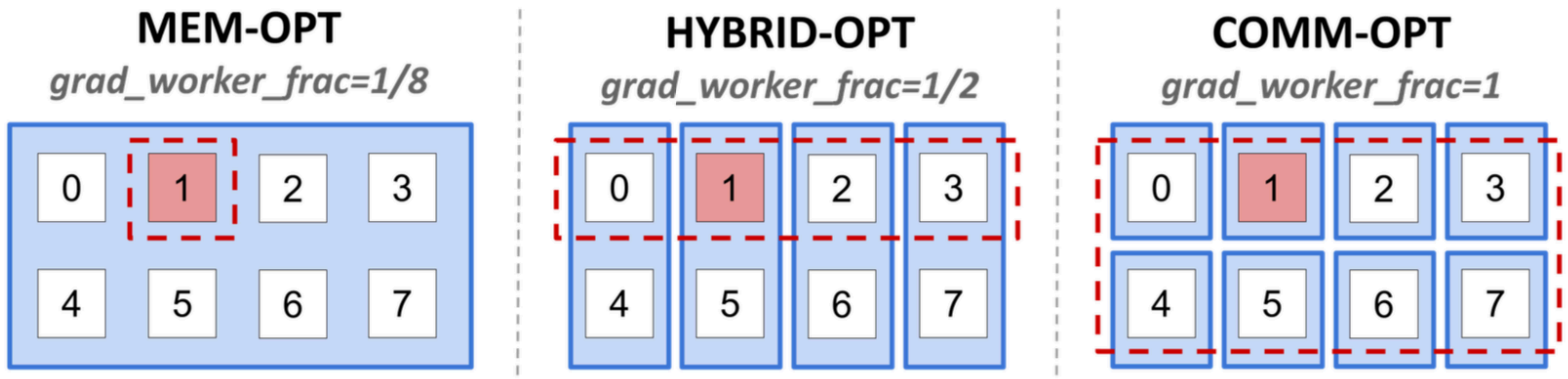


"I'll precondition the gradient for you"

"Go precondition the gradient **yourself**"

HYBRID-OPT & results

tunable memory-comm tradeoff



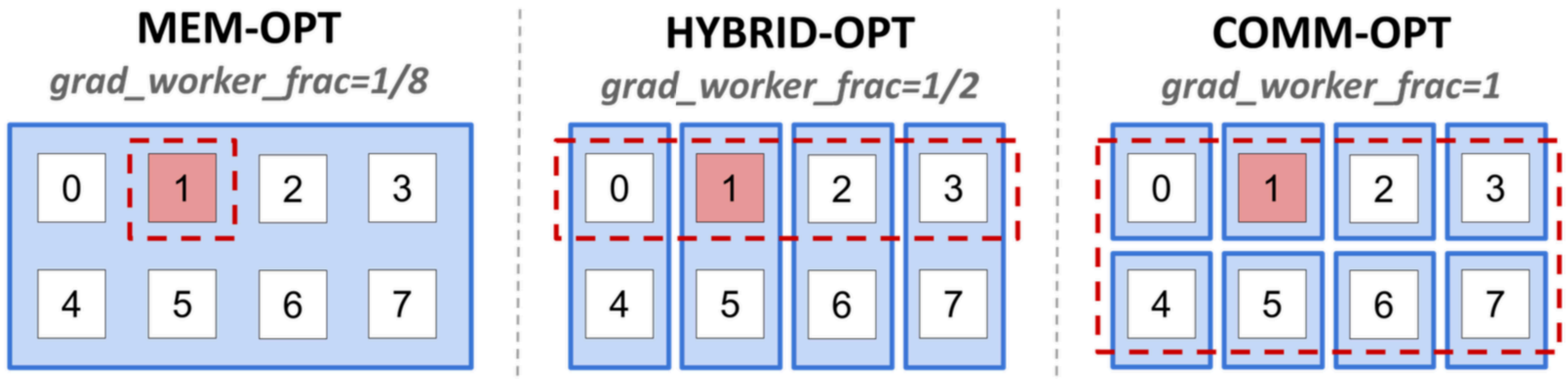
"I'll precondition the gradient for you"

"I'll precondition the gradient for ***some of you*** you"

"Go precondition the gradient **yourself**"

HYBRID-OPT & results

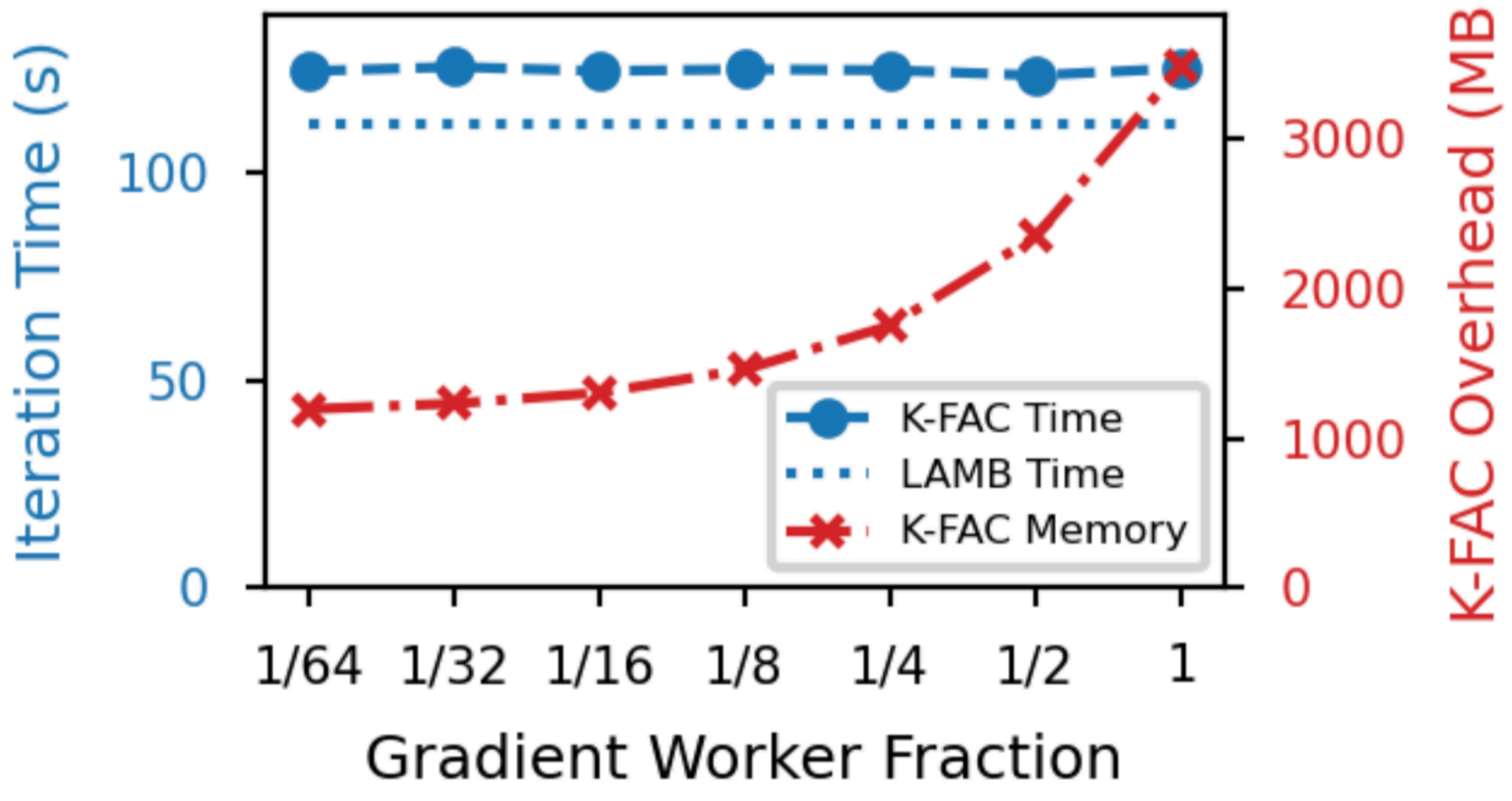
tunable memory-comm tradeoff



"I'll precondition the gradient for you"

"I'll precondition the gradient for ***some of you*** you"

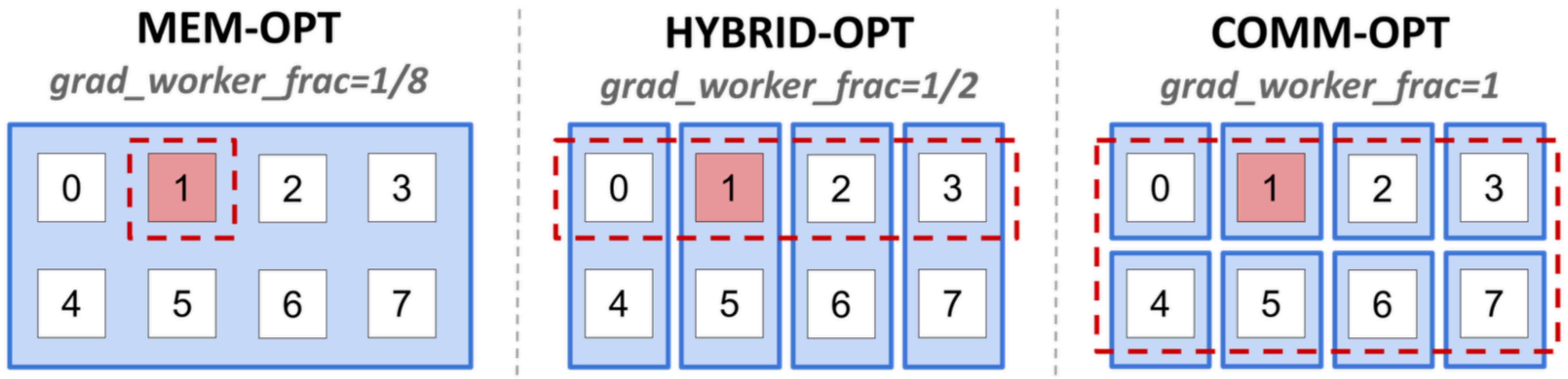
"Go precondition the gradient **yourself**"



(f) BERT-Large Phase 2 (FP16)

HYBRID-OPT & results

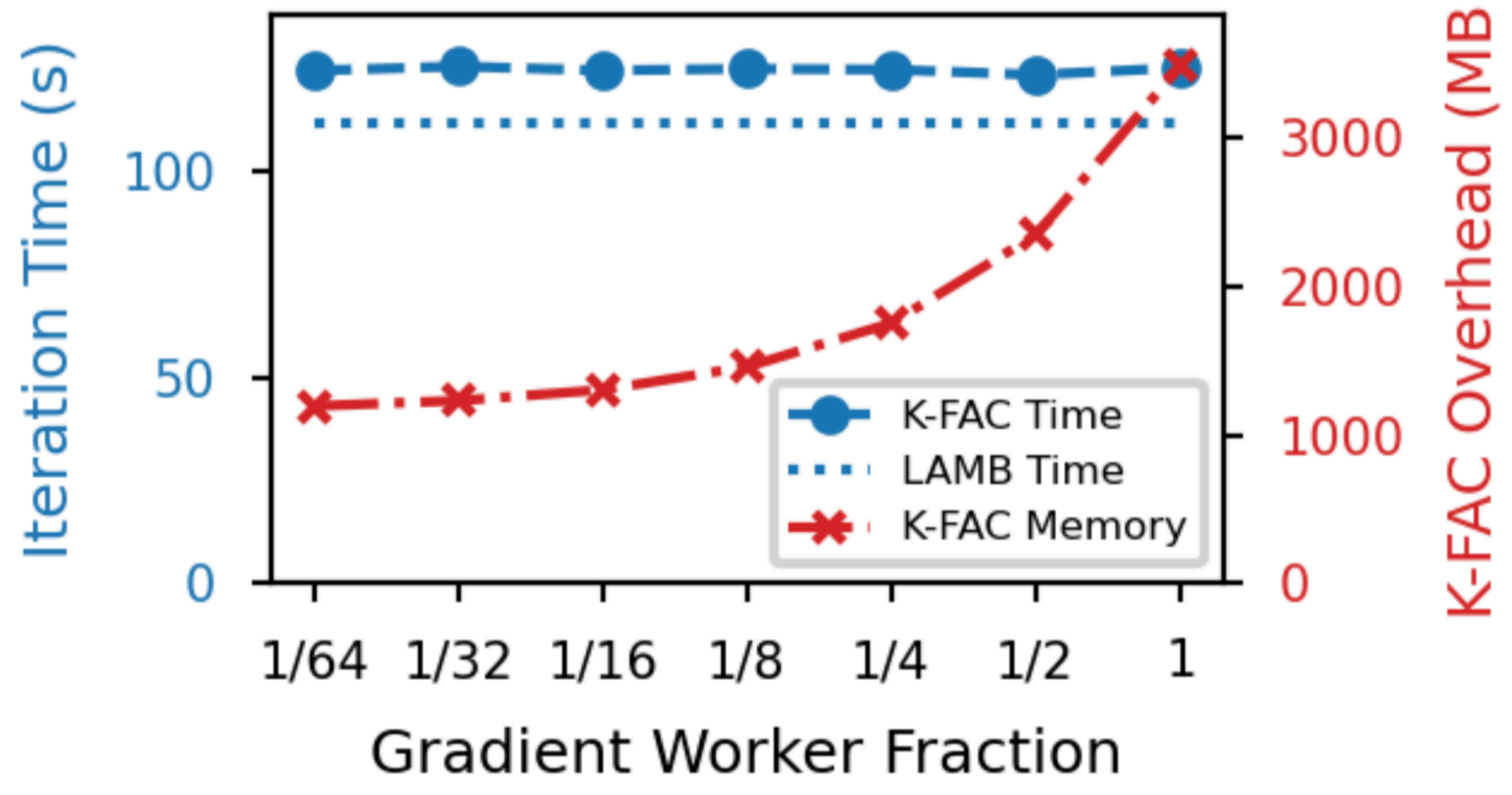
tunable memory-comm tradeoff



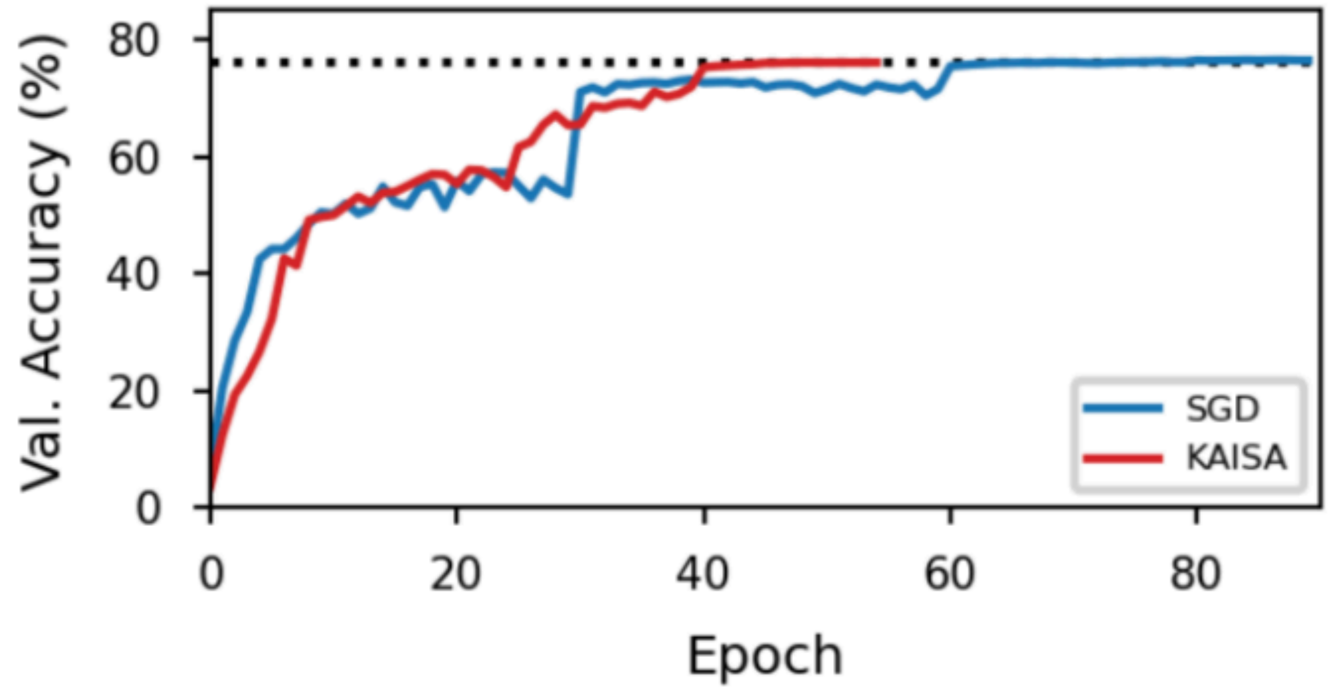
"I'll precondition the gradient for you"

"I'll precondition the gradient for *some of you* you"

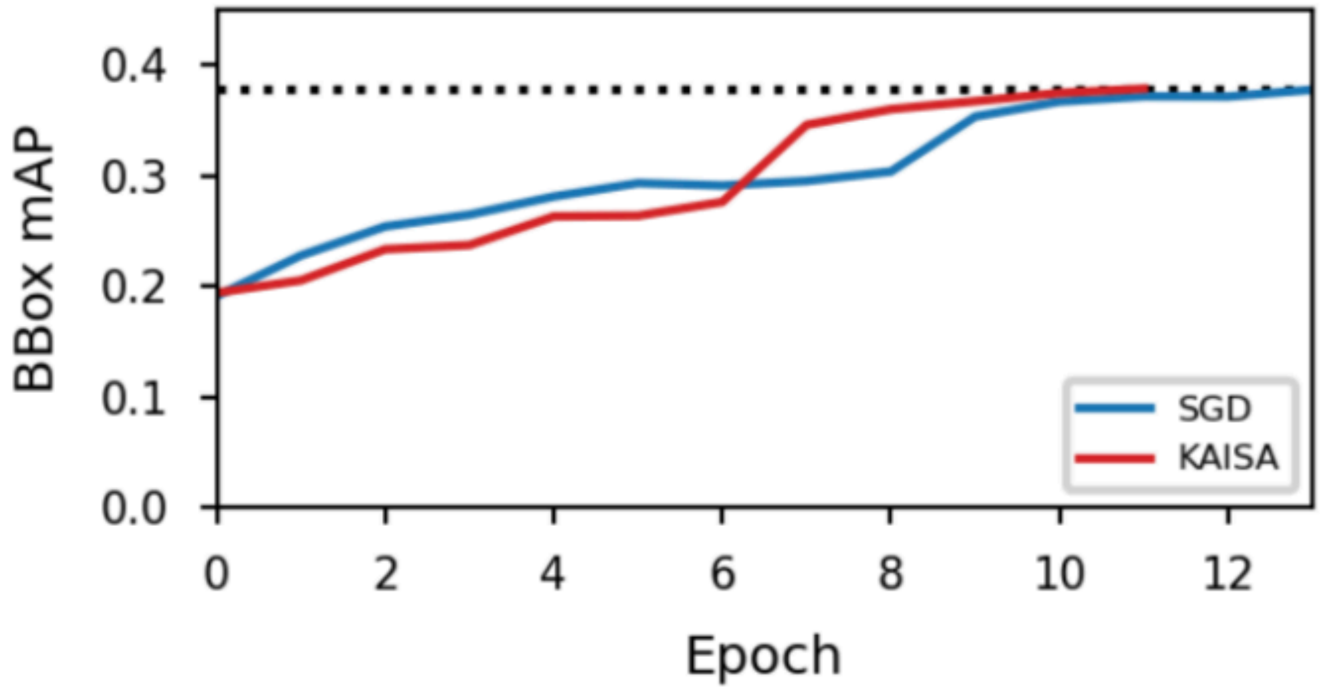
"Go precondition the gradient **yourself**"



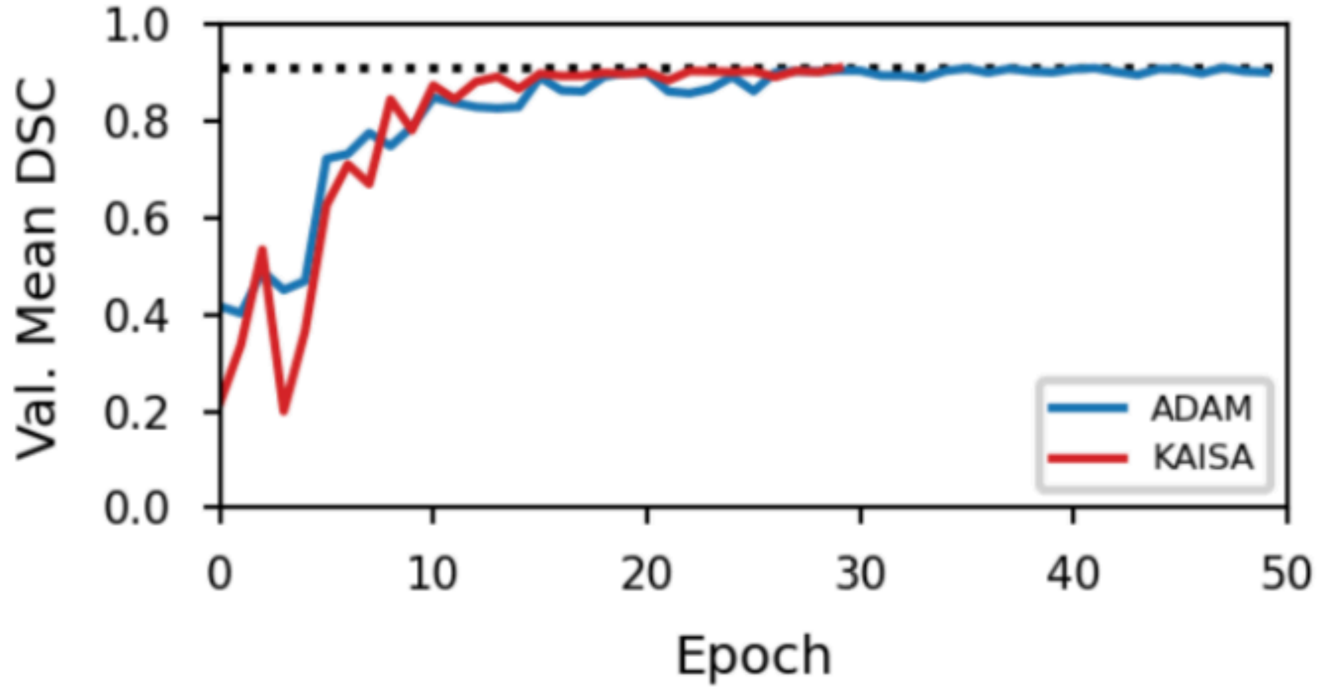
(f) BERT-Large Phase 2 (FP16)



(a) ResNet-50. The time-to-convergence is 268.1 mins for K-FAC and 354.0 mins for momentum SGD.



(b) Mask R-CNN. The time-to-convergence is 115.8 mins for K-FAC and 136.1 for SGD.



(c) U-Net. The time-to-convergence is 10.9 mins for K-FAC and 14.6 for ADAM.