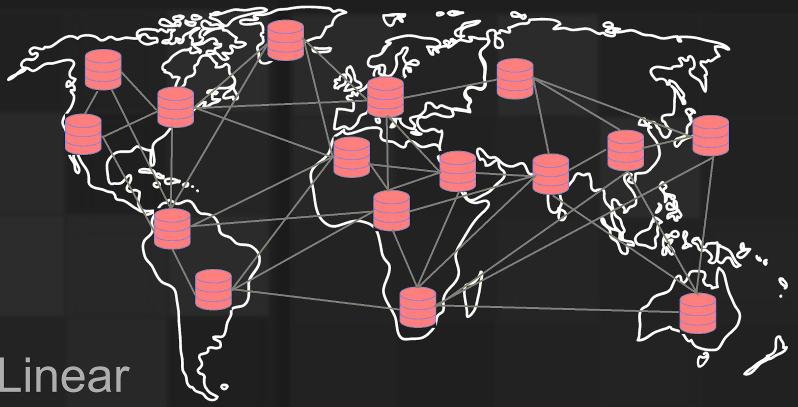
PowerSGD:
Practical Low-Rank Gradient Compression for Distributed Optimization

Thijs Vogels, Sai Praneeth Karimireddy, Martin Jaggi

Why distributed SGD?



Why Distributed?

- Data Privacy
- Resource Distribution
- Data Distribution

Why dSGD?

- Gradients are Linear
- Gradients are Optimal
- Are gradients the smallest statistic?

$$\sum_{s=1}^{S} \nabla_{\mathbf{w}} \mathcal{L}^{(s)} = \nabla_{\mathbf{w}} \mathcal{L}$$

Reducing Bandwidth

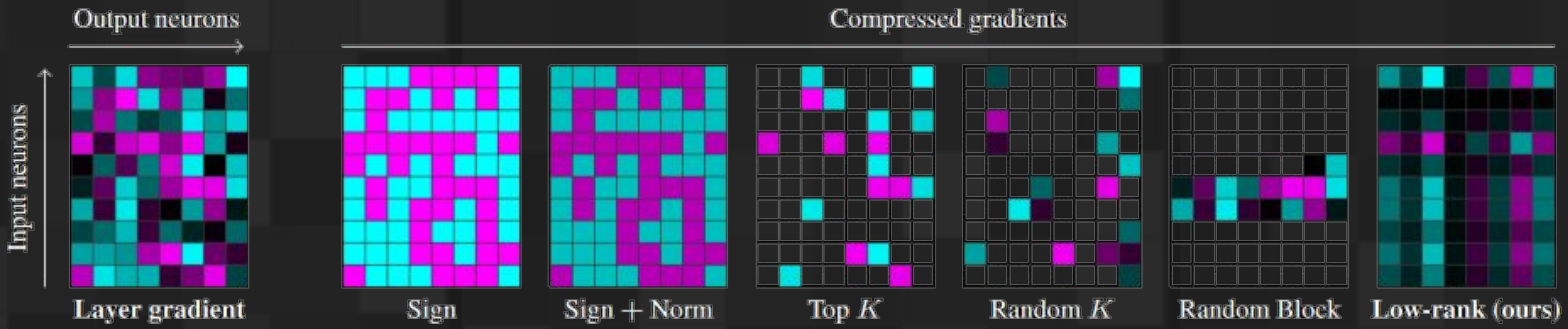
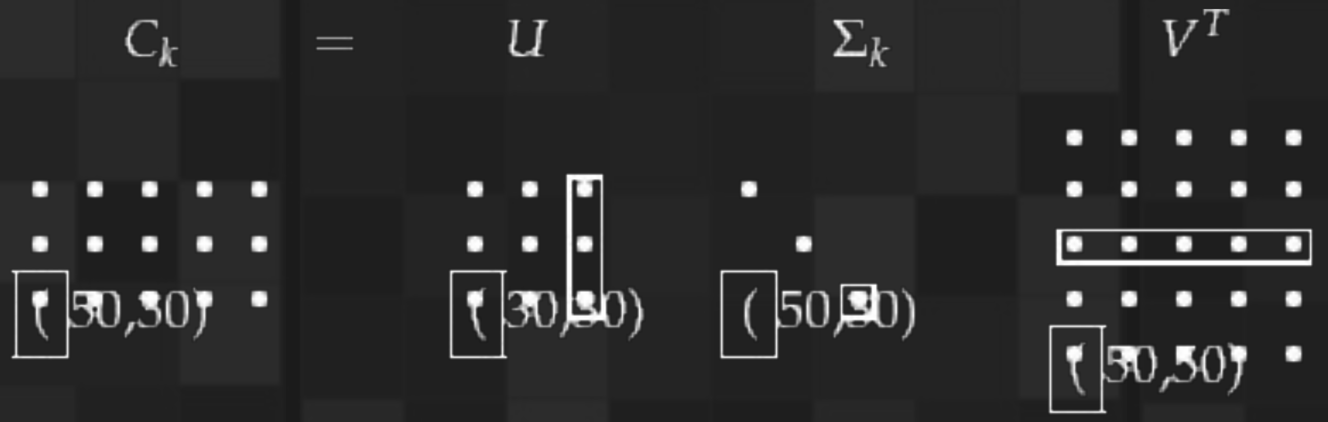


Figure 1: Compression schemes compared in this paper. Left: Interpretation of a layer's gradient as a matrix. Coordinate values are color coded (positive, negative). Right: The output of various compression schemeson the same input. Implementation details arein Appendix G.

Low-Rank Decompositions



► **Figure 18.2** Illustration of low rank approximation using the singular-value decom-position. The dashed boxes indicate the matrix entries affected by "zeroing out" the smallest singular values.

Power Iterations

Algorithm

- initial approximation - random unit vector x_0
- $x_1 = Ax_0$
- $x_2 = AAx_0 = A^2x_0$
- $x_3 = AAAx_0 = A^3x_0$
- ...
- until converges

For large powers of k, we will obrain a good approximation of the dominant eigenvector

Power Iterations (contd.)

We can just remove the dominant direction from the matrix and repeat

So:

- $A = Q\Lambda Q^T$, so $A = \sum_{i=1}^{n} \lambda_i \mathbf{q}_i \mathbf{q}_i^T$
- use power iteration to find $\mathbf{q}_1$ and λ_1
- then let $A_2 \leftarrow A - \lambda_1 \mathbf{q}_1 \mathbf{q}_1^T$
- repeat power iteration on A_2 to find $\mathbf{q}_2$ and λ_2
- continue like this for $\lambda_3, \ \ldots \ , \lambda_n$

QR Decomposition

$$\overset{A}{\begin{bmatrix} | & | & | \\ a_1 & a_2 & a_3 \\ | & | & | \end{bmatrix}} = \underbrace{\overset{Q}{\begin{bmatrix} | & | & | \\ e_1 & e_2 & e_3 \\ | & | & | \end{bmatrix}}}_{\substack{\text{orthogonal} \\ \text{unit vector}}} \underbrace{\overset{R}{\begin{bmatrix} e_1^T \cdot a_1 & e_1^T \cdot a_2 & e_1^T \cdot a_3 \\ 0 & e_2^T \cdot a_2 & e_2^T \cdot a_3 \\ 0 & 0 & e_3^T \cdot a_3 \end{bmatrix}}}_{\substack{\text{upper diagonal} \\ \text{matrix}}}$$

QR Decomposition

Algorithm:

- choose Q_0 such that $Q_0^T Q_0 = I$
- for $k = 1, 2, \ldots$:
  - $Z_k = AQ_{k-1}$
  - $Q_k R_k = Z_k$ (QR decomposition)

PowerSGD

function COMPRESS+AGGREGATE(update matrix $M \in \mathbb{R}^{n \times m}$, previous $Q \in \mathbb{R}^{m \times r}$)

$P \leftarrow MQ$

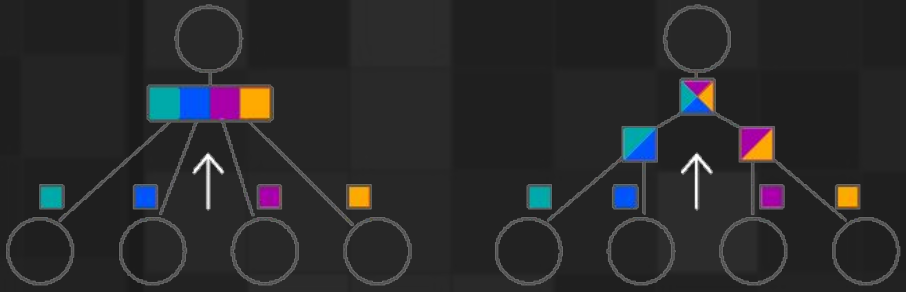
$P \leftarrow \text{ALL REDUCE MEAN}(P)$ ▷ Now, $P = \frac{1}{W}(M_1 + \ldots + M_W)Q$

$\hat{P} \leftarrow \text{ORTHOGONALIZE}(P)$ ▷ Orthonormal columns

$Q \leftarrow M^\top \hat{P}$

$Q \leftarrow \text{ALL REDUCE MEAN}(Q)$ ▷ Now, $Q = \frac{1}{W}(M_1 + \ldots + M_W)^\top \hat{P}$

return the compressed representation $(\hat{P}, Q)$.



(a) Gather (b) Reduce

PowerSGD + Error Feedback

Algorithm 2 Distributed Error-feedback SGD with Momentum

hyperparameters: learning rate γ, momentum parameter λ
initialize model parameters $\mathbf{x} \in \mathbb{R}^d$, momentum $\mathbf{m} \leftarrow \mathbf{0} \in \mathbb{R}^d$, replicated across workers
at each worker $w = 1, \ldots, W$ **do**
  initialize memory $\mathbf{e}_w \leftarrow \mathbf{0} \in \mathbb{R}^d$
  for each iterate $t = 0, \ldots$ **do**
    Compute a stochastic gradient $\mathbf{g}_w \in \mathbb{R}^d$.
    $\Delta_w \leftarrow \mathbf{g}_w + \mathbf{e}_w$ ▷ Incorporate error-feedback into update
    $\mathcal{C}(\Delta_w) \leftarrow \textsc{compress}(\Delta_w)$
    $\mathbf{e}_w \leftarrow \Delta_w - \textsc{decompress}(\mathcal{C}(\Delta_w))$ ▷ Memorize local errors
    $\mathcal{C}(\Delta) \leftarrow \textsc{aggregate}(\mathcal{C}(\Delta_1), \ldots, \mathcal{C}(\Delta_W))$ ▷ Exchange gradients
    $\Delta' \leftarrow \textsc{decompress}(\mathcal{C}(\Delta))$ ▷ Reconstruct an update $\in \mathbb{R}^d$
    $\mathbf{m} \leftarrow \lambda \mathbf{m} + \Delta'$
    $\mathbf{x} \leftarrow \mathbf{x} - \gamma \left(\Delta' + \mathbf{m}\right)$
  end for
end at

Results

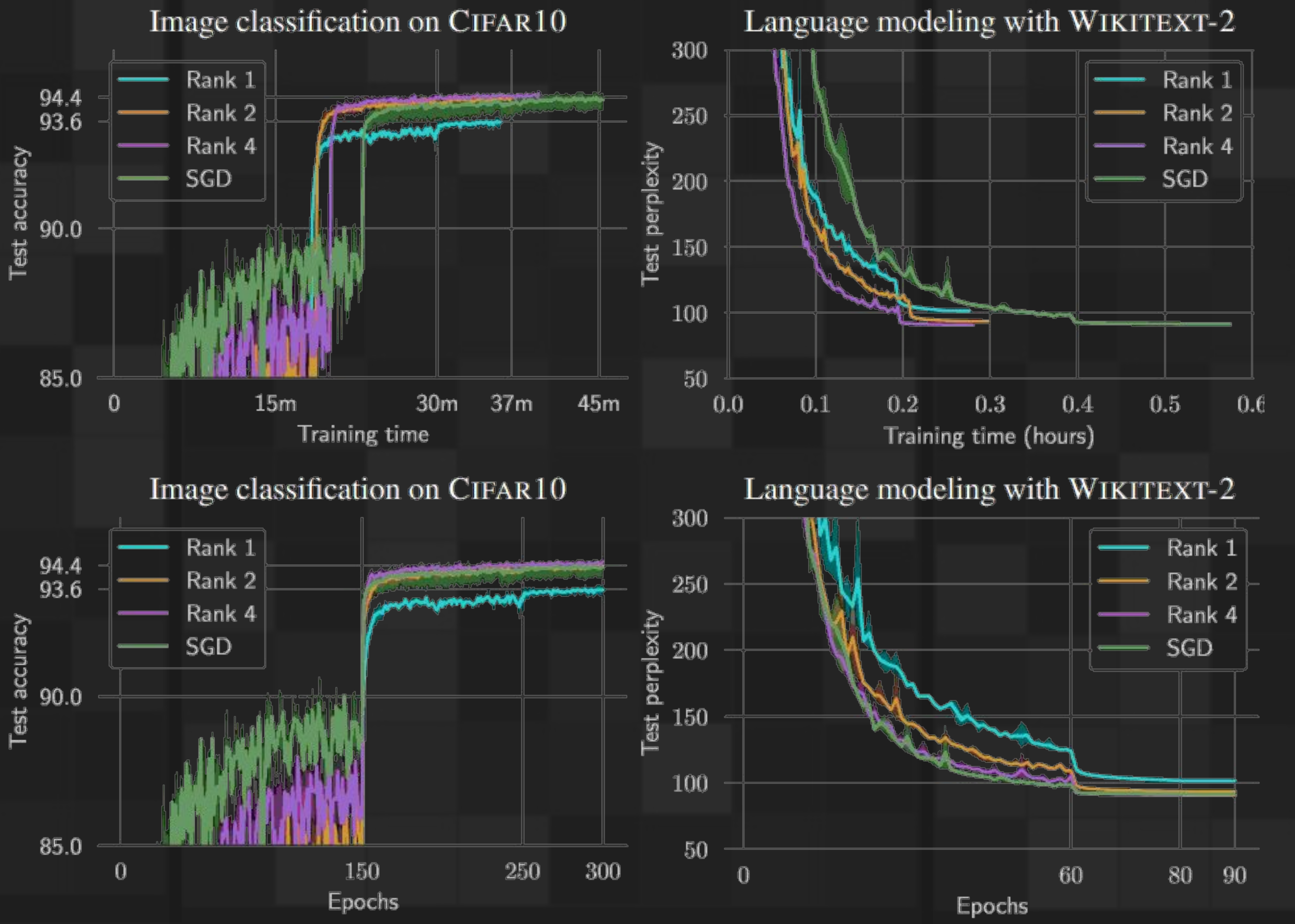


Image classification on CIFAR10
Rank 1
Rank 2
Rank 4
SGD
Test accuracy
94.4
93.6
90.0
85.0
0
15m
30m
37m
45m
Training time
Language modeling with WIKITEXT-2
300
250
200
150
100
50
Test perplexity
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Training time (hours)
Image classification on CIFAR10
Test accuracy
0
150
250
300
Epochs
Language modeling with WIKITEXT-2
Test perplexity
0
60
80
90
Epochs

Our Approach

- Use AD statistics instead of gradients for automatic bandwidth reduction
- Further use structured power iterations