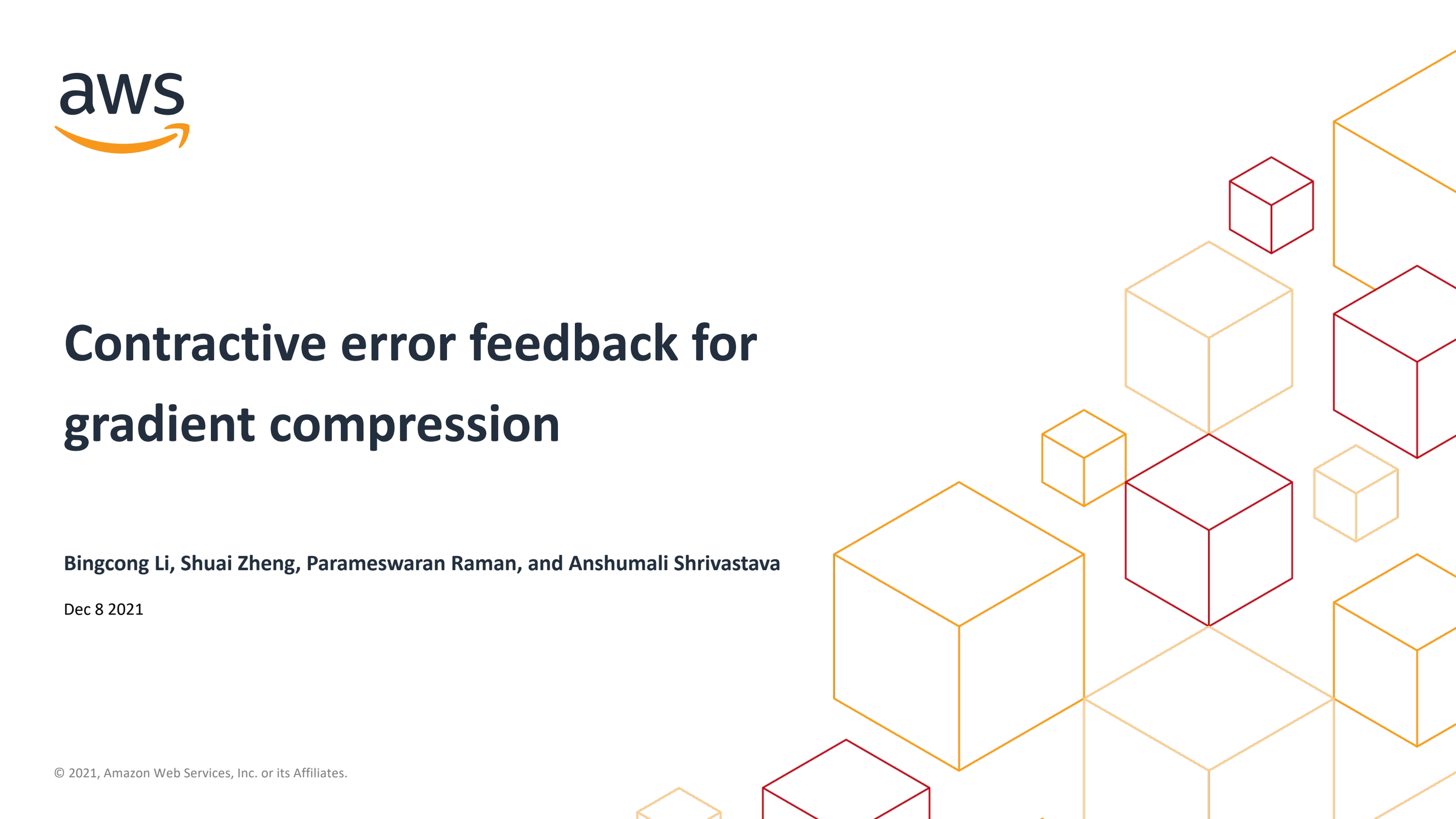


Contractive error feedback for gradient compression

Bingcong Li, Shuai Zheng, Parameswaran Raman, and Anshumali Shrivastava

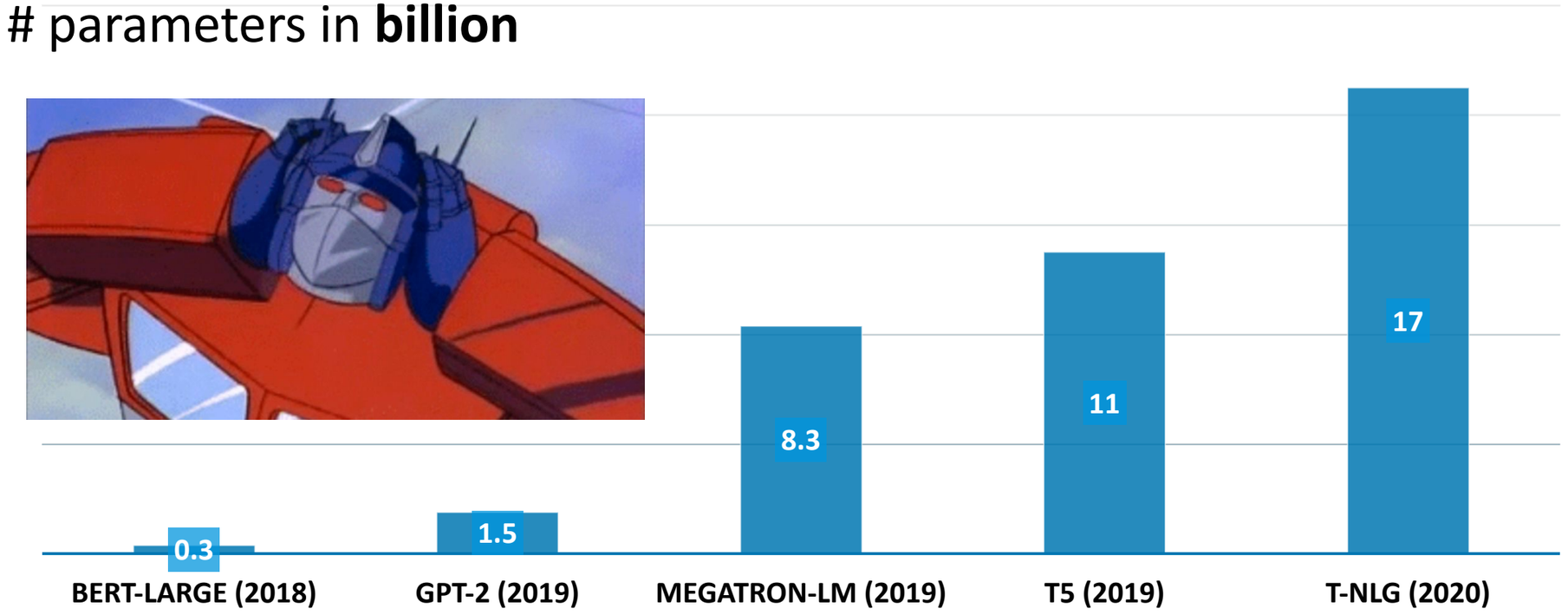
Dec 8 2021

Roadmap

- ❑ **Motivation and project overview**
- ❑ Algorithm design
- ❑ Numerical experiments
- ❑ Distributed training for large models redux

aws

After transformers, neural networks are getting out of control



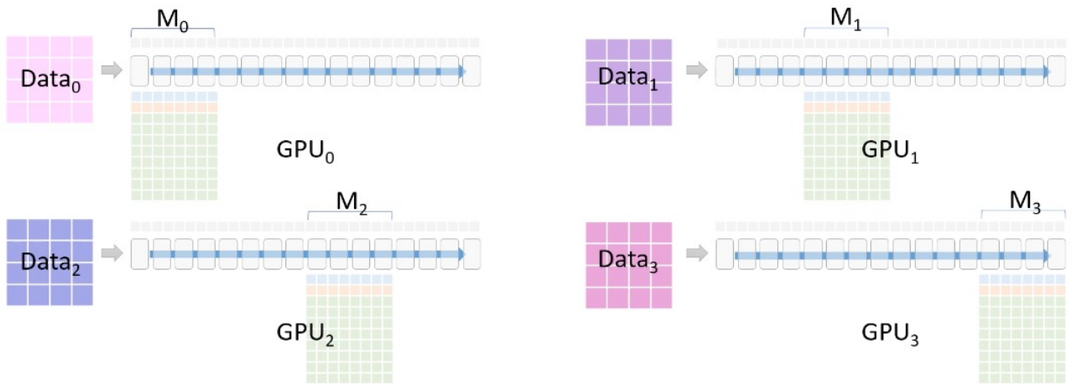
- Memory issues: model does not fit into a GPU such as V100 (16 or 32 GB memory)
- Runtime issues: training requires a few days if not weeks
- Memory is a dimension that is **largely ignored** in communication efficient methods

aws

ZeRO3 increases communication overhead

- Memory: ZeRO3 partitions parameters / gradients / optimizer states

Each cell represents GPU memory used by its corresponding transformer layer



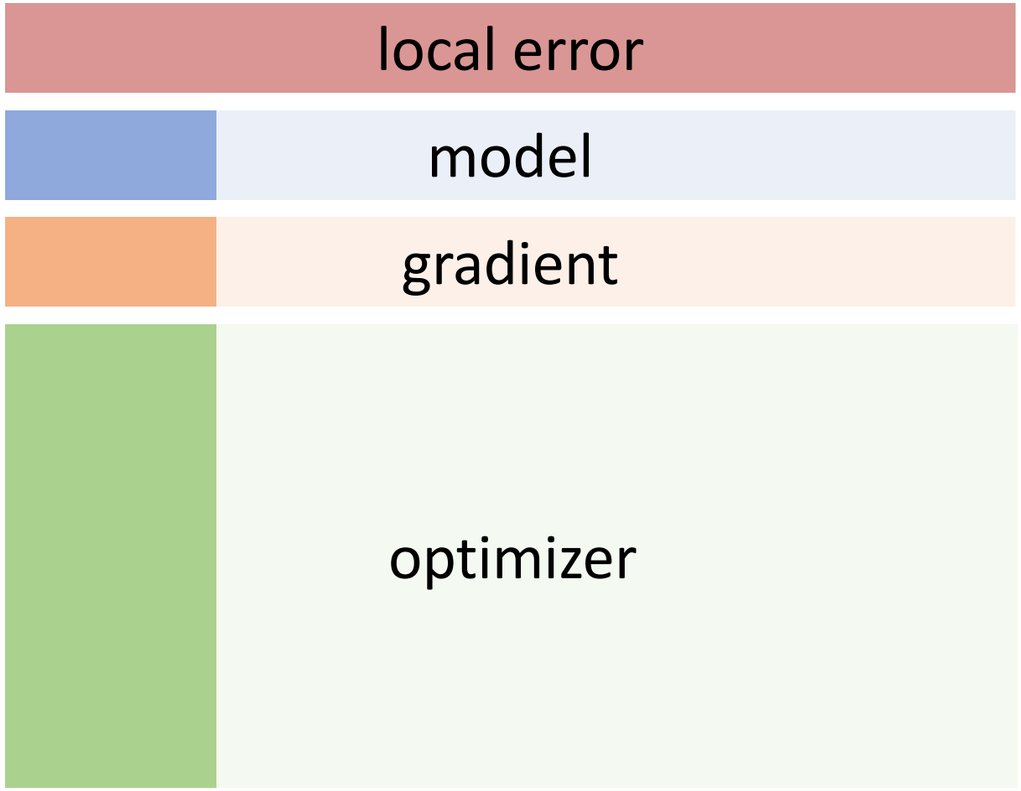
Each GPU is responsible for 1 piece of the end model
ZeRO P_{os+g+p} and Gradient accumulation are used with the 4-way data parallelism

- Communication: ZeRO3 increases communication overhead by 50% to save memory

- **Goal:** understanding the memory and communication tradeoff

aws

How to reduce the communication overhead for ZeRO3

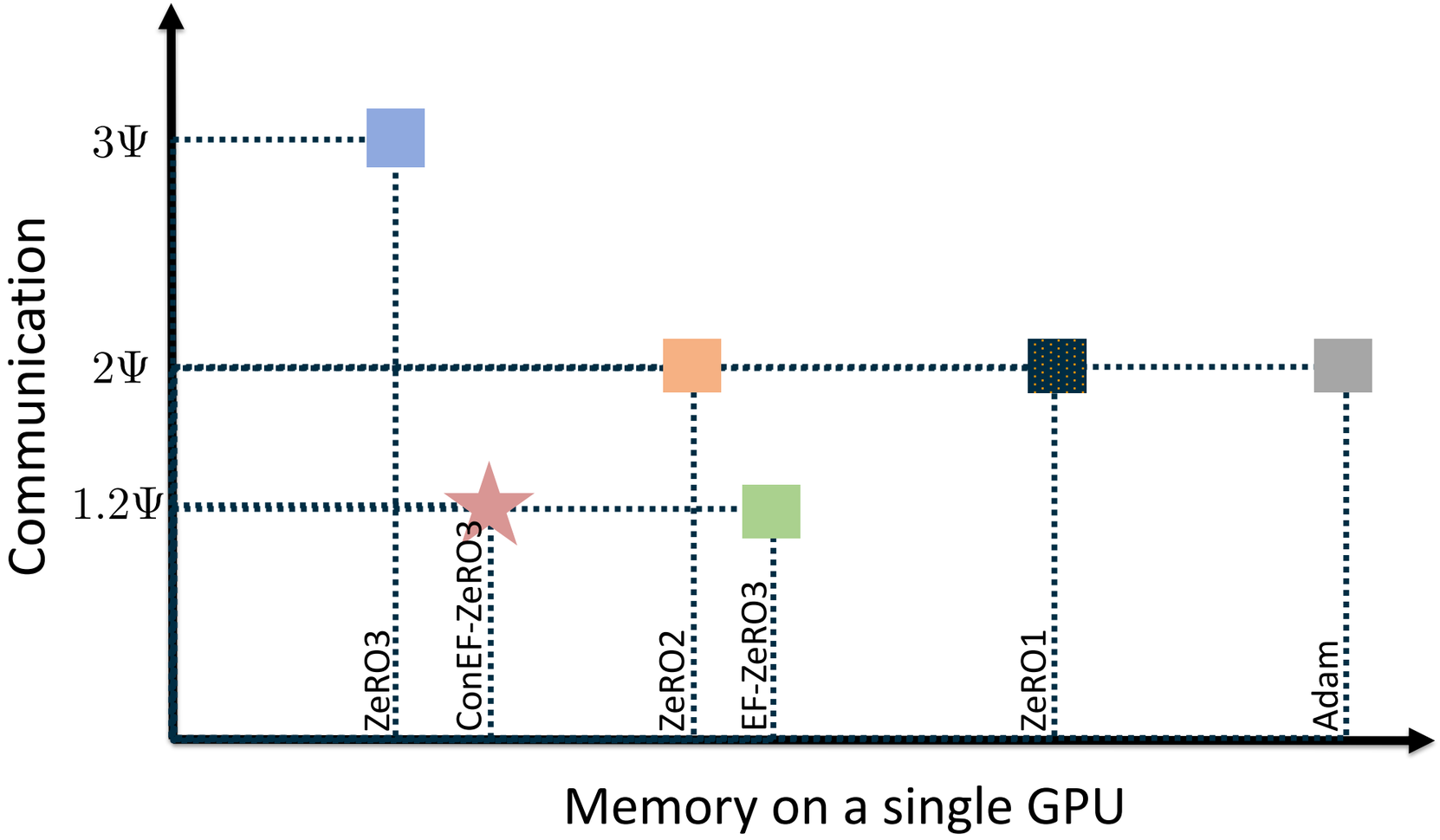


EFSGD

Initialize: $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{e}_0 = \mathbf{0} \in \mathbb{R}^d, \eta$
for $t = 0, 1, \ldots, T-1$ **do**
  assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
  for worker i in parallel **do**
    $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + \mathbf{e}_t^i$
    $\mathbf{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
    $\mathbf{\Delta}_t = \text{Aggregate}(\mathbf{\Delta}_t^i, \forall i)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{\Delta}_t$
    $\mathbf{e}_{t+1}^i = \mathbf{p}_t^i - \mathbf{\Delta}_t^i$
  end for
end for

aws

Memory - communication tradeoff in ZeRO



How to get to ★

- Step 1: algorithm design
- Step 2: test with adam
- Step 3: integrate to ZeRO3

aws

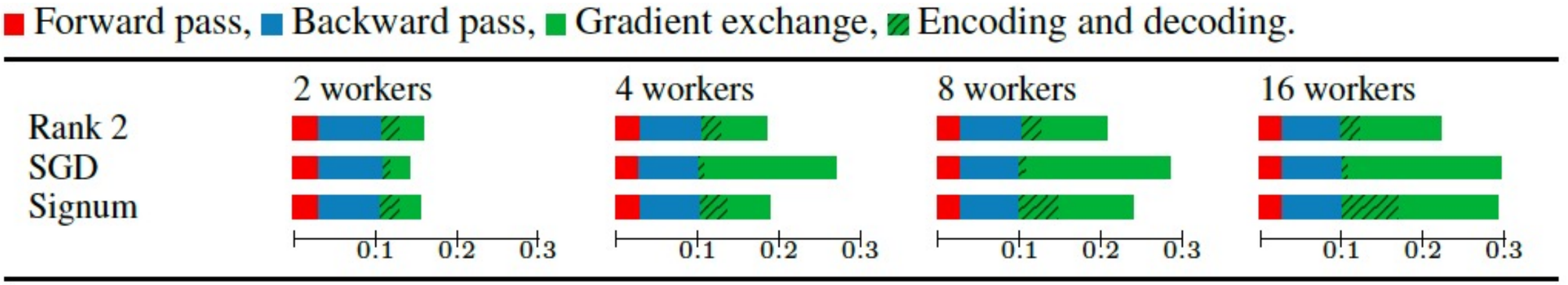
Roadmap

- ❑ Motivation and project overview
- ❑ **Algorithm design**
- ❑ Numerical experiments
- ❑ Distributed training for large models redux

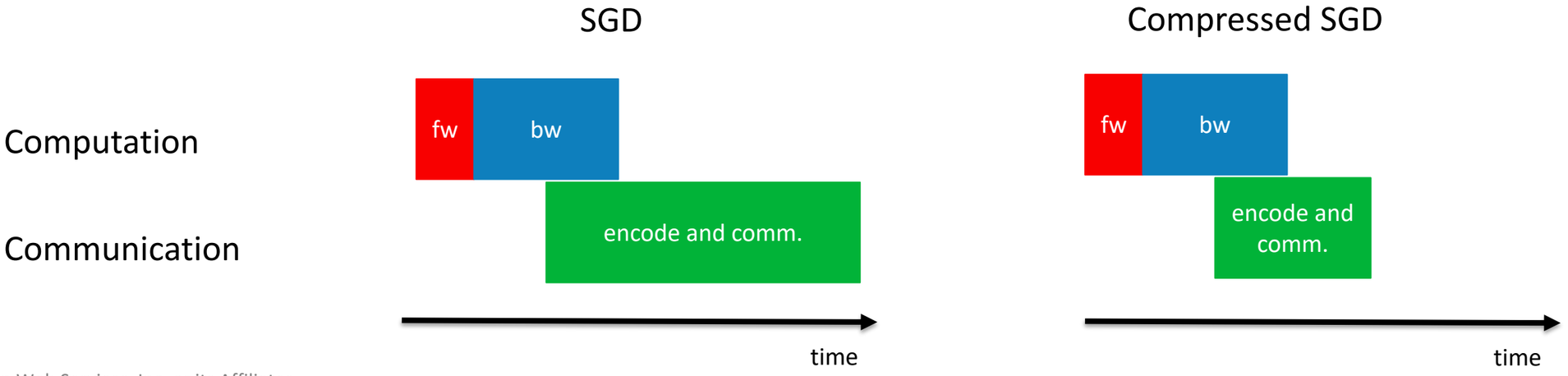
aws

Communication bottleneck

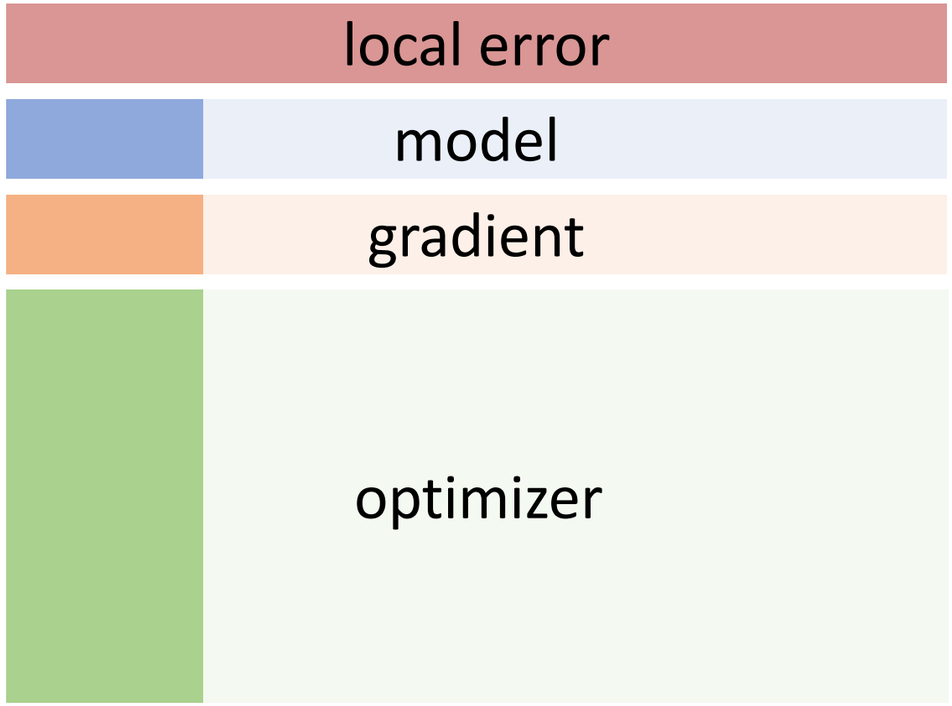
- Communication time increases with workers [VKM' 19]



- Efficient communication reduces idle time



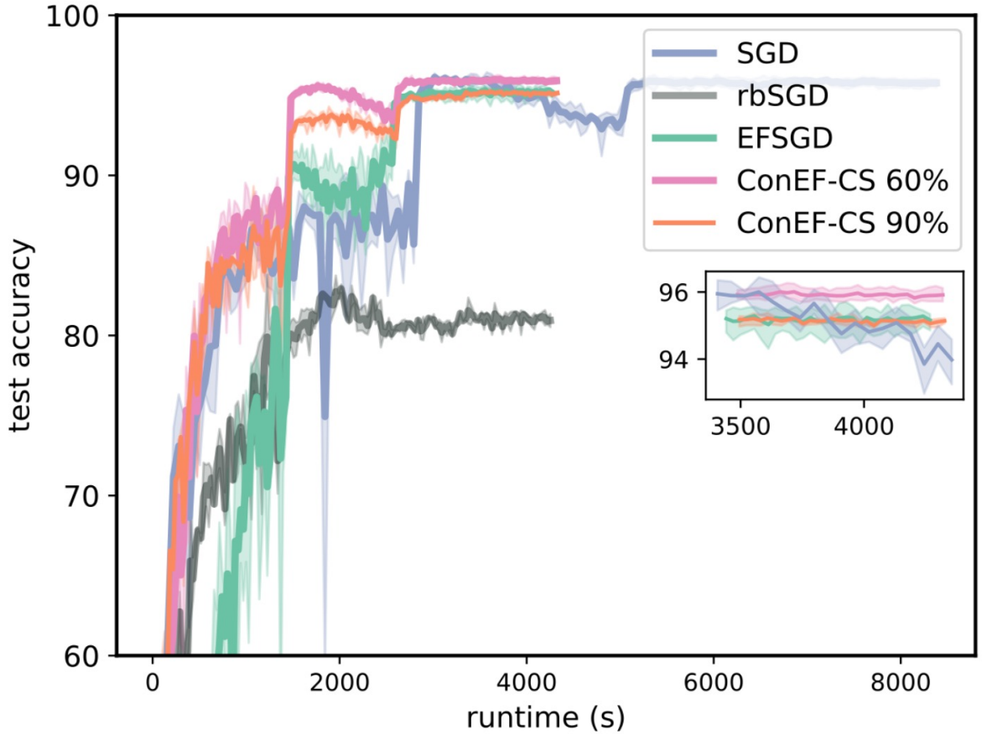
Why EF is necessary



- Local SGD [Stich' 19]
  - Communicate every a few iterations
  - Infeasible for ZeRO3 since optimizer states are partitioned

aws

Why EF is necessary



- Unbiased gradient compressors [ADGLTV' 17]
  - Enlarged gradient variance hurts performance (e.g., accuracy)

aws

Contractive error feedback (ConEF)

ConEF

Initialize: $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{e}_0^i = \mathbf{0} \in \mathbb{R}^d, \forall i, \eta$
for $t = 0, 1, \ldots, T-1$ **do**
  assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
  for worker $i = 1, \ldots, N$ in parallel **do**
    $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + \mathbf{e}_t^i$
    $\boldsymbol{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
    $\boldsymbol{\Delta}_t = \text{Aggregate}(\boldsymbol{\Delta}_t^i, \forall i)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \boldsymbol{\Delta}_t$
    $\mathbf{e}_{t+1}^i = \mathcal{C}(\mathbf{p}_t^i - \boldsymbol{\Delta}_t^i)$
  end for
end for

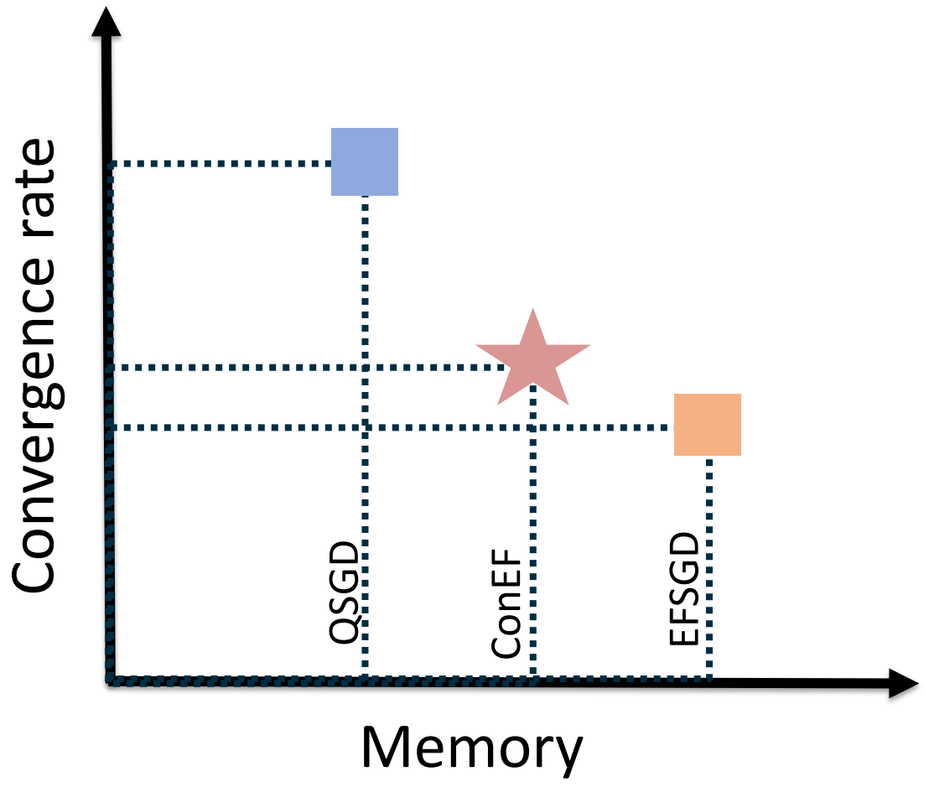
EFSGD

Initialize: $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{e}_0^i = \mathbf{0} \in \mathbb{R}^d, \forall i, \eta$
for $t = 0, 1, \ldots, T-1$ **do**
  assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
  for worker $i = 1, \ldots, N$ in parallel **do**
    $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + \mathbf{e}_t^i$
    $\boldsymbol{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
    $\boldsymbol{\Delta}_t = \text{Aggregate}(\boldsymbol{\Delta}_t^i, \forall i)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \boldsymbol{\Delta}_t$
    $\mathbf{e}_{t+1}^i = \mathbf{p}_t^i - \boldsymbol{\Delta}_t^i$
  end for
end for

- **Key idea:** compressors can be used on local error as well
- Simple operation saves more runtime
- **Convergence:** ConEF finds the sweet spot of QSGD and EFSGD
- **Practicality:** depending on which error compressor to use

aws

Convergence of ConEF



ConEF

Initialize: $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{e}_0^i = \mathbf{0} \in \mathbb{R}^d, \forall i, \eta$
for $t = 0, 1, \ldots, T-1$ **do**
  assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
  for worker $i = 1, \ldots, N$ in parallel **do**
    $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + \mathbf{e}_t^i$
    $\boldsymbol{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
    $\boldsymbol{\Delta}_t = \text{Aggregate}(\boldsymbol{\Delta}_t^i, \forall i)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \boldsymbol{\Delta}_t$
    $\mathbf{e}_{t+1}^i = \mathcal{C}(\mathbf{p}_t^i - \boldsymbol{\Delta}_t^i)$
  end for
end for

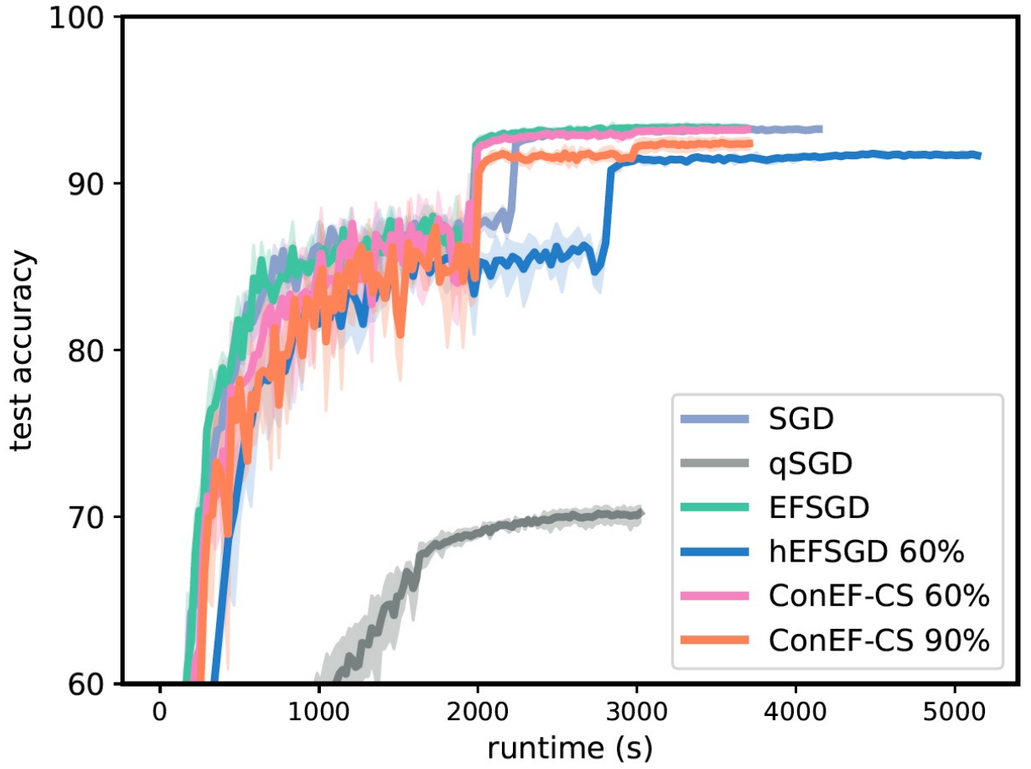
- Assumption on error compressor $\mathbb{E}[\mathcal{C}(\mathbf{x})] = \mathbf{x}$ and $\mathbb{E}[\|\mathcal{C}(\mathbf{x}) - \mathbf{x}\|^2] \leq \theta \mathbb{E}[\|\mathbf{x}\|^2]$
- Assumption on gradient compressor $\exists \delta \in (0, 1),$ s.t. $\mathbb{E}[\|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\|^2] \leq \delta \mathbb{E}[\|\mathbf{x}\|^2]$

aws

Deeper dive into error compressors *C*

- Assumption on error compressor $\mathbb{E}[\mathcal{C}(\mathbf{x})] = \mathbf{x}$ and $\mathbb{E}[\|\mathcal{C}(\mathbf{x}) - \mathbf{x}\|^2] \leq \theta \mathbb{E}[\|\mathbf{x}\|^2]$

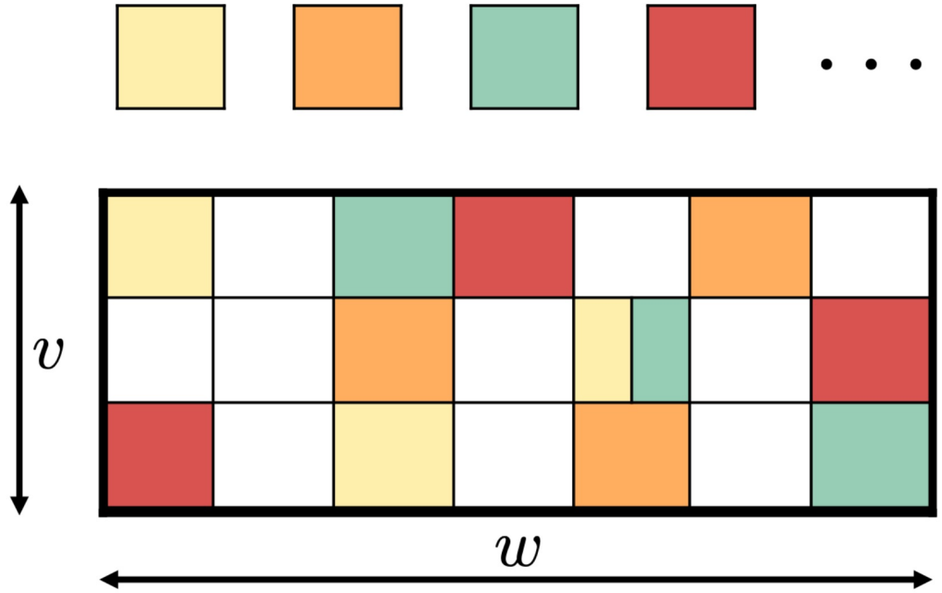
- Unbiased *C* has some generalization merits

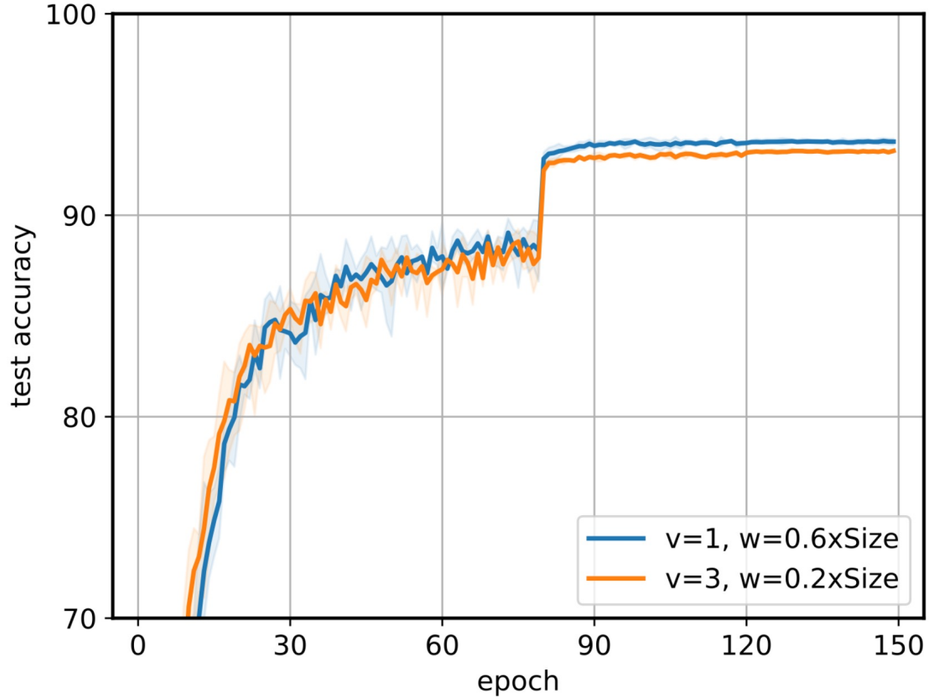


aws

Deeper dive into error compressors *C*

- Rule of thumb: **count sketch** [CCF ‘02]

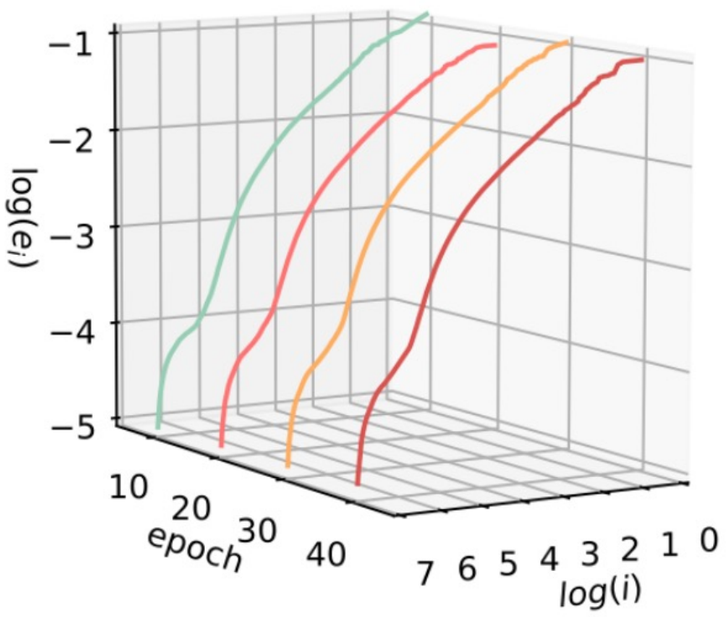




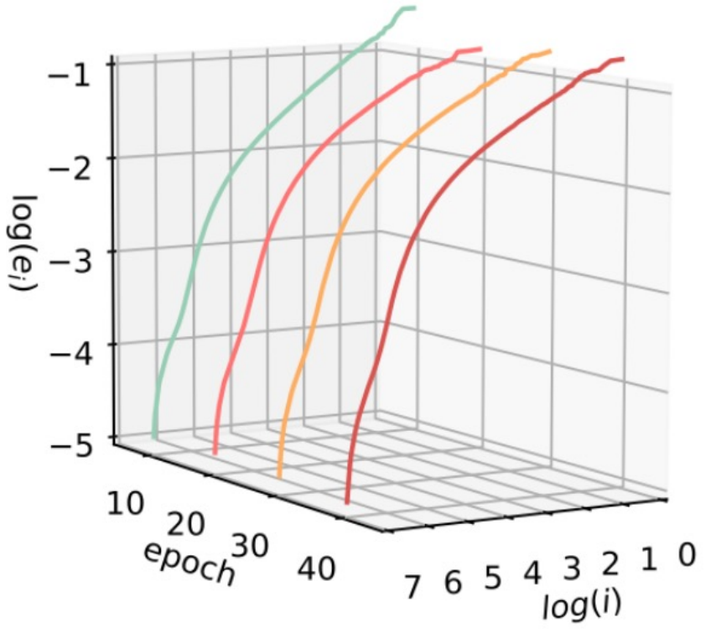
- One can also use e.g., stochastic quantization [ADGLTM’ 17] and its variants

aws

Deeper dive into error compressors *C*



EFSGD



(partial)-EFSGD [AF' 20]

Initialize: $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{e}_0 = \mathbf{0} \in \mathbb{R}^d, \eta$
for $t = 0, 1, \ldots, T-1$ **do**
  assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
  for worker i in parallel **do**
    $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + (1-\beta)\mathbf{e}_t^i$
    $\mathbf{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
    $\mathbf{\Delta}_t = \text{Aggregate}(\mathbf{\Delta}_t^i, \forall i)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{\Delta}_t$
    $\mathbf{e}_{t+1}^i = \mathcal{C}(\beta \mathbf{e}_t + \mathbf{p}_t^i - \mathbf{\Delta}_t^i)$
  end for
end for

Table 1: β vs. sketch size for convergence

β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
smallest sketch size	0.9	0.8	0.7	0.6	0.4	0.4	0.3	0.2	0.1

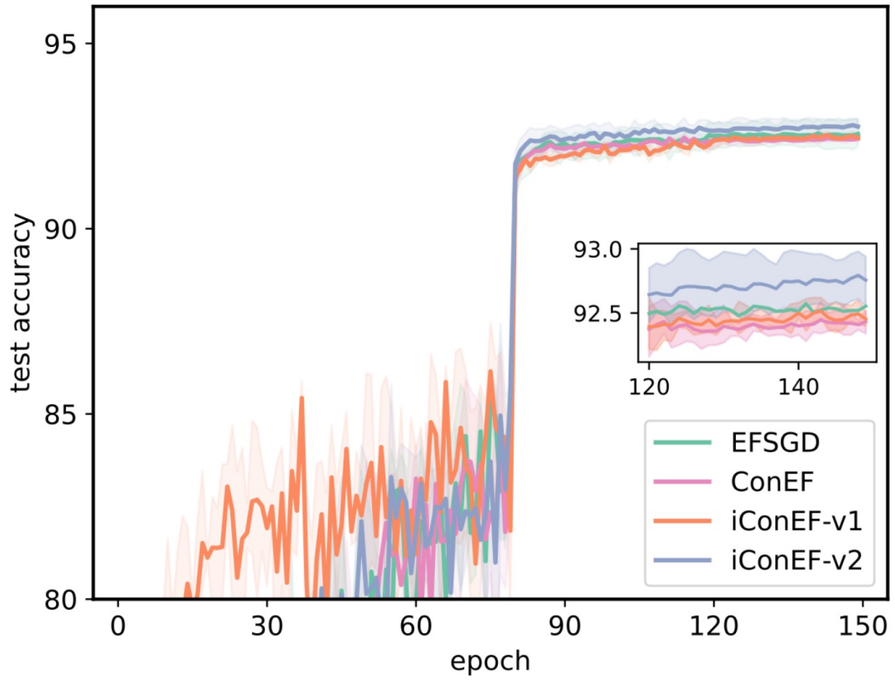
aws

Side results on tighter bounds

- Using ConEF on top of ConEF improves the convergence rate theoretically

improved ConEF (iConEF)

Initialize: $\mathbf{x}_0, \tilde{\mathbf{e}}_0^i = \mathbf{0}, \mathbf{q}_0^i = \mathbf{0}, \eta$
for $t = 0, 1, \ldots, T-1$ **do**
  assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
  for worker $i = 1, \ldots, N$ in parallel **do**
    $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + \tilde{\mathbf{e}}_t^i + \mathbf{q}_t^i$
    $\boldsymbol{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
    $\boldsymbol{\Delta}_t = \text{Aggregate}(\boldsymbol{\Delta}_t^i, \forall i)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \boldsymbol{\Delta}_t$
    if version 1 **then**:
      $\tilde{\mathbf{e}}_{t+1}^i = \mathcal{C}(\mathbf{p}_t^i - \boldsymbol{\Delta}_t^i)$ ▷ iConEF-v1
    else:
      $\tilde{\mathbf{e}}_{t+1}^i = \mathcal{Q}(\mathbf{p}_t^i - \boldsymbol{\Delta}_t^i)$ ▷ iConEF-v2
    end if
    $\mathbf{q}_{t+1}^i = \mathcal{C}(\mathbf{p}_t^i - \boldsymbol{\Delta}_t^i - \tilde{\mathbf{e}}_{t+1}^i)$
  end for
end for



Roadmap

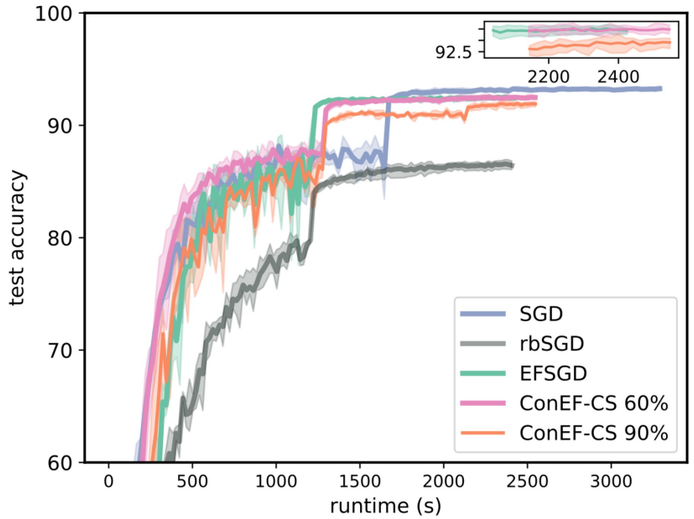
- ❑ Motivation and goal
- ❑ Algorithm design
- ❑ **Numerical experiments**
- ❑ Distributed training for large models redux

aws

Step 1: ConEF for SGD based optimizers

- 4 Amazon p3.8xlarge instances (16 GPUs in total)
- Gradient compressor: random block k (90% reduced comm.)
- Error compressor: count sketch
- **ResNet18 on CIFAR-10**

Algorithm	test accuracy	runtime (min)	memory saving (MB)
SGD	93.20	54.83	-
rbSGD	86.58	40.12	48.2
EFSGD	92.52	40.41	0
ConEF 60%	92.53	42.45	31.5
ConEF 90%	92.02	42.45	44.1

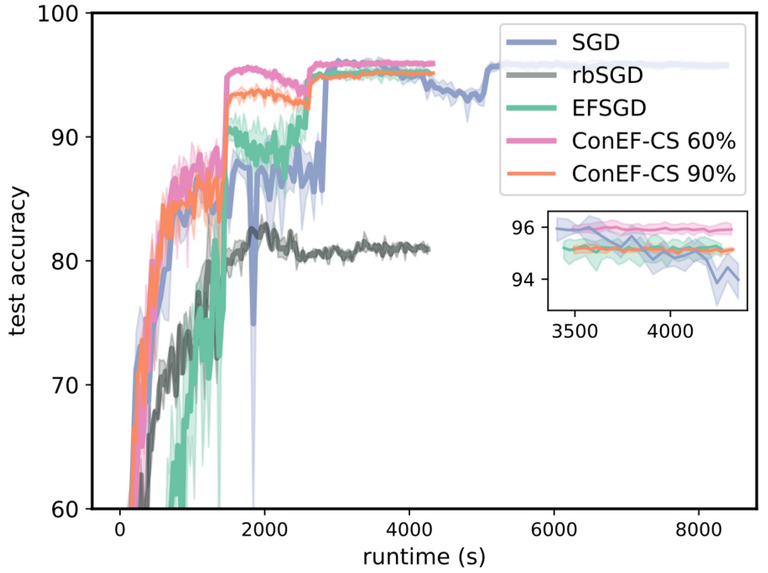


aws

Step 1: ConEF for SGD based optimizers

- 4 Amazon p3.8xlarge instances (16 GPUs in total)
- Gradient compressor: random block k (90% reduced comm.)
- Error compressor: count sketch
- **WideResNet-28-10 on CIFAR-10**

Algorithm	test accuracy	runtime (min)	memory saving (MB)
SGD	96.12	139.70	-
rbSGD	82.92	70.88	122.1
EFSGD	95.28	71.13	0
ConEF 60%	96.00	72.11	76.2
ConEF 90%	95.23	71.98	101.5



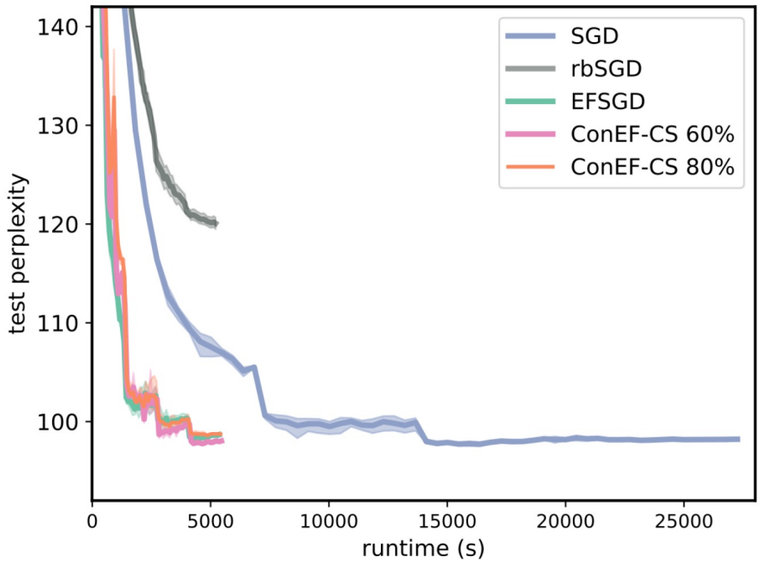
- ConEF is more suitable for larger models

aws

Step 1: ConEF for SGD based optimizers

- 4 Amazon p3.8xlarge instances (16 GPUs in total)
- Gradient compressor: random block k (90% reduced comm.)
- Error compressor: count sketch
- **2-layer LSTM on WikiText-2**

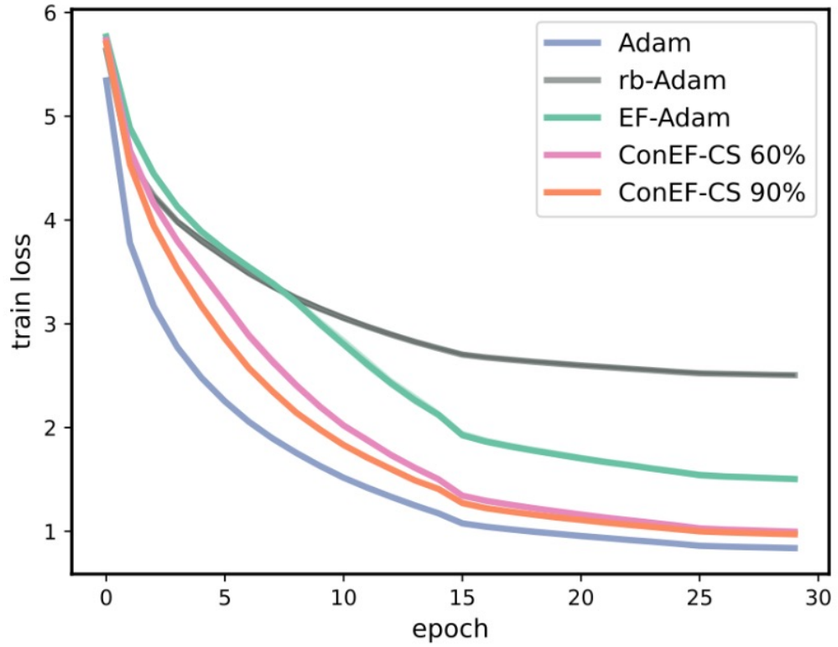
Algorithm	perplexity	runtime (hour)	memory saving (MB)
SGD	97.72	7.57	-
rbSGD	120.07	1.45	262.5
EFSGD	98.29	1.49	0
ConEF 60%	97.76	1.51	180.3
ConEF 80%	98.59	1.50	220.2



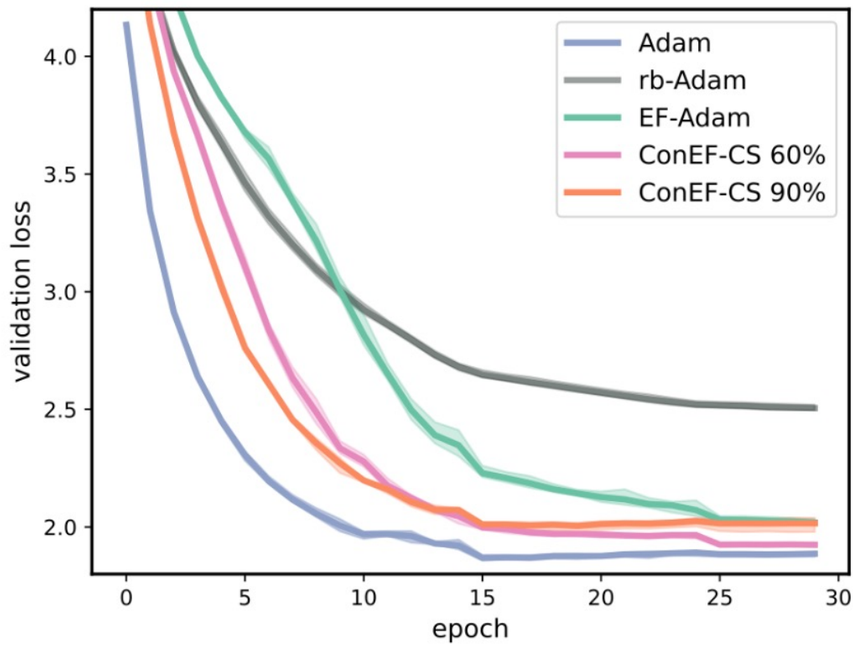
aws

Step 2: ConEF for Adam type optimizers

- A **small transformer** for machine translation on **Multi30K**



(a) train



(b) validation

aws

Step 2: ConEF with Adam type optimizers

- 4 Amazon p3.16xlarge instances (32 GPUs in total)
- Gradient compressor: random block k (95% reduced comm.)
- Error compressor: count sketch

- BERT-BASE pretraining (not completed yet)

Algorithm	out of memory	Phase I loss	Phase II loss	runtime (hour)
LAMB	No	≈1.62		40.2 +
EF-LAMB	Yes	-	-	-
EF-LAMB (mix precision)	No	≈1.76		37.6 +
ConEF 80%	No	≈1.79		38.1 +

- Possible reasons for performance gap:

Step 2: ConEF with Adam type optimizers

- BERT-BASE downstream tasks (not completed yet)

aws

Roadmap

- ❑ Motivation and goal
- ❑ Algorithm design
- ❑ Numerical experiments
- ❑ **Distributed training for large models redux**

aws

Rethinking distributed training for NLP

- Current distributed training paradigm targets **consensus**
  - Communication of stochastic gradients is essential for reaching consensus
  - **Q: is consensus necessary**
  - Amazon, Google, and NVIDIA have their own pretrained BERT
  - **Q: What and how to communicate if consensus in not the ultimate goal**

Figure from https://towardsdatascience.com/how-to-train-bert-aaad00533168

aws

Rethinking distributed training for NLP

- In the old days (ADMM, gossip)
  - Not enough data, strongly convex problem (hence unique solution)
- In the present ... nonuniqueness seems to be necessary
  - Data explosion, highly nonconvex losses with various local minima
  - Multiple manners to translate a specific sentence
  - **Q: is a single function enough**
  - **Q: How can we take advantage of models converging to different local minima**

References

[KRSJ’ 19] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In Proc. of Intl. Conf. on Machine Learning, 2019

[ADGLTM’ 17] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, andMilan Vojnovic. QSGD: Communication-efcient sgd via gradient quantization and encoding. Proc. of Neural Information Processing Systems, 2017

[BWAA’ 18] Jeremy Bernstein, Yu-XiangWang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimization for non-convex problems. In Proc. of Intl. Conf. on Machine Learning, 2018

[Stich’ 19] Sebastian Urban Stich. Local SGD converges fast and communicates little. In Proc. of Intl. Conf. on Learning Representations, 2019.

[VKM’ 19] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. Proc. of Neural Information Processing Systems, 2019

[HHHSCR’ 19] Samuel Horvath, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtarik. Natural compression for distributed deep learning. arXiv preprint arXiv:1905.10988, 2019.

[WRSSR’ 17] Ashia CWilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In Proc. of Neural Information Processing Systems, 2017

[Valiant, ‘84] Leslie GValiant. A theory of the learnable. Communications of the ACM, 27(11):1134-1142, 1984

[RFMAAR’ 21] Ali Ramezani-Kebrya, Fartash Faghri, IlyaMarkov, Vitalii Aksenov, Dan Alistarh, and Daniel MRoy. NUQSGD: Provably communication-efcient data-parallel SGD via nonuniformquantization. Journal of Machine Learning Research, 22(114):1-43, 2021

[nvcomp] https://github.com/NVIDIA/nvcomp

[CCF’ 02] Moses Charikar, Kevin Chen, andMartin Farach-Colton. Finding frequent items in data streams. In International Colloquium on Automata, Languages, and Programming, Springer, 2002.

[AF ‘20] Afshin Abdi and Faramarz Fekri. Quantized compressive samplingof stochastic gradients for effcient communication in distributed deep learning. In Proceedings of the AAAI Conference on Artifcial Intelligence, 2020

aws

Acknowledgement (alphabetical order)

Vasileios Ioannidis

George Karypis

Parameswaran Raman

Xingjian Shi

Anshumali Shrivastava

Bo Yang

Sheng Zha

Shuai Zheng

Thank you!

Q&A

Bingcong Li