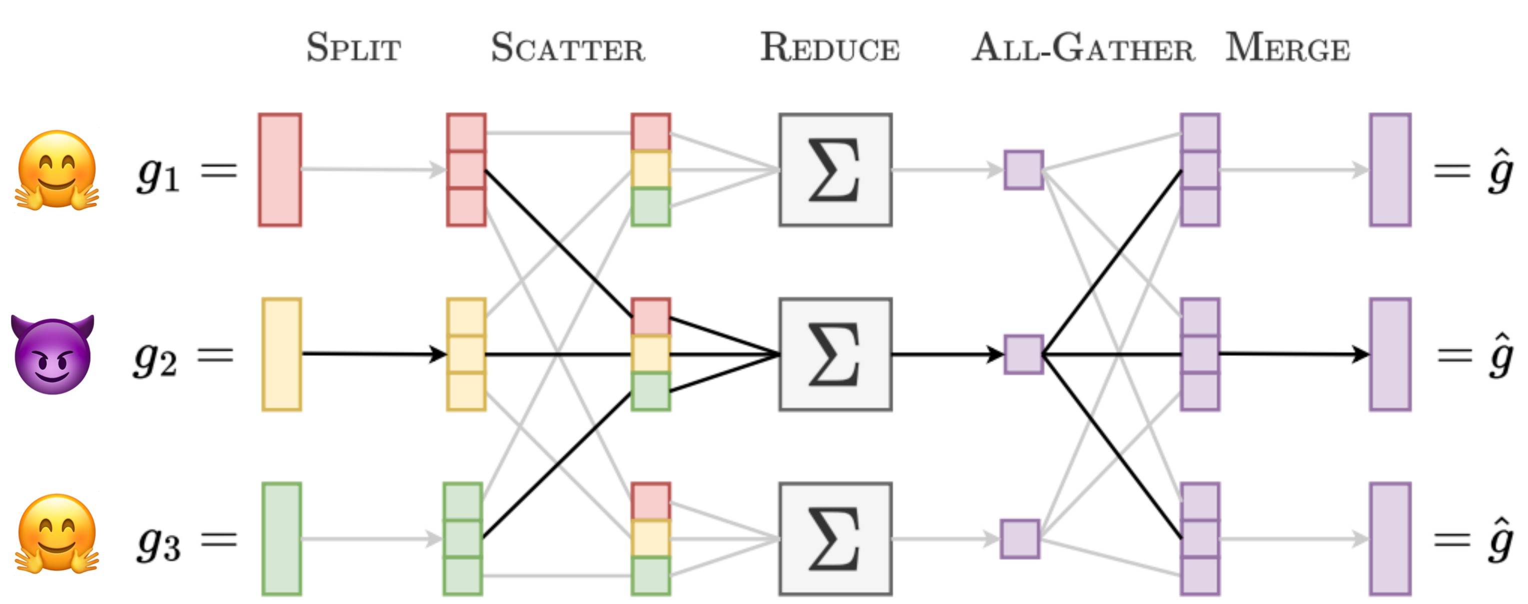
Secure Distributed Training at Scale

Eduard Gorbunov*, Alexander Borzunov*,
Michael Diskin, Max Ryabinin

Motivation

- Many areas of deep learning benefit from large **foundation models** trained on public data.
- These models are usually trained on HPC clusters not available to small labs and independent researchers.
- Instead, several smaller groups can **pool their compute resources together** and train a model that benefits all participants.
- However, any participant can jeopardize such a training run by sending incorrect updates (see the scheme below), unless we use special distributed training algorithms with **Byzantine tolerance**.
- Prior work on Byzantine tolerance involves redundant communication or trusted parameter servers, both infeasible in large-scale deep learning.



Contribution

- We propose a **novel protocol for decentralized Byzantine-tolerant training** suitable for large-scale deep learning, where the extra communication cost does not depend on the number of parameters.
- To achieve that, we modify Butterfly All-Reduce (see the scheme above) with a robust aggregation technique known as CenteredClip (Karimireddy et al., 2020) and several cryptography-based verifications.
- We also propose a heuristic for resisting Sybil attacks from computationally constrained attackers, allowing to accept any number of untrusted peers joining midway through training.

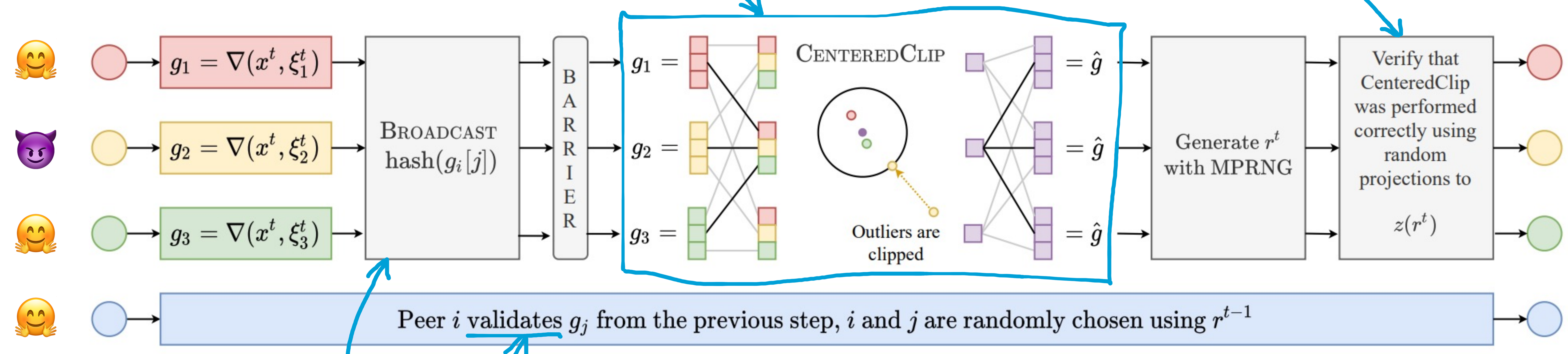
References

- Karimireddy, Sai Praneeth, Lie He, and Martin Jaggi. "Learning from history for byzantine robust optimization." *International Conference on Machine Learning*. PMLR, 2021.
- Allen-Zhu, Zeyuan, et al. "Byzantine-Resilient Non-Convex Stochastic Gradient Descent." *International Conference on Learning Representations.* 2020.

Protocol

Step 1. Use Butterfly All-Reduce, but clip gradient outliers (protects from large perturbations)

Step 2. Validate that peers are not cheating during the clipping procedure



Step 3. Periodically validate gradients of random peers by recalculation (protects from a series of small perturbations)

Convergence Bounds

- We prove that our method converges to any predefined accuracy under realistic assumptions.
- If the required accuracy is high or the number of attackers is low, it converges with the same speed as the usual Parallel SGD without malicious workers.
- Our convergence rates are **state-of-the-art** in the decentralized Byzantine-tolerant setting (and better than SOTA for the centralized Byzantine-tolerant setting if the required accuracy is high).
- We prove strong results for non-convex problems (see below), as well as for convex and strongly convex problems (see in the paper).

Decentralized?	Work	Non-convex
✗	Allen-Zhu et al. (2021)	$\frac{1}{n\varepsilon^4} + \frac{\delta^2}{\varepsilon^4}$
	Karimireddy et al. (2020)	$\frac{1}{\varepsilon^2} + \frac{\sigma^2}{n\varepsilon^4} + \frac{\delta\sigma^2}{\varepsilon^4}$
✓	**This work**	$\frac{1}{\varepsilon^2} + \frac{\sigma^2}{n\varepsilon^4} + \frac{n\delta\sigma^2}{m\varepsilon^2}$

Here, σ^2 is the upper bound on the gradient variance, ε is the target accuracy, n is the total number of peers, δ is the maximal share of malicious peers, m is the number of peers serving as validators on each step.

Experiments

- We ensure that our method **does not harm convergence.**
- We experiment with 7 kinds of attacks while training ResNet-18 and 4 kinds of attacks while training ALBERT-large.
- We test attacks at various stages of training, with various periodicity and number of attackers.
- We show that our method **succeeds to protect** the training run unlike other methods from prior work.