Secure Distributed Training at Scale

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

Speaker: Alexander Borzunov

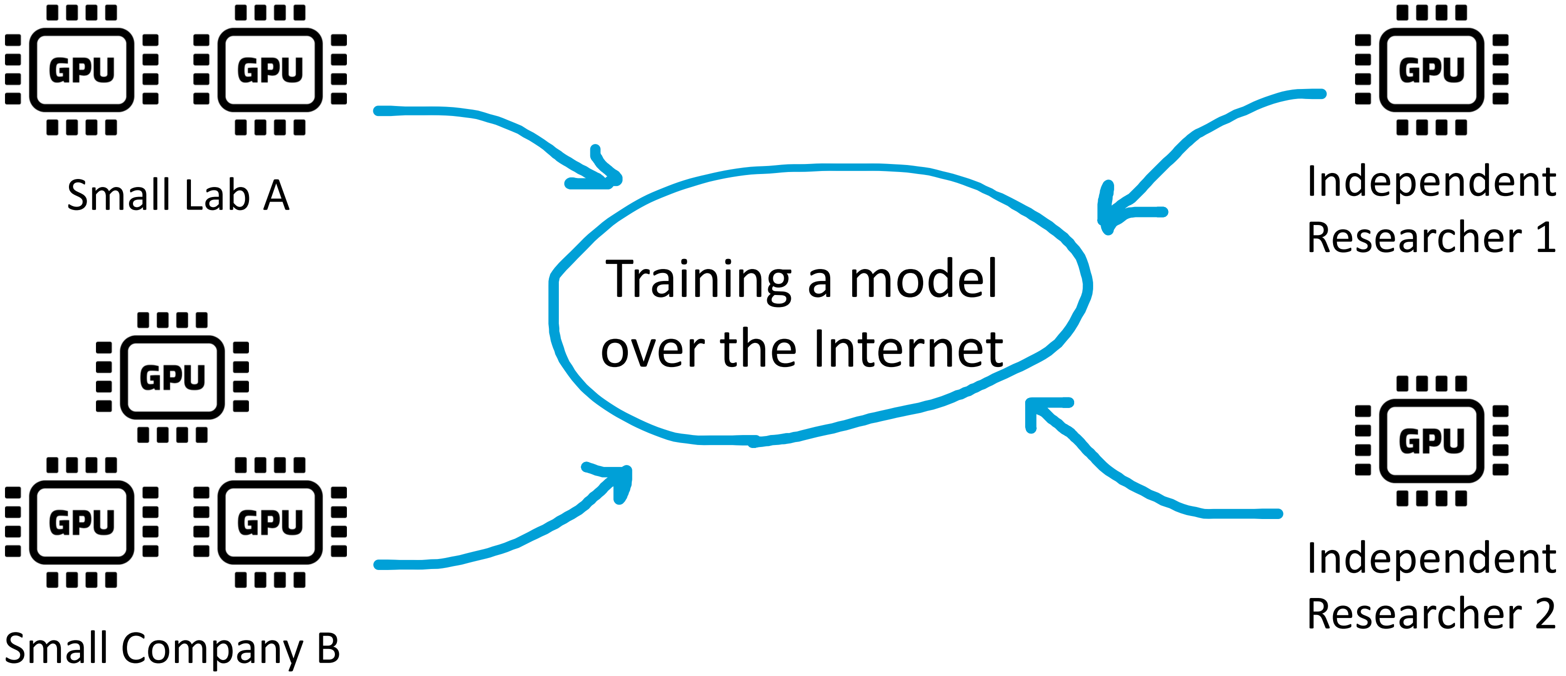
MIPT Mila Yandex HSE University

Motivation

- Many areas of deep learning benefit from using increasingly larger neural networks trained on public data
  - Example: Pre-trained models for NLP and computer vision
- These models are usually trained on expensive HPC clusters not available to small labs and independent researchers

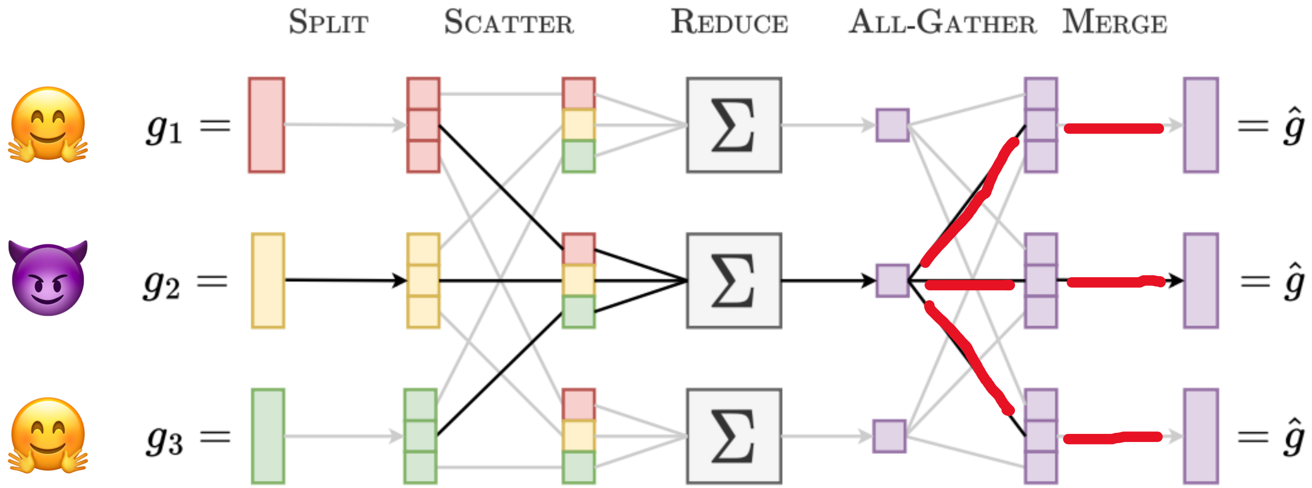
Motivation

- Instead, several smaller groups could **pool their existing compute resources together** and train a model that benefits all participants



Motivation

- However, any participant can jeopardize training by sending incorrect updates, unless we use algorithms with **Byzantine tolerance**



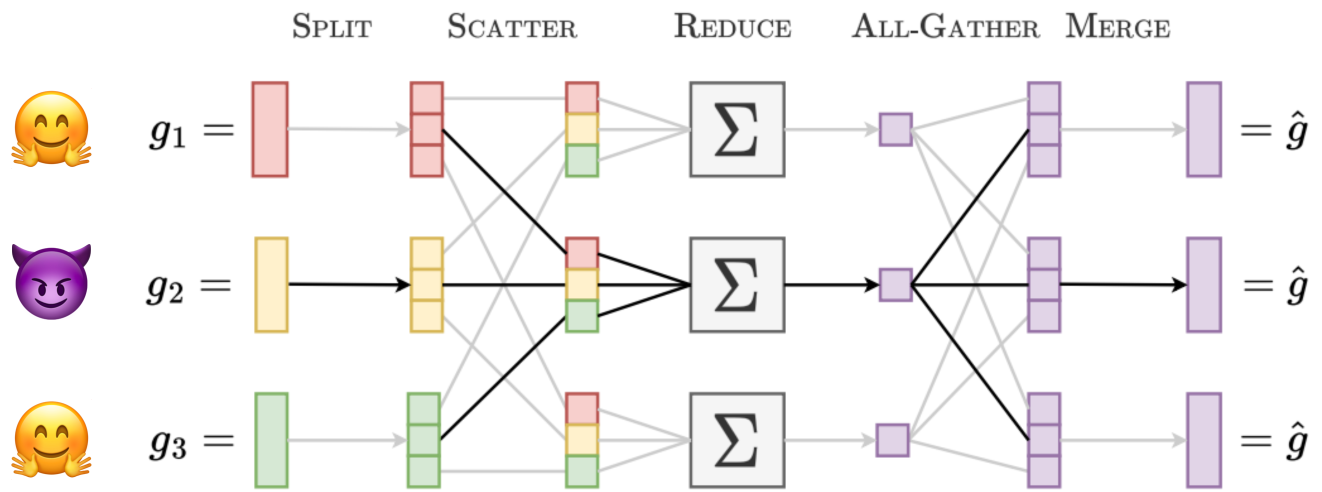
- Prior work on Byzantine tolerance involves redundant communication or trusted parameter servers, infeasible in large-scale deep learning

Contribution

- In this work, we introduce a novel protocol for decentralized Byzantine-tolerant training **suitable for large-scale deep learning**
  - Its extra communication cost does not depend on the number of parameters
- We also propose a heuristic for resisting Sybil attacks, so untrusted peers can join midway through training

Method

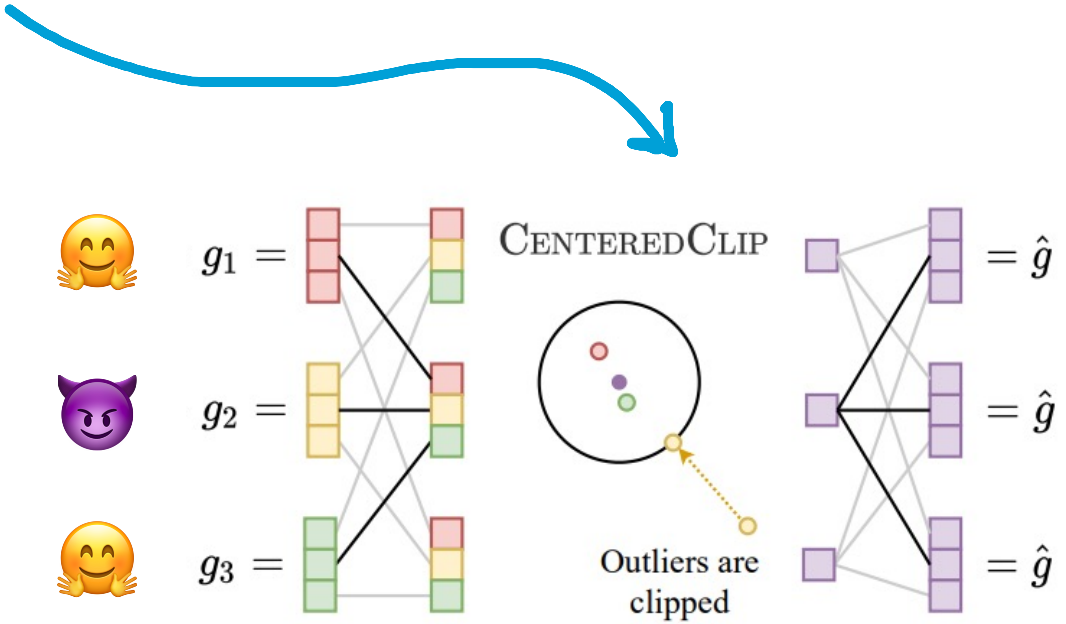
We start with standard Butterfly All-Reduce (Li et al., 2017):



Li, Z., Davis, J., and Jarvis, S. An efficient task-based all-reduce for machine learning applications. In *Proceedings of the Machine Learning on HPC Environments*, MLHPC'17, New York, NY, USA, 2017.

Method

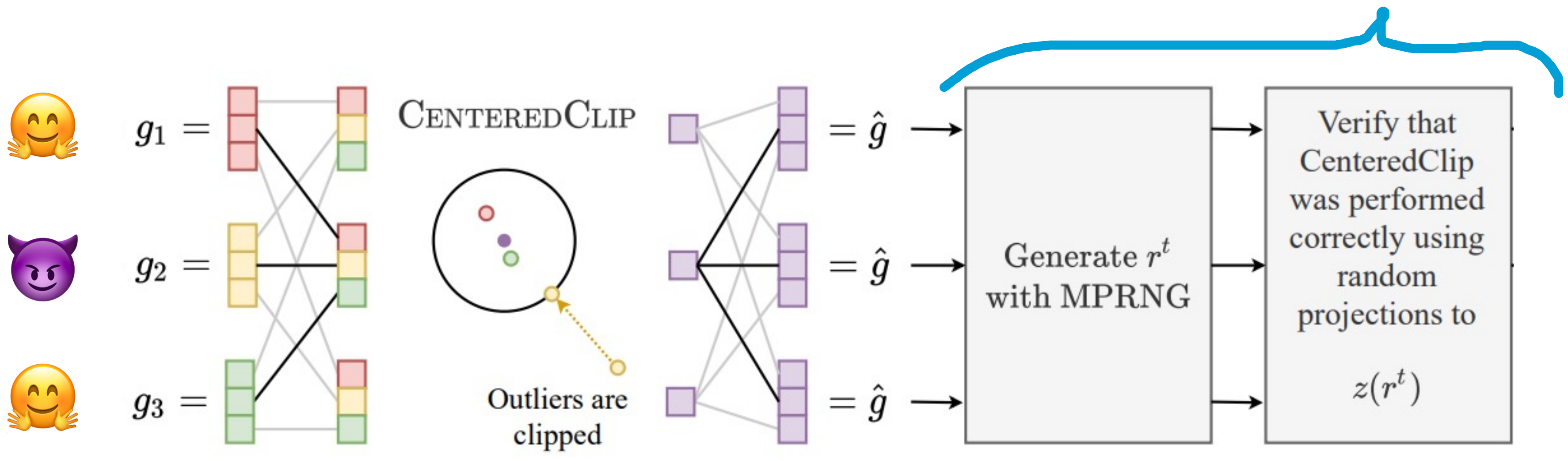
Step 1. To resist large gradient perturbations, we replace the naive averaging with CENTEREDCLIP (Karimireddy et al., 2020)



Karimireddy, Sai Praneeth, Lie He, and Martin Jaggi. "Learning from history for byzantine robust optimization." *International Conference on Machine Learning*. PMLR, 2021.

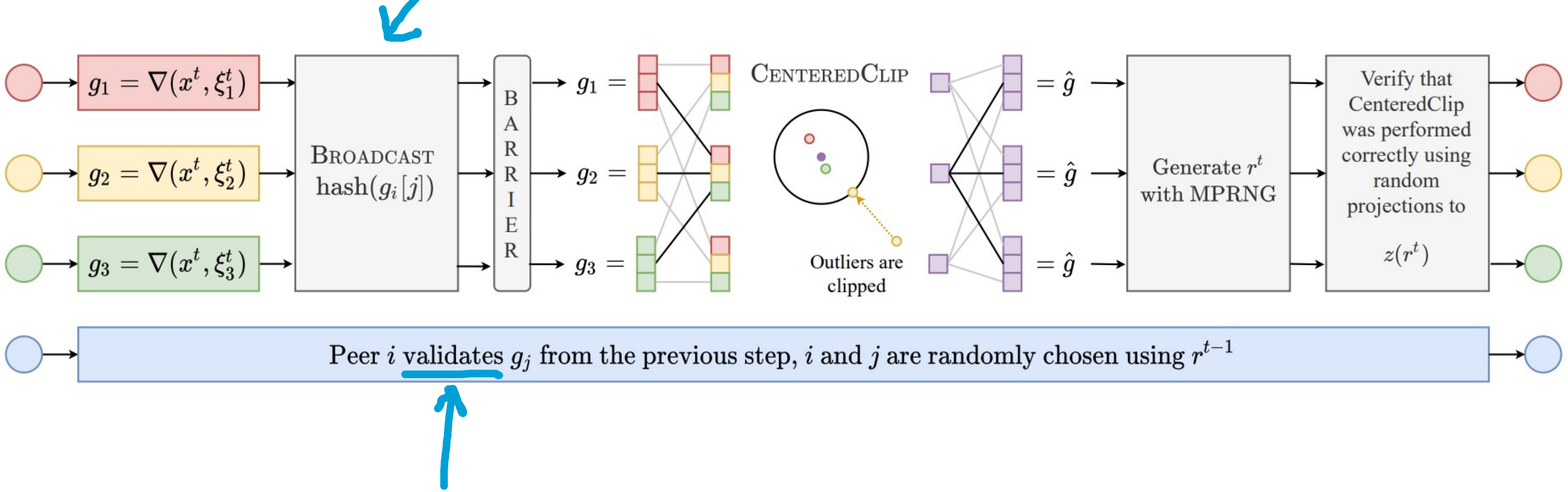
Method

Step 2. Since CENTEREDCLIP is now performed by untrusted peers, we validate that peers are not cheating during the clipping procedure



Method

Step 3. Finally, to resist a series of small gradient perturbations, we periodically verify gradients of random peers by recalculation



Convergence Bounds

- We prove that our method converges to any predefined accuracy under realistic assumptions
- Our convergence rates are **state-of-the-art** in the decentralized Byzantine-tolerant setting
  - And even better than SOTA for the centralized Byzantine-tolerant setting if the required accuracy is high enough

Convergence Bounds

- We prove strong results for non-convex problems, convex, and strongly convex problems

Non-PS?	Work	Non-convex	Convex	Strongly convex
✗	(Alistarh et al., 2018)[(1),(2)]	✗	$\frac{1}{\varepsilon} + \frac{\sigma^2}{n\varepsilon^2} + \frac{\delta^2\sigma^2}{\varepsilon^2}$	$\frac{1}{\mu} + \frac{\sigma^2}{n\mu\varepsilon} + \frac{\delta^2\sigma^2}{\mu\varepsilon}$
	(Allen-Zhu et al., 2021)[(1),(3)]	$\frac{1}{n\varepsilon^4} + \frac{\delta^2}{\varepsilon^4}$	✗	✗
	(Wu et al., 2020)[(4)]	✗	✗	$\frac{L^2}{\mu^2}$ [(5)]
	(Karimireddy et al., 2020)[(6)]	$\frac{1}{\varepsilon^2} + \frac{\sigma^2}{n\varepsilon^4} + \frac{\delta\sigma^2}{\varepsilon^4}$	✗	✗
✓	(Peng et al., 2021)[(6),(7)]	✗	✗	$\frac{1}{\mu\varepsilon} + \frac{n\sigma^2}{\mu^2\varepsilon} + \frac{\lambda^2 d\overline{N}^2}{\mu^2\varepsilon}$
	This work[(8)]	$\frac{1}{\varepsilon^2} + \frac{\sigma^2}{n\varepsilon^4} + \frac{n\delta\sigma^2}{m\varepsilon^2}$	$\frac{1}{\varepsilon} + \frac{\sigma^2}{n\varepsilon^2} + \frac{n\sqrt{\delta}\sigma}{m\varepsilon}$	$\frac{1}{\mu} + \frac{\sigma^2}{n\mu\varepsilon} + \frac{n\sqrt{\delta}\sigma}{m\sqrt{\mu\varepsilon}}$
	This work[(9)]	$\frac{1}{\varepsilon^2} + \frac{\sigma^2}{n\varepsilon^4} + \frac{n^2\delta\sigma^2}{m\varepsilon^2}$	$\frac{1}{\varepsilon} + \frac{\sigma^2}{n\varepsilon^2} + \frac{n^2\delta\sigma}{m\varepsilon}$	$\frac{1}{\mu} + \frac{\sigma^2}{n\mu\varepsilon} + \frac{n^2\delta\sigma}{m\sqrt{\mu\varepsilon}}$
	This work[(10)]	✗	$\left(\frac{G\Lambda_1}{\varepsilon}\right)^{\frac{\alpha}{\alpha-1}}$	$\left(\frac{G^2\Lambda_1}{\mu\varepsilon}\right)^{\frac{\alpha}{2(\alpha-1)}}$
	This work[(11)]	✗	$\left(\frac{G\Lambda_2}{\varepsilon}\right)^{\frac{\alpha}{\alpha-1}}$	$\left(\frac{G^2\Lambda_2}{\mu\varepsilon}\right)^{\frac{\alpha}{2(\alpha-1)}}$

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

Experiments

- Our method **succeeds to protect** the training run from all kinds of attacks, unlike methods from prior work