dSTAR: Straggler Tolerant and Byzantine Resilient Distributed SGD

Jiahe Yan
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90024
yjh020711@g.ucla.edu

Pratik Chaudhari
Department of Electrical and System Engineering
University of Pennsylvania
Philadelphia, PA 19104
pratikac@seas.upenn.edu

Leonard Kleinrock
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90024
lk@cs.ucla.edu

Abstract

Distributed model training needs to be adapted to challenges such as the straggler effect and Byzantine attacks. When coordinating the training process with multiple computing nodes, ensuring timely and reliable gradient aggregation amidst network and system malfunctions is essential. To tackle these issues, we propose *dSTAR*, a lightweight and efficient approach for distributed stochastic gradient descent (SGD) that enhances robustness and convergence. *dSTAR* selectively aggregates gradients by collecting updates from the first k workers to respond, filtering them based on deviations calculated using an ensemble median. This method not only mitigates the impact of stragglers but also fortifies the model against Byzantine adversaries. We theoretically establish that *dSTAR* is (α, f)-Byzantine resilient and achieves a linear convergence rate. Empirical evaluations across various scenarios demonstrate that *dSTAR* consistently maintains high accuracy, outperforming other Byzantine-resilient methods that often suffer up to a 40-50% accuracy drop under attack. Our results highlight *dSTAR* as a robust solution for training models in distributed environments prone to both straggler delays and Byzantine faults.

1 Introduction

Distributed SGD has become a standard way of training large machine learning models due to its scalability and efficiency in processing vast amounts of data in parallel across multiple computing nodes. We consider the classical setting with a single parameter server and N workers [1]. Given $X \in \mathbb{R}^{m \times d}$ representing m d-dimensional data, $y \in \mathbb{Z}^m$ where each element of y is the discrete label of the respective row in X, and a loss function $F(\theta)$ for the dataset, where θ are the model parameters, the parameter server wants to find θ^* that minimizes the loss function F. During each iteration, the parameter server sends model parameters θ to all workers. Each worker contains a unique subset of X to parallelize gradient computation. The worker computes and returns the gradient of θ on the local dataset to the server, which then aggregates the gradients to perform stochastic gradient descent.

While distributed SGD offers enhanced scalability and acceleration, it also introduces fault tolerance concerns in distributed systems. Workers in a distributed system can be Byzantine faulty. The identity of such Byzantine workers is also a priori unknown. Byzantine workers may produce wrong or even

AdvML-Frontiers'24: The 3nd Workshop on New Frontiers in Adversarial Machine Learning@NeurIPS'24, Vancouver, CA.

malicious results back to the parameter server due to various reasons, from system failure to malicious attacks [2]. Averaging, which is the simplest way to aggregate gradients from workers, has been proven fragile to even one worker being Byzantine [3]. To confer Byzantine resilience in distributed SGD, many Gradient Aggregation Rules (GARs) have been proposed to allow learning to occur under a (maximum) number of f Byzantine workers under synchronous and asynchronous settings. The maximum f that can be tolerated is called the *breakdown point*, with the *optimal breakdown point* being $N > 2f$ [3]. That is, as long as the majority of workers are honest, model training can proceed. However, these GARs come with their own challenges. In synchronous SGD, the parameter server needs to wait for slow or unresponsive nodes known as stragglers [4]. In asynchronous SGD, the server will update the model parameter as soon as any worker returns a gradient to avoid stragglers [5]. However, this leads to a smaller batch size per aggregation, effectively introducing noise to the model. Additionally, the server may also receive "stale gradients" computed from outdated θ, potentially causing the model to converge more slowly or even diverge.

To address the dual challenges of Byzantine resilience and straggler tolerance, we present *dSTAR*, a new Byzantine-resilient distributed SGD that selectively waits for k gradients from the fastest workers, selected using a filter that calculates deviations of worker gradients from an ensemble median (where $1 \leq k \leq N$ and is adaptive). In fault-free settings, the fastest-k SGD (or formally, synchronous SGD with backup workers) has been shown to achieve optimal performance as synchronous SGD while mitigating the straggler effect [6]. In the fastest-k SGD, the parameter server only waits for the fastest k workers per iteration before making a gradient descent update. Other gradients will simply be dropped. If we assume the response time of each worker is $i.i.d$, it can be shown that the fastest-k SGD is equivalent to the single-node batch SGD since the server updates based on a uniformly random set of gradients. However, the fastest-k SGD is vulnerable to Byzantine attacks. By definition, Byzantine workers can return gradients anytime they want, whereas the response time of a non-Byzantine worker can be unbounded. Hence, Byzantine workers can always be in the fastest k and compromise training. In this paper, we introduce a new fastest-k variant that can be robust under Byzantine attack as long as the majority of nodes are honest. We show that *dSTAR* consistently produces optimal models under different Byzantine attacks, model architecture, and datasets while other GARs can experience performance drops of 40-50%. Furthermore, since k is adjustable, *dSTAR* offers a configurable spectrum from fully asynchronous to fully synchronous operation. This flexibility allows for tailoring the system dynamics based on specific requirements and constraints of the deployment environment.

2 Related work

Formally, a GAR is robust to Byzantine attacks if it satisfies (α, f)-Byzantine resilience [3]:

Definition 2.1 ((α, f)-Byzantine Resilience)**.** Let $\alpha \in [0, \frac{\pi}{2}]$, $f \in [0, n]$. Let $V_1, V_2, \ldots, V_n$ be any independent identically distributed random vectors in $\mathbb{R}^d$ such that $V_i \sim G$, with $\mathbb{E}[G] = \nabla F$. Let $B_1, B_2, \ldots, B_f$ be any random vectors $\in \mathbb{R}^d$, possibly dependent on the V_i's. An aggregation algorithm A is said to be (α, f)-Byzantine resilient if, for any $1 \leq j_1 < \ldots < j_f \leq n$, the vector $A = A(V_1, \ldots, \underbrace{B_1}_{j_1}, \ldots, \underbrace{B_f}_{j_f}, \ldots, V_n)$ satisfies: 1) $\langle \mathbb{E}[A], \nabla F \rangle \geq (1 - \sin(\alpha))\|\nabla F\|^2 > 0$, and 2) for any $r \in \{2, 3, 4\}$, $\mathbb{E}\|A\|^r$ is bounded above by a linear combination of terms $\mathbb{E}\|G\|^{r_1}, \ldots, \mathbb{E}\|G\|^{r_{n-1}}$ with $r_1 + \ldots + r_{n-1} = r$.

Existing GARs ensure (α, f)-Byzantine Resilience by employing robust statistics to identify candidate gradients to aggregate. Most GARs focus on the fully synchronous setting where all gradients will be collected before applying the aggregation rule. Examples of synchronous GARs are as follows: a). AKSEL averages a subset of gradients based on their squared distances to the coordinate-wise median [7], b). KRUM chooses the gradient with the smallest sum of Euclidean distances with neighbors [3], c). CGE averages a subset of gradients with the smallest norms [8], d). TrMean discards extreme values and aggregates the top $(N - b)$ gradients nearest to the median where b is a hyperparameter [9]. A few algorithms such as KARDAM and Zeno++ focus on the asynchronous setting, where the model can be updated as soon as any gradient is returned. KARDAM uses a sliding window based on gradient aggregation history and empirical Lipschitzness of gradients to filter for good gradients [10]. Zeno++ chooses candidate gradients that lead to a greater descent of the loss value based on a validation set on the parameter server [11]. Nevertheless, synchronous GARs suffer from

stragglers and asynchronous GARs may produce suboptimal models. KARDAM can only support up to one-third of Byzantine workers. Zeno++ also requires manually configuring a gradient threshold, which can be can be time-consuming to optimize. Zeno++ further has a model error bound that is influenced by the presence of asynchronous noise, which can be substantial if stale gradients are utilized more than sparingly. Moreover, asynchronous GARs suffer from "stale gradients" computed from outdated model parameters.

3 Contributions

Traditional synchronous GARs mandate the collection of all workers' gradients for each iteration to ensure convergence because they depend on statistical measures within each iteration. To achieve optimal convergence without waiting for all gradients, *dSTAR* focuses on statistics gained from the training history via a validation set approach similar to Zeno++. The parameter server keeps a unique subset of X as the validation set locally and computes its validation gradient to compare against incoming gradients. Unlike traditional approaches, *dSTAR* determines a filtering threshold dynamically based on the historical ensemble median. *dSTAR* further achieves optimal time complexity and breakdown point as shown in Table 1. The key contributions of our work include: 1). Proposed a new SGD that addresses the straggler effect by waiting for only the k fastest gradients with a dynamically configured filtering threshold while being robust against Byzantine attacks; 2). Showed empirically that the SGD can consistently produce an optimal model; 3). Showed theoretically that the SGD has a linear convergence rate and is Byzantine-resilient.

Table 1: Comparison of different gradient aggregation rules

Method	**Time Complexity**	**Breakdown Point**
Average	$O(Nd)$	$f = 0$
AKSEL	$O(Nd)$	$n > 2f$
TrMean	$O(Nd)$	$n > 2f$
KRUM	$O(N^2 d)$	$n > 2f + 1$
CGE	$O(N(d + \log N))$	$n > 2f$
dSTAR	$\mathbf{O(Nd)}$	$\mathbf{n > 2f}$

4 Assumptions

A1 (Unbiased gradients with bounded variance) The proposed gradient g_i from the set of honest workers S_h are d-dimensional vectors and unbiased estimates of the true gradient and have bounded variance:

$$\forall i \in S_h, g_i \sim G, E[G] = \nabla F, E[G_j - \nabla F_j]^2 = \sigma_j^2, E\|G - \nabla F\|^2 = E\sum_{j=1}^{d}[G_j - \nabla F_j]^2 = d\sigma^2$$

A2 (Lipschitz gradients) The loss function F is Lipschitz continuous with $L > 0$:

$$\forall \theta_1, \theta_2, \|\nabla F(\theta_1) - \nabla F(\theta_2)\| \leq L \|\theta_1 - \theta_2\|$$

A3 (Bounded gradients) The gradients g_i from honest workers and g_v from validation set are all upper bounded by V, the validation set gradient is also lower bounded by V' [11]:

$$\|g_i\|^2 \leq V,\ V' \leq \|g_v\|^2 \leq V,\ 0 < V' \leq V$$

5 Algorithm

We present our new algorithm with its theoretical analysis. Algorithm 1 in the Appendix describes the full pseudo training loop code. *dSTAR* aggregation involves evaluating each received gradient against two key metrics calculated from the validation gradient derived from the parameter server's validation set: the dot product and the squared Euclidean distance. Unlike Zeno++, which requires manually configuring a threshold, *dSTAR* compares both values against the values calculated using

the historical median. The median in a system with optimal breakdown point is robust to Byzantine attack [12]. During the first iteration, we default to aggregate the median of all gradients (i.e. a fully synchronous iteration using MEDIAN GAR) since history is unknown. This procedure serves as a warm-up phase for the filtering of subsequent iterations.

During each subsequent iteration t, given an incoming gradient g_i^t and the local validation set gradient g_v^t, the server computes normalized Euclidean distance $s_i^t = \frac{\|g_i^t - g_v^t\|^2}{\|g_v^t\|}$ and dot product $d_i^t = \left\langle \frac{g_i^t}{\|g_v^t\|}, \frac{g_v^t}{\|g_v^t\|} \right\rangle$. If s_i^t is less than or equal to the normalized Euclidean distance calculated using the historical median gradient and validation set gradient and d_i^t is greater than or equal to the normalized dot product calculated using the historical median gradient and validation set gradient, g_i^t is added to an accepted list. The collection phase stops once k gradients are accumulated or all workers have responded. Since gradients can vary significantly in magnitude across iterations, we included normalization in the calculation for Euclidean distance and the dot product to maintain a consistent scale relative to the validation gradient when evaluating the incoming gradients. The accepted gradients are then averaged to calculate the aggregated gradient $g_{\text{agg}}^t = \frac{1}{k} \sum_{j=1}^{k} g_{\text{accepted}_j}^t$, and the model parameters are updated accordingly: $\theta^{t+1} = \theta^t - \eta g_{\text{agg}}^t$. In experiments, we show that by simply using the first iteration median gradient and validation gradient as this historical threshold, *dSTAR* already reaches top performance. In theory, extending the warmup period to more rounds may improve performance further.

5.1 Time complexity

Calculating Euclidean distance and dot product are both $O(Nd)$. For the first iteration, finding the median using quick select is also $O(N)$ [13]. For all subsequent iterations, the algorithm simply retrieves the recorded median values and evaluates each incoming gradient against these metrics. Hence, the total time complexity for this algorithm is $O(Nd)$. In practice, the effective time complexity is often lower than this theoretical upper bound as $k < N$. Notably, *dSTAR* has a much lower time complexity than methods like KRUM ($O(N^2 d)$), which requires pairwise comparisons among gradients, and CGE ($O(N(d + \log N))$), which requires sorting N gradients per iteration.

5.2 Byzantine resilience analysis

We show that *dSTAR* is $(\alpha - f)$-Byzantine resilient. First, it is important to point out the robustness of the median. For a sequence of higher-dimensional vectors with the optimal breakdown point, the coordinate-wise median will always lie within the range defined by the minimum and maximum values of the honest coordinates for that dimension [12]. Based on this, we illustrate that the aggregated gradient of each iteration satisfies the following two lemmas:

Lemma 5.1. *(Proof in the appendix) Under assumptions A1 to A3, if g_*^t denotes the aggregated gradient for iteration t, it satisifies:*

$$\langle \mathbb{E}[g_*^t], \nabla F \rangle \geq \left(\|\nabla F\| - \sqrt{\frac{2(n-f)}{k} d\sigma^2} (\frac{V}{V'})^{\frac{1}{4}} \right) \|\nabla F\| \tag{1}$$

Lemma 5.2. *(Proof in the appendix) Under assumptions A1 to A3, if g_*^t denotes the aggregated gradient for iteration t, it is upper bounded by a linear combinations of $\mathbb{E}\|G\|^{r_1}, ..., \mathbb{E}\|G\|^{r_{n-1}}$*

Given the two lemmas, *dSTAR* is $(\alpha - f)$-Byzantine resilient under the optimal breakdown point:

Theorem 5.3. *Let $g_1^t, \ldots, g_n^t, g_v^t$ be i.i.d. d-dimensional gradients at iteration t such that $g_i^t \sim G$, with $\mathbb{E}[G] = \nabla F$ and $\mathbb{E}\|G - \nabla F\|^2 = d\sigma^2$. f of $\{g_1^t, \ldots, g_n^t\}$ are replaced by arbitrary values. The dSTAR function selects and aggregates $g_1^t, \ldots, g_k^t$ where $k \leq n$. If $n > 2f$ and $\sqrt{\frac{2(n-f)}{k} d\sigma^2} \left(\frac{V}{V'}\right)^{\frac{1}{4}} < \|\nabla F\|$, then the dSTAR function is (α, f)-Byzantine resilient where $0 \leq \alpha < \frac{\pi}{2}$ is defined by:*

$$\sin \alpha = \frac{\sqrt{\frac{2(n-f)}{k} d\sigma^2} \left(\frac{V}{V'}\right)^{\frac{1}{4}}}{\|\nabla F\|} \tag{2}$$

Remark 5.4. The condition on the norm of the gradient is standard in Byzantine resilience analysis [3]. It can be satisfied at least to some extent by computing gradients using mini-batches on workers. Averaging gradients over a mini-batch divides σ by the squared root of the mini-batch size [14].

5.3 Convergence analysis

Theorem 5.5. *(Proof in appendix) Assume $F(\theta)$ is L smooth, and there exists a global minimum θ^* where $F(\theta^*) \leq F(\theta)\ \forall_\theta$, then after training for T iterations, dSTAR has expected error bound: $\mathbb{E}\left[F(\theta^*) - F(\theta^0)\right] \leq \sum_{t=0}^{T} -\eta\frac{V}{V'}\|\nabla F(\theta^1)\|^2 + \mathbb{O}(V + d\sigma^2)$ where $\nabla F(\theta^{t'})$ represents the honest gradient at certain iteration t'.*

6 Experiments

In this section, we detail the empirical evaluation of *dSTAR*. We evaluated the algorithm and other synchronous GARs on two standard image classification benchmarks: Fashion-MNIST and CIFAR10, with LeNet-5 and ResNet18 architectures respectively. We assessed the resilience of each algorithm by subjecting them to two state-of-the-art Byzantine attacks:

- **"Little" [15]:** The attack disrupts the median gradient computation by introducing spurious gradients that cluster around the mean. Specifically, given N workers in which f workers are Byzantine, the attack: 1). computes the number of required workers for a majority $s = \left\lfloor \frac{N}{2} + 1 \right\rfloor - f$; 2). calculates the maximum z-value, $z_{\max}$, from the standard normal distribution such that the cumulative probability $\phi(z) < \frac{N-s}{N}$; 3). generates a malicious gradient $g_{\text{mal}} = \mu + z_{\max} \cdot \sigma$, using the mean μ and standard deviation σ of non-Byzantine gradients.
- **"Empire" [16]:** The attack employs inner product manipulation to break Byzantine-tolerant GARs. The attack uses the fact that, for gradient descent algorithms to guarantee the descent of the loss, the inner product between the true gradient and the aggregated gradient must be non-negative. Hence, malicious gradients can be generated to make the aggregated gradient point in the opposite direction as the true gradient ($g_{\text{mal}} = -s\mu$) where μ is the honest gradient mean and s is a configurable scaling factor.

We simulate a distributed environment with 25 workers and a Byzantine ratio of 35%. Each worker contains a unique subset of the dataset, comprising random samples across all classes. Network delays are modeled using an exponential distribution with rate β. Honest workers have $\beta = 0.2$ and Byzantine workers have $\beta = 0.001$. The value of β makes no difference for synchronous GARs because they need to wait for all nodes, but for *dSTAR* it makes faulty workers significantly more likely to be in the fastest k, thereby exposing the vulnerability of vanilla fastest-k algorithm. For *dSTAR*, the initial k is set as 8, and the time to aggregate gradients in each iteration will be the time to accept k gradients or the maximum response time from all nodes if our filter cannot accept k gradients, in which case *dSTAR* waits for all nodes to return but only aggregate the accepted ones. For all experiments, we used the Adam optimizer with an initial learning rate of 0.001. The preprocessing steps for Fashion-MNIST included converting images into tensors and normalizing them. For CIFAR10, images are padded on all sides with 4 pixels, randomly cropped into 32×32 pixels, randomly flipped horizontally, and converted to tensors and normalized. Additionally, for CIFAR10, we implemented a cosine annealing scheduler to adjust the learning rate, with a minimum rate set at 0.0001. We also utilized Mixup for data augmentation with a parameter α of 0.4. These preprocessing are added to make the fault-free baseline comparable to SOTA for accurate comparisons.

7 Results

In three of the four experiments, *dSTAR* achieved top accuracy (see Table 2 and 3). Furthermore, *dSTAR* maintains a consistent performance across different Byzantine attacks, whereas other synchronous GARs may have up to 40-50$\%$ drop between the two attacks. This uniformity in performance under various adversarial conditions underscores the robustness and generalized ability of *dSTAR*. The performance of *dSTAR* is particularly notable under the "Empire" attack scenarios, where it is the only algorithm that converges. The full training curves can be found in the Appendix Figures 1 to 4.

Additionally, the goal of designing a fastest-k Byzantine resilient SGD is to mitigate the straggler effect. It has been shown in Table 4 that the selective waiting strategy for k fastest gradients significantly reduces the time required for gradient aggregation per iteration. The reduced wait times can contribute to higher throughput and efficiency, making *dSTAR* particularly suited for time-sensitive applications.

The only setting where our proposed algorithm didn't achieve the best accuracy was on CIFAR10 under the "Little" attack, although the performance is still significantly better than TRMEAN and KRUM and is only 2% lower than CGE. This can be explained by a tradeoff between accuracy and speed, as the accuracy will almost surely improve by waiting for more workers at the cost of a longer waiting time per iteration. Additionally, we default to MEDIAN for the initial iteration, which can be susceptible to the attack since "Little" was designed specifically for MEDIAN GAR. Choosing a different synchronous GAR for the initial iteration or having a longer warm-up phase may also improve performance.

Table 2: Fashion-MNIST accuracies of methods under Little and Empire attacks, including the fault-free baseline.

Method	**Little (%)**	**Empire (%)**	**Fault-Free (%)**
dSTAR	**88.78**	**88.87**	88.86
Trmean	16.55	32.48	89.44
Krum	88.19	40.84	88.22
CGE	88.30	82.44	89.47
Aksel	88.51	75.08	88.67
Average	-	-	**89.65**

Table 3: CIFAR10 accuracies of methods under Empire and Little attacks. The dashed columns indicate that the algorithm failed to converge. The fault-free baseline has an accuracy of 94.33%.

Method	**Empire (%)**	**Little (%)**	**Fault-Free (%)**
dSTAR	91.11	**91.60**	91.23
Trmean	20.50	11.85	93.72
Krum	76.32	10.00	80.38
CGE	**93.45**	41.32	94.19
Aksel	92.44	46.62	93.64
Average	-	-	**94.33**

Table 4: Average time between iterations for synchronous GARs and *dSTAR*

GAR	**Average Time Between Iterations (s)**
Synchronous GAR	7.62
dSTAR	**3.79**

8 Discussion and conclusion

We introduced *dSTAR*, a novel Byzantine resilient distributed SGD algorithm that effectively balances the dual challenges of mitigating straggler effects and defending against adversarial Byzantine attacks in synchronous settings. The experimental results demonstrated that *dSTAR* is robust to various adversarial settings, whereas other synchronous GARs can have performance degradation when facing different Byzantine attacks. The ability of *dSTAR* to deliver such results highlights its potential as a reliable solution for securing distributed SGD processes against an array of threats while ensuring minimal disruption to operational efficiency.

Future work could involve scaling the experiments to more complex models and datasets to provide a more comprehensive understanding of the algorithm's performance and potential adjustments.

Extending our experiments will help ascertain the generalizability of our findings across various domains and applications. Moreover, the integration of *dSTAR* with emerging machine learning paradigms, such as federated learning, represents a promising research direction as well.

References

[1] Martin Abadi et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems 30*, pages 119–129. Curran Associates, Inc., 2017.

[4] J. Dean and L. A. Barroso. The tail at scale. *Communications of the ACM*, 56(2):74–80, 2013.

[5] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[6] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[7] Amine Boussetta et al. Aksel: Fast byzantine sgd. In *24th International Conference on Principles of Distributed Systems (OPODIS 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

[8] Nirupam Gupta, Shuo Liu, and Nitin H. Vaidya. Byzantine fault-tolerant distributed machine learning using stochastic gradient descent (sgd) and norm-based comparative gradient elimination (cge). *arXiv preprint arXiv:2008.04699*, 2020.

[9] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5650–5659. PMLR, 2018.

[10] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, Mahsa Taziki, et al. Asynchronous byzantine machine learning (the case of sgd). In *ICML*, pages 1153–1162, 2018.

[11] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno++: Robust fully asynchronous sgd. In *International Conference on Machine Learning*. PMLR, 2020.

[12] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.

[13] Manuel Blum et al. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.

[14] Léon Bottou. Online learning and stochastic approximations. *Online learning in neural networks*, 17(9):142, 1998.

[15] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems 32*, 2019.

[16] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. *Uncertainty in Artificial Intelligence. PMLR*, 2020.

A Appendix

A.1 Byzantine resilience analysis

First, it is important to point out the robustness of the median. Formally, the median value given the optimal breakdown point is always bounded by two honest values and is Byzantine resilient. We restate Lemma 4 from [12] without proof:

Lemma A.1. *For a sequence composed of f Byzantine values and $n - f$ honest values $x_1, x_2, \ldots, x_{n-f}$, if $f \leq \lceil \frac{n}{2} \rceil - 1$ (the honest values dominate the sequence), then the median value m of this sequence satisfies $m \in [x_{\min}, x_{\max}]$.*

For a sequence of higher-dimensional vectors, the coordinate-wise median maintains the same robustness properties [12]. Specifically, the median for each coordinate will always lie within the range defined by the minimum and maximum values of the honest coordinates for that dimension. Following this lemma, we illustrate that *dSTAR* is Byzantine resilient. For the first iteration, we default to MEDIAN aggregator which is already Byzantine resilient. For any subsequent iteration t, we accept a gradient if its normalized Euclidean distance to the validation gradient of iteration t is not greater than the normalized Euclidean distance of the first iteration coordinate-wise median to the first iteration validation gradient. If we denote the first iteration coordinate-wise median as g_m, the first iteration validation gradient as g_v^1, the t-th iteration validation gradient g_v^t, and an arbitrary gradient received from worker i during iteration t as g_i^t, then we accept g_i^t if the following two inequalities hold:

$$\frac{\|g_i^t - g_v^t\|^2}{\|g_v^t\|} \leq \frac{\|g_m - g_v^1\|^2}{\|g_v^1\|} \tag{3}$$

$$\langle \frac{g_m}{\|g_v^1\|}, \frac{g_v^1}{\|g_v^1\|} \rangle \leq \langle \frac{g_i^t}{\|g_v^t\|}, \frac{g_v^t}{\|g_v^t\|} \rangle \tag{4}$$

From (3), we have:

$$\|g_i^t - g_v^t\|^2 \leq \frac{\|g_v^t\|}{\|g_v^1\|} \|g_m - g_v^1\|^2 \tag{5}$$

Assume all gradients are d-dimensional and come from the same distribution G where $\mathbb{E}\left[G_i - \nabla F_i\right]^2 = \sigma_i^2$ and $\mathbb{E}\|G - \nabla F\|^2 = \mathbb{E}\sum_{i=1}^{d}\left[G_i - g_i\right]^2 = d\sigma^2$, we have:

$$\begin{aligned} \mathbb{E}\|g_m - g_v^1\|^2 &= \mathbb{E}[\sum_{j=1}^{d}((g_m)_j - (g_v^1)_j)^2] \\ &= \sum_{j=1}^{d} \mathbb{E}[(g_m)_j - (g_v^1)_j]^2 \end{aligned} \tag{6}$$

where $(g_m)_j$ represents the j-th dimension of the vector. Since g_m is the coordinate-wise median over first iteration gradients, we have $(g_m)_j \in \left[\min_{\text{correct } i}(g_i^1)_j, \max_{\text{correct } i}(g_i^1)_j\right]$. We thus have:

$$\begin{aligned} \mathbb{E}[(g_m)_j - (g_v^1)_j]^2 &\leq \mathbb{E}\left[\max_{\text{correct } i}\left((g_i^1)_j - (g_v^1)_j\right)^2\right] \\ &\leq \mathbb{E}[\sum_{\text{correct } i}((g_i^1)_j - (g_v^1)_j)^2] \\ &= \sum_{\text{correct } i}\mathbb{E}[((g_i^1)_j - (g_v^1)_j)^2] \\ &= (n-f)\mathbb{E}[((g_i^1)_j - (g_v^1)_j)^2] \\ &= (n-f)2\sigma_j^2 \end{aligned} \tag{7}$$

Thus, we can plug this back to (6) and obtain:

$$\begin{aligned}\mathbb{E}\|g_m - g_v^1\|^2 &= \sum_{j=1}^{d} \mathbb{E}[(g_m)_j - (g_v^1)_j]^2 \\ &\leq \sum_{j=1}^{d} 2(n-f)\sigma_j^2 \\ &\leq 2(n-f)d\sigma^2\end{aligned} \tag{8}$$

With Assumption A3, this gives us an upper bound for the expectation of (5):

$$\begin{aligned}\mathbb{E}\|g_i^t - g_v^t\|^2 &\leq \mathbb{E}\frac{\|g_v^t\|}{\|g_v^1\|}\|g_m - g_v^1\|^2 \\ &\leq 2(n-f)\frac{\sqrt{V}}{\sqrt{V'}}d\sigma^2\end{aligned} \tag{9}$$

Now, we begin to prove the Byzantine Resilience of our algorithm.

Theorem A.2. *Let $g_1^t, \ldots, g_n^t, g_v^t$ be i.i.d. d-dimensional gradients at iteration t such that $g_i^t \sim G$, with $\mathbb{E}[G] = \nabla F$ and $\mathbb{E}\|G - \nabla F\|^2 = d\sigma^2$. f of $\{g_1^t, \ldots, g_n^t\}$ are replaced by arbitrary values. The dSTAR function selects and aggregates $g_1^t, \ldots, g_k^t$ where $k \leq n$. If $n > 2f$ and $\sqrt{\frac{2(n-f)}{k}d\sigma^2}\left(\frac{V}{V'}\right)^{\frac{1}{4}} < \|\nabla F\|$, then the dSTAR function is (α, f)-Byzantine resilient where $0 \leq \alpha < \frac{\pi}{2}$ is defined by:*

$$\sin\alpha = \frac{\sqrt{\frac{2(n-f)}{k}d\sigma^2}\left(\frac{V}{V'}\right)^{\frac{1}{4}}}{\|\nabla F\|} \tag{10}$$

Proof. We first focus on the condition (i) of Byzantine Resilience. Suppose we denote the final aggregated gradient during iteration t as g_*^t, we want to determine an upper bound on $\|\mathbb{E}[g_*^t] - \nabla F\|^2$. If Assumption 1 holds, we have:

$$\begin{aligned}\|\mathbb{E}[g_*^t] - \nabla F\|^2 &\leq \|\mathbb{E}\left(\, g_*^t - g_v^t\right)\|^2 \\ &\leq \mathbb{E}\|\, g_*^t - g_v^t\|^2 \\ &= \mathbb{E}\|\frac{1}{k}\sum_{j=1}^{k}(g_j^t - g_v^t)\|^2 \\ &\leq \frac{1}{k^2}\sum_{j=1}^{k}\mathbb{E}\|g_j^t - g_v^t\|^2 \\ &\leq \frac{2(n-f)}{k}\frac{\sqrt{V}}{\sqrt{V'}}d\sigma^2\end{aligned} \tag{11}$$

If $\sqrt{\frac{2(n-f)}{k}d\sigma^2}(\frac{V}{V'})^{\frac{1}{4}} \leq \|\nabla F\|$, $\mathbb{E}[g_*^t]$ belongs to a ball centered at ∇F with radius $\sqrt{\frac{2(n-f)}{k}d\sigma^2}(\frac{V}{V'})^{\frac{1}{4}}$. This implies:

$$\begin{aligned}\langle \mathbb{E}[g_*^t], \nabla F\rangle &\geq \left(\|\nabla F\| - \sqrt{\frac{2(n-f)}{k}d\sigma^2}(\frac{V}{V'})^{\frac{1}{4}}\right)\|\nabla F\| \\ &= (1 - sin\alpha)\|\nabla F\|^2\end{aligned} \tag{12}$$

So condition (i) of Byzantine Resilience holds when $\sqrt{\frac{2(n-f)}{k}d\sigma^2}(\frac{V}{V'})^{\frac{1}{4}} \leq \|\nabla F\|$. Now we focus on condition (ii). For an accepted gradient g_j^t at iteration t with validation gradient g_v^t, there exists a constant C such that:

$$\begin{aligned}\|g_j^t\| &\leq \|g_j^t - g_v^t\| + \|g_v^t\| \\ &\leq (\frac{V}{V'})^{\frac{1}{4}}\|g_m - g_v^1\| + \|g_v^t\|\end{aligned} \tag{13}$$

$$\begin{aligned}\|g_m - g_v^1\| &= \sqrt{\sum_{j=1}^{d}[(g_m)_j - (g_v^1)_j]^2} \\ &\leq \sqrt{\sum_{j=1}^{d}\max_{\text{correct } i}[(g_i^1)_j - (g_v^1)_j]^2} \\ &\leq \sqrt{\sum_{j=1}^{d}\sum_{\text{correct } i}[(g_i^1)_j - (g_v^1)_j]^2} \\ &\leq \sqrt{\sum_{\text{correct } i}\|g_i^1 - g_v^1\|^2} \\ &\leq C\sum_{\text{correct } i}\|g_i^1 - g_v^1\| \\ &\leq C\sum_{\text{correct } i}\|g_i^1\| + \|g_v^1\|\end{aligned} \tag{14}$$

Putting this back to (13), we have:

$$\|g_k^t\| \leq \|g_v^t\| + C\sum_{\text{correct } i}\|g_i^1\| + \|g_v^1\| \tag{15}$$

Since all terms on the right side are from correct gradients, we can conclude that the norm of each accepted gradient can be bounded by the norm of honest gradients. By triangle inequality, $\|g_*^t\| = \|\frac{1}{k}\sum_k g_k^t\| \leq \frac{1}{k}\sum_k \|g_k^t\|$. So $\mathbb{E}\|g_*^t\|^r$ is upper bounded by linear combinations of $\mathbb{E}\|G\|^{r1}, ..., \mathbb{E}\|G\|^{r_{n-1}}$. Because both conditions are met, we can conclude that dSTAR is Byzantine resilient. □

A.2 Convergence analysis

For iteration t, we denote the k gradients *dSTAR* collects as $g^t = \{g_1^t, g_2^t, ..., g_k^t\}$ and the validation gradient is g_v^t. From our assumptions and Byzantine Resilience proof, we know $V' \leq \|g_v^t\|^2 \leq V$ and $\|g^t\|^2 \leq CV$ for some constant C. Assume $F(\theta)$ captures the loss of θ and is L smooth, and there exists a global minimum θ^* where $F(\theta^*) \leq F(\theta)\ \forall_\theta$, we want to find the error bound for the expected difference $\mathbb{E}[F(\theta^T) - F(\theta^*)]$ after training our model for T iterations, which can be derived using a similar approach as [11].

From smoothness, we have:

$$F(\theta^t) \leq F(\theta^{t-1}) + \langle\nabla F(\theta^{t-1}), \theta^t - \theta^{t-1}\rangle + \frac{L}{2}\|\theta^t - \theta^{t-1}\|^2 \tag{16}$$

For gradient descent update, $\theta^t = \theta^{t-1} - \eta g^t$:

$$
\begin{aligned}
F(\theta^t) &\leq F(\theta^{t-1}) + \langle \nabla F(\theta^{t-1}), -\eta g^t \rangle + \frac{L}{2} \| - \eta g^t \|^2 \\
&\leq F(\theta^{t-1}) + \langle \nabla F(\theta^{t-1}), -\eta g^t \rangle + \frac{L\eta^2}{2} \| g^t \|^2 \\
&\leq F(\theta^{t-1}) + \langle \nabla F(\theta^{t-1}), -\eta g^t \rangle + \frac{L\eta^2}{2} CV,
\end{aligned} \tag{17}
$$

Now we focus on the dot product term:

$$
\begin{aligned}
\langle \nabla F(\theta^{t-1}), -\eta g^t \rangle &= \langle \nabla F(\theta^{t-1}) - g_v^t + g_v^t, -\eta g^t \rangle \\
&= \langle \nabla F(\theta^{t-1}) - g_v^t, -\eta g^t \rangle + \langle g_v^t, -\eta g^t \rangle \\
&\leq \eta \| \nabla F(\theta^{t-1}) - g_v^t \| \, \| g^t \| + \langle g_v^t, -\eta g^t \rangle \\
&\leq \frac{\eta}{2} \| \nabla F(\theta^{t-1}) - g_v^t \|^2 + \frac{\eta}{2} CV + \langle g_v^t, -\eta g^t \rangle
\end{aligned} \tag{18}
$$

Using triangle inequality, we know:

$$
\begin{aligned}
\| \nabla F(\theta^{t-1}) - g_v^t \|^2 \leq 2 \| \nabla F(\theta^{t-1}) - \nabla F(\theta^t) \|^2 + \\
2 \| \nabla F(\theta^t) - g_v^t \|^2
\end{aligned} \tag{19}
$$

$$
\begin{aligned}
\langle \nabla F(\theta^{t-1}), -\eta g^t \rangle \leq \eta \| \nabla F(\theta^{t-1}) - \nabla F(\theta^t) \|^2 + \\
\eta \| \nabla F(\theta^t) - g_v^t \|^2 + \frac{\eta}{2} CV + \\
\langle g_v^t, -\eta g^t \rangle
\end{aligned} \tag{20}
$$

From Assumption 1, $\mathbb{E} \left\| \nabla F(\theta^t) - g_v^t \right\|^2 \leq d\sigma^2$:

$$
\begin{aligned}
\langle \nabla F(\theta^{t-1}), -\eta g^t \rangle \leq \eta \| \nabla F(\theta^{t-1}) - \nabla F(\theta^t) \|^2 + \\
\eta d\sigma^2 + \frac{\eta}{2} CV + \langle g_v^t, -\eta g^t \rangle
\end{aligned} \tag{21}
$$

Using smoothness, we know:

$$
\begin{aligned}
\| \nabla F(\theta^{t-1}) - \nabla F(\theta^t) \|^2 &\leq L^2 \| \theta^{t-1} - \theta^t \|^2 \\
&\leq L^2 \| \eta g^t \|^2 \\
&\leq L^2 \eta^2 CV
\end{aligned} \tag{22}
$$

Now $\langle \nabla F(\theta^{t-1}), -\eta g^t \rangle$ is upper bounded by:

$$
\langle \nabla F(\theta^{t-1}), -\eta g^t \rangle \leq L^2 \eta^3 CV + \eta d\sigma^2 + \frac{\eta}{2} CV + \langle g_v^t, -\eta g^t \rangle \tag{23}
$$

From (4), we know the dot product $\langle g_k^t, g_v^t \rangle$ for each accepted gradient g_k^t is guaranteed to be lower bounded by $\frac{V}{V'} \langle g_m, g_v^1 \rangle$. We have:

$$
\begin{aligned}
\langle g_v^t, g^t \rangle &= \frac{1}{k} \sum_{j=1}^{k} \langle g_v^t, g_j^t \rangle \\
&\geq \frac{1}{k} \sum_{j=1}^{k} \frac{V}{V'} \langle g_m, g_v^1 \rangle \\
&= \frac{V}{V'} \langle g_m, g_v^1 \rangle
\end{aligned} \tag{24}
$$

$$\langle g_v^t, -\eta g^t \rangle \leq -\eta \frac{V}{V'} \langle g_m, g_v^1 \rangle \tag{25}$$

$$\langle \nabla F(\theta^{t-1}), -\eta g^t \rangle \leq -\eta \left[\frac{V}{V'} \langle g_m, g_v^1 \rangle - \left(L^2 \eta^2 + \frac{1}{2} \right) CV - d\sigma^2 \right] \tag{26}$$

Plugging this back to (15), we get:

$$\begin{aligned} F(\theta^t) \leq F(\theta^{t-1}) - \eta \frac{V}{V'} \langle g_m, g_v^1 \rangle + \\ \left(\frac{L\eta^2}{2} + L^2 \eta^3 + \frac{1}{2} \eta \right) CV + \eta d \sigma^2 \end{aligned} \tag{27}$$

$$F(\theta^t) - F(\theta^{t-1}) \leq -\eta \frac{V}{V'} \langle g_m, g_v^1 \rangle + \mathbb{O}(V + d\sigma^2) \tag{28}$$

Since $(g_m)_j \in \left[\min_{\text{correct } i} (g_i^1)_j, \max_{\text{correct } i} (g_i^1)_j \right]$ and gradients are $i.i.d$, we have:

$$\begin{aligned} \langle g_m, g_v^1 \rangle &= \sum_{j=1}^{d} (g_m)_j (g_v^1)_j \\ &\geq \sum_{j=1}^{d} (g_i^1)_j (g_v^1)_j \text{ for some correct } i \text{ on each dimension} \end{aligned} \tag{29}$$

$$\begin{aligned} \mathbb{E} \langle g_m, g_v^1 \rangle &\geq \sum_{j=1}^{d} \mathbb{E}[(g_i^1)_j (g_v^1)_j] \text{ for some correct } i \text{ on each dimension} \\ &= \sum_{j=1}^{d} (\nabla F(\theta^1)_j)^2 \\ &= \|\nabla F(\theta^1)\|^2 \end{aligned} \tag{30}$$

By telescoping and taking the expectation of (28) and using the lower bound in (30), after T iterations we have:

$$\mathbb{E} \left[F(\theta^*) - F(\theta^0) \right] \leq \sum_{t=0}^{T} -\eta \frac{V}{V'} \|\nabla F(\theta^1)\|^2 + \mathbb{O}(V + d\sigma^2) \tag{31}$$

A.3 Full algorithm

Data: initial parameters θ^0, dataset $\{X, y\}$, number of workers N, initial k value k_0, increase k threshold τ_k, batch size n_b, Byzantine ratio f, fixed step size η, number of iterations T, loss function F
Result: final parameters θ^T
Initialization:
$k \leftarrow k_0$, $S \leftarrow$ squared Euclidean distance for first iteration median g_i, $D \leftarrow$ dot product for first iteration median g_i;
Create worker subsets and validation set $X_i, V \neq X_i$ for each worker $i \in [1, N]$;
for $t \leftarrow 1$ **to** T **do**
  Step 1: Broadcast current estimate
  The server broadcasts the current estimate θ^t to all workers;
  Each worker i:
  Receives θ^t from the server;
  if *worker i is honest* **then**
    Draws random batch $x \sim X_i$, computes $g_i = \frac{1}{n_b} \sum_{x_i \in x} \nabla F(\theta^t; x_i)$;
  else
    Computes spurious gradient g_i as per Byzantine strategy;
  end
  Sends g_i back to the server;
  Step 2: *dSTAR* Aggregation
  The server waits for gradients to be returned;
  for *each returned gradient g_i* **do**
    Compute normalized Euclidean distance s_i and dot product d_i with respect to the validation set gradient g_V;
    $s_i \leftarrow \frac{\|g_i - g_v\|^2}{\|g_v\|}$;
    $d_i \leftarrow \langle \frac{g_i}{\|g_v\|}, \frac{g_v}{\|g_v\|} \rangle$;
    if $s_i \leq S$ ***and*** $d_i \geq D$ **then**
      Append g_i to the accepted list;
    end
    if *k gradients have been appended* **then**
      break;
    end
  end
  $g_{\text{agg}} = \frac{1}{k} \sum_{j=1}^{k} g_{\text{accepted}_j}$;
  Update model parameters $\theta^{t+1} \leftarrow \theta^t - \eta g_{\text{agg}}$;
  $t \leftarrow t + 1$;
end

Algorithm 1: Byzantine-Resilient Gradient Aggregation

A.4 Training curves for experiments

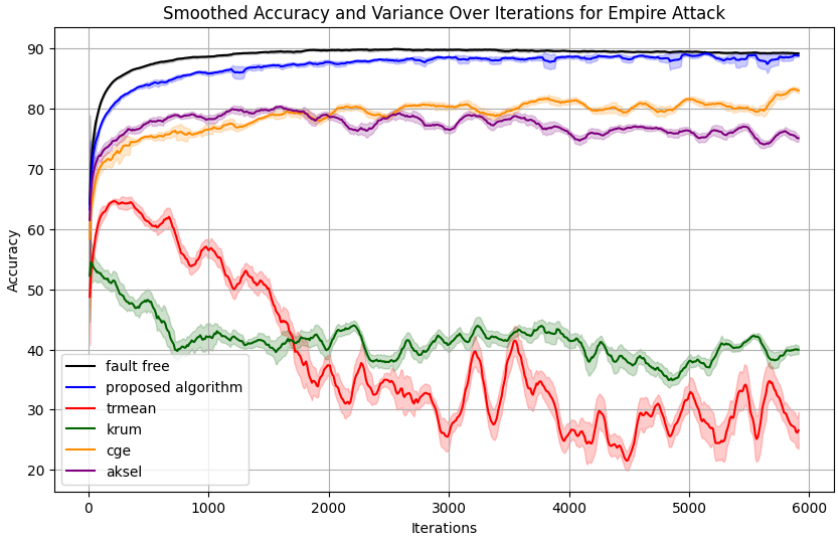

Figure 1: Fashion-MNIST with “Empire" attack

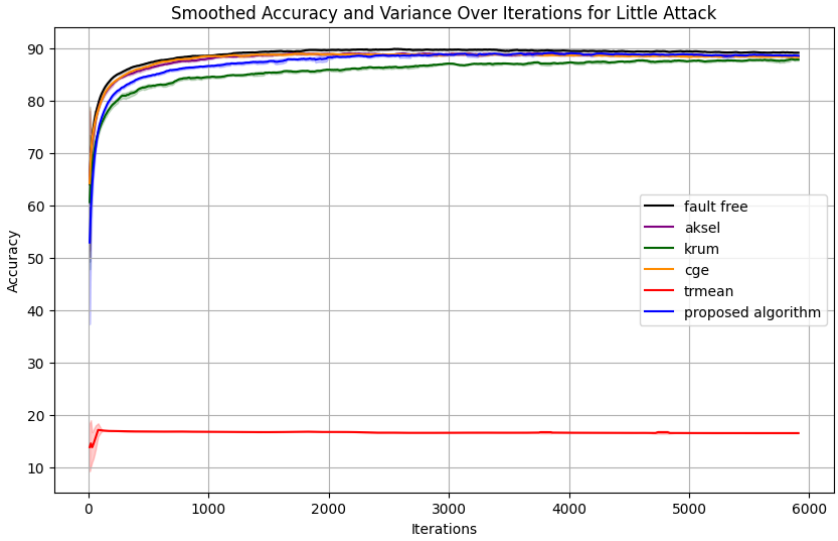

Figure 2: Fashion-MNIST with “Little" attack

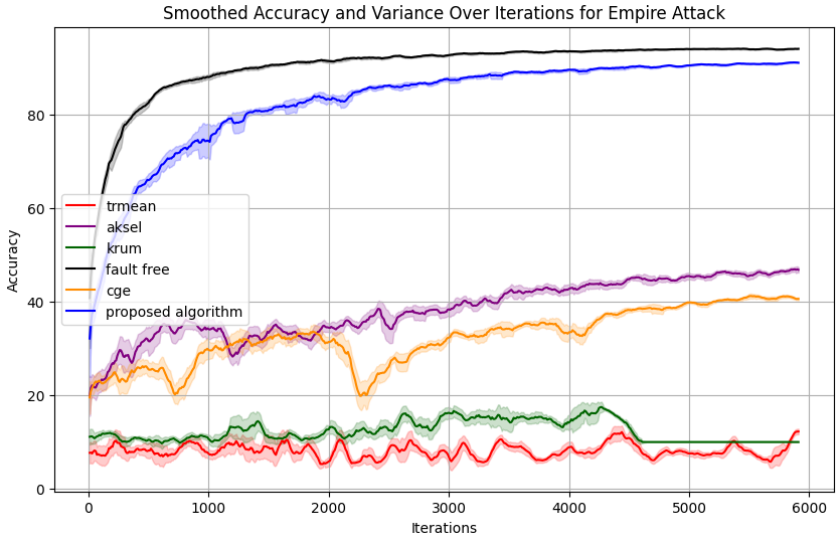

Figure 3: CIFAR10 with “Empire" attack

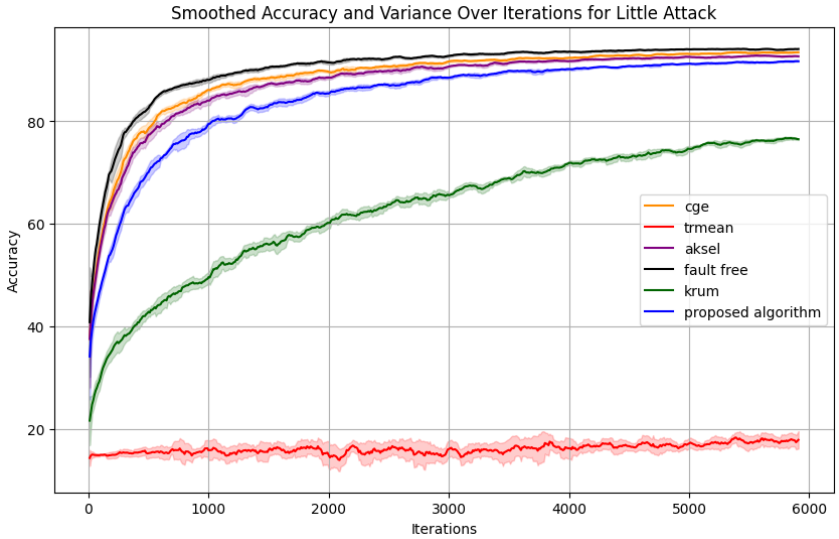

Figure 4: CIFAR10 with “Little" attack