Supplementary Material for NeurIPS 2023 Submission $\#9335$ Flag Aggregator: Distributed Training under Failures and Augmented Losses using Convex Optimization

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Modern ML applications increasingly rely on complex deep learning models and large datasets. There has been an exponential growth in the amount of computation needed to train the largest models. Therefore, to scale computation and data, these models are inevitably trained in a distributed manner in clusters of nodes, and their updates are aggregated before being applied to the model. However, a distributed setup is prone to Byzantine failures of individual nodes, components, and software. With data augmentation added to these settings, there is a critical need for robust and efficient aggregation systems. We define the quality of workers as reconstruction ratios $\in (0, 1]$, and formulate aggregation as a Maximum Likelihood Estimation procedure using Beta densities. We show that the Regularized form of log-likelihood wrt subspace can be approximately solved using iterative least squares solver, and provide convergence guarantees using recent Convex Optimization landscape results. Our empirical findings demonstrate that our approach significantly enhances the robustness of state-of-the-art Byzantine resilient aggregators. We evaluate our method in a distributed setup with a parameter server, and show simultaneous improvements in communication efficiency and accuracy across various tasks.

A Background and Related Work

Researchers have approached the Byzantine resilience problem from two main directions. In the first class of works, techniques such as geometric median and majority voting try to perform robust aggregation [1], [2], [3]. The other class of works uses redundancy and assigns each worker redundant gradient computation tasks [4], [5].

From another aspect, robustness can be provided on two levels. In weak Byzantine resilience methods such as Coordinate-wise median [6] and Krum [3], the learning is guaranteed to converge. In strong Byzantine resilience, the learning converges to a state as the system would converge in case no Byzantine worker existed. Draco [5] and Bulyan [7] are examples of this class. Convergence analysis of iterated reweighing type algorithms has been done for specific problem classes. For example, [8, 9] show that when IRLS is applied for sparse regression tasks, the iterates can converge linearly. Convergence analysis of matrix factorization problems using IRLS-type schemes has been proposed before, see [10, 11].

It is well known that data augmentation techniques help in improving the generalization capabilities of models by adding more identically distributed samples to the data pool. [12, 13, 14]. The techniques have evolved along with the development of the models, progressing from the basic ones like rotation, translation, cropping, flipping, injecting Gaussian noise [15] etc., to now the sophisticated ones (random erasing/masking [16], cutout [17] etc.). Multi-modal learning setups [18, 19, 20], use different ways to combine data of different modalities (text, images, audio etc.) to train deep learning networks.

B Tractability of Computing Flag Aggregators

In this section, we characterize the computational complexity of solving the Flag Aggregation problem via IRLS type schemes using results from convex optimization. First, we present a tight convex relaxation of the Flag Median problem by considering it as an instantiation of rank constrained optimization problem. We then show that we can represent our convex relaxation as a Second Order Cone Program which can be solved using off-the-shelf solvers [21]. Second, we argue that approximately solving such rank constrained problems in the factored space is an effective strategy using new results from [22] which builds on asymptotic convergence in [10]. Our results highlight that the Flag Median problem can be approximately solved using smooth optimization techniques, thus explaining the practical success of an IRLS type iterative solver.

Interpreting Flag Aggregator (equation 5 in the main paper) in the Case $m = 1$. We first present a convex reformulation of the Flag Aggregator problem (5) in the case when the number of subspaces (or columns) is equal to 1. To make the exposition easier, we will also assume that $\lambda = 0$. With these assumptions, and using the fact that $\|y\|_2 = 1$, each term in the objective function of our FA aggregator in (5) can be rewritten as,

$$\sqrt{(1-(y^T\tilde{g}_i))^2)} = \sqrt{y^T\left(I-\tilde{g}_i\tilde{g}_i{}^T\right)y} = \|\tilde{B}_i y\|_2, \tag{1}$$

where we use the notation $\tilde{g}_i = g_i/\|g_i\|$ to denote the normalized worker gradients, $I \in \mathbb{R}^{n\times n}$ is the $n \times n$ identity matrix, and $\tilde{B}_i$ is the square root of the matrix $I - \tilde{g}_i\tilde{g}_i{}^T$. Observe that we can rewrite all the terms in equation (5) in the main paper in a similar fashion as in (1). Furthermore, by relaxing the feasible set to the $n-$Ball given by $\{y \in \mathbb{R}^n : \|y\|_2 \leq 1\}$, we obtain a Second Order Cone Programming (SOCP) relaxation of our FA problem in (5). SOCP problems can be solved using off-the-shelf packages with open source optimization solvers for gradient aggregation purposes in small scale settings, that is, when the number of parameters $n \approx 10^4$ [23, 24, 25]. Our convex reformulation immediately yields insights on why reweighing type algorithm that was proposed in [26] works well in practice – for example see Section 3 in [27] in which various smoothing functions similar to the Flag Median (square) based smoothing are listed as options. More generally, our SOCP relaxation shows that if the smoothed version can be solved in closed form (or efficiently), then a reweighing based algorithm can be safely considered a viable candidate for aggregation purposes.

Tractable Reformulations when $m > 1$ for Aggregation Purposes. Note that for any feasible Y such that $Y^TY = I$, we have that the tr$(Y) = m$.

Remark B.1 (Parametrizing Subspaces using Y.). This assumption is without loss of generality. To see this, first note that in general, a (nondegenerate) subspace $\mathcal{S}$ of a vector space $\mathcal{V}$ is defined as a subset of $\mathcal{V}$ that is closed under linear combinations. Fortunately, in finite dimensions, we can represent $\mathcal{S}$ as a rectangular matrix M by Fundamental theorem of linear algebra. So, we simply use Y to represent the basis of this matrix M that represents the subspace $\mathcal{S}$ in our FA formulation.

Using this, we can rewrite each term in the objective function of our FA aggregator in equation (5) in the main paper as,

$$\sqrt{\text{tr}\left(Y^T\left(\frac{I}{m}-\frac{g_ig_i^T}{\|g_i\|_2^2}\right)Y\right)} = \sqrt{\text{tr}\left(Y^TM_iY\right)}, \tag{2}$$

where $M_i = M_i^T, i = 1, ..., p$ is symmetric matrix with *at most* one negative eigenvalue. Optimization problems involving quadratic functions with negative eigenvalues can be solved globally, in some cases [28, 29]. We consider methods that can efficiently (say in polynomial running time in n) provide solutions that are locally optimal. In order to do so, we consider the Semi Definite programming relaxation obtained by introducing matrix $Z \succeq 0 \in \mathbb{R}^{nm\times nm}$ to represent the term YY^T, constrained to be rank one, and such that tr$(Z) = m$.

By using $\mathbf{vec}(Y) \in \mathbb{R}^{nm}$ to denote the vector obtained by stacking columns of Z, when $m > 1$, we obtain a trace norm constrained SOCP. Importantly, objective function can be written as a sum of terms of the form,

$$\sqrt{\mathbf{vec}(Y)^T \left(I \otimes M_i\right) \mathbf{vec}(Y)} = \sqrt{\operatorname{tr}\left(Z^T \left(I \otimes M_i\right)\right)}, \tag{3}$$

where $\otimes$ denotes the usual tensor (or Kronecker) product between matrices.

Properties of lifted formulation in (3). There are some advantages to the loss function as specified in our reformulation (3). First, note that then our relaxation coincides with the usual trace norm based Goemans-Williamson relaxation used for solving Max-Cut problem with approximation guarantees [30]. Albeit, our objective function is not linear, and to our knowledge, it is not trivial to extend the results to nonlinear cases such as ours in (3). Moreover, even when $M_i \succeq 0$, the $\sqrt{\cdot}$ makes the relaxation *nonconvex*, so it is not possible to use off-the-shelf disciplined convex programming software packages such as CVXPy [31, 32]. Our key observation is that away from 0, $\sqrt{\cdot}$ is a *differentiable* function. Hence, the objective function in (3) is differentiable with respect to Z.

Remark B.2 (Using SDP relaxation for Aggregation.). In essence, if the optimal solution Z^* to the SDP relaxation is a rank one matrix, then by rank factorization theorem, Z^* can be written as $Z^* = \mathbf{vec}(Y^*)\mathbf{vec}(Y^*)^T$ where $\mathbf{vec}(Y^*) \in \mathbb{R}^{mn\times 1}$. So, after reshaping, we can obtain our optimal subspace estimate $Y^* \in \mathbb{R}^{m\times n}$ for aggregation purposes. In the case the optimal Z^* is not low rank, we simply use the largest rank one component of Z^*, and reshape it to get Y^*.

C Solving Flag Aggregation Efficiently

Convergence Analysis when $m = 1$. Note that for the case $m = 1$, that is, FA provides unit vector $y \in \mathbb{R}^n$ to get aggregated gradient as $yy^T G$, we can use smoothness based convergence results in nonconvex optimization, for example, please see [33]. We believe this addresses most of the standard training pipelines used in practice. Now, we focus on the case with $m > 1$.

Now that we have a smooth reformulation of the aggregation problem that we would like to solve, it is tempting to solve it using first order methods. However, naively applying first order methods can lead to slow convergence, especially since the number of decision variables is now increased to m^2n^2. Standard projection oracles for trace norm require us to compute the full Singular Value Decomposition (SVD) of Z which becomes computationally expensive even for small values of $m, n \approx 10$.

Fortunately, recent results show that the factored form smooth SDPs can be solved in polynomial time using gradient based methods. That is, by setting $Z = \mathbf{vec}(Y)\mathbf{vec}(Y)^T$, and minimizing the loss functions $L_i(Y) = \sqrt{\mathbf{vec}(Y)^T \left(I \otimes M_i\right) \mathbf{vec}(Y)}$ with respect to Y, we have that the set of locally optimal points coincide, see [22]. Moreover, we have the following convergence result for first order methods like Gradient Descent that require SVD of $n \times p$ matrices:

Lemma C.1. *If L_i are κ_i-smooth, with a η_i-lipschitz Hessian, then projected gradient descent with constant step size converges to a locally optimal solution to (3) in $\tilde{O}(\kappa/\epsilon^2)$ iterations where $0 \leq \epsilon \leq \kappa^2/\eta$ is a error tolerance parameter, $\kappa = \max_i \kappa_i$, and $\eta = \max_i \eta_i$.*

Above lemma C.1 says that gradient descent will output an aggregation Y that satisfies second order sufficiency conditions with respect to smooth reformulated loss function in (3). All the terms inside $\tilde{O}$ in lemma C.1 are logarithmic in dimensions m, n, lipschitz constant L, and accuracy parameter ϵ.

Remark C.2 (Numerical Considerations.). Note that the lipschitz constant κ of the overall objective function depends on M_i. That is, when M_i has negative eigenvalues, then κ can be high due to the square root function. We can consider three related ways to avoid this issue. First, we can choose a value $m' > m$ in our trace constraint such that $M_i \succeq 0$. Similarly, we can expand (3) (in $\sqrt{\cdot}$) as outer product of columns of Y suggesting that $\tilde{g}\tilde{g}^T$ term need to be normalized by m, thus making $M_i \succeq 0$. Secondly, we can consider adding a quadratic term such as $\|Y\|_{\text{Fro}}^2$ to make the function quadratic. This has the effect of decreasing κ and η of the objective function for optimization. Finally, we can use $m_i = \max(k_i, m)$ instead of $\min$ in defining the loss function as in [26] which would also make $M_i \succeq 0$.

D Proof of Lemma C.1 when $m > 1$.

We provide the missing details in Section C when $m > 1$. To that end, we will assume that each worker i provides the server with a list of k_i gradients, that is, $g_i \in \mathbb{R}^{n\times k}$ – a strict generalization of the case considered in the main paper (with $k = 1$), that may be useful independently. Note that in [26], these g_i's are assumed to be subspaces whereas we do not make that assumption in our FA algorithm.

Now, we will show that the RHS in equation (2) and LHS in equation (3) are equivalent. For that, we need to recall an elementary linear algebra fact relating tensor/Kronecker product, and tr operator. Recall the definition of Kronecker product:

Definition D.1. Let $A \in \mathbb{R}^{d_1\times d_2}$, $B \in \mathbb{R}^{e_1\times e_2}$, then $A \otimes B \in \mathbb{R}^{d_1 e_1 \times d_2 e_2}$ is given by,

$$A \otimes B := \begin{bmatrix} a_{1,1}B & \dots & a_{1,d_2}B \\ \vdots & \ddots & \vdots \\ a_{d_1,1}B & \dots & a_{d_1,d_2}B \end{bmatrix}, \tag{4}$$

where $a_{i,j}$ denotes the entry at the $i-$th row, $j-$th column of A.

Lemma D.2 (Equivalence of Objective Functions.). *Let $Y \in \mathbb{R}^{n\times m}$, $g \in \mathbb{R}^{n\times k}$ (so, $M \in \mathbb{R}^{n\times n}$). Then, we have that,*

$$\mathit{tr}\left(Y^T g g^T Y\right) := \mathit{tr}\left(Y^T M Y\right) = \mathbf{vec}(Y)^T \left(I \otimes M\right) \mathbf{vec}(Y), \tag{5}$$

where $I \in \mathbb{R}^{m\times m}$ is the identity matrix.

Proof. Using the definition of tensor product in equation (4), we can simplify the right hand side of equation (5) as,

$$\begin{aligned}\mathbf{vec}(Y)^T \left(I \otimes M\right) \mathbf{vec}(Y) &= [y_{11}, \cdots y_{n1}, \cdots, y_{1m}, \cdots, y_{nm}] \begin{bmatrix} M & 0 & \dots & 0 \\ \vdots & M & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & M \end{bmatrix} \begin{bmatrix} y_{11} \\ \vdots \\ y_{n1} \\ \vdots \\ y_{1m} \\ \vdots \\ y_{mn} \end{bmatrix} \\ &= \sum_{j=1}^{m} y_j^T M y_j \\ &= \sum_{j=1}^{m} \text{tr}\left(y_j y_j^T M\right) = \text{tr}\left(\sum_{j=1}^{m} y_j y_j^T M\right) = \text{tr}\left(\left(\sum_{j=1}^{m} y_j y_j^T\right) M\right) \qquad (6) \\ &= \text{tr}\left(YY^T M\right) = \text{tr}\left(Y^T M Y\right), \qquad (7)\end{aligned}$$

where we used the cyclic property of trace operator tr$(\cdot)$ in equations (6), and (7) that is, tr$(ABC) =$ tr$(CAB) =$ tr(BCA) for any dimension compatible matrices A, B, C. □

D.1 Proof of Lemma C.1

Recall that, given $\tilde{M}_i = I \otimes M_i$, the lifted cone programming relaxation of FA can be written as,

$$\min_Z \sum_i \sqrt{\text{tr}(Z^T \tilde{M}_i)} \quad \text{s.t.} \quad Z \succeq 0, \;\; \text{tr}(Z) = m, Z = Z^T, \tag{8}$$

where m is the rank of Z or number of columns of Y. We now use the above Lemma D.2 to show that the objective function with respect to Z in the lifted formulation is smooth which gives us the desired convergence result in Lemma C.1.

Proof. let $\tilde{\kappa}_i > 0$,

$$\frac{\partial \sqrt{\text{tr}(Z^T\tilde{M}_i)+\tilde{\kappa}_i}}{\partial Z} = \frac{1}{2\sqrt{\tilde{\kappa}_i + \text{tr}(Z^T\tilde{M}_i)}}\tilde{M}_i, \tag{9}$$

where $\tilde{M}_i = I \otimes M_i$ as in equation (3). Now, since $\tilde{M}_i$ is constant with respect to Z, the gradient term is affected only through a scalar $\sqrt{\text{tr}(Z^T\tilde{M}_i)+\tilde{\kappa}_i}$. So the largest magnitude or entrywise ℓ_∞-norm of the Hessian is given by,

$$\left|\frac{\partial \frac{1}{\sqrt{\text{tr}(Z^T\tilde{M}_i)+\tilde{\kappa}_i}}\|\tilde{M}_i\|_\infty}{\partial Z}\right| = \frac{\|\tilde{M}_i\|_\infty}{2\sqrt{\left(\text{tr}(Z^T\tilde{M}_i)+\tilde{\kappa}_i\right)^3}}. \tag{10}$$

Now, we will argue that the gradient and hessian are lipschitz continuous in the lifted space. Since any feasible $Z \succeq 0$ is positive semidefinite, if $\tilde{M}_i \succeq 0$, then the scalar $\text{tr}(Z^T\tilde{M}_i)$ is at least $m \cdot \lambda_{mn}^{\tilde{M}_i}$ where $\lambda_{mn}^{\tilde{M}_i}$ is the smallest (or $mn-$th) eigenvalue of $\tilde{M}_i$. So, we can choose $\tilde{\kappa}_i = 0 \forall i$. If not, then there is a negative eigenvalue, possibly repeated. So, the gradient might not exist. In cases where $\tilde{M}_i$ has negative eigenvalues, we can choose $\tilde{\kappa}_i = \tilde{\kappa} = \left|\min_i \min\left(\lambda_{mn}^{\tilde{M}_i}, 0\right)\right|$. With these choices, we have that the gradient of the objective function in (3) is lipschitz continuous. By a similar analysis using the third derivative, we can show that Hessian is also lipschitz continuous with respect to Z. In other words, all the lipschitz constant of both the gradient and hessian of our overall objective function is controlled by $\tilde{\kappa} > 0$. Hence, we have all conditions satisfied required for Lemma 1 in [22], and we have our convergence result for FA in the factored space of **vec**(Y). □

Few remarks are in order with respect to our convergence result. First, **is the choice $\tilde{\kappa}$ important for convergence?** Our convergence result shows that a perturbed objective function $\text{tr}(Z^T\tilde{M}_i)+\tilde{\kappa}$ has the same second order stationary points as that of the objective function in the factored form formulated using Y (or **vec**(Y)). We can avoid this perturbation argument if we explicitly add constraints $\text{tr}(Z^T\tilde{M}_i) \geq 0$, since projections on linear constraints can be performed efficiently exactly (sometimes) or approximately. Note that these constraints are natural since it is not possible to evaluate the square root of a negative number. Alternatively, we can use a smooth approximate approximation of the absolute values $\sqrt{\left|\text{tr}(Z^T\tilde{M}_i)\right|}$. In this case, it is easy to see from (9), and (10) that the constants governing the lipschitz continuity as dependent on the absolute values of the minimum eigenvalues, as expected. In essence, no, the choice of $\tilde{\kappa}$ does not affect the nature of landscape – approximate locally optimal points remain approximately locally optimal. In practice, we expect the choice of $\tilde{\kappa}$ to affect the performance of first order methods.

Second, **can we assume $\tilde{M}_i \succ 0$ for gradient aggregation purposes?** Yes, this is because, when using first order methods to obtain locally optimal solution, the scale or norm of the gradient becomes a secondary factor in terms of convergence. So, we can safely normalize each M_i by the nuclear norm $\|M_i\|_* := \sum_{j=1}^{k_i}\sigma_j$ where σ_j is the $j-$th singular value of M_i. This ensures that $I - M_i \succeq 0$, assistant convergence. While $\|M_i\|_*$ itself might be computationally expensive to compute, we may be able to use estimates of $\|M_i\|_*$ via simple procedures as in [34]. In most practical implementations including ours, we simply compute the average of the gradients computed by each worker before sending it to the parameter server, that is, $k_i \equiv k = 1$ in which case simply normalizing by the euclidean norm is sufficient for our convergence result to hold. Our FA based distributed training Algorithm 1 solves the factored form for gradient aggregation purposes (in Step 6) at the parameter server.

Finally, please note that our technical assumptions are standard in optimization literature, that exploits smoothness of the objective function – since the feasible set of Y in (1) is bounded, assumptions are satisfied. Our proof techniques are standard, and we simply use them on our reformulation to obtain convergence guarantee ***second order stationary points*** for IRLS iterations since there exists a tractable SDP relaxation.

D.2 FA Optimality Conditions and Similarities with Bulyan [7] Baseline.

We first restate our Flag Aggregator with $g_i \in \mathbb{R}^{n\times k}$ in optimization terms as follows,

$$\min_{Y:Y^TY=I} A(Y) := \sum_{i=1}^{p} \sqrt{\left(1 - \frac{\operatorname{tr}\left(Y^T g_i g_i^T Y\right)}{\operatorname{tr}\left(g_i^T g_i\right)}\right)} + \lambda \mathcal{R}(Y), \tag{11}$$

and write its associated Lagrangian $\mathcal{L}$ defined by,

$$\mathcal{L}(Y,\Gamma) := \sum_{i=1}^{p} \sqrt{\left(1 - \frac{\operatorname{tr}\left(Y^T g_i g_i^T Y\right)}{\operatorname{tr}\left(g_i^T g_i\right)}\right)} + \lambda \mathcal{R}(Y) + \operatorname{tr}\left(\Gamma^T\left(Y^TY - I\right)\right), \tag{12}$$

where $\Gamma \in \mathbb{R}^{m\times m}$ denotes the Lagrange multipliers associated with the orthogonality constraints in equation (11). In particular, since the constraints we have are equality, there are no sign restrictions on Γ, so they are often referred to as "free". Moreover, since Y is a real matrix, the constraints are symmetric (i.e., $y_i^T y_j = y_j^T y_i$), we may assume that $\Gamma = \Gamma^T$, without loss of generality.

We will introduce some notations to make calculations easier. We will use $\tilde{g}_i \in \mathbb{R}^{n\times k}$ to denote the normalized gradients matrix of the data terms in equation (11). That is, we define

$$\tilde{g}_i := -\frac{1}{\operatorname{tr}\left(g_i^T g_i\right) \cdot \sqrt{\left(1 - \frac{\operatorname{tr}\left(Y^T g_i g_i^T Y\right)}{\operatorname{tr}\left(g_i^T g_i\right)}\right)}} g_i g_i^T =: d_i g_i g_i^T. \tag{13}$$

With this notation, we are ready to use the first optimality conditions associated with the constrained optimization problem in (11) with its Lagrangian in (12) By first order optimality or KKT conditions, we have that,

$$\begin{aligned} 0 = \nabla_Y \mathcal{L}(Y_*, \Gamma_*) &= \left(\sum_{i=1}^{p} \tilde{g}_i \tilde{g}_i^T\right) Y_* + \lambda \nabla \mathcal{R}(Y_*) + 2Y_*\Gamma_* \\ &= GD_*G^TY_* + \lambda \nabla \mathcal{R}(Y_*) + 2Y_*\Gamma_*, \quad \text{(Objective)} \\ 0 = \nabla_\Gamma \mathcal{L}(Y_*, \Gamma_*) &= Y_*^T Y_* - I, \quad \text{(Feasibility)} \end{aligned} \tag{14}$$

where $Y_* \in \mathbb{R}^{n\times m}, \Gamma_* \in \mathbb{R}^{m\times m}$ are the optimal primal parameters, lagrangian multipliers, and $D_* \in \mathbb{R}^{p\times p}_{<0}$ is the diagonal matrix with entries equal to $-d_i < 0$ as in equation (13). We may ignore the Feasibility conditions since our algorithm returns an orthogonal matrix by design, and focus on the Objective conditions. Now, by bringing the term associated with Lagrangian to the other side, and then right multiplying by Γ_*^{-1} inverse of Γ_*, we have that Y_* satisfies the following identity,

$$Y_* = -\frac{1}{2}\left(GD_*G^TY_* + \lambda\nabla\mathcal{R}(Y_*)\right)\Gamma_*^{-1}. \tag{15}$$

By using the identity (15), we can write an equivalent representation of our aggregation rule $Y_*Y_*^TG$ given by,

$$\begin{aligned} Y_*Y_*^TG &= \frac{1}{4}\left(GD_*G^TY_* + \lambda\nabla\mathcal{R}(Y_*)\right)\underbrace{\Gamma_*^{-1}\Gamma_*^{-1}\left(Y_*^TGD_*G^T + \lambda\nabla\mathcal{R}(Y_*)^T\right)G}_{:=\mathfrak{M}_*\in\mathbb{R}^{m\times p}} \\ &\propto \left(GD_*G^TY_* + \lambda\nabla\mathcal{R}(Y_*)\right)\mathfrak{M}_* \\ &= G\underbrace{D_*G^TY_*}_{:=S'_*\in\mathbb{R}^{p\times m}}\mathfrak{M}_* + \lambda\nabla\mathcal{R}(Y_*)\mathfrak{M}_* \\ &= GS'_*\mathfrak{M}_* + \lambda\nabla\mathcal{R}(Y_*)\mathfrak{M}_* \\ :&= GS_{\text{FA}} + \lambda\nabla\mathcal{R}(Y_*)\mathfrak{M}_*, \end{aligned} \tag{16}$$

that is, the update rule of FA can be seen as a left multiplication with the square "flag selection" matrix $S_{\text{SA}} = S'_*\mathfrak{M}_* \in \mathbb{R}^{p\times p}$, and then perturbing with the gradient $\nabla\mathcal{R}(Y_*)$ of the regularization function $\mathcal{R}$ with a different matrix $\mathfrak{M}_*$ as in equation (16). Importantly, we can see in equation

(16) that the (reduced) selection matrix $S \in \mathbb{R}^{p\times m}_{\geq 0}$ in Bulyan [7] is equivalent to the total selection matrix $S_{\text{SA}} \in \mathbb{R}^{p\times p}$ in our FA setup. Moreover, we can also see that domain knowledge in terms of regularization function may also determine the optimal subspace, albeit additively only. We leave the algorithmic implications of our result as future work.

Remark D.3 (Invertibility of Γ_* in Equation (15).). Theoretically, note that Γ is symmetric, so by Spectral Theorem, we know that its eigen decomposition exists. So, we may use pseudo-inverse instead of its inverse. Computationally, given any primal solution Y_* we can obtain Γ_* by left multiplying equation (14) by Y_* and use feasibility i.e., $Y_*^T Y_* = I$. Now, we obtain Γ_*^{-1} columnwise by using some numerical solver such as conjugate gradient (with fixed iterations) on Γ with standard basis vectors. In either case, our proof can be used with the preferred approximation choice of Γ_*^{-1} to get the equivalence as in equation (16).

Remark D.4 (Provable Robustness Guarantees for FA.). Since our FA scheme is based on convexity, it is possible to show worst-case robustness guarantees for FA iterations under mild technical conditions on Y^* – even under correlated noise, see for e.g. Assumption 1 in [35]). In fact, by using the selection matrix S_{FA} in equation (16) in Lemma 1 in [1] and following the proof, we can get similar provable robustness guarantees for FA. We leave the theoretical analysis as future work.

E ADDITIONAL EXPERIMENTS

E.1 The Effect of Regularization Parameter

Our algorithm depends on the regularization parameter λ. Figure 1 below illustrates the effect of this parameter on similarity of aggregated gradient vectors for FA and Multi-Krum. For this experiment, we sample the gradients output by the parameter server across multiple epochs for both FA and Multi-Krum and compute the cosine similarity of corresponding vectors. We repeat the experiment with different λ values. As we can see, for smaller iterations there is some similarity between the gradients computed by FA and Multi-Krum. This similarity is more visible for smaller λ values.

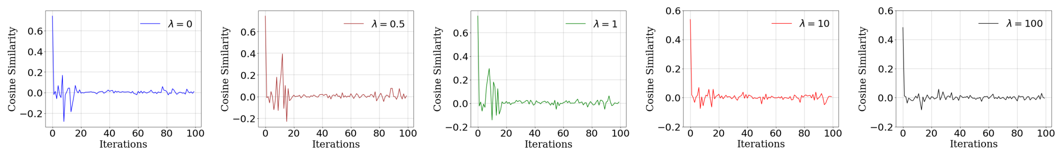


Figure 1: The effect of the regularization parameter λ on similarity of FA performance to Multi-Krum

E.2 Experiments with the Tiny ImageNet dataset

We repeated our experiments with Tiny ImageNet [36] which contains 100000 images of 200 classes (500 for each class) downsized to 64×64 colored images. We fix our batch size to 192 and use ResNet-50 [37] throughout the experiments

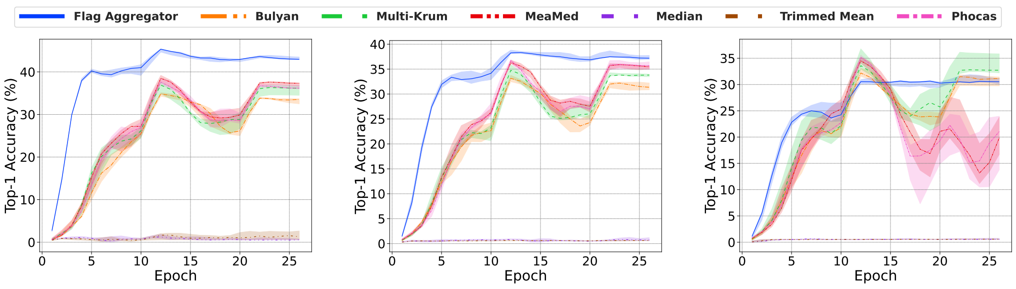
Tolerance to the number of Byzantine workers: In this experiment, we have $p = 15$ workers of which $f = 1, .., 3$ are Byzantine and send random gradients. The accuracy of test data for FA in comparison to other aggregators is shown in Figure 2. As we can see, for $f = 1$ and $f = 2$, FA converges at a higher accuracy than all other schemes. For all cases, FA also converges in $\sim$2x less number of iterations.

Tolerance to communication loss:

We set a 10% loss rate for the links connecting $f = 1, .., 3$ of the workers to the parameter server. Figure 3 shows that our takeaways in the main paper are also confirmed in this setting with the new dataset.

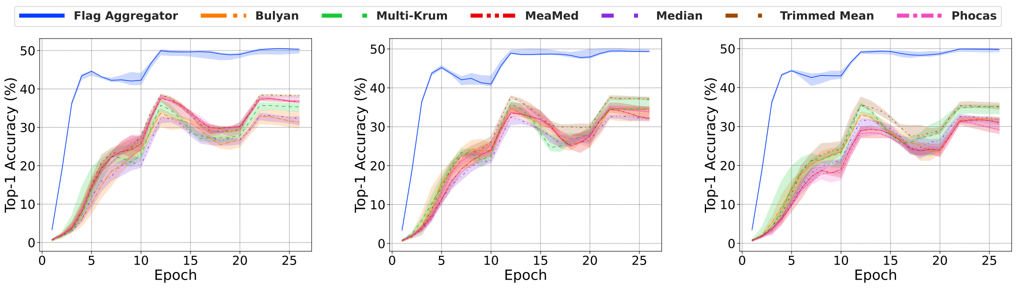
The effect of having augmented data during training in Byzantine workers: We choose two non linear augmentation schemes, Lotka Volterra (shown in rows 1 and 3 of Figure 4) and Arnold's Cat Map (shown in rows 2 and 4 of Figure 4).

As seen from the figure, Arnold's Cat Map augmentations stretch the images and rearrange them within a unit square, thus resulting in streaky patterns. Whereas the Lotka Volterra augmentations



(a) $f = 1$ (b) $f = 2$ (c) $f = 3$

Figure 2: Tolerance to the number of Byzantine workers for robust aggregators.



(a) $f = 1$ (b) $f = 2$ (c) $f = 3$

Figure 3: Tolerance to communication loss

distort the images while keeping the images similar to the original ones. We perform experiments with data augmented with varying shares using the two methods and show the results in Figure 5. For CIFAR-10, we showed the results when all of the samples in Byzantine workers are augmented in Figure 7 in the main paper. For Tiny ImageNet, this case is shown Figure 5a. Figures 5b and 5c show the results under different ratios on CIFAR-10. By changing the ratios we were interested to see if streaky patterns augmented by Arnold's Cat Map would introduce a more adverse effect from Byzantine workers compared to Lotka Volterra. Although the results do not show a significant signal, we can see that the augmentations did impact the overall gradients, and that FA performs significantly better.

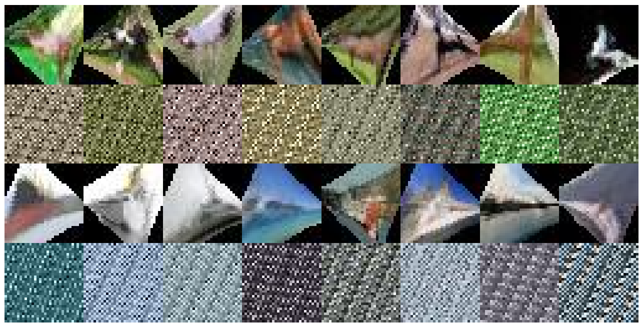
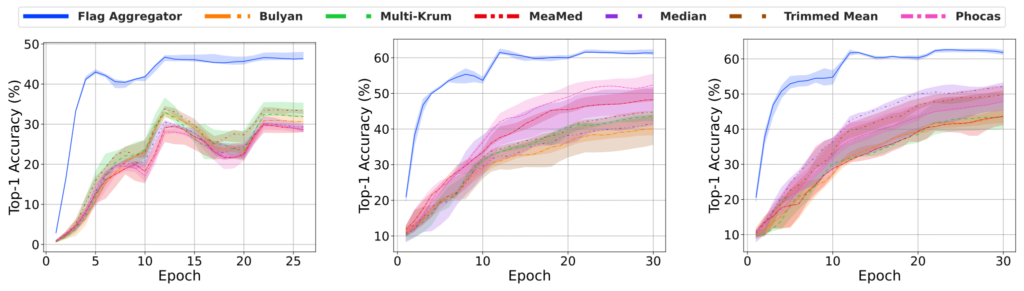


Figure 4: TinyImagenet data with Augmentation: **Row 1**: Lotka Volterra augmentation on Class Horse. **Row 2**: Arnold's Cat Map augmentation on Class Horse. **Row 3**:Lotka Volterra augmentation on Class Ship. **Row 4**: Arnold's Cat Map augmentation on Class Ship.

We have attached our code for running the experiments and reproducing the results in the zip file.



(a) 100% Lotka-Volterra on Tiny ImageNet

(b) 50% Lotka-Volterra, 50% Arnold's Cat Map on CIFAR-10

(c) 33% Lotka-Volterra, 66% Arnold's Cat Map on CIFAR-10

Figure 5: Accuracy of using augmented data in $f = 3$ workers

References

[1] Georgios Damaskinos, El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, and Sébastien Rouault. Aggregathor: Byzantine machine learning via robust gradient aggregation. *Proceedings of Machine Learning and Systems*, 1:81–106, 2019.

[2] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 118–128. Curran Associates Inc., 2017. ISBN 9781510860964.

[4] Shashank Rajput, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Detox: A redundancy-based framework for faster and more robust gradient aggregation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[5] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. DRACO: Byzantine-resilient distributed training via redundant gradients. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 903–912. PMLR, 10–15 Jul 2018.

[6] 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, 10–15 Jul 2018.

[7] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3521–3530. PMLR, 10–15 Jul 2018.

[8] Christian Kümmerle, Claudio Mayrink Verdun, and Dominik Stöger. Iteratively reweighted least squares for basis pursuit with global linear convergence rate. *Advances in Neural Information Processing Systems*, 34:2873–2886, 2021.

[9] Deeksha Adil, Richard Peng, and Sushant Sachdeva. Fast, provably convergent irls algorithm for p-norm linear regression. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Massimo Fornasier, Holger Rauhut, and Rachel Ward. Low-rank matrix recovery via iteratively reweighted least squares minimization. *SIAM Journal on Optimization*, 21(4):1614–1640, 2011.

[11] Yuxin Chen, Yuejie Chi, Jianqing Fan, Cong Ma, and Yuling Yan. Noisy matrix completion: Understanding statistical guarantees for convex relaxation via nonconvex optimization. *SIAM journal on optimization*, 30(4):3098–3121, 2020.

[12] Fanny Yang, Zuowen Wang, and Christina Heinze-Deml. Invariance-inducing regularization using worst-case transformations suffices to boost accuracy and spatial robustness. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1d01bd2e16f57892f0954902899f0692-Paper.pdf.

[13] Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness, 2017. URL https://arxiv.org/abs/1710.11469.

[14] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[15] Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995. doi: 10.1162/neco.1995.7.1.108.

[16] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:13001–13008, Apr. 2020. doi: 10.1609/aaai.v34i07.7000. URL https://ojs.aaai.org/index.php/AAAI/article/view/7000.

[17] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[18] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *International Journal of Computer Vision*, 127:398–414, 2017.

[19] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *ECCV*, 2020.

[20] Yu Huang, Chenzhuang Du, Zihui Xue, Xuanyao Chen, Hang Zhao, and Longbo Huang. What makes multimodal learning better than single (provably). In *NeurIPS*, 2021.

[21] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1-3):193–228, 1998.

[22] Srinadh Bhojanapalli, Nicolas Boumal, Prateek Jain, and Praneeth Netrapalli. Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form. In *Conference On Learning Theory*, pages 3243–3270. PMLR, 2018.

[23] Chungen Shen, Yunlong Wang, Wenjuan Xue, and Lei-Hong Zhang. An accelerated active-set algorithm for a quadratic semidefinite program with general constraints. *Computational Optimization and Applications*, 78(1):1–42, 2021.

[24] Ariel Kleiner, Ali Rahimi, and Michael Jordan. Random conic pursuit for semidefinite programming. *Advances in Neural Information Processing Systems*, 23, 2010.

[25] Hans D Mittelmann. An independent benchmarking of sdp and socp solvers. *Mathematical Programming*, 95(2):407–430, 2003.

[26] Nathan Mankovich, Emily J King, Chris Peterson, and Michael Kirby. The flag median and flagirls. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10339–10347, 2022.

[27] Robert J Vanderbei and Hande Yurttan. Using loqo to solve second-order cone programming problems. *Constraints*, 1(2), 1998.

[28] Hezhi Luo, Xiaodi Bai, Gino Lim, and Jiming Peng. New global algorithms for quadratic programming with a few negative eigenvalues based on alternative direction method and convex relaxation. *Mathematical Programming Computation*, 11(1):119–171, 2019.

[29] A Shapiro and JD Botha. Dual algorithm for orthogonal procrustes rotations. *SIAM journal on matrix analysis and applications*, 9(3):378–383, 1988.

[30] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[31] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[32] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[33] Prateek Jain, Purushottam Kar, et al. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363, 2017.

[34] Liqun Qi. Some simple estimates for singular values of a matrix. *Linear algebra and its applications*, 56:105–119, 1984.

[35] Olga Klopp, Karim Lounici, and Alexandre B. Tsybakov. Robust matrix completion, 2016.

[36] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge, 2015.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.