


Replication / ML Reproducibility Challenge 2022

Reproducibility Report for “ApproxIFER: A Model-Agnostic Approach to Resilient and Robust Prediction Serving Systems”

Anonymous¹, ¹ Anonymous Institution

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
03 February 2023

Published
—

DOI
—

Reproducibility Summary

In this reproducibility report, we study the paper “ApproxIFER: A Model-Agnostic Approach to Resilient and Robust Prediction Serving Systems”. Cloud-assisted AI services allow incapable clients to outsource their computationally-demanding low-latency tasks. This work proposed a model-agnostic inference framework, named Approximate Coded Inference (ApproxIFER) capable of handling slow and erroneous workers.

Scope of Reproducibility – We test the authors’ claims that: ApproxIFER can handle a general number of stragglers and scales better with the number of input queries; and that it is robust against erroneous workers that return incorrect predictions.

Methodology – We adopted and modified the source code received from the authors. Using ResNet-18 and ResNet-34, we run the experiments for different number of stragglers and/or adversarial workers on well-known datasets MNIST and CIFAR-10.


Results – Our reproduction of results were similar to the authors’ and therefore verifies that ApproxIFER has straggler resiliency. We run experiments beyond the paper, and verify the claim that ApproxIFER is model-agnostic. Furthermore, we attempt to extend the paper by introducing Laplace noise to model predictions to make the scheme differentially private.

What was easy – The paper is well-written, and we were able to understand the entire process of the proposed approach fully. Moreover, the parameters applied in the experiments were clear and easy to follow.

What was difficult – The theory behind the proposed framework and algorithm is complicated and required us to understand results from different references. The authors reused the same code file to run experiments with different architectures and on different datasets. This meant we had to rewrite parts of the code to modify it for the relevant dataset and architecture.

Communication with original authors – We were in contact with the authors at the beginning of this study. Thankfully, the authors shared the code of the experiments. We haven’t contacted them thereafter.

Copyright © 2023 Anonymous, released under a Creative Commons Attribution 4.0 International license.

Correspondence should be addressed to .

The authors have declared that no competing interests exists.

Code is available at <https://github.com/nayel71/ml-reproducibility>.

1 Introduction

Powerful cloud-assisted AI services (Amazon 2021; Microsoft 2021; Google 2021) let incapable clients to outsource their computationally-demanding tasks. Even if a single prediction is computationally feasible, such commonly deployed tasks often require low latency. To ease the workload of computations, the idea of prediction serving systems has been proposed. The paper considers a paradigm where the model has been properly trained on workstations with superior computation resources. The goal for the original paper is to apply the trained model to make predictions for unseen, real-world tasks. While each prediction is relatively easy, the computing capacity required to make prediction on a large number of tasks in a small amount of time necessitates the use of multiple workers. Specifically, prediction serving systems distribute the computations of predictions for the input query from a single device to several workers. The workers hold the trained machine learning model and obtain the predictions based on the tasks from the device, then return the results back to the device.

Note that training period of the neural network architecture is not the bottleneck of this framework. Instead, this framework aims to address the following two challenges:

- **Stragglers: “Failed” or “slow” workers**
One cannot always expect all workers to successfully return the predictions. For instance, a worker might face computational failure which may result in the worker failing to make a prediction; or a network congestion might occur, so we might suffer packet losses that require re-transmission of the predictions. Such issues may cause delays leading to high latency. We say a prediction serving system is straggler-resilient if it can tackle stragglers.
- **Erroneous workers: Byzantine adversarial workers**
Erroneous workers in systems return incorrect computational results either unintentionally or adversarially, trying to cause falsified results for the entire tasks. We call a prediction serving system Byzantine-robust if it can tackle the incorrect predictions caused by erroneous workers.

This work [1] proposed a framework design for a straggler and Byzantine resilient prediction serving system, named Approximate Coded Inference (ApproxIFER). Previous works had proposed frameworks that relied upon replication and parity check models to mitigate stragglers, but such systems either accrue significant overheads or can only handle a very small number (often just 1) of stragglers. Coded computing approaches, are limited to polynomial computations and require a large number of workers that depends on the desired computation. Hence, such techniques cannot be directly applied in such prediction serving systems.

In [2], the authors proposed a learning-based approach parity model (ParM) to overcome the limitations of the traditional coding theoretic approaches. But as a result of assigning the same task to multiple workers, ParM requires extra offline training based on the given trained machine learning model for generating a parity model to provide straggler resiliency. ApproxIFER, however, is efficient in that all workers can just predict through the same trained model. Thus, ApproxIFER is model-agnostic since it can be applied on any model architecture.

Beyond the claims of the paper, we attempt to look at ApproxIFER framework through the lens of differential privacy. Due to advanced data fusion and analysis techniques, the private data of users are more vulnerable to attack and disclosure in the big data era. It has been demonstrated that model inversion attacks reveal considerable information about the training data [3, 4, 5, 6, 7]. Differential privacy is a measure for privacy that ensures that the removal or addition of a single item to a database does not substantially affect the outcome of queries. It follows that no risk is incurred by joining the database, providing a mathematically rigorous means of coping with the fact that distributional information may be disclosive. Intuitively, this is done using a randomized function

\mathcal{K} which is an algorithm applied by the worker when releasing their prediction. The mechanism \mathcal{K} typically achieves this by adding appropriately random noise to the true answer. In Section 4, we show that the Laplace mechanism for Private-ApproxIFER is ϵ -differential privacy preserving.

The remainder of this work is structured as follows. In Section 2, we explicitly identify the claims from the original paper that we wish to verify/reproduce through the course of this project. In Section 3, we introduce the approach, ApproxIFER, proposed in the original paper. Section 4 presents the replicated and new results and compares them to the original paper. Then we look at this paradigm from the perspective of privacy, and introduce a ϵ -differentially private scheme Private-ApproxIFER. Finally, Section 5 concludes this work by discussing our study of reproducing the paper.

2 Scope of reproducibility

The original paper proposed the ApproxIFER scheme, a scalable, straggler-resilient, and Byzantine-robust prediction serving system. The approach is heavily based on the rational interpolation technique in order to estimate the predictions even if there are stragglers and/or erroneous workers in the system. The claims in the paper we reproduce in this report are summarized as follows:

- Claim 1: ApproxIFER is a model-agnostic framework
- Claim 2: ApproxIFER is straggler-resilient
Note that ApproxIFER is model-agnostic and can be applied to any machine learning model, so the approach does not depend on the learning models. **maybe move to above?**
- Claim 3: ApproxIFER is Byzantine-robust
The paper proposed an interpolation algorithm to exclude erroneous predictions from Byzantine workers.

We further extend the paper and mathematically prove that the Laplace mechanism for Private-ApproxIFER is ϵ -differential privacy preserving.

3 Methodology

The authors of the original paper used the same file to generate the plots for different datasets. We adopted and built on the code the authors sent us. In particular, we needed to train the models for this paper to tailor for the specific network architecture/dataset. We verify the claims in the following way:

- Claim 1: ApproxIFER is a model-agnostic framework.
We compare the performance of ApproxIFER for different number of stragglers and queries for different underlying networks - ResNet-18 (included in the original paper) and ResNet-34 (not included in the original paper).
- Claim 2: ApproxIFER has straggler-resiliency.
We compare the accuracy of ApproxIFER with the base model (labeled as Berrut in relevant plots) for different number of stragglers and number of queries for various network architectures such as ResNet-18 (included in the original paper) and ResNet-34 (not included in the original paper).
- Claim 3: ApproxIFER is Byzantine Robust.
We test the performance of ApproxIFER with the same experimental setups, including the number of queries and the number of adversarial workers, as the original

paper. The network architectures adopted are ResNet-18 (included in the original paper) and ResNet-34 (not included in the original paper).

The simulations are currently run on Google Colab for the lightweight neural networks. For those that demand large amount of memory and computation power, we plan to implement with Great Lakes. [all on Colab](#)

3.1 Model descriptions

A residual neural network (ResNet) is an artificial neural network (ANN). ResNets typically have a very deep feedforward neural network with hundreds of layers, much deeper than previous neural networks. Skip connections or shortcuts are used to jump over some layers. In this report, we consider the ResNet-18 architecture which is a convolutional neural network with 18 deep layers compared to ResNet-34 which is also a CNN with 34 deep layers.

In this paper, a prediction serving system with N workers in total is considered. The system predicts the input queries based on a pre-trained model, which can be regarded as a function and denoted by f . Note that all N workers hold the same model. The inputs are grouped into K queries, denoted by X_i for $i \in [K-1] := \{0, 1, \dots, K-1\}$. The prediction serving system aims to predict the results based on the K input queries and the given model f , say, $f(X_i)$ for $i \in [K-1]$. We denote the predictions as $Y_j = f(X_j)$ for $j \in [K-1]$. The system can tolerate up to S stragglers and E erroneous workers. Based on the paper, ApproxIFER mainly consists of four modules: input queries grouping, encoder, worker prediction, and decoder. The latter three modules are shown in Figure 1 from [1].

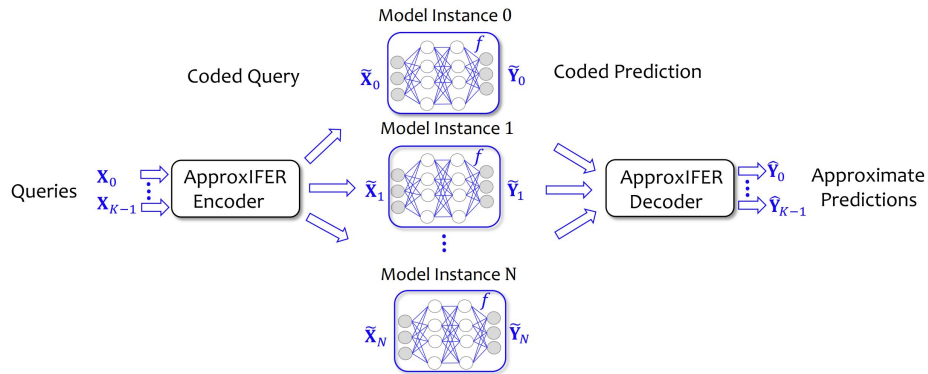


Figure 1. The structure of ApproxIFER [1]

We now describe each module of the architecture in detail.

Input queries grouping – As discussed, the inputs are grouped into a group of K queries, denoted by X_0, \dots, X_{K-1} .

Encoder – The K input queries are coded into N coded queries in the encoder. The coded queries are denoted by \hat{X}_i , where $i \in [N-1]$. The encoding in this paper employs Berrut’s rational interpolant [8]. A rational function u is computed as

$$u(z) = \sum_{j \in [K-1]} X_j \cdot l_j(z),$$

where $l_j(z)$ is defined by

$$l_j(z) = \frac{(-1)^j/(z - \alpha_j)}{\sum_{k \in [K-1]} (-1)^k/(z - \alpha_k)},$$

and the α_k 's are selected as the Chebyshev points of the first kind $\alpha_k = \cos \frac{(2k+1)\pi}{2K}$, for all $k \in [K-1]$. The i th coded query for the i th worker is generated by evaluating u at Chebyshev points of the second kind $\beta_i = \cos(i\pi/N)$. That is, for $i \in [N-1]$, the i th worker receives $\tilde{X}_i = u(\beta_i)$, from the encoding block of the prediction system administrator.

Worker prediction – The i th worker computes the prediction based on the given coded query \tilde{X}_i , yields

$$\tilde{Y}_i = f(\tilde{X}_i) = f(u(\beta_i)),$$

where $i \in [N-1]$.

Decoder – When there are no erroneous workers, i.e., $E = 0$, the decoder waits for K workers then decode. When there are at most $E > 0$ erroneous workers, the decoder waits for $2(E + K)$ workers, apply the proposed ApproxIFER error-locator algorithm to indicate which workers are adversarial workers, then exclude them from the collected predictions on the coded queries.

Either case, the approximate predictions can be recovered from the results of the workers who returned the correct coded predictions, which we denote the set of those indices as \mathcal{F} . The approximate predictions are recovered as

$$\tilde{Y}_j = r(\alpha_j),$$

for all $j \in [K-1]$, where \tilde{Y}_j is the approximate of the prediction of the input query X_j , i.e., $\tilde{Y}_j \approx f(X_j)$, and the rational function r is defined by

$$r(z) = \frac{1}{\sum_{i \in \mathcal{F}} (-1)^i/(z - \beta_i)} \sum_{k \in \mathcal{F}} \frac{(-1)^k}{(z - \beta_k)} f(\tilde{X}_i),$$

where the cardinality of the set \mathcal{F} is, according to the original paper, $|\mathcal{F}| = K$ when $E = 0$ and $|\mathcal{F}| = 2(K + E)$ when $E > 0$. Note that we believe there is a typo in the cardinality of the latter case in the original paper, which should be $2K + E$ instead.

3.2 Datasets

The paper considers well-known datasets such as MNIST, Fashion-MNIST, and CIFAR-10. For the purpose of this report, we have run our tests on MNIST and CIFAR-10.

- **MNIST:** MNIST is a large database of handwritten digits that is commonly used for training various image processing systems. The MNIST database contains 60,000 training images and 10,000 testing images.
- **CIFAR-10:** CIFAR-10 is a balanced dataset of images that are commonly used to train machine learning and computer vision algorithms. The CIFAR-10 dataset contains 60,000 32×32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class

3.3 Hyperparameters

Since this work mainly focuses on approximation predictions on prediction serving systems, there are no need to train new machine learning models, hence there are no hyperparameters for training. All models are trained already and assigned to all workers. Note that the trained models are obtained online.

3.4 Experimental setup and code

The experiments were set up as per the original paper to test the accuracy of ApproxIFER on the CIFAR-10 and MNIST datasets. The original paper considered the ResNet-18 network model, while we additionally test our implementation on ResNet-34 as well. For the Byzantine worker-only experiments, the original paper assume the malicious workers distort their predictions by a zero-mean unit-variance Gaussian noise, i.e., $\tilde{Y}_i = f(\tilde{X}_i) + N_i$ with $N_i \sim \mathcal{N}(0, 1)$. It is claimed in the original paper that choosing different standard deviation σ for the Gaussian noise yields similar performance with the proposed ApproxIFER algorithm. Hence we also included results with $\sigma = 10$ and $\sigma = 100$. In particular, we tested the following cases for our experiments (for both architectures).

- Straggler only
 - $S = 1, K = 8, 12$
 - $S = 2, K = 8$
 - $S = 3, K = 8$
- Byzantine worker only
 - $S = 0, E = 1, 2, 3, K = 12, \sigma = 1$
 - $S = 0, E = 1, 2, 3, K = 12, \sigma = 10$
 - $S = 0, E = 1, 2, 3, K = 12, \sigma = 100$

We modified the authors' code to adapt to different datasets and models.

3.5 Computational requirements

The experiments were run in Google Colaboratory. A comparison of the average and total runtimes are given in Tables 1 to 8 for the straggler-only cases, and Tables 9 to 14 for the Byzantine worker-only cases below.

4 Results

Our reproduced results are in line with those of the original paper.

4.1 Results reproducing original paper

For the straggler-only cases, Figure 2 shows the model accuracy of ResNet-18 as presented in the original paper for all three datasets. Figures 3 and 4 show our reproduced results for MNIST and CIFAR-10 respectively. The plots verify the second claim of the original paper in Section 2. In the figures, "Berrut" and "Centralized" correspond to the proposed ApproxIFER algorithm and the base model, respectively. A comparison of the accuracies are presented below in Table 15. Here, K is the number of queries and S is the maximal number of stragglers in the system.

For the Byzantine worker-only scenario, the model accuracy for all three datasets of ResNet-18 from the original paper is shown in Figure 5. We compare our reproduced results for MNIST and CIFAR-10 with those of the original authors in Table 16. To verify that the scheme, including the ApproxIFER error-locator algorithm, works for other network architectures as well, the original authors show the performance of several models for the CIFAR-10 dataset with the choice $K = 12, S = 0, E = 2$, as shown in Figure 6. For ResNet-34, the performance of the reproduced accuracy is compared with that from the original paper in Table 17b.

Batch Size	CIFAR-10	MNIST
8	0.298601	0.318697
12	0.429150	0.447539

Table 1. Average runtimes (in seconds) for ResNet-18 with $S = 1$

Batch Size	CIFAR-10	MNIST
8	28.967860	97.652539
12	44.250905	143.729705

Table 2. Total runtimes (in seconds) for ResNet-18 with $S = 1$

S	CIFAR-10	MNIST
2	0.311861	0.353068
3	0.465142	0.367419

Table 3. Average runtimes (in seconds) for ResNet-18 with batch size 8

S	CIFAR-10	MNIST
2	30.571228	100.934360
3	32.919277	105.565045

Table 4. Total runtimes (in seconds) for ResNet-18 with batch size 8

E	CIFAR-10	MNIST
1	0.605085	0.766439
2	0.625307	0.761986
3	0.657348	0.780996

Table 9. Average runtimes (in seconds) for ResNet-18 with $S = 0, K = 12, \sigma = 1$

E	CIFAR-10	MNIST
1	0.607650	0.739475
2	0.640402	0.756661
3	0.645748	0.806117

Table 10. Average runtimes (in seconds) for ResNet-18 with $S = 0, K = 12, \sigma = 10$

E	CIFAR-10	MNIST
1	0.634884	0.733602
2	0.643297	0.762474
3	0.674559	0.808710

Table 11. Average runtimes (in seconds) for ResNet-18 with $S = 0, K = 12, \sigma = 100$

Batch Size	CIFAR-10	MNIST
8	0.570133	0.354219
12	0.773196	0.532492

Table 5. Average runtimes (in seconds) for ResNet-34 with $S = 1$

Batch Size	CIFAR-10	MNIST
8	52.904114	105.346835
12	78.956201	156.041335

Table 6. Total runtimes (in seconds) for ResNet-34 with $S = 1$

S	CIFAR-10	MNIST
2	0.568228	0.376716
3	0.606278	0.388680

Table 7. Average runtimes (in seconds) for ResNet-34 with batch size 8

S	CIFAR-10	MNIST
2	55.683345	113.940276
3	60.359065	116.704551

Table 8. Total runtimes (in seconds) for ResNet-34 with batch size 8

E	CIFAR-10	MNIST
1	2.154675	0.838936
2	2.249086	0.862457
3	2.348390	0.874289

Table 12. Average runtimes (in seconds) for ResNet-34 with $S = 0, K = 12, \sigma = 1$

E	CIFAR-10	MNIST
1	2.157174	0.868974
2	2.235968	0.807873
3	2.341396	0.851835

Table 13. Average runtimes (in seconds) for ResNet-34 with $S = 0, K = 12, \sigma = 10$

E	CIFAR-10	MNIST
1	2.117511	0.875757
2	2.224983	0.828852
3	2.383582	0.876940

Table 14. Average runtimes (in seconds) for ResNet-34 with $S = 0, K = 12, \sigma = 100$

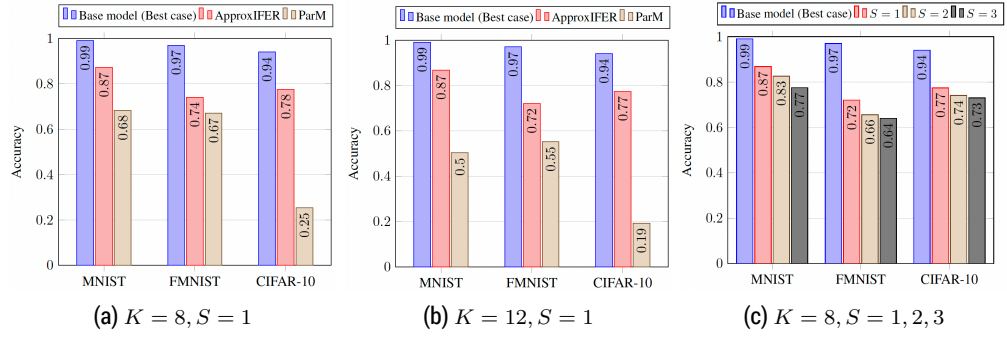


Figure 2. Model accuracies for ResNet-18 [1]

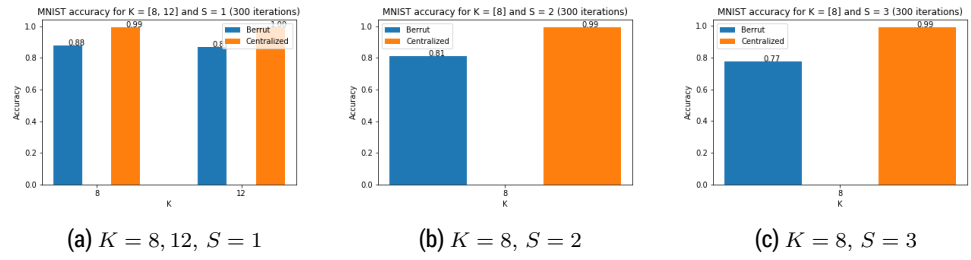


Figure 3. Model accuracies for ResNet-18 on MNIST

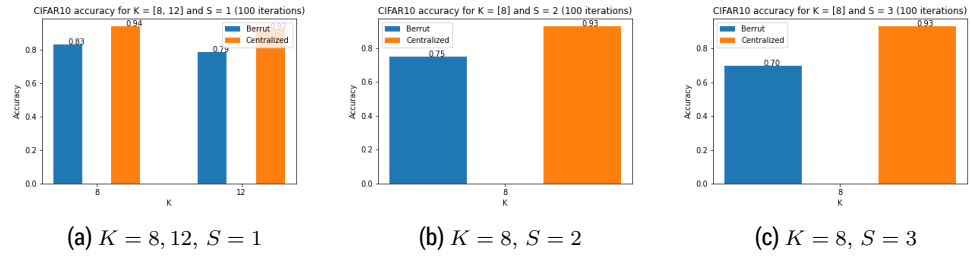


Figure 4. Model accuracies for ResNet-18 on CIFAR-10

(K, S)	Accuracy (authors)	Accuracy (us)	(K, S)	Accuracy (authors)	Accuracy (us)
(8, 1)	0.87	0.88	(8, 1)	0.78	0.83
(12, 1)	0.87	0.86	(12, 1)	0.77	0.79
(8, 2)	0.83	0.81	(8, 2)	0.74	0.75
(8, 3)	0.77	0.77	(8, 3)	0.73	0.70

(a) MNIST

(b) CIFAR-10

Table 15. Accuracy of ApproxIFER with base model ResNet-18

E	Accuracy (authors)	Accuracy (us)	E	Accuracy (authors)	Accuracy (us)
0 (base)	0.99	0.99	0 (base)	0.94	0.94
1	0.97	0.98	1	0.9	0.91
2	0.95	0.97	2	0.9	0.9
3	0.93	0.97	3	0.9	0.9

(a) MNIST

(b) CIFAR-10

Table 16. Accuracy of ApproxIFER with base model ResNet-18 in Byzantine worker-only scenario with $K = 12, S = 0$, and $\sigma = 1.0$

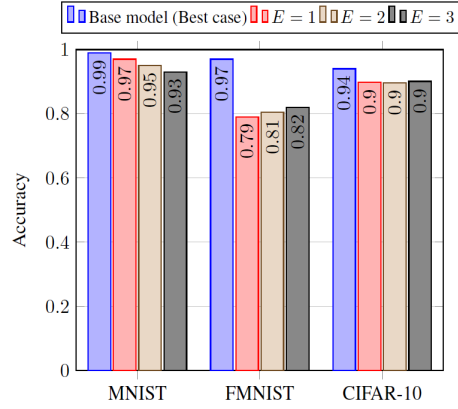


Figure 5. Model accuracies for ResNet-18 with $K = 12$, $S = 0$, $E = 1, 2, 3$ and $\sigma = 1.0$ from [1]

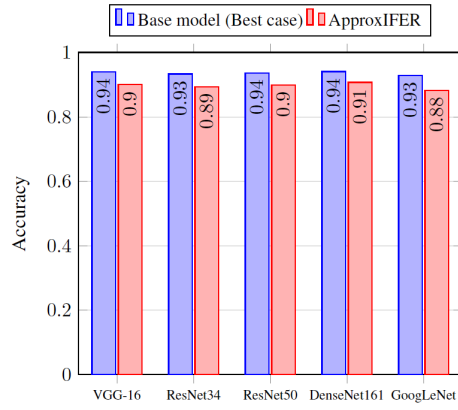


Figure 6. Model accuracies for several models on CIFAR-10 with $K = 12$, $S = 0$, $E = 2$ from [1]

E	Accuracy (authors)	Accuracy (us)	E	Accuracy (authors)	Accuracy (us)
0 (base)	n/a	0.99	0 (base)	0.93	0.94
1	n/a	0.98	1	n/a	0.91
2	n/a	0.98	2	0.89	0.9
3	n/a	0.97	3	n/a	0.9

(a) MNIST
(b) CIFAR-10

Table 17. Accuracy of ApproxIFER with base model ResNet-34 in Byzantine worker-only scenario with $K = 12$, $S = 0$, and $\sigma = 1$

4.2 Results beyond original paper

ResNet-34 – For the straggler-only case, we tested the performance of ApproxIFER with ResNet-34 on the datasets CIFAR-10 and MNIST for the same combinations of K and S as in Section 4.1. Our results support the second claim and are shown in Figure 8 and Figure 9 (in comparison to the authors' original results in Figure 7). Furthermore, we analyzed and compared the runtimes with those on ResNet-18 in Tables 5 to 8, and observed that they follow similar trends. For a comparison of the model accuracies, see Tables 18 and 19.

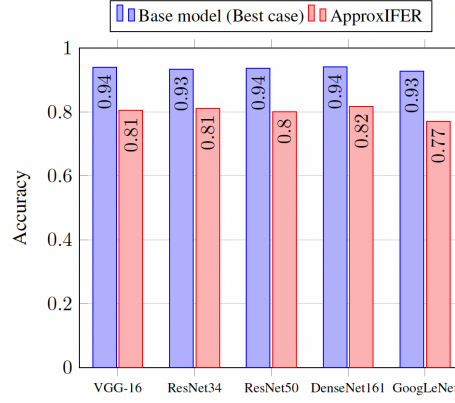


Figure 7. Model accuracies for several models on CIFAR-10 with $K = 8, S = 1$ [1]

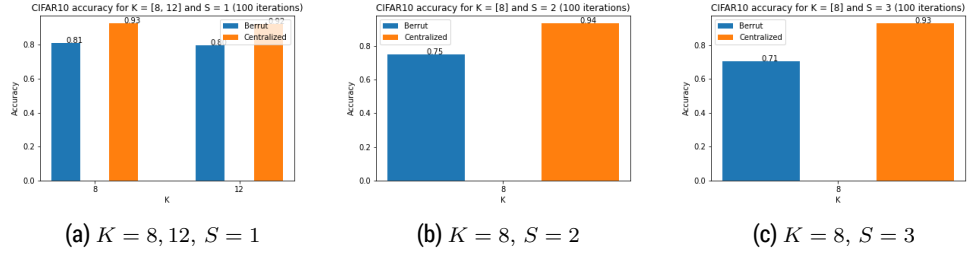


Figure 8. Model accuracies for ResNet-34 on CIFAR-10

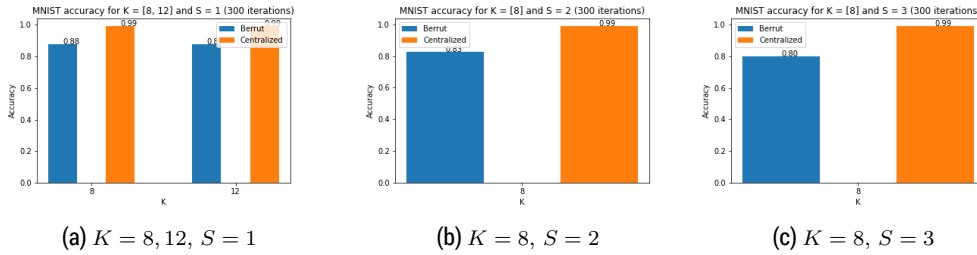


Figure 9. Model accuracies for ResNet-34 on MNIST

For the Byzantine worker-only scenario, the original authors compare the performance of several network architectures for the CIFAR-10 dataset with the choice $K = 12, S = 0, E = 2$. We test the scheme, including the ApproxIFER error-locator algorithm **James**

(K, S)	Base (ResNet-18)	ApproxIFER (ResNet-18)	Base (ResNet-34)	ApproxIFER (ResNet-34)	Base (ResNet-152)	ApproxIFER (ResNet-152)
(8, 1)	0.93	0.79	0.94	0.82	0.84	0.68
(12, 1)	0.92	0.80	0.94	0.80	0.80	0.65
(8, 2)	0.94	0.78	0.93	0.77	0.81	0.64
(8, 3)	0.94	0.71	0.94	0.72	0.78	0.59

Table 18. Accuracy of the base model and ApproxIFER for ResNet-18, ResNet-34, and ResNet-152 on CIFAR-10

(K, S)	Base (ResNet-18)	ApproxIFER (ResNet-18)	Base (ResNet-34)	ApproxIFER (ResNet-34)	Base (ResNet-152)	ApproxIFER (ResNet-152)
(8, 1)	0.94	0.83	0.93	0.81	0.99	0.86
(12, 1)	0.92	0.79	0.92	0.80	0.99	0.85
(8, 2)	0.93	0.75	0.94	0.75	0.99	0.81
(8, 3)	0.93	0.70	0.93	0.71	0.99	0.77

Table 19. Accuracy of the base model and ApproxIFER for ResNet-18, ResNet-34, and ResNet-152 on MNIST

E	$\sigma = 1$	$\sigma = 10$	$\sigma = 100$
0 (base)	0.9902	0.9900	0.9892
1	0.9794	0.9656	0.9747
2	0.9792	0.9525	0.9589
3	0.9719	0.9503	0.9514

(a) MNIST

(b) CIFAR-10

Table 20. Accuracy of ApproxIFER with base model ResNet-34 in Byzantine worker-only scenario with $K = 12, S = 0$, and $\sigma = 1, 10, 100$

ResNet-152 – We also tested the performance of ApproxIFER for the straggler-only case with ResNet-152 on CIFAR-10 and MNIST for the same combinations of K and S as in Section 4.1. Both the base model and the ApproxIFER had relatively lower accuracies on CIFAR-10 (Table 18 and Figure 10), while the base model accuracy of ResNet-152 was almost perfect on MNIST (Table 19 and Figure 11).

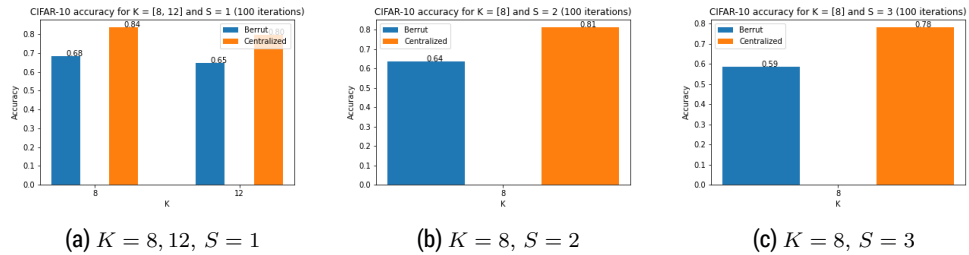


Figure 10. Model accuracies for ResNet-152 on CIFAR-10

Privacy guarantees – The original ApproxIFER does not prevent privacy leakage for the K queries X_0, \dots, X_{K-1} . Thus, to protect privacy from attackers, we propose a novel differentially private scheme Private-ApproxIFER. Note that no one else has tried to address this problem before; hence the results presented in this section are completely new. We first begin by defining local differential privacy.

Definition 1 (ϵ -Local Differential Privacy). A randomized mechanism \mathcal{K} satisfies ϵ -LDP if and only if for any pairs of input values v, v' in the domain of M , and for any possible output $y \in \mathcal{Y}$, it holds

$$\Pr[\mathcal{K}(v) = y] \leq e^\epsilon \cdot \Pr[\mathcal{K}(v') = y]. \quad (1)$$

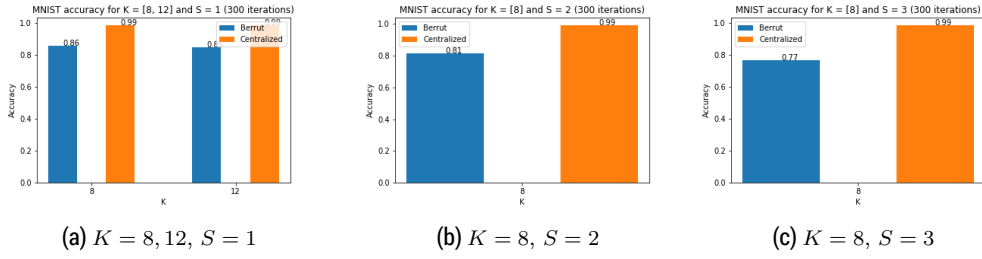


Figure 11. Model accuracies for ResNet-152 on MNIST

We call two datasets neighbouring if they only differ in a single row (single entry).

Definition 2 (Sensitivity). For two neighboring datasets D and D' in an individual client, with a query function $f_i : \mathcal{D} \rightarrow \mathbb{R}$, the l_2 local sensitivity is defined as follows:

$$\Delta = \max_{\text{dist}(D, D')=1} \|f_i(D) - f_i(D')\|_2. \quad (2)$$

Next, we provide the definition of the Laplace distribution.

Definition 3 (Laplace distribution). We denote the Laplace distribution as

$$\text{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right), \quad (3)$$

where the variance of the distribution is $\sigma^2 = 2b^2$.

Based on the K queries $\mathbf{X}_0, \dots, \mathbf{X}_{K-1} \in \mathbb{R}^{m \times n}$, given two neighboring datasets, denoted by $D = \{\mathbf{X}_0, \dots, \mathbf{X}_{K-1}\}$ and $D' = \{\mathbf{X}_0, \dots, \mathbf{X}'_{K-1}\}$, where \mathbf{X}_{K-1} and \mathbf{X}'_{K-1} only differ in a single row, which means that only a single record in these two queries is different. Combining the former definitions, we give the definition of the Laplace mechanism.

Definition 4 (Laplace mechanism). Consider a query function applying on a dataset D , the Laplace mechanism of \mathcal{M}_i is shown as follows:

$$\mathcal{M}_i(D) = f_i(D) + \mathbf{N},$$

where all entries in the noise matrix \mathbf{N} are i.i.d. and drawn from a Laplace distribution with $\mu = 0$, $\sigma = \sqrt{2}\Delta/\epsilon$, then $b = \Delta/\epsilon$.

The following theorem states that Private-ApproxIFER satisfies differential privacy.

Theorem 1. The Laplace mechanism for Private-ApproxIFER preserves ϵ -differential privacy.

Proof. To prove that Private-ApproxIFER is ϵ -differentially private, we need

$$\left| \log \frac{p_{\mathcal{M}_i(D)}(y)}{p_{\mathcal{M}_i(D')}(y)} \right| \leq \epsilon, \quad (4)$$

where $p_{\mathcal{M}_i(D)}(y)$ and $p_{\mathcal{M}_i(D')}(y)$ are the probability density functions for $\mathcal{M}_i(D)$ and

$\mathcal{M}_i(D')$ for adding Laplacian noises, respectively. We have

$$\begin{aligned}
\frac{p_{\mathcal{M}_i(D)}(y)}{p_{\mathcal{M}_i(D')}(y)} &= \prod_{k=1}^{mn} \frac{\exp(-\|f_i(D)_k - y_k\|_1/b)}{\exp(-\|f_i(D')_k - y_k\|_1/b)} \\
&= \prod_{k=1}^{mn} \exp\left(\frac{\|f_i(D')_k - y_k\|_1 - \|f_i(D)_k - y_k\|_1}{b}\right) \\
&\leq \prod_{k=1}^{mn} \exp\left(\frac{\|f_i(D')_k - f_i(D)_k\|_1}{b}\right) \\
&= \exp\left(\frac{\|f_i(D') - f_i(D)\|_1}{b}\right) \\
&= \exp\left(\frac{\|f_i(D') - f_i(D)\|_1}{\Delta/\epsilon}\right) \\
&\leq \exp(\epsilon),
\end{aligned}$$

which yields (4). □

5 Discussion

At this point, we have reproduced the model accuracy results of the proposed ApproxIFER scheme for two neural network models on two datasets, with various combinations of the number of input queues K and the number of stragglers S . The reproduced results match those found in the original paper closely, and hence firmly support Claim 2, that the scheme has straggler-resilience. Training on ResNet-34, we see that the base model has accuracy similar to ResNet-18 (for all K, S), and both their ApproxIFER realizations also have similar accuracies (Table 18 and Table 19). This supports the authors' claim that the ApproxIFER scheme is model-agnostic thereby verifying Claim 1. We continue to work on verifying the performance of the base model and the scheme for other models shown in the paper, and also on testing the performance of the two current models for additional datasets, e.g., Fashion MNIST. In addition, in order to verify Claim 3 regarding the performance of the ApproxIFER scheme under the attack of Byzantine workers, we are simultaneously tackling minor issues in running the error-locator code blocks. Data privacy is a huge concern, especially when outsourcing data/model. Therefore, we extended this work by trying to look at the problem from the perspective of Differential Privacy. We prove that the proposed private-ApproxIFER is ϵ differentially private and therefore is secure against model inversion attacks etc.

5.1 What was easy

The original paper is well-written, which made the proposed framework for the ApproxIFER prediction system easy to follow despite its theoretical difficulty. The parameters applied in the experiments were clear, and we had a good source file to work off of.

5.2 What was difficult

The original paper was conceptually challenging. The paper does not look at any traditional machine learning techniques, but presents a model-agnostic scheme to efficiently use resources for robust prediction systems.

Theoretical Aspects – The theory behind the proposed algorithm is complicated and required further inspection of the references. Although we had the authors' code for the experiments, it was still difficult to implement. There are too many files for many datasets

and different parameter settings for the number of stragglers and erroneous workers. So we had to rewrite various parts of the code to be able to efficiently reproduce the results and make it adaptable to newer instances and architectures.

Additionally, we attempted to present additional theoretical results, which were out of the scope of both the original paper, and the class curriculum. Therefore, understanding the relevant theory to these additional results was also very difficult and challenging.

Experimental Aspects – We were able to obtain pre-trained ResNet-18 and ResNet-34 on CIFAR-10 [9] and pre-trained ResNet-18 on MNIST (provided by the authors). So we had to train ResNet-34 on MNIST for the results in Section 4.2. For that we adopted the code from [10]. The free GPU on Google Colaboratory was not always available. When it was available, the training took 4.76 minutes.

For ResNet-152, we ran into multiple issues while training that were time consuming to fix. Training on CIFAR-10 and MNIST each took longer than 100 minutes even with access to the free GPU on Google Colaboratory. Moreover, running each experiment took around 10 minutes.

5.3 Communication with original authors

We were in contact with the authors at the beginning of this study. Thankfully, the authors shared the code of their experiments with us. We did not need to contact the authors thereafter.

References

1. M. Soleymani, R. E. Ali, H. MahdaviFar, and A. S. Avestimehr. "ApproxIFER: A Model-Agnostic Approach to Resilient and Robust Prediction Serving Systems." In: **Proceedings of the AAAI Conference on Artificial Intelligence** 36.8 (June 2022), pp. 8342–8350.
2. J. Kosaian, K. V. Rashmi, and S. Venkataraman. "Parity Models: Erasure-Coded Resilience for Prediction Serving Systems." In: *SOSP '19* (2019), pp. 30–46.
3. K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. "Practical secure aggregation for privacy-preserving machine learning." In: **Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security**. 2017, pp. 1175–1191.
4. J. So, B. Güler, and A. S. Avestimehr. "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning." In: **IEEE Journal on Selected Areas in Information Theory** 2.1 (2021), pp. 479–489.
5. S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran. "FastSecAgg: Scalable Secure Aggregation for Privacy-Preserving Federated Learning." In: **arXiv preprint arXiv:2009.11248** (2020).
6. Y. Zhao and H. Sun. "Information Theoretic Secure Aggregation with User Dropouts." In: **arXiv preprint arXiv:2101.07750** (2021).
7. J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. "Secure single-server aggregation with (poly) logarithmic overhead." In: **Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security**. 2020, pp. 1253–1269.
8. T. Jahani-Nezhad and M. A. Maddah-ali. "Berrut Approximated Coded Computing: Straggler Resistance Beyond Polynomial Computing." In: **IEEE Transactions on Pattern Analysis and Machine Intelligence** (2022), pp. 1–1.
9. H. Phan. **PyTorch Models Trained on CIFAR-10 Dataset**. https://github.com/huyvnphan/PyTorch_CIFAR10. Version v3.0.1. Jan. 2021.
10. S. Raschka. **Deep Learning Models**. https://github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/cnn/cnn-resnet34-mnist.ipynb. June 2019.