ISBN: 978-981-18-3959-7
2022 the 12th International Workshop on Computer Science and Engineering (WCSE 2022)
doi: 10.18178/wcse.2022.06.014

Differentially Private, Federated Learning on Gene Expression Data for Tumour Classification

Souhail Meftah [1,3 +], Meenatchi S. M. S. Annamalai [1], Dominic J. F. Byrne [1,2], Khin Mi Mi Aung [1] and Bharadwaj Veeravalli [3]

[1] Institute for Infocomm Research, A*STAR, Singapore
[2] The University of Manchester, Manchester, UK
[3] The National University of Singapore, Singapore

Abstract. Over recent years, machine learning (ML) methods have enabled considerable progress to be made within a variety of data-rich research domains. Genomics is one prominent field, with ML-based approaches achieving exciting results in a range of historically difficult tasks. One such area where ML approaches have achieved particular success is the classification of cancerous tissues using gene expression data. Despite this success, recent advances share a common issue with many other areas of ML research - state-of-the-art performance relies upon the aggregation of large training datasets. Many have raised concerns over the implications such large-scale data aggregation practice has for data privacy, and the nature of genomic data makes such concerns all the more pertinent within a cancer classification setting. Because of the sensitivity of such data, strict policies are enacted to protect patient privacy. Whilst currently unavoidable, such policies are a key roadblock to advances in ML research in this domain. Federated learning (FL) aims to solve this issue by allowing ML models to be trained on decentralized data. Despite the promise held by this approach, recent work has demonstrated that FL is not sufficient for fully private training of ML models. The incorporation of differential privacy (DP) into an FL framework has previously been shown to enable decentralized model training in a privacy-preserving manner. We introduce a method to incorporate custom differentially private algorithms directly into federated learning workflows that are not currently implemented by privacy libraries such as Opacus.

Keywords: differential-privacy, federated learning, gene expression

1. Introduction

1.1. Context

A surge in data availability across healthcare disciplines in recent years makes this domain an immensely exciting one for machine learning (ML) research, perhaps none more so than genomics. Rapid improvements in sequencing technologies have brought about a flood of large novel datasets, whose complexity often hampers progress by more traditional approaches. It is unsurprising then, that we have seen numerous successful applications of various ML approaches to classification tasks using gene expression datasets [1-6], such as the one used in the present study.

Despite holding much promise, advances in this research domain typically require the aggregation of large amounts of genomic data, giving rise to significant privacy concerns. Additionally, hospitals often have strict privacy policies that prevent them from sharing their patient data even in encrypted form. These concerns form a significant roadblock to the adoption of ML in healthcare research despite the huge potential.

Federated learning (FL) [7] was introduced to solve the privacy problem by allowing a global ML model to be learned from decentralized local data without the data ever leaving the device. To that end, many implementations of FL have been presented, PySyft [8] being one of the main libraries used in research settings. While this was a promising approach to protect privacy, the recent swathe of membership inference attacks [9] have shown that it is still possible for fully trained models to leak sensitive information about

+ Corresponding author. Tel: +65 9354 1627
E-mail address: stusm@i2r.a-star.edu.sg; souhailmeftah@gmail.com.

individuals in the training dataset, therefore, questioning the adequacy of pure FL approaches in ensuring privacy. Still, FL is particularly useful in the medical context as it provides a way for hospitals to engage in collaborative learning without sharing their patient data directly.

Differential Privacy[10], on the other hand, is a mathematical formalism that solves the privacy problem by providing strong guarantees bounding the information leakage due to the participation of a single user in a dataset. Differentially private (DP) mechanisms, therefore, enable ML to be conducted on sensitive datasets such as the present gene expression dataset without breaching the privacy of any single user. As such, DP provides an enhancement on top of the utility of FL by further providing strong mathematical guarantees that directly protect the privacy of the patient data during the learning process. Hence, federated learning put together with differential privacy can be a more efficient solution to enable secure ML in healthcare research compared to traditional approaches such as Fully Homomorphic Encryption which can be orders of magnitude slower [11] [12].

1.2. Problem Statement

Whilst integrating between FL and DP is not an entirely new concept, current implementations suffer from significant disadvantages. Firstly, some of these implementations are bespoke [13][14] and have not been well-integrated and optimized for workflows involving open-source federated learning libraries such as PySyft. This, therefore, hinders the usage of such frameworks in research as the implementation code may not be published or be general enough for usage in various settings. Secondly, the implementations that do integrate with federated learning libraries such as Opacus [15], exclusively implement the DP-SGD algorithm hindering the usage of the framework with custom DP algorithms such as output perturbation [16] and approximate minima perturbation [17] which may prove to be more suitable in certain settings. Therefore, one of the biggest impediments to more FL-DP integration is the lack of a clear framework that is flexible enough to accommodate custom DP algorithms but also integrates well with and is well optimized for existing FL libraries.

1.3. Contributions

In this work, we present how custom DP algorithms can be implemented, specifically in the PySyft library. Furthermore, we optimize the implementation to make use of primitives in PySyft, namely the Plan architecture, and show how the optimization indeed brings down the communication cost significantly. Finally, we use this method to successfully train a binary cancer classifier in a decentralized, privacy-preserving way. We show that our custom DP implementation results in a smaller accuracy drop from the non-private version compared to prior work done using the Opacus-PySyft integration [18] proving the utility of such a custom framework. Furthermore, as the custom DP algorithm we used can be swapped out for any other algorithm seamlessly, the framework is flexible and well optimized for the PySyft library; which we hope will prove useful to researchers for the development and testing of their custom DP algorithms in the FL context. To that end, we provide our full implementation at: (https://drive.google.com/file/d/17tH8zTjDrswFPGyL7HQA_XiEc9ljyWZc/view?usp=sharing)

2. Preliminaries

2.1. Differential Privacy

Differential Privacy [10] provides strong mathematical guarantees protecting the users' privacy. These guarantees are provided directly in Definition 1. Under this definition, ε is the privacy parameter and δ is the failure probability.

Definition 1 (Differential Privacy [10]). A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $S \subseteq$ Range($\mathcal{M}$) and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $||x - y||_1 \leq 1$:

$$\Pr[\mathcal{M}(x) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(y) \in S] + \delta$$

Differential privacy is a popular formalism due to the theorems that support it such as the Postprocessing theorem (Theorem 1) and Composition theorem (Theorem 2) which allow the results of differentially private mechanisms to be further used and combined without completely breaking the privacy guarantees.

Theorem 1 (Postprocessing Theorem [10]). Let $\mathcal{M}: \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}$ be a randomized algorithm that is (ε, δ)-differentially private. Let $f: \mathcal{R} \to \mathcal{R}'$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M}: \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}'$ is (ε, δ)-differentially private.

Theorem 2 (Basic Composition Theorem [10]). Let $\mathcal{M}_1: \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_1$ and $\mathcal{M}_2: \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_2$ be randomized algorithms that are $(\varepsilon_1, \delta_1)$- and $(\varepsilon_2, \delta_2)$-differentially private respectively. Then their combination defined to be $\mathcal{M}_{1,2}: \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_1 \times \mathcal{R}_2$ by the mapping $\mathcal{M}_{1,2}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differentially private.

2.2. Output Perturbation (OP)

Output perturbation [16] was introduced as an extension from the sensitivity method [10] to the class of empirical risk minimization (ERM) classification problems. Although later methods such as the objective perturbation method used in the DP-SGD algorithm [19] have been shown to be theoretically and empirically superior, the DP-SGD algorithm has already been widely implemented in the Opacus library which provides easy integration with the PySyft library [20]. Since in this paper, we are looking at how custom DP algorithms can be integrated into a PySyft workflow, this algorithm was chosen for its simplicity and computational efficiency. The output perturbation algorithm which is given in Algorithm 1 is proven to be $(\varepsilon, 0)$-differentially private. However, one of the caveats of using output perturbation in real-world algorithms is that the privacy guarantees depend on the exact minima of the function to be found. This is not a realistic assumption as in real-world algorithms, minima are often approximated over large numbers of iterations and computers generally operate under finite precision arithmetic. In order to use output perturbation correctly, modifications are necessary as can be seen in Section 3.5.

Algorithm 1: Output perturbation

Input: Data $\mathcal{D} = \{z_i\}$, Privacy parameter ε, Regularization parameter Λ, Loss function J

Output: Approximate minimizer $\mathbf{f_{priv}}$

1 Draw a vector $\mathbf{b}$ according to pdf $v(\mathbf{b}) = \frac{1}{\alpha} e^{-\beta||\mathbf{b}||}$ where $\beta = \frac{n\Lambda\varepsilon}{2}$ and α is a normalizing constant

2 Compute $\mathbf{f_{priv}} = \text{argmin} J(\mathbf{f}, \mathcal{D}) + \mathbf{b}$

3 **return** $\mathbf{f_{priv}}$

Federated learning, as mentioned previously, enables a global model to be learned from decentralized local data. While there are many system architectures that enable FL, a key primitive in FL architectures is the *plan*. First introduced in [21], plans significantly reduce the bandwidth usage of FL workflows. This is important as high bandwidth protocols lead to increased communication cost and power consumption which are significant costs in decentralized setups which usually consist of mobile devices instead of powerful servers. Plans mitigate the bandwidth usage of FL protocols by compressing potentially n messages for n operations on remote inputs to a single message that represents all n operations - the inputs of which are referenced through pointers. They do so by serializing a sequence of tensor operations into a single message that can be sent to and executed by all parties.

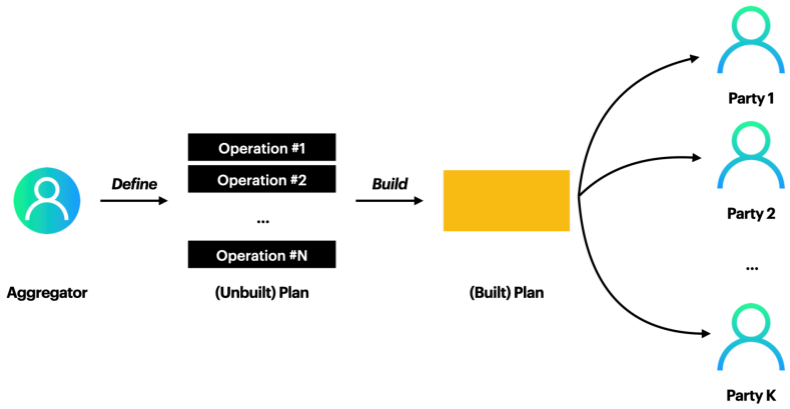


Fig. 1: System architecture for PySyft Pla

Specifically, in PySyft, the *@sy.func2plan* decorator attempts to transform a given python function containing torch operations into a plan. As illustrated in Figure 1, before sending the plan to a remote worker for execution, it has to be first *built* or executed on dummy input for the tensor operations to be tracked and recorded into the message. The built plan can then be sent to each party in a single message who can then execute the plan on their local data and return the result of the plan for aggregation, thus resulting in overall low bandwidth usage.

One of the main issues with the way plans are implemented currently is that since tensor operations need to be traced, only operations that are 'hook'ed by the Pytorch library can be used. This means that operations such as if and while cannot be converted into plans resulting in a major deficiency in the way plans can be used. Furthermore, the probability distributions presented in the torch.distributions package are not hooked as well presenting yet another major challenge to DP integration into FL workflows as almost all DP algorithms require the drawing of noise variables from Laplace or Gaussian distributions. To that end, we present how we mitigated some of these issues to integrate a custom DP algorithm in our implementation which can be seen in Section 3.6.

3. Methodology

3.1. Privacy Analysis

For the privacy analyses presented in this paper, we explicitly disregard the privacy loss due to model selection, feature selection, and hyper-parameter tuning. In order to satisfy differential privacy strictly, it is important to account for the privacy loss in all of these steps. However, as the focus of this paper is on mainly implementing custom deferentially private mechanisms for model training under the federated learning context, without loss of generality, we assume that the privacy parameters would have been split appropriately for each of these steps as necessary. The privacy parameters used in the analysis provided will then be focused entirely on the model training step.

3.2. Dataset

For this work, we consider the breast cancer gene expression data (BC-TCGA), which was made available by The Cancer Genome Atlas Program (TCGA) [22]. These data include expression profiles for 17,814 genes across 590 samples, including 529 samples from breast cancer tissues and 61 samples from phenotypically normal tissues. The 590 samples in the dataset were randomly split into train and test subsets, with a ratio of 80:20.

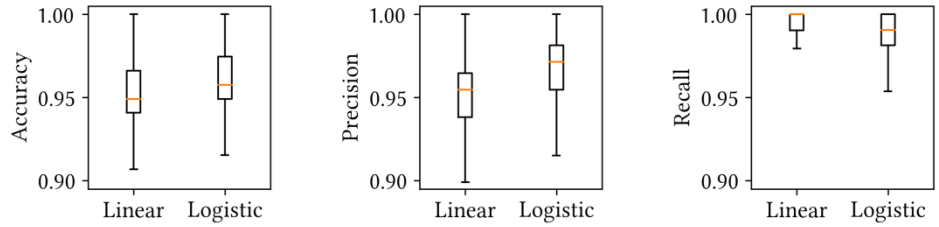


Fig. 2: Comparison between classification performance of Linear Transformation and logistic Regression

3.3. Model Selection

Since the dataset only has a small number of samples and we focus on the integration with differentially private algorithms, we only compared simple models such as a single linear transformation layer and a logistic regression. As can be seen from Figure 2, comparing the accuracy, precision, and recall values over 1000 trials, the logistic regression model has consistently better accuracy and precision values compared to the linear model. Even though the logistic regression has slightly lower recall values compared to the linear model, it is to be noted that it is still considerably high at above 0.95. Therefore logistic regression was fixed as the machine learning model considered in the remaining of the experiments.

3.4. Feature Selection

A key feature of genome-wide expression datasets is their high dimensionality, with the number of features (genes) often being many times greater than the number of samples [23]. This is exemplified by the

tumor expression data used in the present study, which includes expression levels for 17,814 genes from only 590 samples. Given that this gene set represents a significant majority of all human coding genes, we would expect most to play no role in tumorigenesis and would therefore be irrelevant to a tumor classification task. As such, we would expect that failure to remove a significant proportion of these uninformative features would pose issues for model convergence, controlling training time, and improving the accuracy of the resultant classification model. On this basis, we opted to use a feature selection (FS) approach to ameliorate the issues posed by the high dimensionality of these data.

Table 1: The top 9 genes, as selected by each feature selection algorithm

fast-MRMR	CFS	FCBF
ZDHHC18	*POLM*	*COL10A1*
OR2Z1	*IFIT1L*	*EFCBP1*
LOC389791	*KLK9*	*LRRC3B*
OR4K5	*LLGL1*	*CXCL2*
LHX5	*FBN3*	*HOXA4*
GSX1	*FLJ23356*	*CA4*
LINGO4	*KY*	*LOC387911*
OR1K1	*ZNF569*	*WDR51A*
OR2B11	*ZNF526*	*CAV2*

A wide range of FS algorithms have been employed for use with gene expression datasets [24]. We chose to compare the Correlation-based Feature Selection (CFS) [25], Fast Correlation-Based Filter (FCBF) [26]and Minimum Redundancy Maximum Relevance (MRMR) [27] algorithms on both execution time and downstream classification model performance (Table 2). These algorithms in particular were chosen as they have previously been shown to be effective in selecting gene subsets for cancer classification from microarray expression data [28-30]. We used implementations of CFS and FCBF from the scikit-feature python package [31] and an improved implementation of the MRMR algorithm (fast-MRMR) [32].

The expression profiles for each gene were discretized into 5 quantiles for use with FCBF, as this algorithm uses an entropy-based heuristic to select relevant features which require nominal feature variables. The other two algorithms were applied to the data in its raw form. To account for significant differences in the number of features selected by each algorithm and allow for fair comparisons to be made between them, classification performance was assessed using only the 9 highest-ranked features, as selected by each algorithm (Table 1). A logistic regression model was trained on samples from the training subset, using 9 features selected by each algorithm. The three FS algorithms were compared on the accuracy, precision, and recall of the resulting model's predictions on the samples from the test subset (Table 2).

Table 2: Execution time for three feature selection algorithms, as well as the classification performance of a logistic regression model trained on the features selected by each algorithm

Algorithm	Execution Time	Accuracy (%)	Precision (%)	Recall (%)
fast-MRMR	27s	89.0	90.4	98.1
CFS	139m6s	96.0	96.7	98.9
FCBF	17m6s	100	100	100

The results of this preliminary analysis, in which model training was neither federated nor differentially private, clearly indicate FCBF as the best performing FS algorithm of the three tested - the downstream model achieved perfect classification performance on the test set and the runtime of FCBF was significantly shorter than that of CFS. The fast-MRMR algorithm was, by far, the fastest to run. However, the logistic regression model trained on the features it selected achieved the lowest classification performance overall.

Despite its impressive performance during these preliminary experiments, validation of the features selected using FCBF using a logistic regression which incorporated federated, differentially-private training resulted in the model failing to converge. Training the same model on features selected by CFS and fast-MRMR did not reproduce this behavior, suggesting that FCBF is not suited for use with decentralized classification tasks. During federated training, the two models do not exchange parameters every epoch, so as to limit both the communication cost and the noise added by the output perturbation algorithm (See Section 3.5 for further details.). Training the two models separately over multiple epochs in this way may have resulted in the model's failure to converge, although the reason this behavior was specific to FCBF

remains unclear. Zhang et al. have previously reported issues with FCBF suffering from instability due to its use of naive heuristics that are not useful in many situation [33], which may play a role. Yet, further work is required to investigate this apparent issue with FCBF.

After discounting FCBF due to the aforementioned issues with model convergence, CFS was chosen for use in the suggested solution on the basis of its superior downstream classification performance when compared to fast-MRMR. Notably, the execution time for CFS was significantly longer than the other two algorithms - this may pose issues for some use-cases, but for our work classification performance was prioritized.

3.5. Federated Learning and Differential Privacy

For the federated learning setup, we opted for a simple setup consisting of 2 parties, each holding an equal random sample of the total training data. In order to implement the output perturbation algorithm in the federated learning context, we first had to modify the core algorithm to allow for 2 parties to simultaneously learn a model together.

As explained in Section 2.2, the output perturbation algorithm given in Algorithm 1 requires further modifications to be made as the privacy guarantees of the algorithm depend on the exact minimum of the loss function to be found. To mitigate this problem, we introduce a gradient norm bound parameter γ which serves as a stopping condition for the algorithm - an idea that was adapted from [17]. Additionally, the noise distribution was also swapped out from the Laplace distribution to the Gaussian distribution which results in the Approximate output perturbation (AOP) algorithm presented in Algorithm 2 to be a (ε, δ)-differentially private algorithm instead of a $(\varepsilon, 0)$ one.

Algorithm 2: Approximate output perturbation (AOP)

Input: Initial parameters $\theta_0 \in \mathbb{R}^p$, Gradient norm bound γ, Data $\mathcal{D} = \{d_1, ..., d_n\}$, Learning rate η, Privacy parameters (ε, δ), Regularization parameter Λ, Loss function $\mathcal{L}$

Output: Privatized approximate minima $\tilde{\theta}_{approx}$

Function `privatize_output`$(\theta_{approx}, \varepsilon, \delta, n, \Lambda, \gamma)$**:**

  Draw $b \sim N(0, \sigma^2 I_{p\times p})$ where $\sigma = \frac{\left(\frac{n\gamma}{\Lambda}\right)\left(1+\sqrt{2\log\frac{1}{\delta}}\right)}{\epsilon}$

  $\tilde{\theta}_{approx} = \theta_{approx} + b$

  return $\tilde{\theta}_{approx}$

Starting at θ_0, find θ_{approx} s.t. $||\nabla\mathcal{L}(\theta; \mathcal{D})|| \leq \gamma$

$\tilde{\theta}_{approx} =$ `privatize_output`$(\theta_{approx}, \varepsilon, \delta, n, \Lambda, \gamma)$

return $\tilde{\theta}_{approx}$

In the federated learning context for AOP, the training is split into epochs that span across both parties. In each epoch, each party independently trains a model without adding any differential privacy noise based on the dataset available to them. At the end of the epoch, the parameters of the model held by both parties are privatized and combined, then the next set of iterations are run. The algorithm for federated output perturbation is given in Algorithm 3.

Each epoch is (ε', δ')-differentially private following the proof of privacy given in [17]. Therefore by invoking the basic composition theorem (Theorem 2), the federated model training algorithm given in Algorithm 3 that composes T (ε', δ')-differentially private mechanisms where $\varepsilon' = \frac{\varepsilon}{T}$ and $\delta' = \frac{\delta}{T}$ is (ε, δ)-differentially private. Although the advanced composition theorem could have been invoked as well, we only look at a low number of epochs (T) for which the basic composition theorem gives better results.

3.6. *PySyft* Plan Implementation

In practice, the algorithms presented in the previous section are implemented in *PySyft*, with the *privatize_output* function built as a *PySyft plan* which converts the privatization function into a low communication cost tensor operation that can be called by both parties. Ideally, the entire Approximate output perturbation algorithm should be implemented as a plan and sent to the 2 parties. However as explained in Section 2.3, since if and while operations cannot be encoded into a plan and the if operation is essential to act as the stopping condition for the algorithm, the algorithm cannot be encoded into a plan in its entirety.

Algorithm 3: Federated output perturbation

Input: Datasets $\mathcal{D}_{Alice} = \{(x_1, y_1), ..., (x_n, y_n)\}$ and $\mathcal{D}_{Bob} = \{(x'_1, y'_1), ..., (x'_n, y'_n)\}$, Gradient norm bound γ, Learning rate η, Privacy parameters (ε, δ), Number of epochs T, Regularization paramter Λ, Loss function $\mathcal{L}$

Output: Approximate minima θ_{out}

1 Initialize θ_1

2 Set $\varepsilon' = \frac{\varepsilon}{T}$

3 Set $\delta' = \frac{\delta}{T}$

4 **for** $t \in [1..T]$ **do**

5 $\quad \tilde{\theta}_{Alice} = \text{AOP}(\theta_t, \gamma, \mathcal{D}_{Alice}, \eta, (\varepsilon', \delta'), \Lambda, \mathcal{L})$

6 $\quad \tilde{\theta}_{Bob} = \text{AOP}(\theta_t, \gamma, \mathcal{D}_{Bob}, \eta, (\varepsilon', \delta'), \Lambda, \mathcal{L})$

7 $\quad \theta_{t+1} = (\tilde{\theta}_{Alice} + \tilde{\theta}_{Bob})/2$

8 **end**

9 **return** $\theta_{out} = \theta_{T+1}$

Instead, we encoded each iteration of the optimization step (line 5 of Algorithm 2) and the privatize_output function as a plan - a task that presented its own set of challenges. For the iterative step, we found that PySyft's autograd feature does not work with plans well. Therefore, the gradient function had to be manually coded instead of relying on the *loss.backward()* function to automatically generate the gradients. Luckily, since we used a simple logistic regression model, the gradient of the parameters has a simple analytical expression that was coded directly.

For the *privatize_output* function, the main issue arose from the fact that the Laplace and Gaussian distributions available through the *torch.distributions* package were not hooked. This meant that the operation that draws samples from these distributions cannot be encoded into a plan. Luckily, the *torch.rand_like* function that generates uniformly distributed tensors was hooked and could be used. One method for generating random numbers from the normal distribution would be to use the Box-Muller transform [34]. If we had wanted to make the AOP a $(\varepsilon, 0)$-differentially private algorithm, the Box Muller transform would have had to be modified in order to draw the noise variable from the Laplace distribution based on variables drawn from the uniform distribution. However, the *torch.FloatTensor.normal_* function which directly generates a normal distributed tensor was also available and could be used to encode the *privatize_output* function into a plan which was what had to be done.

3.7. Hyperparameter Tuning

For the Federated output perturbation algorithm, there are a few hyperparameters to be tuned such as the gradient norm bound γ, learning rate η, number of epochs T and regularization parameter Λ. As explained previously in Section 3.1, we will be disregarding the privacy analysis for this section as we are mainly focusing on the core training algorithm. Therefore, a grid search is performed to identify the best hyperparameters for the classification task.

4. Results

Following [35], δ is consistently set to be $\frac{1}{n^2} \approx 1 \times 10^{-6}$ and the ε is varied from 10^{-3} to 10 showing various levels of privacy. The accuracies are averaged across 1000 trials so that the accuracies are robust. The resulting change in accuracy to various levels of ε can be seen in Figure 3. Unsurprisingly, for the differentially private learning, the best number of epochs, T was 1 as any increase in the number of epochs resulted in each epoch being significantly less accurate. We see that the federated output perturbation algorithm, even in the non-private setting where no noise is added does slightly better (96.6%) than the simple sklearn training algorithm used previously to validate the model and feature selection (96.0%). Under the private setting, the algorithm performs well under reasonably set privacy parameter of $\varepsilon = 1$ with an accuracy of 92.1% - an accuracy drop of 4.5%. Compared to prior work done by Beguier et. al. [5], which used the DP-SGD algorithm provided by the Opacus library, this is a smaller accuracy drop from the non-private version. Whilst their non-private accuracy is higher than ours at 99.5%, for slightly relaxed privacy parameters ($\varepsilon = 1, \delta = 1 \times 10^{-5}$), their accuracy drops by 6% to 93.5% whereas our accuracy drop is only 4.5%. This suggests that our custom DP implementation is considered more privacy-preserving than the off-the-shelf implementations provided in DP libraries.

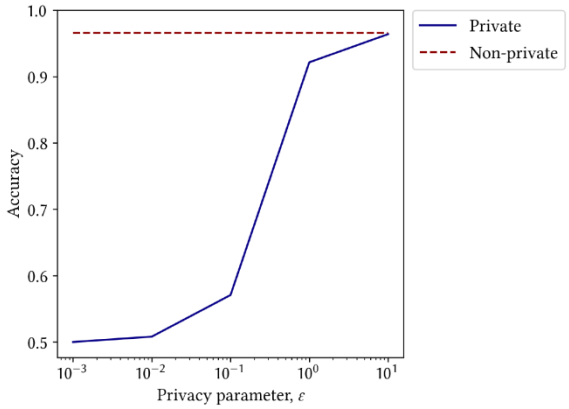
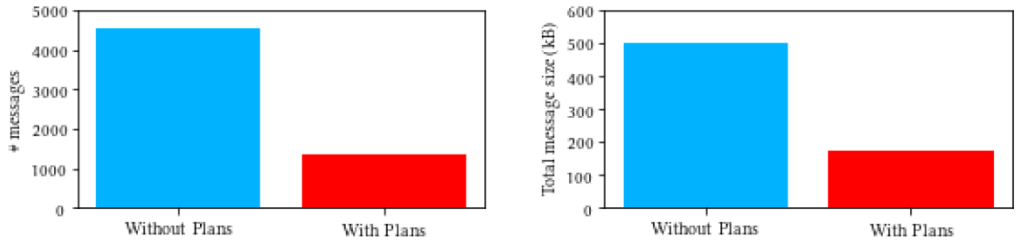


Fig. 3: Prediction accuracy vs privacy parameter ε

In terms of communication cost, by trapping the messaging API provided by PySyft which is used by the parties to communicate with each other, we tracked the number of messages exchanged between the parties. We found that the implementation of plans does in fact heavily reduce the number of messages exchanged as can be seen from the 30.0% reduction in Figure 4a. Furthermore, the total size of the messages exchanged between the parties also significantly went down by 35.4% when plans were implemented as can be seen from Figure 4b. These results reaffirm the fact that plans are indeed useful primitives in the federated learning context.



(a) Number of messages exchanged (b) Total message size (kB)

Fig. 4: Communication cost comparison between PySyft plan implementation and naive PySyft implementation

5. Conclusion

In this paper, we have looked at the challenges surrounding the incorporation of custom differentially private algorithms into federated learning workflows. Mainly focusing on the PySyft library, we show how key primitives such as plans can be leveraged to heavily reduce the communication cost in terms of the number and size of messages exchanged between parties participating in federated learning. However, at the same time, we show that the PySyft library, as it stands currently, imposes a multitude of restrictions that make it hard for custom DP algorithms to be incorporated into it. Nevertheless, we detail how some of these challenges can be overcome and ultimately present a method to integrate the output perturbation algorithm into a PySyft workflow. When averaged over 1000 trials, we achieve a reasonably good accuracy of 92.1% for privacy parameters ($\varepsilon = 1, \delta = \frac{1}{n^2} \approx 1 \times 10^{-6}$) compared to the non-private accuracy of 96.6%. Furthermore, by incorporating plans, we show that the communication cost in terms of the total number of messages and total message size reduces significantly by up to 35% while simultaneously showing the challenges of doing so.

6. Future Work

Our work exposes much of the challenges imposed by the PySyft library that disincentivizes the mainstream adoption of custom differentially private algorithms into federated learning workflows. Possible areas of exploration would be to propose fixes to these challenges in the PySyft library such that more advanced DP algorithms and models can be integrated within the library. We believe that such integration is necessary and will greatly benefit more research conducted in the field of privacy-preserving analytics.

7. Acknowledgements

This research is supported by the Institute for Infocomm Research, A*STAR Research Entities under its RIE2020 Advanced Manufacturing and Engineering (AME) Programmatic Programme (Award A19E3b0099). Dominic's work is supported by the Agency for Science, Technology and Research.

8. References

[1] M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, et al., "Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning," *Nature medicine*, vol. 8, no. 1, pp. 68–74, 2002.

[2] A. C. Tan and D. Gilbert, "Ensemble machine learning on gene expression data for cancer classification," 2003.

[3] X. Jin, A. Xu, R. Bie, and P. Guo, "Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles," in *International Workshop on Data Mining for Biomedical Applications*, pp. 106–115, Springer, 2006.

[4] E. Glaab, J. Bacardit, J. M. Garibaldi, and N. Krasnogor, "Using rule-based machine learning for candidate disease gene prioritization and sample classification of cancer gene expression data," *PloS one*, vol. 7, no. 7, p. e39932, 2012.

[5] H. Salem, G. Attiya, and N. El-Fishawy, "Classification of human cancer diseases by gene expression profiles," *Applied Soft Computing* , vol. 50, pp. 124–134, 2017.

[6] M. Maniruzzaman, M. J. Rahman, B. Ahammed, M. M. Abedin, H. S. Suri, M. Biswas, A. El-Baz, P. Bangeas, G. Tsoulfas, and J. S. Suri, "Statistical characterization and classification of colon microarray gene expression data using multiple machine learning paradigms," *Computer methods and programs in biomedicine*, vol. 176, pp. 173–193, 2019.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[8] PySyftTeam, "Syft: A library for computing on data you do not own and cannot see."

[9] L. Song, R. Shokri, and P. Mittal, "Membership inference attacks against adversarially robust deep learning models," in *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, IEEE, 2019.

[10] C. Dwork, A. Roth, et al., "The algorithmic foundations of differential privacy.," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211– 407, 2014.

[11] S. Meftah, B. H. M. Tan, C. F. Mun, K. M. M. Aung, B. Veeravalli, and V. Chandrasekhar, "Doren: Toward efficient deep convolutional neural networks with fully homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3740–3752, 2021.

[12] J. Wang, C. Jin, S. Meftah, and K. M. M. Aung, "Popcorn: Paillier meets compression for efficient oblivious neural network inference," 2021.

[13] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, and H. V. Poor, "User-level privacy-preserving federated learning: Analysis and performance optimization," *IEEE Transactions on Mobile Computing*, 2021.

[14] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[15] OpacusTeam, "Opacus."

[16] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization.," *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.

[17] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang, "Towards practical differentially private convex optimization," in *2019 IEEE Symposium on Security and Privacy (SP)* , pp. 299–316, IEEE, 2019.

[18] C. Beguier, J. O. d. Terrail, I. Meah, M. Andreux, and E. W. Tramel, "Differentially private federated learning for cancer prediction," arXiv preprint arXiv:2101.02997, 2021.

[19] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

[20] T. Ryffel, "Pysyft + opacus: Federated learning with differential privacy."

[21] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečn`y, S. Mazzocchi, H. B. McMahan, et al., "Towards federated learning at scale: System design," arXiv preprint arXiv:1902.01046, 2019.

[22] C. G. A. Network et al., "Comprehensive molecular portraits of human breast tumours," *Nature*, vol. 490, no. 7418, p. 61, 2012.

[23] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, et al., "Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, no. 6769, pp. 503–511, 2000. Differentially private, Federated Learning for Tumour Classification.

[24] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in bioinformatics*, vol. 2015, 2015.

[25] M. A. Hall, "Correlation-based feature selection for machine learning," 1999.

[26] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning* (ICML-03), pp. 856–863, 2003.

[27] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, 2005.

[28] M. S. Al-Batah, B. M. Zaqaibeh, S. A. Alomari, and M. S. Alzboon, "Gene microarray cancer classification using correlation based feature selection algorithm and rules classifiers," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 15, no. 08, pp. 62–73, 2019.

[29] K. Kavitha, A. Gopinath, and M. Gopi, "Applying improved svm classifier for leukemia cancer classification using fcbf," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 61–66, IEEE, 2017.

[30] M. Akhand, M. A. Miah, M. H. Kabir, and M. H. Rahman, "Cancer classification from dna microarray data using mrmr and artificial neural network," *Cancer*, vol. 10, no. 7, 2019.

[31] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 94, 2018.

[32] S. Ram ́ırez-Gallego, I. Lastra, D. Mart ́ınez-Rego, V. Bol ́on-Canedo, J. M. Ben ́ıtez, F. Herrera, and A. Alonso-Betanzos, "Fast-mrmr: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data," *International Journal of Intelligent Systems*, vol. 32, no. 2, pp. 134–152, 2017.

[33] Y. Zhang and Z. Zhang, "Feature subset selection with cumulate conditional mutual information minimization," *Expert systems with applications*, vol. 39, no. 5, pp. 6078–6088, 2012.

[34] G. E. P. Box and M. E. Muller, "A Note on the Generation of Random Normal Deviates," *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610 – 611, 1958.

[35] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. Naughton, "Bolt-on differential privacy for scalable stochastic gradient descent-based analytics," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1307–1322, 2017.