Enhancing Privacy in Federated Learning with Local Differential Privacy for Email Classification*

Sascha Löbner[1 [0000-0001-9164-1919]], Boris Gogov[1 [0000-0001-7675-5694]], and Welderufael B. Tesfay[1 [0000-0002-1087-2019]]

Chair of Mobile Business & Multilateral Security, Goethe University, 60323 Frankfurt am Main, Germany {sascha.loebner, welderufael.tesfay}@m-chair.de, bgogo@protonmail.com

Abstract. With federated learning, information among different clients can be accessed to train a central model that aims for an optimal use of data while keeping the clients' data local and private. But since its emergence in 2017, several threats such as gradient attacks or model poisoning attacks against federated learning have been identified. Therefore, federated learning cannot be considered as stand alone privacy preserving machine learning technique. Thus, we analyse how and where local differential privacy can compensate for the drawbacks of federated learning while keeping its advantage of combining data from different sources. In this work, we analyse the different communication channels and entities in the federated learning architecture that may be attacked or try to reveal data from other entities. Thereby, we evaluate where local differential privacy is helpful. Finally, for our spam and ham email classification model with local differential privacy, we find that setting a local threshold of F1-Score on the clients' level can reduce the consumption of privacy budget over several rounds, and decrease the training time. Moreover, we find that for the central model a significantly higher F1-Score than those set on the local level for the clients can be achieved.

Keywords: Federated Learning · Differential Privacy · Phishing and Spam Prevention.

1 Introduction

To achieve a good model performance of machine learning (ML) applications, large and up to date datasets are required. Especially for time critical applications, such as spam filters or intrusion detection, up to date ML models are crucial. In these fields, a co-evolutional problem exists so that ML is used in an arms race between attackers and defenders [5]. This is where the adaptability of Federated Learning (FL) [12, 27] is a key feature to improve existing spam and ham classification

* This work was supported by the European Union's Horizon 2020 Research and Innovation Program through the Project CyberSec4Europe under Agreement 830929.

approaches that are based on spam mail detection. While in the past, data among clients was often treated as separated islands or required significant effort for anonymisation, FL aims to build a joint, and up to date central model, while simultaneously keeping the data of each client private. Thereby, clients train their models locally and share only the models' gradients with a central server that calculates a central model that is re-distributes to the clients [27]. The result is an up to date spam and ham classification model that can take the data of many separated databases into account, without sending the data to a central location [21]. Thus, the core of FL for spam and ham classification is the gradient update that is used to keep local data private. But recent research has shown that it is possible to elicit training data and labels from the models gradients [29]. Thus, FL as a stand alone technique is not sufficiently privacy preserving in general [9]. Therefore, an additional layer of privacy is required when aiming to make use of the benefits of FL as a privacy preserving machine learning (PPML) technique.

Especially for content based spam filters that utilise ML to classify emails in spam or ham, performance, communication overhead, and required computational power are the key feature. When clients' data is used to update a model, it is important that the data is kept private. A common techniques for PPML is Differential Privacy (DP) [10, 14]. Although some research that combines FL and DP has been carried out [23] , to the best of our knowledge no model that enriches FL with local DP (LDP) for private email classification exists.

With the insights generated by this work we contribute to overcome the privacy shortcomings in FL by extending the architecture with LDP. This is relevant because with FL spam mail detection can become more up to date while the privacy risk for each client is reduced at the same time. We use qualitative evaluation criteria to identify, against which threads and entities LDP can help to overcome the weaknesses of plain FL. Thus, we identify where additional steps are necessary to build a privacy preserving spam mail detection model. Moreover, we aim to provide an idea of how a FL and LDP model can be set up for spam and ham email classification and how the parameters interact with each other. Finally, we propose a local F1-Score threshold for the clients model to make FL in combination with DP even more efficient. Our contribution is as follows:

R1 Improved Privacy: **With requirement 1, we aim to identify which privacy weaknesses of FL can be overcome with the addition of LDP for application of spam and ham classification.** For the application of spam and ham detection we have evaluated privacy of client, privacy of the model and privacy of results (for details see sec. 4.1). We find an increase of privacy especially for global privacy when clients can trust the central server. Within local privacy where data is required to be kept private against all other parties, we find a need of improvement in the privacy of model and privacy of results.

R2 Efficiency: **With requirement 2, we aim to identify how the combination of LDP and FL can become more efficient what is important for an implementation in practice.** For the application of spam and ham classification we introduce a local F1-Score threshold of 0.85 on the clients' models to overcome the negative impact of LDP on the computation time and effort. We find that this approach still achieves an F1-Score of 0.94 for a noise multiplier of 0.99 and an ϵ value of 23 after 20 federated rounds for 10 clients, while simultaneously increasing performance and efficiency compared to a noise multiplier of 0.7.

2 Attacks Against Federated Learning

In this section we will have a closer look at attacks that are designed specifically against FL. In general, the attacks can be separated into those that aim to reveal user data and those that aim to manipulate the model's predictions.

Gradient Attacks: Wei et al. [26] assert that the sharing of gradients in a FL model can be a privacy risk because of gradient leakage attacks that aim to reveal a worker's training data. If the central server is honest, an adversary could still be able to intercept parameter updates before they reach the central server. Also, an adversary might be able to access locally saved data about the model, e. g. stored gradients, on a compromised worker without seeing the training data. They test the mitigation strategies of gradient perturbation by adding noise and the gradient squeezing with controlled local training iterations. They find that both methods disturb the quality of their privacy leakage attack against the workers. Geiping et al. [9] show for trained deep networks that it is possible to reconstruct high-resolution images from the gradients of models that were trained with deep learning. Moreover, it is possible to completely reconstruct the input to any connected layer of the neuronal network, using gradient inversion attacks. Zhu et al. [29] show how to obtain training data and respective labels after a view rounds of iterations with a deep gradient attack. They obtain images pixel wise and sentences token wise matching texts from the gradients. With LDP for image data, they find that a defensive level of noise has a significant impact on accuracy.

Data Poisoning Attacks: Tolpegin et al. [22] show that data poisoning attacks can significantly significantly reduce the accuracy metrics, e. g. accuracy and recall, of a federated classification model even with only a small number of malicious workers. In such an attack, malicious workers poison the central model by sending gradients trained on mislabeled data. Awan et al. [3] further differentiate between an untargeted attack that aims to reduce the overall testing accuracy of the FL model and targeted attacks that aim to cause misclassification in a certain class. Possible countermeasures proposed are reputation based learning [3] or the identification of malicious participants utilising e. g., loss or error functions [22] or gradient outlier detection [26]. Moreover, Dong et al. [7] propose a solution to overcome the trust problem in byzantine attacks with a dishonest majority.

3 Related Literature

In this section, we present literature related to phishing and spam detection with FL. We also consider alternative machine learning approaches for spam detection as well as other FL models that combine DP with FL.

We identify a view studies that already implement phishing detection with FL. Makkar et al. [15] propose a model for internet attack utilising FL for image spam detection. Moreover, they see especially phishing email detection as a promising application for FL because FL can enhance privacy resulting in an increase of willingness of users to participate in data sharing and thus increase the model's accuracy. Thapa et al. [21] already implemented a FL based phishing mail detection model. Their contribution mainly focuses on different data distributions among clients, different numbers of clients, the scalability of the model and lastly the communication overhead. They find that an increased number of workers influences the convergence of the model's accuracy slightly negatively. Moreover, they point out that there exists a trade off between communication overhead and privacy. Furthermore, they assert that transfer learning can increase the convergence of the model's accuracy. The integration of other privacy preserving techniques such as Homomorphic Encryption (HE) or DP are identified as future work to further improve the model's privacy-protection.

Also other approaches for spam email detection with ML exist. Following the Text Retrieval Conference (TREC), spam can be defined as any indiscriminately sent, unsolicited email. The authors find that most of the techniques for email spam filtering are based on reputation, textual content or multimedia content, where ML belongs to the group of content based filters [5]. In their work, they identify Naive Bayes, Support Vector Machines and Decision Trees as technologies of choice. Also, Dada et al. [6] analyse different ML approaches and find that most state-of-the-art techniques for email spam filters cannot learn in real-time.

FL is combined with differential privacy also by other researchers. Geyer et al. [10] focus on the probability of attacks against a FL model performed by any entity participating in the FL architecture. They try to hide the contribution of the workers during the training by using DP on the local client data. In their results, they assert that LDP can reach high accuracy if the group of participating workers is high enough. Basu et al. [4] propose a contextualised transformer based text classification model based on FL, including DP utilising financial text data. They find that the performance increases with an increasing ϵ and decreasing noise. Wei et al. [25] also introduce a (ϵ, δ)-differential privacy based model that adds noise to the client's parameters. They calculate a convergence bound on the loss function of their FL model. First they find, that with increased privacy the accuracy is reduced, second that with the number of workers the convergence performance increases, and third that an optimal number of maximum aggregation times exists, with regard to convergence performance and level of protection.

To the best of our knowledge, no model exists that combines FL and DP for a spam and ham email classification problem.

4 Methodology

In this section, we will take a closer look at the requirements of this paper, which aims to contribute to the current state of research in spam and ham email classification. Furthermore, we present our metrics to evaluate our requirements.

4.1 Requirement Elicitation

R1: Improved Privacy: With this requirement, we aim to identify which privacy weaknesses of FL can be overcome with the addition of LDP. To fulfil R1 we have derived qualitative evaluation criteria that represent the different levels at which privacy can be leaked.

R2: Efficiency: A combination of LDP and FL is only useful in practice if it is efficient. We evaluate efficiency by comparing different quantitative evaluation parameters, e. g., run time, the evaluation metric F1-Score, or the achieved ϵ value. On the one hand, a high F1-Score is required for a reliable spam and ham classification, on the other hand, the clients' data needs to be kept private while at the same time the costs of communication and computation are kept low.

4.2 Evaluation Criteria

Qualitative metrics of privacy: To evaluate R1 we introduce a framework of qualitative privacy metrics that is provided in Figure 1. In general, Tanuwidjaja et al. [20] differentiate between three different qualitative metrics of privacy in PPML from which we elicit the following three metrics for our approach:

First, we investigate the privacy of the clients who share their gradients with the central server and use the spam filter to keep their emails private. Neither central server nor other clients should be able to reveal data from any clients' email. If no entity can access the clients data, this is defined as local privacy. Local privacy is of great importance if the central server is malicious. Global privacy is the protection against third parties, except the central server [12].

Second, the privacy of the model relates to the model that is built by the central server. No party, including the central server, should know the logic of the model. Tanuwidjaja et al. [20], and Yang et al. [27] propose HE or Secure Multiparty Computation (SMPC) to overcome this thread. With regard to FL, we understand privacy of the model as the precise weights of a client's local model.

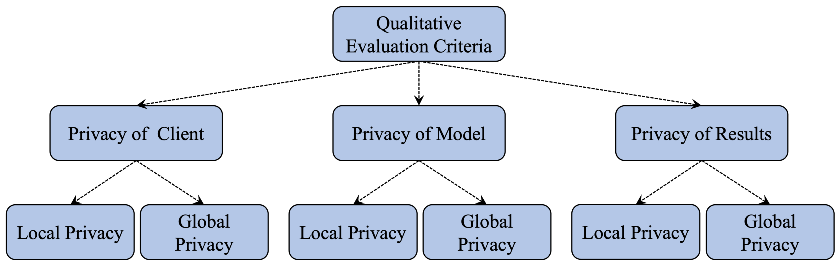
Third, the protection of the privacy of the results specifies that no entity should be able to know the classification results for a certain instance [20]. Again, on a local privacy level, this includes all entities participating in the architecture, such as central server, and clients as well as third parties from outside.

Quantitative evaluation metrics: As evaluation metric for our machine learning model, we have chosen the F1-Score that is defined as the harmonic mean of precision and recall and that prevents the model from hiding false positives or false negatives [19]. Especially false positives can cause much worse problems and costs in spam email detection [5]. To evaluate the achieved level of privacy we will use the ϵ value as it is derived by the opacus implementation [28]. Moreover, we take also training time and number of required federated rounds into account.

5 Approach

In this section, we describe the data set, local data preparation and our model.

Fig. 1: Qualitative evaluation criteria, elicited from [20, 12, 27].



Data and Computation: For our model, we have use the improved Enron dataset 2020 as provided by the Natural Language Processing Group of Athens University of Economics[1]. Their version already contains spam and ham labeled data, with 33,722 messages out of which 17,171 are labeled as spam (50.9 %) and 16,545 messages are labeled as ham (49,1 %). The ham messages in the Enron dataset are benchmarked to protect the original messages and their content. From a statistical point of view, benchmarked messages are in their sequence of tokens very close to the original ones [17]. Our model runs on a CPU: Intel 8700K (6 cores), RAM: 32GB, and GPU: nVidia 1080Ti (11GB VRAM, CUDA 11.2).

FL and LDP architecture: In Figure 2, we provide the architecture of our privacy preserving spam mail detection demonstrator including the FL architecture and the local DP update process. For the spam and ham email classification problem, our architecture follows the horizontal FL structure [27].

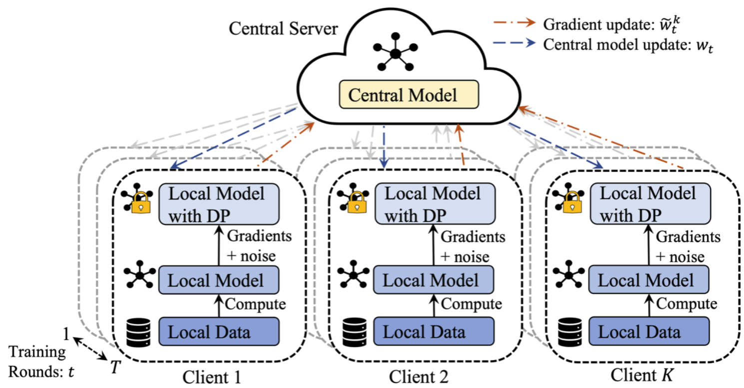
Before starting Algorithm 1, data is collected on the local devices. In our demonstrator, we simulate the devices with different independent clients on one server. Therefore, we do not take data exchange problems caused by different devices or applications into account. We assume that the user who is using an email service has labelled the spam email by putting them into the spam folder. The emails in the spam folder we define as spam. All other emails are treated as ham. Thus, we work with already labelled datasets, the labelling of data does not take place but is provided for a better understanding of our process. McMahan et al. [16] have introduced a federated averaging algorithm that we used as a starting point for our model that we will describe in the following (see Alg. 1):

First, the central server initialises the starting weights w_0 for the global model, the number of clients K to which the data will be distributed and the number of training rounds T after the algorithm stops (see line 2). In practice, the stopping criteria can be different, e.g. F1-Score, for testing we use T.

Second, for each training round t all clients $k \in K$ do in parallel the computation of the local model `client_update`(k, w_t) (see line 5). McMahan et al. [16] show an extension in the selection of k by using a random subset S_t of m clients. For testing purposes, we have not implemented this step in our model.

[1] `http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/index.html`

Fig. 2: Privacy preserving architecture, extending FL [27] with LDP.



Third, the function `client_update`$(k, w_t, \epsilon, \delta)$ takes the weight w from the current global model for each client k and the DP parameters ϵ and δ as input (see line 10). In general, ϵ can be interpreted as a parameter for privacy leakage we aim to keep as low as possible and δ as the probability of accidental privacy leakage. In our model, in parallel all clients perform the local Long Short-Term Memory (LSTM) model with the predefined local epochs E and the batch size B. $\eta \nabla l(w_t; b)$, thereby describing the optimisation function of the neuronal network. Later, we will extend the client date by adding a local threshold that stops the iteration over E if $F1-Score \geq threshold$ is satisfied. The F1-Score is calculated based on the local training data and the local model. The threshold is a tradeoff between privacy and achievable accuracy. Note that the accuracy in epoch i can be lower than the *threshold* but bigger in epoch $i+1$.

Fourth, we use (δ, ϵ) DP to add noise to the gradients. With our approach, we follow the definition of Dwork et al. [8] for (δ, ϵ) privacy:

$$\forall\ x: \ \Pr[M(D) = x] \leq \exp(\epsilon) \cdot \ \Pr[M(D') = x] + \delta.$$

This statement implies that for assuming a dataset D' that differs in only 1 entry from dataset D, the probabilities of the x never differs more than $\exp(\epsilon)$ from each other. In our implementation, we use the Python library Opacus, an extension of Differential Privacy Stochastic Gradient Descent (DP-SGD) [1] that ensures that the condition given above is satisfied for every model update [28]. In general, in DP-SGD gradients are computed sample wise, their $l2$ norm is clipped and they are aggregated as batch gradients, adding Gaussian noise. While we have only indicated this procedure in line 15, more details can be found in [25, 1].

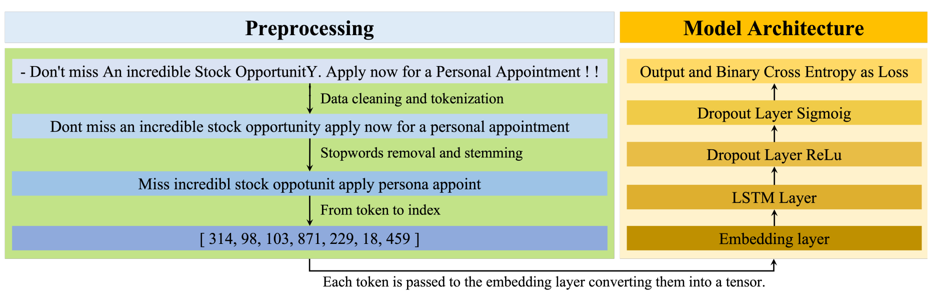
Fifth, the central server uses the received gradients in round t from each client k and computes a central model using the weighted average as aggregation function (see line 6). The iteration over t continues until T is reached.

Data pre-processing and model structure: In Fig. 3 we show the different steps of local pre-processing on each client. The example shows how a spam message

Algorithm 1 $k \in \{1, 2, ..., K\}$: $K \coloneqq$ # clients; $B \coloneqq$ local minibatch size, $E \coloneqq$ # local epochs, $t \in \{1, 2, ..., T\}$: T training rounds, and $\eta \coloneqq$ learning rate. [16]

`central_server:`
initialize w_0, K, T
for each round $t = 1, 2, ...$ **do**
    for each client $k \in K$ **in parallel do**
        $\tilde{w}^k_{t+1} \leftarrow$ `client_update`$(k, w_t, \epsilon, \delta)$
        $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \tilde{w}^k_{t+1}$
`client_update(`k, w_t, ϵ, δ`)`
$B \leftarrow$ (split P_k into batches of size B)
for each local epoch i form 1 to E **do**
    for batch $b \in B$ **do**
        $w^k_{t+1} \leftarrow w_t - \eta \nabla l(w_t; b)$
        $\tilde{w}^k_{t+1} \leftarrow w^k_{t+1} + noise$
return $\tilde{w}^k_{t+1}$ to server

Fig. 3: Steps of pre-processing local data on each client.



is altered during the pre-processing from data cleaning to indexing. After the pre-processing, the data is transferred to the embedding layer, where each word is mapped to its proper vector. We chose the LSTM architecture since it performs well on binary text classification [11], e.g., Amjad et al. [2] compared text classification methods with the result that LSTM slightly outperforms NB, MNN and CNN. Finally, we use dropout layers to prevent the model from overfitting. The model is implemented using PyTorch [18] and published via GitHub[2].

6 Results

In this section, we present the results of our model that combines LDP and FL for the application of privacy preserving spam and ham classification.

Figure 4a shows the test results of our model with regard to different levels of noise. The green line represents the values without any level of noise and is therefore without LDP. We will use this model as the base model. In Table 1

[2] `https://github.com/supaboy1999/federated-spam-ham`

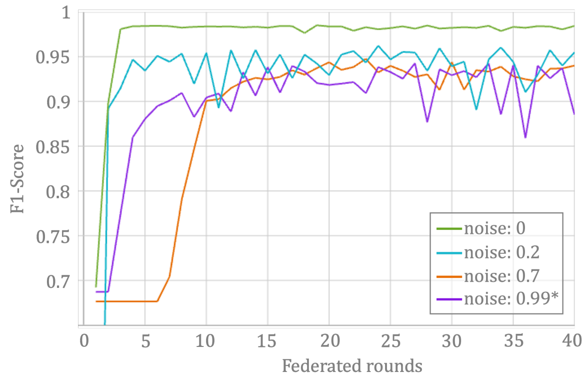
	Federated Rounds								
	5		10		20		40		
Noise	ϵ	F1	ϵ	F1	ϵ	F1	ϵ	F1	Time total
0	n/a	0.9843	n/a	0.9838	n/a	0.9837	n/a	0.9845	137 min.
0.2	322	0.9344	580	0.9544	1201	0.9294	2185	0.955	148 min.
0.7	24	0.8804	29	0.9044	44	0.9185	57	0.8853	152 min.
0.99*	12	0.6763	17	0.9007	23	0.9438	34	0.9401	144 min.

* Additional threshold at F1-Score of 0.85

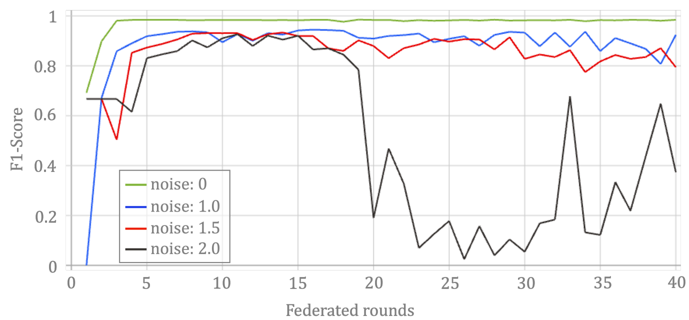
Table 1: Privacy budget (ϵ) and F1-Score of the central model after t rounds of federated training and with different levels of noise

we show the precise F1-Score scores after 5, 10, 20 and 40 rounds for the noise multiplier set to 0.2, 0.7 and 0.99. For the noise level of 0.99 we have implemented a local threshold of the F1-Score of 0.85 to increase the performance. More precisely, a client stops running a model locally if the local F1-Score of 0.85 is achieved. This is done to consume less privacy budget in each training round, to prevent overfitting, and to decrease the training time. In Table 1 also the noise level and respective ϵ value are shown as well. Where the noise multiplier is 0 and no DP is used, we have labeled the ϵ value as not applicable (n/a). A first result that can be noted is that for all test runs, the achieved F1-Score fluctuates within a certain range. With increasing amounts of noise added to the gradients the fluctuation increases as well. For the green line without any noise, the fluctuation is the lowest and in a range from 0.970 (10 rounds) to 0.986 (40 rounds). Up to 40 federated rounds, the F1-Score is increasing overall. The blue line shows a noise multiplier of 0.2. It can already be noticed that the blue line in Figure 4a is much more fluctuating compared to the green line. The blue line is still converging fast and achieves a F1-Score of 0.955 after 40 rounds, 0.0295 lower than the green line. The orange line converges much later at federated round 5 and exhibits the worst F1-Score of 0.8853. More interestingly, the purple curve starts stronger fluctuation after round 32. Taking a closer look at Table 1 we can see that with more federated rounds, the ϵ increases significantly. This happens because with every additional federated round, more data is leaked. Therefore, it is of interest to have a limited number of training rounds to reduce data leakage.

While we can only observe a trend in Figure 4a, the phenomenon becomes clearer when increasing the level of noise (see Figure 4b). With an increasing noise multiplier, the model converges later and the F1-Score decreases. While at round 11 the noise multiplier of 1, 1.5 and 2.0 seem to converge equally, after 15 rounds, a significant loss in the accuracy metric of F1-Score can be noticed. At this stage, the model shows an F1-Score of 0.982 for the noise multiplier 0, an F1-Score of 0.94 for 1.0, an F1-Score of 0.92 for 1.5, and an F1-Score of 0.909 for 2.0. After 20 rounds the F1-Score of the black line, with a noise multiplier of 2.0 has significantly dropped. While there is a trend toward higher fluctuation and a slight decrease in performance, the blue line with a noise multiplier of 1.0 and the red line with a noise multiplier of 1.5 still provide good results.



(a) Levels of noise: (0, 0.2, 0.7, 0.99*).



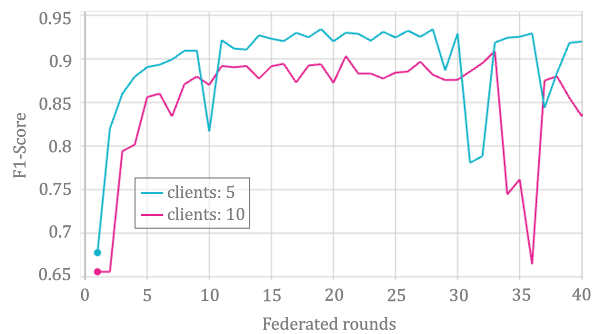
(b) Levels of noise: (0, 1.0, 1.5, 2.0).

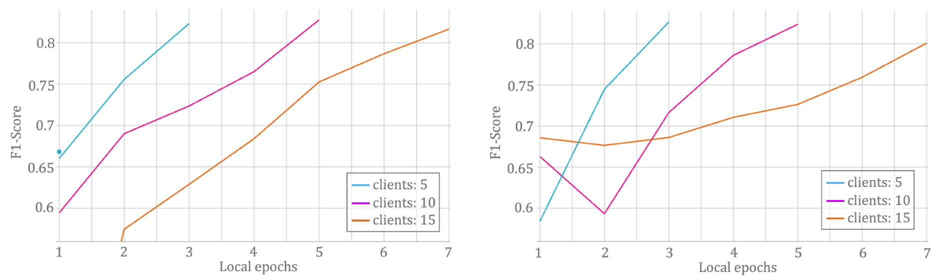
Fig. 4: FL with LDP, 10 clients, showing F1-Score of central model after t rounds.

In Figure 5, we investigate the effect of varying numbers of clients. In the experiment, we always distributed the testing set with 33,716 among all clients. Thus, we had 6148 instances per client for the 5 clients test and 3072 instances for the 10 clients test. We can see that the F1-Scores for both tests stabilise after 11 federated rounds and start to drop after 30 rounds. Therefore, the most privacy preserving number of federated rounds in both cases lies between 11 and 20 federated rounds. In Table 2, it can be observed that the F1-Score of the central model is highest after 20 rounds. Taking a closer look at the ϵ values, epsilon is constantly increasing but until 40 rounds it is significantly lower than without the local F1-Score threshold. Comparing the ϵ values of 5 and 10 clients, the noise for 5 clients where the data sets are bigger is much lower. Therefore, in a real world implementation, it is important to set a limit of instances per client in each federated round, to not consume unnecessary privacy budget.

In Figure 6 we show the local epochs and the F1-Score of two randomly selected clients that have participated in the same training as shown in Figure 5. In addition, we also provide the result for 15 clients in Figure 6 where each client

Fig. 5: FL combined with LDP and different number of clients (5, 10), local F1-Score threshold 0.8, F1-Score of the central model after t rounds.





(a) Random client 1.

(b) Random client 2.

Fig. 6: Local F1-Score after t fed. rounds, showing the epochs of 2 random clients.

Federated Rounds:		5		10		20		40	
Clients	*Noise*	ϵ	F1	ϵ	F1	ϵ	F1	ϵ	F1
5	0.99*	5.746	0.856	7.778	0.870	10.58	0.920	11.89	0.920
10	0.99*	7.367	0.891	10.12	0.817	12.95	0.873	25.71	0.8343

* Additional threshold at F1-Score of 0.80

Table 2: Privacy budget (ϵ) and F1-Score of the central model after t rounds of FL, noise level 0.99, local F1-Score threshold of 0.8 and clients $k = 5$; $k = 10$.

had 2048 instances for training. While the local model with 5 clients reaches the F1-Score threshold of 0.8 very fast, 10 and 15 clients take much more epochs. This implies that with more data, e. g. 6000 instances, the model is more efficient. It can also be observed that the accuracy is slightly above the threshold of 0.8. We have chosen 0.8 because we see this value as the best compromise between privacy and security. This happens because in our current implementation, the training stops if the F1-Score is equal or above the threshold in a certain epoch.

7 Discussion

In this section, we will evaluate our model against the elicited requirements, point out the impact of our results and illustrate future work.

R1 Improved Privacy: In Table 3, we show at which level, with regard to the qualitative evaluation criteria derived in Section 4.2, LDP can help in our approach to decrease the risk of revealing private data. Overall, we see an increase in privacy protection in comparison to plain FL by adding LDP to the FL model on the clients' level. The architecture of FL with in allows all clients' data to be kept local but does not protect against attacks. With our approach of using LDP on the gradients of the local model in combination with setting a threshold to the local F1-Score clients' private data is more likely to be protected. This is also reflected by the ϵ values presented in Table 2. Therefore, we see local privacy and global privacy enhanced for the clients' data.

Privacy of client we see improved for both, global and local privacy. Since the global model is trained on the gradients of the clients, the protection with regard to global privacy should be at a minimum as good as for the local clients. With global privacy, the probability that a certain client is attacked by another clients is reduced, due to the larger group and weighted central model gradients.

Privacy of model is in brackets for local privacy because the local privacy enhances if the ϵ is kept on a low level. But it has to be ensured that the central server cannot learn from the gradient updates over several federated rounds. Scenarios might exist, where the highest F1-Score possible, has to be achieved, thus, a lower ϵ is chosen. Therefore, we indicate this risk with brackets. For global privacy, we do not see this issue and have indicated an increase in privacy.

Privacy of results ensures that the classification result into spam or ham for a certain instance in the data is not revealed to any other party [20]. We evaluate the privacy of results equally to the privacy of model because again, over several federated rounds, the risk exists that the central server learns about the model. Learning about a certain client's model is much easier from for the central server compared to clients because clients are not obfuscated in the crowd. Therefore, we put the local privacy in brackets and evaluate global privacy as achieved.

To further increase the privacy of the central model against attacks from third parties, the implementation of DP on the gradients of the central model becomes important. This would also further lower the risk of an attack performed by another client because of the additional DP layer. Although, no global DP is implemented in our model, LDP already reduces the risk of gradient leakage.

With regard to gradient attacks, LDP is likely to protect the clients' data. Especially with introducing a local threshold, the noise level can be increased while ϵ is kept low over several rounds. Additional steps, e.g., obfuscating the clients' data in a bigger crowd to complicate the linkage of several training rounds of a certain client can help to protect their data. Also, HE and SMPC are suitable to extend our model, to solve the problem of dishonest central servers [27].

With regard to data poisoning attacks, LDP does not help because it does only protect the clients' privacy. Countermeasures [22], might be less effective

Privacy	Privacy of Client	Privacy of Model	Privacy of Results
Global Privacy	✓	✓	✓
Local Privacy	✓	(✓)	(✓)

Table 3: Evaluation of qualitative privacy metrics. In global privacy, clients trust the central server while local privacy aims to keep data private against all parties.

because label flipping is less likely to be detected. Therefore, further mitigation strategies, e.g., gradient outlier detection [26], are required.

R2 Efficiency: The decrease of model performance caused by noise was already foreseen as also noted in other studies (see [4]). More interesting is that the model with LDP converges later, which has a negative impact on computation time and effort. But with regard to a spam filter the usability on different types of devices is essential. Especially mobile devices, e.g., smartphones and laptops, exhibit a limited battery charge and the computational power might also be required for more urgent or important tasks. Therefore, an increase of federated rounds has a negative impact on the implementation for such devices. Especially as users may not evaluate privacy as a key feature they tend to use a lower privacy level [13].

To keep the computational power required for training low, we propose to introduce a local threshold for the F1-Score as an additional step, so that the local training stops when the defined F1-Score (e.g. 0.8) is reached. We have shown in our results (see table 2) that even with a low threshold for the F1-Score of 0.8, combining several local gradients will result in a high F1-Score for the central model. A small drawback of this method is that more federated training rounds are required until the model converges, which increases the communication overhead a little. Moreover, the local threshold might be exceeded in the epoch, when the preset F1-Score is reached. To overcome this issue and to guarantee the preset threshold the model that was trained 1 epoch earlier could be used instead. Besides these drawbacks, we provide a solution for FL spam email detection that achieves a high accuracy by using simultaneously a high noise level with an acceptable ϵ value. Especially for local devices, this can become a problem since the computation might take too much power and makes the device unusable.

Impact: For companies the ability to set different levels of privacy is of special interest, especially if certain departments require stricter privacy settings than others. E.g., a customer service does not want to be too restrictive to reduce false positives so that customers messages do not get lost. On the other hand, banks want to be more restrictive, as they are often attacked and need a higher security level, while false positives are more accepted in view of the risks. While internal privacy in a company is important, the sharing of gradients among companies is also of interest. As we have shown, even if gradients of a simplified model are shared, the combined model can exceed the accuracy of the participating companies. In a real world implementation, all entities e. g. private users, companies and governmental institutions could share their data to build the most up to date spam filter possible. By implementing our proposed approach, we foresee a positive impact on data protection rights and the sovereignty of clients. Also other domains e.g., spoofing detection can gain useful insights from our approach.

Limitations: Since the email text data the model is built on is exclusively in English, characteristics from other languages are not considered. This can lead to structural bias in the model and cause misclassifications. Also, the model cannot deal with data poisoning attacks, thus, additional mitigation steps to mitigate are necessary for a real world implementation. Moreover, the model was built in a stable testing environment. Real world devices and larger amounts of data can bring their own difficulties that should be considered in future implementations.

Future Work: In the future, we plan to test the qualitative evaluation metrics by realising the attacks against our prototype to have a more reliable evaluation based on a theoretical approach. This can also help to better understand the optimal ϵ value that should be achieved for each worker. So far, we preset the local F1-Score threshold for each client equally. As a next step, we plan to investigate how the model reacts to different local accuracy thresholds which enables us to to increase the number of participants and to address the individual privacy needs of each client. Along with this approach we are interested to test for natural persons as users of the spam and ham classification, how the combination of other PPML techniques meets their personal preferences by comparing user acceptance criteria and PPML characteristics. Besides the F1-Score we also plan to introduce precision as threshold metric. Again, even with a very low threshold, the central model can still a high accuracy. Another extension, is the implementation of poisoning detection and identification of local model bias [24].

8 Conclusion

In this work, we identify the benefits and drawbacks from combining FL and LDP for clients' gradients in a spam and ham email classification model. This is highly relevant for future implementations because spam detection relies on up to date models that can be adapted quickly. FL is very strong in learning rare events and distributing them quickly among all clients. Nonetheless, it is vulnerable against gradient attacks from inside and outside the network. With our work, we show that local DP helps to minimise the clients' privacy risk, while F1-Score and performance are kept on a considerably high level. But besides this, additional countermeasures are required to increase also the protection level of the global model against malicious clients. Regarding the performance and a loss of accuracy, a major drawback of DP, we show that a local threshold can help to compensate these issues. We find that setting a local threshold for the F1-Score on the clients' level can reduce the consumption of privacy budget, reduces both: the training time, and overfitting of the model. We find that DP, along with FL has a high potential to create strong, up to date spam filters among different entities. The next steps are to investigate different privacy thresholds among the clients, and to test the resilience against attacks of our model.

Bibliography

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

[2] Amjad, M., Voronkov, I., Saenko, A., Gelbukh, A.: Comparison of text classification methods using deep learning neural networks. In: Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing) (2019)

[3] Awan, S., Luo, B., Li, F.: Contra: Defending against poisoning attacks in federated learning. In: European Symposium on Research in Computer Security. pp. 455–475. Springer (2021)

[4] Basu, P., Roy, T.S., Naidu, R., Muftuoglu, Z.: Privacy enabled financial text classification using differential privacy and federated learning. arXiv preprint arXiv:2110.01643 (2021)

[5] Bhowmick, A., Hazarika, S.M.: E-mail spam filtering: a review of techniques and trends. Advances in electronics, communication and computing pp. 583–590 (2018)

[6] Dada, E.G., Bassi, J.S., Chiroma, H., Adetunmbi, A.O., Ajibuwa, O.E., et al.: Machine learning for email spam filtering: review, approaches and open research problems. Heliyon 5(6), e01802 (2019)

[7] Dong, Y., Chen, X., Li, K., Wang, D., Zeng, S.: Flod: Oblivious defender for private byzantine-robust federated learning with dishonest-majority. Cryptology ePrint Archive (2021)

[8] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference. pp. 265–284. Springer (2006)

[9] Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients-how easy is it to break privacy in federated learning? Advances in Neural Information Processing Systems 33, 16937–16947 (2020)

[10] Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557 (2017)

[11] Jain, G., Sharma, M., Agarwal, B.: Optimizing semantic lstm for spam detection. International Journal of Information Technology 11(2) (2019)

[12] Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine 37(3), 50–60 (2020)

[13] Löbner, S., Tesfay, W.B., Nakamura, T., Pape, S.: Explainable machine learning for default privacy setting prediction. IEEE Access 9, 63700–63717 (2021)

[14] Löbner, S., Tronnier, F., Pape, S., Rannenberg, K.: Comparison of de-identification techniques for privacy preserving data analysis in vehicular data sharing. In: Computer Science in Cars Symposium. pp. 1–11 (2021)

[15] Makkar, A., Ghosh, U., Rawat, D.B., Abawajy, J.: Fedlearnsp: preserving privacy and security using federated learning and edge computing. IEEE Consumer Electronics Magazine (2021)
[16] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
[17] Metsis, V., Androutsopoulos, I., Paliouras, G.: Spam filtering with naive bayes-which naive bayes? In: CEAS. vol. 17. Mountain View, CA (2006)
[18] Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
[19] Powers, D.M.: Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. preprint arXiv:2010.16061 (2020)
[20] Tanuwidjaja, H.C., Choi, R., Baek, S., Kim, K.: Privacy-preserving deep learning on machine learning as a service—a comprehensive survey. IEEE Access 8, 167425–167447 (2020)
[21] Thapa, C., Tang, J.W., Abuadbba, S., Gao, Y., Zheng, Y., Camtepe, S.A., Nepal, S., Almashor, M.: Fedemail: Performance measurement of privacy-friendly phishing detection enabled by federated learning (2020)
[22] Tolpegin, V., Truex, S., Gursoy, M.E., Liu, L.: Data poisoning attacks against federated learning systems. In: European Symposium on Research in Computer Security. pp. 480–501. Springer (2020)
[23] Triastcyn, A., Faltings, B.: Federated learning with bayesian differential privacy. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 2587–2596. IEEE (2019)
[24] Tronnier, F., Pape, S., Löbner, S., Rannenberg, K.: A discussion on ethical cybersecurity issues in digital service chains. In: Cybersecurity of Digital Service Chains, pp. 222–256. Springer, Cham (2022)
[25] Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q., Poor, H.V.: Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security 15, 3454–3469 (2020)
[26] Wei, W., Liu, L., Loper, M., Chow, K.H., Gursoy, M.E., Truex, S., Wu, Y.: A framework for evaluating client privacy leakages in federated learning. In: European Symposium on Research in Computer Security. Springer (2020)
[27] Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) 10(2), 1–19 (2019)
[28] Yousefpour, A., Shilov, I., Sablayrolles, A., Testuggine, D., Prasad, K., Malek, M., Nguyen, J., Ghosh, S., Bharadwaj, A., Zhao, J., Cormode, G., Mironov, I.: Opacus: User-friendly differential privacy library in PyTorch. arXiv preprint arXiv:2109.12298 (2021)
[29] Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. Advances in Neural Information Processing Systems 32 (2019)