Private Federated Learning with Domain Adaptation

Daniel W. Peterson, Pallika Kanani, Virendra J. Marathe Oracle Labs
{daniel.peterson,pallika.kanani,virendra.marathe}@oracle.com

Abstract

In a federated learning (FL) system, users can colloborate to build a shared model without explicitly sharing data, but model accuracy degrades if differential privacy guarantees are required during training. We hypothesize that domain adaptation techniques can effectively address this problem while increasing per-user prediction accuracy, especially when user data comes from disparate distributions. We present and analyze a mixture of experts (MoE) based domain adaptation approach that allows effective collaboration between users in a differentially private FL setting. Each user contributes to (and benefits from) a general, shared model to perform a common task, while maintaining a private model to adjust their predictions to their particular domain. Using both synthetic and real-world datasets, we empirically demonstrate that these private models can increase accuracy, while protecting against the release of users' private data.

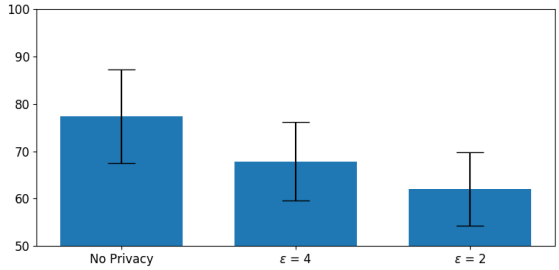

Figure 1: Classifier accuracy (higher is better) on the spam dataset comprising 15 users cooperating in a FL setting (more details on the benchmark in Evaluation section). The model architectures in all the FL test runs are identical. Introduction of DP-induced noise significantly compromises accuracy.

Introduction

Federated Learning (FL) is a distributed Machine Learning (ML) paradigm that enables multiple parties to jointly retrain a shared model without sharing their data with any other parties (Bonawitz et al. 2019; Konecný, McMahan, and Ramage 2015), offering advantages in both scale and privacy. In FL, multiple parties wish to perform essentially the same task using ML, with a model architecture that is agreed upon in advance. Each user wants the best possible classifier for their individual use, but has a limited budget for labeling their own data. Pooling the data of multiple users could improve model accuracy, because accuracy generally increases with increased training data. The FL framework allows them to effectively pool their labeled data, without explicitly sharing it. Although the initial focus of FL has been on targeting millions of mobile devices (Bonawitz et al. 2019), the benefits of its architecture are evident even for enterprise settings. Consider an ML service being offered to enterprise customers: the number of users of an ML service may be much smaller, but privacy concerns are paramount.

Recognizing that just FL is not sufficient to guarantee data privacy, researchers have proposed the addition of Differential Privacy (DP) (Dwork 2006; Dwork and Roth 2014; Dwork et al. 2006) to FL (Abadi et al. 2016; Geyer, Klein, and Nabi 2017; Konecný et al. 2016; McMahan et al. 2016). Informally, differential privacy aims to provide a bound on the variation in the model's output based on the inclusion or exclusion of a single data point used in its training set. Introducing "noise" in the training process (typically inputs, objective function, gradients, or outputs) makes it difficult to guarantee whether any particular data point was used to train the model. While this noise is structured to enforce formally provable privacy guarantees for the data point (Dwork et al. 2006), it can degrade accuracy of model predictions. Table 1 depicts performance of an FL model that enforces different levels of DP guarantees for training data (No Privacy, $\varepsilon = [2, 4]$). Clearly, as the DP related bounds grow tighter (smaller value of ε), indicating better privacy guarantees, the FL model delivers worse performance, despite the larger training dataset available through FL.

We consider the setting in which individual user data comes from diverse domains. This is common, because each user can have a different data-generating process. There exists a large body of work on domain adaptation in non-FL systems (Ben-David et al. 2010; Crammer, Kearns, and Wortman 2008; Kouw and Loog 2018; Pan and Yang 2010; Daumé III 2009). In domain adaptation, a model trained over a dataset from a source domain is further refined to adapt to a dataset from a different target domain. We hypothesize

that along with bridging the data distribution gap, domain adaptation can also address the aforementioned problem of accuracy reduction in differentially private FL.

In this work, we use a domain adaptation technique, Mixture of Experts (MoE), where outputs of a number of domain-expert models are combined to derive the refined output. More specifically, we propose a framework to augment the FL setting with per-user domain adaptation. We show that our approach improves model accuracy for all users, using both real and synthetic data. Furthermore, the improvement is much more pronounced when differential privacy bounds are imposed on the FL model.

We use differentially private FL to train a public, general model on the task. We also learn a private, domain-specific model using each user's own data. Each user combines the output of the general and private models using a mixture of experts (MoE) (Masoudnia and Ebrahimpour 2014; Nowlan and Hinton 1991) to make their final predictions. The two "experts" in the mixture are the general FL model and the domain-tuned private model, so we refer to our system as federated learning with domain experts (FL+DE). For privacy in the general model, we use FL with differentially private stochastic gradient descent (DPSGD) (Abadi et al. 2016). The private domain models are trained using ordinary stochastic gradient descent (i.e. without differential privacy noise). In principle, the two model architectures can be identical or radically different, but for convenience we maintain a common model architecture for the general (public) and private models. Using a MoE architecture allows the general and private models to influence predictions differently on individual data points.

We demonstrate on synthetic as well as a real-world spam detection (Bickel 2006) datasets that our system significantly outperforms the accuracy of differentially private federated learning (DPFL). This largely boils down to two factors. First, the private models provide domain adaptation, which is known to typically increase accuracy in each domain. On a real-world classification task, we observe that domain adaptation improves accuracy even when using non-private SGD. Second, the private models allow noise-free updates, because there is no need to conceal private data from the private model. While the accuracy of the DPFL system degrades by 10.5% in the low-noise setting, the performance of FL+DE degrades by $<$0.5%. In the high-noise setting, the accuracy of the differentially-private FL system degrades by 15.3% and FL+DE accuracy degrades by only 2.7%. We additionally analyze the implications of our MoE architecture on the shared general model's stability, and the impact of individual users on FL+DE's accuracy and stability.

Background

Federated Learning (FL) Federated learning (Konecný, McMahan, and Ramage 2015) is a distributed ML paradigm that enables collaborative training of a model between multiple parties – user mobile devices in the original work (Bonawitz et al. 2019). In FL, the model is re-trained at each party using the party's private data. The modified model's gradients are then shipped back to the *federation server*. The federation server periodically aggregates the gradients it receives from a plurality of parties, and applies them to the central model. This updated central model's aggregated gradients are then pushed back to all the parties, which subsequently replace their locally re-trained models with this central model. The cycle keeps repeating throughout the lifetime of the model. Figure 2 shows the overall FL architecture. Users can dynamically join the federation or drop out. The framework is structured to be resilient to such changes. Noting privacy concerns, more recent work has proposed addition of differential privacy to FL (Geyer, Klein, and Nabi 2017; Konecný et al. 2016; McMahan et al. 2016).

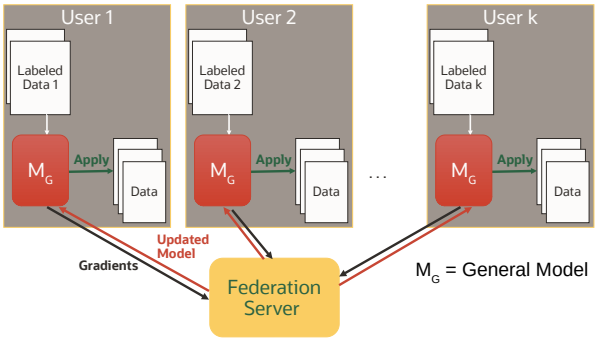

Figure 2: The Federated Learning setting.

Differential Privacy (DP) *Differential privacy* (Dwork et al. 2006) is a mathematically quantifiable privacy guarantee for a data set used by a computation that analyzes it. While it originally emerged in the database and data mining communities, triggered by privacy concerns in ML (Fredrikson, Jha, and Ristenpart 2015; Fredrikson et al. 2014; Hitaj, Ateniese, and Perez-Cruz 2017; Korolova 2010; Shokri et al. 2017; Tramèr et al. 2016), DP has garnered enormous traction in the ML community over the last decade (Abadi et al. 2016; Carlini et al. 2019; Chaudhuri, Monteleoni, and Sarwate 2011; Differential Privacy Team 2017; Dimitrakakis et al. 2017; Fredrikson et al. 2014; Fredrikson, Jha, and Ristenpart 2015; Park et al. 2016a,b; Sarwate and Chaudhuri 2013).

In DP, the privacy guarantee applies to each individual item in the data set and is formally specified in terms of a pair of data sets that differ in at most one item. Specifically, consider an algorithm A such that $A : D \mapsto R$, where D and R are respectively the domain and range of A. Now consider two data sets d and d' that differ from each other in exactly one data item. Such data sets are considered *adjacent* to each other in the DP literature. Algorithm A is said to be (ε, δ)-differentially private if the following condition holds true for all adjacent d and d' and any subset of outputs $O \subseteq R$:

$$P[A(d) \epsilon O] \leq e^{\varepsilon} P[A(d') \epsilon O] + \delta \quad (1)$$

In other words, ε represents the upper bound for variance between the probabilities with which A generates an output from O from inputs d or d'. δ represents the probability with which the ε bound may not hold (information on the input data set may be leaked). ε-differential privacy is guaranteed when $\delta = 0$.

Enforcement of DP typically translates into introduction of a "correction" in algorithm A to ensure that the differential privacy bound holds for any two adjacent inputs. This correction is commonly referred to as the *noise* introduced in the algorithm, its input, or output to ensure that the (ε, δ)-differential privacy bound holds. While a disciplined introduction of noise guarantees DP, the noise itself leads to accuracy degradation in the output produced by A. In the context of ML, the algorithm is a model being trained using sensitive private data sets, and accuracy degradation can significantly hamper the model's utility.

In the FL context, DP is introduced by adding noise to the gradients generated at each user's site (Abadi et al. 2016; Song, Chaudhuri, and Sarwate 2013). These noisy gradients are sent to the federation server that aggregates and applies them to the model. As we showed in Table 1, this noise can severely degrade the trained model's accuracy, to the extent of making it ineffective compared to per-user locally trained models (see Evaluation section).

Domain Adaptation Domain adaptation is the challenge of adjusting ML models to perform on different data sets or different target tasks. A classing domain adaptation setup models a large amount of labeled data drawn from one distribution (source domain), and a pool of unlabeled or partially-labeled data drawn from another distribution (target domain). Because the domains have different distributions, a model trained only on source-domain data is unlikely to perform optimally on the target domain. Many domain-adaptation techniques have been proposed that successfully leverage labeled data in the source domain to improve model performance in the target domain (Crammer, Kearns, and Wortman 2008; Daumé and Marcu 2006; Daumé III 2009; French, Mackiewicz, and Fisher 2017; Samdani and Yih 2011; Sun and Shi 2013).

Of particular interest to our work is the Mixture of Experts (MoE) technique used for domain adaptation (Masoudnia and Ebrahimpour 2014; Nowlan and Hinton 1991), which bears some resemblance to other domain adaptation techniques (Guo, Shah, and Barzilay 2018; Tu and Sun 2012). Each user learns a domain-specific expert model, and then makes its final prediction using a mixture of its domain-specific expert and the collaborative, general model.

Our Model

At its core, our proposal is to mix the outputs of a collaboratively-learned general model and a local domain expert. Participating users have their independent set of labeled training examples that they wish to keep private, drawn from user-specific domain distributions. These users collaborate to build a general model for the task. At the same time, users maintain private, domain-adapted expert models. The final predictor is a weighted average of the outputs from the general and private models. These weights are learned using a MoE architecture (Masoudnia and Ebrahimpour 2014; Nowlan and Hinton 1991), so the entire model can be trained with gradient descent.

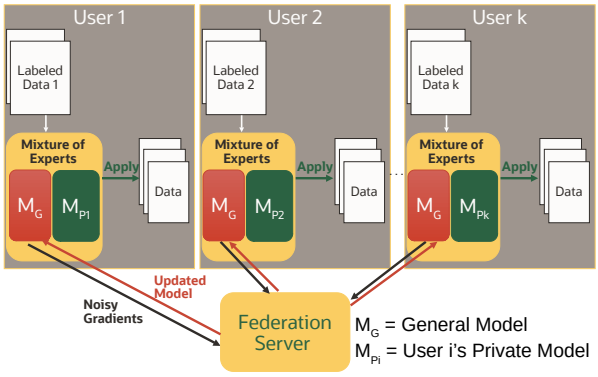

Figure 3: The Federated Learning setting with Domain Experts, one per participating user. Each user uses its local MoE model for both, training and inference.

Figure 3 depicts our overall system architecture. M_G is the general model that is trained by the FL framework. (ε, δ)-differential privacy is enforced by adding noise to the gradients sent back to the federation server. All the M_{P_i} models are domain experts tuned to the respective users' data distributions. A user uses its private labeled data to retrain its private model M_{P_i} along with M_G. M_{P_i} is completely private to the user and hence does not need DP-inducing noise. The MoE framework at each user combines outputs of the two models, tuning the MoE output to better suit the user's data distribution if needed.

More formally, let M_G be a general model, with parameters Θ_G, so that $\hat{y}_G = M_G(x, \Theta_G)$ is M_G's predicted probability for the positive class, or perhaps a regressed value[1]. M_G is shared between all users, and is trained on all data using FL with differentially private SGD (Abadi et al. 2016), enabling each user to contribute to training the general model.

Similarly, let M_{P_i} be a private model of user i, parameterized by Θ_{P_i}, and $\hat{y}_{P_i} = M_{P_i}(x, \Theta_{P_i})$ be the model's predicted probability. Although M_{P_i} could have a different architecture from M_G, in this work we initialize M_{P_i} as an exact copy of M_G. Neither M_{P_i}, nor gradient information about it, is shared with any other party, so M_{P_i} can be updated exactly, without including privacy-related noise.

These two models are combined using a *gating function*,

[1] Although we tested only binary classification and regression in our experiments, there are obvious extensions to multiclass problems.

Algorithm 1 Minibatch Update for Mixture of Experts at user i (Outline).

input User i, their N examples $x_1, x_2, \ldots, x_N$ and corresponding labels $y_1, y_2, \ldots, y_N$, general model M_G and its parameters Θ_G, private model M_{P_i} and its parameters Θ_{P_i}, gating function α_i and its parameters Θ_{α_i}, learning rate η_t, noise scale σ, group size L, gradient norm bound C

Initialize $\mathbf{g}$, $\mathbf{g}_G$, $\mathbf{g}_{P_i}$ to 0.
Take a random sample L_t with sampling probability L/N
for $x_t \in L_t$ **do**
    Compute prediction $\hat{y}_t$ with equation 2
    Compute loss $l_t = ||y_t - \hat{y}_t||$
    $\mathbf{g}_G += compute\_clipped\_gradient(l_t, \Theta_G, C)$
    $\mathbf{g}_{P_i} += compute\_gradient(l_t, \Theta_{P_i})$
    $\mathbf{g}_{\alpha_i} += compute\_gradient(l_t, \Theta_{\alpha_i})$
end for
$\mathbf{g}_G += gaussian\_noise(0, C * \sigma^2)$
$\Theta_{P_i} \leftarrow \Theta_{P_i} - \frac{\eta_t}{L} \mathbf{g}_{P_i}$
$\Theta_{\alpha_i} \leftarrow \Theta_{\alpha_i} - \frac{\eta_t}{L} \mathbf{g}_{\alpha_i}$
Send $\mathbf{g}_G/L$ to the Federation Server.

$\alpha_i(x)$, that can learn which model to trust as a function of the input. In our experiments, we set $\alpha_i(x) = S(w_i^T \cdot x + b_i)$, where $S(x)$ is the sigmoid function, and w_i and b_i are learned weights. Although there are many other choices for the gating function, this choice is simple, differentiable, and allows smooth mixing across the boundary between the two models.

Thus the final output $\hat{y}_i$ depends on learned parameters Θ_G, Θ_{P_i}, w_i, and b_i, and all are updated via SGD. The final output that user i uses to label data is

$$\hat{y}_i = \alpha_i(x) M_G(x, \Theta_G) + (1 - \alpha_i(x)) M_{P_i}(x, \Theta_{P_i}). \quad (2)$$

We advise, in line with standard MoE practice, that the training of the gating function parameters is separated from the training of the training of the expert models themselves. In all experiments, we split the training data in two and train the parameters of $\alpha_i(x)$ separately from the models M_G and M_{P_i}. We also found value adding a penalty term that discourages the gating function from always preferring only one expert, of the form $|(\sum_x \alpha_i(x)) - 0.5 * N_x|$, where N_x is the number of data points used to compute the update to the parameters of $\alpha_i(x)$.

Algorithm 1 depicts a single batch of MoE training for user i at a high level. During training, each user will perform this procedure many times, while receiving updates to the general model from the federation server. The differentially-private gradient computation $\mathbf{g}_G$ is based on algorithm by Abadi et. al. (Abadi et al. 2016). We apply equation 2 to control the gradient aggregate $\mathbf{g}$ that is eventually sent to the federation server.

The private model M_{P_i} and weighting mechanism α_i work together to provide a significant benefit over differentially private FL. First, by allowing individual domain adaptation, they boost accuracy. Second, because they allow noise-free updates, they prevent the accuracy loss associated with more stringent privacy requirements, which add noise to the general model.

Over time, a user's gating function $\alpha_i(x)$ learns whether to trust the general model or the private model more for a given input, and the private model M_{P_i} needs to perform well on only the subset of points for which the general model fails. The failure of the general model is expected, and corrected, so the mixture of experts can tolerate a poor-quality general model so long as it is reliably correct in at least some region of the input space.

Effects of the Gating Function on Gradients

One advantage of our MoE setup is that the loss is differentiable with respect to all model parameters - the parameters of the private and general models, and the gating function. When $\alpha_i(x) << 1$, the general model makes little contribution to the user's prediction. But this gating function also gates the gradients, because the general model makes little contribution to the loss. Symmetrically, the gradients for the private model vanish as $\alpha_i(x) \to 1$.

For the private model, gating the gradients is perfectly reasonable. Once the gating function is sure that the general model is making accurate predictions on a particular data point, there is no need to penalize the private model for its performance in that region. The tradeoff in model quality is not symmetric for the shared model, however, because the quality of the shared model affects multiple users. Especially if new users are joining the federation over time, it is still desirable that the general model performs as well as possible while satisfying privacy concerns.

On the other hand, allowing the gating function to suppress updates to the general model may be beneficial for the privacy of user data. Once a users' gating function has learned to trust their private model on a particular data point, there is little incentive for them to share further information about that data point with the federation pool. However, if the overall quality of the general model is not a part of the objective, it only needs to make satisfactory recommendations in a small region of the input space, particularly the regions commonly shared amongst several users. Different regions of reliability correspond to different local optima, and there are many locally-optimal solutions where the general model is an extremely poor model for the overall task, but is still a useful expert for most users on some portion of inputs. We characterize this phenomenon by observing the behavior of the model during training, and evaluation of the shared model on a held-out dataset.

We examine the effects of allowing the gating function to suppress gradients by implementing an alternative algorithm, that still uses an MoE framework. In the modified algorithm, users compute the loss and gradient of the general model on their data points, and send gradients according to the DP SGD procedure, regardless of the weight $\alpha_i(x)$. Then they compute the final prediction using the MoE, and make noise-free updates to their private model and gating function as before. This makes the training of the general model equivalent to traditional FL, while allowing the flexibility of

Algorithm 2 Alternative Minibatch Update for Mixture of Experts at user i (Outline).

input User i, their N examples $x_1, x_2, \ldots, x_N$ and corresponding labels $y_1, y_2, \ldots, y_N$, general model M_G and its parameters Θ_G, private model M_{P_i} and its parameters Θ_{P_i}, gating function α_i and its parameters Θ_{α_i}, learning rate η_t, noise scale σ, group size L, gradient norm bound C

1: **Initialize** $\mathbf{g}$, $\mathbf{g}_G$, $\mathbf{g}_{P_i}$ to 0.
2: Take a random sample L_t with sampling probability L/N
3: **for** $x_t \in L_t$ **do**
4: Compute prediction $\hat{y}_t$ with equation 2
5: Compute loss $l_t = ||y_t - \hat{y}_t||$
6: Compute loss $l_t^G = ||y_t - M_G(x_t, \Theta_G)||$
7: $\mathbf{g}_G += compute\_clipped\_gradient(l_t^G, \Theta_G, C)$
8: $\mathbf{g}_{P_i} += compute\_gradient(l_t, \Theta_{P_i})$
9: $\mathbf{g}_{\alpha_i} += compute\_gradient(l_t, \Theta_{\alpha_i})$
10: **end for**
11: $\mathbf{g}_G += gaussian\_noise(0, \sigma)$
12: $\Theta_{P_i} \leftarrow \Theta_{P_i} - \frac{\eta_t}{L}\mathbf{g}_{P_i}$
13: $\Theta_{\alpha_i} \leftarrow \Theta_{\alpha_i} - \frac{\eta_t}{L}\mathbf{g}_{\alpha_i}$
14: Send $\mathbf{g}_G/L$ to the Federation Server.

the MoE to improve predictions for each user. This change is explicitly demonstrated in Algorithm 2, lines 6-7.

Evaluation

The main hypothesis of this work is that domain adaptation techniques can improve accuracy in a federated learning setting, and that this accuracy improvement holds even when noise is added to protect the privacy of the gradient updates. We illustrate the effectiveness of our domain adaptation technique on two datasets. Furthermore, we analyze effects of our MoE-based domain adaptation approach on the overall accuracy and stability of the general model.

Synthetic Dataset

The first dataset is a synthetic regression problem. Two users attempt to fit a linear model of the function $f(x, y) = 5x - 2y + 0.5y^3$. Each has input data drawn from a distinct 2-dimensional Gaussian, and because of these domain differences, they get different exposure to the nonlinear y^3 term. We draw 2500 training examples, 500 validation examples, and 500 test examples for each user, all from that user's 2d Gaussian, then compute $f(x, y)$. The users aim to minimize root mean squared error (RMSE) on the test set. The baseline error is computed with each user fitting a single linear model to their training data. We then compute RMSE for each user if the users collaborate to build a single linear model using FL, and augmenting FL with private domain experts (Algorithm 1). Figure 4 shows the synthetic data, the target function, and the learned gating functions for both users. To see the effects of differential privacy, we test with low noise ($\sigma = 2$) and high noise ($\sigma = 4$), following prior work (Abadi et al. 2016).

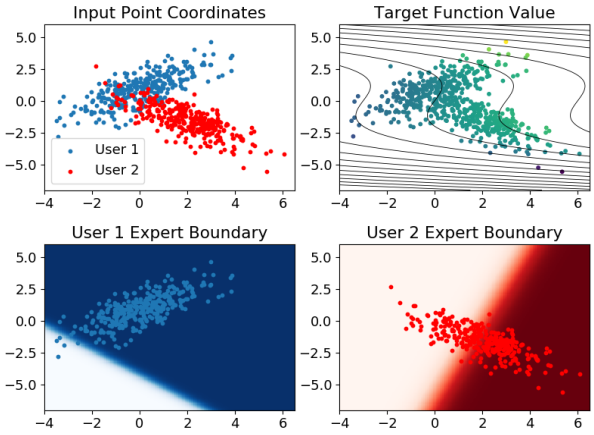

Figure 4: Visualizing the synthetic data experiment. Axes for all figures represent the (x, y) coordinates of data points. Left to right: (a) (x, y) coordinates of test data points sampled from distinct 2d Gaussians. (b) Target values of nonlinear function $f(x, y)$. (c) Values of the MoE gating function, $\alpha_1(x, y)$ learned by User 1. In the darker region, the private domain expert is preferred, while the general model is preferred in the lighter region. (d) The gating function $\alpha_2(x, y)$ of User 2, which uses the shared model in a different region than User 1.

Test errors for the baseline, FL, and domain-adapted FL systems are provided in Table 5. Algorithm 1 provides the best results of any model, and graceful degradation compared to differentially-private FL as the noise increases. FL alone provides a lower error for both users if there are no privacy concerns, but as we increase the noise we apply to the gradient, we observe a dramatic increase in error. The system with domain experts is more expressive than a single linear model - it learns a linear model and gating function for each user on top of the shared linear model - so it is unsurprising that RMSE is lower when no noise is added to the gradients. However, Algorithm 1 does not degrade in performance as much as FL when noise is added to the shared gradient updates. In the worst case, the performance degrades only to the baseline level (where each user has a linear model for its entire dataset).

Domain Adaptation for Spam Detection

The second dataset is a real-world domain adaptation dataset for spam detection, which was released as part of the ECML PKDD 2006 Discovery Challenge (Bickel 2006). The task is to classify whether an email in a user's inbox is spam, and personalizing the spam filtering for each user. The amount of data available per user is limited, so it is expected that collaboration can increase the quality of the classifier. However, each user has a different inbox, so domain adaptation is required. The dataset was originally designed to test methods of unsupervised domain adaptation, but using the evaluation dataset labels, which are now publicly available, we simulate 15 users collaborating to build a spam classifier in a

System	User 1 RMSE	User 2 RMSE
Baseline	15.32	10.95
FL, $\sigma = 0$	12.75	9.67
FL, $\sigma = 2$	13.79	12.68
FL, $\sigma = 4$	12.59	19.49
Alg 1, $\sigma = 0$	12.12	**9.41**
Alg 1, $\sigma = 2$	**12.05**	9.73
Alg 1, $\sigma = 4$	13.78	10.95

Figure 5: Test error for regression models trained on synthetic data (lower is better). The domain-only baseline system trains a separate model for each user on their data. Traditional FL, and our system of FL with domain experts (Alg 1) are tested with various noise levels, σ, for differential privacy.

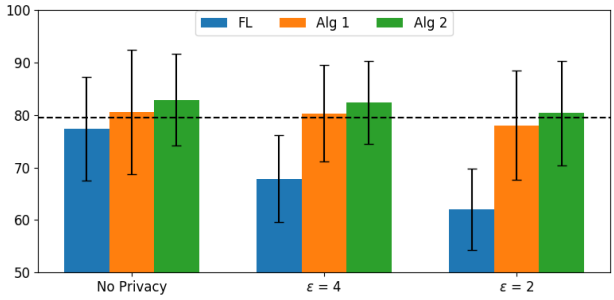

Figure 6: Classifier accuracy on the spam dataset (higher is better). FL and our system (Algorithm 1) are evaluated as gradient noise is increased (for differential privacy). The dashed horizontal line indicates domain-only baseline performance. Error bars show performance variance across users.

supervised setting. In this case, we measure classifier accuracy, averaged across all users. We use the dataset from task b. For each of our users, we train on 50 labeled examples, leaving 350 examples for testing. The baseline system trains one classifier for each user, using in-domain data only, and we also train a collaborative FL model, and domain-adapted FL models using both Algorithm 1 and Algorithm 2.

In our experiments, we fix $\delta = 10^{-5}$ and compute σ using the moments accountant (Abadi et al. 2016) to provide guarantees of $\epsilon \in [2, 4]$, simulating relatively strict and relatively permissive differential privacy guarantees. We fix the number of epochs to 20, motivated by making the best use of the privacy budget and avoiding overtraining. If we increase the number of training epochs, we notice the following trends. First, the non-private FL system would improve, because it has more time to fit the data[2]. Second, the DPFL systems would deteriorate, because of the extra noise required to maintain a fixed privacy budget across more epochs. Third, the MoE system will begin to overfit the private expert and the gating function will learn to always prefer the private model over the general model, leaving performance equivalent to the individual baseline models. Early stopping is widely used as a regularization technique, and we recommend using it in our MoE-based domain adaptation system.

The results are illustrated in Figure 6. Once again, FL with domain experts provides the best overall accuracy, and maintains its performance as noise is added to provide differential privacy. We see a small increase in accuracy using Algorithm 2, rather than Algorithm 1, across the board. The accuracy of the baseline system is 79.56% ($\pm$ 12.2%), averaged across users. In the absence of noise, FL achieves 77.34% mean accuracy, Algorithm 1 achieves 80.61% mean accuracy, and Algorithm 2 achieves 82.91% mean accuracy in the same number of training epochs. When we add moderate privacy guarantees ($\epsilon = 4$), FL classification accuracy drops to 67.85%, but Algorithms 1 and 2 maintain nearly identical performance (80.29% and 82.41% accuracy). With stronger privacy guarantees ($\epsilon = 2$), the FL system accuracy drops further to 62.04%, and Algorithm 1 drops below the baseline performance to 77.99%, but Algorithm 2 maintains an 80.37% accuracy. The trends show that the MoE domain adaptation technique can protect the accuracy of the predictions, even when noise has degraded the accuracy of the FL model.

Effect of the Gating Function on Gradients

Empirical Gradient Sizes During Training It is worth getting an estimate of how much the gating function actually affects the gradients passed during training. We measure the average L2-norm of the gradients across each batch as we train the spam detection model for 15 users. We consider both traditional FL, where the gradient sizes are unconstrained, and the DP-SGD procedure, where each example has its gradient clipped to a maximum L2-norm (in our case, the norm of the gradient for each parameter matrix is independently clipped to a magnitude of 1). In both cases,

[2]20 epochs gives sufficient time for the individual baseline models to converge.

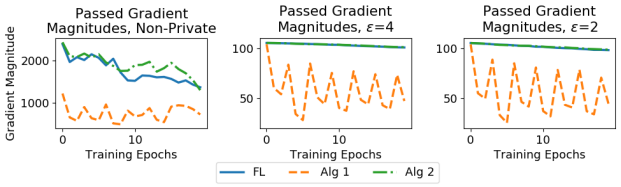

Figure 7: Average magnitude of gradients passed to the federation server as training progresses. Allowing the gating function to reduce the gradients has a nearly immediate effect in suppressing the gradients sent to the general model, whether using ordinary or differentially-private SGD.

we observe a marked reduction in the magnitude of gradients sent over the course of training, if the gating function is allowed to suppress gradients that would otherwise be sent to the general model.

It is clear from Figure 7 that allowing $\alpha_i(x)$ to gate the gradients reduces the total magnitude of gradient updates, whether we clip gradients to a bounded magnitude or not, and that Algorithm 2 passes essentially the same magnitude of gradients to the federation server as FL without the mixture of experts. The averages shown here are across multiple restarts and all users.

Stability of the General Model's Predictions We also wish to test how often the general model M_G gets caught in local optima if the gradients sent to the general model are gated. One way to measure this is to restart the model with another initialization, and evaluate whether it tends to make the same predictions on a fixed dataset. The spam detection dataset includes a small dataset of 100 labeled emails not associated to any particular user. We use this held-out data to analyze how much the shared model varies across runs.

We ran the model with ten distinct random initializations, and compared the pairwise differences in predictions on these held-out emails. We also ran the model with a single fixed initialization, but perturbed the training by holding out a single user at each run. Both types of perturbations are extremely plausible in a real-world FL scenario - we have no guarantees that we're in an optimal random initialization, or that the user base stays fixed.

Table 1 and Table 2 show that the predictions of the shared model are much more similar across runs if full gradients are passed, compared to allowing the gating function to suppress gradients for "private" data points. The general model is also more accurate using Algorithm 2 compared to Algorithm 1. The instability and inaccuracy of the general model makes only a slight impact on per-user accuracy, though, as seen in Table 6, suggesting that the users are typically able to find satisfactory work-arounds for any given general model, and that Algorithm 1 tends to stop in local optima that differ greatly based on initialization.

Overall, Algorithm 2 provides much more stability, a boost to accuracy on the general model, and a slight improvement in per-user accuracy. These properties are likely to make it more useful on larger datasets.

Model	**Privacy Budget**	$P(\hat{y}_1 \neq \hat{y}_2)$	**Accuracy**
FL	∞	0.38 ± 0.13	0.65 ± 0.11
FL	4	0.38 ± 0.13	0.56 ± 0.06
FL	2	0.43 ± 0.12	0.55 ± 0.07
Alg 1	∞	0.50 ± 0.22	0.53 ± 0.06
Alg 1	4	0.47 ± 0.12	0.51 ± 0.08
Alg 1	2	0.51 ± 0.14	0.50 ± 0.09
Alg 2	∞	0.35 ± 0.12	0.65 ± 0.06
Alg 2	4	0.35 ± 0.08	0.53 ± 0.06
Alg 2	2	0.43 ± 0.10	0.52 ± 0.06

Table 1: Stability of predictions of the shared model across random re-initializations. There is a high variance across runs, so we notice that the shared model may make dramatically different predictions on the same held-out data. However, FL and Alg 2 provide significantly more stable predictions than Alg 1, whose shared model has a roughly 50% chance of producing the same prediction on the same data.

Model	**Privacy Budget**	$P(\hat{y}_1 \neq \hat{y}_2)$	**Accuracy**
FL	∞	0.31 ± 0.10	0.64 ± 0.09
FL	4	0.37 ± 0.10	0.54 ± 0.07
FL	2	0.44 ± 0.09	0.52 ± 0.07
Alg 1	∞	0.42 ± 0.15	0.59 ± 0.07
Alg 1	4	0.46 ± 0.12	0.48 ± 0.07
Alg 1	2	0.49 ± 0.12	0.47 ± 0.09
Alg 2	∞	0.35 ± 0.11	0.64 ± 0.09
Alg 2	4	0.39 ± 0.11	0.52 ± 0.06
Alg 2	2	0.47 ± 0.10	0.50 ± 0.06

Table 2: Stability of predictions of the shared model with a fixed initialization, as individual users are dropped from the training pool. There is a high variance across runs, so we notice that the shared model may make dramatically different predictions on the same held-out data. Again, FL and Alg 2 provide significantly more stable predictions than Alg 1. Overall, variance in predictions due to dropping a user is slightly less than variance due to initialization differences.

Related Work

In their work on DP in deep learning, Abadi et. al. (Abadi et al. 2016) formulated privacy loss of data used to train a deep learning model as a random variable. The moments of this random variable were used to derive tighter bounds for the cumulative privacy loss. The privacy loss was computed in their system using a *moments accountant* module. In our work, we use a subsequent incarnation of the moments accountant module (tf-; Mironov 2017) to derive our training data's privacy loss in the general model M_G's training.

Our core idea of maintaining two models per party is somewhat similar to Daumé and Marcu's work (Daumé and Marcu 2006), where they propose training three models, a "source-domain" model, a "target-domain" model, and a "general" model. The composition of these models during training and for inference is quite complex, whereas our approach uses a simple weighted averaging algorithm. Furthermore, their work targets two domains, whereas ours is more suitable for FL and can target a multitude of domains, one for each participating party.

Domain adaptation and federated learning have been studied in privacy-preserving and secure settings. One line of work focuses on protecting privacy in a classic domain adaptation setup (Guo et al. 2018), where a well-tuned model on a source domain is adapted to perform better in a target domain with more limited data. More recently, unsupervised domain adaptation technique for federated learning has been proposed (Peng et al. 2019), but their setup assumes multiple source domains, and a single target domain, with no joint model trained between users. They also do not take additional privacy measures such as differential privacy into account. The one model per node setting is also explored using multi-task learning (Smith et al. 2017), but they also do not consider additional privacy in federated learning. Another line of work focuses on secure federated learning (Liu, Chen, and Yang 2018), but uses additively homomorphic encryption to ensure privacy in a two-party federated learning context. This is different from ε-differential privacy, and does not maintain a collaborative general model. Each of these systems considers one part of our set-up, but no prior work combines efforts of collaborative learning combined with private domain adaptation.

Attention (Bahdanau, Cho, and Bengio 2014) has been used in a FL environment to weight updates from different users (Ji et al. 2018), allowing users with extremely unusual gradient updates to have a smaller disruptive effect on the general FL model. While this generally improved perplexity on unseen data, improving the general model, it does not allow users with unusual datasets to improve prediction quality on their domain.

The PATE architecture (Papernot et al. 2018) is yet another class of distributed ML systems that uses privately trained models of participating parties as sources for consensus-based labeling of data for a new user to help it train its model on its private data. The models trained for individual users act as an ensemble of "teachers" for the new party that is training a new model for itself. The consensus based labeling provides the privacy guarantees for each party. This approach could be used to build the general model, rather than differentially-private FL, but we have not yet tested its effectiveness in conjunction with domain-adaptation techniques.

Conclusion

This work demonstrates that adding private, per-user domain adaptation to a collaborative model-building framework can increase accuracy for all users, and is especially beneficial when privacy guarantees begin to diminish the utility of the collaborative general model.

Our implementation of domain adaptation employs a mixture of experts, with each user learning a domain expert model and a private gating mechanism. This domain adaptation framework is another contribution of our work, and allows us to train the entire model with gradient descent. We demonstrate that it works well in practice on both regression and classification tasks.

In future work, we aim to expand our analysis to include larger datasets and different architectures. We intent do consider other mechanisms for building a collaborative model (e.g., PATE), and alternative domain adaptation techniques (e.g., hypothesis transfer learning). We expect that the general setup of learning one collaborative generalist and a private domain adaptation mechanism will be useful in many settings and for many types of models, but that the best particular architecture could depend on the task. In situations where data privacy is not a concern, for example, the best performance may come from training an FL system and then adding domain adaptation, even with the MoE architecture we propose here, because joint training may overfit the MoE before the general model has had adequate time to learn a high-quality model.

References

???? TensorFlow Privacy *https://github.com/tensorflow/privacy*.

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A Theory of Learning from Different Domains. *Machine Learning* 79(1-2): 151–175. ISSN 0885-6125.

Bickel, S. 2006. ECML-PKDD Discovery Challenge 2006 Overview.

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konecný, J.; Mazzocchi, S.; McMahan, H. B.; Overveldt, T. V.; Petrou, D.; Ramage, D.; and Roselander, J. 2019. Towards Federated Learning at Scale: System Design. *CoRR* abs/1902.01046.

Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; and Song, D. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *28th USENIX Security Symposium*, 267–284.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially Private Empirical Risk Minimization. *The Journal of Machine Learning Research* 12: 1069–1109.

Crammer, K.; Kearns, M.; and Wortman, J. 2008. Learning from Multiple Sources. *Journal of Machine Learning Research* 9: 1757–1774.

Daumé, III, H.; and Marcu, D. 2006. Domain Adaptation for Statistical Classifiers. *Journal of Artificial Intelligence Research* 26(1): 101–126. ISSN 1076-9757.

Daumé III, H. 2009. Frustratingly Easy Domain Adaptation. *CoRR* abs/0907.1815. URL http://arxiv.org/abs/0907.1815.

Differential Privacy Team. 2017. Learning with Privacy at Scale, *https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html*.

Dimitrakakis, C.; Nelson, B.; Zhang, Z.; Mitrokotsa, A.; and Rubinstein, B. I. P. 2017. Differential Privacy for Bayesian Inference Through Posterior Sampling. *The Journal of Machine Learning Research* 18(1): 343–381.

Dwork, C. 2006. Differential Privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP*, 1–12.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, 265–284.

Dwork, C.; and Roth, A. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9(3–4): 211–407.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, 1322–1333.

Fredrikson, M.; Lantz, E.; Jha, S.; Lin, S.; Page, D.; and Ristenpart, T. 2014. Privacy in Pharmacogenetics: An End-to-end Case Study of Personalized Warfarin Dosing. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, 17–32.

French, G.; Mackiewicz, M.; and Fisher, M. 2017. Self-ensembling for visual domain adaptation. *arXiv preprint arXiv:1706.05208* .

Geyer, R. C.; Klein, T.; and Nabi, M. 2017. Differentially Private Federated Learning: A Client Level Perspective. *CoRR* abs/1712.07557.

Guo, J.; Shah, D. J.; and Barzilay, R. 2018. Multi-Source Domain Adaptation with Mixture of Experts. *arXiv preprint arXiv:1809.02256* .

Guo, X.; Yao, Q.; Tu, W.; Chen, Y.; Dai, W.; and Yang, Q. 2018. Privacy-preserving Transfer Learning for Knowledge Sharing. *arXiv preprint arXiv:1811.09491* .

Hitaj, B.; Ateniese, G.; and Perez-Cruz, F. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 603–618.

Ji, S.; Pan, S.; Long, G.; Li, X.; Jiang, J.; and Huang, Z. 2018. Learning Private Neural Language Modeling with Attentive Aggregation. *arXiv preprint arXiv:1812.07108* .

Konecný, J.; McMahan, B.; and Ramage, D. 2015. Federated Optimization: Distributed Optimization Beyond the Datacenter. *CoRR* abs/1511.03575. URL http://arxiv.org/abs/1511.03575.

Konecný, J.; McMahan, H. B.; Ramage, D.; and Richtárik, P. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *CoRR* abs/1610.02527.

Korolova, A. 2010. Privacy Violations Using Microtargeted Ads: A Case Study. In *2010 IEEE International Conference on Data Mining Workshops*, 474–482.

Kouw, W. M.; and Loog, M. 2018. An introduction to domain adaptation and transfer learning. *CoRR* abs/1812.11806. URL http://arxiv.org/abs/1812.11806.

Liu, Y.; Chen, T.; and Yang, Q. 2018. Secure Federated Transfer Learning. *arXiv preprint arXiv:1812.03337* .

Masoudnia, S.; and Ebrahimpour, R. 2014. Mixture of Experts: A Literature Survey. *Artificial Intelligence Review* 42(2): 275–293. ISSN 0269-2821.

McMahan, H. B.; Moore, E.; Ramage, D.; and y Arcas, B. A. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629.

Mironov, I. 2017. Renyi Differential Privacy. *CoRR* abs/1702.07476. URL http://arxiv.org/abs/1702.07476.

Nowlan, S. J.; and Hinton, G. E. 1991. Evaluation of adaptive mixtures of competing experts. In *Advances in neural information processing systems*, 774–780.

Pan, S. J.; and Yang, Q. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10): 1345–1359. ISSN 1041-4347.

Papernot, N.; Song, S.; Mironov, I.; Raghunathan, A.; Talwar, K.; and Erlingsson, Ú. 2018. Scalable Private Learning with PATE. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Park, M.; Foulds, J.; Chaudhuri, K.; and Welling, M. 2016a. Practical Privacy For Expectation Maximization. *CoRR* abs/1605.06995.

Park, M.; Foulds, J. R.; Chaudhuri, K.; and Welling, M. 2016b. Private Topic Modeling. *CoRR* abs/1609.04120.

Peng, X.; Huang, Z.; Zhu, Y.; and Saenko, K. 2019. Federated Adversarial Domain Adaptation.

Samdani, R. Y.; and Yih, W.-t. 2011. Domain adaptation with ensemble of feature groups. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Sarwate, A. D.; and Chaudhuri, K. 2013. Signal Processing and Machine Learning with Differential Privacy: Algorithms and Challenges for Continuous Data. *IEEE Signal Processing Magazine* 30(5): 86–94.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. 2017. Federated Multi-Task Learning.

Song, S.; Chaudhuri, K.; and Sarwate, A. D. 2013. Stochastic gradient descent with differentially private updates. In *IEEE Global Conference on Signal and Information Processing*, 245–248.

Sun, S.-L.; and Shi, H.-L. 2013. Bayesian multi-source domain adaptation. In *2013 International Conference on Machine Learning and Cybernetics*, volume 1, 24–28. IEEE.

Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M. K.; and Ristenpart, T. 2016. Stealing Machine Learning Models via Prediction APIs. In *Proceedings of the 25th USENIX Conference on Security Symposium*, 601–618.

Tu, W.; and Sun, S. 2012. Dynamical ensemble learning with model-friendly classifiers for domain adaptation. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, 1181–1184. IEEE.