Scalable and Fault Tolerant Platform for Distributed Learning on Private Medical Data

Alborz Amir-Khalili[1], Soheil Kianzad[2],
Rafeef Abugharbieh[1], and Ivan Beschastnikh[2]

[1] Biomedical Signal and Image Computing Lab, University of British Columbia
[2] Computer Science, University of British Columbia

Abstract. Medical image data is naturally distributed among clinical institutions. This partitioning, combined with security and privacy restrictions on medical data, imposes limitations on machine learning algorithms in clinical applications, especially for small and newly established institutions. We present InsuLearn: an intuitive and robust open-source[†] platform designed to facilitate distributed learning (classification and regression) on medical image data, while preserving data security and privacy. InsuLearn is built on ensemble learning, in which statistical models are developed at each institution independently and combined at secure coordinator nodes. InsuLearn protocols are designed such that the liveness of the system is guaranteed as institutions join and leave the network. Coordination is implemented as a cluster of replicated state machines, making it tolerant to individual node failures. We demonstrate that InsuLearn successfully integrates accurate models for horizontally partitioned data while preserving privacy.

1 Introduction

State-of-the-art machine learning (ML) techniques have shown promise in medical image analysis. Performance of successful ML techniques, e.g., deep learning, is directly tied to the amount of data available during the training phase as more data allows the trained statistical model to account for rarely occurring patterns and increases specificity to outliers. Compared to natural image datasets, medical image datasets typically contain fewer instances as they are subject to certain restrictions, namely: (i) **privacy:** medical data are privacy-sensitive as access to and transfer of patient data is restricted by privacy legislation; (ii) **distribution:** medical data are naturally distributed among different institutions; and (iii) **size:** medical image data requires considerable per-instance storage space and bandwidth for transfers between institutions. These restrictions limit medical data accessibility and constrain ML to data available at a single institution. This hurts collaboration and newly established, small, and rural institutions that

This work is supported in part by the Institute for Computing, Information and Cognitive Systems (ICICS) at UBC.

[†] Open-source code available at: `https://github.com/DistributedML/InsuLearn`.

do not have access to substantial datasets [1]. Unfortunately, most ML systems used by the medical image analysis community are not designed to overcome unique system requirements imposed by the restrictions above, which adds to the reluctance of clinical institutions to contribute data to these systems. All of the above points to a need for distributed ML systems that allow for multiple clinical data sources to contribute data without compromising data privacy.

Supporting data collaboration in a privacy-preserving way is challenging [2]. One solution is to generate a centralized *warehouse* that aggregates anonymized (sanitized) features extracted from different data sources. However, warehousing complicates data management and is ineffective at preventing adversaries from extracting sensitive information [3]. Furthermore, ethics review boards require a thorough audit of the proposed research before the sanitized data can be shared.

Private multi-party ML (PMPML) is an emerging alternative to warehousing, e.g., Google Inc. has recently revealed work in this space [4]. Among proposed PMPML methods, popular trends include: (T1) aggregation of privately learned models [1, 5–7] or (T2) using distributed privacy-preserving solvers to iteratively train a global model [4, 8]. Many T1 methods [5–7] require auxiliary publicly-available and annotated datasets, which limits their applicability. T2 methods, on the other hand, impose a predetermined global model that is sensitive to initialization and is thus difficult to design and refine. In contrast with T1 methods, T2 methods are designed for domains with more participants, i.e., smartphone users around the world [4] versus a few participating clinical institutions, and are thus ill-suited for the research and development of novel medical ML solutions.

To address the needs outlined above, we contribute **InsuLearn**: a scalable open-source solution that supports distributed T1 approaches in which every data source (node) privately trains local models that are then aggregated into a shared global model using an ensemble technique. InsuLearn does not require a publicly-available dataset and relies on other nodes to cross-validate locally trained models. As long as local models can be cross-validated at other nodes, InsuLearn does not impose any other requirements on the parameterization of the locally trained models. To our knowledge InsuLearn is the first open-source platform for the development of PMPML algorithms.

InsuLearn most closely resembles Li. et. al. [1] but, compared to InsuLearn, the peer-to-peer system of Li. et. al. is idealistic as: (i) It lacks a secure coordination node, implying that the local models trained at peers are known by all peers. This violates privacy since a malicious peer A can forward a crafted model to peer B and use B's response to infer data at B. (ii) The system was not implemented nor deployed on a real distributed system. (iii) The approach does not handle node/network failures, and does not discuss guarantees such as liveness, which is an ability of the system to make progress in spite of failures.

InsuLearn ensures data-security with built-in incentives to improve performance, deter abuse, and guarantee liveness as institutions join and leave the system. The key contributions of our work are: (i) We designed and implemented an open-source privacy-preserving distributed ML system that can interface with popular ML tools. (ii) We simultaneously improve privacy, fault tolerance, and liveness guarantees of the system by using replicated coordinator nodes. (iii) We propose a secure model aggregation scheme that is robust to noise and can be applied to popular classification and regression methods. (iv) We deployed our system prototype on Microsoft Azure cloud computing service to substantiate our claims regarding scalability and fault tolerance.

2 Methodology

InsuLearn uses ensemble learning techniques and supports different regression and classification tasks. Medical data is never transferred from one institution to another; instead, locally trained models are collected, sanitized, cross-validated, and aggregated into a global model with the help of coordinator nodes. InsuLearn is developed in Go, uses TCP for all network communication, and interfaces with the powerful ML toolboxes in MATLAB via MATLAB Engine C API[†]. This integration enables InsuLearn to deploy a wide selection of advanced ML algorithms and different ensemble strategies (e.g., boosting and bagging).

The basic representation of InsuLearn consists of a trusted secure coordinator node s_0 connected to $H = \{h_i : i = 1, 2, ..., N\}$ nodes representing N different institutions. We start with the following assumptions: 1) s_0 is secure, fault tolerant, and non-malicious; 2) h_i may be malicious, i.e., may try to infer information about data at other nodes and may upload false results (noise) into the system; and 3) malicious nodes do not collude and do not control the majority of the data across all the nodes. We first make the above assumptions and present our aggregation technique for generating a global model G from a set of locally trained models. We then present a more robust version of the system in which we relax the fault tolerance assumption of s_0 by replacing s_0 with a set $S = \{s_r : r = 1, 2, ..., R\}$ of R replicated coordinator nodes.

Model Aggregation: The goal is to generate a global predictor model $G(\mathbf{x}; \Theta) = \hat{\mathbf{y}}$ by aggregating a set of trained local predictor models $M = \{m(\mathbf{x}; \theta_i) = \hat{\mathbf{y}} : i = 1, 2, ..., N\}$ parametrized by $\Theta = \{\theta_i : i = 1, 2, ..., N\}$, where each θ_i is independently trained on a corresponding fraction of the data $X = \{\mathbf{x}_i : i = 1, 2, ..., N\}$ and labels $Y = \{\mathbf{y}_i : i = 1, 2, ..., N\}$ in the system. In this context, training entails finding parameters θ_i that minimize the error $\epsilon(\hat{\mathbf{y}}_i, \mathbf{y}_i)$ between label $\mathbf{y}_i$ and prediction $\hat{\mathbf{y}}_i$; this process may be expressed as

$$\underset{\theta_i}{\operatorname{argmin}}\ \epsilon(m(\mathbf{x}_i; \theta_i), \mathbf{y}_i). \tag{1}$$

The global model $G(\mathbf{x}; \Theta, W, D) = \sum_{\forall i} w_i m(\mathbf{x}; \theta_i) / \sum_{\forall i} w_i$, is defined as a weighted average of independently trained local models, where the weights $w_i \in W$

$$w_i = d_i \sum_{\forall j} e^{-R_i(\epsilon(m(\mathbf{x}_j; \theta_i), \mathbf{y}_j))} \tag{2}$$

are obtained from a combination representing the sample size $d_i \in D$ of local data $\mathbf{x}_i$ on which θ_i was trained and an exponential loss computed by a

[†] Open-source code available at: `https://github.com/DistributedML/InsuLearn`.

ranking function $R_i : \epsilon(m(\mathbf{x}_j;\theta_i),\mathbf{y}_j) \subset \mathbb{R} \to \mathbb{N}$, which captures the predictive performance of parameters θ_i on data $\mathbf{x}_j$ of node h_j using an error metric ϵ. To eliminate the contribution of poor performing or potentially malicious models, every weight w_i that is below the median value of all weights W is set to zero.

General System Protocol: The proposed aggregation approach requires all θ_i models to be cross-validated using the data $\mathbf{x}_i$ from all N nodes, which can be stored in a mapping of $E(i,j) : \mathbb{R}^{N\times N} \to \epsilon(m(\mathbf{x}_j;\theta_i),\mathbf{y}_j)$. E should not be made available to h_i as it may be malicious and use $E(i,j)$ to make inferences about the data $\mathbf{x}_j$ at h_j. We use the coordinator node s_0 as an intermediary for all interactions between H. This simplifies the burden and dependency between H as they become stateless (i.e., are not required to record preceding events in their interaction with the coordinator) by shifting all state information, aside from the state of sensitive data stored at each node, to s_0. Therefore, a node h_i in the system can only perform the following interactions with the coordinator:

1) generate a new model and send $\{d_i,\theta_i\}$ to s_0;
2) request and receive global model G from s_0; and
3) receive and test incoming model parameters θ_* from s_0 by computing the error $\epsilon(m(\mathbf{x}_i;\theta_*),\mathbf{y}_i))$ and communicate the computed error back to s_0.

Since, h_i cannot associate θ_* with any other node in H[‡], it cannot compromise the privacy of the other nodes. On the other hand, the stateful coordinator s_0 must maintain all state information. Specifically, s_0 must do the following:

1) maintain a database of participating nodes H and associated $\{\Theta,D,E,W\}$;
2) receive θ_i from h_i, anonymize $\theta_{i\to *}$ and forward θ_* to $h_j, \forall h_j \in H, j \neq i$;
3) receive $\epsilon(m(\mathbf{x}_j;\theta_*),\mathbf{y}_j)$ from h_j and update $E(*\to i,j)$;
4) commit changes to $\{\Theta,D,W\}$ after E is completed; and
5) receive global model G requests from h_i, generate G, and send G to h_i.

The singular s_0 coordinator is impractical for real applications. If s_0 fails, all state information and progress by the system will be lost. Next, we detail improvements to achieve stronger liveness guarantees in InsuLearn.

Fault Tolerance: We make the coordinator fault tolerant by replacing s_0 with S, a cluster of R coordinators [9] that replicate copies of state $\{\Theta,D,E,W\}$. This replicated version of InsuLearn can survive up to $\lceil R/2\rceil - 1$ coordinator failures (a necessary failures upper-bound for all majority-based consensus protocols). InsuLearn coordinators use the Raft consensus algorithm [10] to guarantee liveness and maintain strong consistency of replicated state. We use Raft because it has several well-tested, industry-standard, and open-source implementations.

Raft is a leader-based consensus algorithm. A leader is elected from S whose commands are confirmed and replicated by other replicas in S. In case a leader fails, Raft will elect a new leader without compromising liveness and the consistency of $\{\Theta,D,E,W\}$, which are replicated by all of the nodes in S.

Nodes contact any of the replicated coordinators to join the system. In case of a coordinator failure a node transparently switches to another coordinator. New nodes can join the system at any time.

[‡] In fact h_i does not know the size of H nor the nodes in H.

Partial Model Update: The coordinators can process incoming local models $\{d_i, \theta_i\}$ and requests for G asynchronously as changes to $\{\Theta, D, W\}$ and aggregation into G is performed only by the leader. The leader coordinator does not commit changes to $\{\Theta, D, W\}$ until E is completed; meaning that θ_i in Θ and associated $\{W, D\}$ are not replaced or set to zero when a new θ_i is submitted by h_i until $\{d_i, \theta_i\}$ is tested by all other nodes in H.

InsuLearn incentivizes nodes to test the models of other nodes. It maintains a *testQueue* in S that prevents h_i from submitting new models until h_i has performed all of the pending tests. During operation, some of the test nodes may go offline or be slow to test θ_* (e.g., as a denial of service attack by a malicious h_i). In this situation, delaying the commit to $\{\Theta, D, W\}$ until all test node results are available may indefinitely block the system. InsuLearn coordinators overcome this problem by actively computing a running sum of test data size d_i over all nodes responding to tests on $\theta_{i \to *}$. A *partial commit* of a model update may then be performed if the associated sum exceeds a predetermined fraction, e.g., half of the total available distributed data $\sum_{\forall i} d_i$, followed by additional partial commits with subsequent updates to $E(i, .)$.

3 Results and Discussions

To assess the validity of the learned global model G, we test InsuLearn on a classification task using a skin segmentation dataset and a regression task using real Parkinson's telemonitoring data; both of which are publicly available from the UCI ML repository [11]. The skin segmentation dataset consists of a large binary classification problem containing 245,057 instances (50,000 of which were randomly selected for our experiments) with 3 attributes, and the telemonitoring data consists of 5,875 instances with 26 attributes. For both tasks 25% of the instances are randomly selected for testing and the remainder is used for training. Training set instances are randomly split across $N = 20$ nodes and then the training data is augmented with an additional 25% of random noise instances, which are distributed across the nodes.

We assess trade-offs associated with secured distributed learning by comparing regression and classification errors of our proposed approach to a naïve secure aggregation approach (where the weights are equal $w_i = 1/N$) and an unsecure warehouse approach (where we simply pool all data prior to training). Ten different built-in MATLAB classification and regression methods are used in our evaluation and the test errors are computed on 100 randomly distributed datasets, averaged, and presented Fig. 1. To test the effects of a deliberate attack on the system, we perform our experiments on (i) a case where noise is randomly distributed across all nodes (results shown in Fig. 1), and on (ii) a case where noise is injected into a single malicious node (results in supplementary material).

Model Aggregation Performance: The test results clearly demonstrate a reduction in test error with InsuLearn's aggregation scheme over the naïve approach for all learning models except for decision tree (DT) regression (C1 in Fig. 1). In each experiment we also computed the total number of nodes whose

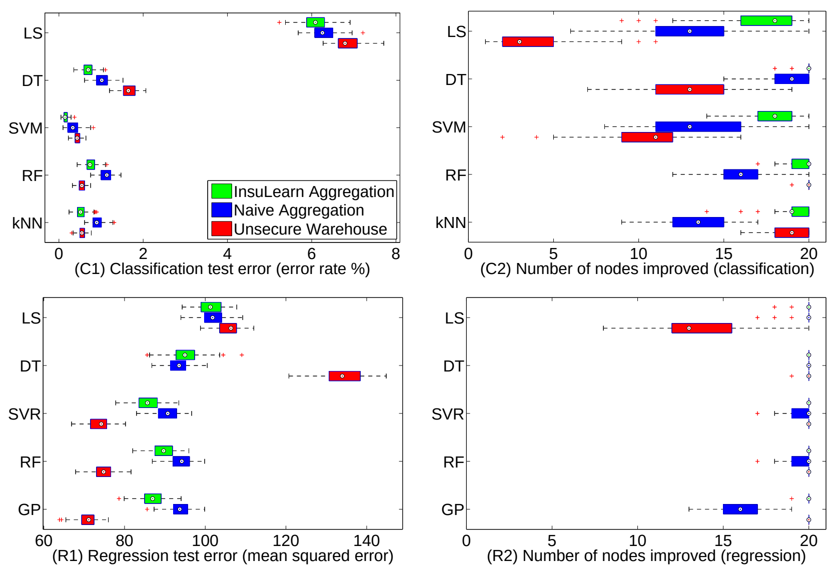
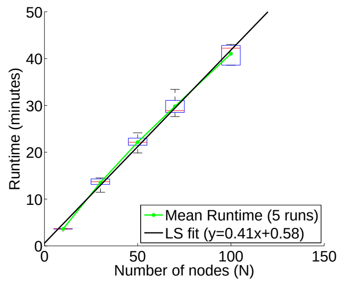


Fig. 1: Performance of three global model aggregation methods. Each method is tested using **(C1)** five different MATLAB classification and **(R1)** five regression functions on real data (randomly redistributed over 20 nodes and averaged over 100 trials). At each trial, the total number of nodes whose local model was improved upon by the global **(C2)** classification and **(R2)** regression model is also computed and presented. Results indicate that our proposed aggregation outperforms the naïve method.

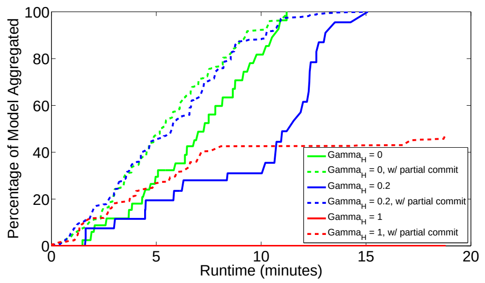
local model was improved upon by the global model (C2 and R2 in Fig. 1) and observed that, compared to the naïve global model, more nodes benefited from the global model learned by InsuLearn. InsuLearn outperforms the warehouse approach on simpler models, i.e., least squares (LS) and DT, but this is due to the fact that LS and DT are weak learners that are improved by ensembling. Compared to more advanced methods that already incorporate ensembling like random forests (RF), kernel transformations like support vector regression (SVR), and Gaussian process (GP) optimization, the performance with InsuLearn is no better than warehousing. This is expected because of an unavoidable trade-off between data privacy and accuracy of the estimated model. In case of bagging, for example, the trade-off manifests in a loss of the ability to train on bootstrap samples from data that spans across multiple nodes. It is important to note that an improvement over locally trained models are not guaranteed, even with advanced methods implemented on the unsecure warehousing approach (e.g., RF classification on unsecure warehouse, Fig. 1 C2).

It is also important to note that our proposed approach is generic and can be applied to *all* classification and regression methods. Tailoring the aggregation scheme to each method may improve performance but our generic approach is sufficient to aggregate the different local models into one.

As expected, the distribution of noise did not effect the warehouse approach. On the other hand, while our proposed method performs marginally better in LS



(a) Scalability



(b) Fault tolerance and partial commits

Fig. 2: **(a)** Scalability of InsuLearn with number of nodes, and **(b)** the advantage of partial commits during node failures.

tests when noise is injected at a single node, performance drops for more complicated models when noise is not randomly distributed. However, the performance of our proposed method is still better than the naïve approach.

Scalability: We simulated a real deployment of InsuLearn by deploying it on the Microsoft Azure cloud to measure scalability and fault tolerance. Coordinators and nodes were deployed, each in an independent, identical `DS1_V2` virtual machine with 1 core and 3.5 GB of memory and connected over a virtual network. To test the scalability of the system, we set the number of coordinators $R = 7$ and scaled the total number of emulated institutions $N = \{10, 30, 50, 70, 100\}$. The behavior at each node was automated with a state machine that restricts the node to only submit its local model once to the system. Nodes were added to the system with a randomized $\delta \in (0, 2)$ minute delay from the start of the coordinators. We measured the time it took to generate a complete global model, encapsulating models from all N nodes, over 5 independent runs and observed linear performance scaling with N (Fig. 2a). Model testing at nodes is independent and nodes can perform training in parallel. Idempotent communication and commutative update operations protect the system from a large number of highly concurrent nodes and improves InsuLearn's scalability.

Fault Tolerance To assess fault tolerance, we fixed $R = 7$ and $N = 30$, updated the state machines to emulate random intermittent fail-and-restart with frequency γ_H for nodes that have pushed their models to the coordinators, and measured progress in terms of percentage of global model G committed over the run of the system (Fig. 2b). We tested the system for cases where $\gamma_H = \{0, 0.2, 1\}$ failures per minutes and with/without the partial commit feature to see how partial commits affect the global model update rate. We observed that once the fail-and-start frequency surpasses a certain rate, progress in the system halts as nodes leave as soon as they submit their local models while there is not sufficient time to train on new models before the next failure. However, with the partial commit feature, a significant portion of the change to G is preserved.

Security: InsuLearn assumes that coordinators in S are secure. Although coordinators only know the results of the tests and sources of committed model, they can obtain more knowledge by manipulating test requests. For example, a rogue coordinator can ask a node h_i to test a number of malicious models and thereby infer information about data at h_i. Solving this problem requires nodes to detect and reject suspicious requests. One solution is to use a Byzantine fault tolerance algorithm such as PBFT [12]. In this approach a node checks that a consensus among coordinators has been reached prior to testing the model. A PBFT-based system with $3t$ coordinators can withstand t malicious coordinators. However, the extra rounds of communication would impose a substantial performance penalty.

4 Conclusions

We presented a fault tolerant distributed PMPML system called InsuLearn in which, with the exception of the anonymized local model, no information from one node is shared with other nodes. We compared InsuLearn against an unsecure warehousing and naïve model averaging approaches and showed that InsuLearn is fault tolerant and has reliable performance, surpassing a naïve aggregation approach. Our work shows the feasibility of ML on highly sensitive distributed medical image data. We hope that our open-source GitHub project will be adopted by others and encourage the development of an accessible and secure distributed ML toolkit that can facilitate medical research that is otherwise impractical.

References

1. Li, Y., Bai, C., Reddy, C.K.: A distributed ensemble approach for mining healthcare data under privacy constraints. Information Sciences **330** (2016) 245–259
2. Ohno-Machado, L.: To share or not to share: that is not the question. Science Translational Medicine **4**(165) (2012) 165cm15
3. Fabian, B., Göthling, T.: Privacy-preserving data warehousing. Int. J. Bus. Intell. Data Min. **10**(4) (October 2015) 297–336
4. McMahan, H.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. (2016)
5. Hamm, J., Cao, P., Belkin, M.: Learning privately from multiparty data. In: International Conference on Machine Learning. (2016) 555–563
6. Xie, L., Plis, S., Sarwate, A.D.: Data-weighted ensemble learning for privacy-preserving distributed learning. In: ICASSP, IEEE (2016) 2309–2313
7. Wu, Y., Jiang, X., Kim, J., Ohno-Machado, L.: Grid Binary LOgistic REgression (GLORE): building shared models without sharing data. Journal of the American Medical Informatics Association **19**(5) (2012) 758–764
8. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Computer and Communications Security, ACM (2015) 1310–1321
9. Schneider, F.B.: Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Computing Surveys **22**(4) (1990) 299–319
10. Ongaro, D., Ousterhout, J.K.: In search of an understandable consensus algorithm. In: USENIX Annual Technical Conference. (2014) 305–319
11. Lichman, M.: UCI machine learning repository (2013)
12. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems **20**(4) (November 2002) 398–461