# On Outsourcing Artificial Neural Network Learning of Privacy-Sensitive Medical Data to the Cloud


1st Dimitrios Melissourgos
*University of Florida*
Gainesville, FL, USA
dmelissourgos@ufl.edu

2nd Hanzhi Gao
*University of Florida*
Gainesville, FL, USA
gaohanzhi@phhp.ufl.edu

3rd Chaoyi Ma
*University of Florida*
Gainesville, FL, USA
ch.ma@ufl.edu

4th Shigang Chen
*University of Florida*
Gainesville, FL, USA
sgchen@cise.ufl.edu

5th Samuel S. Wu
*University of Florida*
Gainesville, FL, USA
samwu@biostat.ufl.edu



***Abstract*—Machine learning and artificial neural networks (ANNs) have been at the forefront of medical research in the last few years. It is well known that ANNs benefit from big data and the collection of the data is often decentralized, meaning that it is stored in different computer systems. There is a practical need to bring the distributed data together with the purpose of training a more accurate ANN. However, the privacy concern prevents medical institutes from sharing patient data freely. Federated learning and multi-party computation have been proposed to address this concern. However, they require the medical data collectors to participate in the deep-learning computations of the data users, which is inconvenient or even infeasible in practice. In this paper, we propose to use matrix masking for privacy protection of patient data. It allows the data collectors to outsource privacy-sensitive medical data to the cloud in a masked form, and allows the data users to outsource deep learning to the cloud as well, where the ANN models can be trained directly from the masked data. Our experimental results on deep-learning models for diagnosis of Alzheimer's disease and Parkinson's disease show that the diagnosis accuracy of the models trained from the masked data is similar to that of the models from the original patient data.**



***Index Terms*—Neural network privacy, Orthogonal transformation, Matrix masking, Medical data privacy**


## I. Introduction

Artificial intelligence is transforming biomedical research and medical practice [1], [2]. Machine learning has been used for prediction, classification, and statistical inference, often outperforming human experts in diagnosis [3]. Deep-learning artificial neural networks (ANN) have become increasingly popular in medical research [4]–[10]. Studies have shown that ANNs provide better performance than conventional machine learning algorithms and linear regression methods [11]–[14].

While big medical data present unprecedented opportunities for building fine-grained ANN models for medical research and practice, the tasks of training and continuously refining ANN models with data from tens of thousands of patients, each with hundreds or thousands of attributes including possibly images and genetic data, are notoriously computation-intensive and time-consuming [15], [16]. Outsourcing such computation and the data to the cloud is an obvious solution. The problem is that exposing sensitive patient data to the cloud admins and others with access to the cloud storage (possibly through illegal means such as cyber-attacks) may lead to incompliance with the expansive laws and regulations that govern medical data privacy. Encryption could be the answer. Most prior efforts focused on using homomorphic encryption to perform *inference* (such as classification) based on ANN models that have already been trained [17]–[22]. The key problem of outsourcing the expensive *training* operations to the cloud, without leaking any patient data from their distributed sources, remains open, not only because the computation over homomorphically encrypted data is expensive and the ANN models are non-linear, but also because it is difficult to compute over medical data that are encrypted from multiple sources with different keys.

One solution of privacy-preserving deep learning over distributed data is federated learning [23]–[27] or more broadly multi-party secure computation [28]–[31]. However, they require all data collectors (called *data clients* in this paper) to perform synchronized computations together. Take federated learning as example. It requires all clients to synchronously train their models locally based on their raw data and send the gradients of the model parameters after each local training iteration to a centralized server (or cloud), where the gradients from all clients are combined to update the model parameters, which are then distributed to the clients in order to update their local models for the next training iteration [32]. This model requires all data clients (i.e., data-collecting doctors and medical researchers) to acquire the computing resources and the technical expertise for local deep-learning computation, which is opposite from the desire of outsourcing such computation to the cloud, as discussed earlier. Moreover, it is highly inconvenient that whenever a biostatistician wants to perform a deep-learning study, all data clients (i.e., data-collecting doctors and medical researchers) need to be summoned to participate and carry out their respective local computation.

This paper sets forth the following requirements for cloud-based ANN learning of privacy-sensitive medical data:

1) *Outsourcing Requirement:* The data clients outsource the job of ANN training to the cloud, and they are relieved from local model training. After contributing their data, they are not required to participate in the computation of future data analyses.
2) *Data Privacy Requirement:* Data should be stored at the cloud in a privacy-preserving form. If the cloud is compromised, the medical information of individuals should not be leaked.
3) *Efficiency Requirement:* The efficiency of training ANN

models over the privacy-protected data should be comparable to that of training ANN models over the raw data.

Homomorphically encrypted data cannot meet the efficiency requirement and the outsourcing requirement because of the difficulty of computing over data encrypted by different clients with different keys. This paper proposes a new method of cloud-based ANN training with *masked data*. *Matrix masking*, formulated by [33]–[39] with provable privacy, alters the components of a data matrix by performing orthogonal transformations. It has been used in privacy-preserving statistical analyses including linear regression, contingency table analysis, Cox proportional hazard regression, and logistic regression. To the best of our knowledge, this paper is the first to investigate the use of matrix masking as an effective privacy-preserving technique for cloud-based ANN training. We show theoretically that the difference between the cost function of the ANN model trained over the masked data and the cost function of the model over the raw data is bounded by a constant. We perform experiments based on two real data sets, one for Alzheimer's disease [40] and one for Parkinson's disease [41], to train diagnosis ANN models over masked data on the cloud and compare them with the benchmark models trained from the raw data. The diagnosis performance of the models trained with the masked data was similar to the performance of those trained with the raw data. Moreover, the runtime of training ANN models over the masked data is similar to doing so over the raw data, while data masking incurs small computational overhead.

## II. Related Work

Various methods have been proposed for multiple parties to pool their data together for deep learning while preserving data privacy. They can generally be grouped into three categories.

1) *Privacy-preserving inference:* While the ANN model is trained with raw data, the inference uses encrypted data, such that a user can use the model for inference without leaking its input data.
2) *Federated learning:* The ANN is trained jointly from multiple clients by sending their local parameters, such as gradients, to the cloud, which synthesizes the local values and sends the results back.
3) *ANN training on encrypted data with a trust authority:* The ANN is trained on data encrypted with credentials from a trust central server.

The privacy-preserving inference is typically performed over homomorphically encrypted user input, and the result (in the encrypted form) is sent back to the user, which decrypts for the actual model output [17]–[22]. These studies do not consider training the ANN models on encrypted data. Nor do they meet our efficiency requirement.

In federated learning [23]–[27], the data clients perform local ANN training themselves, which does not meet our outsourcing requirement. Moreover, all clients have to perform training together, which may be inconvenient in practical settings when a large global consortium of medical researchers who share data may work at different schedules on different research works with different expertise. The same issues exist for multi-party computation and garbled circuits [28]–[31], where the data clients do not outsource their local computations that often incur high communication and computation overhead [42].

CryptoNN by [43] relies on a centralized trusted authority (TA) to distribute its public key to all clients for encrypting their data. Establishing and maintaining a TA is not always feasible in practice. If the TA is compromised, all data may be leaked. In addition, training over encrypted data is far more expensive than doing so over the raw data.

Outsourcing the training operations to the cloud, without leaking any raw data, remains an open problem under the three requirements listed in the introduction.

## III. Preliminaries

### A. System Model

Consider an example of collecting the Alzheimer's disease (AD) patient data from a consortium of medical centers, hospitals and clinics in order to build deep-learning diagnosis models that are trained based on the collected patient data and are used by taking new patient information as input and classifying new patients as having AD or not. There is extensive work on building such a model based on a centralized data set [44]. However, if the data is scattered at different medical centers/hospitals/clinics and cannot be directly shared due to patient privacy restrictions, the prior work based on a centralized data set cannot be applied.

We assume that not all doctors and medical researchers in the consortium have access to adequate computing resources or deep-learning expertise to carry out local ANN model training or other multi-party computation for federated learning. It is desired to outsource data and computation and to leverage software and hardware from an MLaaS (Machine Learning as a Service) cloud provider, such as [45], [46] or [47]. We assume that the data collection process spans across long time (e.g., multiple years) and globally, which makes it difficult to coordinate synchronized computations or establish a centralized trusted authority.

The data contributors such as the doctors/researchers at the medical centers/hospitals/clinics in the consortium of the above example are called the *data clients*. The computing/storage platform that the clients outsource their data to is referred to as the *cloud* in this paper. The biostatisticians that use all or part of the outsourced data to learn ANN models on the cloud are called the *data users*. Different users may choose to use different subsets of data (e.g., from different countries) or choose different attributes of the data (e.g., with or without MRI images) to train different models. The doctors that use the models to diagnose new patients are called the *model users*. The sets of data clients, data users and model users may overlap. For example, a team in a hospital may have members that collect data, members that build models, and other members that use the models to diagnose.

### B. Problem Statement

The problem we address in this paper is to outsource distributed patient data from multiple data clients to the cloud, where deep-learning models are trained from the outsourced data for disease diagnosis (for example, AD diagnosis), with three requirements: (1) Data is outsourced to the cloud in a privacy-preserving form such that raw data about individual patients will not be leaked even if access to the cloud storage is compromised. (2) All model training operations are performed by the cloud. The inference operations (actual diagnosis) may be performed by the model users if they download the models from the cloud. The inference accuracy of the models should be close to the accuracy of the benchmark models trained from the raw data directly. (3) The privacy-preserving model training should be efficient (or even comparable to training with raw data).

### C. Threat Model

*Compromised cloud.* We assume that the cloud cannot be fully trusted as its admins have access to the data, other internal threats to the data cannot be ignored, and its storage could be compromised by an outside adversary. The patient data are outsourced from the data clients to the cloud in a privacy-preserving form. Even if an adversary obtains the sourced data, it should not be able to learn the raw data of individual patients.

*Curious clients.* Other clients may be curious about learning the raw data from a target client. If clients must cooperate in transform their data together in a privacy-preserving form before outsourcing to the cloud, we must make sure that the intermediate results of this multi-party computation do not give any clue to the raw data. If each client independently produces its outsourced data based entirely on local secrets, then the threat from curious clients is minimized.

*Model poisoning.* A malicious client could potentially poison a deep-learning model with incorrect input. That could result in the model parameters being erroneously altered and consequently the model being less accurate.

## IV. Training Artificial Neural Network on Masked Data

We propose to outsource matrix-masked data from the clients to the cloud. To motivate for our solution, we give a toy example. Consider a patient record, which is a vector with a certain number of attribute values:

$$x = \begin{bmatrix} 70 & 1.8 & 80 & ... \end{bmatrix},$$

where the first attribute is the patient's age in years, the second attribute is the height in meters, and the third attribute is the weight in kilograms. If we scale this vector by a random factor (e.g. $a = 5$), then the input vector becomes

$$ax = \begin{bmatrix} 350 & 9 & 400 & ... \end{bmatrix}.$$

On the one hand, the masked patient record does not have a direct meaning to an adversary that might observe it (e.g. no human reaches the age of 350 years). On the other hand, an ANN is able to extract the same features from $x$ or $ax$ as scaling the vector does not change the relationships between its attributes.

However, for simple scaling, it is often possible for the adversary to guess the scaling factor $a$ and recover approximate values of the elements in vector $x$, particularly when some attributes are restricted in their value ranges.

We need a more secure way of preserving the relationship between the attributes and a more robust way against guessing. Matrix masking provides a solution to this problem. It operates on many patient records (denoted as $X$) together, where $X$ is a matrix whose rows are patient records and columns are attributes. The multiplication of $X$ by an orthogonal matrix $A$, i.e., $AX$, preserves the angles and distances of the column vectors (attributes) in $X$ between vectors. $AX$ keeps the first and second moment statistics of $X$, providing adequate information for ANN model training. A random orthogonal matrix alters the values in each patient record, while preserving the relationship between the attributes. A large orthogonal matrix is practically impossible to guess as has been proven in [48] and [49], which showed that large random matrix masking $AX$ is computationally secure in protecting the privacy of data $X$.

We now present how each data client $i$ performs random orthogonal matrix masking to its patient data $[X_i, y_i]$, before sending the masked data to the cloud, where $X_i$ is the matrix of patient records and $y_i$ is the response vector — using Alzheimer's disease (AD) as example, for each patient in $X_i$, the corresponding element in $y_i$ is zero for non-AD and one for AD. We stress that data masking is done once by the client before the masked data is outsourced to the cloud. After that, the data client is not involved when any data user uses the data to train an ANN model. Let $n_i$ be the number of patient records, i.e., the number of rows in $X_i$. The process for generating a random orthogonal matrix $A_i$ is as follows: First, using the Gram-Schmidt process, we find an orthogonal basis $B_i$ of the vector space generated by the vectors we want to keep invariant. Specifically, we want to keep invariant the vector of $y_i$ and the vector of ones. Hence, $B_i$ has two column vectors. We then randomly generate $n_i - 2$ column vectors, concatenate them to $B_i$ for an $n_i x n_i$ matrix $C_i$, and perform QR factorization on $C_i$, which gives us two orthogonal matrices, $Q_1$ and $Q_2$ [36]. We let $A_i = Q_1 Q_2^T$. Therefore, $A_i$ is a matrix that can perform orthogonal transformations to the patient data $[X_i, y_i]$ and also has the properties $A_i y_i = y_i$ and $A_i 1_n = 1_n$, where $1_n$ is a vector of $n$ ones. Note that each client can generate its own data masking matrix $A_i$. The clients do so independently of each other.

Each client sends their masked data set $[A_i X_i, y_i]$ to the cloud, which aggregates them together by stacking them vertically. For $m$ data clients, the cloud gathers

$$[AX, y] = \begin{bmatrix} [A_1 X_1, y_1] \\ [A_2 X_2, y_2] \\ \vdots \\ [A_m X_m, y_m] \end{bmatrix}, \tag{1}$$

where

$$A = \begin{bmatrix} A_1..... \\ ...A_2... \\ \vdots \\ .....A_m \end{bmatrix} \text{ and } X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \quad (2)$$

Any data user can now use all or a portion of $[AX, y]$ to train an ANN model (such as feed-forward neural networks in our experiments) in the cloud. For example, one may use subsets of attributes to build and compare models to learn each attribute relevance to inference accuracy. After a model is trained, the user downloads the model and apply it to new patient records (which are not masked). The user may also perform privacy-preserving inference in the cloud [17]–[22].

Will the model trained from the masked data be similar to the model trained from the raw patient data? We provide our intuition below, with theoretical and experimental results later. Consider the raw data matrix $X_i$ at the $i$th data client with $n_i$ patient records and the random mean-invariant orthogonal masking matrix $A_i$. Each row in the masked data $M_i = A_i X_i$, called a *masked record*, is a weighted linear sum of all patient records in $X_i$, where $A_i$ specifies the weights — its $i$th column specifies the weights for the $i$th masked record in $M_i$. Every patient record contributes a share (fraction) of itself to every masked record. The $i$th row in $A_i$ specifies the shares (fractions) of the $i$th patient record in $X_i$ that are contributed to the masked records in $M_i$. Because the sum of each row in $A_i$ is one, all the fractions of a patient record contributed to $M_i$ adds up to its whole. In that sense, $M_i$ carries the same amount of patient information as $X_i$ does, albeit in a random mixed form. When we feed a row vector from $M_i$ to a neural network, we are actually feeding $n_i$ fractional raw patient records simultaneously. If the neural network were entirely linear and treated each input sample independently, we would see the same model parameters trained either from $X_i$ or from $M_i$. A feed-forward neural network treats input samples independently, but its activation functions are not linear. Due to this non-linearity, the model parameters trained from $M_i$ may not be identical to those from $X_i$, depending on the choice of activation function. But they could very well be similar, as our experiments have clearly suggested.

To guard against model poisoning where a malicious attempts to poison a deep-learning model with incorrect patient data, we introduce a data verification mechanism to each client's masked data. Consider the masked data $M_i$ from an arbitrary client. First, we build a model with the masked data from all other clients, and test the model's accuracy with a test data set (which could be de-identified raw data from a source that allows the outsourcing). Second, we build another model with the masked data from all clients (including $M_i$), and test the model's accuracy. If the model with $M_i$ is significantly worse than the model without $M_i$, we reject $M_i$.

## V. Model from Masked Data v.s. Model from Raw Data

It has been proved in [48], [49] that the masked data can achieve strong privacy without leaking information about individual patients in the raw data. More specifically, suppose an adversary has obtained the masked data $M$ and let $X$ be a random variable for any raw data that can produce the masked data. Under certain easy-to-satisfy conditions, the restricted support of $X$ will be practically intractable and that its posterior probability density will remain the same as its prior probability density over the restricted support, which means that the information learned from the masked data does not improve the knowledge of the adversary about which possible values of the patient records are more likely.

Below we show that masking does not significantly change the utility of the data in ANN model training. More specifically, we want to show that the ANN model $N$ that is trained with the raw data is similar to the ANN model $\hat{N}$ trained with the masked data. This property comes from the fact that the difference between the cost functions of $N$ and $\hat{N}$ is bounded by a constant, while our experiments will demonstrate that the bound is tight as the two models perform similarly. The proof of the theorem below can be found in the appendix.

**Theorem V.1.** *Let $X$ be a full-rank matrix and $y$ be a response vector in the data set $[X, y]$. Let $A \in \mathbb{R}^{n \times n}$ be an orthogonal matrix that satisfies $Ay = y$ and $1_n^T A = 1_n^T$, where $1_n^T$ is the transpose of a vector of $n$ ones. Let $C$ and $\hat{C}$ be the cost of the ANN after feeding the matrix $[X, y]$ and $A[X, y]$, respectively. Then, the difference of $C$ and $\hat{C}$ is bounded, i.e., $|C - \hat{C}| \leq K$, where $K$ is a constant depending on the raw data and the masking matrix, and $|\cdot|$ is the element-wise absolute value function.*

## VI. Experimental Evaluation

We conduct extensive experiments to compare the accuracy of a model that has been trained on raw data to the accuracy of a model that has been trained on masked data. We use the Alzheimer's Disease Neuroimaging Initiative (ADNI) data set [40] and the Parkinson's Outcome Project (POP) data set [41]. In this section, we first present the data sets and the experimental setup, including the experimental method and the ANN hyper-parameters. Afterwards, we demonstrate the experimental results for binary classification.

### A. Data Sets

We use two separate data sets for the experiments. The first one is the Alzheimer's Disease (AD) data set [40], which has 4 attributes in matrix form and MRI images of the brain of the patients in image form. The 4 attributes in the matrix form are the following: age in years, gender with 0 for males and 1 for females, MMSE score (Mini-Mental State Examination) in the range $[18, 30]$ and APOE (Apolipoprotein E gene) with 0 for non-existence of the gene and 1 for existence. The distributions of the attributes are shown in Figures 1 - 4, and the distribution of the response variable is shown in Figure 5, where AD denotes

Alzheimer's disease and NL denotes normal cognition. The MRI images are Axial PD/T2 FSE images of the brain of the patients as shown in Figures 6 and 7. We use the min-max normalization method for this data set as well.

Additionally, we perform experiments on the combination of matrix and image data of the AD data set. The purpose of this experiment is to show that our method can work with image data as well as matrix data and that their combination can produce a more accurate ANN than each data set separately. We transform the image data into matrix form before combining them with the non-image data.

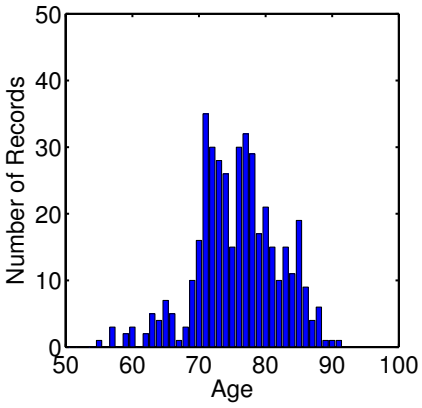



Fig. 1: The age attribute distribution of the ADNI data set.

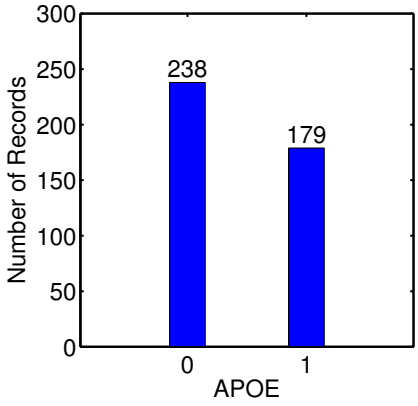



Fig. 2: The APOE attribute distribution of the ADNI data set.

The second data set is based on the Parkinson's Outcome Project (POP) [41]. We have 4908 patients with 13 attributes extracted from their POP Data Collection Form and we generate a data matrix, where each patient's record forms a row with 13 columns. We use the Hoehn and Yahr stage (HYstage) as the response, which is a categorical variable between 1 and 5, showing how much the disease has progressed. Since we perform binary classification, we dichotomize this variable by transforming it into a binary attribute where the response is 0 when the HYStage is 1 or 2 (i.e. mild cases of PD) and the response is 1 when the HYStage is 3, 4 and 5 (i.e. severe cases of PD).

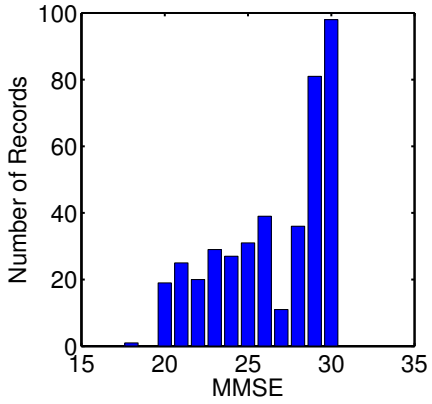



Fig. 3: The MMSE attribute distribution of the ADNI data set.

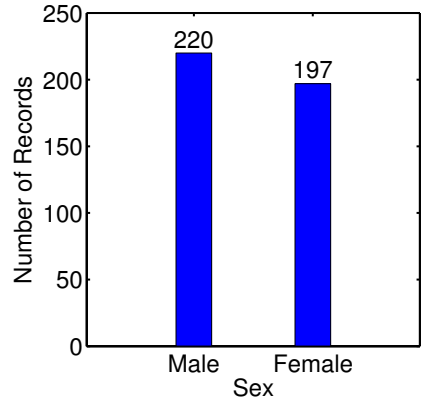



Fig. 4: The sex attribute distribution of the ADNI data set.

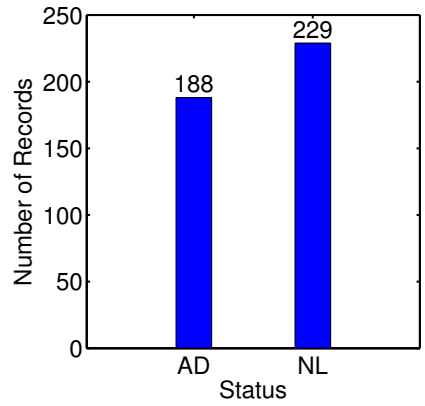



Fig. 5: The status distribution of the ADNI data set.

After extracting the response vector, there are 12 attributes, which are either categorical or numerical. Some of the attributes are age, sex, ethnicity, disease duration, cognitive score and others. For each categorical attribute, we transform it into one or more attributes with binary values (0 or 1) with one-hot encoding. More specifically, for a categorical attribute with only 2 categories, we transform it into a binary attribute where 0 stands for the first category and 1 stands for the second category; for a categorical attribute which has more than 2 categories, we

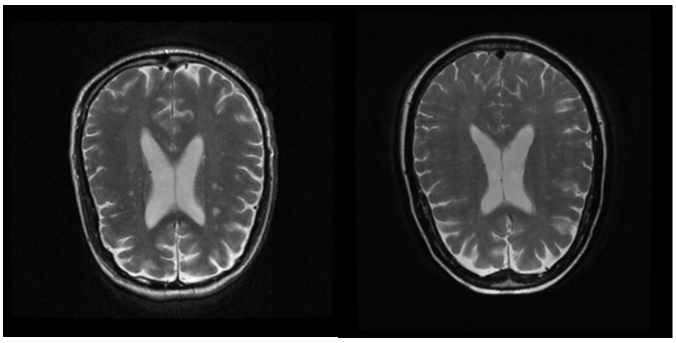


Fig. 6: MRI images of two patients without AD.

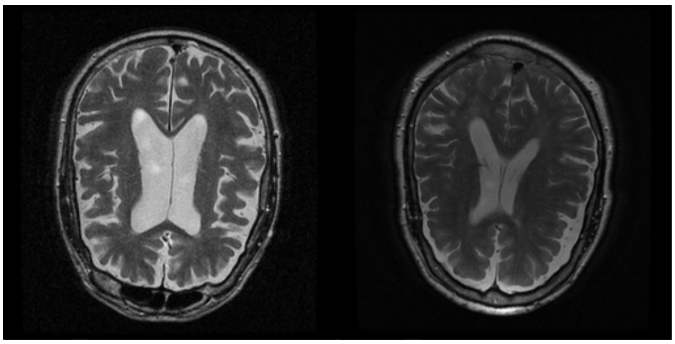


Fig. 7: MRI images of two patients clinically diagnosed with AD.

transform it into $p$ binary attributes where the $i^{th}$ attribute is 1 when and only when it belongs in the $i^{th}$ category. With this transformation, the 12 attributes are extended to 36 attributes. It is well known that one-hot encoding can provide better linearity and improve the network's accuracy. We observed higher accuracy by using one-hot encoding as well. Finally, we use the min-max method to normalize each attribute into the range $[0, 1]$, which also improves linearity.

We split the data set into two sub-sets, where the first contains approximately 80% of the data for training and the second contains 20% of the data for testing. The training data set contains 3908 rows, from which 1954 have a response of 0 and 1954 have a response of 1. The test data set contains 1000 rows from which 500 have a response of 0 and 500 have a response of 1.

### B. Experimental Method

We use a powerful server equipped with an 8-core CPU at 3.7GHz, 16GB of RAM and a 2080TI GPU, to simulate the cloud, which can train an ANN model in a timely manner, due to the powerful GPU. We also simulate several common desktops to act as the clients. Each client has a training data set and a testing data set, which are part of the data sets in Section VI-A.

In our experiments, we compare the accuracy of the trained ANN using raw and masked data. We first let the client send the raw training data to the server and, in turn, the server returns the raw-trained model to the client, which is denoted as $M$. Then, we let the client mask the training data using the proposed method, send it to the sever and the server returns the masked-trained model to the client, which is denoted as $\hat{M}$. Finally, we test and compare the accuracy of the two trained models using the same raw testing data at the client end. The aforementioned results are presented in Figures 8 and 9 for the ADNI and PD data sets, respectively.

### C. Artificial Neural Network Architecture

We use a basic feed-forward neural network (FFNN) in our experiments with binary crossentropy loss function. Additionally, we use the "adam optimizer" in keras library of python as the stochastic gradient descent optimizer for both training and testing, ReLU as the activation functions for all hidden layers and sigmoid as the activation function for the output layer. We run the experiments for 400 epochs and use batch size of 50. The number of the neurons in each layer depends on the input vector size. More specifically, the ANN for the AD data set has three hidden layers. The number of neurons in each hidden layers is 4 for the non-image data, 117 for the MRI image data and 121 for their combination. The PD data set has 36 neurons in each of the three hidden layers. We vary the number of clients for accuracy comparison under these settings.

### D. Results for Binary Classification

The response vector is binary, that is, it only has values 0 or 1 as explained in Section VI-A. We vary the number client machines between 1 and 10 for the AD data set, having 35 patients each. We also vary the number of clients between 1 and 78 for the PD data set, having 50 patients each. From Figures 8 and 9 we can make two significant observations. First, the accuracy increases monotonically with the increase of the number of clients. Second, the accuracy of the masked-trained model $\hat{M}$ is comparable to the accuracy of the raw-trained model $M$.

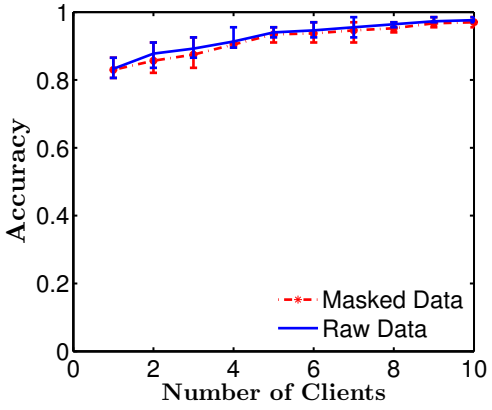



Fig. 8: Model accuracy w.r.t. number of clients in the ADNI data set.

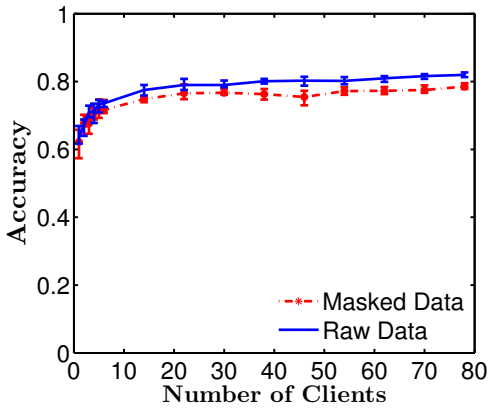



Fig. 9: Model accuracy w.r.t. number of clients in the PD data set.

### E. Results for Attribute Combination

In order to test whether our method works with image data and with combinations of attributes, we transform the brain MRI images of the patients into matrices and combine them with the existing four non-image attributes.

Figure 10 shows the performance of the ANN models with the ADNI data set on raw and masked data, without combining image and non-image data. On the contrary, Figure 11 shows the performance of the ANN with the same data set when we combine the image and matrix data in various configurations.

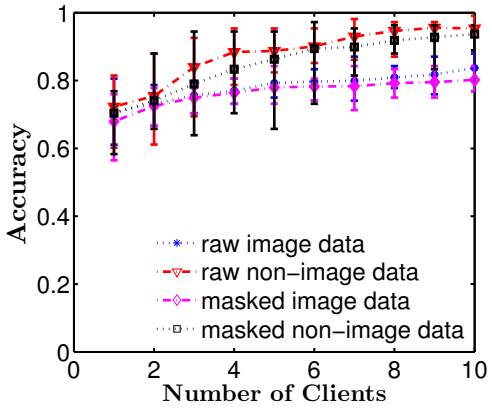



Fig. 10: Raw and masked data ANN models without attribute combination.

### F. Run Time of Masking

The masking process requires the generation of a masking matrix $A$ in each client, which is explained in Section IV. This process has the following two elements: 1) perform 2 QR decompositions with complexity $\mathcal{O}(n^3)$ and 2) perform a matrix multiplication between the generated random orthogonal matrix $A$ and the data set $[X_r, y_r]$, which has complexity $\mathcal{O}(n^3)$ as well. Therefore, the overall complexity of matrix masking is $\mathcal{O}(n^3)$.

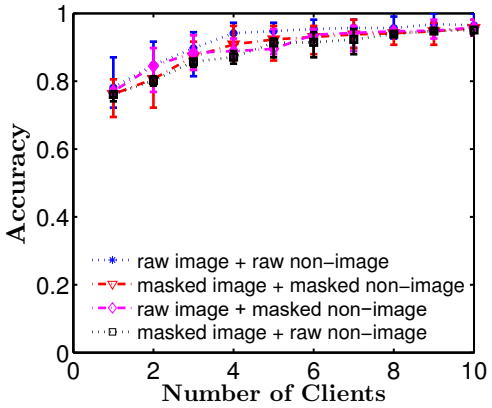



Fig. 11: Raw and masked data ANN models with attribute combination.

Although the runtime is exponential, Figure 12 shows that for data sets with hundreds or a few thousand records at each client, the masking operation is negligible compared to the training operation (e.g. a data set with 500 records requires only 48 milliseconds for mask creation and multiplication without the use of a GPU). In this experiment, we use mini-batches of size 50 and 400 epochs.

For larger data sets, the runtime complexity can be reduced with the use of block-diagonal masks. For example, instead of creating a $5000 \times 5000$ masking matrix, we can create 10 matrices of size $500 \times 500$, place them in the diagonal of the $5000 \times 5000$ matrix and fill the rest of the elements with zeros, i.e.,

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_m \end{bmatrix}.$$

That would reduce the masking time of a data set with 5000 records from 11.1 seconds to 0.48 seconds, since we only need to create 10 matrices of size $500 \times 500$ and each such mask takes only 48 milliseconds to create. Note that we used a CPU for matrix masking and a GPU for ANN training (which is significantly faster than a CPU). When a CPU is used for both operations, the masking time becomes even more negligible, since the ANN training would take considerably longer.

## VII. Conclusion and Future Work

In this paper we presented a new technique for outsourcing privacy preserving ANN training. We set forth four requirements that we believe to be important in order to create a practical system. Our technique is based on matrix masking, which performs orthogonal transformations to the data before it is sent from the clients to the cloud. The cloud trains an ANN model on the aggregate masked data it has received from the clients and gives it to all the clients to be used with new raw data. In this work, we argue theoretically and experimentally that the accuracy the clients will get from the masked-trained

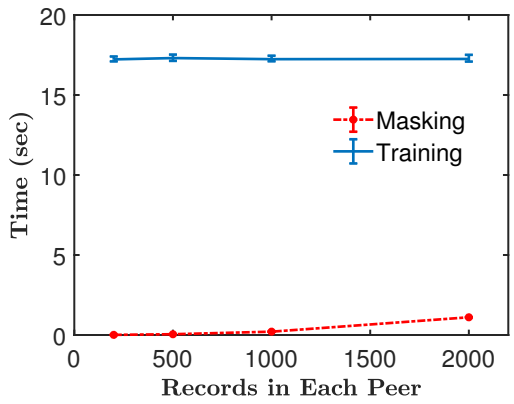



Fig. 12: Comparison between the runtime of the training process and the masking process.

ANN model is comparable to the accuracy they would get if the model was trained on raw data.

In the future, we plan to continue this work by exploring the capabilities of our technique on image classification (such as the MNIST data set) and Convolutional Neural Networks. We also plan to expand our work on regression and unsupervised learning.

## References


[1] Abhimanyu S Ahuja. The impact of artificial intelligence in medicine on the future role of the physician. *PeerJ*, 7:e7702, 2019.

[2] Nishita Mehta, Anil Pandit, and Sharvari Shukla. Transforming healthcare with big data analytics and artificial intelligence: A systematic mapping study. *Journal of biomedical informatics*, 100:103311, 2019.

[3] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

[4] Nida Shahid, Tim Rappon, and Whitney Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PloS one*, 14(2):e0212356, 2019.

[5] Abbas Sheikhtaheri, Farahnaz Sadoughi, and Zahra Hashemi Dehaghi. Developing and using expert systems and neural networks in medicine: a review on benefits and challenges. *Journal of medical systems*, 38(9):1–6, 2014.

[6] Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. Artificial intelligence in healthcare: past, present and future. *Stroke and vascular neurology*, 2(4), 2017.

[7] Sk Saddam Ahmed, Nilanjan Dey, Amira S Ashour, Dimitra Sifaki-Pistolla, Dana Bălas-Timar, Valentina E Balas, and João Manuel RS Tavares. Effect of fuzzy partitioning in crohn's disease classification: a neuro-fuzzy-based approach. *Medical & biological engineering & computing*, 55(1):101–115, 2017.

[8] Alain Yee-Loong Chong, Martin J Liu, Jun Luo, and Ooi Keng-Boon. Predicting rfid adoption in healthcare supply chain from the perspectives of users. *International Journal of Production Economics*, 159:66–75, 2015.

[9] Erhan Elveren and Nejat Yumuşak. Tuberculosis disease diagnosis using artificial neural network trained with genetic algorithm. *Journal of medical systems*, 35(3):329–332, 2011.

[10] Radhwane Benali, Fethi Bereksi Reguig, and Zinedine Hadj Slimane. Automatic classification of heartbeats using wavelet neural network. *Journal of medical systems*, 36(2):883–892, 2012.

[11] Wen-Jing Niu, Zhong-Kai Feng, Bao-Fei Feng, Yao-Wu Min, Chun-Tian Cheng, and Jian-Zhong Zhou. Comparison of multiple linear regression, artificial neural network, extreme learning machine, and support vector machine in deriving operation rule of hydropower reservoir. *Water*, 11(1):88, 2019.

[12] Necdet Süt and Mustafa Şenocak. Assessment of the performances of multilayer perceptron neural networks in comparison with recurrent neural networks and two statistical methods for diagnosing coronary artery disease. *Expert Systems*, 24(3):131–142, 2007.

[13] Gwang-Hee Kim, Jae-Min Shin, Sangyong Kim, and Yoonseok Shin. Comparison of school building construction costs estimation methods using regression analysis, neural network, and support vector machine. 2013.

[14] In-Su Han and Chang-Bock Chung. Performance prediction and analysis of a pem fuel cell operating on pure oxygen using data-driven models: A comparison of artificial neural network and support vector machine. *International Journal of Hydrogen Energy*, 41(24):10202–10211, 2016.

[15] Mark JJP Van Grinsven, Bram van Ginneken, Carel B Hoyng, Thomas Theelen, and Clara I Sánchez. Fast convolutional neural network training using selective data sampling: Application to hemorrhage detection in color fundus images. *IEEE transactions on medical imaging*, 35(5):1273–1284, 2016.

[16] Jacob Adlers and Gustaf Pihl. Prediction of training time for deep neural networks in tensorflow, 2018.

[17] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[18] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210. PMLR, 2016.

[19] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.

[20] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[21] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.*, 2017:35, 2017.

[22] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1209–1222, 2018.

[23] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[24] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[25] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[26] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[27] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[28] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631, 2017.

[29] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

[30] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[31] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A

hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 707–721, 2018.

[32] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[33] George T Duncan, Robert W Pearson, et al. Enhancing access to microdata while protecting confidentiality: Prospects for the future. *Statistical Science*, 6(3):219–232, 1991.

[34] Daniel Ting, Stephen E Fienberg, and Mario Trottini. Random orthogonal matrix masking methodology for microdata release. *International Journal of Information and Computer Security*, 2(1):86–105, 2008.

[35] Samuel S Wu, Shigang Chen, Deborah Burr, and Long Zhang. New technologies for privacy protection in data collection and analysis. In *Joint Statistical Meetings 2014*, pages 2444–2458, 2014.

[36] Samuel S Wu, Shigang Chen, Deborah Burr, and Long Zhang. A new data collection technique for preserving privacy. *The Journal of privacy and confidentiality*, 7(3):99, 2016.

[37] Qinglin Pei, Shigang Chen, Yao Xiao, and Samuel S Wu. Applying triple-matrix masking for privacy preserving data collection and sharing in hiv studies. *Current HIV research*, 14(2):121–129, 2016.

[38] Samuel S Wu, Shigang Chen, Abhishek Bhattacharjee, and Ying He. Collusion resistant multi-matrix masking for privacy-preserving data collection. In *2017 IEEE 3rd international conference on big data security on cloud (bigdatasecurity), IEEE international conference on high performance and smart computing (HPSC), and IEEE international conference on intelligent data and security (ids)*, pages 1–7. IEEE, 2017.

[39] You Zhou, Yian Zhou, Shigang Chen, and Samuel S Wu. Achieving strong privacy in online survey. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 710–719. IEEE, 2017.

[40] Shangran Qiu, Prajakta S Joshi, Matthew I Miller, Chonghua Xue, Xiao Zhou, Cody Karjadi, Gary H Chang, Anant S Joshi, Brigid Dwyer, Shuhan Zhu, et al. Development and validation of an interpretable deep learning framework for alzheimer's disease classification. *Brain*, 143(6):1920–1933, 2020.

[41] Parkinsons Foundation. Parkinson's outcomes project. https://www.parkinson.org/research/Parkinsons-Outcomes-Project, 2021. [Online; Accessed: 2021-05-15].

[42] Zhiqiang Yang, Sheng Zhong, and Rebecca N Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 92–102. SIAM, 2005.

[43] Runhua Xu, James BD Joshi, and Chao Li. Cryptonn: Training neural networks over encrypted data. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1199–1209. IEEE, 2019.

[44] Li Zhang, Mingliang Wang, Mingxia Liu, and Daoqiang Zhang. A survey on deep learning for neuroimaging-based brain disorder analysis. *Journal of Frontiers in Neuroscience*, 14, October 2020.

[45] Microsoft Azure. Microsoft azure machine learning. https://azure.microsoft.com/en-us/services/machine-learning, 2021. [Online; Accessed: 2021-05-15].

[46] Amazon Web Services. Machine learning on amazon web services. https://aws.amazon.com/machine-learning, 2021. [Online; Accessed: 2021-05-15].

[47] Google. Google ai platform. https://cloud.google.com/ai-platform, 2021. [Online; Accessed: 2021-05-15].

[48] Long Zhang. *On security properties of Random Matrix Masking*. PhD thesis, University of Florida, 2014.

[49] Aidong Adam Ding, Guanhong Miao, and Samuel Shangwu Wu. On the privacy and utility properties of triple matrix-masking. *Journal of Privacy and Confidentiality*, 10(2), 2020.

[50] Dimitrios Melissourgos, Hanzhi Gao, Chaoyi Ma, Shigang Chen, and Sam Wu. On outsourcing artificial neural network learning of privacy-sensitive medical data to the cloud. *https://www.cise.ufl.edu/∼sgchen/Publications/tech.pdf*, 2021.

[51] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[52] Slavko Simic. On a global upper bound for jensen's inequality. *Journal of mathematical analysis and applications*, 343(1):414–419, 2008.

[53] Xiang Gao, Meera Sitharam, and Adrian E Roitberg. Bounds on the jensen gap, and implications for mean-concentrated distributions. *arXiv preprint arXiv:1712.05267*, 2017.

## VIII. Appendix

### A. Utility Proof

Below we prove Theorem V.1, presented in Section V.

*Proof:* We assume that the cost function used is the binary cross-entropy [51], which is the most commonly used cost function in a classification problem. Let $a^L$and $\hat{a}^L$ be the output probabilities of the NN after feeding the raw data $X$ and the masked data $AX$, respectively. Then, the formula for the cost of the raw data is given by,

$$\begin{aligned} C &= \frac{-1}{n}\Big(y^T \ln(a^L) + (1-y)^T \ln(1-a^L)\Big) \\ &= \frac{-1}{n}\Big(y^T A^T A \ln(a^L) + (1-y)^T A^T A \ln(1-a^L)\Big) \\ &= \frac{-1}{n}\Big(y^T A \ln(a^L) + (1-y)^T A \ln(1-a^L)\Big), \end{aligned}$$

given that the masking matrix $A$ has the property $Ay = y$, $1^T A = 1^T$, and $A^T A = I$.

Similarly, the cost of the masked data $A[X, y]$ is $\hat{C} = \frac{-1}{n}\Big(y^T \ln(\hat{a}^L) + (1-y)^T \ln(1-\hat{a}^L)\Big)$.

We need to show that $C$ and $\hat{C}$ are close in the sense that the differences between $C$ and $\hat{C}$ are bounded, and thus the gradient descent algorithm can be implemented to the masked data as well. That is,

$$\begin{aligned} |C - \hat{C}| &= |\frac{-1}{n}\Big(y^T A \ln(a^L) + (1-y)^T A \ln(1-a^L) - y^T \ln(\hat{a}^L) - (1-y)^T \ln(1-\hat{a}^L)\Big)| \\ &= \frac{1}{n}|y^T(A\ln(a^L) - \ln(\hat{a}^L)) + (1-y)^T(A\ln(1-a^L) - \ln(1-\hat{a}^L))| \\ &\leq \frac{1}{n}\Big(y^T|A\ln(a^L) - \ln(\hat{a}^L)| + (1-y)^T|A\ln(1-a^L) - \ln(1-\hat{a}^L)|\Big), \end{aligned}$$

where $|\cdot|$ is the element-wise absolute value function.

Let $M_0$ be the initialization of the NN, which can be treated as a function that returns the output probablities given the input data. We can find a composite function $F_1(\cdot) = \ln(M_0(\cdot))$. In other words, $F_1(X) = \ln(a^L)$ and $F_1(AX) = \ln(\hat{a}^L)$. Similarly, we can find a function $F_2$ such that $F_2(X) = \ln(1-a^L)$ and $F_2(AX) = \ln(1-\hat{a}^L)$.

Therefore, the inequality can be re-written as $|C - \hat{C}| \leq \frac{1}{n}\Big(y^T|AF_1(X) - F_1(AX)| + (1-y)^T|AF_2(X) - F_2(AX)|\Big)$.

For simplicity, we assume that $X$ has only one variable, and $F_1$ and $F_2$ are $\alpha$-Höider continuous, i.e., for $\alpha > 0$ and $i = 1, 2, \ldots, n$, there exist postive constants $K_1$ and $K_2$ such that $|F_1(X) - F_1(A_{i\cdot}X)| \leq K_1|X - A_{i\cdot}X|^\alpha$ and $|F_2(X) - F_2(A_{i\cdot}X)| \leq K_2|X - A_{i\cdot}X|^\alpha$. Note that it is straightforward to extend the results to multivariate cases where $X$ is a matrix.

The bounds are then given below,

$$\begin{aligned} |A_{i\cdot}F_1(X) - F_1(A_{i\cdot}X)| &= |\sum_{j=1}^{n} A_{ij}\big(F_1(X)_j - F_1(A_{i\cdot}X)\big)| \\ &\leq \sum_{j=1}^{n} |A_{ij}||F_1(X)_j - F_1(A_{i\cdot}X)| \\ &= |A_{i\cdot}||F_1(X) - F_1(A_{i\cdot}X)| \\ &\leq K_1|A_{i\cdot}||X - A_{i\cdot}X|^\alpha, \end{aligned}$$

where $F_1(X)_j$ is the $j^{th}$ element in the vector $F_1(X)$. Note that $|A_{i\cdot}|$ and $|X - A_{i\cdot}X|^\alpha$ are all available while masking.

Similarly, we have $|A_{i\cdot}F_2(X) - F_2(A_{i\cdot}X)| \leq K_2|A_{i\cdot}||X - A_{i\cdot}X|^\alpha$.

Thus,

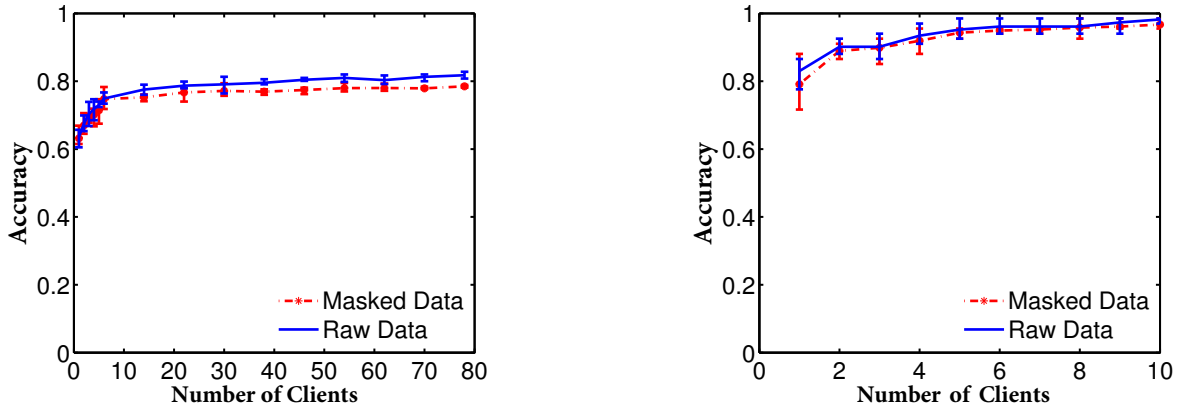
$$
\begin{aligned}
|C-\hat{C}| &\leq \frac{1}{n}\Big(y^T|A\ln(a^L)-\ln(\hat{a}^L)|+(1-y)^T|A\ln(1-a^L)-\ln(1-\hat{a}^L)|\Big)\\
&= \frac{1}{n}\Big(y^T|AF_1(X)-F_1(AX)|+(1-y)^T|AF_2(X)-F_2(AX)|\Big)\\
&\leq \frac{1}{n}\Big(y^TK_1\begin{bmatrix}|A_1.||X-A_1.X|^\alpha\\|A_2.||X-A_2.X|^\alpha\\\vdots\\|A_n.||X-A_n.X|^\alpha\end{bmatrix}+(1-y)^TK_2\begin{bmatrix}|A_1.||X-A_1.X|^\alpha\\|A_2.||X-A_2.X|^\alpha\\\vdots\\|A_n.||X-A_n.X|^\alpha\end{bmatrix}\Big)\\
&= \frac{1}{n}\Big(y^TK_1+(1-y)^TK_2\Big)\begin{bmatrix}|A_1.||X-A_1.X|^\alpha\\|A_2.||X-A_2.X|^\alpha\\\vdots\\|A_n.||X-A_n.X|^\alpha\end{bmatrix}.
\end{aligned}
$$

Therefore, the differences between $C$ and $\hat{C}$ are bounded. Also, it has been shown in [52] and [53] that the bounds can be further tightened depending on the properties of $F_1$ and $F_2$, such as convexity, continuity, and so on. In our work, we demonstrate the tightness through experiments in Section VI.

■

### *B. Mean Squared Error*

In this paper, we used binary crossentropy as our loss function for binary classification. In Figure 13 below, we show that our method works with other loss functions, such as the Mean Squared Error (MSE).




(a) PD data set

(b) ADNI data set.

Fig. 13: Model accuracy w.r.t. number of clients in the PD and ADNI data sets, with MSE loss function.