Dopamine: Differentially Private Secure Federated Learning on Medical Data

Mohammad Malekzadeh [*1] **Burak Hasircioglu** [*1] **Mital Nitish** [*1] **Kunal Katarya** [*1] **Mehmet Emre Ozfatura** [1] **Deniz Gunduz** [1]

Abstract

While a vast amount of valuable medical data are available in institutions distributed across states and countries, the patients' privacy requirements are strong barriers against utilizing the patient's data for improving medical diagnostic models. We propose *Dopamine*: a system that enables privacy-preserving training of deep neural network on distributed medical images. *Dopamine* introduces a customization of differentially-private stochastic gradient descent for federated learning that, in combination with secure multi-party aggregation, achieves accurate model training while providing well-bounded privacy guarantees for patients. Experimental results on the diabetic retinopathy task shows *Dopamine* achieves the similar differential privacy bound of the centralized training counterpart, while it also get the better classification accuracy than the federated learning with parallel differential privacy. *Dopamine* is open-sourced at: `https://github.com/ipc-lab/private-ml-for-health`.

1. Introduction

Accessibility and accuracy in the diagnosis of health and medical problems is of utmost importance, specially for people of developing countries. The recent advances in digital communications, *e.g.* high-speed 5G networks, are enabling the *training* of machine learning models via the availability of valuable medical datasets, that are annotated via experts but distributed among different hospitals across countries and continents. Achieving a well-trained model facilitates the possibility of disease prediction on the medical data of new patients who do not easily have access to adequate experts or facilities. Specifically, medical images are the type of data that are highly useful, for instance in faster diagnosis of skin disease (Esteva et al., 2017), lung cancer (Dunnmon et al., 2019), or diabetic retinopathy (Gulshan et al., 2016). Medical images are mostly of very sensitive nature with high resolutions that may easily be used for malicious re-identification of the data owner (Dwork et al., 2017). To protect the patients' privacy, medical datasets are rarely shared with other parties. Thus, there has been a surge of interest in privacy-preserving learning from distributed medical data (Kaissis et al., 2020).

To this end, *federated learning* (McMahan et al., 2017) is used for training a deep neural network model where each hospital maintains its private dataset locally, while collaborating with a global server in updating a global model on its local datasets. At each iteration, the server sends the current model to each participating hospital, then hospitals run the model on their local datasets, and hospitals send the newly updated model to the server. The hospitals' submitted updated are susceptible to information leakage about the patients' privacy, due to the model over-fitting to or remembering the data on which it was trained (Carlini et al., 2019).

To bound the possibility of inferring whether the data of a patient is used during model training or not, Gaussian random noises are added to the updated models to satisfy *differential privacy* (DP) (Dwork et al., 2014). Depending on the task, and its corresponding privacy and utility requirements, there are several flavors of federated learning with differential privacy, elaborated in Section §4. However, current approaches cannot establish a useful trade-off between the accuracy of the trained model and the promised privacy guarantee to the patient.

We propose *Dopamine*: a customization of differentially-private stochastic gradient descent for federated learning that, in combination with homomorphic encryption based secure aggregation, achieves accurate model training while providing well-bounded privacy guarantees for patients. Experimental results on the diabetic retinopathy dataset (Choi et al., 2017), as well as three other benchmark datasets in image processing, show that *Dopamine* outperforms the existing counterparts in terms of the established accuracy-privacy trade-off. More specifically, *Dopamine* achieves a similar differential privacy bound as the centralized training coun-

* Equal contribution [1]Information Processing and Communications Lab, Department of Electrical and Electronic Engineering, Imperial College London, UK. Corresponding Author: Mohammad Malekzdeh <m.malekzadeh@imperial.ac.uk>.

terpart, while it also achieves better classification accuracy than naive federated learning with differential privacy. To our knowledge, this is the first analysis of privacy-preserving machine learning on the well-known medical dataset: Diabetic Retinopathy (DR) (Choi et al., 2017). Moreover, based on secure multiparty computation (SMPC) (), we provide a web-application to perform private inference on a patient's data after training. This app keeps both the patient's data and the server's model protected from the other party and only reveals the prediction.

Our main contributions are:

- We design and evaluate *Dopamine*: a system for performing federated training of deep neural networks while satisfying patient-level differential privacy. We also provide theoretical analysis of the guaranteed privacy.
- We publish a benchmark environment that facilitates further research on privacy-preserving federated learning on distributed medical image datasets.
- We provide an application for performing private inference that uses SMPC such that it protect both the patient's sensitive data and the server's model.

2. Background

Notation. We use lower-case *italic*, *e.g.* x, for variable scalars; upper-case *italic*, *e.g.* X, for constant scalars; standard **bold** font, *e.g.* $\mathbf{X}$, for vector, matrices, and tensors; blackboard font, *e.g.* $\mathbb{X}$, for sets; and calligraphy font, *e.g.* $\mathcal{X}$, for functions and algorithms.

2.1. Differential Privacy

When we want to publish the results of a computation $\mathcal{F}(\mathbb{D})$, over a private dataset $\mathbb{D}$, differential privacy (DP) helps us to bound the ability of an adversary in distinguishing whether any specific sample data was included in the dataset $\mathbb{D}$ or not. DP has a worst-case threat model, assuming that all-but-one can collude and adversaries can have access to any source of side-channel information.

Definition 1 (**Differential Privacy**). Given $\epsilon, \delta \geq 0$, a mechanism (*i.e.* algorithm) $\mathcal{M}$ satisfies $(\epsilon, \delta)-$differential privacy if for all pairs of neighboring datasets $\mathbb{D}$ and $\mathbb{D}'$ differing in only one sample, and for all $\mathbb{Y} \subset Range(\mathcal{M})$, we have

$$\Pr\big(\mathcal{M}(\mathbb{D}) \in \mathbb{Y}\big) \leq e^{\epsilon} \Pr\big(\mathcal{M}(\mathbb{D}') \in \mathbb{Y}\big) + \delta, \qquad (1)$$

where ϵ is the parameter that specifies the privacy loss (*i.e.* e^{ϵ}), δ is the probability of failure in assuring the upper-bound on the privacy loss, and the probability distribution $\Pr$ is over the internal randomness of the mechanism $\mathcal{M}$, holding the dataset fixed (Dwork et al., 2006; 2014). When $\delta = 0$, it is called *pure* DP, which is a stronger but less flexible in terms of the mechanism design.

For approximating a deterministic function $\mathcal{F}$, such as the mean or sum, a typical example of $\mathcal{M}$ in Equation (1) is a zero-mean Laplacian or Gaussian noise addition mechanism (Dwork et al., 2014), where the variance is chosen according to the *sensitivity* of $\mathcal{F}$ that is the maximum of the absolute distance $|\mathcal{F}(\mathbb{D}) - \mathcal{F}(\mathbb{D}')|$ among all pairs of neighboring datasets $\mathbb{D}$ and $\mathbb{D}'$ (Dwork et al., 2006). In general, to be able to calculate the sensitivity of $\mathcal{F}$, we need to know, or to bound, the $Range(\mathcal{F})$; otherwise we cannot design a reliable DP mechanism.

Definition 2 (**Gaussian Mechanism**). Gaussian mechanism adds a Gaussian noise to a function $\mathcal{F}(x)$ so that the response to be released is (ϵ, δ)-differentially private. It is defined as

$$\mathcal{M}(x) = \mathcal{F}(x) + \mathcal{N}(\mu = 0, \sigma^2 = \frac{2 \ln (1.25/\delta)(\Delta_2 f)^2}{\epsilon^2}),$$

where $\Delta_2(\mathcal{F})$ is the L_2 sensitivity of $\mathcal{F}(x)$ to the neighboring datasets (Dwork et al., 2014).

DP mechanisms are in two types: central DP and local DP. In central DP, we consider a model where there is a trusted server, thus the patients original data is shared with that server. The trusted server then run $\mathcal{M}$ on the collected data before sharing the result with other parties. In local DP, every patient randomly transforms data at the patient's side without needing to trust the server or other parties (Erlingsson et al., 2014; Wang et al., 2017; Erlingsson et al., 2019). Thus, in terms of privacy guarantee, local DP ensures that the probability of discovering the true value of the user's shared data is limited to a mathematically defined upper-bound. However, central DP gives such an upper-bound for the probability of discovering whether a specific user has shared their data or not.

DP mechanisms are mostly used for computing aggregated statistics, such as the sum, mode, mean, most frequent, or histogram of the dataset (Bittau et al., 2017; Ding et al., 2017; Cormode et al., 2018). DP mechanisms can also be used for training machine learning models on sensitive datasets while providing differential privacy guarantees for the people that are included in the training dataset (Abadi et al., 2016; Bonawitz et al., 2019; Kifer et al., 2020).

2.2. Federated Learning

Many parameterized machine learning problems can be modeled as a *stochastic optimization problem*

$$\min_{\mathbf{w} \in \mathbb{R}^d} := \mathbb{E}_{\mathbb{D} \sim \mathcal{D}} \mathcal{L}(\mathbf{w}, \mathbb{D}), \qquad (2)$$

where $\mathbf{w}$ denotes the model parameters, $\mathbb{D}$ is a random dataset sampled from the true, but unknown, data distribution $\mathcal{D}$, and $\mathcal{L}$ is a task-specific empirical loss function.

The core idea of the federated learning strategy is to solve the stochastic optimization problem, Equation (2), in a distributed manner such that participating users do not share their local data set with each other but the model parameters to seek a consensus. Accordingly, the objective of K users that collaborate to solve Equation (2) is reformulated as the sum of K loss functions that defined over different sample datasets, where each corresponds to a different user:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\mathbb{D}_k \sim \mathcal{D}_k} \mathcal{L}(\mathbf{w}, \mathbb{D}_k) \tag{3}$$

where $\mathcal{D}_k$ denotes the portion of the dataset allocated to the user k. Stochastic Gradient Decent (SGD) is a common approach to solve machine learning problems that are represented in the form of Equation (2). Hence, FL considers the problem that written according to the distributed scenario in Equation (3) and aim to solve it using SGD method in a decentralized way.

2.3. Homomorphic Encryption

Fully Homomorphic Encryption (FHE) allows computation of arbitrary functions $\mathcal{F}$ on encrypted data, thus enabling a plethora of applications like private computation offloading. Gentry (**?**) was the first to show that FHE is possible with a method that includes the following steps: (i) to construct a somewhat homomorphic encryption (SHE) scheme that can evaluate functions of low degree, (ii) to simplify the decryption circuit, (iii) to evaluate the decryption circuit on the encrypted ciphertexts homomorphically to obtain new ciphertexts with a reduced inherent noise. The third step is called bootstrapping, and this allows computations of functions of arbitrary degree. Here, the degree of a function is the number of operations that must be performed in sequence to compute the function. For example, to compute x^2, one multiplication is necessary, therefore its degree is 1, while to compute x^3, 2 multiplications ($x * x$ and $x * x^2$) are necessary, and therefore its degree is 2.

The security of homomorphic encryption schemes is based on the Learning With Errors (LWE) problem (**?**), or its ring variant called Ring Learning With Errors (RLWE) (**?**), the hardness of which can be shown to be equivalent to that of classical hard problems on ideal lattices (**?**). All FHE schemes add a *small noise* component during encryption of a message with a *public key*, which makes the decryption very hard unless one has access to a *secret key*. The decrypted message obtained consists of the message corrupted by a small additive noise, which can be removed if the noise is "small enough"(**?**). Each computation on ciphertexts increases the noise, which ultimately grows large enough to fail the decryption. The bootstrapping approach is used to lower the noise in the ciphertext to a fixed level. Addition of ciphertexts increases the noise level very slowly, while multiplication of ciphertexts increases the noise very fast. When FHE is only used for aggregation, the bootstrapping step, which is a computationally expensive procedure, is not usually necessary since the noise never grows to a large enough size.

While machine learning models compute on floating point parameters, encryption schemes work only on fixed point parameters, or equivalently, integer parameters, which can be obtained by appropriately scaling and rounding fixed point values. Given an integer u to encode and an integer base b, a base-b expansion of u is computed, and represented as a polynomial with the same coefficients. The expansion uses a 'balanced' representation of integers modulo b as the coefficients, that is, when b is odd, the coefficients are chosen to lie in the range $\left[\frac{-(b-1)}{2}, \frac{(b-1)}{2}\right]$, and when b is even, the coefficients are chosen to lie in the range $\left[\frac{-b}{2}, \frac{(b-1)}{2}\right]$, except if b is two, when the coefficients are chosen to lie in the set $\{0, 1\}$. For example, if $b = 2$, the integer $26 = 2^4 + 2^3 + 2^1$ is encoded as the polynomial $x^4 + x^3 + x$. If $b = 3$, $26 = 3^3 - 3^0$ is encoded as the polynomial $x^3 - 1$. Decoding is done by computing the polynomial representation at $x = b$. To map an integer to a polynomial ring $R = \mathbb{Z}[x]/f(x)$, where $f(x)$ is a monic polynomial of degree d, the polynomial representation of the integer is viewed as being a plaintext polynomial in R. The base b is called the *plaintext modulus*. The most popular choice of $f(x)$ is $x^d + 1$, with $d = 2^n$ called the *polynomial modulus*. Let $q > 1$ be a positive integer, then we denote the set of integers $[-\frac{q}{2}, \frac{q}{2}]$ by $\mathbb{Z}_q$, and the set of polynomials in R with coefficients in $\mathbb{Z}_q$ by R_q. The integer q is called the *coefficient modulus*, and is usually set much larger than the plaintext modulus.

For multi-dimensional vectors, the Single-Instruction-Multiple-Data (SIMD) technique, also known as *batching*, encodes a given array of multiple integers into a single plaintext polynomial in R_q. A single computation instruction on such a plaintext polynomial is equivalent to simultaneously executing that instruction on all the integers in that array, thus speeding up the computation by many orders of magnitude. The plaintext modulus is assumed to be larger than any integer in the given array. The length of the array is assumed to be equal to the degree of the polynomial modulus d, and in case that its size is less than d, then it is padded with zeros to make its length equal to d. Lagrange interpolation is performed over the array of integers to obtain a polynomial of degree $d - 1$, thus obtaining the plaintext polynomial in R_q encoding d integers.

3. *Dopamine*'s Methodology

Problem Formulation. Let every patient i have a labeled data $(\mathbf{x}_j, y_j)$. Let every hospital $k \in \{1, 2, \ldots, K\}$ own a dataset, $\mathbb{D}_k$, with an unknown number of patients. Let a deep neural network model, $\mathcal{M}$, take input $\mathbf{x}$ and produce the prediction $\hat{y} = \mathcal{M}(\mathbf{X})$. Let the global server own a small validation dataset $\mathbb{D}_G$. The goal is to run an algorithm in a server to train $\mathcal{M}$ by monitoring the following requirements:

- **Utility**: the prediction accuracy of $\mathcal{M}$ on $\mathbb{D}_G$.
- **Threat Model:** patients only trust hospitals; hospitals are honest; hospitals assume the server is honest but curios; hospitals and server do not trust any other third parties.
- **Privacy**: the weakest acceptable bound on the differential privacy loss is defined by patient-level (ϵ, δ)-DP. The hospitals aim to ensure their patients a computational DP against the server during the training, and an information-theoretical DP against the server and any other third parties after training.

It should be noted that when a DP mechanism uses cryptography at some point of the process, then it cannot guarantee a pure information-theoretical DP as every cryptography system is only robust to computationally-bounded adversaries. Therefore, this kind of DP guarantees are called computational DP. In the case of *Dopamine*, we assume that adversaries are computationally bounded during the training, which is a typical assumption for almost every current secure system.

3.1. *Dopamine*'s Training

We train *Dopamine* by using distributed stochastic gradient descent, that is all K hospitals send their model updates after every local iteration. First, every hospital samples data points from its local dataset independently with probability q. Due to independent sampling, the batch size is not fixed, but is a binomial random variable with parameters q and N^i. We denote the set of samples at iteration t by the hospital k as $\mathbb{D}_k^t$. Let $|\mathbb{D}_k^t|$ and γ denote the batch size in the local updates and the learning rate respectively. Moreover for DP, the maximum value L_2 norm of per-sample gradients is C. If a per-sample gradient has a norm greater than this, then its norm is clipped according to the procedure defined in (Abadi et al., 2016).

Our training procedure is given in Algorithm 1.

We assume that the workers do not trust the parameter server in our implementation. In that case, the immediate solution that presents itself is for each worker to add a larger amount of noise to the model parameter updates to keep them differentially private from the parameter server. However, this approach has the undesirable effect of decreasing the accuracy of the trained model due to the addition of a larger noise. A better solution is to employ secure aggregation of the model parameters in the parameter server, which prevents the parameter server from learning the individual parameter updates sent from the workers. Since the individual model parameters are now hidden from the parameter server, the workers do not need to add the extra noise to keep their model parameter updates differentially private from the server, thus increasing the accuracy of the trained model significantly.

Algorithm 1 *Dopamine*'s Training

Input: K: number of hospitals, $\mathbb{D}$: distributed dataset, $\mathbf{w}$: model's trainable parameters, $\mathcal{L}(\cdot, \cdot)$: loss function, q: sampling probability, σ: noise scale, C: gradient norm bound, η: learning rate, β: momentum, T: number of iterations, (ϵ, δ): bounds on the patient-level differential privacy loss.
Output: $\mathbf{w}$: final parameters.
$\mathbf{w}_G^0 =$ random initialization.
$\hat{\epsilon} = 0$
for $t : 1, \ldots, T$ **do**
  for $k : 1, \ldots, K$ **do**
    $\mathbf{w}_k^t = \mathbf{w}_G^{t-1}$
    $\mathbb{D}_k^t = Sampling(\mathbb{D}_k)$ //by uniformly sampling each item in $\mathbb{D}_k$ independently with probability q.
    for $\mathbf{x}_i \in \mathbb{D}_k^t$ **do**
      $\mathbf{g}^t(\mathbf{x}_i) = \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_k^t, \mathbf{x}_i)$
      $\bar{\mathbf{g}}^t(\mathbf{x}_i) = \mathbf{g}^t(\mathbf{x}_i) / \max\left(1, \frac{||\mathbf{g}^t(\mathbf{x}_i)||_2}{C}\right)$
    end for
    $\widetilde{\mathbf{g}}_k^t = \frac{1}{|\mathbb{D}_k^t|}\left(\sum_i \bar{\mathbf{g}}^t(\mathbf{x}_i) + \mathcal{N}(0, \frac{\sigma^2 \cdot C^2 \cdot \mathbf{I}}{\sqrt{K}})\right)$
    $\hat{\mathbf{g}}_k^t = \widetilde{\mathbf{g}}_k^t + \beta \hat{\mathbf{g}}_k^{t-1}$ //$\hat{\mathbf{g}}_k^0 = 0$
    $\mathbf{w}_k^t = \mathbf{w}_k^t - \eta \hat{\mathbf{g}}_k^t$
  end for
  $\hat{\epsilon} = CalculateRenyiDP(\delta, q, \sigma, t)$
  if $\hat{\epsilon} > \epsilon$ **then**
    return w_G^{t-1}
  end if
  $\mathbf{w}_G^t = \frac{1}{K}\left(SecureAggregation(\sum_k \mathbf{w}_k^t)\right)$
end for

Lemma 1. If every agent adds Gaussian noise distributed as $\mathcal{N}(\mu = 0, \sigma^2 = \frac{2\ln(1.25/\delta)(\Delta_2 \mathcal{F})^2}{\epsilon^2})$, where $\Delta_2 \mathcal{F} = \frac{C}{|\mathbb{D}_k^t|\sqrt{K}}$, to the model updates before sending them to the parameter server, then the averaged model update received by the parameter server is (ϵ, δ)-DP.

Proof. Let us try to get the effect of one data point in one of the hospitals, let us say hospital k. Let t denote the iteration index. Let $w_k^t - w_G^{t-1}$ be the noiseless model update of the hospital k. Due to secure aggregation, the parameter server

directly sees

$$w_G^t - w_G^{t-1} = \frac{1}{K}\sum_{k=1}^{K}(w_k^t - w_G^{t-1})$$

where w_k^t is the global model parameters at the global round t. Since the each per-sample gradient is bounded by C and the hospital samples $|\mathbb{D}_k^t|$ data points, $\left\|w_k^t - w_G^{t-1}\right\|_2 \leq \frac{C}{|\mathbb{D}_k^t|}$, $\forall k \in [K]$. Note that changing one sample in any of the hospitals can only change the gradients at that agent. Combining these with the triangle inequality, we have

$$\Delta_2(w_G^t - w_G^{t-1}) = \frac{C}{|\mathbb{D}_k^t|K}.$$

Thus the effective noise added to $\theta_p - \theta_{p-1}$ must have a variance $\frac{2\ln(1.25/\delta)C^2}{\epsilon^2 K^2 |\mathbb{D}_k^t|^2}$. Now assume that every agent k adds a noise $n_k \sim \mathcal{N}(\mu = 0, \sigma^2)$ to its model update. Then the effective noise added to $w_G^t - w_G^{t-1}$ is $\frac{1}{K}\sum_{k=1}^{K} n_k$. Its variance is $\frac{\sigma^2}{K}$. Since

$$\sigma^2 = \frac{2\ln(1.25/\delta)C^2}{|\mathbb{D}_k^t|^2 K}$$

the variance of the effective noise is $\frac{2\ln(1.25/\delta)C^2}{|\mathbb{D}_k^t|^2 K^2}$ □

3.2. Secure Aggregation

Secure aggregation helps in providing a computational DP guarantee, even during the training procedure, so that the hospitals do not need to trust the server. We use the Brakerski-Fan-Vercauteren (BFV) scheme (Fan & Vercauteren, 2012) for homomorphic encryption. The security of the HE scheme depends on the assumption of hardness of the RLWE problem (**?**). The RLWE problem can be reduced to the shortest vector problem over ideal lattices (**?**), which is assumed to be robust to even quantum attacks. For security parameter λ, random element $s \in R_q$, $\Delta = \lfloor \frac{q}{b} \rfloor$, and a discrete probability distribution $\chi = \chi(\lambda)$, the BFV scheme is described as follows:

- **SecretKeyGen** (1^λ)**:** sample $\mathbf{s} \leftarrow \chi$, and output $\mathbf{sk} = \mathbf{s}$.
- **PublicKeyGen (sk):** sample $\mathbf{a} \leftarrow R_q$, $e \leftarrow \chi$, and output

$$\mathbf{pk} = \left([-(\mathbf{as} + \mathbf{e})]_q, \mathbf{a}\right). \tag{4}$$

- **Encrypt**$(\mathbf{pk}, \mathbf{m})$**:** To encrypt a message $\mathbf{m} \in R_b$, let $\mathbf{p}_0 = \mathbf{pk}[0]$ and $\mathbf{p}_1 = \mathbf{pk}[1]$, and sample $\mathbf{u}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi$, and output

$$\mathbf{ct} = \left(\left[\mathbf{p}_0\mathbf{u} + \mathbf{e}_1 + \Delta.\mathbf{m}\right]_q, \left[\mathbf{p}_1\mathbf{u} + \mathbf{e}_2\right]_q\right). \tag{5}$$

- **Decrypt**$(\mathbf{sk}, \mathbf{ct})$**:** Set $\mathbf{s} = \mathbf{sk}$, $\mathbf{c}_0 = \mathbf{ct}[0]$, and $\mathbf{c}_1 = \mathbf{ct}[1]$, and compute

$$\left[\left[\frac{b.[\mathbf{c}_0 + \mathbf{c}_1\mathbf{s}]_q}{q}\right]\right]_b. \tag{6}$$

We use the open source PySEAL library, which provides a python API for the original SEAL library published by Microsoft (**?**). We simulate the federated learning environment over the message passing interface (MPI), with the rank 0 process modeling the server, and the rank $1, \ldots, K$ processes model hospitals that participate in the training. The plaintext modulus b is chosen to be 40961, while the polynomial modulus d is chosen to be $x^{4096} + 1$. Therefore, each ciphertext can pack $d = 4096$ integers with magnitude less than 40961 for SIMD operations, thus speeding up the communication and computation on encrypted data by 4096 times. The methodology is as follows:

1. Key generation and distribution: A public key and secret key are generated by the rank 1 process (hospital 1). The public key is broadcasted to all involved parties, while the secret key is shared with the hospitals but not the server.
2. The server initializes the model to be trained, and sends the model to the hospitals in the first round.
3. Each hospital performs an iteration on its local model based on Algorithm 1, at the end of which each hospital has the updated model.
4. Each active hospital:
   - flattens the tensor of model parameters into a 1D array $\mathbf{w}$,
   - converts the floating point values in the array to 3 digit integers by multiplying each value by 10^3, and then rounding it to the nearest integer.
   - partitions the array $\mathbf{w}$ into chunks of $d = 4096$ parameters, denoted by $\mathbf{w}_1, \ldots, \mathbf{w}_l$, where $l = \lceil \frac{\text{length}(\mathbf{w})}{4096} \rceil$,
   - encrypts the chunks $\mathbf{w}_1, \ldots, \mathbf{w}_l$, with the public key, into ciphertexts $\mathbf{ct}_1, \ldots, \mathbf{ct}_l$, respectively, using the batching technique, and sends the list of ciphertexts to the server.
5. The server adds the received encrypted updates and sends the encrypted aggregated model to the hospitals for the next round.
6. The hospitals decrypt the received aggregate array $\mathbf{w}$, scale and convert each integer value to a floating point value of correct precision, divide each value by the number of hospitals that were active in the previous round, and finally reshape $\mathbf{w}$ back to the structure of the original model.

7. Go to step 3.

At the end of the above procedure the hospitals send the final unencrypted updated models to the server, who aggregates them and updates the global model for the final time to obtain the final trained model.

4. Evaluation

4.1. Dataset

Diabetic retinopathy is an eye condition that can cause vision loss in diabetic patients. It is diagnosed by examining scans of the blood vessels of a patient's retina - thereby making this an image classification task. A diabetic retinopathy dataset (Choi et al., 2017), available at `https://www.kaggle.com/c/aptos2019-blindness-detection/data`, exists to solve this task. The problem is to classify the images of patients' retina into five categories (DR is used here as an abbreviation for Diabetic Retinopathy): *No DR*, *Mild DR*, *Moderate DR*, *Severe DR*, and *Proliferative DR*. The dataset consists of 2931 images, of variable sizes, for training, and 731 for testing. To deal with the variation in image dimensions, all images were resized into the same dimensions of 224×224 during pre-processing.

4.2. Neural Network Architecture

We use SqueezeNet (Iandola et al., 2016): a convolutional neural network architecture that has 50x fewer parameters than the famous AlexNet, but it is shown that it can maintain the AlexNet-level accuracy on the famous ImageNet dataset. For both private training and inference on a neural network, the size of the model is a very important factor which is one of the main motivations for using this architecture. Moreover, SqueezeNet can achieve a classification accuracy of about 80% on the diabetic-retinopathy dataset while the cuurrent best accuracy reported on the Kaggle's competition[1] on this dataset is about 83% using a much larger neural networks EfficientNet-B3 that has 18x more parameters than SqueezeNet. Here we use the same Pytorch implementation that is available at `https://pytorch.org/docs/stable/_modules/torchvision/models/squeezenet.html`.

4.3. Evaluation Settings for Training Stage

Overall, there are several approaches for training a deep neural network that provide us different trade-offs between utility, privacy, and complexity. We compare *Dopamine* against the following well-known approaches:

[1] `https://www.kaggle.com/c/aptos2019-blindness-detection/notebooks`

1. **Non-Private Centralized (C).** Every hospital shares the original data with the server such that a centralized dataset $\mathbb{D}$ is available for training the model at the server's side. In this setting the problem is reduced to architecture search and optimization procedure for training a neural net on the $\mathbb{D}$. This setting provides the best achievable utility, a moderate complexity, but a bad privacy. It does not provide any sort of privacy guarantee for the patients and it makes a single point of failure in the case of data breach. Moreover, in situations where patients are from different countries or private organizations, it is almost impossible reaching agreement of such single trusted curator.

2. **Centralized with Central DP (CDP).** When a trusted curator is available, one can trade some utility in the setting C to guarantee central DP for the patients, by using a differentially private training mechanism such as DPSGD (Abadi et al., 2016). The complexity of this seating CDP is almost similar as the setting C, except for the fact that existing DP mechanisms make the training of neural networks slower and less accurate, as these mechanisms require performing additional tasks such as per-sample gradient clipping and noise addition.

3. **Non-Private Federated (F).** Considering that a trusted curator is not realistic in many scenarios, due to legal, security, and business constraints, hospitals can collaborate via a federated training scenario instead of directly sharing their patients' sensitive data. In this setting F, every hospital has its own dataset that may be different with other hospitals datasets in terms of the samples' quantity and quality. The server runs a federated learning algorithm, *e.g.* FedAvg (McMahan et al., 2017). While this setting F introduces some serious communication complexities, it removes the need for trusted curator and enables an important layer of privacy protection. Basically, after sharing the data in the settings C and CDP, hospitals have no more control on the type and amount of information that can be extracted from the patients' data. However, in this setting F, hospitals do not share the original data, but some aggregated statistics or the results of a computation over the dataset. Even more, hospitals can decide stop releasing information at any time, thus having more control over the type and amount of information sharing. Although it seems that the utility of this setting F might be lower than the setting C, there are many studies showing that in some setting and with a more sophisticated algorithms, one can achieve the same utility as the setting C.

4. **Federated with Parallel DP (FPDP).** Similar to the setting CDP, one can apply a DP mechanisms to the

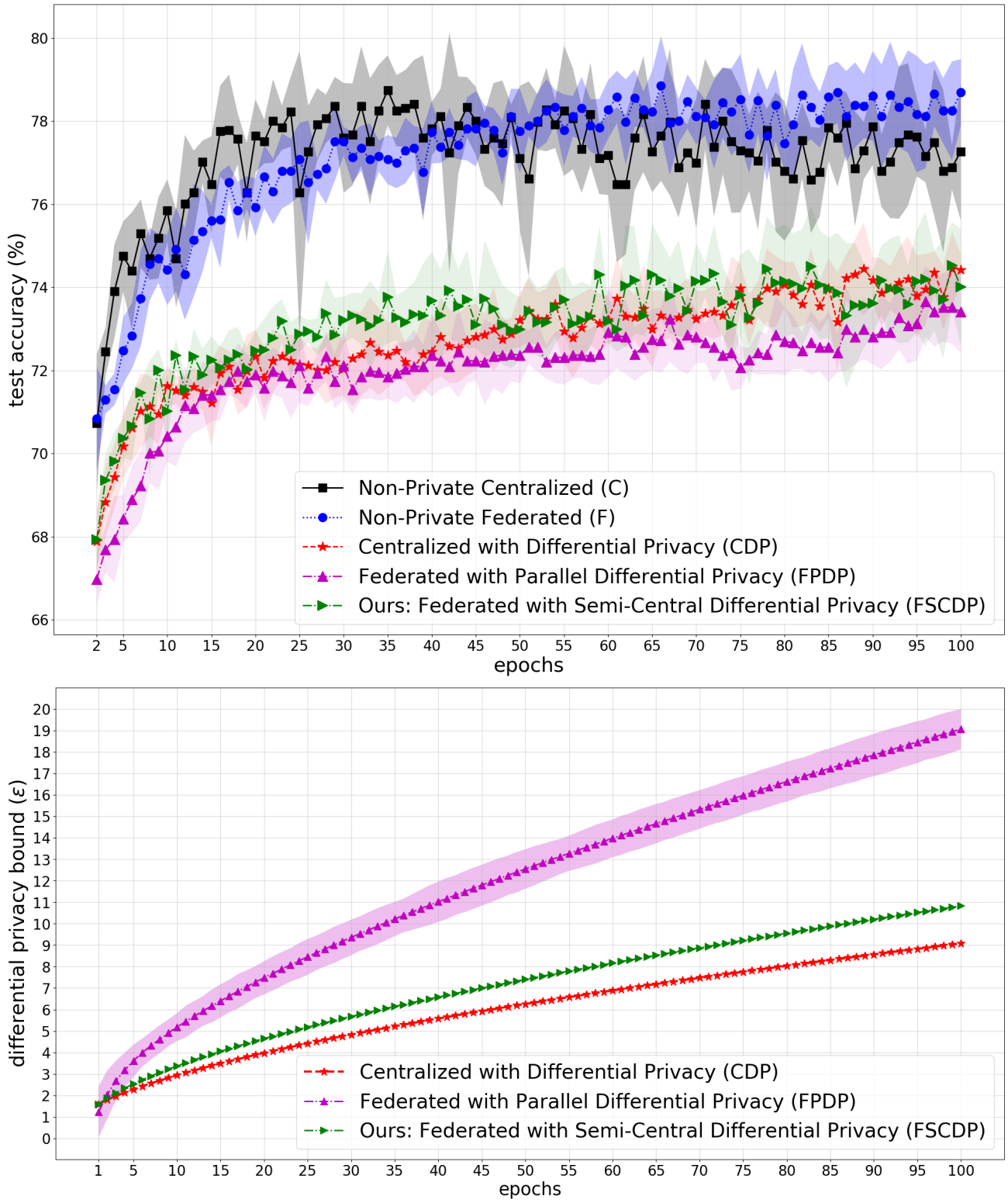

Figure 1. Comparison of classification accuracy vs. differential privacy bound on Diabetic Retinopathy dataset (Choi et al., 2017). Our FSCDP (*Dopamine*) achieves a very close differential privacy bound, of the centralized training counterpart, while it also get better classification accuracy than federated learning with parallel differential privacy.

local training procedure at hospitals' side. Thus, every hospital locally provides a central DP guarantee. Notice that this is different form a local DP guarantee as again an hospital is a trusted curator. Here, a much better privacy is provided than the previous settings C, CDP, and F. However, as the size of each local dataset $\mathbb{D}^i$ is much smaller than the size of the whole dataset $\mathbb{D}$ in setting CDP, more noise should be added to achieve the same privacy ϵ-DP that cause less utility.

5. **Federated with Semi-Central DP (FSCDP).** The setting F, but the hospitals add noise to their gradients and then run a secure aggregation procedure to aggregate the model at the server. This is similar to FPDP in the sense that the hospitals add noise to their gradients but thanks to the secure aggregation, the variance of the noise hospitals add less than FPDP by an order of K. This is the training procedure described in Algorithm 1. For details about the variance of the noise, please refer to Lemma 1.

4.4. Experimental Results

Figure 1 compares the classification accuracy and differential privacy bound of the trained SqueezeNet on the diabetic retinopathy dataset (Choi et al., 2017). We see that our proposed FSCDP achieves a very close differential privacy bound, of the centralized training counterpart, while it also get better classification accuracy than federated learning with parallel differential privacy. The details of the hyper-parameters used and the source code to reproduce the results are provided at `https://github.com/ipc-lab/private-ml-for-health/tree/main/private_training/`

5. Private Inference

Research in the area of private inference of neural networks has been rich recently (), where the main focus has been to reduce the gap between the private inference and plain inference. Our main challenge was to find an easy-to-use API to perform private inference using SMPC for large ImageNet type models. While frameworks like XONN (**?**) and CrypTFlow (**?**) have demonstrated private inference on large ImageNet models with tens of layers and 1000 classes with reasonable efficiency, they are not easily integrable with the training process done using PyTorch because they require understanding a completely different software. Other libraries for private inference that we explored were Crypten (), TF-Encrypted () and PySyft (), which are easier to integrate with the popular ML frameworks like TensorFlow and PyTorch, since they provide an identical API as these popular frameworks. However, these are still under development, and we faced a lot of trouble running private inference across parties (the model owner and the patient) on different machines, for large models. Even though it is possible to run simulations of private inference on simple models using PySyft and CrypTen using virtual parties, we faced difficulty in running the private inference over the web. However, these frameworks are under rapid development, with the PySyft community developing an extension library PyGrid for distributed private training and inference using SMPC, and hopefully soon it will be possible for seamless private inference on arbitrary models.

On the other hand, we also need to protect the privacy of new patients when performing inference on their data using the trained model. In such cases, it is highly desirable to allow the input data to remain secure at the patient's side. Conversely, we would like to incentivize institutions to invest in the expensive process of training and maintains of better machine learning diagnostic tools, as well as protecting the institution's intellectual property (IP). Therefore it is also important to keep the characteristics and parameters of the trained model private during inference. In order to circumvent the apparent contradiction of doing inference on private data with a private model, tools like *secure multiparty computation* (**?**) and *homomorphic encryption* (**?**) can be used. Using such tools, *private inference* can be performed on patients' medical images so that neither the patient's image is disclosed to the server, nor the trained model to a potentially malicious patient. In this paper we discuss the related methods and requirement for building such infrastructure to perform private inference. We also propose XXX....

Consider that the global agent/service provider finally has the trained model, and a user patient in possession of medical data wants to independently receive a diagnosis from the model without revealing his data to the service provider. An option is for the service provider to transfer the model to the user, who then can do inference locally. Differential private training prevents the user from doing *membership attacks* using the model. However, it is not in the interest of the service provider to reveal its model (trained by investing considerable amount of resources) to all users of its service. Hence, we propose the following architecture which keeps the user data as well as the model private, from the provider and the user respectively, by employing secure multiparty computation (SMPC). We consider a local hospital/general practitioner (GP) as the crypto-provider who is responsible for generating the intermediate random values used for encryption in the SMPC protocols between the user and the service provider, while itself learning nothing about the data, model or the result of the inference. We assume that the crypto-provider does not collude with either of the parties. For the implementation of SMPC algorithms, we use the PySyft library by Openmined (Ryffel et al., 2018). This library performs additive secret sharing. The procedure is as follows:

- Consider there are 3 workers - patient worker, service provider worker, and the hospital worker (crypto-provider).
- The service provider generates two secret shares of the trained model, and shares one secret share with the patient worker, and keeps one secret share on the worker associated with itself.
- Similarly, the app on the patient's device generates two secret shares of his medical image, and sends one secret share to the service provider worker, and keeps one secret share on the worker associated with itself.
- SMPC protocols are implemented to compute a forward pass on the encrypted NN model with the encrypted image, giving the result in an encrypted form.
- The encrypted shares of the result are delivered back to the app on the patient's device, thus reconstructing the result.

5.1. Demo Application

The team created a demo application to simulate the eventual use of the proposed system, from a patient's perspective. Deployed in the real world, such an app would perform private and secure inference on a patient's retina scan, bringing operational benefits to the hospitals that adopt the use of the app. The distribution of labels in the dataset is skewed towards the lower-risk categories, with close to 50% of scans being classified as not having any form of Diabetic Retinopathy. Automating the diagnosis of these results would allow a hospital to devote its time, resources and money to patients that are at a risk of vision loss. The demo app has an intuitive user interface, shown in images _ and _ below. Assuming a patient has been sent a digital copy of their scan by the hospital, the app prompts the patient to upload the scan and briefly explains that their data will remain private. It then provides the patient with a diagnosis, within seconds,

¡insert 2 screenshots here¿

On the backend, this application was implemented using Flask and PyTorch. Model exchange with a global agent was simulated by downloading pre-trained model weights hosted on a google drive folder. The weights are downloaded and used within the scope of an inference function, and are hence automatically deleted when the function returns a prediction. This simulates a core functionality of the application - the patient's data never leaves their device in it's original, unencrypted form, and model data is never saved on the device either. Another key functionality of the application is Secure Multi-Party Communciation for private inference. Unfortunately, we were unable to simulate private inference due to shortcomings in existing libraries for SMPC. ¡talk about pysyft and why it sucks here, computation is too slow for complex models¿. The implementation of an SMPC protocol for deep CNNs remains a potential future development of this project.

6. Related Work

(Sheller et al., 2018) apply FL, without any privacy guarantee, for the segmentation of brain tumor images using a deep convolutional neural network, known as U-Net. They compare FL with a baseline collaborative learning scenario where hospitals iteratively train a shared model in turn, and show that FL outperforms such baseline. (Pfohl et al., 2019) employ FL with FedAvg and DPSGD algorithms to train logistic regression and feedforward networks with one hidden layer for in-hospital mortality and prolonged length of stay prediction using the patients electronic health records. Comparing FL with a centralized approach, when DP is used in both cases, their experimental results show that while centralized training can achieve a good DP bound with $\epsilon \approx 1$, but FL with DP performs poorly in terms of both accuracy and ϵ.

Considering the differences in type of the device, number of sensors, and the placement of the sensors that is used in each hospital for collecting Electroencephalography (EEG) data, (Gao et al., 2019) propose an application of a FL for training a classifier over heterogeneous data; however they do not consider any formal privacy guarantee. (Li et al., 2020) use FL among hospitals for training an fMRI image classification and add some noise to the trained parameters before sharing them with the server which does not follow the requirement of a valid differentially private mechanism, thus it does not provide a formal privacy guarantee. (Choudhury et al., 2020) offer a k-anonymity based privacy guarantee to generate a syntactic tabular dataset, instead of the original dataset, and use it for participating in a FL. (Li et al., 2019) proposes an FL system for brain tumour segmentation in a setting where each hospital owns a dataset of MRI scans. However, their implementation does not satisfy patient-level DP, but parameter-level DP, which for neural networks is not a meaningful privacy protraction. (Kumar et al., 2020) propose the CrypTFlow (**?**) and claim that this framework performs private inference on a retina image in about 30 seconds.

7. Conclusion and Future Work

We propose an algorithm, called *Dopamine*, for training a deep neural network on private datasets of medical images that are distributed across several hospitals. Experiments on a benchmark dataset of medical images show that *Dopamine* can outperform existing baselines by providing a better accuracy-privacy trade-off.

There are some important directions to follow in the future work. (i) Current privacy-analysis only allow us to have one local iteration. We observed that allowing local hospitals to perform more than one iteration can improve accuracy by about 3%, but in this case we need to have a separate privacy analysis which we aim to do in future. (ii) Existing differential privacy libraries do not allow using any kind of neural network architectures. They are either not compatible with more recent architectures, like EfficientNet, or they need too much resources. Making DP training faster is still an open area of research and engineering. (iii) Current libraries do not allow us to implement secure aggregation on large neural networks and more than two parties, to build an end-to-end implementation of *Dopamine* we aim to look for more scalable FHE libraries. (v) Finally, we aim to evaluate *Dopamine* on other medical datasets, and investigate into accuracy improvement while keeping the similar privacy protection.

References

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

Bittau, A., Erlingsson, U., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 441–459. ACM, 2017.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., Van Overveldt, T., Petrou, D., Ramage, D., and Roselander, J. Towards federated learning at scale: System design. In *Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA*, 2019.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 267–284, 2019.

Choi, J. Y., Yoo, T. K., Seo, J. G., Kwak, J., Um, T. T., and Rim, T. H. Multi-categorical deep learning neural network to classify retinal images: A pilot study employing small database. *PLOS ONE*, 12:1–16, 11 2017.

Choudhury, O., Gkoulalas-Divanis, A., Salonidis, T., Sylla, I., Park, Y., Hsu, G., and Das, A. Anonymizing data for privacy-preserving federated learning. *arXiv preprint arXiv:2002.09096*, 2020.

Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., and Wang, T. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 1655–1658, 2018.

Ding, B., Kulkarni, J., and Yekhanin, S. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pp. 3571–3580, 2017.

Dunnmon, J. A., Yi, D., Langlotz, C. P., Ré, C., Rubin, D. L., and Lungren, M. P. Assessment of convolutional neural networks for automated classification of chest radiographs. *Radiology*, 290(2):537–544, 2019.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

Dwork, C., Smith, A., Steinke, T., and Ullman, J. Exposed! a survey of attacks on private data. 2017.

Erlingsson, Ú., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, 2014.

Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., and Thakurta, A. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2468–2479. SIAM, 2019.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

Fan, J. and Vercauteren, F. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `https://eprint.iacr.org/2012/144`.

Gao, D., Ju, C., Wei, X., Liu, Y., Chen, T., and Yang, Q. Hhhfl: Hierarchical heterogeneous horizontal federated learning for electroencephalography. *arXiv preprint arXiv:1909.05784*, 2019.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., Kim, R., Raman, R., Nelson, P. C., Mega, J. L., and Webster, D. R. Development and Validation of a Deep Learning Algorithm for Detection

of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, 316(22):2402–2410, 12 2016. ISSN 0098-7484. doi: 10.1001/jama.2016.17216. URL https://doi.org/10.1001/jama.2016.17216.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Kaissis, G. A., Makowski, M. R., Rückert, D., and Braren, R. F. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, pp. 1–7, 2020.

Kifer, D., Messing, S., Roth, A., Thakurta, A., and Zhang, D. Guidelines for implementing and auditing differentially private systems. *arXiv preprint arXiv:2002.04049*, 2020.

Kumar, N., Rathee, M., Chandran, N., Gupta, D., Rastogi, A., and Sharma, R. Cryptflow: Secure tensorflow inference. In *IEEE Symposium on Security and Privacy*. IEEE, May 2020. URL https://www.microsoft.com/en-us/research/publication/cryptflow-secure-tensorflow-inference/.

Li, W., Milletarì, F., Xu, D., Rieke, N., Hancox, J., Zhu, W., Baust, M., Cheng, Y., Ourselin, S., Cardoso, M. J., et al. Privacy-preserving federated brain tumour segmentation. In *International Workshop on Machine Learning in Medical Imaging*, pp. 133–141. Springer, 2019.

Li, X., Gu, Y., Dvornek, N., Staib, L., Ventola, P., and Duncan, J. S. Multi-site fmri analysis using privacy-preserving federated learning and domain adaptation: Abide results. *arXiv preprint arXiv:2001.05647*, 2020.

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

Pfohl, S. R., Dai, A. M., and Heller, K. Federated and differentially private learning for electronic health records. *arXiv preprint arXiv:1911.05861*, 2019.

Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J., Rueckert, D., and Passerat-Palmbach, J. A generic framework for privacy preserving deep learning. *CoRR*, abs/1811.04017, 2018. URL http://arxiv.org/abs/1811.04017.

Sheller, M. J., Reina, G. A., Edwards, B., Martin, J., and Bakas, S. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*, pp. 92–104. Springer, 2018.

Wang, T., Blocki, J., Li, N., and Jha, S. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pp. 729–745, 2017.