Research Article

Tosin A. Adesuyi and Byeong Man Kim*

A Neuron Noise-Injection Technique for Privacy Preserving Deep Neural Networks

<https://doi.org/10.1515/comp-2020-0133>

Received Nov 11, 2019; accepted Apr 04, 2020

Abstract: Data is the key to information mining that unveils hidden knowledge. The ability to revealed knowledge relies on the extractable features of a dataset and likewise the depth of the mining model. Conversely, several of these datasets embed sensitive information that can engender privacy violation and are subsequently used to build deep neural network (DNN) models. Recent approaches to enact privacy and protect data sensitivity in DNN models does decline accuracy, thus, giving rise to significant accuracy disparity between a non-private DNN and a privacy preserving DNN model. This accuracy gap is due to the enormous uncalculated noise flooding and the inability to quantify the right level of noise required to perturb distinct neurons in the DNN model, hence, a dent in accuracy. Consequently, this has hindered the use of privacy protected DNN models in real life applications. In this paper, we present a neuron noise-injection technique based on layer-wise buffered contribution ratio forwarding and ϵ -differential privacy technique to preserve privacy in a DNN model. We adapt a layer-wise relevance propagation technique to compute contribution ratio for each neuron in our network at the pre-training phase. Based on the proportion of each neuron's contribution ratio, we generate a noise-tuple via the Laplace mechanism, and this helps to eliminate unwanted noise flooding. The noise-tuple is subsequently injected into the training network through its neurons to preserve privacy of the training dataset in a differentially private manner. Hence, each neuron receives right proportion of noise as estimated via contribution ratio, and as a result, unquantifiable noise that drops accuracy of privacy preserving DNN models is avoided. Extensive experiments were conducted based on three real-world datasets and their results show that our approach

was able to narrow down the existing accuracy gap to a close proximity, as well outperforms the state-of-the-art approaches in this context.

Keywords: privacy, differential privacy; deep neural networks; perturbation; security

1 Introduction

The recent breakthrough in the area of image recognition, text processing, and data classification has been attributed to the depth of deep neural network hidden layers and its nonlinearity characteristics to perform complex task. The role of rich informative dataset in the process cannot be overemphasized as it contains records that enable good features extraction by DNN models. However, some of these records contain sensitive information and thus, require application of privacy preserving techniques that safeguard it from malicious attacks. Social networks and search engine companies harvest huge amounts of data from their respective users and leverage on the computational capacity of GPU farms to deploy DNN models on a massive scale [1]. The astonishing results of these models (F_θ) have led to their utility in precise speech recognition [2] and image recognition that surpass that of humans [3]. This success had been lauded by many authors, however the makeup of these DNN models contains social network users' personal information, and this can lead to privacy violation when subjected to model-inversion attacks that can reveal details which may subsequently be used for malicious purposes. A case of an adversary with the capability to derive portions of training data from a neural network model via model-inversion attack has been cited in literature [4]. Model inversion and model extraction attacks are the most common attacks on DNN models. In model inversion, an adversary uses θ (a model parameter) to infer information about entities in the training dataset, while in the latter, an adversarial client attempt to learn the nearest approximation of F_θ in fewest queries [5]. Hence, there is a need to protect DNN models and their parameters in order to preserve sensitive information in their training dataset.

*Corresponding Author: Byeong Man Kim: Department of Software Engineering, Kumoh National Institute of Technology, Gumi, South Korea; Email: bmkim@kumoh.ac.kr

Tosin A. Adesuyi: Department of Software Engineering, Kumoh National Institute of Technology, Gumi, South Korea; Email: atadesuyi@kumoh.ac.kr

Recently, studies in [6–10] proposed some techniques for preserving privacy in deep neural networks. The techniques include fully homomorphic encryption (FHE) [8], (ϵ, δ) -differential privacy [11], and the adaptive Laplace mechanism [8] to mention just a few. Privacy preserving DNN models does exhibit lower accuracy as compared to non-privacy preserving counterparts due to enormous noise and lack of techniques to identify points to inject noise without denting accuracy and trading-off privacy. Consequently, this has discouraged the full adoption of privacy protected deep neural networks in today's industries and a contention in the research domain that has not been resolved. Prominent stated-of-the-art methods that offer solutions to the aforementioned are ϵ -differential privacy [12] and layer-wise relevance propagation (LRP) [13]. ϵ -differential privacy model maintains privacy to an extent where nothing is revealed about distinct records in a dataset to an adversary within the model (high confidentiality) [14]. According to the author in [15], differential privacy is regarded as a process of adding consciously fabricated noise into the response to a query, and with a mathematical proof that a re-identifier agent will not be able to disclose info about the identities in the database. The technique has also been certified as the best practice to circumvent re-identification attacks synonymous with the release of microdata [16]. More so, it has been applied in healthcare to guarantee privacy during health data aggregation from cloud servers [17], and in wireless sensors, to preserve privacy of node locations which are regarded as sensitive information [18]. The Layer-wise Relevance Propagation (LRP) technique is a technique used to compute the impact of a particular image region on the output of a classifier [13]. The technique has been seen to be very efficient to decompose image prediction pixel-by-pixel [19] and local renormalization of layers in deep neural networks [20]. In our work, we adapt this mechanism to estimate the contribution ratio of each distinct neuron to the output value produced from our deep neural network.

In this paper, we propose a neuron noise-injection technique based on layer-wise buffered contribution ratio forwarding approach to ensure privacy of DNN models. The contributions of this paper are summarized as follows:

- We adapt the LRP approach to estimate neuron contribution ratio in the DNN layer-by-layer such that computations within neurons are influenced according to their contribution ratio.
- Noise flooding is used to preserve privacy in DNN models and thus decline accuracy. Our approach eliminates this by applying the Laplace mechanism to fabricate noise and redistribute this noise into a

tuple based on the proportion of each neuron's contribution ratio from the pre-trained network. We subsequently inject this into our DNN to preserve privacy without a significant drop in accuracy.

The rest of the paper is organized as follows: Section 2 reviews background on ϵ -differential privacy, deep neural network, LRP, and highlighted related works. In Section 3 and 4, we presented our methodology, experimental results and discussion, while Section 5 concludes the paper.

2 Background

In this section, an overview of differential privacy, fundamental principles of deep neural networks, and a brief on layer-wise relevance propagation are presented. It also highlights various related works in the context of our research.

2.1 ϵ -Differential Privacy

ϵ -differential privacy was proposed by authors in [12] as a strong privacy preserving approach which can be enforced using the Laplace mechanism [21]. The approach uses random noise to guarantee that noticeable information does not change much even when a record is altered in the database [22, 23]. ϵ -differential privacy is currently perceived as an incontestable mechanism to guarantee privacy [24] in an advent of a malicious intruder having detailed knowledge about a particular database. It served as the privacy currency that a data owner can use to control the extent to which sensitive information can be revealed [25]. This technique is valid if the result of a mechanism over a dataset is indistinguishable in the existence and non-existence of a tuple in a dataset [26]. The technique is put into practice using the Laplace mechanism to inject calculated noise into a query with its magnitude quantified by privacy budget ϵ . A Formal definition for differential privacy is stated as follows:

Definition 1 ((ϵ) -differential privacy [12, 27])). . A randomized mechanism M is said to satisfy ϵ -differential privacy iff datasets D and D' differ by one entity and have all sets of possible outputs Y that conform to the following:

$$e^{-\epsilon} \leq \frac{\Pr[M(D) = Y]}{\Pr[M(D') = Y]} \leq e^{\epsilon}.$$

ϵ is a parameter that defines privacy budget such that the smaller the value of ϵ , the stronger the privacy enforced and vice versa. From existing works, several values of ϵ

have been proposed and used. However, the model proposed by [28] recommends that the value of ϵ less than 1 is preferable, consequently guaranteeing strongest privacy. In practice, ϵ -differential privacy is executed by injecting fabricated Laplace noise into the output of a query with a view to blur the sensitivity of a real-valued function [12, 24]. The maximum differences in the addition or deletion of a single record is regarded as the sensitivity of a real-valued function:

Definition 2. Let f be a function $f : D \rightarrow R^d$ over an arbitrary domain D , where sensitivity of f satisfies

$$\Delta f = \max_{D, D'} \sum_{i=1}^d |f(D)_i - f(D')_i| \quad (1)$$

$$= \max_{D, D'} \|f(D) - f(D')\|_1$$

such that D' is a replica of dataset D but differs in values existing in its last tuple, and f is a query function on both D and D' . A Laplace distribution such that it produces a calibrated noise, given sensitivity of a function f , satisfies ϵ -differential privacy [12]:

Theorem 1. Let f be a function $f : D \rightarrow R^d$ over an arbitrary domain D . The computation

$$M(X) = f(X) + \text{Lap} \left(\frac{\Delta f}{\epsilon} \right) \quad (2)$$

guarantees ϵ -differential privacy, where X denotes a record in D and M represents a mechanism that injects Laplace noise into record X to protect its privacy.

2.2 Deep Neural Network

Recent breakthroughs in neural networks have led to the birth of deep neural networks also known as deep learning. Deep learning uses data representation and learning techniques that are based on machine learning, which has gained much research interest [29]. This learning technique has

contributed immensely both in practice and research innovations in the AI field for fault diagnosis from raw sensor data [30], machine language translation, image visualization, speech and image recognition [31], and intrusion detection [29]. Deep neural networks introduced non-linearity learning potentiality that facilitates solutions to complex data [1], and have been shown to deliver superior performance to their traditional counterparts in the AI domain [3, 32–34]. Their internal structure consist of a multiple layer network that enhances their ability to mine

salient features from data in a non-linear pattern. Figure 1 gives a graphical overview of a DNN.

Figure 1: Deep neural network layers

There are three important variables in a DNN, namely: weight w , bias b and input x (data). Apart from the input layer of a DNN, other layers l captures the output of the previous layer as their input as represented in Eq. 3. In order for a network to handle complex computations, nonlinear functionality is introduced via activation function (e.g. ReLU : $\sigma(x) = \max(0, x)$, Sigmod : $y = \frac{1}{(1+e^{-x})}$)

$$x_i^{l+1} = \sigma \left(\sum_{j=1}^k (x_j^l w_{ij}^l) + b_i^{l+1} \right) \quad (3)$$

Classification errors are minimized in a DNN by iteratively training the initial random weights on a set of target training samples. Thereafter, network performance is tested on a separate set of test values [35]. One of the main features of DNN to counter errors and strengthen learning during data training is called backpropagation. DNN computes its gradient through backpropagation by a means of chain rule derivatives after backward propagation of error found at the output layers [36]. Errors (loss) are computed as the differences between the estimated value $\hat{y}$ obtained at the output layer and the target value y known as the label. The cost of the loss is estimated via cross-entropy error function and subsequently fed back into the network model to update parameters such as weights and bias.

2.3 Brief on Layer-wise Relevance Propagation (LRP)

LRP was proposed by Bach *et al.* in 2015 as a means to clarify which pixels of an image play an important role in a

classification decision [20]. LRP denotes the assumption that a classifier can be decomposed into several computational layers. This implies a forward propagation and a backward propagation flow in a neural network. During forward propagation, the image input is captured in pixels form into the network, then multiplied by network weights w with the addition of a bias b , and computationally made nonlinear through an activation function. The process proceeds until it gets to the final layer to produce a value. To illustrate further, we can represent the network output as $F(x)$, where F denotes a classifier and x is the image input in form of pixels $(x_1, x_2, x_3, \dots, x_n)$. For concrete purpose, let us say $F(x) = 1.255$; this value is fed back into the network from the output layer in a reversed manner for backward propagation. This is done to estimate the relevance ratio of each pixel layer by layer up to the input layer where the process is terminated.

Pixel relevance is passed on as a message either forward ($\rightarrow$) or backward ($\leftarrow$) between layers (l and $l+1$). A sample scenario is given in Figure 2 to expound the idea of the LRP process in a neural network. In the forward propagation (see Figure 2a), neuron n receives pixel value x_1 and then computes $r_{n \rightarrow z}^{(l, l+1)} = \sigma(x_1 \cdot w + b)$, thereafter forwards it as a message to neuron z . x_2 likewise goes through the same process to arrive at neuron z as $r_{s \rightarrow z}^{(l, l+1)} = \sigma(x_2 \cdot w + b)$. At z , $F(x)$ is estimated as the total relevance. This is subsequently passed back into the network as R_z to compute relevance contributed by n and s as $R_{n \leftarrow z}^{(l, l+1)}$ and $R_{s \leftarrow z}^{(l, l+1)}$ respectively (see Figure 2b). More elaborate details in relation to our work are further given in Figure 3 & 4.

Figure 2: LRP flows: (a) forward propagation (b) backward propagation

Figure 3: A computational exposition on internal structure of forward and backward flow process. (a) Forward flow with dummy weight values. (b) Backward flow relevance forwarding between neuron z and neuron n . (c) Relevance forwarding and contribution ratio computation for neuron m during backward flow process

Figure 4: An internal structure of our proposed model showing a sample scenario for input x_1

2.4 Review of Existing Privacy Preserving Models

The aim of privacy preserving methods are to safeguard sensitive data against a third party (an adversary) who wants to gain access without the data owner's consent. Authors of [37] applied a multi-objective algorithm to hide sensitive information in order to achieve privacy preserving data mining (PPDM). In IoT devices, data may be collected from people without their consent, hence posing privacy risk [38]. However, authors in [39] proposed a user-friendly framework to achieve consents in Bluetooth low-energy IoT devices. The framework is seen to have improved the privacy policy agreement in IoT appliances. Also, a survey of privacy enhancing technologies (PETs) to mitigate privacy risk, privacy legislation, and their challenges in IoT is found in [38]. Authors in [40] gave an overview of approaches to safeguard privacy in the industrial internet of things (IIoT), and emphasized a secure algorithm for endpoints. Several approaches have been proposed on privacy preserving DNN models, data mining, biometric recognition systems, IoT environment and wireless sensor networks by many authors. Some of these approaches include the multiple objective particle swarm optimization technique that relies on density clustering approach to hide sensitive information (CMPSO) [41], DeepZeroID based on homomorphic encryption [42], hierarchical-clustering mechanism [43], and differential privacy [17, 18]. The differentially private models sometimes performed learning on clean data and use either the Laplace mechanism or an exponential mechanism to preserve privacy in the model [10]. A Sample approach to exponential mechanism can be found in [44, 45], while that of the Laplacian mechanism are seen in work done by authors in [46–48]. A modified ϵ -differential privacy called parameter-wise ϵ -differential was proposed in [7] based on three approaches: anonymized-first, learning-first, and anonymized-learning. In the anonymized-first approach, the original database was first anonymized before training, while in the learning-first approach, machine learning process was carried out first, then the model was anonymized. However, in the anonymized-learning approach, all parameter values of the machine learning were anonymized in the training process. Differentially private stochastic gradient descent was proposed in [6] by clipping the norm of the gradient for each example. Noise was subsequently added to the gradient of each training batch before an update. The approach was combined with PCA during pre-training process, resulting in a differentially private PCA. Authors of [49] applied ϵ -differential privacy in deep auto-encoder by distorting the objective functions rather than

its output. Their work was based on the prediction of human behavior in health social networks. [8] proposed an adaptive Laplace mechanism by combining differential privacy and layer-wise relevance propagation (LRP). The relevance obtained through the LRP process was altered with calculated noise to preserve privacy in the training model. Also, authors in [9] proposed a fully homomorphic encryption (FHE) technique to improve on Cryptonets' solution. Their approach was to make cryptonet (CNN based) layers deeper to improve accuracy through the use of low multiplicative polynomial approximation that lessens the computational cost of FHE.

3 Methodology

In this paper we proposed a neuron noise-injection approach for privacy preserving deep neural networks. The approach is based on neuron contribution ratio and noise-tuple perturbation factory adapted from ϵ -differential privacy. Our focus is to combine these two techniques and infuse them into a deep neural network model to foster classification accuracy and to guarantee privacy. The primary goal of noise-tuple perturbation factory is to fabricate noise needed to protect the training model while the neuron contribution ratio in summary is aimed at estimating the required amount of noise to be injected into a neuron in the deep neural network.

3.1 Overview

The main focus of this paper is to present an approach that closes the accuracy gap between a differentially private DNN and a non-private DNN model and preserve privacy at the same time. To achieve this, there is a need to employ a technique that evaluates computations at neuron points within the neural network and also computes the right amount of noise for privacy preservation without degrading accuracy of the network. Therefore, we adapt an LRP technique to estimate contribution ratio of each distinct neuron layer by layer at the pre-training phase of our DNN. The output of this process is a tuple of contribution ratio. The contribution ratio tuple is then buffered along with its layer mapping identity and then forwarded to the noise-tuple perturbation factory (see Figure 4). In the noise-tuple perturbation factory, the contribution ratio tuple is used to generate a tuple of noise via a Laplace mechanism. The noises in the tuple are not of the same proportion but are based on the contribution ratio from

the pre-training phase, thus, are injected into neurons having their corresponding layer mapping identity in the actual training network. Finally, we also protect the loss function to preserve the parameters of the training phase. We tested our approach using tensorflow libraries and the results shows the effectiveness of our proposed method in closing up accuracy disparity and delivery of strong privacy preservation based on the privacy budget used.

3.2 Neuron Contribution Ratio Propagation

The neuron contribution ratio is originally adapted from Layer-wise Relevance Propagation (LRP): an approach which has been proposed and adopted in literature [8, 13, 19, 20]. The contribution ratio propagation involves a forward flow process that produces F_θ at the output layer and a backward flow process that feeds back F_θ as Cr_z into the network in a reversed manner, as presented in [20]. The Cr_z is used to compute neuron contribution ratio at each layer of the network until it reaches the input layer. The purpose of the neuron contribution ratio is to avoid noise flooding into the network neuron by using its value to compute appropriate quantity of Laplace noise required to be injected into the neurons of our training network as denoted in Eq. 19. An internal structure of this whole process is presented in Figure 4. For clarity, we consider the forward and backward flow process scenario of input tuple (x_1, x_2, x_3) from either dataset D or D' with emphasis on x_1 for the simplicity of neuron flow. The distinction between these two datasets is that they differ only in their last tuple. For example, D' is a duplicate of D but the last record in D' is substituted with another record that differs from what exists in D . In the forward flow, each neuron (p, j, v, m, k, t, n, s) forwards its computational values to the neurons in the next layer until they get to the final neuron z to produce F_θ as shown in Figure 3a. This flow is computed using Eq. 3. The F_θ is passed back as total contribution ratio Cr_z into the network, and this marks the beginning of backward flow process. The backward flow enables the network to compute ratio to what each neuron has contributed during the forward flow. This was not possible in the forward flow process because total contribution ratio is required to be initially calculated and then used to compute ratios contributed by distinct neurons. The value of Cr_z is passed to neuron n in the preceding layer to compute its contribution ratio $Cr_n^{(L-1)}$. Subsequently, neuron n passes on its value $R_{m \leftarrow n}^{(l+1, L-1)}$ to neuron m as shown in a sample approach given in Figure 3b & 3c. Computationally we can depict the backward flow as a tracking of neuron

forward contribution such that from Figure 3 we have:

$$r_{n \rightarrow z} = x_n \cdot w_z \quad (4)$$

$$r_z = r_{n \rightarrow z} + r_{s \rightarrow z} + b_z \quad (5)$$

$$R_{n \leftarrow z}^{(L-1, L)} = \begin{cases} \frac{r_{n \rightarrow z}}{r_z + \beta} \cdot Cr_z, & r_z \geq 0 \\ \frac{r_{n \rightarrow z}}{r_z - \beta} \cdot Cr_z, & r_z < 0 \end{cases} \quad (6)$$

$$Cr_n^{(L-1)} = R_{n \leftarrow z}^{(L-1, L)} \quad (7)$$

where r_z , $r_{n \rightarrow z}$, $r_{s \rightarrow z}$, and $R_{n \leftarrow z}^{(L-1, L)}$ denote relevance of z , relevance forwarded by neuron n to z , relevance forwarded by neuron s to z , and relevance pass-back from z (in layer L) to n (in layer $L-1$) respectively (see Figure 3b).

Let $r_{m \rightarrow n}$ represent relevance forwarded by m to n , $r_{(l+1) \rightarrow n}$ denotes all neurons relevance in layer $l+1$ passed to n , and $Cr_m^{(l+1)}$ depicts contribution ratio at m such that:

$$r_{m \rightarrow n} = x_m \cdot w_n \quad (8)$$

$$r_{(l+1) \rightarrow n} = \sum_{h \in (l+1)} r_h + b_n \quad (9)$$

$$R_{m \leftarrow n}^{(l+1, L-1)} = \begin{cases} \frac{r_{m \rightarrow n}}{r_{(l+1) \rightarrow n} + \beta} \cdot Cr_n^{(L-1)}, & r_{(l+1) \rightarrow n} \geq 0 \\ \frac{r_{m \rightarrow n}}{r_{(l+1) \rightarrow n} - \beta} \cdot Cr_n^{(L-1)}, & r_{(l+1) \rightarrow n} < 0 \end{cases} \quad (10)$$

$$Cr_m^{(l+1)} = \sum_{q \in (L-1)} R_{m \leftarrow q}^{(l+1, L-1)}, \quad (11)$$

where $R_{m \leftarrow n}^{(l+1, L-1)}$ is the relevance pass-back from n to m (see Figure 3c). Furthermore, we compute the contribution ratio at neuron p as $Cr_p^{(l)}$. Hence, we represent relevance forward from p to m as $r_{p \rightarrow m}$, all relevance passed from layer l to m as $r_{l \rightarrow m}$ and, $R_{p \leftarrow m}^{(l, l+1)}$ as the relevance pass-back from m to p . Therefore:

$$r_{p \rightarrow m} = x_p \cdot w_m \quad (12)$$

$$r_{l \rightarrow m} = \sum_{h \in l} r_h + b_m \quad (13)$$

$$R_{p \leftarrow m}^{(l, l+1)} = \begin{cases} \frac{r_{p \rightarrow m}}{r_{l \rightarrow m} + \beta} \cdot Cr_m^{(l+1)}, & r_{l \rightarrow m} \geq 0 \\ \frac{r_{p \rightarrow m}}{r_{l \rightarrow m} - \beta} \cdot Cr_m^{(l+1)}, & r_{l \rightarrow m} < 0 \end{cases} \quad (14)$$

$$Cr_p^{(l)} = \sum_{g \in (l+1)} R_{p \leftarrow g}^{(l, l+1)}. \quad (15)$$

β is used as a stabilizer to control the value of r as in [7]. The process continues until Cr_{x_1} , Cr_{x_2} and Cr_{x_3} are reached (see Figure 4) and then stops. The final product of this technique in a single complete circle (batch B of X) is a set of tuples of neuron contribution ratio nCr_x . This is further normalized to obtain a single tuple contribution ratio at each layer over the entire dataset Cr_D .

$$Cr_B = \frac{1}{|B|} \sum_{x \in B} nCr_x \quad (16)$$

nCr_x can be expressed further as $nCr_x = ((Cr_{x_1}, \dots, Cr_{x_n})_1, \dots, (Cr_{x_1}, \dots, Cr_{x_n})_j)$.

$$Cr_D = \frac{|B|}{|D|} \sum_{B \in D} Cr_B = \frac{1}{|D|} \sum_{B \in D} \sum_{x \in B} nCr_x \quad (17)$$

where $Cr_D = (Cr_{x_1}, Cr_{x_2}, Cr_{x_3})$.

During the process of capturing tuple Cr_D , the neuron contribution ratio is buffered at every layer of the DNN. Details about this process are presented in the next section.

3.3 Contribution Ratio Buffer

The essence of contribution ratio buffer Cr_b is to trap contribution ratio Cr from set of neurons in each hidden layer. The Cr_b is placed at every hidden layer to store a tuple of Cr^I for all neurons in a particular layer I (I is used later in this paper to represent any layer in the network in order to simplify notations) during backward flow process of relevance propagation. Datasets D and D' go through this process. The storing procedure is illustrated using Figure 4 as:

$$\begin{aligned} Crb^L &= Cr_z^{(L)} \\ Crb^{L-1} &= \{Cr_n^{(L-1)}, Cr_s^{(L-1)}\} \\ Crb^{l+1} &= \{Cr_m^{(l+1)}, Cr_k^{(l+1)}, Cr_t^{(l+1)}\} \\ Crb^l &= \{Cr_p^{(l)}, Cr_j^{(l)}, Cr_v^{(l)}\}. \end{aligned}$$

Thereafter, all Crb are merged into a single tuple Crb_{merge} and Cr'_{merger} for D and D' respectively, and are subsequently forwarded to the noise-tuple perturbation factory as:

$$Crb_{merge} = \{Crb^l \cup Crb^{l+1} \cup Crb^{L-1} \cup Crb^L\}$$

as in the case of D .

3.4 Noise-Tuple Perturbation Factory

This phase receives tuple of Crb_{merge} , Cr'_{merger} , and neighboring datasets D and D' as input, and captures the

total value of Cr as Cr_{total} and Cr'_{total} . It then computes the global sensitivity Δ_f based on the maximum value of D and D' . However, in this paper, our mathematical representation focuses mostly on dataset D . Since both datasets undergo the same process, therefore, the representations can easily be adapted for dataset D' by simply adding apostrophe symbol (') to it.

$$Cr_{total} = \sum_{Cr^I \in Crb_{merge}} Cr^I \quad (18)$$

Furthermore in the noise-tuple perturbation factory, we calculate the noise spreading proportionality δ for each Cr^I in the Crb_{merge} using the Laplace mechanism and a chosen privacy budget ϵ_1 such that I represents an instance of a layer corresponding to the neuron where Cr was estimated.

$$\delta^I \leftarrow \left(Lap \left(\frac{\Delta_f}{\epsilon_1} \right) \times \frac{Cr^I}{Cr_{total}} \right) \quad (19)$$

$$\delta_{tuple} \leftarrow \delta^I \quad (20)$$

Hence a tuple of δ is yielded: $\delta_{tuple} = \{\delta_1^I, \delta_2^I, \delta_3^I, \dots, \delta_H^I\}$ as illustrated in Algorithm 1. The δ_{tuple} consists of proportioned noise to be injected into neurons of our network training phase shown in Figure 4. It should be noted that the quantity of these noises differs based on the contribution ratio value of each neuron ω . Each neuron can be further expressed as:

$$\begin{aligned} \omega_i^I &= \sum_{j=1}^k (x_j \cdot w_{ij}) + b_i \text{ where } x_j \in D \\ F_\theta &= \{\omega_1^I, \omega_2^I, \omega_3^I, \dots, \omega_H^I\} \end{aligned} \quad (21)$$

Algorithm 1 Noise Perturbation Factory

- 1: **function** noiseFabrication(D, D', Crb_{merge})
 - 2: # let Crb_{merge} represent contribution ratio set for dataset D and D' respectively:
 - 3: compute global sensitivity Δ_f using equation 1:
 - 4: $\Delta_f = \max_{D, D'} \|f(D) - f(D')\|_1$
 - 5: compute total Cr : $Cr_{total} = \text{sum}(Crb_{merge})$ #Eq. 18
 - 6: #initialize privacy budget ϵ : ϵ_1
 - 7: #compute noise spread proportionality δ from equation 19 & 20
 - 8: **for** $Cr^I \in Crb_{merge}$ **do**
 - 9: $\delta^I \leftarrow \left(Lap \left(\frac{\Delta_f}{\epsilon_1} \right) \times \frac{Cr^I}{Cr_{total}} \right)$
 - 10: $\delta_{tuple} \leftarrow \delta^I$
 - 11: **end for**
 - 12: **return** $\delta_{tuple} = (\delta_1^I, \delta_2^I, \delta_3^I, \dots, \delta_H^I), \epsilon_1$
-

F'_θ is also computed in a similar manner. Therefore, we preserved distinct ω of our DNN model F_θ in the training phase with the Laplace noise-injection mechanism Φ as:

$$\begin{aligned}\Phi(D) &\approx \Phi(F_\theta) \\ &= \{(\omega_1^I + \delta_1^I), (\omega_2^I + \delta_2^I), (\omega_3^I + \delta_3^I), \dots, (\omega_H^I + \delta_H^I)\}.\end{aligned}$$

Similarly to authors in [21], we ensure that the Φ process on each neuron in F_θ is ϵ_1 -differentially private based on lemma 1.

Lemma 1. Let D and D' be any two neighboring datasets that yield DNN models F_θ and F'_θ respectively. Let $\Phi(D)$ and $\Phi(D')$ be the noise-injection mechanism on neurons in F_θ and F'_θ respectively given response r . There exist an inequality that satisfies ϵ_1 -differentially private condition based on definition 1 in section 2.1:

$$e^{-\epsilon} \leq \frac{\Pr[\Phi(F_\theta) \in r]}{\Pr[\Phi(F'_\theta) \in r]} \leq e^\epsilon.$$

Proof. Suppose F_θ and F'_θ models are from dataset D and D' respectively and are differentiated by neurons $(\omega_H^I \& \omega_H'^I)$ in the last tuple. Without loss of generality, there exist the assumption that the Laplace noise injection process Φ with privacy budget ϵ_1 on the models, yielded a response r with a probability less or equal to e^ϵ such that:

$$e^{-\epsilon} \leq \frac{\Pr\left[\left(\omega_H^I + \text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right)\right) \in r\right]}{\Pr\left[\left(\omega_H'^I + \text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right)\right) \in r\right]} \leq e^\epsilon,$$

where $\text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right)$ follows a Laplace distribution to generate noise denoted as δ^I , and $\delta_H^I \in \delta_{tuple}$ is the proportion allotted to ω_H^I based on its C_r

$$\begin{aligned}e^{-\epsilon} &\leq \frac{\Pr\left[\left(\omega_H^I + \delta_H^I\right) \in r\right]}{\Pr\left[\left(\omega_H'^I + \delta_H^I\right) \in r\right]} \leq e^\epsilon \\ e^{-\epsilon} &\leq \frac{\Pr\left[\left(\omega_1^I + \delta_1^I\right), \dots, \left(\omega_H^I + \delta_H^I\right) \in r\right]}{\Pr\left[\left(\omega_1'^I + \delta_1^I\right), \dots, \left(\omega_H'^I + \delta_H^I\right) \in r\right]} \leq e^\epsilon \\ e^{-\epsilon} &\leq \frac{\Pr\left[\Phi\left(\omega_1^I, \omega_2^I, \omega_3^I, \dots, \omega_H^I\right) \in r\right]}{\Pr\left[\Phi\left(\omega_1'^I, \omega_2'^I, \omega_3'^I, \dots, \omega_H'^I\right) \in r\right]} \leq e^\epsilon \\ e^{-\epsilon} &\leq \frac{\Pr\left[\Phi(F_\theta) \in r\right]}{\Pr\left[\Phi(F'_\theta) \in r\right]} \leq e^\epsilon\end{aligned}$$

Since both F_θ and F'_θ are derivable from D and D' , then we can also say that:

$$e^{-\epsilon} \leq \frac{\Pr[\Phi(F_\theta) \in r]}{\Pr[\Phi(F'_\theta) \in r]} \leq e^\epsilon \approx e^{-\epsilon} \leq \frac{\Pr[\Phi(D) \in r]}{\Pr[\Phi(D') \in r]} \leq e^\epsilon.$$

We also affirm that the Laplace noise-injection process on neurons ω_i^I implies perturbation on $x \in D$, and it preserves privacy based on lemma 2. $\square$

Lemma 2. Let Φ be a randomized perturbation mechanism on x and x' from neighboring datasets D and D' respectively. The mechanism preserves privacy, for example on D such that:

$$\Phi(D) = \left\{x_i + \text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right), \dots, x_n + \text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right)\right\}.$$

Proof. Let x_n and x'_n represent the last tuple in dataset D and D' . By retaining generalization there exist an assumption that $\omega_H^I(\omega_H'^I)$ are the last neuron of $F_\theta(F'_\theta)$ model. Hence,

$$\Phi\left(\omega_H^I\right) = \{\omega_H^I + \delta_H^I\}_{\omega_H^I \in F_\theta} \approx x_n + \text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right).$$

From Eq. 21, we derived

$$\begin{aligned}\omega_H^I &= \sum_{j=1}^n (x_j \cdot w_{Hj}) + b_H \\ \omega_H^I &= b_H + (xw_H)_1 + \dots + (xw_H)_n \\ \omega_H^I &= b_H + x_n^I \\ \Phi\left(\omega_H^I\right) &= b_H + \left\{x_n^I + \text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right)\right\}.\end{aligned}$$

where b_H can be a negligible bias having value < 1 :

$$\{\omega_H^I + \delta_H^I\}_{\omega_H^I \in F_\theta} \approx b_H + \{x_n^I + \text{Lap}\left(\frac{\Delta_f}{\epsilon_1}\right)\}_{x_n^I \in D}.$$

Having established proof of lemma 1 & 2, we validate that the privacy of tuples (X) in dataset D are protected in a differential privacy manner such that sensitivity is bounded based on lemma 3. $\square$

Lemma 3. Suppose f_D and $f_{D'}$ are query functions over two neighboring datasets D and D' respectively such that we denote their representations as follows:

$$f_D = \sum_{X \in D} X; f_{D'} = \sum_{X' \in D'} X'$$

Then, the following inequality holds:

$$\left\| \sum_{X \in D} X - \sum_{X' \in D'} X' \right\|_1 \leq 2 \max_X \sum_{X \in D} \|X\|_1$$

Proof. Let X_n and X'_n be the last tuple in datasets D and D' respectively. Without loss of generality, there exist an assumption that both datasets differ in the last tuple. Therefore,

$$\begin{aligned}\left\| \sum_{X \in D} X - \sum_{X' \in D'} X' \right\|_1 &= \sum_{i=1}^n \|X_n - X'_n\|_1 \\ &\leq \sum_{i=1}^n \|X_n\|_1 + \sum_{i=1}^n \|X'_n\|_1\end{aligned}$$

$$\leq 2 \max_X \sum_{X \in D} \|X\|_1$$

where n is the number of tuples in D . A detailed process of the training phase in Figure 4 is presented in section 4, 4.2, and 4.3. $\square$

3.5 Perturbation of the Loss Function

Having preserved the privacy of our DNN internal entities, it is expedient to protect the junction at which the target value y is accessed. Hence, we preserve the privacy of the loss function L_θ for our DNN model training phase to prevent an adversary from compromising the training parameter such as weight and bias through the L_θ . Therefore, L_θ is being protected at every training Batch B . This we achieved using loss function given cross-entropy error function as L_θ via estimated value $\hat{y}$, thus:

$$\hat{y} = \sigma(\omega^L) \quad (22)$$

$$L_\theta = - \sum_{i=1}^t y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) L'_\theta \quad (23)$$

is likewise computed and t is a set of tuples that denote a single batch in D . We set loss function sensitivity as:

$$\Delta L = \max_{L_\theta, L'_\theta} \|f_L(L_\theta) - f_L(L'_\theta)\|_1 \quad (24)$$

and fabricate Laplace noise for perturbation with privacy budget ϵ_2 . Therefore we have:

$$\Phi(L_\theta) = f_L(L_\theta) + \text{Lap}\left(\frac{\Delta L}{\epsilon_2}\right). \quad (25)$$

In this paper, we affirm that the task at L_θ at each complete batch of our network is ϵ_2 -differentially private based on the following lemma:

Lemma 4. Let f_L and f'_L denote query functions on two neighboring loss functions L_θ and L'_θ from batches $B \in D$ and $B' \in D'$ respectively. The following inequality exists that predefines their sensitivity as:

$$\Delta L = \left\| \sum_{L_\theta \in D} L_\theta - \sum_{L'_\theta \in D'} L'_\theta \right\| \leq 2 \max_{L_\theta} \sum_{L_\theta \in D} \|L_\theta\|_1$$

Proof. Assume that B and B' are the last batches from D and D' respectively and are differentiated in the last tuple such that $L_{\theta_n}(L'_{\theta_n})$ is the last tuple lost function in $B(B')$

during training phase. There exists an inequality that predefines their sensitivity as:

$$\begin{aligned} \Delta L &= \left\| \sum_{L_\theta \in D} L_\theta - \sum_{L'_\theta \in D'} L'_\theta \right\| = \sum_{i=1}^n \|L_{\theta_n} - L'_{\theta_n}\| \\ &\leq \sum_{i=1}^n \|L_{\theta_n}\| + \|L'_{\theta_n}\| \leq 2 \max_{L_\theta} \sum_{L_\theta \in D} \|L_\theta\|_1 \end{aligned}$$

where n is the number of tuples in D . $\square$

Lemma 5. Equation 25 preserves ϵ_2 -differential privacy even in the presence of adversary query function f_L .

Proof. Let L_θ and L'_θ be any pair loss function. Suppose a noise-injection mechanism Φ having subset $S \subseteq \text{range}(\Phi)$. The probability density on both loss function results to response $r \in S$ is proportional to $e^{-\|f_L(L_\theta)-r\|_1(\frac{\epsilon}{\Delta L})}$ and $e^{-\|f_L(L'_\theta)-r\|_1(\frac{\epsilon}{\Delta L})}$. Hence, we have:

$$\begin{aligned} \frac{e^{-\|f_L(L_\theta)-r\|_1(\frac{\epsilon}{\Delta L})}}{e^{-\|f_L(L'_\theta)-r\|_1(\frac{\epsilon}{\Delta L})}} &= e^{\|f_L(L'_\theta)-r\|_1(\frac{\epsilon}{\Delta L}) - \|f_L(L_\theta)-r\|_1(\frac{\epsilon}{\Delta L})} \\ &\leq e^{\|f_L(L'_\theta)-f_L(L_\theta)\|_1(\frac{\epsilon}{\Delta L})} \leq e^{|\Delta f|(\frac{\epsilon}{\Delta L})} \leq e^\epsilon \end{aligned}$$

$\square$

3.6 Privacy Budget Point Recap

In this section, we further expatiate on the relevant steps of our proposed approach by explicitly stating entries of Laplace noise-injection into our privacy preserving DNN model. Our model has two entry levels which include: at distinct neuron points in the entire hidden layer and at the loss function backpropagation point. At distinct neuron points (see Figure 4: training phase) each neuron is made ϵ_1 -differentially private as a result of the Laplace noise-injection mechanism Φ , which is basically dependent on Algorithm 1 and largely determined by each neuron's contribution ratio Cr . Lemma 1, 2 & 3 and their proofs were used to theoretically validate the privacy preserving process on neurons as ϵ_1 -differentially private. Therefore, the privacy budget for that process is ϵ_1 . Finally, we injected Laplace noise with a privacy budget of ϵ_2 through the loss function using the cross entropy error function. In like manner, we stated lemma 4 & 5 and their proofs to back up our privacy preservation claims. However, the overall privacy budget is $\epsilon = (\epsilon_1 + \epsilon_2)$. Our theoretical proof of privacy preservation base on the given lemmas correspond to what other authors [6–8, 48–50] have done in related literature. Algorithm 2 gives an overview of cogent steps in our approach.

Algorithm 2 Perturbation Model

```

1: function PerturbModel ( $D, D', Crb_{merge}, T$ )
2: # total number of batches  $T$ 
3:  $\delta_{tuple}, \epsilon_1 = \text{noiseFabrication}(D, D', Crb_{merge})$ 
4:  $\delta_{tuple} = \{\delta_1^I, \delta_2^I, \delta_3^I, \dots, \delta_H^I\}$ 
5: #compute each neurons in the network using equation 21
6:  $\omega_i^I = \sum_{j=1}^k (x_j \cdot w_{ij}) + b_i$  where  $x_j \in D$ 
7: #add Laplace noise for neuron perturbation
8:  $\Phi(D) \approx \Phi(F_\theta) = \{(\omega_1^I + \delta_1^I), (\omega_2^I + \delta_2^I), (\omega_3^I + \delta_3^I), \dots, (\omega_H^I + \delta_H^I)\}$ 
9: #compute estimate value  $\hat{y}$  from equation 22
10:  $\hat{y} = \sigma(\omega^L)$ 
11: randomly initialize  $\theta_0$ 
12: for each  $t \in [T]$  do
13:   #select a random training batch of neuron sample  $\omega_{B_t}$  and perturb via Laplace noise injection
14:    $L_{\theta_t} = -\sum_{i=1}^{B_t} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$  #Eq. 23
15:    $\Delta L = \max_{L_{\theta_t}, L'_{\theta_t}} \|f_L(L_{\theta_t}) - f_L(L'_{\theta_t})\|_1$  #Eq. 24
16:    $\Phi(L_{\theta_t}) = f_L(L_{\theta_t}) + \text{Lap}(\frac{\Delta L}{\epsilon_2})$  #Eq. 25
17:   #compute the gradient decent
18:    $\theta_{t+1} \leftarrow \theta_t - \eta_t L_{\theta_t}$  #  $\eta$  is the learning rate
19: end for
20: return  $\theta_T$  and compute total privacy budget ( $\epsilon = \epsilon_1 + \epsilon_2$ )

```

3.7 Data Utility Concern

Differential privacy has two major data processing models, namely non-interactive and interactive models. In the non-interactive model, the publisher (a person who released the model) allows client to have direct access to the privacy preserved dataset, while in the interactive model, clients are given access to the data via an interface that secure privacy [51]. In the two categories, having strong privacy guarantee and at the same time retaining high data utility is a major concern. This is because data utility depends on meaningful query result upon request by a genuine client who has same access point with an adversarial client. Our own processing model is a little different from the aforementioned processes because clients do not have direct or intermediate interaction with the dataset but only with the published model F_θ or trained parameters θ . Also, it is the client that supplies data to these entities for prediction or classification. However, an adversary masquerades as a genuine client and subjects F_θ or θ to various attacks to unveil the training data. Therefore, we protect these entities with a differential privacy technique during the DNN training phase such that they are usable for pre-

diction/classification purposes. Therefore, data utility is not a concern in privacy preserving deep neural network as the classification accuracy of our private model presented in the experiment section justifies this (its utility).

4 Experimental Results and Discussion

In this section, we experiment our proposed model on two numerical datasets, Dermatology and Wiscconsin Diagnosis Breast Cancer (WDBC) accessed from UCI Machine Learning Repository [52, 53]. Both datasets have been used in existing literature [54–57] and have been found to be effective for classification process. The dermatology dataset consists of 6 classes of erythematous squamous disease, 12 clinical and 22 histopathological measurements of patients, resulting in a total of 34 attributes (ID, diagnosis, 30 real-valued input features) with 366 instances. The WDBC consist of 2 classes: malignant and benign with 32 attributes and a total of 569 instances. For both datasets, 10-fold cross-validation was used. In our experiment, we developed a non-private DNN model and differentially private DNN based on varying values of ϵ , and further made a comparative analysis based on the accuracy achieved from both models. Furthermore, we evaluate the performance of our approach with existing method in [8] as it is relatively close to ours. We finally benchmark our result with the state-of-the-art accuracy for MINST dataset. The MINST dataset consists of 10 classes (digits), 60,000 training examples and 10,000 testing examples of handwritten digit recognition images [58]. Each sample is of size 28×28 gray-scale image. The dataset has been popularly used in substantial amount of literature [6, 8, 9] as a benchmark to validate private and non-private model accuracy differences. All experiments were conducted using Tensorflow libraries on a single NVIDIA GTX 1080 Ti 16GB GPU, 32GB RAM, core i7 with 3.60GHZ CPU. All graphs and table results in this section are specifically for dataset D because it is our focus dataset, while the essence of D' is to assist in accomplishing differential privacy in the case of the private model (privacy preserving model).

4.1 Pre-Training Process of Contribution Ratio

The essence of the pre-training process is to compute neuron contribution ratio. This process is targeted towards the private model only, as a means to estimate neuron con-

tribution ratio needed to quantify Laplace noise to be injected into network neurons in the actual training phase. We designed a DNN architecture just as the real training process. The network architecture has the same number of layers and neurons as the actual learning process but a smaller epoch size. We monitored closely and recorded each computations layer-by-layer through the Tensorflow session variables, and tracked every computation output at the activation function point. Values tracked include: weights, bias, and outputs from the computation of Eq. 3. None of these entities is single-valued but in tuples of matrices, and each value in the tuples represents a distinct neuron in the network. The task on entities involves matrix computation which may be too complex and could distract from the core of this paper. The pre-training process was carried out in a forward flow phase which terminated at the output layer (see Figure 4). The estimated value F_θ produced was feed into the network to proceed the backward flow process. The F_θ together with values of the tracked entities were used to compute contribution ratio Cr of each neuron of the network. In order to make this support application of differential privacy at the training phase, there is a need to have two neighboring datasets that differ only in the last tuple. The reason behind this is to maintain privacy within a dataset in the absence or presence of a tuple (record). More so, it makes it possible to estimate global sensitivity Δ_f from Eq. 1 and loss function sensitivity ΔL from Eq. 24. Based on this, we duplicate each of our datasets as D & D' and alter the last tuple in D' . We wrote a pre-training object-oriented class in python that executes Eq. 4-15 and tracks all needed values using Tensorflow library and later returns them. Then, we called the class twice by passing D & D' one after the other. Thereafter, we were able to capture $Cr_{b_{merge}}$ and $Cr_{b'_{merger}}$. These were forwarded to the noise-tuple factory to compute required noise to preserve privacy at the training phase. The internal computations are graphically presented in Figure 3 and also the process flow overview is given in Figure 4. We likewise followed this class paradigm during the actual training process of our datasets to build a private model as discussed in the following subsections (4.2, 4.3, and, 4.4). The class structure affords us a convenient way to execute algorithm 1, track loss functions (L_θ & L'_θ) and finally compute Eq. 24. These were a challenge for us without adopting the aforementioned class paradigm. We observed that the entire pre-processing phase does not incur extra significant computational cost different from the real training process. It was actually lower because a smaller epoch size was required.

4.2 Dermatology Dataset

We designed two models with the same network structure using the dermatology dataset. The training phase for both the non-private and the differentially private DNN models had 34 input features, 3 hidden layers with a ReLU activation function at each hidden layer, and also an output layer of 6 classes along with a cross-entropy loss function. The same architecture was also used for the pre-training aspect of the private DNN: for the computation of neuron's contribution ratio Cr . During training in the non-private DNN, we set the learning rate to $10e-3$ using adam optimizer to minimize the loss. We also initialized a batch-size of 50 in 50 epochs for over 2000 iteration steps. Consequently, from Figure 5a, a 99.2% prediction accuracy was reached. However, in the privacy preserving model, the epoch size was changed to 100 at 3000 iterations per epoch to facilitate the learning process and enhance prediction accuracy. This is because we recorded a poor result using 2000 iterations. Initializing 3000 iterations per epoch implies that 50 tuples from the dataset go through the training mood (are looped-thru) within each epoch, 3000 times. This technique really boosted learning with good accuracy for the test data. We introduced Laplace noise-injection categorized based on the value of ϵ with respect to the neuron contribution ratio. In Figure 5b, privacy budget is set to $\epsilon = 1$ for moderate Laplace noise; in Figure 5c, $\epsilon = 0.5$ for much noise; while in Figure 5d, for heavy noise, $\epsilon = 0.1$. As a result, prediction accuracy of 98.6%, 96%, and 95.6% respectively was recorded. As previously mentioned, the smaller the value of ϵ , the larger the noise that is produced. Therefore, neurons with lower Cr receive a larger portion of the noise. It is deducible from figs. 5a-b-c-d, that in the first 20 epochs, the non-private model achieves higher accuracy as compared to the private model. Also, the prediction (test) accuracy slightly drops as the fabricated Laplace noise injected increases. However, the training accuracies are almost stable at 99%. Figures 5b, 5c, 5d and Table 1 present a graphical and a summary view of the prediction accuracies. The close in the accuracy gap, despite the

Table 1: Result summary on training and test datasets

Differentially private model	Prediction Accuracy (%)	
ϵ value	dermatology	breast cancer
$\epsilon = 1$	98.6	99.0
$\epsilon = 0.5$	96.0	98.2
$\epsilon = 0.1$	95.6	96.0
Non-private model	99.2	99.5

Figure 5: Dermatology dataset prediction accuracy for non-private and privacy preserving DNN

increase in injected noise, shows the efficacy of the neuron contribution ratio as a means of minimizing accuracy degradation in the case of heavy noise.

4.3 Breast Cancer Dataset

As in the previous dataset, we designed a non-private and a differentially private DNN models with the use of breast cancer dataset as training samples. Both models had 30 input features, 3 hidden layers along with ReLu function, and an output layer of 2 classes. A batch size of 100 and 50 epoch size over 2000 iteration steps with a cross-entropy

loss function was used in the non-private DNN model. However, for the private model, we learnt from the experience of the previous dataset and increased the number of iteration steps to 3000 within an epoch to ensure a good learning process in the face of noise-injected neurons. A total of 100 epochs was used during training. As shown in Figure 6a, the non-private model achieved a prediction accuracy of 99.5%, meanwhile in the privacy preserving model, as similarly done in the dermatology dataset, we set $\epsilon = 1$ to yield a prediction accuracy of 99%, and later introduced much noise: having $\epsilon = 0.5$, the accuracy slightly dropped to 98.2%. We decided to test the tolerance performance of our model in the face of heavy noise with $\epsilon = 0.1$,

Figure 6: Breast cancer dataset prediction accuracy for non-private and privacy preserving DNN

and a 96% prediction (test) accuracy was recorded. These privacy budgets ϵ were used because strong differential privacy are best achieved within this range [28]. Although our model accuracy dropped, the accuracy gap between the non-private model and the privacy preserving model is still very small (99.5%: 99%, 98.2%, 96%) as shown in Table 1 and Figure 6b, 6c, & 6d. It is observable that in the first few epochs, the non-private model attained a better accuracy than the privacy preserving model, however after a few more epochs, the accuracy shoot up in both training and testing samples. Neuron Cr did not only play an important role by preventing noise flooding on all neurons in the network but also redirected a larger portion of the injected

Laplace noise to neurons with lesser Cr and vice versa. As a result, accuracy deterioration when noise increases was minimized.

4.4 Performance Evaluation with MNIST Dataset

Since MNIST already has benchmark accuracy for non-private model, we only focus on the private model. We design our privacy preserving DNN model with two convolution layers of 32 and 64 features respectively, two fully-connected layers and an output layer of 10 classes. Be-

(a) much noise $\epsilon = 0.5$ (b) heavy noise $\epsilon = 0.1$

Figure 7: MNIST dataset prediction accuracy for privacy preserving DNN

tween the two fully-connected layers, we introduced a batch normalization technique and also a ReLU activation function after every layer except the output layer where we used a softmax activation function. Initially, dropout was used and the result we got was far from the benchmark. We later introduced a batch normalization technique and we achieved a desired result. Also, a batch size of 1800 was used in 25 epochs. We set our model learning rate to $10e-4$, and within each epoch we set 100 iterations to boost the training accuracy. Our privacy budget ϵ was set to $\epsilon = 0.5$ and $\epsilon = 0.1$ for much and heavy noise respectively. Our approach reached a prediction accuracy of 98.9% when much noise was injected, and 98.3% under heavy noise. In Figure 7 we can see in the two graphs (7a & 7b) that our model attained 96% in just 4 epochs and become stable after 15 epochs to reach its best prediction accuracy test. From this experiment, our best performance was when we set $\epsilon = 0.5$. In Table 2, we presented the performance of our privacy preserving model by comparing our results with widely accepted state-of-the-art MNIST accuracy benchmark [59] and N. Phan *et al.*'s AdLM approach best prediction accuracy [8] which has outperformed other similar techniques [6] from existing research. From Table 2, it is clear that our accuracies are closer to those of the non-private MNIST accuracy in all cases. We decided to set the same ϵ value of 0.5 as in AdLM to have a fair comparison. Our model still outperformed N. Phan *et al.*'s result. What we deduced from this is that neuron contribution ratio is a major factor to significantly reduce accuracy degradation and its efficacy lies in the use of noise tuple.

Table 2: Performance on MNIST dataset

Model	Prediction Accuracy (%)
MNIST benchmark	99.70
AdLM ($\epsilon = 0.5$)	93.66
Our Approach	
($\epsilon = 0.5$)	98.90
($\epsilon = 0.1$)	98.30

5 Conclusions

In this paper, we proposed a neuron noise-injection technique to preserve privacy of deep neural networks. The approach is focused on boosting the acceptability of privacy preserving deep neural network in the industry by narrowing down the existing accuracy disparity with non-private deep neural network without jeopardizing privacy. Consequently, a differential privacy technique was introduced with the use of noise tuple which largely depends on the network neuron's contribution ratio. Experiments were conducted over three datasets from dermatology, breast cancer, and MNIST, and the prediction accuracy achieved in the privacy preserving DNN model is in a close proximity to the non-private DNN model, thus, closing down the accuracy gap. This is evident in the AdLM ($\epsilon = 0.5$) approach on the MNIST dataset with existing accuracy gap of 6.04% being narrowed down to 0.8% via our proposed method. Our approach vividly enhanced the effectiveness of differential privacy in deep neural network models with mini-

mal or no accuracy degradation by taking cognizance of neuron contribution ratio. In this research, privacy preservation is only proven theoretically, based on the given lemmas. This approach corresponds to what other authors in the research domain have done. Also, to the best of our knowledge, differential privacy mechanism has only been proven or verified mathematically. However in our future work, we plan to develop a model inversion attack to evaluate privacy within a DNN model protected by differential privacy.

Acknowledgement: This research was supported by Kumo National Institute of Technology (202001260001).

References

- [1] Shokri R., Shmatikov V., Privacy-Preserving Deep Learning, CCS'15 Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, 1310-1321
- [2] Hannun A., Case C., Casper J., Catanzaro B., Diamos G., Elsen E., Prenger R., Satheesh S., Sengupta S., Coates A., et al., Deepspeech: Scaling up end-to-end speech recognition, 2014, arXiv:1412.5567
- [3] He K., Zhang X., Ren S., Sun J., Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, 2015, arXiv:1502.01852
- [4] Fredrikson M., Jha S., Ristenpart T., Model inversion attacks that exploit confidence information and basic countermeasures, In CCS. 2015, 1322-1333 ACM
- [5] Lowd D., Meek C., Adversarial Learning. KDD '05 Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, 2005, 641-647
- [6] Abadi M., Chu A., Goodfellow I., McMahan H. B., Mironov I., Talwar K., Zhang L., Deep Learning with Differential Privacy, 2016, arXiv:1607.00133v2
- [7] Sci Y., Okumura H., Ohsuga A., Privacy-Preserving of Deep Neural Networks, IEE Computer Society, 2016, 1418-1425
- [8] Phan N., Wu X., Hu H., Dou D., Adaptive Laplace Mechanism: Differential Privacy Preservation in Deep Learning, 2018, arXiv: 1709.05750v2
- [9] Chabanne H., Wargny A. D., Milgram J., Morel C., Prouff E., Privacy-Preserving Classification on Deep Neural Network, Sefran Identity & Security, Cryptology ePrint Archive, 2017
- [10] Ji Z., Lipton Z. C., Elkan C., Differential Privacy and Machine Learning: a Survey and Review., 2015 arXiv: 1412.7584v1, 1-30
- [11] He J., Cai L., Differential Private Noise Adding Mechanism and Its Application on Consensus, 2017, arXiv:1611.08936v2 [cs.IT]
- [12] Dwork C., McSherry F., Nissim K., Smith A., Calibrating Noise to Sensitivity in Private Data Analysis, In Proc., Theory of Cryptography, 2006, 265-284
- [13] Binder A., Bach S., Montavon G., Muller K. R., Samek W., Layer-wise Relevance Propagation for Deep Neural Network Architectures, In: Kim K., Joukov N. (eds) Information Science and Applications (ICISA); Lecture Notes in Electrical Engineering, vol. 376, Springer, Singapore, 2016, 913-922
- [14] Phan N., Wu X., Dou D., Preserving Differential Privacy in Convolutional Deep Belief Networks, 2018, arXiv: 1706.8839v2
- [15] Ohm P., Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization, 57 UCLA L. REV., 2010, 1701-1756
- [16] Chin A., Klinefelter A., Differential Privacy as a Response to the Reidentification Threat: The Facebook Advertiser Case Study. 90 N. C. L. REV., 2012, 1452-1454
- [17] Ren H., Li H., Liang X., He S., Dai Y., Zhao L., Privacy-Enhanced and Multifunctional Health Data Aggregation under Differential Privacy Guarantees, Sensors 2016, 16, 1463
- [18] Wang J., Zhu R., Liu S., Cai Z., Node Location Privacy Protection Based on Differentially Private Grids in Industrial Wireless Sensor Networks, Sensors 2018, 18, 410
- [19] Bach S., Binder A., Montavon G., Klauschen F., Muller K. R., Samek W., On Pixel-Wise Explanations for Non-linear Classifier Decisions by Layer-wise Relevance Propagation, PLoS ONE; 2015, 10(7), p. e0130140
- [20] Binder A., Montavon G., Bach S., Muller K. R., Samek W., Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers, 2016, arXiv:1604.00825v1
- [21] Zhang J., Zhang S., Xiao X., Yang Y., Winslett M., Functional Mechanism: Regression Analysis under Differential Privacy, Proceedings of the VLDB Endowment, 2012, 5(11), 1364-1375
- [22] Dwork C., Differential privacy. In Encyclopedia of Cryptography and Security (2nd Ed.), 2011, 338-340
- [23] Dwork C., Kenthapadi K., McSherry F., Mironov I., Naor M., Our data, ourselves: Privacy via distributed noise generation; In International Conference on the Theory and Applications of Cryptographic Techniques. 2006, 486-503
- [24] Friedman A., Schuster A., Data Mining with Differential Privacy, In Proc., ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, 493-502
- [25] McSherry F. D., Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis. In SIGMOD'09: Proceedings of the 2009 ACM SIGMOD International Conference on management of Data, 2009, 19-30
- [26] Li H., Xiong L., Ohno-Machado L., Jiang, X. Privacy Preserving RBF kernel Support Vector Machine, BioMed Research International, 2014
- [27] Charest A., Han Y., On the Meaning and Limits of Empirical Differential Privacy. Journal of Privacy and Confidentiality, 2017, 7(3), 53-66
- [28] Dwork C., Differential privacy: A Survey of Results; In TAMC, 2008, 1-19
- [29] Yang Y., Zheng K., Wu C., Yang Y., Improving the Classification Effectiveness of Intrusion Detection by Using Improved Conditional Variational AutoEncoder and Deep Neural Network, Sensors 2019, 19, 2528
- [30] Zhang R., Peng Z., Wu L., Yao B., Guan Y., Fault Diagnosis from Raw Sensor Data Using Deep Neural Networks Considering Temporal Coherence, Sensors 2017, 17, 549.
- [31] Shokri R., Shmatikov V., Privacy-Preserving Deep Learning, In ACM Conference on Computer and Communications Security (CCS), 2015, 910
- [32] Graves A., Mohamed A. R., Hinton G., Speech recognition with deep recurrent neural networks, In ICASSP, 2013
- [33] Taigman Y., Yang M., Ranzato M., Wolf L., Deepface: Closing the gap to human-level performance in face verification., In CVPR., 2014

- [34] Krizhevsky A., Sutskever I., Hinton G. Imagenet classification with deep convolutional neural networks, In NIPS., 2012
- [35] Ciresan D., Meier U., Schmidhuber J., Multi-column Deep Neural Networks for Image Classification., 2012, arXiv:1202.2745v1 [cs.CV].
- [36] Pattanayak, S. Pro Deep Learning with Tensorflow: A Mathematical Approach to Advanced Artificial Intelligence in Python, Apress Media, 2017, 109
- [37] Lin J. C. –W., Zhang Y., Zhang B., Fournier-Viger P., Djenouri Y., Hiding Sensitive Itemsets with Multiple Objective Optimization, In Soft Computing, 2019
- [38] Cha S. –C., Hsu T. –Y., Xiang Y., Yeh K. –H., Privacy Enhancing Technologies in the Internet of Things: Perspectives and Challenges, IEEE Internet of Things Journal, 2019, 6(2), 2159-2187
- [39] Cha S. –C., Chuang M. –S., Yeh K. –H., Huang Z. –J. Su C., A User-Friendly Privacy Framework for Users to Achieve Consents With Nearby BLE Devices, IEEE Access, 2018, Vol. 6, 20779-20787
- [40] Zhou L., Yeh K. –H., Hancke G., Liu Z., Su C., Security and Privacy for the Industrial Internet of Things: An Overview of approaches to safeguarding endpoints, IEEE Signal Processing Magazine 2018, 76-87
- [41] Wu J. M. –T., Lin J. C. –W., Fournier-Viger P., Djenouri Y., Chen C. –H., Li Z., The Density-Based Clustering Method for Privacy-Preserving Data Mining, Mathematical Biosciences and Engineering. 2019; 16(3), 1718-1728
- [42] Salem M., Taheri S., Yuan J. –S., Utilizing Transfer Learning and Homomorphic Encryption in a Privacy Preserving and Secure Biometric Recognition System. Computers 2019, 8, 3.
- [43] Lin J. C. –W., Wu J. M. –T., Fournier-Viger P., Djenouri Y., Chen C. –H., Zhang Y., A Sanitization Approach to Secure Shared Data in an IoT Environment, IEEE Access, 2019, Vol. 7, 25359-25368.
- [44] Blum A., Ligett K., Roth A., A learning theory approach to non-interactive database privacy, In ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2008, 609-618
- [45] Chaudhuri K., Sarwate A. D., Sinha K., Near-optimal differentially private principal components, In Advances in Neural Information Processing Systems, 2012, 998-1006
- [46] Ji Z., Elkan C, Differential Privacy based on importance weighting, In Machine Learning, 2013, 163-183
- [47] Sala A., Zhao X., Wilson C., Zheng H., Zhao B. Y., Sharing graphs using differentially private graph models, In Internet Measurement Conference, 2011, 81-98
- [48] Vaidya J., Shafiq B., Basu A., Hong, Y, Differentially private naïve Bayes classification, In Web Intelligence, 2013, 571-576
- [49] Phan N., Wang Y., Wu X., Dou D., Differential Privacy Preservation for Deep Auto-Encoder: An Application of Human Behavior Prediction, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAA-16), 2016, 1309-1316.
- [50] Adesuyi T. A., Kim B. M., A layer-wise Perturbation based Privacy Preserving Deep Neural Network. In Proc. IEEE International Conference on Artificial Intelligence in Information and Communication, 2019, 389-394.
- [51] Wang J., Liu S., Li Y., A Review of Differential Privacy in Individual Data Release. International Journal of Distributed Sensor Networks, 2015
- [52] <http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/>, Accessed: 20th August, 2018
- [53] <https://archive.ics.uci.edu/ml/machine-learning-databases/dermatology/>, Accessed: 4th September, 2018
- [54] Tutuncu K., Koklu M., Comparison of Classification Techniques on Dermatological Dataset. International Journal of Biomedical Science & Bioinformatics, 2016, 3(1)
- [55] Patel S., Patel A., An Empirical Study of Applying Artificial Neural Network for Classification of Dermatology Disease. Indian Journal of Science and Technology, 2017, Vol 10(17), DOI:10.17485/ijst/2017/v10i17/1127 08
- [56] Mandal S., Banerjee I., Cancer Classification Using Neural Network, International Journal of Emerging Engineering Research and Technology, 2015, 3(7)
- [57] Abdel-Ilah L., Šahinbegović H., Using machine learning tool in classification of breast cancer, In: Badnjević A. (eds) CMBEBIH 2017; IFMBE Proceedings, vol 62, Springer, Singapore, 2017, 3-8
- [58] LeCun Y., Bottou L., Bengio Y., Haffner P., Gradient-based learning applied to document recognition, Proceedings of the IEEE, 1998, 86(11)
- [59] Deotte C., Digit Recognizer: Learn computer vision fundamentals with the famous MNIST data <https://www.kaggle.com/c/digit-recognizer/discussion/61480>. Accessed 14th April, 2019