Article

Multi-Party Privacy-Preserving Logistic Regression with Poor Quality Data Filtering for IoT Contributors

Kennedy Edemacu and Jong Wook Kim *

Department of Computer Science, Sangmyung University, Seoul 03016, Korea; k.edemacu@muni.ac.ug

* Correspondence: jkim@smu.ac.kr

Abstract: Nowadays, the internet of things (IoT) is used to generate data in several application domains. A logistic regression, which is a standard machine learning algorithm with a wide application range, is built on such data. Nevertheless, building a powerful and effective logistic regression model requires large amounts of data. Thus, collaboration between multiple IoT participants has often been the go-to approach. However, privacy concerns and poor data quality are two challenges that threaten the success of such a setting. Several studies have proposed different methods to address the privacy concern but to the best of our knowledge, little attention has been paid towards addressing the poor data quality problems in the multi-party logistic regression model. Thus, in this study, we propose a multi-party privacy-preserving logistic regression framework with poor quality data filtering for IoT data contributors to address both problems. Specifically, we propose a new metric *gradient similarity* in a distributed setting that we employ to filter out parameters from data contributors with poor quality data. To solve the privacy challenge, we employ homomorphic encryption. Theoretical analysis and experimental evaluations using real-world datasets demonstrate that our proposed framework is privacy-preserving and robust against poor quality data.

Citation: Edemacu, K.; Kim, J.W. Multi-Party Privacy-Preserving Logistic Regression with Poor Quality Data Filtering for IoT Contributors. *Electronics* **2021**, *10*, 2049. <https://doi.org/10.3390/electronics10172049>

Academic Editor: Cheng Siong Chin

Received: 3 August 2021

Accepted: 20 August 2021

Published: 25 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: IoT; logistic regression; homomorphic encryption; multi-party; gradient similarity; data quality

1. Introduction

The combined usage of machine learning techniques (e.g., logistic regression) with the internet of things (IoT) is expected to improve service delivery in several application domains such as industries, smart mobility, cyber-physical systems, smart cities, smart health, etc. [1]. The success of these machine learning techniques, and in particular logistic regression, depends on the availability of massive training data. In several tasks, multiple IoT parties contribute their data for the model training. In conventional multi-party logistic regressions, a server is required to store, process, and share data from geographically distributed IoT data contributors. However, the centrally stored data can easily attract attention and become an attack target. Furthermore, due to the sensitivity of the stored data (e.g., biomedical data), the server cannot be completely trusted [2]. Thus, privacy preservation is a major challenge in a multi-party setting.

Several efforts have aimed at addressing the privacy challenge in multi-party settings. Multi-party computation techniques are employed to achieve a privacy-preserving logistic regression in [3–5]. To prevent information leakage to the central server, homomorphic encryption (HE) techniques are applied during the model training process in [2,6–8]. HE allows arithmetic operations to be correctly performed on encrypted data, i.e., the result of the operation on the encrypted data is equal to the result of the same operation on the data in the plaintext form. The above property allows the central server to update and run a multi-party logistic regression model without obtaining any confidential information.

Although these studies have addressed the challenge of information leakage in the central server, none has paid attention to the aspects of data quality during model learning.

The problem of data quality contributed by IoT devices during model training in distributed machine learning has long been realized. For example, an attacker can take control of a health sensor and intentionally contribute wrongfully labeled data for model training. The poor quality data can affect the training efficiency and the effectiveness of the model. Zhao et al. [9] pointed out that this problem can be relaxed by allowing the central server to validate the intermediate parameters received from the distributed data contributors or participants before integrating them in the global model update. However, allowing the central server access to the validation dataset and the trained model exposes enough information that might compromise privacy. In [10], our proposed data quality check framework is limited to the multi-party machine learning paradigm of [11], where only weights are transferred during the global model update.

In this work, we aim to provide a solution to the problem of data quality and privacy during multi-party logistic regression model training. We follow the distributed model learning approach of [12]. In this method, the entire dataset is horizontally partitioned and shared amongst the participants. In other words, the dataset comprises local datasets of geographically distributed IoT data contributors whose data fields are all identical. We categorize an IoT participant as a noise-free participant (NFP) or a noisy participant (NDP) depending on the quality of data it contributes. Each participant trains a replica of the model using their own local dataset and uploads the intermediate gradients to the central server for updating the global model. To filter out the poor quality data (noisy data), we propose a metric *gradient similarity* ($Gsim$). A participant's intermediate gradients can only be included in the global model update if and only if its $Gsim$ is above a given threshold. We adopt HE for privacy preservation. A summary of our contributions is presented below:

1. We propose a novel metric $Gsim$ in a distributed setting used to determine the quality of the data contributed by the IoT participants;
2. We combine $Gsim$ with HE to design a multi-party privacy-preserving logistic regression model that filters out poor quality data during the model training;
3. We perform analysis and conduct experiments with real-world datasets to demonstrate the effectiveness of our designed framework.

The rest of the paper is organized as follows. In Section 2, we present the related works. Section 3 presents the preliminary concepts. We present our proposed system in Section 4. Privacy and effectiveness analysis of our proposed framework are presented in Section 5. Sections 6 and 7 present the experiments and the conclusion, respectively.

2. Related Work

Logistic regression models have long been widely applied in various fields for classification purposes. In medicine, ref. [13–15] used logistic regression to predict breast cancer. Thottakkara et al. [16] demonstrated that logistic regression is one of the best machine learning models for predicting postoperative sepsis and kidney injuries. In economics, Kovacova et al. [17], employed logistic regression to forecast bankruptcy in Slovakian companies. In engineering, Caesarendra et al. [18] combined relevance vector machine with logistic regression to assess machine degradation and predict when it is susceptible to failure. Mair et al. [19] used logistic regression to assess the contamination of underground water. In another application, logistic regression is used to discriminate between deep and shallow-induced micro-earthquakes [20]. Ref. [21] examined the performance of logistic regression models in real-time to demonstrate their effectiveness. Regarding IoT networks, ref. [22] combined IoT with logistic regression to detect and predict acute stress in patients. Devi and Neetha combined logistic regression with IoT to predict traffic congestion in smart city environments [23,24].

With the increasing demand for privacy, several studies have aimed at addressing the privacy challenges in logistic regression. Bos et al. [25] considered prediction on encrypted data with a logistic regression model. The work is based on an already trained model, and thus it does not consider the model training process. Our work differs from [25] by

focusing on training the logistic regression model using data from multiple parties in a privacy-preserving manner.

Using a secure multi-party computation technique, Slavkovic et al. [26] performed secure logistic regression on vertically and horizontally partitioned datasets. This work does not consider the data quality aspect. Our work differs from [26] by focusing only on horizontally partitioned data and it filters out poor quality data during the model training.

Han et al. [27] employed homomorphic encryption and bootstrapping to train a logistic regression model using encrypted data. They also tested their proposed scheme to predict encrypted data. The proposed scheme is computationally intensive. This work did not consider the filtering out of noisy data which is the focus of our work.

In [28], De et al. designed a version of logistic regression for text classification in privacy-preserving manner. In a case study, the authors used their model to detect on-line hate speech. The privacy of this scheme is achieved through secure multi-party computation techniques which are known to exhibit high computation and communication overheads. We achieve privacy in our scheme using HE. A differential private version of logistic regression is designed in [29]. Data secrecy is not considered in the work. Our work can complement [29] by adding the data secrecy and poor quality data filtering functionalities.

In [30], a HE scheme, HEAAN [31], is used to protect data privacy during a logistic regression model training. To improve the training efficiency, the authors proposed an ensemble method that minimizes the number of iterations during the training. The authors claimed that the model convergence is achieved with only 60% of the iterations. However, this work does not consider the data quality during model training.

Fan et al. [32] proposed a privacy-preserving logistic regression model for classifying big data in the cloud. The proposed work encrypts data using a full HE scheme [33] before its uploaded to the cloud. To perform the complex operations, the authors approximated the sigmoid activation function using Taylor's theorem. This framework focuses only on privacy-preserving predictions using a trained logistic model, however, our work focuses on the secure training of a logistic regression model.

Ref. [34] designed a high performance and secure multi-party logistic regression training framework. The framework achieves its security using a two-party computation scheme in which the parties securely exchange their data through a trusted server. The performance improvement is achieved by efficiently approximating the activation function through the partial running of a secure comparison protocol. Compared to our proposed work, this framework does not consider data quality during the model training and it is unscalable.

Du et al. [35] employed the differential privacy mechanism [36] to preserve privacy in multi-party logistic regression model training. The authors approximated the objective function using Taylor's expansion. Then, they designed a noise addition mechanism that injects noise into the coefficients of the approximated objective function. Their scheme preserves privacy but does not guarantee model effectiveness due to the delicate balance between privacy and utility in differentially private systems. Our proposed work uses an encryption approach to preserve privacy during model training.

In [37], training of a logistic regression model is carried out using encrypted data. The data are encrypted using a somewhat HE scheme [38]. The data are then passed through an approximated optimization algorithm (fixed Hessian method) during the model training. This work is different from ours because it uses the fixed Hessian method while our work relies on the stochastic gradient descent algorithm. This work trains on the encrypted data while our proposed work encrypts the intermediate parameters. Additionally, the work does not consider data quality during the model training.

Cheng et al. [39] aimed at improving the computation performance of federated logistic regression. The authors designed a framework that summarizes the computation-critical HE operations of federated logistic regression for joint storage, IO and computation

optimization on GPU. Our work can be integrated with the proposed framework to design an efficient multi-party privacy-preserving logistic regression that is robust to noisy data.

Ref. [40] employed a secret sharing with a multi-party computation technique to train a multi-party logistic regression model in a privacy-preserving manner. The proposed scheme employs the Newton–Raphson method during model optimization. Unlike [40], our work relies on the gradient descent method. Our work also filters out poor quality data during model training.

3. Preliminaries

In this section, we present the summary of the fundamental concepts such as logistic regression, homomorphic encryption, stochastic gradient descent, etc., that are employed in this work.

3.1. Logistic Regression

Consider a dataset $\mathcal{D}^{(N \times d)} = \{x^i, y^i\}_{1 \leq i \leq N}$ with N records and d dimensions for classification using a logistic regression model. Where, $x^i \in \mathbb{R}^d$ is the input, $y^i \in \{0, \dots, k-1\}$ is an outcome, k is the possible number of outcomes. The model $f(x^i, W)$ generates a probability for each outcome value as:

$$f(x^i, W) = F_{softmax}(x^i \cdot W + b) \quad (1)$$

where $W \in \mathbb{R}^{k \times d}$ is a weight vector and $b \in \mathbb{R}^k$ is the bias. $F_{softmax}(t)$ is the softmax function [41] normally given as:

$$F_{softmax}(t) = \frac{1}{\sum_0^{k-1} e^{t_k}} (e^{t_0}, \dots, e^{t_{k-1}})$$

The objective function in logistic regression is the cross-entropy error function commonly defined as:

$$\mathcal{L}(W, x^i, y^i) = \mathbb{E}_{(x^i, y^i)} [\rho(f(x^i, W), y^i)] \quad (2)$$

where, $\rho(t, y) = -\sum_0^{k-1} 1_{y=k} \log(t_k)$ and $1_{y=k}$ is an indicator function defined as:

$$1_{y=k} = \begin{cases} 1; & y = k \\ 0; & otherwise \end{cases}$$

The training phase aims to find W^* for which $\mathcal{L}(W, x^i, y^i)$ is the minimum.

3.2. Stochastic Gradient Descent (SGD)

The SGD algorithm repeatedly updates the model parameters during the training phase until the global minimum is reached. For the training of the above logistic regression model, SGD updates the weight parameters as follows:

$$W = W - \alpha \cdot \Delta G \quad (3)$$

where α is a learning rate and $\Delta G = \delta \mathcal{L}(W, x^i, y^i) / \delta W$ is the gradient with respect to W . In practice, instead of feeding each that item (x^i, y^i) at a time, the inputs are fed as a group known as a batch. Therefore, in this work, (X, Y) might be encountered instead of (x^i, y^i) .

3.3. Distributed Stochastic Gradient Descent

With distributed SGD [42], it is assumed that each participant trains the model using its own local dataset and shares its intermediate gradients with the central server to update the global model after each training round. For the local training of the model,

each participant i runs the standard SGD to update the weight locally and generate an intermediate gradient as:

$$\Delta G^i = W_{new}^i - W_{old}^i \quad (4)$$

where W_{old}^i is the original weight and W_{new}^i is the new updated local weight. The central server updates the global weights W_{global} as:

$$W_{global} = W_{global} + \sum_{i=1}^p \Delta G^i. \quad (5)$$

where p is the number of parameter contributors at each training round.

3.4. Homomorphic Encryption (HE)

HE allows algebraic operations to be correctly performed on encrypted data with the result remaining in encrypted form. In this work, we employ the additive HE scheme, the Paillier algorithm [43]. Given two messages M_1 and M_2 encrypted using the same Paillier algorithm public-key, an addition operation can be performed as follows:

$$E(M_1) + E(M_2) = E(M_1 + M_2) \quad (6)$$

where $E(.)$ denotes an encryption under the Paillier algorithm. The algorithm also supports limited multiplication operations as:

$$E(M)^r = E(rM) \quad (7)$$

where r is a random constant and M is a message.

3.5. Similarity Measurement

Different methods are often used to measure the similarity between vectors. In this work, we adopt the cosine similarity [44,45]. Consider two vectors $A = \{a_1, \dots, a_f\}$ and $B = \{b_1, \dots, b_f\}$. The cosine similarity between A and B can be measured as:

$$\text{sim}(A, B) = \sum_{i=0}^{f-1} \frac{a_i}{\|A\|} \frac{b_i}{\|B\|} \quad (8)$$

In a scenario where A and B are encrypted with the Paillier algorithm, the encrypted cosine similarity is measured as:

$$E(\text{sim}(A, B)) = \prod_{i=0}^{f-1} E\left(\frac{a_i}{\|A\|} \frac{b_i}{\|B\|}\right) \quad (9)$$

In this case, $\text{sim}(A, B)$ is a value between -1 and 1 .

3.6. Security Model

In this work, we consider a set of IoT participants who aim to train a logistic regression model on their private data through a central server. We assume the dataset is horizontally partitioned, i.e., each participant holds a sub-population of the centralized dataset. We also assume that the participants share a HE private key unknown to the central server during the setup phase.

We assume the central server and the initiator are semi-honest, i.e., they follow the protocol but are curious about the data. We also assume that the central server does not collude with any of the participants. However, we assume the rest of the participants (data contributors) are malicious but non-colluding, i.e., they can maliciously inject poor quality data during the training phase but cannot collude to poison the model. External attacks,

i.e., attacks that originate from outside of the proposed system (such as phishing attacks, traceability attacks, intrusion attacks, etc.) are not considered in this work.

4. Our Multi-Party Privacy-Preserving Logistic Regression with Poor Quality Data Filtering

In this section, we present our proposed framework, but first, we discuss the *Gsim* metric.

4.1. Gradient Similarity (*Gsim*)

Here, we present the *Gsim* metric. From the definition in Equation (2), the objective function $\mathcal{L}(W, x^i, y^i)$ depends on the weight W , the data point x^i and the label y^i . Remember, in SGD, the weight W^t is updated as $W^{(t-1)} - \alpha \cdot \Delta G$ according to Equation (3). Additionally, remember that $\Delta G = \delta \mathcal{L}(W^t, x^i, y^i) / \delta W^t$. Therefore, in a multi-party logistic regression in which each participant locally and independently updates W^t , the initial weights $W^{(t-1)}$ for all the participants are the same. If the data batches (X, Y) for the participants are similar, they generate similar ΔG .

For example, consider two data contributors (participants) p_1 and p_2 in an IoT network with their datasets (x_1^i, y_1^i) and (x_2^i, y_2^i) , respectively. Assume p_1 and p_2 train a common logistic regression model and update their local weights according to the standard SGD as shown in Equation (3). It is important to note that the gradients generated during the local updates depend on the objective function $\mathcal{L}(W, x^i, y^i)$ which is itself dependent on the x^i data points. p_1 generates the gradient $\Delta G_1 = \delta \mathcal{L}(W^t, x_1^i, y_1^i) / \delta W^t$ and p_2 generates the gradient $\Delta G_2 = \delta \mathcal{L}(W^t, x_2^i, y_2^i) / \delta W^t$.

Therefore, if (x_1^i, y_1^i) and (x_2^i, y_2^i) are similar, the two participants p_1 and p_2 are likely to generate similar gradients, i.e., $\Delta G_1 \approx \Delta G_2$. The *Gsim* metric employs the similarity computation technique discussed in Section 3.5 to compute similarity between the gradients in a privacy-preserving manner.

4.2. System Overview

In this work, we follow the distributed model learning approach of [12] in which the dataset $\mathcal{D}^{N \times d}$ comes from p data contributors (participants), i.e.,

$$\mathcal{D}^{N \times d} = \{(x_1^i, y_1^i), \dots, (x_p^i, y_p^i)\}.$$

The goal is to securely learn a more powerful and effective model than the one obtained when training using a single participant's local dataset. A high-level architecture of our system is presented in Figure 1. The system comprises a central server, multiple participants and an initiator. The participants are categorized into NFPs and NDPs depending on the quality of data they contribute. The initiator is simply an NFP who initiates the *Gsim* computation. The selection of the initiator can be based on, for example, a known reputation for contributing quality data. The gradients from the rest of the participants will be compared with the initiator's gradients. Note that the selection of the initiator is critical to the success of our system. A single homomorphic private key is shared by the initiator and the rest of the participants except the central server.

In general, the initiator and the rest of the participants independently perform the model training using their private local datasets. The training is carried out synchronously. The central server aggregates the gradients uploaded by the initiator and the participants. However, participants' gradients can only be included in the aggregation if and only if their associated *Gsim*s are above a set threshold value. A *Gsim* is collaboratively and securely computed by the initiator, the central server, and each participant. The learning processes of the model initiator and a participant are shown in Figures 1a,b. The only difference between the learning processes is the roles played by the initiator and the participant during *Gsim* computation. Details of the processes are shown in Section 4.3.

Figure 1. An architecture of our proposed multi-party privacy-preserving logistic regression with poor quality data filtering. (a) The learning process of the initiator. (b) The learning process of a participant i .

To achieve the above goal, we make the following system assumptions:

- All the system entities are available during the model learning;
- The data exchanged between the entities reach their respective destinations;
- There is a mechanism for the participants and the initiator to securely obtain the shared private key;
- There exists a participant with quality data that serves as the initiator.

4.3. Model Learning

The initiator, the central server and the participants jointly learn the model as follows.

4.3.1. Initiator

The initiator, just like any other participant, independently trains a replica of the model using its private local dataset. It runs the pseudo-code shown in Algorithm 1. At each training round, the initiator first downloads an encrypted global weight $E(W_{global})$ from the central server. It decrypts $E(W_{global})$ using the shared private key to generate W_{global} . It then labels W_{global} as a replica W_{old}^j . Next, it updates W_{old}^j by running the standard SGD on its private local dataset according to Equation (3). The local update generates a new weight W_{new}^j . The initiator then computes an intermediate gradient ΔG^j according to Equation (4). After computing the gradient ΔG^j , the initiator performs two tasks:

1. First, it initiates *Gsim* computation which is conducted as follows: The initiator generates $\frac{g_i^j}{\|\Delta G^j\|}$, where $g_i^j \in \Delta G^j$. It then encrypts $\frac{g_i^j}{\|\Delta G^j\|}$ as $E(\frac{g_i^j}{\|\Delta G^j\|})$ and uploads it to the central server.
2. Second, the initiator encrypts ΔG^j as $E(\Delta G^j)$ and also uploads it to the central server.

Algorithm 1: Initiator's pseudo-code

```

1: begin
2:   Download the global model  $E(W_{global})$  from the central server
3:   Decrypt  $E(W_{global})$  as  $W_{global}$ 
4:   Label  $W_{global}$  as  $W_{old}^j$ 
5:   Update  $W_{old}^j$  to  $W_{new}^j$  using the SGD
6:   Compute the gradient  $\Delta G^j$ 
7:   Generate  $g_i^j / \|\Delta G^j\|$  for computing Gsim
8:   Encrypt  $g_i^j / \|\Delta G^j\|$  as  $E(g_i^j / \|\Delta G^j\|)$ 
9:   Encrypt  $\Delta G^j$  as  $E(\Delta G^j)$ 
10:  Upload  $E(\Delta G^j)$  and  $E(g_i^j / \|\Delta G^j\|)$  to the central server
11:  Repeat the above steps until the model converges
12: end
  
```

Note that the two tasks can be performed concurrently in parallel. The process is repeated until convergence is reached.

4.3.2. Central Server

The central server runs the pseudo-code shown in Algorithm 2. It stores and updates the global weight $E(W_{global})$, and makes it available for the initiator and the participants to download. It does so without stealing any private data from the IoT data contributors. To update $E(W_{global})$, the central server performs the following tasks:

1. It collaborates with the initiator and the rest of the participants to compute the *Gsim*. It does so by first blinding $E(\frac{g_i^j}{\|\Delta G^j\|})$ received from the initiator according to Equation (7) as $E(r \frac{g_i^j}{\|\Delta G^j\|})$, where $r \neq 0$ is randomly chosen. It then sends the blinded value to all the participants;
2. It then receives a blinded gradient similarity from each participant, i.e., it receives $Gsim_i \cdot r$ from each participant i , where $Gsim_i$ is the gradient similarity associated with the participant i . $Gsim_i$ can easily be extracted by the central server through division of $Gsim_i \cdot r$ by r ;
3. It also receives $E(\Delta G^j)$ from the initiator and $E(\Delta G^i)$ from each of the other participants. It then updates $E(W_{global})$ using a modified version of Equation (5) as:

$$E(W_{global}) = E(W_{global}) + E(\Delta G^j) + \sum_{i=1}^{\tau} E(\Delta G^i) \quad (10)$$

where τ is the number of participants whose respective $Gsim_i$ is greater than a threshold value T .

This is repeated until convergence is reached. The encryption of the parameters prevents the central server from obtaining any confidential information about the private local data of the IoT data contributors.

Algorithm 2: Central Server's pseudo-code

```

1: begin
2:   Store the global model  $E(W_{global})$ 
3:   Send  $E(W_{global})$  to all the participants and the initiator
4:   Receive  $E(g_i^j / \|\Delta G^j\|)$  and  $E(\Delta G^j)$  from the initiator
5:   Choose  $r \neq 0$  randomly
6:   Blind  $E(g_i^j / \|\Delta G^j\|)$  as  $E(r \cdot g_i^j / \|\Delta G^j\|)$ 
7:   Send  $E(r \cdot g_i^j / \|\Delta G^j\|)$  to all the participants
8:   Receive  $Gsim_{i.r}$  and  $E(\Delta G^i)$  from each participant
9:   Update  $E(W_{global})$  according to Equation (10)
10:  Repeat steps 3-9 until the model converges
11: end

```

4.3.3. Participant

Each participant executes the pseudo-code shown in Algorithm 3. Like the initiator, each participant independently trains the global weights using its own private local dataset. At the start of each training round, each participant downloads the global weight $E(W_{global})$ from the central server and decrypts it as W_{global} . Each participant labels W_{global} as a replica W_{old}^i . Each participant then updates W_{old}^i by running the standard SGD on its own private local dataset according to Equation (3) to generate an updated weight W_{new}^i . Next, each participant generates its intermediate gradient ΔG^i according to Equation (4). Each participant then performs three tasks:

1. Each participant generates $\frac{g_i^i}{\|\Delta G^i\|}$, where $g_i^i \in \Delta G^i$ (i.e., $g_i^i \in \{g_1^i, \dots, g_f^i\}$).
2. Each participant receives $E(r \frac{g_i^j}{\|\Delta G^j\|})$ from the server and computes a blinded gradient similarity according to a modified version of Equation (9) as:

$$E(Gsim_{i.r}) = \prod_{i=0}^{f-1} E(r \frac{g_i^i}{\|\Delta G^i\|} \frac{g_i^j}{\|\Delta G^j\|}) \quad (11)$$

Each participant then decrypts $E(Gsim_{i.r})$ to $Gsim_{i.r}$ and uploads the result to the central server.

3. At the same time, each participant encrypts ΔG^i as $E(\Delta G^i)$ and uploads it to the central server.

Tasks 1, 2 and 3 can be executed concurrently in parallel. This is repeated until convergence is reached.

Algorithm 3: Participant's pseudo-code

```

1: begin
2:   Download the global model  $E(W_{global})$  from the central server
3:   Decrypt  $E(W_{global})$  as  $W_{global}$ 
4:   Label  $W_{global}$  as  $W_{old}^i$ 
5:   Update  $W_{old}^i$  to  $W_{new}^i$  using the SGD
6:   Compute the gradient  $\Delta G^i$ 
7:   Generate  $g_i^i / \|\Delta G^i\|$ 
8:   Receive  $E(r \cdot g_i^i / \|\Delta G^i\|)$  from the central server
9:   Compute  $E(Gsim_i.r)$  according to Equation (11)
10:  Decrypt  $E(Gsim_i.r)$  as  $Gsim_i.r$ 
11:  Encrypt  $\Delta G^i$  as  $E(\Delta G^i)$ 
12:  Upload  $E(\Delta G^i)$  and  $Gsim_i.r$  to the central server
13:  Repeat the above steps until the model converges
14: end

```

5. Analysis

Here, we analyze our proposed framework in terms of privacy and effectiveness. We conclude the section by presenting the limitations of our proposed framework.

5.1. Privacy

We analyze the privacy of our proposed scheme under the security model presented in Section 3.6. Under the threat of a semi-honest but non-colluding central server, no information on the private local data of any participant or the initiator leaks to the central server. The server receives encrypted parameters such as $E(\Delta G^i)$ and $E(\Delta G^j)$. Additionally, the aggregation of the received intermediate gradients and the global weight updates are performed in encrypted form. Therefore, other than the identification of potential NFPs and NDPs through the gradient similarity score, no private data information is leaked to the central server.

Under a semi-honest initiator, no information about the private local data of the other participants is leaked to the initiator. The initiator only has access to the updated global weights which cannot reveal information about the gradients of the other participants. Under the threat of a malicious participant, the participant cannot obtain information about the private local data of any other participant or the initiator from the downloaded global weights. Additionally, the participant cannot re-engineer the gradient of the initiator from the received similarity computation component from the central server. This is because, it receives $E(r \frac{g_i^j}{\|\Delta G^j\|})$ instead of $E(\Delta G^j)$, which is even made harder to re-engineer by the blinding factor r .

In a case of collusion between the other participants and the initiator, our system can resist collusion of up to $p - 2$, where p is the total number of IoT data contributors in the system including the initiator.

5.2. Effectiveness

We analyze the effectiveness of our proposed system by discussing the characteristics of logistic regression model training. The training process of logistic regression is to learn weights that minimize the loss function. The process involves a continuous update of a randomized weight until a fine-tuned weight is obtained. The update direction is normally determined by the gradient. However, the gradient is influenced by the data points as it depends on the loss function which itself depends on the data points. Therefore, given similar datasets, a model update using the datasets will most likely be heading in the same direction. In the presence of noise, this direction might be reversed. Thus,

affecting the training convergence and the effectiveness of the model. By measuring gradient similarities, we are able to identify datasets that are likely to reverse the update direction and eliminate their associated parameters during global weight updates. Hence, maintaining the effectiveness of the model and improving the training efficiency.

5.3. Limitations

We derive the limitations of the proposed framework from the assumptions made during its design. First, we limit our design to the semi-honest and non-colluding security assumption. Although our system can adequately preserve participants' data privacy in the event of collusion, there is a possibility the model effectiveness might be affected. For example, the central server is responsible for setting the threshold value. If the central server colludes with a participant, the participant can carefully calibrate the amount of fake data to prevent dropping below the threshold value. Thus, allowing parameters from poor quality data to be considered during the global model update. Second, the model learning process in our proposed framework is performed synchronously, i.e., all the entities have to carry out the model training at the same time. In practical settings with heterogeneous devices, there are always differences in computation power and latency. This makes the synchronous process unrealistic. Finally, our design only considers the horizontal partitioning and not the vertical partitioning of the dataset. This limits its application in environments where participants hold different data features.

6. Experiments

In this section, we experimentally evaluate our proposed scheme using real-world datasets.

6.1. Experimental Setup

We performed our experiments using a desktop computer having Intel-core i5-6500 CPU with a speed of 3.20 GHz, a GeForce GT 710 GPU and a RAM of 16 GB running Ubuntu 20.04 operating system.

We simulated the IoT participants, i.e., an initiator, and the NFPs and NDPs. Specifically, we simulated one (1) initiator, four (4) NFPs and one (1) NDP in our first experiment, and one (1) initiator, three (3) NFPs and one (1) NDP in our second experiment. The local datasets of the initiator and the NFPs are noise-free while a fraction of NDP's dataset is filled with noisy data. All the IoT data contributors (initiator, NFPs and NDP) share the same homomorphic private key.

We designed two baselines (*stand-alone* and *noise-unfiltered*) for bench-marking. In the stand-alone, the model is trained using only a local dataset. There are no collaborations involved during the training process and hence, there is no need to deploy a central server and perform homomorphic operations. In the noise-unfiltered baseline, the model is trained collaboratively by multiple participants. Thus, a central server is employed and homomorphic operations are performed. However, there is no poor quality data filtering process involved, i.e., the gradients from all the IoT participants (NFPs and NDPs) are integrated during the global model update.

We performed all the implementations using Theano 1.0.5 and Python 3.8.5. We adapted the logistic regression code in [46] for the local model training of each participant and the initiator. The remaining steps of the participant and initiator procedures were implemented with Python. We also implemented the server procedure in Python. For the homomorphic operations, we employed the Paillier library in [47] and set the key-length to be 1024-bits. In each training round, the Paillier library was used to encrypt the intermediate parameters.

We experimented using three real-world datasets: MNIST [48], notMNIST [49] and CIFAR-10 [50] datasets. During the experiment, each participant and the initiator executed the standard SGD algorithm for their respective local training with a learning rate of 0.13 and a batch size of 100 for the first experiment, and a learning rate of $2e(-5)$ and batch size of 64 for the second experiment. The initiator and each participant uploaded their

encrypted intermediate gradients after 5 local epochs. The gradient similarity thresholds were set at 0.15 and 0.10 for the first and second experiments, respectively.

6.2. Experiment I (Experiment with MNIST and notMNIST Datasets)

First, we performed experiments using the MNIST as the primary dataset and notMNIST as the noise dataset. MNIST is a dataset of 28×28 gray-scale handwritten images of digits 0–9. The dataset consists of 70,000 images of which 60,000 form the training set and 10,000 form the test set. notMNIST is a dataset of 28×28 gray-scale images of letters A–J. The dataset comprises 519,000 images of which 500,000 form the training set while 19,000 form the test set. In our experiment, the initiator and each participant were allocated 10,000 training samples of the MNIST dataset as their local datasets. However, for the NDP some fraction of the allocated dataset is replaced by samples of the notMNIST dataset which acts as the noise dataset.

6.3. Experiment I Results

First, we present the results of the gradient similarity, i.e., we demonstrate how the participants' gradients are similar to the initiator's gradients as the training proceeds. As shown in Figure 2a,b, the gradient similarities for the NFPs are all above 0.22. Meanwhile, for the NDP, the gradient similarity is mostly below 0.2. However, the gradient similarity even drops below -0.2 when the percentage of noise data is increased to 50%, as shown in Figure 2b. This confirms our hypothesis that participants with less similar datasets generate less similar intermediate gradients.

Figure 2. Variation of gradient similarity against the number of training rounds. (a) NDP noise data are 30%. (b) NDP noise data are 50%.

We also demonstrated the training convergence of our scheme in comparison to the baseline schemes as shown in Figure 3a,b. It can be observed that our scheme converges faster and achieves the best accuracy as compared to the baseline schemes. This is because the cumulative training dataset is bigger and the training process is hardly affected by poor quality data. The stand-alone converges fairly fast but is not as accurate as our scheme. This is mainly due to the inadequate amount of data used for the training. The noise-unfiltered baseline achieves the worst performance. It takes forever to converge, and it is less accurate. The performance deteriorates even more with more noise, as shown in Figure 3b when the amount of noise data in NDP is increased to 50%. This is due to the noise data affecting the training process and model effectiveness and thus, the more the noise, the more the effect.

Figure 3. Training convergence against the number of training rounds. (a) NDP noise data are 30%. (b) NDP noise data are 50%.

6.4. Experiment II (Experiment with the CIFAR-10 and notMNIST Datasets)

In this experiment, we evaluate our scheme using the CIFAR-10 dataset as our primary dataset and notMNIST as the noise dataset. The CIFAR-10 dataset comprises 60,000 $32 \times 32 \times 3$ images of 10 classes of objects. The training set comprises 50,000 images and the test set comprises 10,000 images. Here, the initiator and each NFP was allocated 10,000 CIFAR-10 images. The remainder was allocated to the simulated NDP. However, portions of the NDP's images were replaced with samples from notMNIST dataset which acts as the noise dataset. The samples were selected randomly and padded to match the dimensions of the CIFAR-10 dataset. All the data were normalized.

6.5. Experiment II Results

First, we present the gradient similarity variation during the training. Here, we consider two cases. In the first case, 30% of the NDP's dataset is replaced with the noisy dataset and in the second case, 50% of the NDP's dataset is replaced with the noisy dataset. The results are shown in Figure 4. It can be observed that, in both cases, the gradient similarities between the initiator's gradients and the NFPs' gradients are above 0. Meanwhile, the gradient similarities between the initiator's gradients and the NDP's gradients can be negative. However, more noise makes it more negative in the second case, as shown in Figure 4b.

Figure 4. Variation of gradient similarity against the number of training rounds. (a) NDP noise data are 30%. (b) NDP noise data are 50%.

We also present the training convergence of our scheme in comparison with the baseline schemes. Similarly, we consider two cases with varying amounts of noise data in the NDP's dataset. The results are shown in Figure 5. As expected, we can see that in both cases our proposed scheme performs better than the two baselines. There is a slight delay in the convergence of our scheme when the NDP's data noise is increased to 50%. This is because, with more noise, there is a slight chance that the set threshold might miss filtering out some noise. The stand-alone converges faster, but it is less effective compared

to our scheme. This is mainly due to the inadequate amount of data used for its training. The noise-unfiltered baseline performs the worst in both cases. This is because the noise might cause the gradients to diverge and hence the poor performance.

Figure 5. Training convergence against the number of training rounds. (a) NDP noise data are 30%. (b) NDP noise data are 50%.

Table 1 shows the effectiveness of our proposed scheme through the validation accuracy comparison with other schemes. It is worth noting that our proposed scheme performs adequately. For the MNIST dataset, it achieves an accuracy of 0.949 which is statistically significant with over 95% confidence level (i.e., p -value < 0.05). While for the CIFAR-10 dataset, it achieves an accuracy of 0.797 with over 95% confidence level (i.e., p -value < 0.05). Our scheme is only bettered by the [27] which is trained on noise-free encrypted data. The reason for the drop in accuracy could be due to some parameters associated with the primary data being dropped during the noise filtering. The noise-unfiltered performs the worst as the noise diverges the gradients.

Table 1. Validation Accuracy Comparison.

Schemes	Datasets	
	MNIST	CIFAR-10
Our Scheme (Privacy Preserving)	94.9	79.7
[21] (No Privacy Preservation)	92.1	N/A
Stand-alone	90.1	71.8
Noise-unfiltered	73.4	45.9
[27] (Data Encrypted)	96.4	N/A

6.6. Gradient Similarity Computation Overhead

We present the computation overhead caused by computing the gradient similarity score. The computation overheads for the entities are presented in Table 2. It can be seen that the initiator experiences the most overhead due to the encryption of all the $g_i^j / \parallel \Delta G^j \parallel$ components. The central server simply blinds the encrypted $E(g_i^j / \parallel \Delta G^j \parallel)$ and hence experience the least overhead. The overhead experienced by each participant is due to the multiplication of the encrypted elements using the constants and the decryption of the final blinded $Gsim_i$.

Table 2. Computation Overhead for Gradient Similarity.

Dataset	Entities		
	Initiator(s)	Central Server(s)	Participant(s)
MNIST	3.01	0.11	2.13
CIFAR-10	3.07	0.14	2.16

The experimental results confirm that noise can negatively affect the effectiveness of logistic regression models and thus filtering out noise is a necessity in multi-party settings.

7. Conclusions

In this work, we demonstrated how poor quality data affect the effectiveness of a logistic regression model. We proposed and integrated gradient similarity with homomorphic encryption to design a multi-party logistic regression learning framework in which no private information is leaked during model training while filtering out poor quality data from IoT data contributors. We demonstrated the effectiveness of our proposed framework by experimenting using real-world datasets. The results show that our framework is adequately effective while being robust to noisy data. Therefore, the framework is beneficial in multi-party logistic regression learning scenarios in which privacy and data quality are critical.

In the future, more experiments using different datasets can be conducted. Extending the framework for other machine learning algorithms can be explored. Furthermore, modifying the scheme for asynchronous training offers an interesting possibility that can be investigated. Additionally, improving the security of the framework by integrating differential privacy can be considered.

Author Contributions: Conceptualization, K.E.; methodology, K.E.; software, J.W.K.; validation, K.E. and J.W.K.; formal analysis, K.E.; investigation, K.E. and J.W.K.; resources, J.W.K.; data curation, K.E. and J.W.K.; writing—original draft preparation, K.E.; writing—review and editing, J.W.K.; visualization, K.E.; supervision, J.W.K.; project administration, J.W.K.; funding acquisition, J.W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by a 2021 research Grant from Sangmyung University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Savazzi, S.; Nicoli, M.; Rampa, V. Federated learning with cooperating devices: A consensus approach for massive IoT networks. *IEEE Internet Things J.* **2020**, *7*, 4641–4654. [\[CrossRef\]](#)
2. Kim, A.; Song, Y.; Kim, M.; Lee, K.; Cheon, J.H. Logistic regression model training based on the approximate homomorphic encryption. *BMC Med. Genom.* **2018**, *11*, 83. [\[CrossRef\]](#)
3. Mohassel, P.; Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 19–38.
4. El Emam, K.; Samet, S.; Arbuckle, L.; Tamblyn, R.; Earle, C.; Kantarcioglu, M. A secure distributed logistic regression protocol for the detection of rare adverse drug events. *J. Am. Med. Inform. Assoc.* **2012**, *20*, 453–461. [\[CrossRef\]](#)
5. Nardi, Y.; Fienberg, S.E.; Hall, R.J. Achieving both valid and secure logistic regression analysis on aggregated data from different private sources. *J. Priv. Confid.* **2012**, *4*, [\[CrossRef\]](#)
6. Aono, Y.; Hayashi, T.; Phong, L.T.; Wang, L. Privacy-preserving logistic regression with distributed data sources via homomorphic encryption. *IEICE Trans. Inf. Syst.* **2016**, *99*, 2079–2089. [\[CrossRef\]](#)
7. Wu, S.; Teruya, T.; Kawamoto, J.; Sakuma, J.; Kikuchi, H. Privacy-preservation for stochastic gradient descent application to secure logistic regression. In Proceedings of the 27th Annual Conference of the Japanese Society for Artificial Intelligence, Toyama, Japan, 4–7 June 2013; Volume 27, pp. 1–4.
8. Xie, W.; Wang, Y.; Boker, S.M.; Brown, D.E. Privlogit: Efficient privacy-preserving logistic regression by tailoring numerical optimizers. *arXiv* **2016**, arXiv:1611.01170.
9. Zhao, L.; Wang, Q.; Zou, Q.; Zhang, Y.; Chen, Y. Privacy-preserving collaborative deep learning with unreliable participants. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1486–1500. [\[CrossRef\]](#)
10. Edemacu, K.; Jang, B.; Kim, J.W. Reliability Check via Weight Similarity in Privacy-Preserving Multi-Party Machine Learning. *arXiv* **2021**, arXiv:2101.05504.
11. Phuong, T.T.; Phong, L.T. Privacy-preserving deep learning via weight transmission. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 3003–3015. [\[CrossRef\]](#)
12. Shi, H.; Jiang, C.; Dai, W.; Jiang, X.; Tang, Y.; Ohno-Machado, L.; Wang, S. Secure multi-party computation grid LOGistic REGression (SMAC-GLORE). *BMC Med. Inform. Decis. Mak.* **2016**, *16*, 89. [\[CrossRef\]](#)
13. Didarloo, A.; Nabilou, B.; Khalkhali, H.R. Psychosocial predictors of breast self-examination behavior among female students: An application of the health belief model using logistic regression. *BMC Public Health* **2017**, *17*, 861. [\[CrossRef\]](#)

14. Liu, L. Research on logistic regression algorithm of breast cancer diagnose data by machine learning. In Proceedings of the 2018 International Conference on Robots & Intelligent System (ICRIS), Changsha, China, 26–27 May 2018; pp. 157–160.
15. Sultana, J.; Jilani, A.K. Predicting breast cancer using logistic regression and multi-class classifiers. *Int. J. Eng. Technol.* **2018**, *7*, 22–26. [CrossRef]
16. Thottakkara, P.; Ozrazgat-Baslanti, T.; Hupf, B.B.; Rashidi, P.; Pardalos, P.; Momcilovic, P.; Bihorac, A. Application of machine learning techniques to high-dimensional clinical data to forecast postoperative complications. *PLoS ONE* **2016**, *11*, e0155705. [CrossRef]
17. Kovacova, M.; Klietnik, T. Logit and Probit application for the prediction of bankruptcy in Slovak companies. *Equilibrium. Q. J. Econ. Econ. Policy* **2017**, *12*, 775–791. [CrossRef]
18. Caesarendra, W.; Widodo, A.; Yang, B.S. Application of relevance vector machine and logistic regression for machine degradation assessment. *Mech. Syst. Signal Process.* **2010**, *24*, 1161–1171. [CrossRef]
19. Mair, A.; El-Kadi, A.I. Logistic regression modeling to assess groundwater vulnerability to contamination in Hawaii, USA. *J. Contam. Hydrol.* **2013**, *153*, 1–23. [CrossRef] [PubMed]
20. Mousavi, S.M.; Horton, S.P.; Langston, C.A.; Samei, B. Seismic features and automatic discrimination of deep and shallow induced-microearthquakes using neural network and logistic regression. *Geophys. J. Int.* **2016**, *207*, 29–46. [CrossRef]
21. Palvanov, A.; Im Cho, Y. Comparisons of deep learning algorithms for MNIST in real-time environment. *Int. J. Fuzzy Log. Intell. Syst.* **2018**, *18*, 126–134. [CrossRef]
22. Pandey, P.S. Machine Learning and IoT for prediction and detection of stress. In Proceedings of the 2017 17th International Conference on Computational Science and Its Applications (ICCSA), Trieste, Italy, 3–6 July 2017; pp. 1–5; [CrossRef]
23. Devi, S.; Neetha, T. Machine Learning based traffic congestion prediction in a IoT based Smart City. *Int. Res. J. Eng. Technol.* **2017**, *4*, 3442–3445.
24. Muthuramalingam, S.; Bharathi, A.; Gayathri, N.; Sathiyaraj, R.; Balamurugan, B. IoT based intelligent transportation system (IoT-ITS) for global perspective: A case study. In *Internet of Things and Big Data Analytics for Smart Generation*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 279–300.
25. Bos, J.W.; Lauter, K.; Naehrig, M. Private predictive analysis on encrypted medical data. *J. Biomed. Inform.* **2014**, *50*, 234–243. [CrossRef]
26. Slavkovic, A.B.; Nardi, Y.; Tibbits, M.M. “Secure” Logistic Regression of Horizontally and Vertically Partitioned Distributed Databases. In Proceedings of the Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), Omaha, NE, USA, 28–31 October 2007; pp. 723–728.
27. Han, K.; Hong, S.; Cheon, J.H.; Park, D. Logistic regression on homomorphic encrypted data at scale. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 9466–9471.
28. De Cock, M.; Dowsley, R.; Nascimento, A.C.; Reich, D.; Todoki, A. Privacy-preserving classification of personal text messages with secure multi-party computation: An application to hate-speech detection. *arXiv* **2019**, arXiv:1906.02325.
29. Zhang, J.; Zhang, Z.; Xiao, X.; Yang, Y.; Winslett, M. Functional Mechanism: Regression Analysis under Differential Privacy. *Proc. VLDB Endow.* **2012**, *5*, 1364–1375. [CrossRef]
30. Cheon, J.H.; Kim, D.; Kim, Y.; Song, Y. Ensemble method for privacy-preserving logistic regression based on homomorphic encryption. *IEEE Access* **2018**, *6*, 46938–46948. [CrossRef]
31. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 409–437.
32. Fan, Y.; Bai, J.; Lei, X.; Zhang, Y.; Zhang, B.; Li, K.C.; Tan, G. Privacy preserving based logistic regression on big data. *J. Netw. Comput. Appl.* **2020**, *171*, 102769. [CrossRef]
33. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
34. De Cock, M.; Dowsley, R.; Nascimento, A.C.; Railsback, D.; Shen, J.; Todoki, A. High performance logistic regression for privacy-preserving genome analysis. *BMC Med. Genom.* **2021**, *14*, 1–18. [CrossRef]
35. Du, W.; Li, A.; Li, Q. Privacy-preserving multiparty learning for logistic regression. In Proceedings of the International Conference on Security and Privacy in Communication Systems, Singapore, 8–10 August 2018; pp. 549–568.
36. Dwork, C. Differential privacy: A survey of results. In Proceedings of the International Conference on Theory and Applications of Models of Computation, Changsha, China, 18–20 October 2008; pp. 1–19.
37. Bonte, C.; Vercauteren, F. Privacy-preserving logistic regression training. *BMC Med. Genom.* **2018**, *11*, 13–21. [CrossRef]
38. Fan, J.; Vercauteren, F. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* **2012**, *2012*, 144.
39. Cheng, X.; Lu, W.; Huang, X.; Hu, S.; Chen, K. HAFLO: GPU-Based Acceleration for Federated Logistic Regression. *arXiv* **2021**, arXiv:2107.13797.
40. Ghavamipour, A.R.; Turkmen, F.; Jian, X. Privacy-preserving Logistic Regression with Secret Sharing. *arXiv* **2021**, arXiv:2105.06869.
41. kmdanielduan. Logistic-Regression-on-MNIST-with-NumPy-from-Scratch. 2019. Available online: <https://github.com/kmdanielduan/Logistic-Regression-on-MNIST-with-NumPy-from-Scratch> (accessed on 3 August 2020).
42. Gong, M.; Feng, J.; Xie, Y. Privacy-enhanced multi-party deep learning. *Neural Netw.* **2020**, *121*, 484–496. [CrossRef]

-
43. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
 44. Gomez-Barrero, M.; Maiorana, E.; Galbally, J.; Campisi, P.; Fierrez, J. Multi-biometric template protection based on homomorphic encryption. *Pattern Recognit.* **2017**, *67*, 149–163. [[CrossRef](#)]
 45. Nautsch, A.; Isadskiy, S.; Kolberg, J.; Gomez-Barrero, M.; Busch, C. Homomorphic encryption for speaker recognition: Protection of biometric templates and vendor model parameters. *arXiv* **2018**, arXiv:1803.03559.
 46. Deep Learning Tutorials. Available online: <http://deeplearning.net/tutorial/> (accessed on 3 August 2020).
 47. CSIRO's Data61. Python Paillier Library. 2013 Available online: <https://github.com/data61/python-paillier> (accessed on 23 May 2020).
 48. LeCun, Y.; Cortes, C.; Burges, C. MNIST Handwritten Digit Database. 2010. Available online: <http://yann.lecun.com/exdb/mnist> (accessed on 3 August 2020).
 49. Bulatov, Y. Notmnist Dataset. 2011. Available online: <http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html> (accessed on 6 August 2020).
 50. Krizhevsky, A.; Nair, V.; Hinton, G. The Cifar-10 Dataset. 2014. Available online: <http://www.cs.toronto.edu/kriz/cifar.html> (accessed on 7 August 2020).