*International Journal of Advanced Smart Convergence Vol.12 No.2 56-66 (2023)*
*http://dx.doi.org/10.7236/IJASC.2023.12.2.56*

**IJASC 23-2-6**

# Privacy-Preserving Deep Learning using Collaborative Learning of Neural Network Model

Hye-Kyeong Ko

*Assistant Professor, Department of Computer Engineering, Sungkyul University, Korea*
*ellefgt@sungkyul.ac.kr*

***Abstract***

*The goal of deep learning is to extract complex features from multidimensional data use the features to create models that connect input and output. Deep learning is a process of learning nonlinear features and functions from complex data, and the user data that is employed to train deep learning models has become the focus of privacy concerns. Companies that collect user's sensitive personal information, such as users' images and voices, own this data for indefinite period of times. Users cannot delete their personal information, and they cannot limit the purposes for which the data is used. The study has designed a deep learning method that employs privacy protection technology that uses distributed collaborative learning so that multiple participants can use neural network models collaboratively without sharing the input datasets. To prevent direct leaks of personal information, participants are not shown the training datasets during the model training process, unlike traditional deep learning so that the personal information in the data can be protected. The study used a method that can selectively share subsets via an optimization algorithm that is based on modified distributed stochastic gradient descent, and the result showed that it was possible to learn with improved learning accuracy while protecting personal information.*

***Keywords:*** *Privacy-Preserving, Deep Learning, Neural Network Model, Collaborative Stochastic Gradient Descent*

## 1. Introduction

Recent advances in artificial intelligence-based deep learning techniques have produced breakthroughs in long-running AI research such as image and text recognition and language translation [1]. Large corporations such as Google, Facebook, and Apple have used large quantities of training data that were collected from their users. Large numbers of GPUs were used to perform calculations in order to efficiently utilize large-scale deep learning [2][3]. Studies on deep learning have produced good learning results, but the training data used for learning has posed serious personal information issues. As the numbers of centrally collected personal texts, images, and videos have increased, so have the threats to personal information. For example, users cannot

Manuscript Received: March. 22, 2023 / Revised: March. 24, 2023 / Accepted: March. 27, 2023
Corresponding Author: ellefgt@sungkyul.ac.kr
Tel: +82-31-467-8113
Assistant Professor, Department of Computer Engineering, Sungkyul University, Korea

delete the data that is collected by corporations or control how it is used, and they do not have a say in what can be learned through the data.

In addition, images and voice recordings capture sensitive information such as faces and license plates, as well as other people's voices. The monopoly of large internet corporations on big data that is collected from large numbers of users has led to a monopoly on AI models trained by this data, and while users can receive help in the form of virtual assistants and machine translations for foreign language web pages, most of the models that have been created via shared data are owned by the corporations that made them [1]. In many regions, there are laws or regulations that forbid the sharing of people's data, and a result, clinical researchers are only able to perform deep learning using datasets that belong to their own organizations. It is known that the use of neural network models becomes increasingly helpful for training models as the datasets for training become larger and more diverse. However, because researchers cannot use data from other organizations when training their models, their learning models are inevitably inferior as a result [4][5].

This study has designed a system for collaborative deep learning which can protect the privacy of personal information. The proposed system presents a method which can train a neural network model with various people's input in order to obtain benefits from other participants without sharing the inputted data. The proposed method selectively shares parameters during training so that participants can benefit from other participants' models without sharing the training input. This paper is organized as follows. Chapter 2 examines existing studies on the gradient descent method as applied to machine learning and deep learning. Chapter 3 shows how the proposed method works using a distributed stochastic gradient descent method. Chapter 4 analyzes the proposed technique through experiments. Finally, Chapter 5 presents this paper's conclusion which states that the proposed method protects the personal information in participants' training data without reducing the model's accuracy.

## 2. Related Studies

### 2.1 Deep Learning and Personal information

Deep learning is a process of learning nonlinear features from complex data, and it is superior to previous techniques for speech recognition and image recognition. Deep learning shows promise in analyzing genetic data, and from a privacy perspective, the training data that is used to create deep learning models contains sensitive personal information; therefore, special attention to personal information protection is needed when performing training for deep learning [1]. Previous papers on personal information protection in machine learning have mainly focused on other traditional machine learning algorithms rather than deep learning, and they have discussed the personal information in the data used to train models or used as input for existing models, the personal information within the models, and the personal information in the models' output [1][2]. Techniques that are based on secure multiparty computation (SMC) can help to protect the intermediate stages of computation when multiple parties perform collaborative machine learning using only their own input[6]. In general, SMC techniques incur a considerable performance overhead, and the use of these techniques in deep learning that protects personal information remains an unsolved problem. The techniques that these models use to protect personal information include personal information protecting stochastic inference [7], personal information protecting speaker identification, and encrypted data computation [8]. The goal of this paper is to collaboratively train neural networks that each participant can use individually.

Deep learning is a process of learning nonlinear features and functions from complex data. Deep learning

has proven to be superior to past techniques for image recognition, speech recognition, and face detection [1]. Deep learning shows promise in analyzing biomedical data related to cancer and genetics [1][9]. From a privacy perspective, the training data that is used to create such models contains sensitive content, highlighting the need for deep learning methods which protect personal information. Recently, there have been studies on parallel deep learning which have presented techniques for parallelizing the stochastic gradient descent method in GPU/CPU clusters [9] and performing distributed computation during neural network training [10]. However, these techniques used methods of controlling training that do not give consideration to the personal information in the training data.

### 2.2 Stochastic Gradient Descent

Deep learning aims to extract complex features from multidimensional data and use them to create models that connect input and output. Deep learning architectures are simultaneously composed of multi-layer networks, and abstract features are calculated via nonlinear functions. This paper's training input focuses on supervised learning.

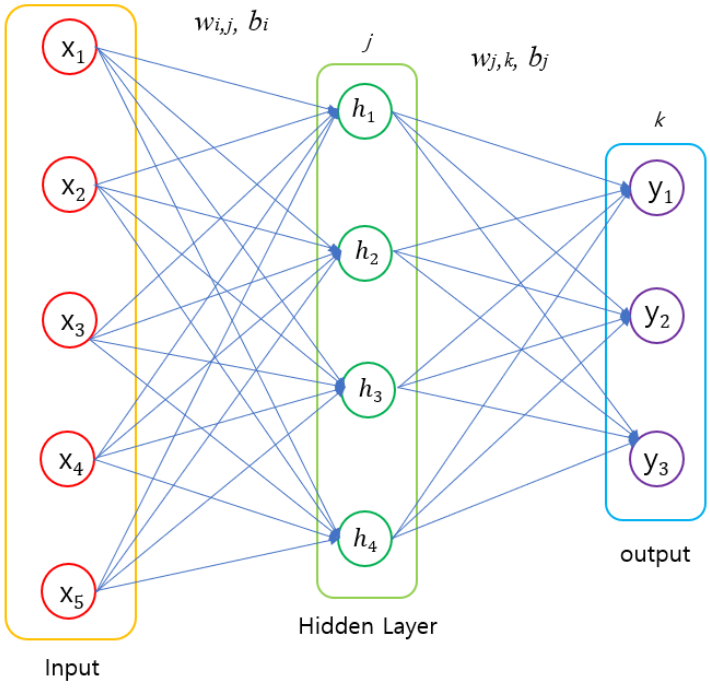


**Figure 1. Neural Network with Hidden Layers**

Multilayer networks are the most common form used in deep learning, and Figure 1 shows a neural network with hidden layers. Each node in the figure models a single neural network, and the red circle nodes represent bias nodes. The matrix $W_k$ includes the weight values that are used to calculate the activation function in each layer. In a typical multilayer network, each neuron receives not only the output of the neuron in the previous layer but also the signal that is sent from the sending neuron. Also, a nonlinear activation function is applied to all inputs and calculated [9][11]. Each neural is responsible for maximally deflecting signals from special neural that step down by 1 as well as the output of nerves below the previous one. The output vector of the *k*

layer neuron is $b_k = f(W_k b_k - 1)$. Here, $f$ is the activation function, and $W_k$ is the weight value matrix that determines the contribution of each input signal. In general, it is made up of unprocessed features that were extracted directly from higher layers, and the output of previous layers corresponds to the abstract response created by the model. The nonlinear function f and the weight value matrix determine the features extracted in each layer, and data regarding the parameter (weight value matrix) values that maximize the classification accuracy of the neural network is automatically learned [9]. Learning the neural network's parameters is a nonlinear optimization problem, and in supervised learning, the objective function is the neural network's output.

The algorithm that is generally used to solve such problems is the modified gradient descent method. Gradient descent starts with a set of neural network parameters and calculates the slope of the optimized nonlinear function at each step and updates the parameters to minimize the slope. This process continues until the algorithm converges on the optimum. Stochastic gradient descent is a method that calculates the gradient for one item of extracted data and applies the gradient descent algorithm. Rather than using all of the data, it uses some of the data that has been randomly extracted. The amplitude of the results during the intermediate stages of the learning process is large and unstable, and the algorithm's speed is very fast. Also, because it processes the data one at a time, its error rate is high, and it cannot utilize all of the GPU's performance [11]. The methods that have been introduced to compensate for these disadvantages use mini batches, which is more efficient than using entire batches and reduces noise in the stochastic gradient descent method [9]. The slope of the parameters can be averaged for all available data, and the algorithm is inefficient for learning large datasets. For stochastic gradient descent, in the most simple case when the slope of an extremely small subset of the overall dataset is calculated, the data sample that corresponds to the maximum probability is selected randomly at each optimization stage.

For example, assuming that there are one million training data, it would be necessary to perform one million times one million or one trillion operations to calculate the batch gradient descent method. When stochastic gradient descent is used, only one million operations are required even when machine learning is performed using all one million data. When stochastic gradient descent is used, an approximation of batch gradient descent is calculated; however, because the weight updating time is fast, the cost function's convergence value is found more quickly in practice. This formula was used to design the system for personal information protection and stochastic gradient descent that is discussed in this paper.

## 3. Distributed Stochastic Gradient

The learning of neural network parameters is a nonlinear optimization problem. In supervised learning, the objective function is the output of the neural network, and the algorithms that are used to solve this problem are generally modified forms of gradient descent. Gradient descent starts with a set of neural network parameters and calculates the gradient of the optimized nonlinear function at each step and updates the parameters to minimize the slope. These steps are repeated until the algorithm converges on the optimum solution. The gradient of the parameters can be averaged for all available data, and the algorithm known as gradient descent is inefficient for learning large datasets. Stochastic gradient descent is a method that calculates the gradient of a small subset of the overall dataset, and the method proposed in this paper is a selective collaborative deep learning method. The method that is assumed in this paper is independently performed when

updating other parameters while performing gradient descent, and the distributed stochastic gradient descent method updates a small number of parameters during each learning iteration.

### 3.1 Distributed Parameter Updating

In selective SGD, the learner first selects the portion of the parameters that will be updated in each iteration. During selection, the parameters with larger slopes are selected from the current values.

$$W_j = W_j - k\frac{\partial L_i}{\partial W_j} \tag{1}$$

Equation (1) shows the stochastic gradient descent method. Here, $W$ represents the weight value parameters to be updated, and $\frac{\partial L_i}{\partial W_j}$ represents the slope of the result value for the parameters. $k$ is the learning rate, and in practice, it is set in advance to something like *0.01*. For each minibatch $i$, the loss function $W_j$ is calculated for all parameters $W_j$ as in SGD. The goal of selective SGD is for two or more participants to learn simultaneously while being independent. Figure 2 shows the working principles of the selective SGD method that is proposed in this paper. In Figure 2, the parameter server downloads the latest values of most-updated local parameter and uploads gradient of selected parameters and add them to local parameters.

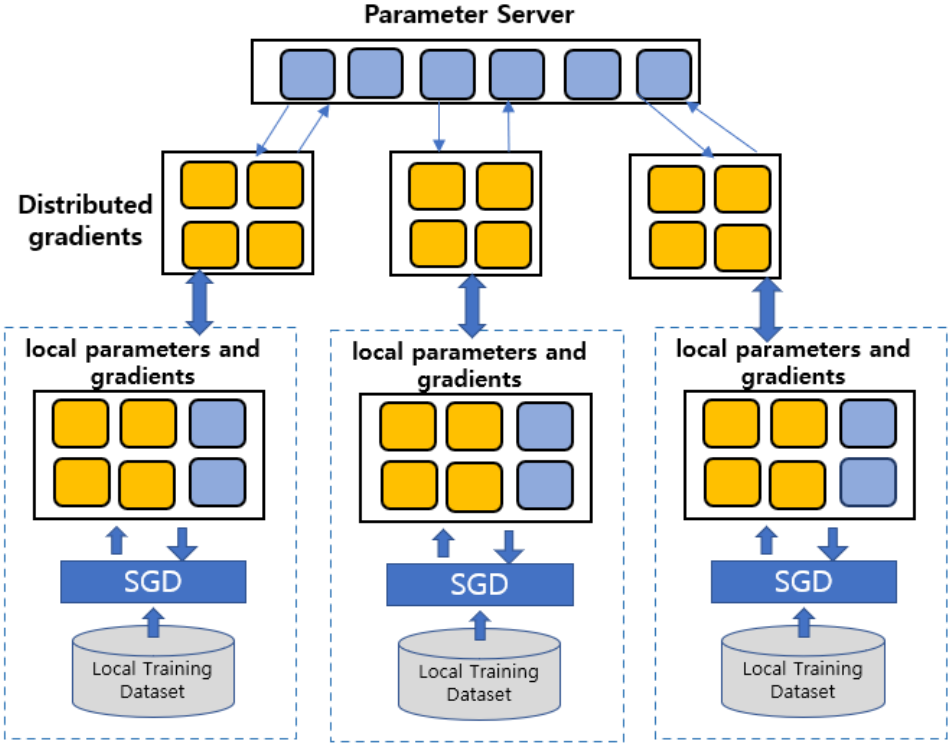


**Figure 2. Operation of Distributed Stochastic Gradient**

We assume that each participant $i$ maintains a neural network parameter $W$. The parameter server maintains a separate parameter vector $W$, and each participant can initialize its local parameters by downloading the latest

values from the parameter server. In local training, participants share the slopes that are calculated for several parameters, and each participant controls which slopes are shared and how free they are. The sum of all the slopes that are calculated for the given parameters determines the scale of descent for local optimization of the parameter.

### 3.2 Training Method

The participants do not actually see the data, but they benefit from the other participants' training data, and they can create more accurate models through the limited training data that is solely their own. Participants can directly share the slopes, and the update process can be hidden, or secure calculations can be made through trusted central servers. The server adds the slopes to the parameter values, and each of the participants downloads a subset of the parameters from the server and uses it to update their own models. The update frequency and the moving average of the parameters plus the slope are used as criteria for downloading a given parameter. For example, assume that there are *N* participants who have local personal datasets that can be used for training. Assume that there is a parameter server that is available to all and is responsible for maintaining the latest values of the parameters, and the parameters are implemented on an actual server or designed using a distributed system. Table 1 shows the types of parameters used in the experiments.

**Table 1. Experimental Parameters**

| | |
|---|---|
| k | Learning rate of stochastic gradient descent |
| $\theta_{d,}\ \theta_u$ | Selected for download and upload Fraction of Parameters |
| v | Boundaries for gradient values shared with other participants |
| z | Threshold for gradient selection |

Each participant initializes the parameters, creates a dataset, and begins training. The algorithm includes an exchange protocol so that the participants can use the parameter server to upload the slopes of the selected neural network parameters and download the latest SGD parameter values. Algorithm shows pseudocode for a participant's distributed SGD algorithm.

**Algorithm. Distributed SGD algorithm for participants**

Select the initial parameters *W* and the learning rate *k*.
Repeat until similar minimum values are obtained.

1. Download the $|\theta_d * W|$ parameters from the server relocate the corresponding load parameters.
2. Perform SGD on the local dataset and update the local parameters *W* corresponding to (1).
3. Calculate the gradient of the vector *W* that has been changed in all the local parameters for SGD.
4. Upload *I* to the server, *I* being the index of the gradient of $|\theta_u * W|$ that was selected according to the following conditions: the largest values: sort gradients in W and upload $\theta_d$ fraction of them, starting from the biggest and randomly subsample the gradients whose value is above threshold *z*

The participant downloads the $\theta_d$ part of the parameters from the server and overwrites the local parameters with the downloaded values. Participants can independently arrive at parameters and avoid overloading

parameters in a single participant's training dataset, and each participant can independently and individually evaluate new data without interacting with other participants. The parameters can control the collaborative learning process, unlike the actual neural network parameters that are being learned, and each participant can maintain the neural network parameters *W* while the parameter server maintains a separate parameter vector *W*. Distributed SGD generally does not make assumptions regarding which parameters need to be updated by other participants or the update speed. Some participants can perform more updates because they have more computing power and processing capacity.

- The method for selecting and sharing the slopes considers two criteria.

(1) In the gradient descent algorithm, large values are found, and $\theta_u$ values are selected accurately.

(2) A random subset of values that are larger than the threshold value is selected. Fewer slopes are shared because the number of larger slopes may be less than the parameter's $\theta_u$ part.

Before the selected slope *W* is uploaded, the values are curtailed so that they are within the range [-v, v], and random noise is added to prevent the values from revealing too much information about the training data. The participants update *W* with the bound (*W*, *v*), and random noise is added before uploading. A minibatch is a randomly selected training dataset with a size *M*. Distributed SGD does not make assumptions regarding which parameters need to be updated by other participants or the update speed. Some participants may not be able to upload the distributed parameters due to network errors, but distributed SGD is known to be efficient at asynchronous parameter updating. The parameter server initializes the parameter vector W and handles upload and download requests from participants. When someone uploads a gradient, the server adds it to a global parameter that is responsive to the uploaded value of *W*, updates the metadata, and adds that update to each parameter *W*. To increase the weight of the recently updated parameter, the server periodically multiplies the weight. This parameter can get the latest value from the server when downloading. The participant decides which part of the parameters to download by setting $\theta_d$.

## 4. Experiments and Evaluation

In this study, the proposed method was evaluated using the SVHN dataset [11] which is a deep learning dataset that has been used by many related studies and contains house numbers obtained from Google Streetview images. In the SVHN dataset, the images have a size of 32 x 32 and contain RGB color data for each pixel that has been converted to YUV. The images are clustered around the house numbers, and many of the images include obstructions. The dataset consisted of a training set with 100,000 items and a test set with 10,000 items. The training set that was used in the experiments was normalized by subtracting the average and dividing by the standard deviation of the data sample. In the neural network for the SVHN dataset, the size of the input layer was 3,072. The goal of the training was to classify the input as one of 10 possible numbers; therefore, the size of the output layer was 10.

In the experiment evaluations, a convolutional neural network (CNN) was used. CNNs [11] are widely used for image and video recognition, and this study performed evaluations using CNN values that were outputted by the Torch7 nn package [12] . The figures show the functions that were used in each layer and the connections between layers. Table 2 shows the number of parameters, and Table 3 shows the number of artificial neural network CNN parameters.

**Table 2. Size of datasets**

| | SVHN |
|---|---|
| train | 100,000 |
| test | 10,000 |

**Table 3. Number of neural network parameters**

| | SVHN |
|---|---|
| CNN | 298,732 |

In the experiments, two scenarios were analyzed. The first scenario used a centralized SGD method on the entire dataset. The training data was gathered into a single dataset, and training was performed with this dataset using standard stochastic gradient descent, which exposed the personal information in the dataset. The second scenario was able to limit personal information exposure by employing distributed SGD and using certain criteria to select which slopes would be uploaded.

The evaluation was performed on the learning rate setting ($k$ = 0.01 and 0.001) with SGD minibatch sizes of 1 and 32. 1,000 SVHN data samples were used, and they had values in the portion of parameters $\theta_u$={1, 0,1, 0.01, 0.001} that was selected for sharing in selective SGD. During downloading, the $\theta_d$ parameter was set to *1*. Table 4 shows the maximum accuracy that was achieved by distribute SGD for different mini-batch sizes in the CNN architecture.

**Table 4. Proposed SGD in the CNN architecture.**

| | SGD | 0.1 | 0.01 | 0.001 | Independence SGD |
|---|---|---|---|---|---|
| SVHN.CNN | 0.8819 | 0.8274 | 0.7548 | 0.6228 | 0.5211 |

The minibatch size was *1*, and accuracy was compared with independent SGD. To demonstrate the efficiency of this paper's proposed method in comparison to traditional gradient descent, this study evaluated the accuracy of proposed SGD and SGD when training a convolutional neural network with the SVHN dataset. The proposed SGD can achieve higher accuracy than SGD because updating a small fraction of the parameters prevents the neural network weights from jointly memorizing the training data. By sharing a small portion of the gradient during the gradient descent stage, nearly the same accuracy as SGD was achieved, and selective parameter sharing had almost no effect on the overall SGD. During the training process, setting the minibatch size to *1* achieved high probability, and convergence occurred very quickly. Distributed SGD demonstrates personal information exchanges, and distributed SGD made better use of the training data in the CNN data. It is affected by each participant having *1,000* data samples in the case of the SVHN dataset.

Figure 3 show the comparison of different values of meta-parameters (mini-batch size and fraction of shared gradients). In general, participants can choose a lesser value for the parameter by training with a calibration data set. For example, public data sets that do not relate to personal information. By sharing a small fraction of the gradient at each gradient descent step, we were able to achieve accuracy similar to that of SGD.

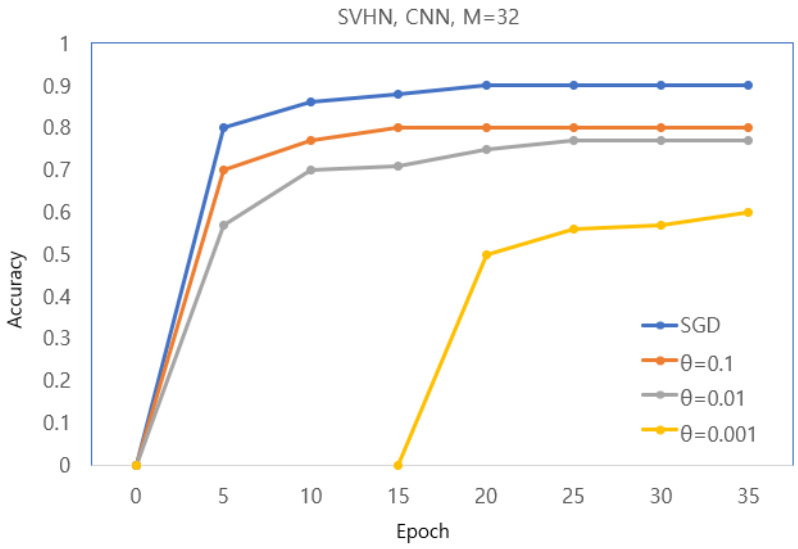


**Figure 3. Comparison of distributed SGD for different mini-batch sizes**

When the mini-batch size is set to 32, the convergence is slower but smoother because we average the gradients over many training data points during gradient descent. To prevent direct exposure of personal information, the participants are not shown the training dataset, unlike traditional deep learning, and the personal information in the data can be protected. The size and function of the data are kept secret, and the training data can be deleted immediately. When the proposed learning model is used, all participants perform learning without revealing the input or output data and without any communication with other participants. As such, there is no personal information exposure as compared to traditional deep learning methods.

## 5. Conclusions

In traditional deep learning, all training data is generally exposed to the company that is performing the training, and the individuals who contribute the data have no right to control it. There is a risk that the individual's sensitive data will be leaked to organizations that can access the data by attackers who have attacked the company's data storage. In addition, the owners of the data cannot directly use the models that are trained via traditional deep learning training methods, and when these models are used, they are exposed to personal information risks via input by the companies that own the models. This paper's proposed system aims to resolve the threats to personal information that are posed by deep learning. The proposed privacy protecting deep learning system can resolve these problems, protect the personal information in the training data, and guarantee people's rights regarding the purpose of the training. This paper has proposed a selective training method based on the selective stochastic gradient descent method. The proposed method is effective for CNN neural networks, and it protects the personal information in the participants' training data without reducing the accuracy of the resulting model. As such, effective personal information protection is provided when deep learning is performed on data that cannot be shared by the owners due to confidentiality issues.

This paper's proposed method was used to evaluate the SVHN dataset, which is used as a standard for image classification algorithms. The accuracy of the models that were created by distributed participants was similar

to the case in which a single entity had control of the entire dataset and used it to train a centralized model, resulting in a violation of privacy. In the case of the SVHN dataset, 99.12% accuracy was achieved when the participants shared 10% of the parameters, and 98.52% accuracy was achieved when they shared 1% of the parameters. In comparison, the maximum accuracy of the centralized, privacy-violating model was 99.19%. Because the proposed method does not directly reveal all of the training data, system leaks are an indirect portion of a portion of the neural network parameters. The proposed method is effective for CNN neural networks, and it protects the personal information in the participants' training data without reducing the accuracy of the resulting model. Therefore, it provides effective personal information protection when deep learning is performed on data that cannot be shared by the owners due to confidentiality issues. The proposed method is effective for CNN neural networks, and it protects the personal information in the participants' training data without reducing the accuracy of the resulting model. Therefore, it provides effective personal information protection when deep learning is performed on data that cannot be shared by the owners due to confidentiality issues. In future studies, it will be necessary to examine how to use differentiated personal information in the parameter updater in order to minimize indirect leaks. In addition, we plan to compare the proposed method with the federated learning method in additional experiments.

## Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT).(NO.NRF-2021R1A2C1012827) in (2023).

## References

[1] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al, "Deep speech: Scaling up End-to-End Speech Recognition," arXiv:1412.5557, 2014.
DOI: https://doi.org/10.48550/asXiv.1412.5567

[2] Seokwon, Lee, "Development of a Method for ACF Bonding Based on Machine Vision," The Journal of the Convergence of Culture Technology (JCCT), Vol. 4, No. 3, pp. 209-212, 2018.
DOI: https://doi.org/10.17703/JCCT.2018.4.3.209

[3] H. Lee and J. Choi, "Implementation of Smart Ventilation Control System using IoT and Machine Learning," The Journal of the Institute of Internet, Broadcasting and Communication (JIIBC), Vol. 20, No. 2, 2020.
DOI: https://doi.org/10.7236/JIIBC.2020.20.2.283

[4] D. Shultz, "When your Voice Betrays You," Science, Vol. 347, No. 6221, 2015.
DOI: https://doi.org/10.1126/science.347.6221.494

[5] Yo-Seop, Lee, "Analysis on Machine Learning-as-a-Service," International Journal of Advanced Culture Technology (IJACT), Vol. 6, No. 4, pp. 303-308, 2018. \
DOI: https://doi.org/10.17703/IJACT2018.6.4.303

[6] J. Bos, K. Lauter, and M. Naehrig, "Private Predictive Analysis on Encrypted Medical Data," Informatics, Vol. 50, pp. 234-243, 2014.
DOI: https://doi.org/10.1016/j.jbi.2014.04.003

[7] M. Pathak, S. Rane, W. Sun, and B. Raj, "Privacy Preserving Probabilistic Inference with Hidden Markov Models," In Proc. of 2011 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1-3, 2011.
DOI: https://doi.org/10.1109/ICASSP.2011.5947696

[8] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural Networks over Encrypted Data," arXiv:1412.6181, 2014.
DOI: https://doi.org/10.48550/arXiv.1412.6181

[9] M. Liang, Z. Li, T. Chen, and J. Zeng, "Integrative Data Analysis of Multi-Platform Cancer Data with a Multimodal Deep Learning Approach," In Proc. of IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 12, No. 4, pp. 928-937, 2015.
DOI: https://doi.org/10.1109/TCBB.2014.2377729.

[10] M. Pathak, S. Rane, W. Sun, and B. Raj, "Multiparty Different Privacy via Aggregation of Locally Trained Classifiers," In Proc. of International Conference on Neural Information Processing Systems, pp. 1876-1884, 2010.

[11] Y, LeCun, L, Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," IEEE, Vol. 86, No. 11, pp. 2278-2324, 1998.
DOI: https://doi.org/10.1109/5.726791.

[12] https://github.com/torch/nn.