Deakin SIT Research Project: Deep Leakage from Gradients

Submitted as Research Report in SIT723

SUBMISSION DATE: 25/09/2022

T2-2022

Aliraza Khowaja

STUDENT ID 219554473

COURSE - Master of Information Technology Professional (S464)

Supervised by: Dr. Adnan Ahmed

Abstract

Background

A popular technique in contemporary multi-node machine learning systems is gradient exchange (e.g., distributed training, collaborative learning). Gradients were once thought to be safe to interchange, meaning that doing so would prevent the training data from being compromised. This study demonstrates that it is possible to get the private training data from the publicly available gradients, nevertheless. Deep Leakage from Gradient is the name given to this leakage. The effectiveness of leakage on computer vision models has been empirically validated.

Objectives

This study exhibits four defence tactics, Densenet 121, Inception net V3, Lenet, and Resnet 18, to stop the deep leakage.

Results

The study has observed that ResNet-18 outperforms LeNet, while DenseNet-121 is susceptible to leakage. Further, IncepionNet v3 fails to produce any image. The requirement of more variables during the process of optimization makes algorithms requiring larger model size less susceptible to leakages. However, larger model size has its limitation in the way of increasing the computational costs. DenseNet 121 is vulnerable to this Leakage. The magnitude of Mean Squared Error of DenseNet 121 is comparable to that of ResNet-18 but higher than the MSE of LeNet. However, it is not possible to recreate the image from the leakage using InceptionNet v3.

Conclusion

This study shows the process used to recover the input given only the gradients of the loss function with respect to the model's parameters. Sharing gradients among different trainers is a method of collaborative model training while keeping the data secured and separated from each trainer.

Key Words: Deep Learning, Machine Learning, leakage, ReNet-18, LeNet, DenseNet-121, InceptionNet v3

Contents

List of Figures

List of Tables

1 Introduction

To accelerate training on massive data sets, distributed training becomes important. In a distributed learning system, each worker does the computation in parallel, which is synchronized by exchanging gradients (both parameter server [1] and all-reduce [2]). Data is split naturally as a result of the distribution of computation, with each client having its own training data and only exchanging gradient information during training. It enables the use of data from several sources without the need to centralize them when training a model. When the training set contains sensitive data, this strategy—known as collaborative learning—is frequently used [3]. As an illustration, several hospitals collaborate to train a model without transferring patient medical information [4].

Large-scale machine learning tasks frequently involve distributed training and collaborative learning. However, is the privacy of each participant's training data sets protected by the "gradient sharing" scheme? People typically assume that gradients are secure to share and won't reveal the training data. Recent research has demonstrated that gradients can reveal certain characteristics of the training data, such as the presence of samples with particular characteristics in the batch [5] and the ability to create images using generative adversarial networks that resemble the training images [5]. Here, we take into account a trickier scenario: are we totally capable of stealing the training data from gradients? Can we acquire the training data in reverse given a machine learning model F() and its weights W and the gradients w with respect to a pair of input and labels? Contrary to conventional opinion, we demonstrate that this is in fact feasible.

This study illustrates Deep Leakage from Gradients (DLG), which is the idea that sharing gradients might expose sensitive training information. In just a few rounds, four optimization techniques are shown that can acquire both the training inputs and the labels. The operation is carried out by first randomly generating two "dummy" inputs and labels, after which the standard forward and backward are carried out. Instead of maximizing model weights as in conventional training, the dummy inputs and labels are optimized to reduce the gap between the dummy gradients and actual gradients once the dummy gradients have been derived from the dummy data. The dummy data closely resemble the original ones because to matching gradients. The private training data (inputs and labels) will be completely disclosed once the optimization is complete.

Conventional shallow leakages can only produce comparable simulated images and require additional label information. The results produced by DLG are the exact original training samples rather than artificial look-alike substitutes because deep leakage is an optimization process and does not depend on any generative models. As a result, DLG does not require any additional prior information about the training set and can instead infer the label from shared gradients. The performance of our system has been tested on tasks involving vision (classification of images) (masked language model). DLG efficiently recovers all of the training data on a variety of data sets and tasks in a few gradient steps. People have to be made aware of the need to reconsider the safety of gradients in light of the recent discovery of such a significant leakage from gradients. The multi-mode machine learning system is severely challenged by deep leakage. According to this study, the basic gradient sharing system does not always effectively safeguard the confidentiality of training data. The parameter server, which typically does not keep any training data, has the ability to steal the local training data from every participant in centralized distributed training. It gets worse with distributed training that is decentralized since each participant can take the private training data of their neighbors.

1.1 Aims Objectives:

In distributed learning systems like Collaborative Learning and Federated Learning, etc., it is generally accepted that exchanging gradients won't leak private training data. Recently, Zhu et al. [17] published a method that demonstrates the potential to retrieve private training data from gradients that are shared publicly. They combine the dummy data and appropriate labels using their Deep Leakage from Gradient (DLG) approach under the guidance of shared gradients.

The aim of this study lies in demonstrating how Deep Leakage from Gradients can be stopped. Sharing the gradients may expose sensitive training information. This paper proposes four optimization techniques that can, in a limited number of iterations, obtain both the training inputs and the labels. A pair of "dummy" inputs and labels is produced at random before the attack is carried out as usual forward and backward. Instead of maximising model weights as in conventional training, the dummy inputs and labels are optimised to reduce the gap between the dummy gradients and actual gradients once the dummy gradients have been derived from the dummy data. The dummy data closely resembles the original ones because to matching gradients. The private training data

(inputs and labels) will be completely disclosed once the optimization is complete. This study exhibits four defence tactics to stop the deep leakage: Densenet 121, Inception net V3, Lenet, and Resnet 18.

The objectives of this study include:

- Demonstration of the possibility of obtaining private training data from the publicly shared gradients. Deep learning is being used in several areas including healthcare. Leakage of training data raises several privacy and regulatory concerns.

- DLG can disclose pixel-wise accurate images using just the gradients. While traditional strategies typically only create partial qualities or synthetic substitutes and require more knowledge to address.

- Analyzing the assault challenges in various contexts and discussing various attack protection techniques.

- Creation of a framework to prevent deep learning leakage.

- Comparing different algoritms to identify the best algorithm to prevent leakage.

1.2 Research Questions (RQs)

Research questions are crucial because they influence not only how the protocol is developed and how the study is designed, but also how sample sizes are calculated and how powerful the study will be. The research questions in this study are given below:

1. To assess the possibility of obtaining private training data from publicly shared gradients.

2. To assess the requirements to reconstruct accurate images pixel-wise.

3. To examine the assault challenges in various contexts and talk about possible attack prevention measures to prevent potential data loss.

4. To create a framework to prevent deep learning leakage.

5. To compare different algoritms to identify the best algorithm to prevent leakage.

1.3 Structure:

The remaining sections of this study have been structured as under:
Chapter 2. Literature Review: This chapter deals with a review of the extant literature on the subject. The aim of the literature review is to understand the methods used in earlier studies and find the gaps in the studies.

Chapter 3. Methodology: This chapter describes the methodology used in this study.

Chapter 4. Artefact Development Approach: This chapter discusses the architecture, design, and function of the algorithms that have been used in this study.

Chapter 5. Empirical Evaluation: This chapter evaluates the algorithms on the basis of the research questions.

Chapter 6. Results and Discussion: This chapter discusses the results obtained in this study.

Chapter 7. Threats to validity: This chapter discusses the limitations of this study.

Chapter 8. Conclusion and Future Work: This chapter concludes the study and provides scope for future studies.

2 Literature Review

Distributed Training

Deep neural networks, for example, require a lot of computer power to train. Many studies focused on distributed training to speed up the training process in order to complete it in a fair amount of time. Numerous efforts have been done to increase the scalability of distributed training, both at the algorithm and framework levels [6]. Due to its consistent performance when scaling up, synchronous SGD is adopted as the backbone by the majority of them. In general, there are two types of distributed training: with and without a parameter server. In both methods, each node updates its local weights first before sending gradients to neighboring nodes. In the centralized mode, each node receives the gradients after they have been aggregated. Gradients are traded between nearby nodes in the decentralized mode. The training data is sensitive to privacy in many application contexts. For instance, hospitals are not permitted to share information about a patient's health. Collaborative learning, which allows two or more users to jointly train a model while the training data set never leaves each participant's local server, has lately been popular [7] as a means of preventing sensitive information from being stolen. The only network-wide common data are the gradients. This method has been applied to develop predictive keyboards to enhance typing experience [8], train models for medical treatments across several hospitals [3], and analyze patient survival scenarios from different nations [4].

Shallow Leakage from Gradients

Previous research has looked into ways to infer training data information from gradients. The gradients already reveal a significant amount of information for some layers. For instance, in language tasks, the embedding layer only generates gradients for words that occur in training data, revealing the phrases that other participants have trained on [9]. However, such a leak is "shallow" since the words are out of sequence and make it difficult to determine the meaning of the original statement. Fully connected layers

are a different example where it is possible to deduce output feature values by observing gradient updates. The size of the features is far larger than the size of the weights, hence this cannot apply to convolutional layers. Some recent research creates learning-based techniques to infer batch attributes. They demonstrate that a binary classifier trained on gradients can tell whether a specific data record (membership inference [10]) or a record with specific characteristics is part of the batch of the other participant. Additionally, they train GAN models [11] to synthesize images that resemble training data from the gradient [9], but the attack is constrained and only functions when all members of the class resemble one another (e.g., face recognition).

Different picture categorization techniques have been presented for many diverse demands due to the rapid progress of artificial intelligence [12]. Their precision and adequate performance standards, however, still require improvement. Examples of architectures with better performance in the classification of medical pictures are AlexNet, VGG, GooLeNet, and ResNet. However, these networks struggle with vanishing gradients, over fitting, and convergence issues [13]. Therefore, CNN provides precise picture classification choices to enable features based on DenseNet [14]. The Gao et al. [12] study's inability to be applied to binary class classification is one of its limitations. In addition, Multipath-DenseNet can provide predictions with fewer parameters while the other baselines require a larger number of parameters [13]. DenseNet exceeded accuracy in Hasan et al's study on COVID-19 patient prediction using CT image categorization based on DenseNet CNN [15]. Recall is 95 %, while total accuracy is 92%. The average calculation time for the DenseNet-121 model is 195.35 seconds. The study had some flaws. The first was that a few hundred photographs might not be enough to make a meaningful prediction. The absence of Grad Cam representations of the used DenseNet-121 model, which may have improved readability, is the study's second flaw. Applying additional types of architecture, such as DenseNet-161, DenseNet-169, and DenseNet-201, can further support DenseNet for research and development.

Federated Learning:

Federated learning [16], a machine learning framework that trains machine learning models cooperatively among several clients without submitting their data to a central server, has lately gained popularity. In particular, the clients download the global model and compute the local gradients or model update using the private data. The global model parameters are then updated when the local gradients or model modifications are aggregated and submitted to the server. Only the gradient data or model updates from

the clients will be sent to the central server throughout this process. Because of this, it is believed that client data is secure, and federated learning has been identified as a promising new privacy-protection strategy.

Gradient Leakage Attacks

The privacy of the user may still not be completely protected via federated learning. A malicious attacker is capable of rebuilding clients' local data by taking advantage of the shared gradients or model changes, according to recently published gradient leaking attack methods [17]. For instance, the work [17] looks for input data samples with gradients that are the closest to the real gradients shared among clients by Euclidean distance. Since the L-BFGS [18] optimization solver used by the technique [17] demands that the network be smooth, their attack cannot be effective in deep models or non-smooth models. The attacking approach [19] maximizes the cosine-similarity between the gradients of the created samples and ground truth gradients to solve a similar optimization problem to reconstruct clients' data.

By utilizing the matching of the batch normalization statistics between generated samples and the ground truth and emphasizing the group consistency among various candidates, another work [20] enhances earlier attacks on high-resolution images. Using a Bayesian framework, a more recent work [21] aggregates known attacks [17] to overcome protections based on Gaussian noise perturbation on the gradient and provides Bayes optimal countermeasures for various defenses. Another line of research takes into account the possibility that an attacker would be interested in using the label information in federated learning clients' data.

As an illustration, the technique [22] suggests that label information be extracted based on the relationship between various rows of the gradient matrix of the final linear layer. However, only a batch of a single image can satisfy the relationship. The argument in the work [20] can be used to a batch of many photos, but it depends on the assumption that the batch does not contain repeated labels.

In a different study, Guan et al [23] used cytological pictures to train the Inception-v3 deep convolutional neural network (DCNN) model to distinguish cervical lymphadenopathy. The Inception-v3 DCNN model demonstrated excellent potential for

easing the diagnosis of cervical lymphadenopathy using cytological images after training with a large dataset. NHL was the most difficult cytology type for DCNN to differentiate, according to the analysis of the misdiagnosed cases.

When Yang et al. [24] compared eleven neural network architectures, including DenseNet 121, InceptionNet v3, and ResNet 18, for small data sets of lung images from COVID-19 patients in order to improve clinical decisions, they found that a neural network model with more layers did not always perform better overall. The choice of neural networks with residual connectivity (such as ResNet) and automatic search functionality (such as EfficientNet) typically yields superior results. It should be mentioned that employing different hyperparameters has an impact on the performance of neural network models. However, in general, neural networks with residual connections (ResNet) and capabilities for automatic search (EfficientNet) perform better during migration.

Valeria Maeda-Gutierrez and others [25] compared CNN architectures for classifying diseases in tomato plants. The authors' primary focus was on contrasting the abilities of AlexNet, GoogleNet, Inception V3, ResNet 18, and ResNet 50 using various performance indicators. When these types of models are compared, CNNs stand out since they don't need any time-consuming preprocessing, have a faster convergence rate, and perform well during training. The results obtained by InceptionNet v3, and Resnet-18 have been summarised in Table 1.

Performance Measures	InceptionNet v3	ResNet-18
Accuracy	98.65	99.06
Precision	98.29	98.76
Sensitivity	97.84	98.67
Specificity	99.85	99.89
F-Score	98.05	98.71
AUC	99.0	99.20
Time(min)	649.85	147.73

Table 1: Performance of CNN architectures [25]

The outcomes demonstrate that, although having a large number of layers, Inception V3 had the worst performance when compared to the other architectures.

3 Research Design & Methodology

3.1 Research Design

This work shows how to recover the input given only the gradients of the loss function with respect to the model's parameters. Sharing gradients among different trainers is a method of collaborative model training while keeping the data secured and separated from each trainer.

3.2 Setup

The high order gradients are computed by implementing algorithms, and we have selected PyTorch as our experiment platform. For the image tasks, we have employed L-BFGS optimizer with Sigmoid activation having a learning rate of 1, a history size of 100, and a maximum iteration limit of 20. All trainable parameters' slopes should match, is our goal. Notably, DLG has no limitations on the model's level of convergence; therefore the attack can occur at any moment while the training is being conducted. All of our tests have used weights that have been randomly initialised to be more inclusive. The following subsections contain more information about each activity.

3.3 Methodology

Changes made to make the Deep Leakage from Gradients to work for Resnet18, DenseNet121, InceptionNet V3. is shown in Table 2

Among the other changes include modification of Pytorch's Resnet-18 implementation to replace all ReLU activation layer by Sigmoid activation layer. By minimizing the L2 distance between the real data's gradients and randomized data' gradients using the L-BFGS optimizers, the randomized data can be transformed in a way to that makes it

Model	Lenet	Resnet18	DenseNet121	InceptionNet V3
Image size	32x32	224x224	224x224	299x299
Format	scaled to [0,1]	normalized using the mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225] for each channel	normalized using the mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225] for each channel	normalized using the mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225] for each channel
Model size	60,000	11 millions	8.1 millions	6 millions
Activation	Sigmoid	Original ReLU is replaced by Sigmoid	Original ReLU is replaced by Sigmoid	Original ReLU is replaced by Sigmoid
Convergence	Converges in about 100 training steps	Requires more than 300 steps to converge due to bigger image size and more parameters in the gradients	Converges in about 50 steps	Did not converge at 300 steps

Figure 1: Table 2: Changes made to Deep Leakage from Gradients

converge to the real data.

This paper has used L-BFGS optimizer and the activation has been done using Sigmoid to avoid the vanishing gradients at the second order. To accommodate that, Pytorch's Resnet-18 implementation has been modified to replace all ReLU activation layer by Sigmoid activation layer. Further, images input to the ResNet have been normalized by the following transformation layer: torch vision transforms. Normalize ([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]). The input to the ResNet-18 is images with minimum dimension of 224 (instead of 32 for LeNet).

The third change is that the input to the ResNet-18 is images with minimum dimension of 224 (instead of 32 for LeNet). Therefore, the image has been rescaled and cropped to 224x224 before feeding to the model. The effect of this change is that for the same training steps (300), LeNet is less noisy than ResNet-18. This means that the ResNet-18 will require much more training steps to achieve the visually similar output of LeNet.

4 Artefact Development Approach

DenseNet 121 DenseNet is a more advanced architecture of CNN for visual object recognition that uses fewer parameters to provide state-of-the-art performance. DenseNet and ResNet are quite similar, with a few key differences. While ResNet blends the output from the previous layer with the subsequent layers using an additive attribute (+), DenseNet employs concatenated (.) attributes to mix the output from the previous layer with the subsequent layer. Through tightly linking all layers, the DenseNet Architecture seeks to address this issue. This study used the DenseNet-121 architecture, which is one of three DenseNet models (DenseNet-121, DenseNet-160, and DenseNet-201). Details of the DenseNet-121 are as follows: 1 classification layer (16), 2 denseblock (1 1 and 3 3 conv), 5 convolution and pooling layers, 3 transition layers (6,12,24), and 1 transition layer. The output layers (lth) of conventional CNNs are typically calculated by applying a non-linear transformation Hl (.) to the output of the previous layer Xl-1.

amsmath

$$X_l = H_l(X_l - i) \tag{1}$$

DenseNets concatenate rather than sum the layer output functionality maps and the inputs. The lth layer receives inputs from the features of all preceding layers, and DenseNet provides a simple communication mechanism to improve information flow across layers. The equation is therefore rewritten as:

$$X_l = H_l[X_0, X_1, X_2..., X_i - 1 \tag{2}$$

where [X0, X1, X2,..., Xl-1,] is a single tensor made up of the output maps of earlier layers concatenated. Hl(.) is the only non-linear transformation function among the functions. Three main processes make up this function: activation (ReLU), pooling and convolution, and batch normalisation (BN) (CONV). However, the growth rate k aids in the following generalisation of the lth layer: k[l] = (k[0] + k(l-1)). where k[0] is referred to as the channel count. Figure 2 shows the architecture of DenseNet.

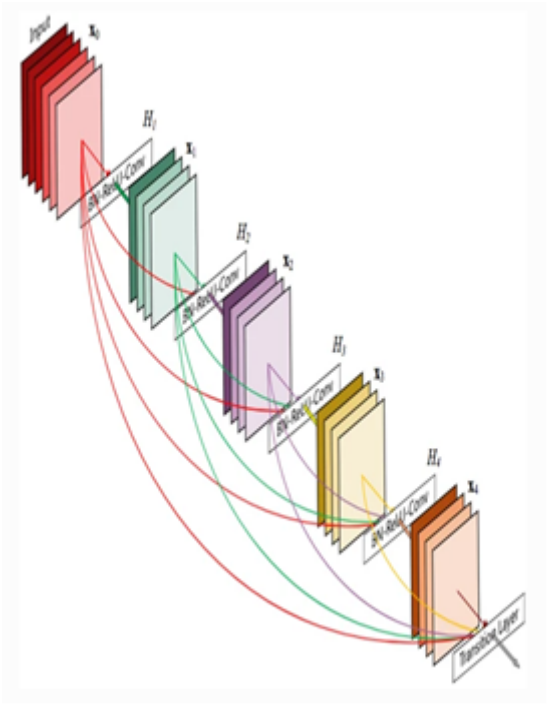

Figure 2: Architecture of DenseNet [26]

InceptionNet v3

On the ImageNet dataset, the DCNN model Inception-v3 has an accuracy rate of more than 78.1%. The model is the result of numerous concepts that have been established by various researchers over the years. Convolutions, average pooling, max pooling, concatenations, dropouts, and fully linked layers are some of the symmetric and asymmetric building components that make up the model itself. The model makes considerable use of batch normalisation, which is also applied to the activation inputs. To calculate loss, Softmax is used.

Inception-v3 has three different types of Inception modules: Inception A, Inception B, and Inception C. The well-designed convolution modules used in Inception can both produce distinguishing characteristics and minimise the number of parameters. Each Inception module is made up of parallel convolutional and pooling layers. The Inception modules use small convolutional layers like 3 X 3, 1 X 3, 3 X 1, and 1 X 1 layers to minimise the number of parameters. Three Inception A modules, five Inception B modules, and two Inception C modules are stacked one on top of the other in Inception-v3.

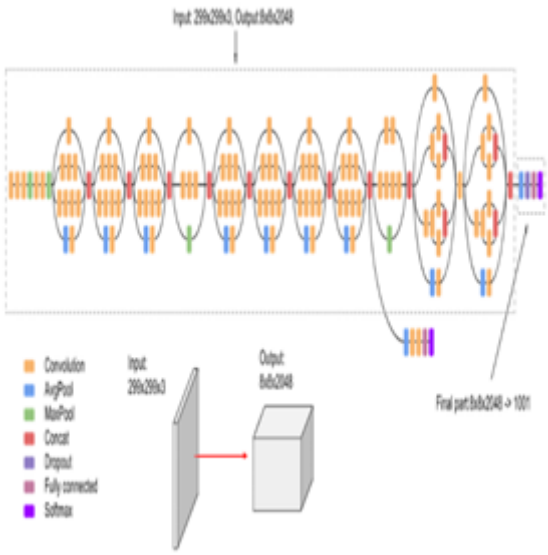

Figure 3: Architecture of InceptionNet v3 [23]

Inception-default v3's input image size is 299 X 299 pixels. The feature map dimensions after the convolutional layers and Inception modules were 55 with 2,048 channels. Figure 3 shows the architecture of InceptionNet v3.

LeNet

LeCun et al. proposed the convolutional neural network topology known as LeNet in 1998 [27]. LeNet or LeNet-5, in general, is a straightforward convolutional neural network that performs well in large-scale image processing because its artificial neurons may respond to a portion of the surrounding cells in the coverage range. LeNet, a prototype of the first convolutional neural network, possesses the fundamental components of a convolutional neural network, including the convolutional layer, pooling layer, and complete connection layer, providing the groundwork for its future advancement. Seven layers make up LeNet-5. Every other layer can train parameters in addition to input [28].

Convolution layer Layer C1 has six 5x5 convolution kernels and a 28x28 feature mapping, which can prevent input picture data from straying outside the convolution kernel's bounds. The subsampling/pooling layer, Layer S2, generates six feature graphs with a 14x14 size. Each feature map's cell is linked to the matching feature map in C1's 2x2 neighbourhood. A convolution layer with 16 5-5 convolution kernels is Layer C3.

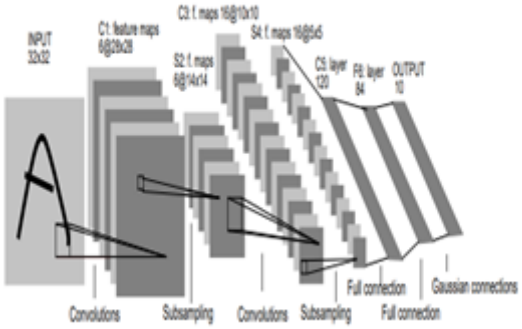

Figure 4: Architecture of LeNet [27]

Each continuous subset of the three feature maps in S2 is the input for the first six C3 feature maps. The input for the following six feature maps comes from the input of the following four continuous subsets. The input for the following three feature maps comes from the input of the following four discontinuous subsets. The input for the final feature graph is comprised of all S2 feature graphs. With a size of 2x2 and an output of 16 5x5 feature graphs, layer S4 is comparable to layer S2.

One hundred twenty 5x5 convolution kernels make up Layer C5, a convolution layer. Each cell on the 16 feature graphs of S4 is linked to the 5*5 neighbourhood. Here, the output size of C5 is 1*1 since the feature graph size of S4 is also 5x5. S4 and C5 are thus totally interconnected. Because C5's output size will be more than 1x1 and LeNet-5's input size increases but its structure stays the same, C5 is referred to as a convolutional layer rather than a fully connected layer. Fully coupled to C5, the F6 layer outputs 84 feature graphs. Figure 4 shows the architecture of LeNet.

ResNet 18

One of the models created by He et al. in 2016 is the deep residual network, also known as ResNet. This architecture eliminated problems with deep learning training, which is time-consuming and limited to a fixed number of layers. Applying to skip connections or take shortcuts provides an explanation for the complexity ResNet developed. The ResNets model has an advantage over other architectural models in that performance does not suffer as the design becomes more complex. Additionally, it is now possible to train networks more effectively and computation calculations are made lighter. By excluding connections on two to three levels that contain ReLU and batch normalisation

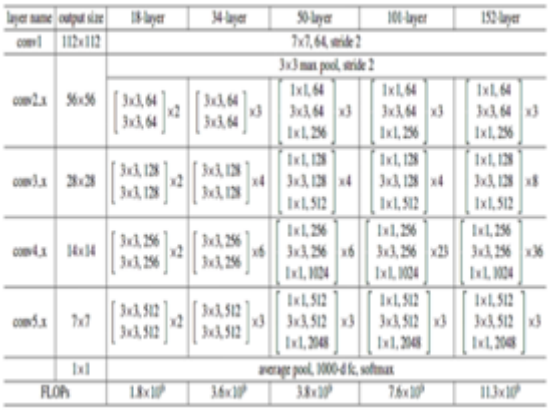

Figure 5: Architecture of ResNet-18 [30]

among the architectures, the ResNet model is implemented [20]. ResNet requires residual learning in multiple layers. The residual block on ResNet is calculated as [29]:
y = F(x, W + x)
where x represents the input layer; y represents the output layer; and F function represents the residual map.
If the dimensions of the input and output data are the same, residual block on ResNet can be achieved. Additionally, each ResNet block has three layers (ResNet-34 networks) or two layers (ResNet-18 networks) (ResNet-50 and ResNet-101 networks). By performing convolution 7 X 7 and max-pooling with size 3 X 3 and stride number 227, the first two layers of the ResNet architecture resemble GoogleNet. We employed the ResNet-18 and ResNet-50 models in this study. The fundus image is resized into a 224 x 224 grid. Stochastic Gradient Descents (SGD) are used to initialise the ResNet weights with conventional momentum parameters.The architecture of the ResNet is shown in Figure 5.

Mean Squared Error (MSE)
The model error is defined using the relation,

$$\in = x' - x \tag{3}$$

where x represents a vector of n observations and $\hat{x}$ represents the corresponding model predictions. Error can be decomposed into parts, the sum of which is equal to the total [31],

$$\in = \sum_{j=1}^{m} \in_j \tag{4}$$

Like all other errors, MSE can also be decomposed. For two error components, the MSE decomposition is given by the relation [31],

$$MSE(\in) = \frac{1}{n}\sum_{i=1}^{m} \in_i^2 \tag{5}$$

$$MSE(\in) = \frac{1}{n}\sum_{i=1}^{m} (\in_{1i} + \in_{2i})^2 \tag{6}$$

$$]MSE(\in) = \frac{1}{n}\sum_{i=1}^{m} (\in_{1i}^2 + 2 \in_1 i \in_2 i + \in_{2i}^2) \tag{7}$$

$$MSE(\in) = MSE(\in_1) + MSE(\in_2) + \frac{2}{n}\sum_{i=1}^{n} \in_1 i \in_2 i \tag{8}$$

where n is the number of observations. The last term on the right-hand side of the above equation is similar to the covariance between MSE1 and MSE2. For highly correlated MSE1 and MSE2 ,the sum of their products will be large; while the sum of the products will cancel if they are uncorrelated [31].

5 Empirical Evaluation

The efficacy of the models to prevent leakage has been evaluated on the basis of the value of Mean Squared Error by plotting a MSE versus steps plot both for the algorithm. Higher value of Mean Squared Error reduces the effectiveness of the algorithm.

6 Results & Discussion

In comparison with LeNet, ResNet-18 has much more parameters (11 millions vs 60000 of LeNet), which makes the derivative calculation much more computationally expensive. Hence, simpler the model, the faster it is to reconstruct the original image.

Figure 6 shows the input used to generate gradients.

Figure 7 shows the evolution of the original randomly initialized image to the

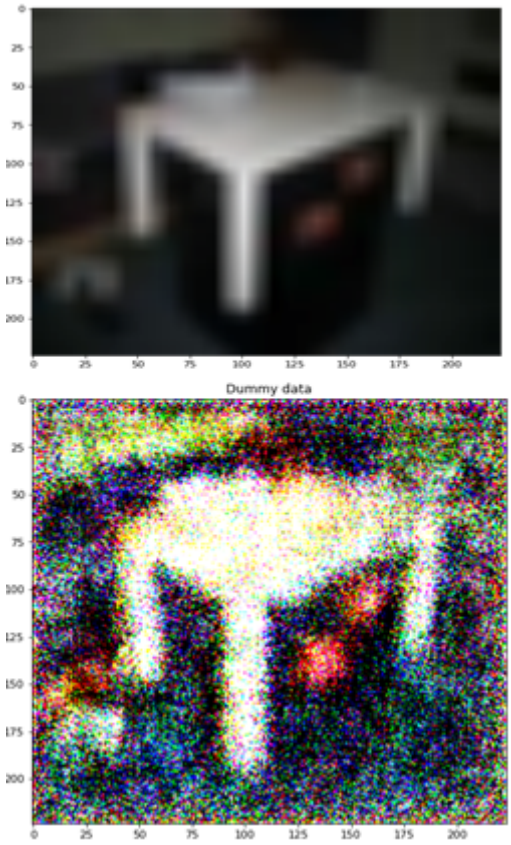

Figure 6: Input used to generate gradients.

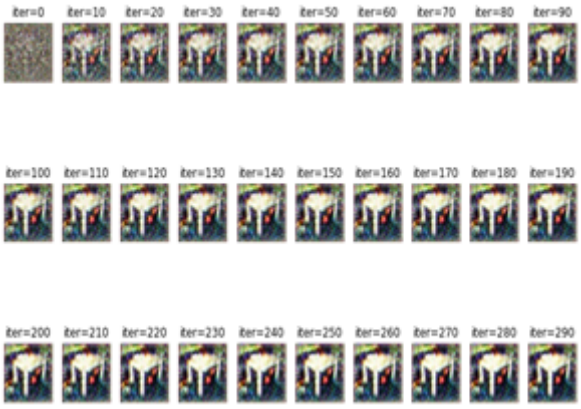

Figure 7: Reconstructed image

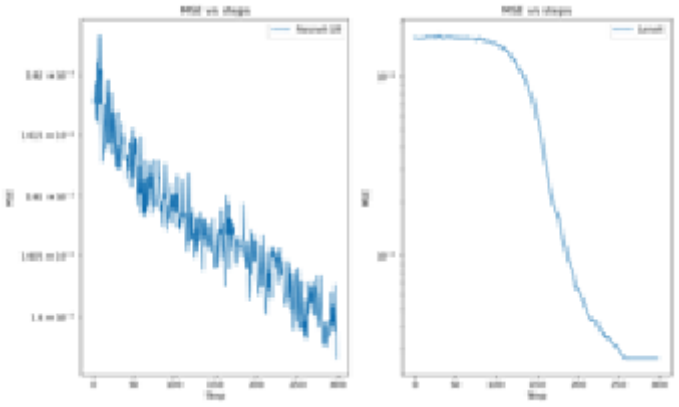

Figure 8: Comparing plot of MSE vs Step of ResNet-18 and LeNet

reconstructed image at step 300.

The difficulty in reconstructing input using Deep Leakage from Gradients from ResNet-18 over LeNet can also be seen in the plot of Mean Square Error vs step:

Figure 7 shows that for the same number of training steps, the MSE for Lenet is 1 order smaller compared to ResNet-18. This shows the efficacy of ResNet-18 over LeNet in preventing attacker to reconstruct the sensitive data.

The DenseNet-121 has the same pre-processing steps with the ResNet-18. Images are resized and cropped to 224 X 224, normalized. For InceptionNet V3, images were rescaled to 299 X 299. The bigger image size probably requires longer training time. All

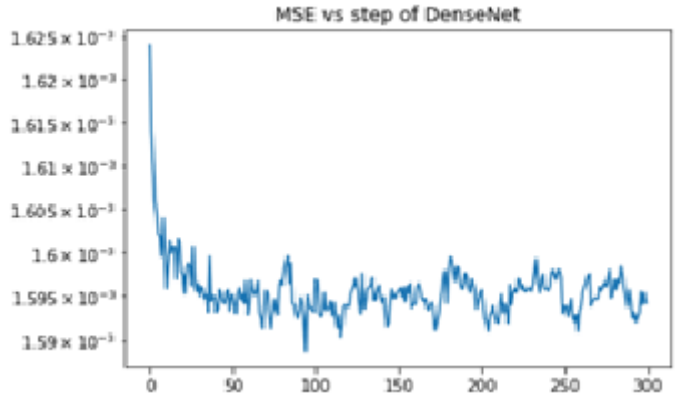

Figure 9: Plot of MSE vs Step of DenseNet-121

Figure 10: Reconstructed image using DenseNet-121

models have their activation changed from ReLU to Sigmoid.

The graph below (Figure 9) shows the MSE of the reconstructed image using Leakage of Gradients from a DenseNet-121 models.

The plot in Figure 9 shows that DenseNet-121 is susceptible to Leakage. The MSE is at the same order of magnitude compared to ResNet-18, both are not as low as LeNet. However, the convergences are much faster (only takes about 50 steps compared to 300 steps in ResNet). The image reconstructed after 300 steps is visually the same compared to ResNet:

However, with InceptionNet v3, the image cannot be reconstructed from the Leakage. First, the InceptionNet v3 has a batch normalization layer that requires the input batch to be at least 2 (compared to 1 in other models). Even when setting the batch size to 2, the MSE still does not converge. This indicates that InceptionNet v3 is a good choice if we want to keep the training data secret.

The plot in Figure 11 shows MSE vs steps for InceptionNet v3 as well as the reconstructed

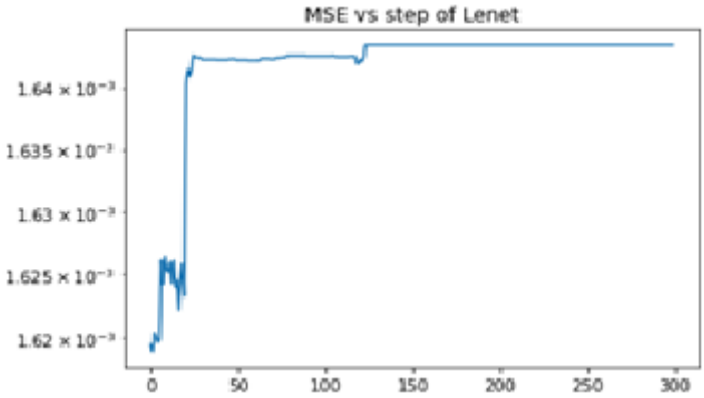

Figure 11: Plot of MSE vs Step of InceptionNet v3

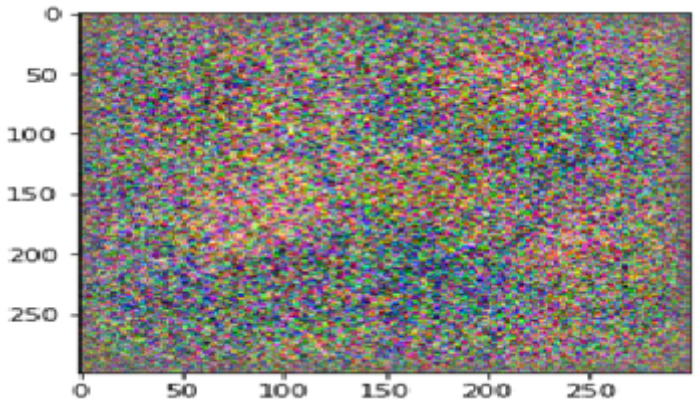

Figure 12: Reconstructed image using InceptionNet v3

image.

The study has observed that while ResNet-18 with a model size of 11 million outperforms LeNet which has a model size of 60000. DenseNet-121 (model size 8.1 million) is susceptible to leakage, while IncepionNet v3 (model size 6 million) fails to produce any image. The result implies that requirement of more variables during the process of optimization makes algorithms requiring larger model size less susceptible to leakages. However, larger model size has its limitation in the way of increasing the computational costs.

7 Threats to Validity

This study has replaced ReLu activation layer with Sigmoid activation layer to avoid the vanishing gradients. This makes it computationally expensive and inefficient. Further,

ReLu has a better convergence performance. The gradient of the ReLu function is either 00 for a<0a<0 or 11 for a>0a>0 enabling putting numerous layers.

8 Conclusion & Future Work

The scope of this study lies in understanding the process used to recover the input given only the gradients of the loss function with respect to the model's parameters. Sharing gradients among different trainers is a method of collaborative model training while keeping the data secured and separated from each trainer.

This study has observed that ResNet-18 has 11 million parameters compared to LeNet's 60000, which significantly increases the computing cost of calculating the derivative. We can therefore draw the conclusion that it takes less time to reconstruct the original image the simpler the model is.

The outcome of DenseNet 121 indicates that it is vulnerable to this Leakage. The magnitude of Mean Squared Error of DenseNet 121 is comparable to that of ResNet-18 but higher than the MSE of LeNet. However, it is not possible to recreate the image from the leakage using InceptionNet v3. Deep Learning from Gradients has convergence related problems and is ineficient in finding discovering ground-truth labels repeatedly.

8.1 Future Work

Future researchers may undertake studies on preventing leakages by increasing the batch size and up-scaling the batch size and resolution of the input images. Research can also be conducted on cryptology as a method of preventing leakage.

9 Reference

1. F. Iandola, M. Moskewicz, K. Ashraf and K. Keutzer, "FireCaffe: Near-Linear Acceleration of Deep Neural Network Training on Compute Clusters", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. Available: 10.1109/cvpr.2016.284.

2. P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations", Journal of Parallel and Distributed Computing, vol. 69, no. 2, pp. 117-124, 2009. Available: 10.1016/j.jpdc.2008.09.002.

3. A. Jochems et al., "Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital – A real life proof of concept", Radiotherapy and Oncology, vol. 121, no. 3, pp. 459-467, 2016. Available: 10.1016/j.radonc.2016.10.002.

4. A. Jochems et al., "Developing and Validating a Survival Prediction Model for NSCLC Patients Through Distributed Learning Across 3 Countries", International Journal of Radiation Oncology*Biology*Physics, vol. 99, no. 2, pp. 344-352, 2017. Available: 10.1016/j.ijrobp.2017.04.021.

5. L. Melis, C. Song, E. De Cristofaro and V. Shmatikov, "Exploiting Unintended Feature Leakage in Collaborative Learning", 2019 IEEE Symposium on Security and Privacy (SP), 2019. Available: 10.1109/sp.2019.00029.

6. B. Recht, C. Re, S. Wright and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent", In Advances in neural information processing systems, pp. 639-701, 2011.

7. H. McMahan, E. Moore, D. Ramage and S. Hampson et al, "Communication-efficient learning of deep networks from decentralized data", arXiv, 2016. Available: arXiv:1602.05629.

8. J. Koneˇcný, H. Brendan McMahan, F. Yu, P. Richtarik, A. Suresh and D. Bacon, "Federated learning: Strategies for improving communication efficiency", In NIPS Workshop on Private Multi-Party Machine Learning, 2016.

9. L. Melis, C. Song, E. De Cristofaro and V. Shmatikov, "Exploiting Unintended Feature Leakage in Collaborative Learning", 2019 IEEE Symposium on Security and Privacy (SP), 2019. Available: 10.1109/sp.2019.00029.

10.R. Shokri, M. Stronati, C. Song and V. Shmatikov, "Membership inference attacks against machine learning models", 2017 IEEE Symposium on Security and Privacy (SP), pp. 3-18, 2017.

11. I. Goodfellow et al., "Generative adversarial nets", Advances in neural information

processing systems, pp. 2672-2680, 2014.
12. L. Gao, L. Zhang, C. Liu and S. Wu, "Handling imbalanced medical image data: A deep-learning-based one-class classification approach", Artificial Intelligence in Medicine, vol. 108, p. 101935, 2020. Available: 10.1016/j.artmed.2020.101935.
13. B. Lodhi and J. Kang, "Multipath-DenseNet: A Supervised ensemble architecture of densely connected convolutional networks", Information Sciences, vol. 482, pp. 63-72, 2019. Available: 10.1016/j.ins.2019.01.012.
14. P. Carcagnì et al., "Classification of Skin Lesions by Combining Multilevel Learnings in a DenseNet Architecture", Lecture Notes in Computer Science, pp. 335-344, 2019. Available: 10.1007/978-3-030-30642-7-30.
15. N. Hasan, Y. Bao, A. Shawon and Y. Huang, "DenseNet Convolutional Neural Networks Application for Predicting COVID-19 Using CT Image", SN Computer Science, vol. 2, no. 5, 2021. Available: 10.1007/s42979-021-00782-7.
16. R. Shokri and V. Shmatikov, "Privacy-preserving deep learning", in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1310- 1321.
17. L. Zhu, Z. Liu and S. Han, "Deep leakage from gradients", arXiv, pp. 17-31, 2019.
18. D. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization", Mathematical Programming, vol. 45, no. 1-3, pp. 503-528, 1989. Available: 10.1007/bf01589116.
19. J. Geiping, H. Bauermeister, H. Dröge and M. Moeller, "Inverting gradients–how easy is it to break privacy in federated learning?", arXiv:2003.14053, 2020.
20. H. Yin, A. Mallya, A. Vahdat, J. Alvarez, J. Kautz and P. Molchanov, "See through gradients: Image batch recovery via grad inversion", in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16337–16346.
21. M. Balunovi´c, D. Dimitrov, R. Staab and M. Vechev, "Bayesian framework for gradient leakage", arXiv preprint arXiv:2111.04706, 2021.
22. B. Zhao, K. Mopuri and H. Bilen, "idlg: Improved deep leakage from gradients", arXiv preprint arXiv:2001.02610, 2020.
23. Q. Guan et al., "Deep convolutional neural network Inception-v3 model for differential diagnosing of lymph node in cytological images: a pilot study", Annals of Translational Medicine, vol. 7, no. 14, pp. 307-307, 2019. Available: 10.21037/atm.2019.06.29.
24. Y. Yang et al., "A comparative analysis of eleven neural networks architectures for small datasets of lung images of COVID-19 patients toward improved clinical decisions", Computers in Biology and Medicine, vol. 139, p. 104887, 2021. Available: 10.1016/j.compbiomed.2021.104887.
25. V. Maeda-Gutiérrez et al., "Comparison of Convolutional Neural Network

Architectures for Classification of Tomato Plant Diseases", Applied Sciences, vol. 10, no. 4, p. 1245, 2020. Available: 10.3390/app10041245.
26. G. Huang, Z. Liu, L. Van Der Maaten and K. Weinberger, "Densely Connected Convolutional Networks", 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. Available: 10.1109/cvpr.2017.243.
27. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998. Available: 10.1109/5.726791.
28. Y. LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition", Neural Computation, vol. 1, no. 4, pp. 541-551, 1989. Available: 10.1162/neco.1989.1.4.541.
29. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition ,", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.
30. A. Gummenson, "Prostate Cancer Classification using Convolutional Neural Networks", Masters, Centre for Mathematical Sciences Lund Institute of Technology, Lund University Lund, Sweden, 2016.
31. T. Hodson, T. Over and S. Foks, "Mean Squared Error, Deconstructed", Journal of Advances in Modeling Earth Systems, vol. 13, no. 12, 2021. Available: 10.1029/2021ms002681.