# Deep Leakage from gradients

Ligeng Zhu, Zhijian Liu, Song Han

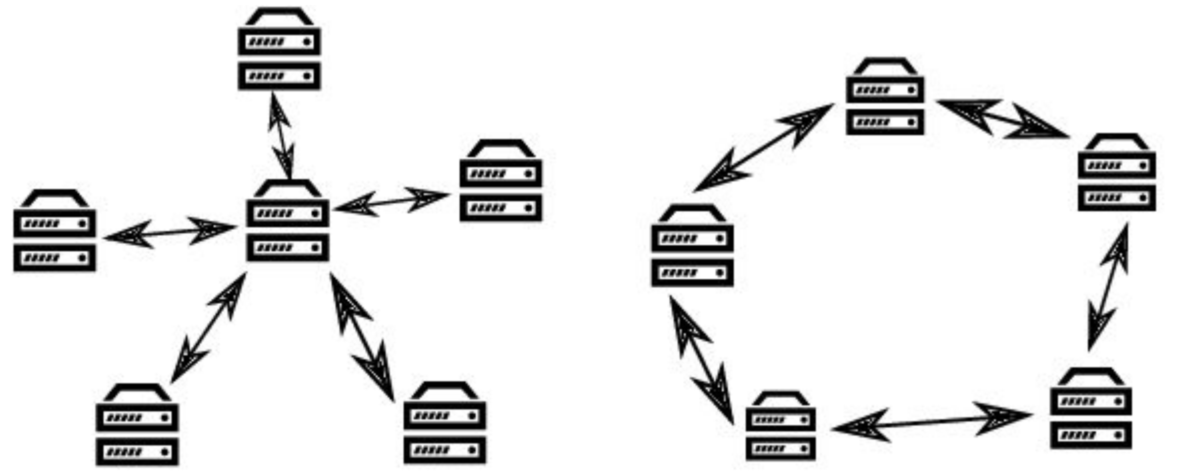
NeurIPS 2019

박범수

- BACKGROUND
- METHOD
- EXPERIMENTS
- CONCLUSION

# Background

- Distributed Training


(a) Parameter Server (b) Decentralized Architecture

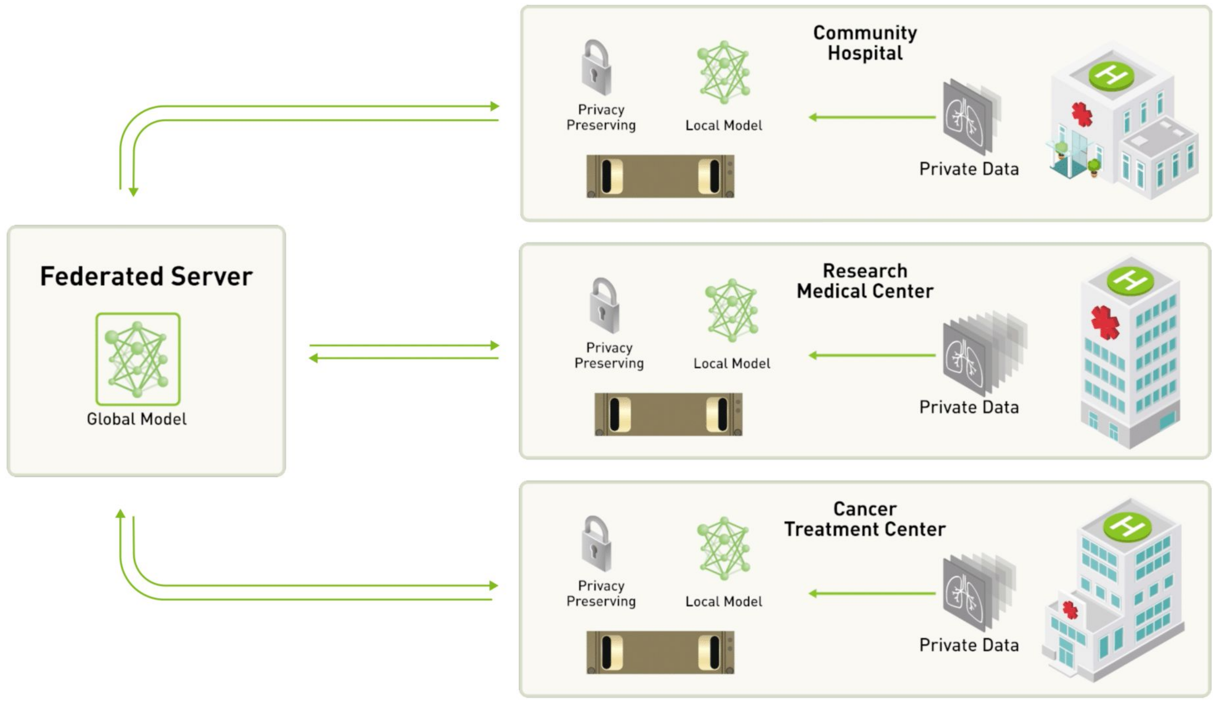
Distributed learning
커다란 모델을 training시키기 위한 방법
- centralized(parameter server)
- Decentralized(parameter server X)

Method
1. 각 노드에서 local weights update
2. Gradients를 다른 node와 공유

출처: https://www.researchgate.net/figure/A-centralized-distributed-learning-architecture-and-a-decentralized-distributed-learning_fig1_332342249

## Background

- Federated Training



Federated learning
데이터를 분산시켜 학습을 수행

- 2017년 resource활용을 위해 google이 제안
- 데이터의 집적이 발생하지 않음

**-> 개인정보 보호의 장점**이 생김
-> Really?

출처: https://resources.nvidia.com/en-us-federated-learning/what-is-it?%5D

## Background: “Shallow” Leakage

### Previous works

- (1) (CoRR 2018) “Exploiting unintended feature leakage in collaborative learning”

  -> adversarial participant can inter the presence of exact data point in others’ data
  -> 특정 개인정보가 누구에게 있는지 point 가능

- (2) (IEEE SP 2017) “Membership inference attack: given a data record and black-box access to a model,”

  -> 특정 참여자가 가입되어 있는지 여부를 알아낼 수 있음

  -> 모델이 Overfit하거나 data가 representative하지 않으면, 모델이 정보를 leak한다.

- (3) (CoRR 2017) “Deep models under the GAN: information leakage from collaborative deep learning”

  -> GAN을 이용해 다른 유저가 학습한 파라미터로 피해자를 닮도록 생성모델을 mimic

  -> 피해자의 local 모델이 improve하는 동안 GAN공격 가능

## Method

### Standard synchronous distributed training

- At each step t, every node i samples a minibatch $(x_{t,i}, y_{t,i})$ from its own dataset to compute the gradient

$$\nabla W_{t,i} = \frac{\partial \ell(F(\mathbf{x}_{t,i}, W_t), \mathbf{y}_{t,i})}{\partial W_t}$$

- The gradients are averaged across the N servers and then used to update the weights

$$\nabla W_t = \frac{1}{N} \sum_{j}^{N} \nabla W_{t,j}; \quad W_{t+1} = W_t - \eta \nabla W_t$$

- Given gradients, we aim to steal participant k's training data $(x_{t,k}, y_{t,k})$. F() and $W_t$ are default

## Method

### Training Data Leakage through Gradients Matching

- First, randomly initialize a dummy input x’ and label input y’. And feed these “dummy data” into models and get “dummy gradients”

$$\nabla W' = \frac{\partial \ell(F(\mathbf{x}', W), \mathbf{y}')}{\partial W}$$

- Optimizing the dummy gradients close to original also makes dummy data close to real data

$$\mathbf{x}'^*, \mathbf{y}'^* = \arg\min_{\mathbf{x}', \mathbf{y}'} ||\nabla W' - \nabla W||^2 = \arg\min_{\mathbf{x}', \mathbf{y}'} ||\frac{\partial \ell(F(\mathbf{x}', W), \mathbf{y}')}{\partial W} - \nabla W||^2$$

- Distance $||\nabla W0 - \nabla W||^2$ is differentiable w.r.t x’ and y’. Thus can be optimized using standard gradient-based methods. We need assumption that F is twice differentiable, which holds for majority ML models

## Method

Standard synchronous distributed training

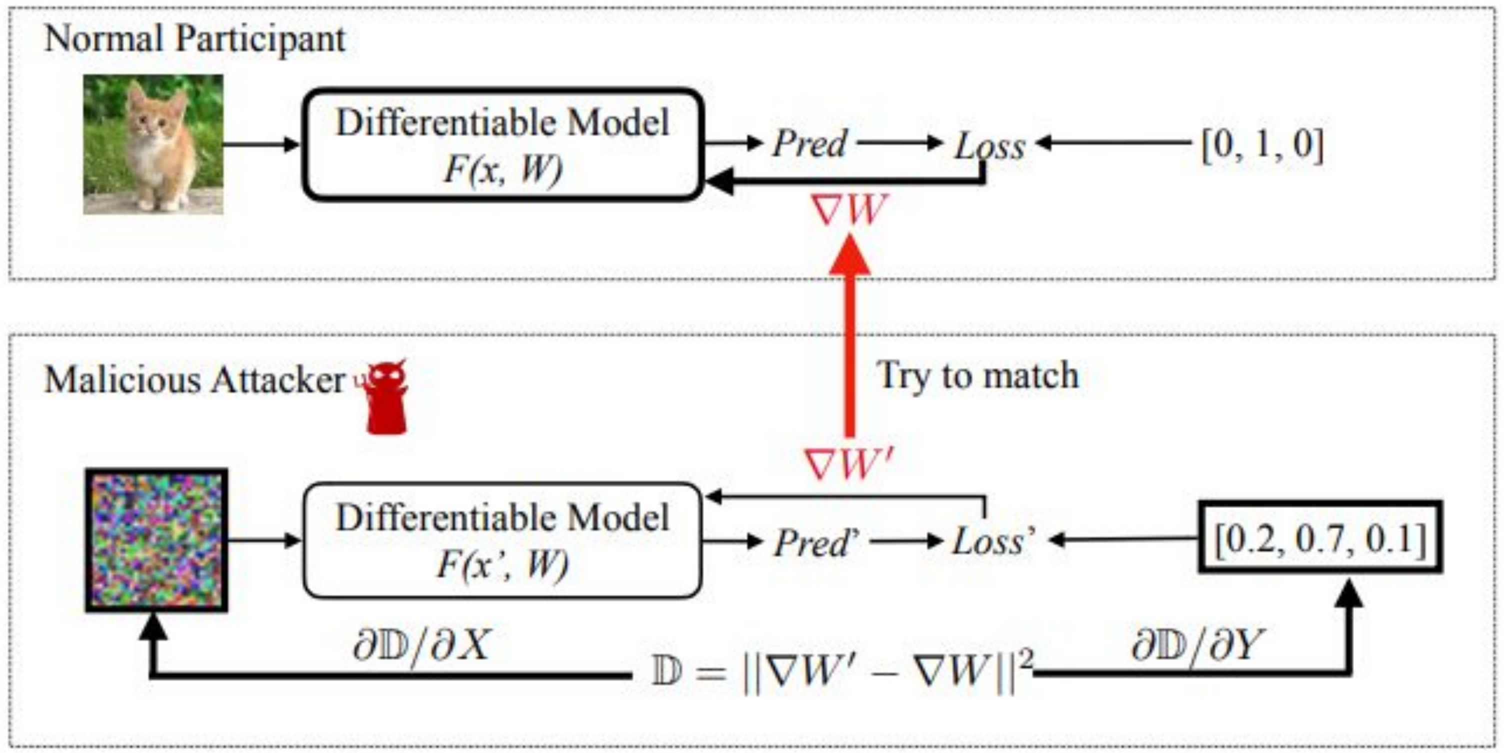
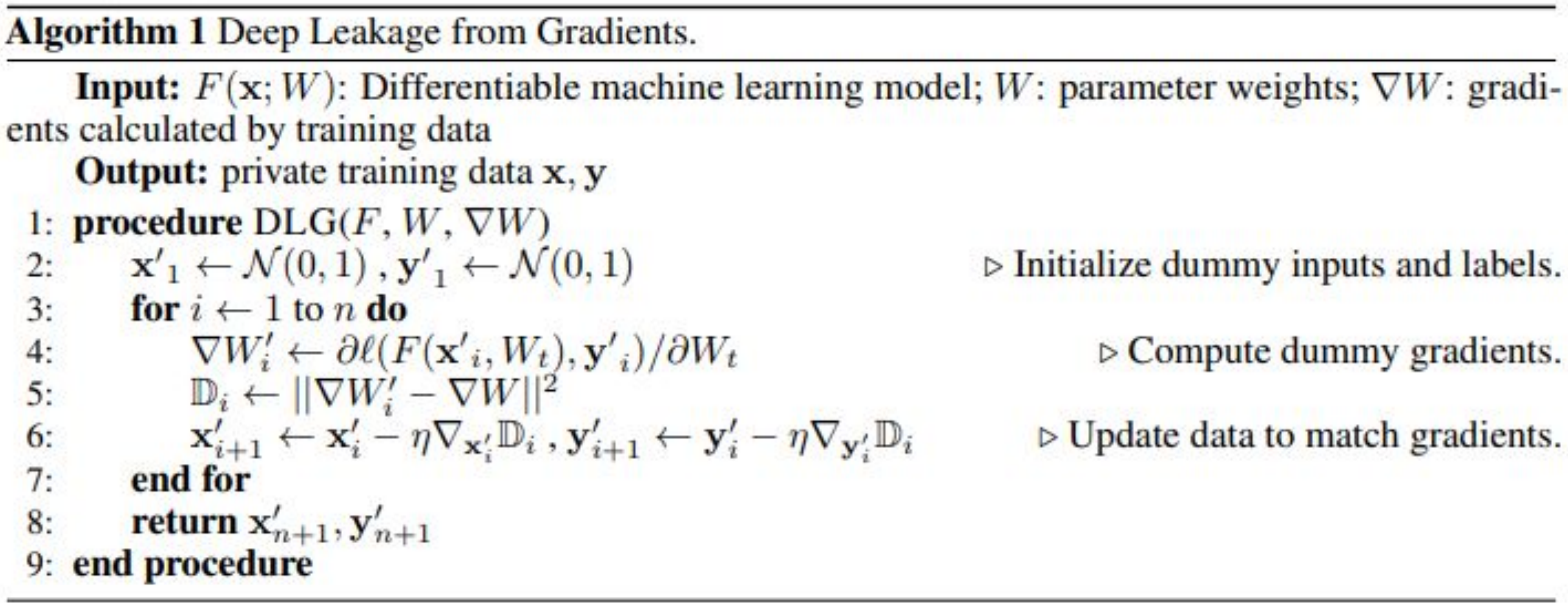


Figure 2: The overview of our DLG algorithm. Variables to be updated are marked with a bold border. While normal participants calculate $\nabla W$ to update parameter using its private training data, the malicious attacker updates its dummy inputs and labels to minimize the gradients distance. When the optimization finishes, the evil user is able to obtain the training set from honest participants.

## Method

### Training Data Leakage through Gradients Matching



**Algorithm 1** Deep Leakage from Gradients.

**Input:** $F(\mathbf{x}; W)$: Differentiable machine learning model; $W$: parameter weights; $\nabla W$: gradients calculated by training data

**Output:** private training data $\mathbf{x}, \mathbf{y}$

```
1: procedure DLG($F$, $W$, $\nabla W$)
2:     $\mathbf{x}'_1 \leftarrow \mathcal{N}(0,1)$ , $\mathbf{y}'_1 \leftarrow \mathcal{N}(0,1)$          ▷ Initialize dummy inputs and labels.
3:     for $i \leftarrow 1$ to $n$ do
4:         $\nabla W'_i \leftarrow \partial \ell(F(\mathbf{x}'_i, W_t), \mathbf{y}'_i)/\partial W_t$          ▷ Compute dummy gradients.
5:         $\mathbb{D}_i \leftarrow ||\nabla W'_i - \nabla W||^2$
6:         $\mathbf{x}'_{i+1} \leftarrow \mathbf{x}'_i - \eta \nabla_{\mathbf{x}'_i} \mathbb{D}_i$ , $\mathbf{y}'_{i+1} \leftarrow \mathbf{y}'_i - \eta \nabla_{\mathbf{y}'_i} \mathbb{D}_i$          ▷ Update data to match gradients.
7:     end for
8:     return $\mathbf{x}'_{n+1}, \mathbf{y}'_{n+1}$
9: end procedure
```

## Method

### Training Data Leakage through Gradients Matching

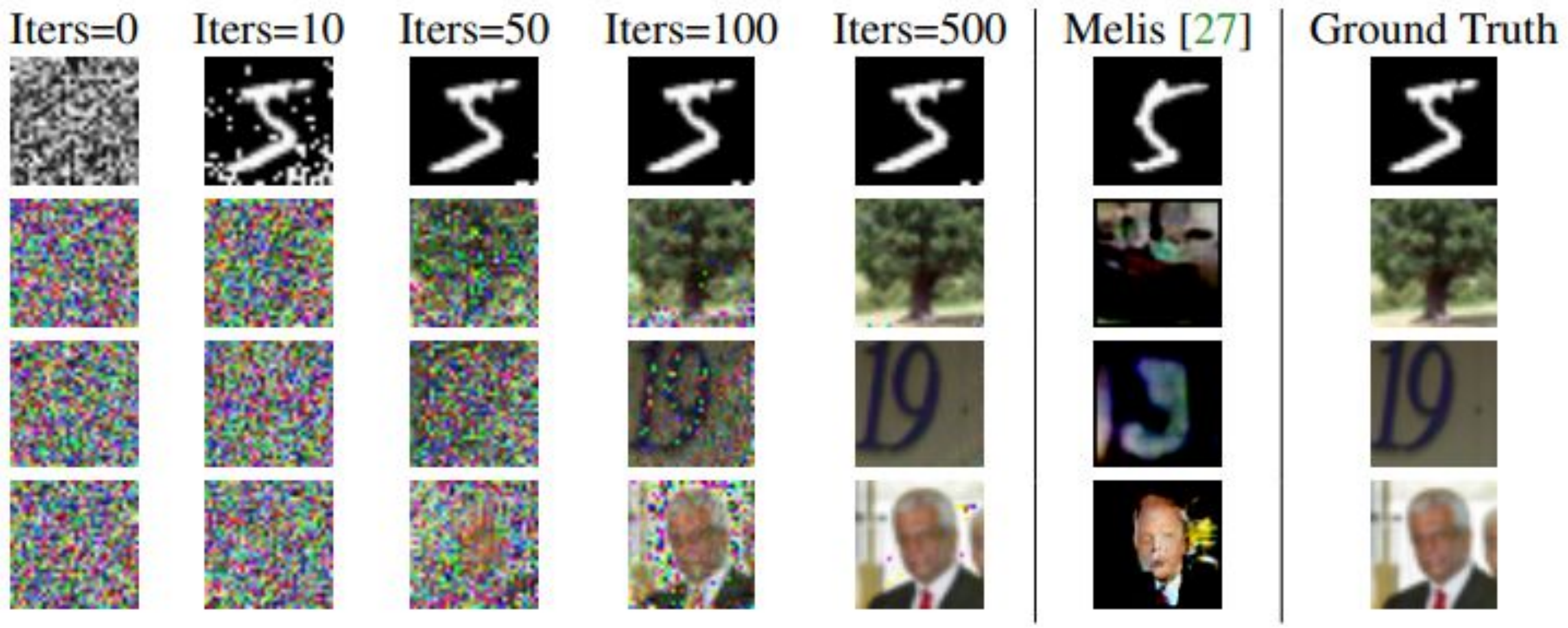


Figure 3: The visualization showing the deep leakage on images from MNIST [22], CIFAR-100 [21], SVHN [28] and LFW [14] respectively. Our algorithm fully recovers the four images while previous work only succeeds on simple images with clean backgrounds.

## Method

### Deep Leakage for Batched Data

- When batch size N>=1, the algorithm would be too slow to converge.

- Instead of updating the whole batch, update a single training sample instead

$$\mathbf{x}'^{\,i \bmod N}_{t+1} \leftarrow \mathbf{x}'^{\,i \bmod N}_{t} - \nabla_{\mathbf{x}'^{\,i \bmod N}_{t+1}} \mathbb{D}$$
$$\mathbf{y}'^{\,i \bmod N}_{t+1} \leftarrow \mathbf{y}'^{\,i \bmod N}_{t} - \nabla_{\mathbf{y}'^{\,i \bmod N}_{t+1}} \mathbb{D}$$

- Then observe fast and stable convergence.

| | BS=1 | BS=2 | BS=4 | BS=8 |
|---|---|---|---|---|
| ResNet-20 | 270 | 602 | 1173 | 2711 |

Table 1: The iterations required for restore batched data on CIFAR [21] dataset.

- BACKGROUND
- METHOD
- EXPERIMENTS
- CONCLUSION

## Experiments - Image

- PyTorch
- L-BFGS, learning rate 1.0
- Randomly initialized weights

- ResNet-56
- Data: MNIST, CIFAR-100, SVHN, LFW
- Model change
  - 1. activation ReLU to Sigmoid
  - 2. removing stride
  - -> to be twice-differentiable

# Experiments - Image

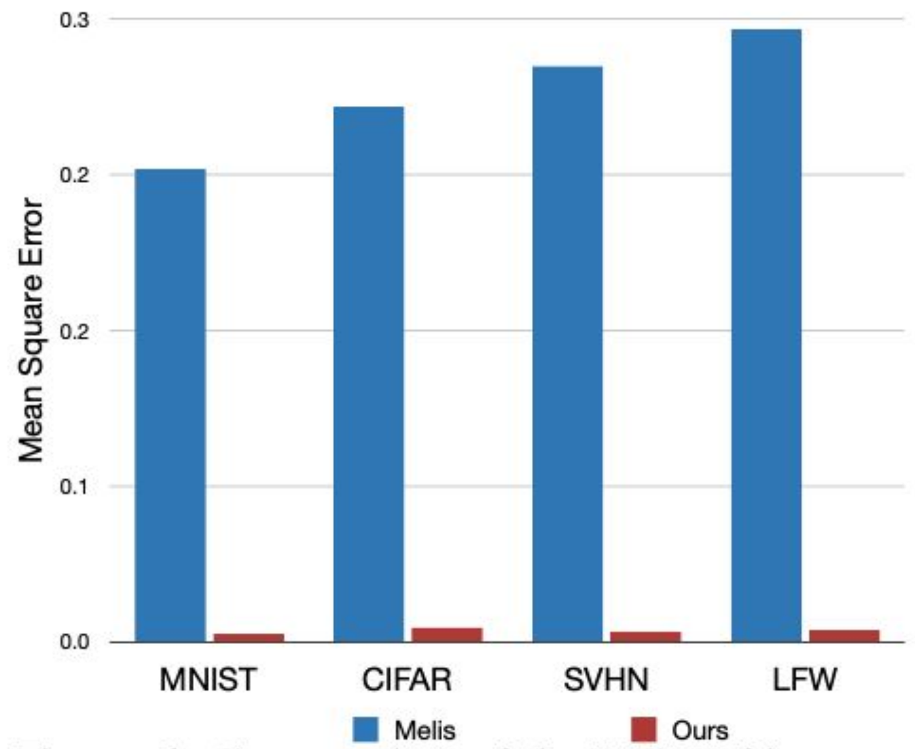


Figure 6: Compassion of the MSE of images leaked by different algorithms and the ground truth. Our method consistently outperforms previous approach by a large margin.

- Gradients 간 distance를 줄이는 것이 data간의 거리를 줄인다

- Complex face images는 여러 iteration이 필요

- MNIST같은 clean background는 recover이 쉬움

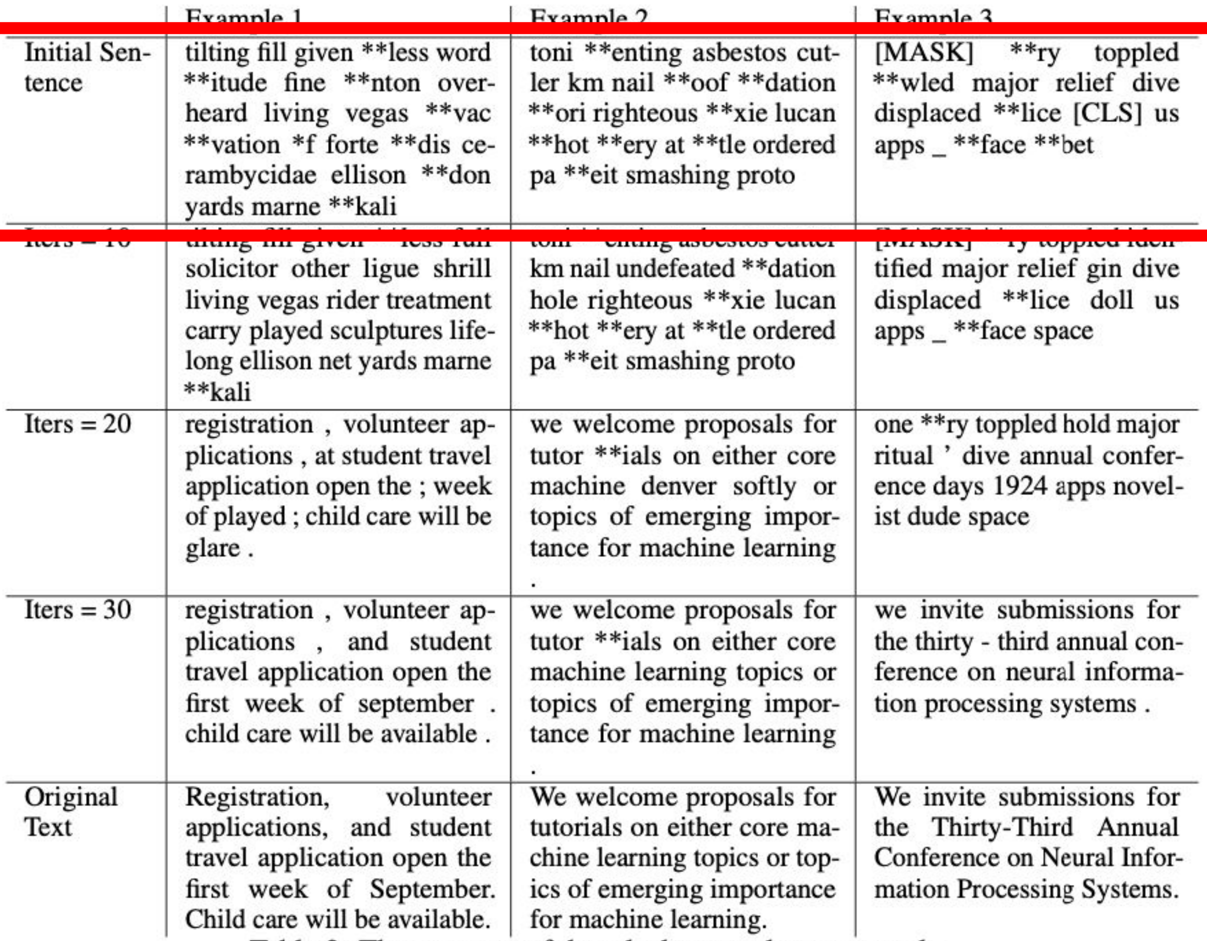
- MSE 비교 결과 GAN기반방식보다 우월

## Experiments – Language

- PyTorch
- L-BFGS, learning rate 1.0
- Randomly initialized weights

- Task: masked language model task
- Backbone: BERT
- Embedding space

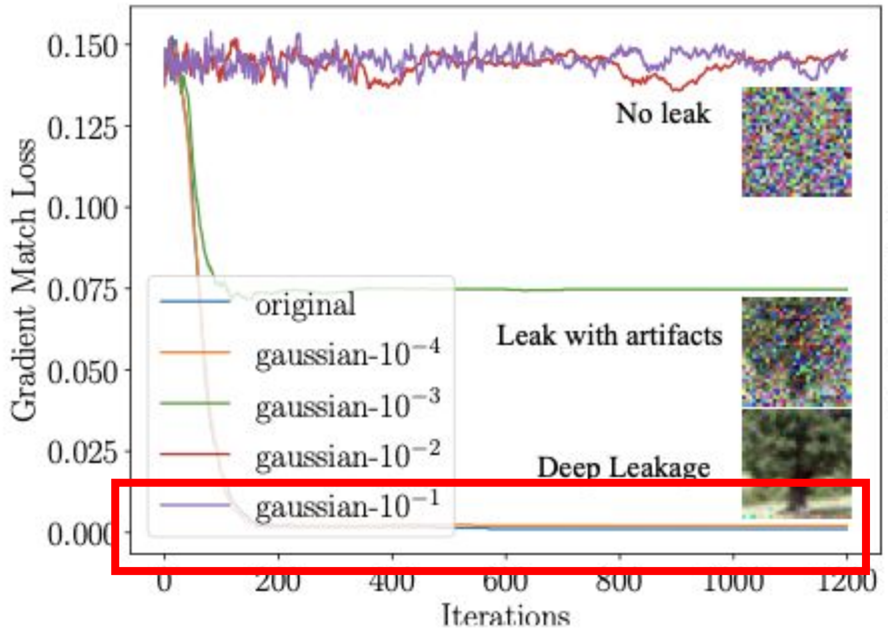
# Experiments – Language



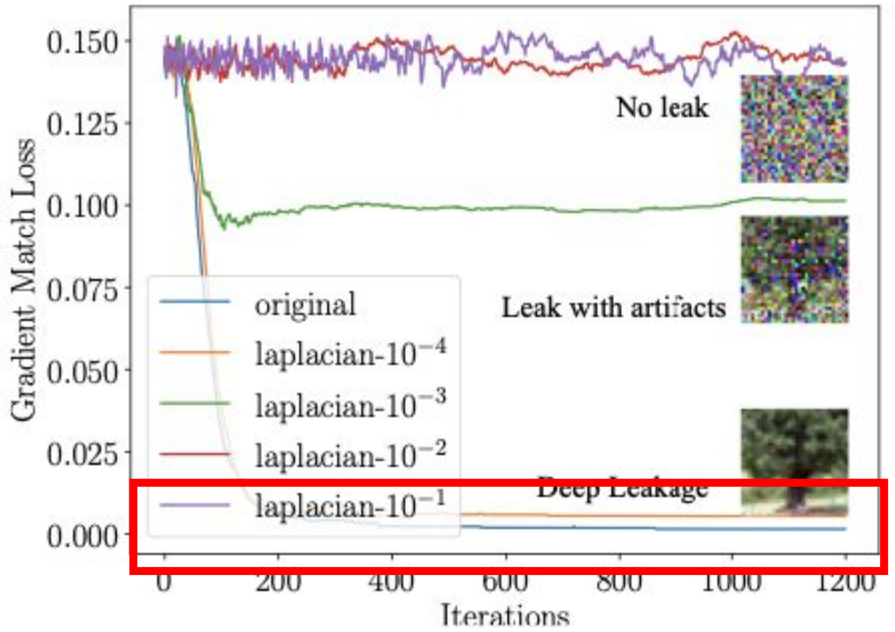
| | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| Initial Sentence | tilting fill given **less word **itude fine **nton overheard living vegas **vac **vation *f forte **dis cerambycidae ellison **don yards marne **kali | toni **enting asbestos cutler km nail **oof **dation **ori righteous **xie lucan **hot **ery at **tle ordered pa **eit smashing proto | [MASK] **ry toppled **wled major relief dive displaced **lice [CLS] us apps _ **face **bet |
| Iters = 10 | tilting fill given **less full solicitor other ligue shrill living vegas rider treatment carry played sculptures lifelong ellison net yards marne **kali | toni **enting asbestos cutter km nail undefeated **dation hole righteous **xie lucan **hot **ery at **tle ordered pa **eit smashing proto | [MASK] **ry toppled identified major relief gin dive displaced **lice doll us apps _ **face space |
| Iters = 20 | registration , volunteer applications , at student travel application open the ; week of played ; child care will be glare . | we welcome proposals for tutor **ials on either core machine denver softly or topics of emerging importance for machine learning . | one **ry toppled hold major ritual ' dive annual conference days 1924 apps novelist dude space |
| Iters = 30 | registration , volunteer applications , and student travel application open the first week of september . child care will be available . | we welcome proposals for tutor **ials on either core machine learning topics or topics of emerging importance for machine learning . | we invite submissions for the thirty - third annual conference on neural information processing systems . |
| Original Text | Registration, volunteer applications, and student travel application open the first week of September. Child care will be available. | We welcome proposals for tutorials on either core machine learning topics or topics of emerging importance for machine learning. | We invite submissions for the Thirty-Third Annual Conference on Neural Information Processing Systems. |

Table 2: The progress of deep leakage on language tasks.

## Defense – (1) Noisy Gradients

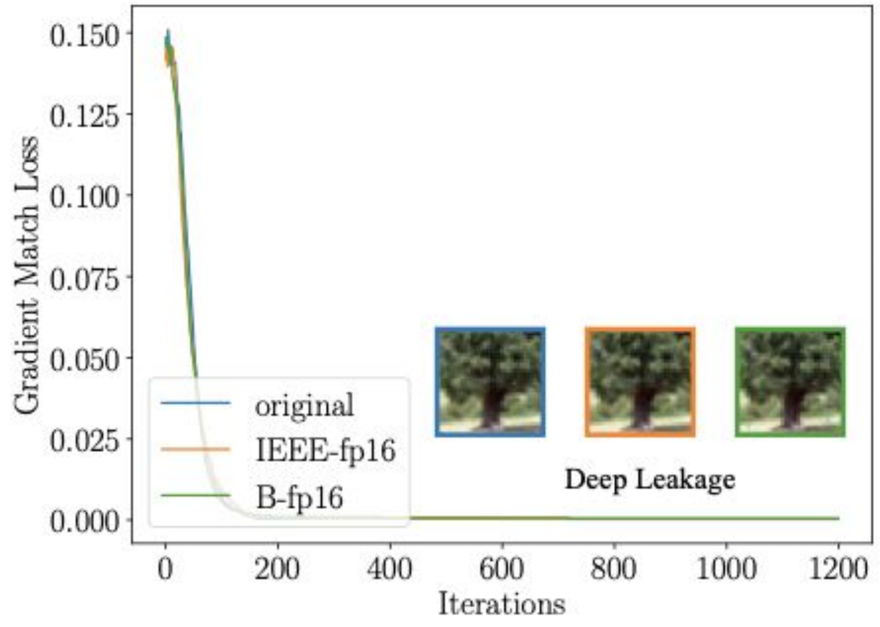


(a) Defend with different magnitude Gaussian noise.



(b) Defend with different magnitude Laplacian noise.

## Defense – (1) Noisy Gradients



(c) Defend with fp16 convertion.

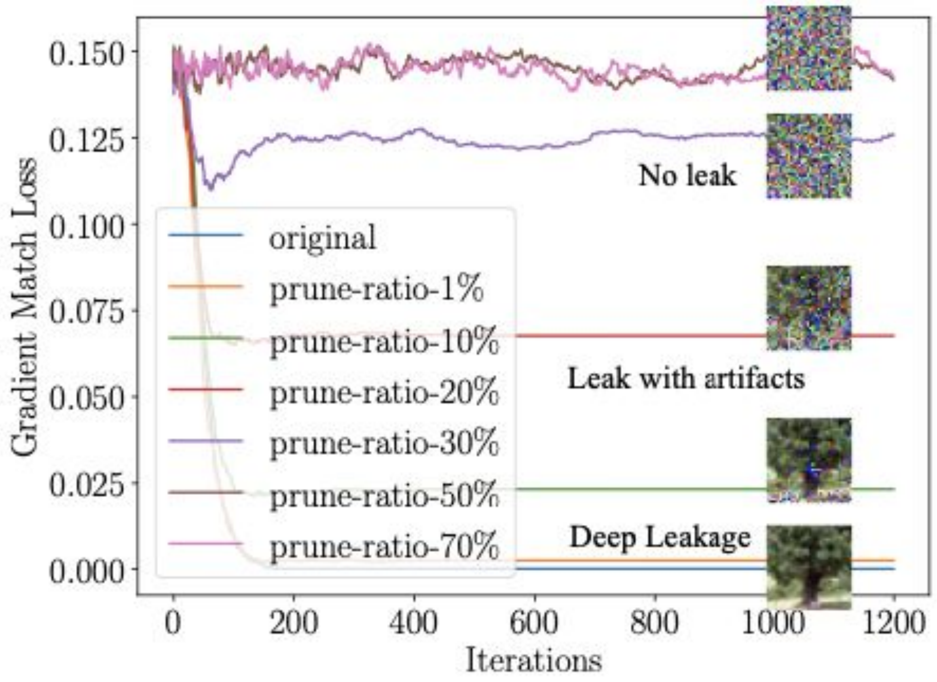
**Half Precision**

– GPU의 메모리 절약, 속도 향상

– Int8 사용시, leakage 방지

– But, Model performance가 떨어지는 문제가 발생

## Defense – (2) Gradient compression



(d) Defend with gradient pruning.

**Gradient Compression**

- Gradient의 압축정도에 따라 Leakage측정

- Prune ration > 20%, recover 장애발생

- 300x까지 압축 가능 -> DLG 방지 가능

## Defense – (3) Large Batch, Cryptograpy

**Large Batch, Upscaling**
– Batch size가 커지면 leakage가 difficult
– BS 8, Resolution 64x64가 max

**Cryptograpy**
– Gradients의 암호화
– But, requires integers gradients

- BACKGROUND
- METHOD
- EXPERIMENTS
- CONCLUSION

## Conclusion

- Deep Leakage from gradients(DLG)를 제안
- Local training data를 public shared gradients로부터 obtain
- GAN, extra prior data info가 필요 없음
- 원본을 pixel, token 단위로 복구할 수 있음을 보임
- Modern multi-node learning system에 challenge

## Reference

- **Deep Leakage from gradients** https://arxiv.org/pdf/1906.08935.pdf

- (CVPR2016) Firecaffe: near-linear acceleration of deep neural network training on compute clusters.
- (CoRR 2018) Exploiting unintended feature leakage in collaborative learning
- (2017 IEEE SP) Membership inference attacks against machine learning models
- (2017 CoRR) Deep models under the GAN: information leakage from collaborative deep learning

- Half Precision https://hoya012.github.io/blog/Mixed-Precision-Training/
- L-BFGS https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=sw4r&logNo=221475376608
- one-hot encoding https://azanewta.tistory.com/46
- second-order derivative https://byjus.com/maths/second-order-derivative/
- Scaling Distributed Machine Learning with the Parameter Server
- All-Reduce https://brunch.co.kr/@chris-song/96