2024 鐵人賽 – 我數學就爛要怎麼來學 DNN 模型安全

Day 21 – Deep Leakage from Gradients

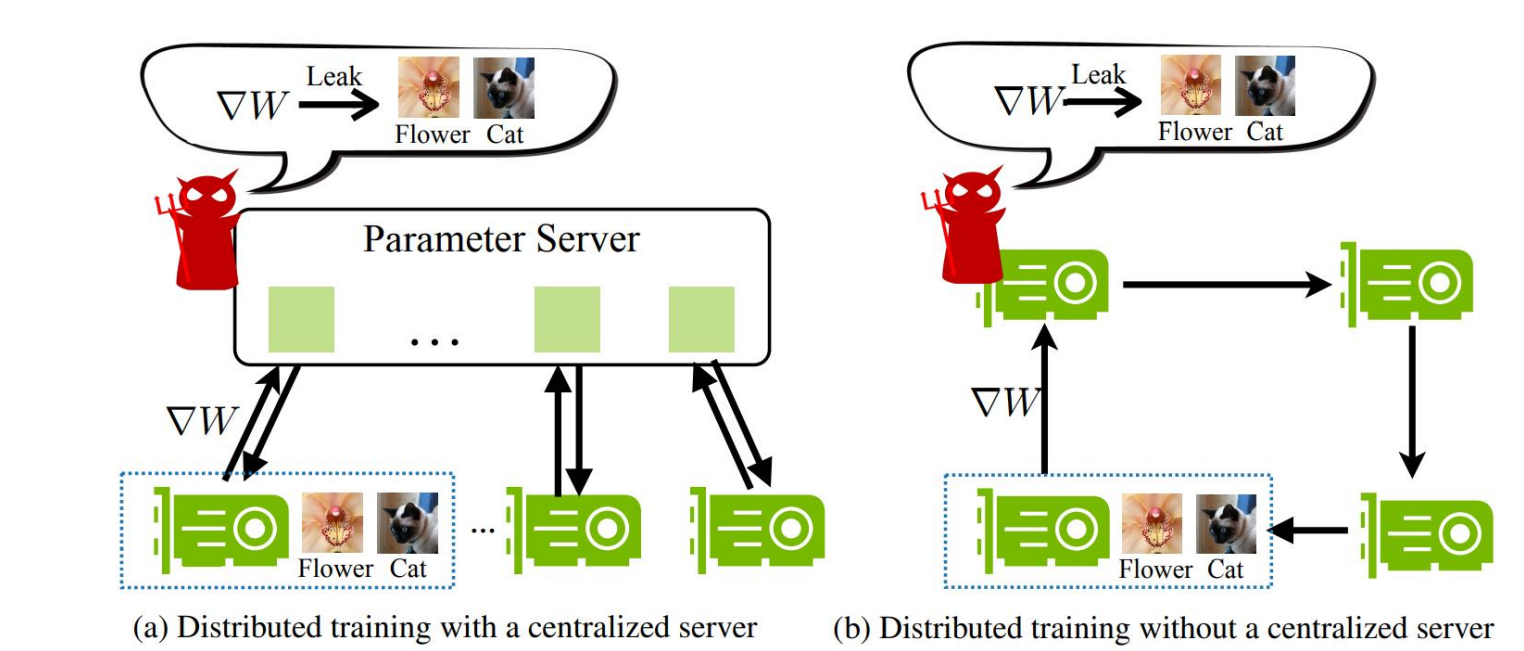
大綱

- Deep Leakage from Gradients
  - 攻擊手法原理
  - 使用條件及時機
  - 程式實作
- 結論

攻擊手法原理

https://arxiv.org/pdf/1906.08935

- ML05:2023 Model Theft
  - 攻擊者有權限去讀取機器學習模型內的結構及參數
  - 當攻擊知道某一組測試資料的 Gradients 時就有機會回推出該資料數值跟對應到的 Label

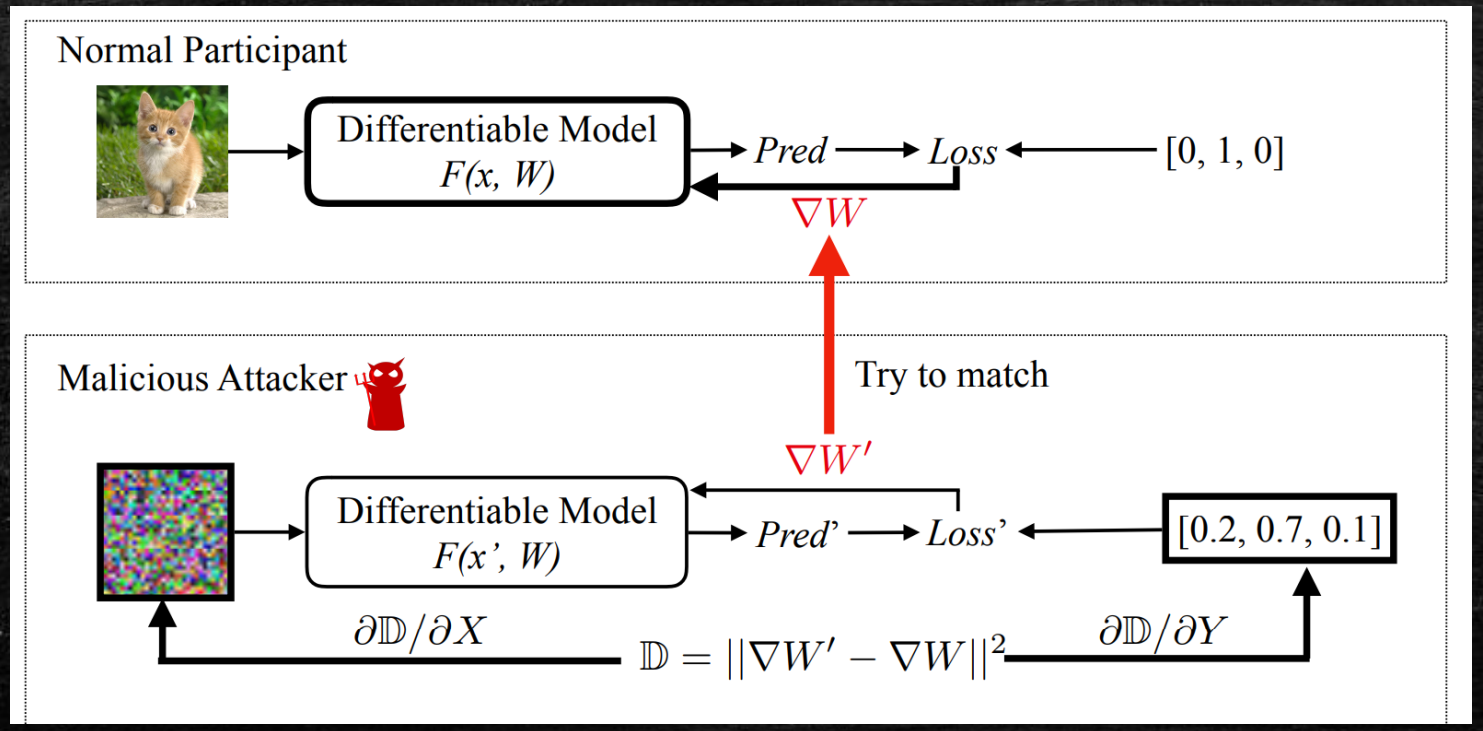


(a) Distributed training with a centralized server (b) Distributed training without a centralized server

攻擊手法原理

Figure 2: The overview of our DLG algorithm. Variables to be updated are marked with a bold border. While normal participants calculate ∇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.

- Deep Leakage from Gradients (2019)
  - 簡而言之，當製造出一組資料及標籤的梯度跟得到的一致時，這組資料及標籤就接近當初的輸入資料



https://arxiv.org/pdf/1906.08935

程式實作 – 演算法

- Deep Leakage from Gradients (2019)

Algorithm 1 Deep Leakage from Gradients.

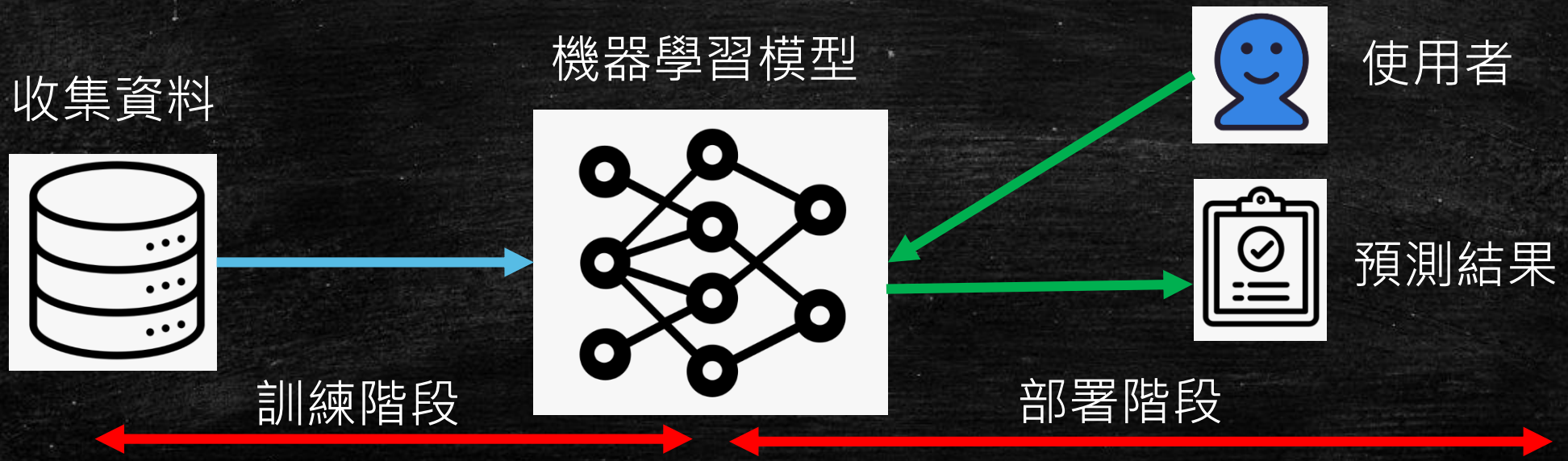
Input: $F(\mathbf{x}; W)$: Differentiable machine learning model; W: parameter weights; ∇W: gradients calculated by training data

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

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

使用條件及時機

- 時機點：訓練或是部署階段
- 前提：攻擊者必須擁有機器學習模型以及輸入資料梯度
- 攻擊效果：可以從梯度回推出輸入數值及標籤



程式實作 – 參考資料 - 神之一手模型

- 這個演算法不好做，因為 loss function 變成**輸入資料、標籤得到的梯度差異**，然後更新資料變成回去**更新輸入資料跟標籤**
- 回想一下以前的模型的情況是 loss function 是**預測值跟實際值的差異**，然後更新資料是**更新模型的權重資料**
- 所以，要想辦法把輸入資料，標籤數值搞成模型權重去做更新，想法會很有趣

https://medium.com/@EastGeno/deep-leakage-from-gradients-%E5%BE%9E%E6%A2%AF%E5%BA%A6%E6%8B%BF%E5%88%B0%E4%BD%A0%E7%9A%84%E8%B3%87%E6%96%99-d23232c03bd2

程式實作

- 先想辦法克服怎麼讓模型可以接收輸入資料、輸出標籤後回傳梯度

```
# 定應屬於自己的 base model
class myModule(tf.keras.Model):
    def __init__(self) :
        super(myModule,self).__init__()
        # 這邊只做一層是因為做兩層的回覆效果就不好了
        #self.dense1 = Dense(128,activation=tf.nn.sigmoid)
        self.dense2 = Dense(10,activation=tf.nn.softmax)

    def call(self, inputs, training=True) :
        #x = self.dense1(inputs)
        #x = self.dense2(x)
        x = self.dense2(inputs)
        return x

    # 這邊在模型內定義一個 function, 傳入輸入資料跟標籤會回傳當下 gradient


my_model = myModule()
```

結論

- DLG 的概念不難，但實作還蠻難的，我當初想到腦袋都打結了
- 參考資料的神之一手模型真的很值得認真 trace code 看看，會讓人驚呼到原來還可以這樣玩模型啊