《PDGAN: A Novel Poisoning Defense Method in Federated Learning Using Generative Adversarial Network》

文章的主要思想是：

提出了一种新型的防投毒的生成对抗网络 PDGAN。 PDGAN 可以从模型更新中，对联邦学习中每个参与者的局部训练数据进行重构，并对其准确度进行审计。其准确性低于预定义阈值的参与者将被标识为攻击者，并且将在此迭代中的训练过程删除攻击者的模型参数。 在 MNIST 和 FASHION-MNIST 数据集上进行的实验表明，本文的方法确实可以防御联合学习中的中毒攻击。

PDGAN：

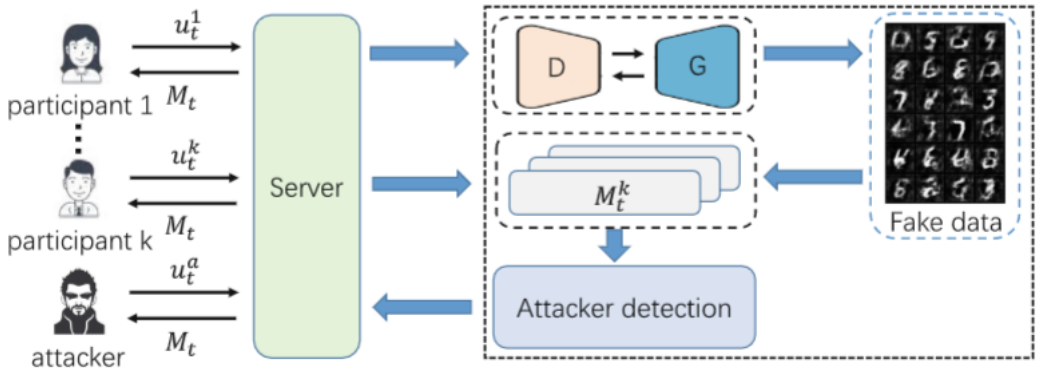


Fig. 3. Overview of the proposed PDGAN method in the federated learning.

1. 服务器具备辅助数据集。由于各客户端数据差异性大，辅助数据集难以涵盖所有数据类别，本文为解决这个问题，在服务器上实现了 GAN 来重建参与者训练数据。（服务器侵犯隐私？）
2. 在获得生成的数据之后，服务器通过使用每个参与者上载的更新 u 和先前迭代的全局模型 Mt-1 为每个参与者建立分类模式 M。
3. GAN 只能用来重建训练数据，但无法访问数据标签。 故服务器将生成的数据提供给每个参与者模型，然后获得预测结果，并指定出现次数最多的标签是每个数据的真实标签。
4. 获得生成的数据及其标签后，就可以计算出参与者模型的准确性。 服务器通过预定义的准确性阈值 n 将参与者分为良性参与者和攻击者两个集群。
5. 如果一个参与者被判定为攻击者，则 PDGAN 将在此迭代中忽略其更新。 算法 1 显示了 PDGAN 下的联邦学习算法。

Algorithm 1. PDGAN in Federated Learning

Data: Parameters updates u_t^k from participant k at iteration t; global model M_{t-1} at iteration $t-1$; Auxiliary data X_{aux}; Labellist L; Accuracy threshold η

Initialize Generator $\mathcal{G}$
for *Iteration t* **do**
  Receive updates from the selected participants, $\{u_t^1, u_t^2, ..., u_t^k\}$
  Generate X_{fake} from $\mathcal{G}$
  Update the Discriminator by the participant updates, $\mathcal{D}_t = \mathcal{D}_{t-1} + \frac{1}{N}\sum_{k=1}^{N} u_t^k$
  Train $\mathcal{D}_t$ by X_{aux} and X_{fake}, and Train $\mathcal{G}$
  if $t \geq d\_iter$ **then**
    for *k=1 to N* **do**
      Initialize participant classification model, $M_t^k = M_{t-1} + u_t^k$
      foreach *x in* X_{fake} **do**
        $L[k][x] = M_t^k(x)$
      end
    end
    Assign labels for X_{fake} based on L
    Calculate accuracy a^k of each participant classification model on X_{fake}
    Initialize the sum of benign updates $S = 0$ and the number of benign participants $NC = 0$
    for *k=1 to N* **do**
      if $a^k \geq \eta$ **then**
        $S = S + u_t^k$
        $NC = NC + 1$
      end
    end
    $M_t = M_{t-1} + \frac{S}{NC}$
    Sent M_t to all participants
  end
  else
    Federated learning averages updates to construct new global model and send the new global model to participants
  end
end

结构：

1. 假设辅助数据集中只包含一些类的数据，这些类数据被鉴别器用来实现联邦学习中的真假任务。
2. 使用全局模型作为鉴别器。

Table 1. Network structure of PDGAN

Discriminator	$32^2 \times 1 \xrightarrow{Conv\ (stride\ =\ 2),\ LeakyReLU,\ Dropout} 16^2 \times 64 \xrightarrow{Conv\ (stride\ =\ 2),\ BatchNorm,\ LeakyReLU} 8^2 \times 64 \xrightarrow{Conv\ (stride\ =\ 2),\ BatchNorm,\ LeakyReLU} 4^2 \times 64 \xrightarrow{Conv\ (stride\ =\ 1),\ BatchNorm,\ LeakyReLU} 2^2 \times 128 \xrightarrow{Conv\ (stride\ =\ 1),\ BatchNorm,\ LeakyReLU} 4^2 \times 128 \xrightarrow{Conv\ (stride\ =\ 1),\ LeakyReLU} 2^2 \times 128 \xrightarrow{AvgPool2d,\ FC,\ Softmax} 11$
Generator	$100 \xrightarrow{Deconv,\ BatchNorm,\ LeakyReLU} 4^2 \times 256 \xrightarrow{Deconv,\ BatchNorm,\ LeakyReLU} 8^2 \times 128 \xrightarrow{Deconv,\ BatchNorm,\ LeakyReLU} 32^2 \times 1 \xrightarrow{Tanh} 32^2 \times 1$

实验：

数据集：

MNIST、Fashion-MNIST

设备：

Intel Xeon W-213336GHZ CPU, Nvidia Quadro P5000 GPU with 16G RAM and RHIL7.5 OS

攻击实验设置：

10 个参与方

单攻击者模式：1 个攻击者，9 个良性参与方。

多攻击者模式：3 个攻击者，7 个良性参与方。

数据重建（GAN）：

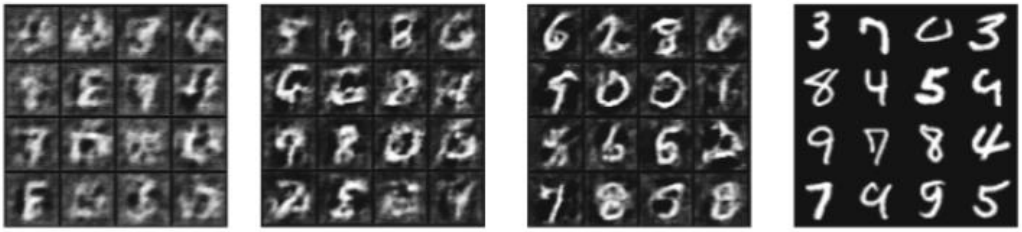
单攻击者下，服务器实现 GAN。

MNIST 数据集：图 4。

服务器中的辅助数据包括两类数据（数字 0 和数字 4），它们是提供给鉴别器的真实数据。 经过 400 次迭代后，生成的图像不会模糊，可以用作检测攻击者的审核数据。

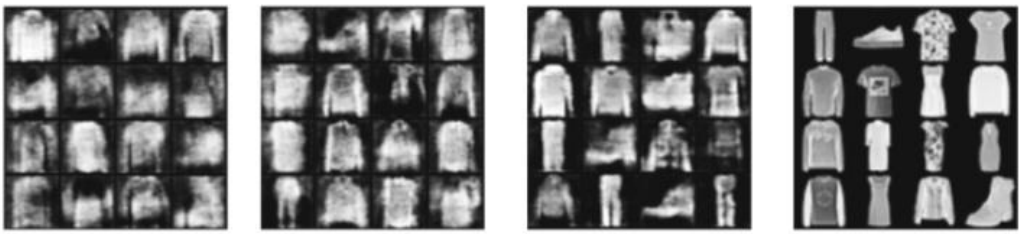
FASHION-MNIST 数据集：图 5。

辅助数据包括 3 个分类的数据（衣服，外套，凉鞋）。 由于 FASHION-MNIST 的图像比 MNIST 更复杂，因此需要更多的迭代来生成不模糊的图像。



(a) 100 iteration (b) 200 iteration (c) 400 iteration (d) real data

Fig. 4. MNIST reconstruction performance

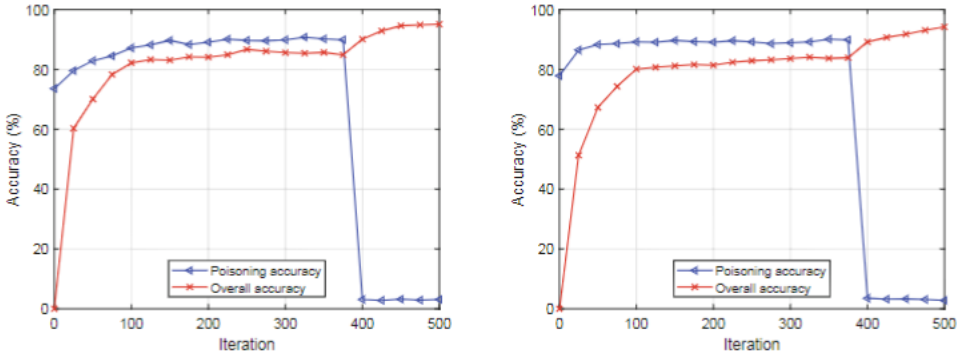


(a) 200 iteration (b) 400 iteration (c) 600 iteration (d) real data

Fig. 5. Fashion-MNIST reconstruction performance

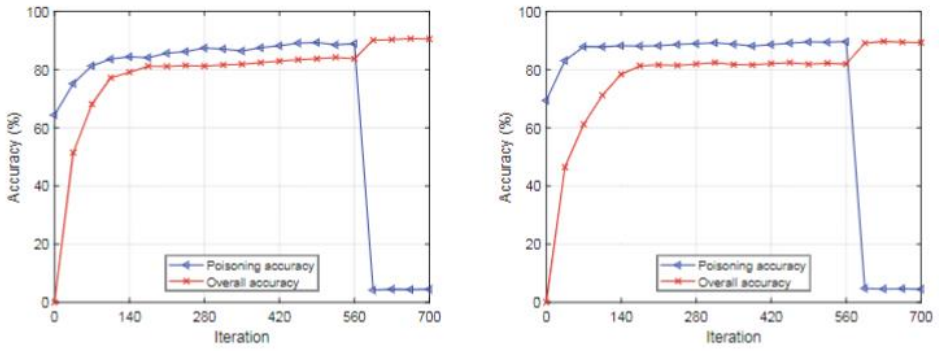
防御实验：

图 6 和图 7 分别表示在 MNIST 数据集和 Fashion-MNIST 数据集上的防御实验。在 MNIST 数据集上的防御实验中，PDGAN 在第 400 代时利用生成数据进行检测；在 Fashion-MNIST 数据集上的防御实验中，PDGAN 在第 600 代时利用生成数据进行检测。效果很显著。



(a) Single attacker (b) Three attackers

Fig. 6. Detection mechanism on MNIST dataset



(a) Single attacker (b) Three attackers

Fig. 7. Detection mechanism on Fashion-MNIST dataset

文章特点与分析：

1. 训练时检测并剔除毒害。
2. 利用 GAN 技术进行防御。
3. 服务器反向生成数据集，是否侵犯隐私？
4. GAN 生成可靠稳定的数据，需要迭代的代数太多了。