Sphinx: Enabling Privacy-Preserving Online Learning over the Cloud

分享在 Sphinx 论文中的一些思考、理解、推理、质疑和求证

吴亚伦

北京交通大学 · 网络空间安全

2023 年 03 月 24 日

机器学习与隐私保护

思考以下问题：（互动讨论，5-10 分钟）

- 1. 为什么研究机器学习的隐私保护？不做隐私保护会有什么危害吗？

机器学习与隐私保护

思考以下问题：（互动讨论，5-10 分钟）

- 1. 为什么研究机器学习的隐私保护？不做隐私保护会有什么危害吗？
- 2. 机器学习的隐私保护，保护对象是谁？

机器学习与隐私保护

思考以下问题：（互动讨论，5-10 分钟）

- 1. 为什么研究机器学习的隐私保护？不做隐私保护会有什么危害吗？
- 2. 机器学习的隐私保护，保护对象是谁？
- 3. 你平时进行机器学习实验的时候会进行隐私保护吗？(如果有，则讲述如何进行保护的？如果没有，阐明原因。)

MLaaS 的演变历程

从零开始搭建机器学习模型的 4 个步骤：

- 1. 首先需要选择一个服务器/工作站（考虑 CPU、显卡、主板、内存、硬盘）

MLaaS 的演变历程

从零开始搭建机器学习模型的 4 个步骤：

- 1. 首先需要选择一个服务器/工作站（考虑 CPU、显卡、主板、内存、硬盘）
- 2. 安装服务器的系统软件（安装 CentOS、Ubuntu 或者 Windows 等，俗称“装机”）

MLaaS 的演变历程

从零开始搭建机器学习模型的 4 个步骤：

- 1. 首先需要选择一个服务器/工作站（考虑 CPU、显卡、主板、内存、硬盘）
- 2. 安装服务器的系统软件（安装 CentOS、Ubuntu 或者 Windows 等，俗称“装机”）
- 3. 安装服务器的应用软件（安装 Anaconda、Python、CUDA、PyTorch 等，俗称“搭环境”）

MLaaS 的演变历程

从零开始搭建机器学习模型的 4 个步骤：

- 1. 首先需要选择一个服务器/工作站（考虑 CPU、显卡、主板、内存、硬盘）
- 2. 安装服务器的系统软件（安装 CentOS、Ubuntu 或者 Windows 等，俗称“装机”）
- 3. 安装服务器的应用软件（安装 Anaconda、Python、CUDA、PyTorch 等，俗称“搭环境”）
- 4. 编写或复现机器学习模型代码，运行。

MLaaS 的演变历程

第一阶段：基本方法/传统方法

- 方案介绍：按部就班的四步走。
- 操作步骤：
  - 1. 买服务器/工作站（买买买）。
  - 2. 安装系统软件（“装机”）
  - 3. 安装应用软件（“搭环境”）
  - 4. 编写或复现机器学习代码，运行。
- 存在问题：
  - 1. 成本高，需要一次性购置设备（目前一台低配的工作站约需 3 5 万）
  - 2. 部署慢，需要装机装系统、还需要安装基础的应用软件，很繁琐。

MLaaS 的演变历程

第二阶段：基础设施即服务（Infrastructure as a Service, IaaS）

- 方案介绍：商家提供云服务器和系统软件（Ubuntu/CentOS 等）基础设施的“租赁”服务，按时长收费。
- 提出背景：
  - 传统方法购买设备成本高 =>IaaS 从“买”变“租”，降低短期成本。
  - 传统方法系统软件部署慢 =>IaaS 提供服务，支持任意系统一键安装。
- 典型产品：阿里云、腾讯云（提供云主机、云存储）
- 存在问题：
  - 尽管解决了设备购置成本高、系统软件部署慢的问题，但是应用软件（Anaconda、Python、CUDA、PyTorch 等）的部署，也是一个很繁琐的过程。

MLaaS 的演变历程

第三阶段：平台即服务（Platform as a Service, PaaS）

- 方案介绍：商家除了提供云服务器和系统软件（Ubuntu、CentOS 等）之外，还提供 Anaconda、CUDA、PyTorch 等应用环境的安装。
- 提出背景：
  - Iaas 应用软件部署慢 =>PaaS 支持自定义软件镜像，一键安装开发环境。
- 典型产品：Google Colab、AutoDL（提供算力租赁服务）
- 存在问题：
  - 尽管具有成本低、系统软件部署快、应用软件部署的特点，但美中不足的是：机器学习模型复现遇到了困难（对于非专业人士很不友好，考虑“能不能不写代码就能用？”）。

MLaaS 的演变历程

第四阶段：软件即服务（Software as a Service, SaaS）

- 方案介绍：商家直接提供全套服务：服务器到位了、系统搭建了、环境配好了、模型搞定了，用户直接使用就可以了。
- 提出背景：
  - PaaS 模型复现困难 =>SaaS 直接提供全套服务。
- 典型产品：谷歌云机器学习引擎、Amazon ML
- 存在问题：虽然已经尽最大可能方便了用户，但还是觉得不够完美：面对众多模型，商家如何一一提供？

MLaaS 的演变历程

第五阶段：机器学习即服务（Machine Learning as a Service, MLaaS）

- 个人理解：首先 MLaaS 是 SaaS 在机器学习领域的一种应用；其次 MLaaS 对 SaaS 进行改进了，引入了第三方（除了商家和用户之外的第三方）。这有点像消费者、京东自营和京东非自营三者之间的关系。
- 典型产品：谷歌云机器学习引擎、Amazon ML

MLaaS 的演变历程

传统方法->IaaS->PaaS->SaaS->MLaaS 的演变过程

优点:

- 部署速度越来越快；成本越来越低。

缺点:

- 一方（指：自己）-> 两方（指：云服务商和自己）-> 三方（指：云服务商、第三方模型提供者和自己）：隐私安全性越来越低，因此：隐私保护问题不得不被考虑。

论文基本信息

2022 IEEE Symposium on Security and Privacy (SP)

Sphinx: Enabling Privacy-Preserving Online Learning over the Cloud

Han Tian[1] Chaoliang Zeng[1] Zhenghang Ren[1] Di Chai[1,2] Junxue Zhang[1,2] Kai Chen[1] Qiang Yang[1]
[1]Hong Kong University of Science and Technology [2]Clustar

- 题目：Sphinx: Enabling Privacy-Preserving Online Learning over the Cloud（Sphinx: 实现云端保护隐私的在线学习）
- 会议：2022 IEEE Symposium on Security and Privacy (SP)
- 团队：香港科技大学团队 & 星云（Clustar）
- 定位：是一篇使用同态加密（CKKS）和差分隐私进行隐私保护的文章（偏应用）。

论文基本信息

学术团队/核心作者:

- 星云 (Clustar): 深圳致星科技有限公司，注于高性能通用 AI 算力、隐私计算算力、网络算力的研发与场景应用，官网地址: https://www.clustarai.com/
- 陈凯: 星云创始人、香港科大-微信人工智能技术联合实验室任、智能网络与系统实验室主任; 研究方向: 大数据和人工智能底层系统、隐私计算等。
- 张骏雪: 星云联合创始人 &CTO、香港科大博士生; 研究方向: 数据挖掘、联邦学习、网络安全等。

论文内容与贡献

- 提出了 Sphinx，第一个考虑训练和推理两阶段的一个高效和隐私保护的在线深度学习系统。

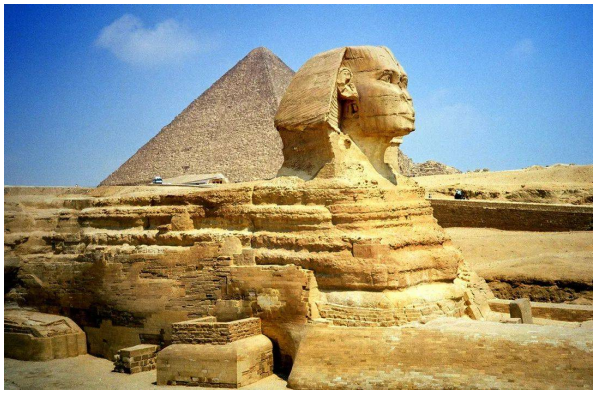
论文内容与贡献

- 提出了 Sphinx，第一个考虑训练和推理两阶段的一个高效和隐私保护的在线深度学习系统。
- 将神经网络中的线性层分成了两部分，分别使用了不同的加密协议，并从理论和实验的角度进行了验证。(使用同态加密加密所有的偏置分量 b、使用差分隐私扰乱线性分量 ω)。

论文内容与贡献

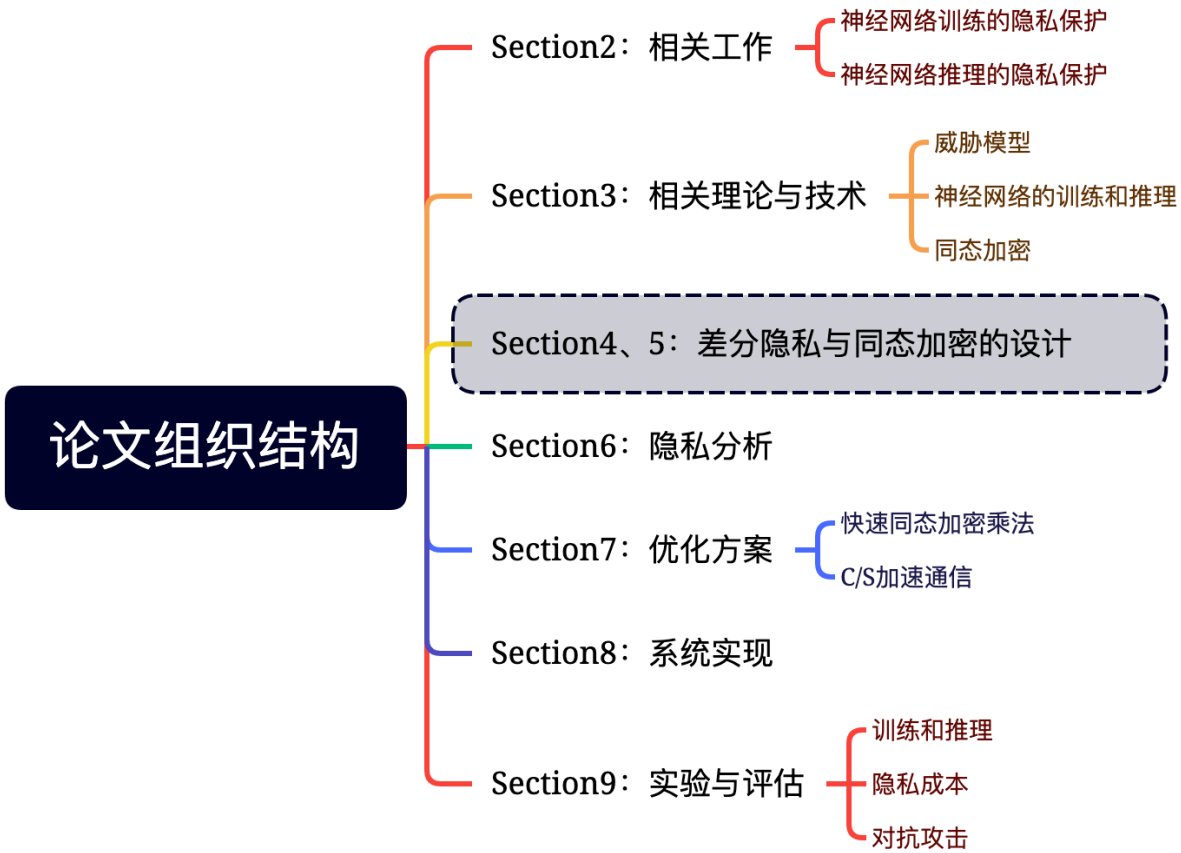
- 提出了 Sphinx，第一个考虑训练和推理两阶段的一个高效和隐私保护的在线深度学习系统。
- 将神经网络中的线性层分成了两部分，分别使用了不同的加密协议，并从理论和实验的角度进行了验证。(使用同态加密加密所有的偏置分量 b、使用差分隐私扰乱线性分量 ω)。
- 巧妙运用了 CKKS 的一些特性，加速了训练过程；在推理阶段引入了预处理过程，加速了推理效率、减小了通信开销。

小插曲：为什么选择了这篇文章？



- 很多论文都是 A:B 的结构，比如：“DLFuzz: Differential Fuzzing Testing of Deep Learning Systems”、“TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing”、“Deepsafe: A data-driven approach for checking adversarial robustness in neural networks”，被论文的题目吸引了。
- 推理：之所以起名叫“Sphinx”，是因为本论文中神经网络中的偏置单元 b 和线性单元 ω 用了不同的加密协议（有“狮身人面”内味了）。

论文组织结构



同态加密与差分隐私技术

同态加密

背景：假设我有两个数 m_1 和 m_2，我希望把它们扔给云计算平台做加法运算，但又不希望云计算平台知道 m_1、m_2 的值。设 $m_1 = 100$, $m_2 = 200$，步骤如下：

- （本地）生成一对钥匙，公钥 pub 和密钥 priv，公钥用于加密，密钥用于解密。
- （本地）公钥 pub 分加密 m_1 和 m_2，如：$E_{pub}(m_1) = 1234$，$E_{pub}(m_2) = 4321$
- （云计算平台）使用 Add_{pub} 函数对 $E_{pub}(m_1)$ 和 $E_{pub}(m_2)$ 进行计算，即：$Add_{pub}(1234, 4321) = 12345678$
- （本地）使用密钥 priv 解密 $Add_{pub}(1234, 4321)$，得到 $D_{priv}(12345678) = 300$（严格意义上讲，是个近似值）。

同态加密与差分隐私技术

差分隐私

差分隐私是属于密码学的远亲，并没有使用传统意义上的加密解密，它是通过一种扰动的方法，对原始数据进行保护。(下面举一个简单例子，如有不当请指出)

- 假设有一个婚恋网站，系统只能够查询单身人数（有约束条件），攻击者的任务也是查找单身人数。
- 现在进行一次查询，发现系统有 2 人单身；然后张三跑去登记了，又进行一次查询发现系统显示 3 人单身。我们就可以推断出“张三单身”。在这里，张三作为一个样本的出现，让攻击者获得了奇怪的知识。而差分隐私要做的就是使攻击者获得的知识不会因为新样本的出现而发生变化。
- 简单的差分隐私操作：添加噪声，第一次查询的时候显示 2.5，第二次显示 2.5，这样攻击者便不能判断张三是不是单身。

同态加密与差分隐私技术

同态加密与差分隐私技术对比

表 1: 同态加密与差分隐私技术对比

隐私保护方案	同态加密	差分隐私
隐私保护性	强	弱
计算效率	慢 (消耗计算资源)	快

- 差分隐私：大多数是在模型中（或输出结果中）添加噪声，隐私保护能力不如同态加密；但是效率特别高，在隐私保护性要求不高的情况下用的比较多。
- 同态加密：效率比较低，非常消耗计算资源，但是隐私保护能力非常强。

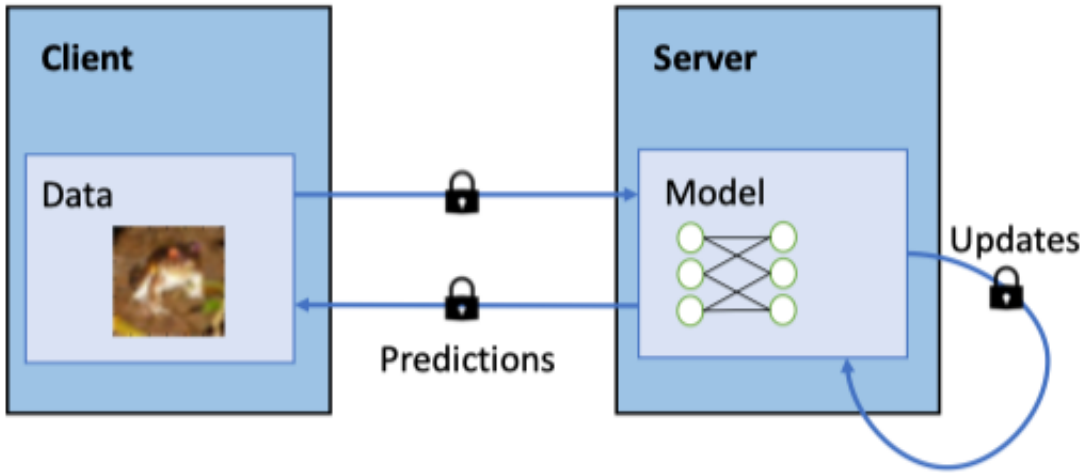
现有的隐私保护方案

本文对现有的隐私保护方案进行归纳，可分为以下三种情况：

- 1）专注于推理阶段的隐私保护。例如：[LJLA17](CCS'17)，MLaaS 场景下用户要保护数据，服务商要保护模型。作者提出了一个两方计算框架 MiniONN，将已经训练好的模型转化为一个不经意传输模型实现了数据和模型的保护。
- 2）专注于训练阶段的隐私保护。例如：[NRPH19](CVPR'19)，作者评估了非交互式方式在加密数据上训练 DNN 的可能性，通过简化网络、选择图像合适的分辨率等技巧，提高了加密训练的效率。
- 3）甚至是假设存在一个可信的中心或者多个非共谋的服务器。例如：[MZ17](SP'17)，作者设计了 2 个半可信非共谋的服务器，交互执行密文下的计算。(作者认为这个假设是不现实的)。

现有的隐私保护方案

因此，作者提出了 Sphinx：



- 取消了他认为不现实的假设（作者认为：存在可信中心或者多个非共谋的服务器是不现实的），在本文中只有 2 方：用户（客户端）和服务商（服务端），没有可信的中心。
- 训练阶段的隐私保护、推理阶段的隐私保护都要兼顾。

训练协议设计

设计思路：

- 差分隐私通过添加噪声，在一定程度上可以起到保护隐私的作用（与纯明文比，是能保护隐私的）。
- 由于深度神经网络中涉及了大量的矩阵乘法运算，前人已有论证：完全使用同态加密算法进行计算，虽然隐私性强，但效率特别低。
- 而 CKKS（本文使用的同态加密算法）支持明文与密文之间的混合操作，且要比密文与密文之间的运算耗费的计算资源少，效率要高。（求证：前面所述是在一篇技术分享里看到的一段话，待求证。）
- 因此，作者对隐私性做了“妥协”，不完全使用同态加密了，而是部分使用同态加密（偏置分量 b），部分使用差分隐私（线性分量 ω）。
- 思考：是否可以对线性分量 ω 进行同态加密，对偏置分量进行差分隐私？同态加密的加法和乘法，哪个效率更高？

训练协议设计

设计概览：

本文假设神经网络模型是简单的分层结构，每层的公式为：

$$f(x) = \sigma(W^T x + b)$$

其中，σ 为非线性激活函数（以及池化操作），W 为权重向量，b 为偏置。Sphinx 使用同态加密对偏置 b 进行加密，使用差分隐私对权重 W 的计算进行扰动。协议的设计考虑了性能问题，此处如果完全使用同态加密，那计算速度会大打折扣。所以对于权重的计算仅使用了差分隐私的方案（明文）。

热知识：非线性的激活函数可以提高模型的复杂度，但是同态加密无法处理非线性计算，因此非线性部分 f 是在用户端完成的。

训练协议设计

训练协议设计：初始化

- 用户：首先生成公钥私钥对 PK 和 SK，并且将公钥 PK 以及初始模型结构发送给服务器；服务器：初始化模型参数 W^0 和 $[b^0]$。

训练协议设计：训练过程

- 用户：采样数据，将样本加密得到 $[x_i]$，然后将加密后的数据发送给服务器；服务器：服务器进行线性计算 ${W_i}^T[x_i]+[b_i]$ 得到一个中间的密文，即为 $[c_i]$，然后服务器将 $[c_i]$ 发送给用户端进行解密；用户收到 $[c_i]$ 后，进行如下计算：1）解密；2）非线性计算 f；3）加密得到 $[x_{i+1}]$；

Algorithm 1: Sphinx Training Algorithm

Input : *Client*: Training dataset X with size N, noise scale σ, gradient norm bound C; *Server*: Model M with K layers, loss function $\mathcal{L}(\theta)=\frac{1}{N}\sum_j \mathcal{L}(\theta,\mathbf{x_j})$, learning rate η_t, batch size B

Output: Trained model parameters $\theta^T=\{\mathbf{W}^T,[\mathbf{b}^T]\}$

The client creates an encryption key pair (PK, SK) and send the public key to the server;

The server initializes the plaintext linear components $\mathbf{W}^0$ and the encrypted bias components $[\mathbf{b}^0]$ with PK;

for $t \leftarrow 0$ **to** $T-1$ **do**

Client:

Takes a random sample batch $\mathcal{B}=\{\mathbf{x_1},\mathbf{x_2},...\mathbf{x_B}\}$ with sampling probability B/N, encrypt them with PK and send the encrypted batch to the server;

Server:

Computes the gradients for the linear components $[\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})]$ and bias components $[\nabla_{\mathbf{b}}\mathcal{L}(\mathbf{x_j})]$ according to Equation 3,4 for each sample $\mathbf{x_j}$;

Computes the average gradients of bias components: $[\overline{\mathbf{g}}_{\mathbf{b}}] \leftarrow \frac{1}{B}(\sum_j[\nabla_{\mathbf{b}}\mathcal{L}(\mathbf{x_j})])$;

Updates the bias components: $[\mathbf{b}^{t+1}] \leftarrow [\mathbf{b}^t]-\eta_t[\overline{\mathbf{g}}_{\mathbf{b}}]$;

Sends $[\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})]$ back to the client;

Client:

Decrypts $[\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})]$ with SK;

Clips the gradient: $\nabla_{\mathbf{W}}\widetilde{\mathcal{L}}(\mathbf{x_j}) \leftarrow \max(1, \frac{\|\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})\|_2}{C})$;

Computes the average gradients of linear components: $\overline{\mathbf{g}}_{\mathbf{W}} \leftarrow \frac{1}{B}(\sum_j \nabla_{\mathbf{W}}\widetilde{\mathcal{L}}(\mathbf{x_j}))$;

Adds noise: $\widetilde{\mathbf{g}}_{\mathbf{W}} \leftarrow \overline{\mathbf{g}}_{\mathbf{W}}+N(0,\frac{\sigma^2C^2}{B^2}\mathbf{I})$;

Sends $\widetilde{\mathbf{g}}_{\mathbf{W}}$ back to the server;

Server:

Update the linear components: $\mathbf{W}^{t+1} \leftarrow \mathbf{W}^t-\eta_t\widetilde{\mathbf{g}}_{\mathbf{W}}$;

训练协议设计

训练协议设计：训练过程

- 疑问：对输入样本 $[x_i]$ 的加密是用的什么算法？
- 优化：反向传播时，需要用到 f 的一阶导数。为了减少通信量，在进行非线性计算时便计算了激活函数 f 的导数，跟数据一起发送给服务器并缓存起来。

Algorithm 1: Sphinx Training Algorithm

Input : *Client*: Training dataset X with size N, noise scale σ, gradient norm bound C; *Server*: Model M with K layers, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_j \mathcal{L}(\theta, \mathbf{x_j})$, learning rate η_t, batch size B

Output: Trained model parameters $\theta^T = \{\mathbf{W}^T, [\mathbf{b}^T]\}$

The client creates an encryption key pair (PK, SK) and send the public key to the server;

The server initializes the plaintext linear components $\mathbf{W}^0$ and the encrypted bias components $[\mathbf{b}^0]$ with PK;

for $t \leftarrow 0$ **to** $T-1$ **do**

  Client:

  Takes a random sample batch $\mathcal{B} = \{\mathbf{x_1}, \mathbf{x_2}, ...\mathbf{x_B}\}$ with sampling probability B/N, encrypt them with PK and send the encrypted batch to the server;

  Server:

  Computes the gradients for the linear components $[\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})]$ and bias components $[\nabla_{\mathbf{b}}\mathcal{L}(\mathbf{x_j})]$ according to Equation 3,4 for each sample $\mathbf{x_j}$;

  Computes the average gradients of bias components: $[\bar{\mathbf{g}}_{\mathbf{b}}] \leftarrow \frac{1}{B}(\sum_j [\nabla_{\mathbf{b}}\mathcal{L}(\mathbf{x_j})])$;

  Updates the bias components: $[\mathbf{b}^{t+1}] \leftarrow [\mathbf{b}^t] - \eta_t[\bar{\mathbf{g}}_{\mathbf{b}}]$;

  Sends $[\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})]$ back to the client;

  Client:

  Decrypts $[\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})]$ with SK;

  Clips the gradient: $\nabla_{\mathbf{W}}\tilde{\mathcal{L}}(\mathbf{x_j}) \leftarrow \max(1, \frac{\|\nabla_{\mathbf{W}}\mathcal{L}(\mathbf{x_j})\|_2}{C})$;

  Computes the average gradients of linear components: $\bar{\mathbf{g}}_{\mathbf{W}} \leftarrow \frac{1}{B}(\sum_j \nabla_{\mathbf{W}}\tilde{\mathcal{L}}(\mathbf{x_j}))$;

  Adds noise: $\tilde{\mathbf{g}}_{\mathbf{W}} \leftarrow \bar{\mathbf{g}}_{\mathbf{W}} + N(0, \frac{\sigma^2 C^2}{B^2}\mathbf{I})$;

  Sends $\tilde{\mathbf{g}}_{\mathbf{W}}$ back to the server;

  Server:

  Update the linear components: $\mathbf{W}^{t+1} \leftarrow \mathbf{W}^t - \eta_t\tilde{\mathbf{g}}_{\mathbf{W}}$;

训练协议设计：更新过程

- 偏置单元 b：服务器中本身拥有的就是加密版本（同态加密），直接更新到模型中。
- 线性单元 W：服务器将线性单元的梯度发送给用户，用户解密后添加噪声再给服务器，服务器更新模型（差分隐私）。

推理协议设计

推理协议设计：单个共享密钥（P_i）生成过程

思想：推理阶段，不再用复杂的同态加密、解密操作。

- 用户：对每一层神经网络的输入生成一个 mask 向量 r_i，服务器：为每一层神经网络的输出生成一个 mask 向量 s_i，其中 i 是第 i 个线性层。
- 用户：使用同态加密的方式对 r_i 进行加密，发送 $[r_i]$ 给服务器。
- 服务器：把加密后的 $[r_i]$ 当做一个样本，计算 $W_i^T \cdot [r_i] + [b_i] - s_i \longrightarrow [W_i^T \cdot r_i + b_i - s_i]$，并将结果发送给用户。
- 用户：解密得到：$P_i = W_i^T \cdot r_i + b_i - s_i$。
- 疑问：没明白 mask 向量 r_i 和 s_i 是怎么得到的。

推理协议设计

推理协议设计：样本 X 推理过程

- 用户：预先挑选一个计算好的 mask 向量 r_i，计算 $x - r_i$，并将 $x - r_i$ 发送给服务器。
- 服务器：计算 $W_i^T \cdot (x - r_i) + s_i$，并将结果发送给用户。
- 用户：将计算的结果 $W_i^T \cdot (x - r_i) + s_i$ 与之前的 $P_i = W_i^T \cdot r_i + b_i - s_i$ 相加，得到 $W_i^T \cdot x + b_i$
- 部分参数抵消，直接获得推理的结果，r_i 起到了保护样本 x 的作用，但不涉及加密解密操作。

实验环境设置

表 2: 实验环境设置

实验环境	具体配置及参数
物理机数量	2 台（客户端和服务端）
物理机 CPU	至强 E5-2683CPU，128G 内存
操作系统	Ubuntu18.04LTS
网络环境	通过 10Gbps 局域网连接
编程语言	C++、SEAL3.6（同态加密库）

- 科普：10Gbps，万兆带宽，大约每秒能传输 1G 多的数据。
- 疑问：未提到 GPU，是 GPU 不支持还是忽略了 GPU？(不同算力效果是不一样的。)
- 质疑：1）目前广域网只普及到千兆级别，万兆尚未普及。2）若 MLaaS 部署在局域网，那就“不需要”考虑隐私保护了。

实验环境设置

表 3: 模型参数设置

模型参数	MNIST 模型	CIFAR-10 模型
batch size	500	2000
优化算法	随机梯度下降 (SGD)	随机梯度下降 (SGD)
学习率	0.1	0.05

- MNIST 数据集（0-9 共 10 个分类），主要用于分类任务；CIFAR-10（共 10 个分类），也主要用于分类任务，与 MNIST 数据集不同的是，CIFAR-10 数据集是彩色图像。
- 实验结果：作者在 MNIST 和 CIFAR-10 数据集上得到的最佳准确率（Accuracy）分别为 98.5% 和 78.8%。

神经网络结构

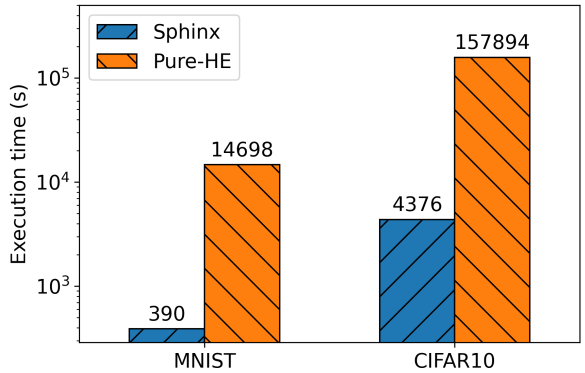
1) *Input*: $\mathbb{R}^{28\times28}$
2) *Convolution*: window size 5×5 with stride (1,1), number of channels 16, no pad, with bias. Outputs: $\mathbb{R}^{16\times24\times24}$
3) *ReLU*: ReLU non-linear function. Outputs: $\mathbb{R}^{16\times24\times24}$
4) *Average Pooling*: window size 1×2×2, stride (2,2). Outputs: $\mathbb{R}^{16\times12\times12}$
5) *Convolution*: window size 5×5 with stride (1,1), number of channels 16, no pad, with bias. Outputs: $\mathbb{R}^{16\times8\times8}$
6) *ReLU*: ReLU non-linear function. Outputs: $\mathbb{R}^{16\times8\times8}$
7) *Average Pooling*: window size 1×2×2, stride (2,2). Outputs: $\mathbb{R}^{16\times4\times4}$
8) *Fully Connected*: hidden neuron number 100. Outputs: $\mathbb{R}^{100\times1}$
9) *ReLU*: ReLU non-linear function. Outputs: $\mathbb{R}^{100\times1}$
10) *Fully Connected*: hidden neuron number 10. Outputs: $\mathbb{R}^{10\times1}$

- 卷积核的通道数 = 输入的通道数
- 卷积核的数量 = 输出的通道数

模型训练实验分析

1. 训练时间对比分析

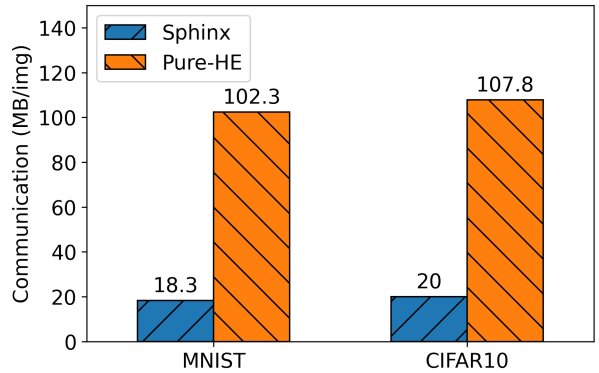
- 右图为 Sphinx 和纯同态加密算法 (Pure-HE) 在 1 个 batch 上训练时间的比较，从图中可以看出，Sphinx 比 Pure-HE 的训练时间减少了 35 倍（在 MNIST 数据集上的训练时间减少了 36.68 倍，在 CIFAR-10 上减少了 35.08 倍）。
- 注：Pure-HE 整个模型都是加密的，所有的同态运算发生在密文与密文之间。而 Sphinx 由于只对偏置单元 b 进行了加密，所以是密文与明文之间的同态运算。



模型训练实验分析

2. 通信开销对比分析

- 右图为 Sphinx 和纯同态加密算法 (Pure-HE) 在 1 个 batch 上通信开销的比较，从图中可以看出，Sphinx 比 Pure-HE 的通信开销减少了 5 倍。
- 原因分析：Sphinx 通过保护线性分量为明文，避免了大多数繁重的同态操作（如：密文与密文矩阵乘法、缩放和重线性化）
- 疑问：为什么训练时间和通信开销的对比分析实验中使用的是 batch 而不是 epoch？(引发部署可行性的思考)【注意：接下来本人要闹笑话了！】



部署可行性的思考

相关名词

- epoch：训练的轮数，1 epoch 指所有样本都经过了 1 次训练；batch：参数更新是分批进行的，1 批就是 1 个 batch，1 批训练完称为 1 iteration(迭代)；batch-size：每个 batch 中样本数量。
- 例如：MNIST 训练集中有 60000 张图片，作者设置的 batch size 为 500。指 60000 张图片分成了 120 批 (batch=120)，每批有 500 张 (batch size=500)，训练完 1 批称为 1 次迭代 (iteration+1)。

可行性思考

- 如果 390s 是指的“1 epoch 的时间”，推论：勉强能用（i7CPU+3060 单卡 + 本机 +Resnet50 模型 1 个 epoch 约 370s)；如果 390s 是指的“1 iteration 的时间”，推论：无法实际部署（上述低配版 3.12s）

模型推理实验分析

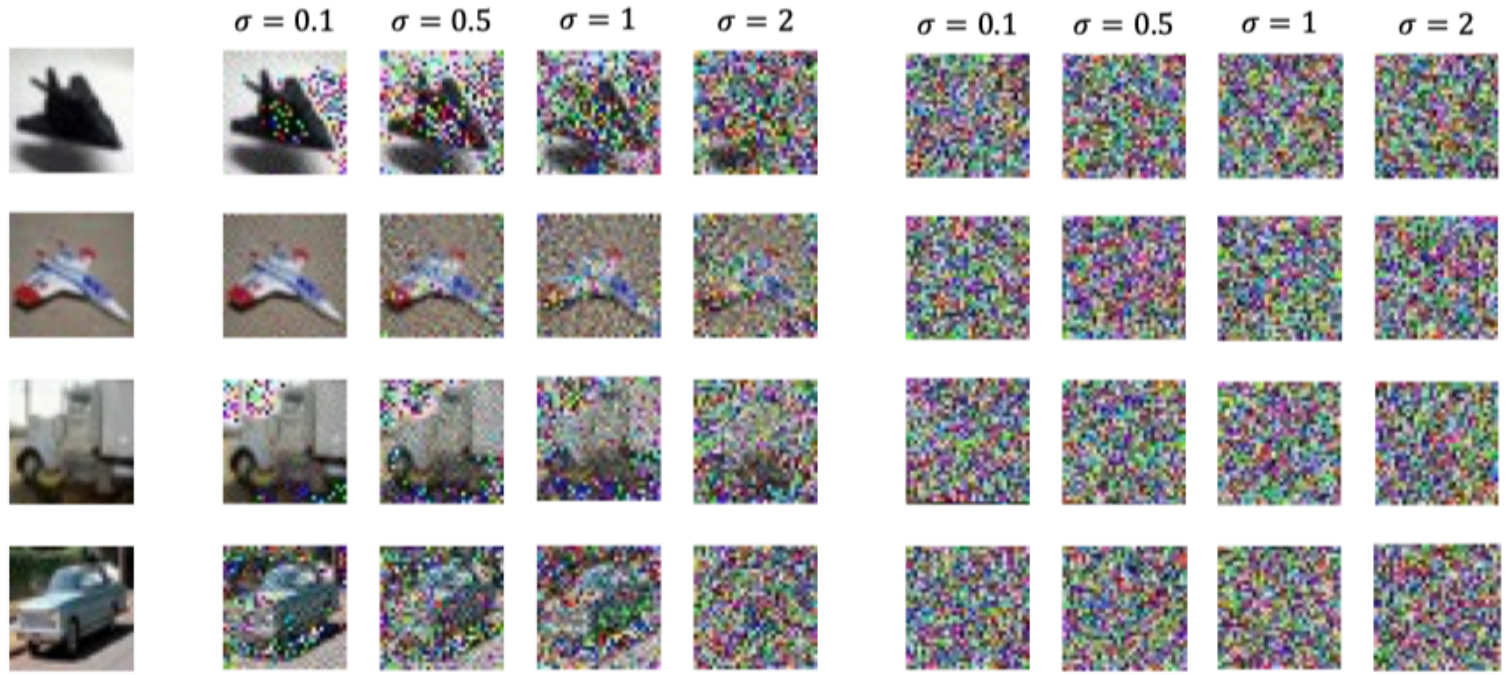
右图为单个样本的推理延迟、通信开销，结论如下：

- 与 Pure-HE 相比 Sphinx-FP 推理延迟减少了近 30 倍，通信成本减少了近 4 倍，但由于同态操作，推理延迟和通信成本仍然很高。
- 然而，Sphinx 与 Pure-HE 比，实现了更低的推理延迟（$5.8 \cdot 10^4$ 和 $4.2 \cdot 10^5$ 倍）和通信开销（$1.9 \cdot 10^5$ 和 $3.6 \cdot 10^4$），可以与最先进的方法 Delphi 媲美。
- 但 Delphi 和 Gazelle 等私有推理协议需要公共模型，因此不适用于在线学习。

	Framework	Time (s)		Communication (MB)	
		preproc.	online	preproc.	online
MNIST	Pure-HE	-	2915	-	13134
	Sphinx-FP	14.4	108	-	3014
	Gazelle	12.46	1.44	566.2	127.3
	Delphi	6.34	0.09	91.8	0.42
	Sphinx	6.01	**0.05**	87.6	**0.07**
CIFAR-10	Pure-HE	-	33909	-	53048
	Sphinx-FP	130	1065	-	13022
	Gazelle	42.65	4.82	1907	623
	Delphi	48.9	0.18	145	5.45
	Sphinx	48.3	**0.08**	128	**1.46**

TABLE V: The benchmarks of various inference protocols. Sphinx-FP prepares *zero ciphertexts* in the preprocessing phase. Delphi requires a plain public model on the server, thus cannot be used for privacy-preserving online learning.

对抗攻击分析



(a) Original (b) All noisy (c) Sphinx

为了评估 Sphinx 在实际场景中对已知攻击的防御效果，作者进行了梯度匹配攻击（梯度匹配攻击 [ZMB20] 是一种利用模型的梯度恢复输入图像和标签的攻击方法）。实验结果表明：Sphinx 模型防御梯度匹配攻击能力更强。

总结

本文的优点

- 取消了不现实的假设，同时考虑了训练、推理两阶段的隐私保护问题（数据隐私和模型参数隐私）。

本文的缺点

总结

本文的优点

- 取消了不现实的假设，同时考虑了训练、推理两阶段的隐私保护问题（数据隐私和模型参数隐私）。
- 题目吸引读者，用 Sphinx 比喻了算法设计方案（此条为本人的推断）。

本文的缺点

总结

本文的优点

- 取消了不现实的假设，同时考虑了训练、推理两阶段的隐私保护问题（数据隐私和模型参数隐私）。
- 题目吸引读者，用 Sphinx 比喻了算法设计方案（此条为本人的推断）。
- 使用了 CKKS 的一些特性，对原始模型进行了优化，提高了计算效率，减小了通信开销；mask 向量设计的很巧妙（但本人脑子懂了手没学会）。

本文的缺点

总结

本文的优点

- 取消了不现实的假设，同时考虑了训练、推理两阶段的隐私保护问题（数据隐私和模型参数隐私）。
- 题目吸引读者，用 Sphinx 比喻了算法设计方案（此条为本人的推断）。
- 使用了 CKKS 的一些特性，对原始模型进行了优化，提高了计算效率，减小了通信开销；mask 向量设计的很巧妙（但本人脑子懂了手没学会）。

本文的缺点

- 文章的组织结构不是很喜欢，导致阅读困难。

总结

本文的优点

- 取消了不现实的假设，同时考虑了训练、推理两阶段的隐私保护问题（数据隐私和模型参数隐私）。
- 题目吸引读者，用 Sphinx 比喻了算法设计方案（此条为本人的推断）。
- 使用了 CKKS 的一些特性，对原始模型进行了优化，提高了计算效率，减小了通信开销；mask 向量设计的很巧妙（但本人脑子懂了手没学会）。

本文的缺点

- 文章的组织结构不是很喜欢，导致阅读困难。
- 文章中部分涉及深度学习的知识出现码字错误。

总结

本文的优点

- 取消了不现实的假设，同时考虑了训练、推理两阶段的隐私保护问题（数据隐私和模型参数隐私）。
- 题目吸引读者，用 Sphinx 比喻了算法设计方案（此条为本人的推断）。
- 使用了 CKKS 的一些特性，对原始模型进行了优化，提高了计算效率，减小了通信开销；mask 向量设计的很巧妙（但本人脑子懂了手没学会）。

本文的缺点

- 文章的组织结构不是很喜欢，导致阅读困难。
- 文章中部分涉及深度学习的知识出现码字错误。
- 实验不太严谨，训练过程的实验环境与隐私保护场景相违背。

参考文献列表 I

[LJLA17] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations.
In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 619–631, 2017.

[MZ17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning.
In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

参考文献列表 II

[NRPH19] Karthik Nandakumar, Nalini Ratha, Sharath Pankanti, and Shai Halevi.
Towards deep neural network training on encrypted data.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[ZMB20] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen.
idlg: Improved deep leakage from gradients.
arXiv preprint arXiv:2001.02610, 2020.

感谢各位的聆听