Pencil: Private and Extensible Collaborative Learning without the Non-Colluding Assumption

此次介绍的是清华大学Xuanqi Liu等人发表在NDSS24的论文关于隐私保护机器学习训练的论文：
https://www.ndss-symposium.org/ndss-paper/pencil-private-and-extensible-collaborative-learning-without-the-non-colluding-assumption/

开源代码如下：
https://github.com/lightbulb128/Pencil

1. 背景与动机

目前针对隐私保护机器学习训练的工作可以分为以下几类：联邦学习，基于安全多方计算的方案，和基于同态加密的方案。不同的方案在效率、安全性、和扩展性等反面各种不同。如下表所以：1）联邦学习只关注原始数据的隐私，却忽视了每一轮迭代中的模型隐私。2）安全多方计算一般适用的场景是外包计算，即数据提供方将自己的数据以秘密分享的形式分给多个外包服务器，外包服务器调用安全多方计算协议实现模型训练。但是这需要额外的安全假设要求，即外包服务器不能合谋。如果数据提供方作为安全计算节点执行安全计算协议，那么则需要抵抗$n-1$方合谋的安全协议（例如SPDZ），系统的效率会大大降低。3）基于同态加密的方案因为公私钥的绑定，目前一般只能适用一个数据拥有方。

Category	Representative framework	Techniques used*	Data privacy	Model privacy	Against collusion	Extensibility
Horizontal FL	[39], [10], [9]	Local SGD	✓	×	✓	✓
Vertical FL	[21], [19], [28]	Local SGD	✓	×	✓	✓
MPC (2 servers)	[3], [42]	GC, SS	✓	✓	×	✓†
MPC (3 servers)	[41], [47], [59]	GC, SS	✓	✓	×	✓†
MPC (4 servers)	[11], [33], [14]	GC, SS	✓	✓	×	✓†
MPC (n servers)	[15], [13]	GC, SS	✓	✓	✓	✓‡
Data outsourcing / cloud	[43], [24]	HE	✓	×	N/A	×
Data outsourcing / cloud	[56]	HE, DP	✓	×	N/A	×
Pencil	Ours	HE, SS, DP	✓	✓	✓	✓

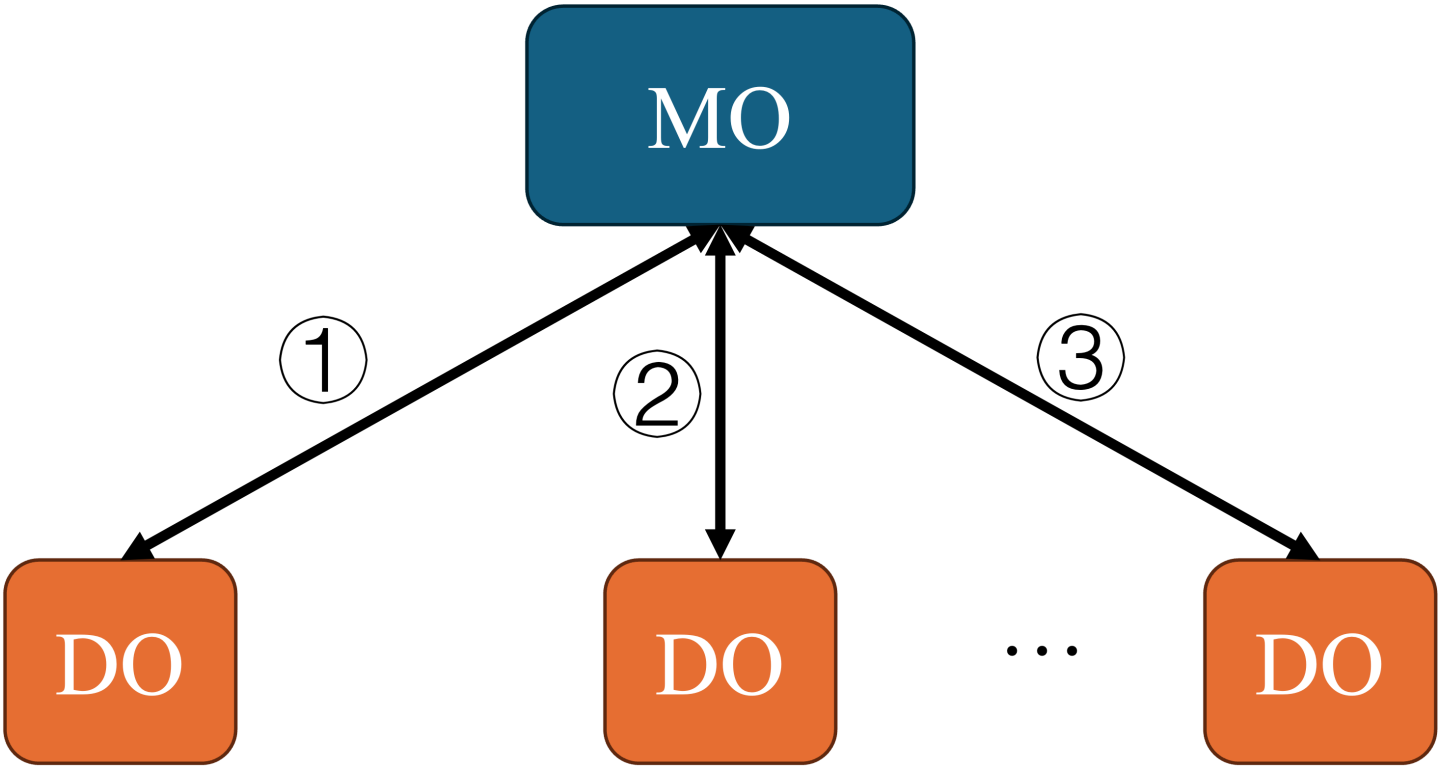
* SGD is for stochastic gradient descent, GC for garbled circuits, SS for secret sharing, HE for homomorphic encryption and DP for differential privacy.
† If MO and DOes choose to secretly share their model and data to third-party MPC servers, extensibility is achieved but the approaches are secure only if these servers are not colluding with each other.
‡ The general n-PC protocol against collusion suffers from a scalability problem: including more parties would greatly increase the computation overhead. See § VI-E for experimental results.

TABLE I: Comparison of prior art related with private collaborative training.

和上述方案不同，Pencil面向的场景是一个模型提供方-多个数据提供方的训练场景，最终训练的模型只公开给模型提供方进行后续工作。因此，上述中的FL无法满足本文的要求（FL中每次迭代训练需要将模型公开给所有数据提供方）。同时，本文试图得到比上述MPC方案更优秀的效率和扩展性，可以很自然的拓展到多个数据提供方；同时，又可以不需要计算服务器不合谋的假设。为了解决这个问题，本文在安全两方计算的基础上提出了框架Pencil。

本文需要基于格的同态加密方案和两方秘密分享，之前已经介绍过相关知识，在此不做赘述。

2. 系统架构



如上图所示，Pencil系统中有一个模型拥有方（MO）和多个数据拥有方（DO）。MO维护模型，保存模型的权重参数，DO则拥有私有数据，模型架构和超参数时公开信息。Pencil的模型训练流程和FL有些类似，但是不完全相同，大致如下：

1. 在每次训练中，MO的输入为当前模型参数，同时MO选择一个DO进行安全两方计算。本文使用Cheetah和CryptoFlow2的相关协议设计了模型训练中的前向计算和反向传播更新算法。最后更新的权重参数加上差分隐私扰动公开给MO。
2. MO得到当前轮次更新的模型，然后再选择下一个DO进行下一轮的计算更新。如此迭代，直到模型收敛。

3. 协议设计

3.1 线性层协议

模型线性层（例如全连接和卷积）的前向计算则是参考了Cheetah的方案，利用同态加密实现。形式化算法如下：

Algorithm 1: Evaluation of linear layer f

Input: The input $\langle \mathbf{X} \rangle$ shared between MO and DO; MO holds the weights $\mathbf{W}$ and the bias $\mathbf{b}$.

Output: The output shares $\langle \mathbf{Y} \rangle$ of $\mathbf{Y} = \mathbf{W} \circ \mathbf{X} + \mathbf{b}$.

1 DO sends encrypted $[\![\langle \mathbf{X} \rangle_1]\!]$ to MO;

2 MO evaluates $[\![\mathbf{W} \circ \mathbf{X}]\!] = \mathbf{W} \circ ([\![\langle \mathbf{X} \rangle_1]\!] + \langle \mathbf{X} \rangle_0)$ using homomorphic plaintext-ciphertext additions and multiplications;

3 MO chooses random mask $\mathbf{s}$ and calculates $[\![\langle \mathbf{Y} \rangle_1]\!] = [\![\mathbf{W} \circ \mathbf{X}]\!] - \mathbf{s}$; MO sends $[\![\langle \mathbf{Y} \rangle_1]\!]$ back for decryption;

4 DO outputs $\langle \mathbf{Y} \rangle_1$; MO outputs $\langle \mathbf{Y} \rangle_0 = \mathbf{s} + \mathbf{b}$.

计算得到Y之后，可以得到∇Y，进而可以根据链式规则计算前一层的权重梯度。算法如下：

Algorithm 2: Weight gradient $\nabla_{\mathbf{W}}$ calculation

Input: MO and DO input secret shares of $\langle \mathbf{X} \rangle$ and $\langle \nabla_{\mathbf{Y}} \rangle$.

Output: MO receives $\nabla_{\mathbf{W}} = \nabla_{\mathbf{Y}} \odot \mathbf{X}$.

1 DO sends encrypted $[\![\langle \mathbf{X} \rangle_1]\!], [\![\langle \nabla_{\mathbf{Y}} \rangle_1]\!]$ to MO;

2 MO evaluates

$$[\![\nabla_{\mathbf{W}}^{\text{cross}}]\!] = \langle \nabla_{\mathbf{Y}} \rangle_0 \odot [\![\langle \mathbf{X} \rangle_1]\!] + [\![\langle \nabla_{\mathbf{Y}} \rangle_1]\!] \odot \langle \mathbf{X} \rangle_0$$

3 MO chooses random mask $\mathbf{s}$ and sends $[\![\nabla_{\mathbf{W}}^{\text{cross}} - \mathbf{s}]\!]$ back for decryption;

4 DO evaluates

$$\widetilde{\nabla_{\mathbf{W}}} = \nabla_{\mathbf{W}}^{\text{cross}} - \mathbf{s} + \langle \nabla_{\mathbf{Y}} \rangle_1 \odot \langle \mathbf{X} \rangle_1$$

5 DO adds a perturbation $\mathbf{e}$ to $\widetilde{\nabla_{\mathbf{W}}}$;

6 MO finishes by calculating

$$\nabla_{\mathbf{W}} = \widetilde{\nabla_{\mathbf{W}}} + \mathbf{s} + \langle \nabla_{\mathbf{Y}} \rangle_0 \odot \langle \mathbf{X} \rangle_0$$

3.2 线性层预计算优化

在算法1和2中，算法都需要计算大量的明文-同态密文乘法。这些计算在训练的每一步都需要进行，因此会带来大量的开销。本文提出了一种distinguishable却hard的方法来优化上述计算，使得Pencil只需要在预计算做常数次明文-密文乘法，而在线计算则不再需要同态操作。

给定 $\mathbf{u}$, 变量 $\mathbf{v}$: 给定$\mathbf{u}$，在预计算阶段生成随机$\mathbf{v}'$，MO和DO可以计算得到$\langle \mathbf{uv}' \rangle$。进一步，在线计算阶只需要：1）DO发送$\mathbf{v}-\mathbf{v}'$给MO，输出$\langle \mathbf{uv} \rangle_1 = \langle \mathbf{uv}' \rangle_1$；2）MO计算输出$\langle \mathbf{uv} \rangle_0 = \mathbf{u}(\mathbf{v}-\mathbf{v}') + \langle \mathbf{uv}' \rangle_0$。

但是上述方案对于不同的$\mathbf{v}_i$需要生成不同的$\mathbf{uv}'_i$。否则，简单复用的话会泄漏不同$\mathbf{v}$之间的差值。但是，对每一个$\mathbf{v}_i$生成一个$\mathbf{uv}'_i$，那么就会带来巨大的开销。为了提升效率，本文提出如下方案：

1. 首先生成m个随机$\mathbf{v}'_i$，并计算所有$\mathbf{uv}'_i$的秘密分享。
2. 在线计算阶段，对于每一个$\mathbf{v}$，DO生成m个非0的参数k_i，计算

$$\widetilde{\mathbf{v}} = v - \sum_{i\in[m]} k_i \cdot \mathbf{v}'_i$$

3. 最后，两方分别计算

$$\langle \mathbf{uv} \rangle_0 = \mathbf{u}\widetilde{\mathbf{v}} + \sum_{i\in[m]} k_i \cdot \langle \mathbf{uv}'_i \rangle_0$$
$$\langle \mathbf{uv} \rangle_1 = \sum_{i\in[m]} k_i \cdot \langle \mathbf{uv}'_i \rangle_1$$

变量 $\mathbf{u}$: 在实际计算中，$\mathbf{u}$在预计算阶段是不确定的。因此，所以无法直接用上述优化方法。但是类似的，Pencil可以将上述方法对称的应用到$\mathbf{u}$，即生成多个$\mathbf{u}'_i$用以计算$\mathbf{uv}'_j$。具体协议如下：

Algorithm 3: $\mathsf{P}(\circ, \mathbf{u}, \mathbf{v})$: Preprocessing optimization for calculating the shares of $\mathbf{u} \circ \mathbf{v}$

Input: A predefined linear operation $\circ$; in the online phase, MO inputs $\mathbf{u}$ and DO inputs $\mathbf{v}$.

Output: The two parties receive shares of $\langle \mathbf{u} \circ \mathbf{v} \rangle$.

Preprocessing $\mathsf{P}_{\mathsf{Prep}}(\circ)$**:**

MO selects m random masks $\mathbf{u}'_i \sim \mathbf{u}, i \in [m]$;

DO selects m random masks $\mathbf{v}'_j \sim \mathbf{v}, j \in [m]$, and sends their encryption $[\![\mathbf{v}'_j]\!]$ to MO;

MO selects m^2 masks $\mathbf{s}_{ij} \sim (\mathbf{u} \circ \mathbf{v}), i, j \in [m]$;

MO evaluates $[\![\langle \mathbf{u}'_i \circ \mathbf{v}'_j \rangle_1]\!] = \mathbf{u}'_i \circ [\![\mathbf{v}'_j]\!] - \mathbf{s}_{ij}$ for $i, j \in [m]$, and sends them back for decryption;

MO and DO keeps shares of $\langle \mathbf{u}'_i \circ \mathbf{v}'_j \rangle$ for all $i, j \in [m]$

$$\langle \mathbf{u}'_i \circ \mathbf{v}'_j \rangle_0 = \mathbf{s}_{ij}$$

$$\langle \mathbf{u}'_i \circ \mathbf{v}'_j \rangle_1 = \mathbf{u}'_i \circ \mathbf{v}'_j - \mathbf{s}_{ij}$$

Online $\mathsf{P}_{\mathsf{Online}}(\circ, \mathbf{u}, \mathbf{v})$**:**

MO randomly picks scalars $k_i, i \in [m]$; MO sends to DO all k_i and

$$\tilde{\mathbf{u}} = \mathbf{u} - \sum_{i \in [m]} k_i \cdot \mathbf{u}'_i$$

MO and DO produces shares of $\langle \mathbf{u} \circ \mathbf{v}'_j \rangle$ for all $j \in [m]$ as

$$\langle \mathbf{u} \circ \mathbf{v}'_j \rangle_0 = \sum_{i \in [m]} k_i \cdot \langle \mathbf{u}'_i \circ \mathbf{v}'_j \rangle_0$$

$$\langle \mathbf{u} \circ \mathbf{v}'_j \rangle_1 = \tilde{\mathbf{u}} \circ \mathbf{v}'_j + \sum_{i \in [m]} k_i \cdot \langle \mathbf{u}'_i \circ \mathbf{v}'_j \rangle_1$$

DO randomly picks scalars $\ell_j, j \in [m]$; DO sends to MO all ℓ_j and

$$\tilde{\mathbf{v}} = \mathbf{v} - \sum_{j \in [m]} \ell_j \cdot \mathbf{v}'_j$$

MO and DO produces shares of $\langle \mathbf{u} \circ \mathbf{v} \rangle$ as

$$\langle \mathbf{u} \circ \mathbf{v} \rangle_0 = \mathbf{u} \circ \tilde{\mathbf{v}} + \sum_{j \in [m]} \ell_j \cdot \langle \mathbf{u} \circ \mathbf{v}'_j \rangle_0$$

$$\langle \mathbf{u} \circ \mathbf{v} \rangle_1 = \sum \ell_j \cdot \langle \mathbf{u} \circ \mathbf{v}'_j \rangle_1$$

$j \in [m]$

上述代码可以很自然拓展到多个DO，只需要在mask变量$\mathbf{u}$和$\mathbf{v}$时选择不同的随机数$k_i \& \ell_j$。

但是上述技术并不满足不可区分性，但是想要从上述协议中抽取具体的数值信息还是困难的。对于不同的m, f，搜索的难度如下：

m	f	Search space	RSA-k	Time to Break
2	10	20 bits	< 512	62.1 seconds
2	25	50 bits	< 512	2114 years
4	25	100 bits	~ 2048	2.38×10^{18} years
8	25	200 bits	~ 7680	3.02×10^{48} years

TABLE IX: Hardness of the adaptive attack against the pre-processing optimization. RSA-k means the RSA modulus bit length offering equivalent security guarantees. Time to break is evaluated or estimated using the CIFAR10 dataset.

3.3 非线性层 & 同态优化

对于非线性层，本文用调用Cheetah和CryptFlow2来计算ReLU、2D-Average池化层、和截断。具体可以看之前的博客。
对于同态计算优化，本文利用GPU优化了同态加法、乘法计算效率$10\times$以上。

4. 实验评估

本文针对多个模型进行了实验，测试了训练准确率、通信和计算开销等。准确率如下：

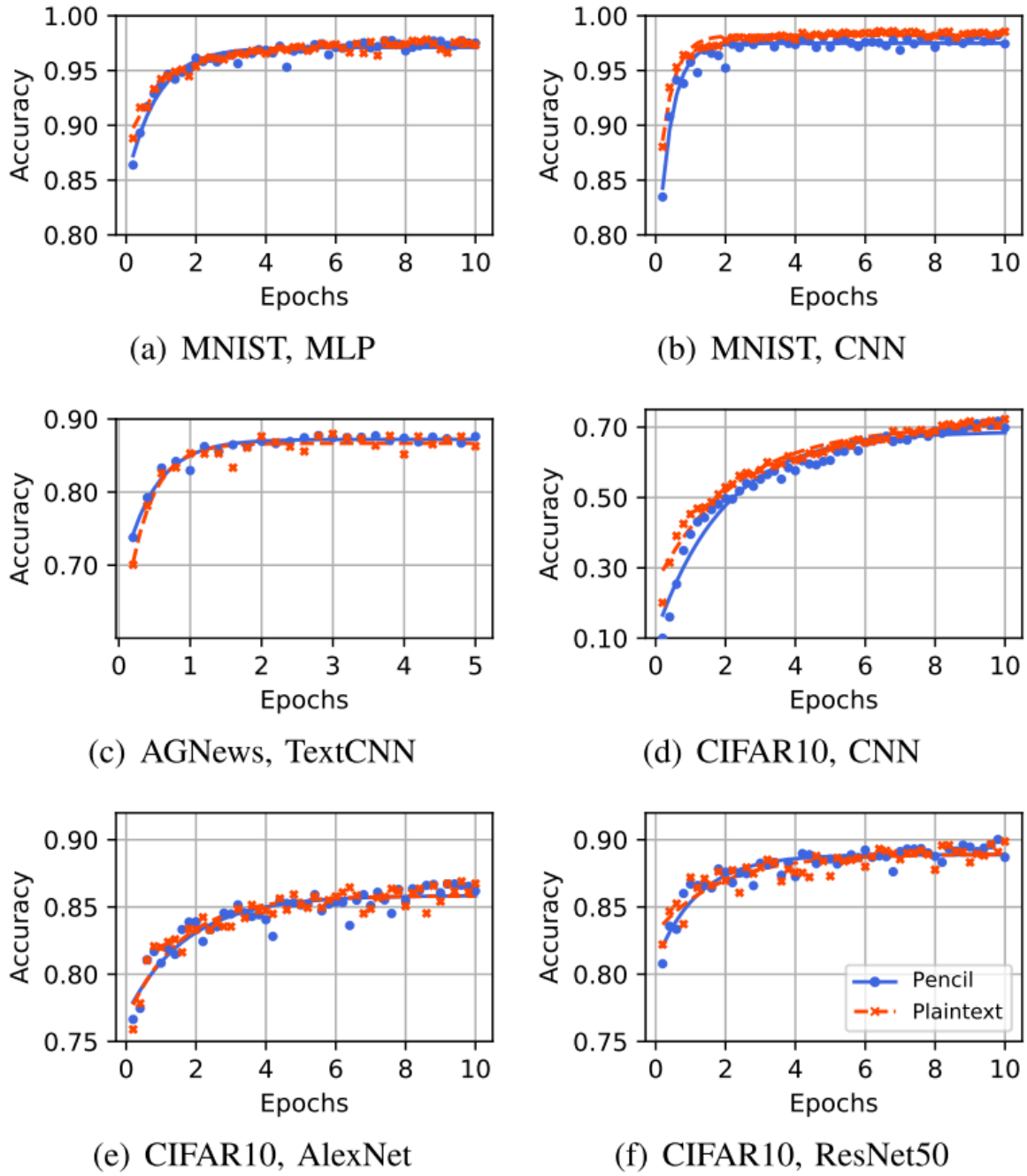


Fig. 1: Test accuracies for trained models. (a) $\sim$ (d) are for models trained from scratch; (e) and (f) are for models trained via transfer learning.

Scenario	Task	Model	Pencil			Pencil$^+$				
			Online			Preprocessing		Online		
			TP_{LAN}	TP_{WAN}	C	T_{prep}	C_{prep}	TP_{LAN}	TP_{WAN}	C
Train from scratch	MNIST	MLP	9.73×10^4	5.12×10^4	1.66	0.02	3.35	26.52×10^4	19.87×10^4	0.23
	MNIST	CNN	7.70×10^4	4.43×10^4	1.71	0.02	4.13	13.72×10^4	10.75×10^4	0.36
	AGNews	TextCNN	0.37×10^4	0.53×10^4	14.62	0.27	19.28	0.76×10^4	1.07×10^4	6.74
	CIFAR10	CNN	0.18×10^4	0.12×10^4	44.89	0.70	83.12	0.22×10^4	0.15×10^4	34.90
Transfer learning	CIFAR10	AlexNet	0.52×10^4	0.39×10^4	11.33	0.91	46.00	1.55×10^4	1.24×10^4	2.90
	CIFAR10	ResNet50	1.83×10^4	1.17×10^4	5.48	0.30	15.96	8.05×10^4	5.89×10^4	0.82

TABLE III: Training costs for different ML tasks. For the online phase, TP stands for the throughput (samples/hour) of the training system, and subscript LAN, WAN indicate the network settings; C stands for the online communication (MB) per sample. For Pencil$^+$, we also report the time (T_{prep}, hours) and communication (C_{prep}, GB) of preprocessing. Note that the preprocessing overhead is one-time overhead.