DATA ANONYMISATION FOR NEURAL NET TRAINING USING PRIVACY PRESERVING TECHNIQUES

Amel Sellami[1], Amine Kerkeni[2], Karim Beguir[3]

InstaDeep™

ABSTRACT

One of the biggest concern in modern machine learning applications is the user's privacy. To tackle this problem, we introduce tools like Homomorphic encryption(HE), Secure Multi-Party Computation (SMPC) that keeps the privacy of the data, decentralization of the model's intelligence and helps in building scalable applications that were not even possible before the era of artificial intelligence.
Within this poster, we are going to demonstrate that it is feasible to build privacy-preserving applications using Homomorphic encryption(HE) and secure multi-party computation (SMPC).

I. HOMOMORPHIC ENCRYPTION

1- WHAT IS HOMOMORPHIC ENCRYPTION ?

A cryptosystem is considered fully Homomorphic if it displays both additive and multiplicative homomorphism in an asymmetric way.
In the Paillier cryptosystem, if the public key is the modulus m and the base g, then the encryption of a message x is: $\mathcal{E}(x)=g^x r^m \bmod m^2$
for some random $r \in \{0, \ldots, m\text{-}1\}$
The homomorphic property is then:

$$\mathcal{E}(x_1)\cdot\mathcal{E}(x_2) = (g^{x_1} r_1^m)(g^{x_2} r_2^m) \bmod m^2 = g^{x_1+x_2}(r_1 r_2)^m \bmod m^2$$
$$= \mathcal{E}(x_1+x_2)\mathcal{E}(x_1)\cdot\mathcal{E}(x_2) = (g^{x_1} r_1^m)(g^{x_2} r_2^m) \bmod m^2 = g^{x_1+x_2}(r_1 r_2)^m \bmod m^2 = \mathcal{E}(x_1+x_2)$$

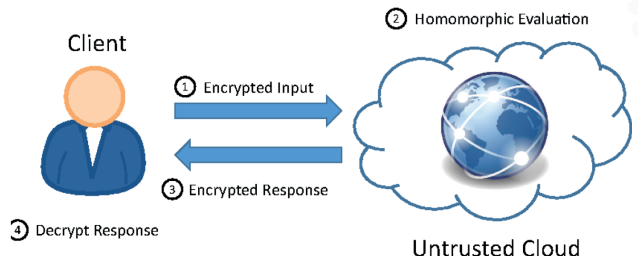

Figure 1.0 Homomorphic Encryption illustration

2- IMPLEMENTATION AND RESULTS

In Homomorphic encryption scheme, we encrypt the data and then perform the mathematical computation of a deep learning model on them. One problem in this approach is that it is useless for the ones who do not possess the decryption key, also, it is hard to train a model on different data sources which are mostly the use case scenario in most of the machine learning production applications.
That's why we flipped the process. We encrypt the neural network instead of encrypting the data. We train the model on decrypted data in Figure1.0. The data that we have used is for an XOR neural network.
We approximated the activation function using Taylor expansion and we only encrypted the weights of the neural network.
The prediction was accurate but the training was so slow and tuning was tricky because of the combination of noise in the decryption mechanism and low precision.
The tables 1.0 and 2.0 shows the results.

Eta	*0.01*	*0.001*
Hospital 1	*39.83*	*302.45*
Hospital 2	*120.99*	*315.17*
Hospital 3	*49.58*	*315.17*

Table 1.0 : Difference Between errors when changing eta with a fixed length number of iterations 100

Iterations	*50*	*100*	*200*
Hospital 1	*184.61*	*302.45*	*392.35*
Hospital 2	*187.08*	*315.17*	*431.26*
Hospital 3	*184.09*	*297.71*	*379.24*

Table 2.0 : Difference Between errors when changing the number of iterations with a fixed eta equals to 0.001

II.From HE to SMPC

We have used the Paillier HE scheme. We believe that HE alone is not sufficient and solve the problems of model complexity adding to it that the model is slow and uses hard computations. In the literature, there are other algorithms that we could have tested like the YASHE algorithm but instead, we have chosen to test another protocol to preserve the privacy in an AI application which is SMPC.
HE and SMPC are modern cryptography algorithms, they can be used interchangeably to perform the same task when the former uses hard computations and fewer interactions and the latter uses more interactions and fewer computations.

References

[1] Efficient Integer Vector Homomorphic Encryption. Angel Yu, Wai Lok Lai, James Payor Massachusetts Institute of Technology, May 2015
[2] Secure Computation for Machine Learning With SPDZ, Valerie Chen Valerio Pastro Mariana Raykova [cs] arxiv.org/abs/1901.00329
[3] Morten Dahl, mortendahl.github.io/2017/04/17/private-deep-learning-with-mpc/
[4] Andrew Trask, iamtrask.github.io/2017/03/17/safe-ai/

III. MULTI PARTY COMPUTATION

1- WHAT IS MULTI PARTY COMPUTATION?

Secure Multi-Party Computation also know as (SMPC) is a generic cryptographic primitive that enables distributed parties to jointly compute an arbitrary functionality without revealing their own private inputs and outputs.
This is similar to performing local training on local unencrypted data and then encrypts the gradients to upload it, the figure 2.0 below shows an illustration the kind of application that can use this technique, such as Hospitals,
In a multi party computation protocol there are:

- ***n*** participants: ***n1, n2, ...n***
- ***n*** inputs ***xi***,
- A function f that the participants want to evaluate on all the inputs
  i.e. the goal is to compute:
  Y = F(x1, ..., xn))
- Secret Sharing Scheme
- A dealer D who has a secret s
- ***n*** parties ***P1,..., Pn.***

A secret-sharing scheme is a method by which the dealer distributes shares of s to the n parties such a way that:
(1) any subset of k + 1 parties can reconstruct the secret from its shares.
(2) any subset of k parties cannot retrieve any partial information on the secret s.

Figure 2.0 , Multi party computation

2- IMPLEMENTATION AND RESULTS

With SMPC, the training data is secured by secret sharing. The data that we have used is the Diabetes data existing in the literature. The choice of that data is based on the importance of medical data privacy. The model that we have tested is a simple two-layer network where Taylor expansion is used to approximate the sigmoid activation function, we reduced the number of multiplications in a model using Horner's method.
We tested the robustness of this technique by adding layers to the model, Figure 2.0 and Figure 3.0. What we have noticed is that after 10000 Iteration, the resulted weights and predictions explodes. This phenomenon is due to our approximation which did not allow the model to get more confidant. Increasing the degree of the polynomial was not enough. A better alternative was to use instead of the standard approximation, the interpolation over an interval. Keeping the degree of the polynomial low increased the efficiency of the model. While using secret sharing in SMPC gave promising results the collapsing results showed that we need more advanced techniques, that take into consideration the speed of the Network and improving other properties like the number of rounds by Garbled circuits for instance, which is a technique used that we did not explore in our work.

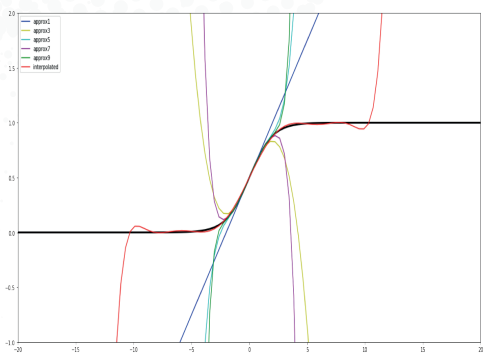

Figure 2.0 Approximation variation of the polynomial

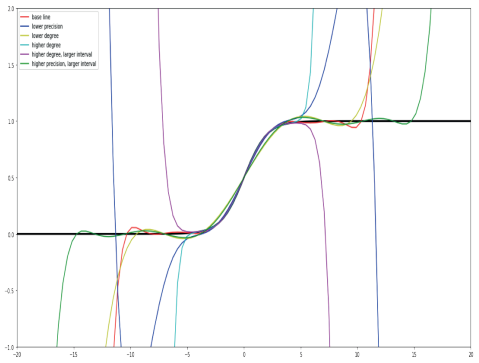

Figure 3.0 Degree variation of the polynomial

Conclusion

To conclude, we have demonstrated that we can use HE or SMPC to keep the privacy of the model or the data in a machine learning application. We have demonstrated that HE, is sufficient in only certain kind of applications. SMPC is one of the solutions to build a powerful pipeline but needs more research to help generalization in a deep learning state of the art models.
For future work in privacy-preserving techniques, we recommend benefiting from Differential privacy and Federated learning.