Secure Multi-party Computation in the Context of Deep Learning

CS6160 : Cryptology

Raj Patil : CS18BTECH11039

2021-11-22

Contents

1 Introduction

1.1 Context

The application of Machine Learning (ML, henceforth) models have been proliferating, fairly rapidly, in several domains; elucidating some:-

- Facial Recognition
- Self Driving Cars
- Marketing (demographics specific advertising)
- Medical Imaging (examining chest x-rays)

All these use-cases require assimilating large datasets. Most data, being of sensitive nature, needs to remain private when training the concerned model. These datasets may be sourced via different media:

- Crowd-sourcing
  - smart phones heavily facilitate such ventures
- a Centralized Trustee (Medical/Financial Institutions)

Moreover, during inference of client supplied data, the model-owners may also wish to mask the internals of the model. One major catch is that these models usually need to be deployed in scenarios with limitations on the compute (smart-phones, for instance): this calls for quantization of the models: introducing its own set of challenges.

Summarizing: one needs to enable the usage of quantized neural networks with privacy measures in place to protect the model and the data, without sacrificing the original models' accuracies.

1.2 Outline

This report intends to highlight the recent developments in the aforementioned space, in the context of cryptography, and leave the reader with a generic insight as to how one may go about designing their own protocols, assisted by some concrete examples.

We first explore what Secure Multi-party Computation(MPC, henceforth) is, alongside mathematical and pragmatic approaches to the same, followed by a quick review of the structure of a neural network from the perspective of our objective(secure evaluation of quantized neural networks).

As and when we know enough, we'll revisit the problem statement, breaking it down into orthogonal pursuits.

Once the pre-requisites are dealt with, the contributions of the principle paper being reviewed[1] are consolidated.

Finally, we visit the possibilities that are realized due to advancements in secure computation and what probable research threads may be pursued in the future.

2 Secure Multi-Party Computation

MPC is a sub-domain of cryptography with the aim to allow multiple parties to collaboratively compute a function over their inputs which they wish to keep private. Traditionally, the adversary is not a part of the normal functionality of the cryptosystem but in this case, the parties wish to protect their data from each other as well. Note that one possible solution would be to provide all the private data to a trusted third party which then computes and returns, only, the output to the interested parties. However, The need for MPC protocols arises due to the fact that one doesn't live in a Utopian world and trust is a logistically infeasible resource for some applications.

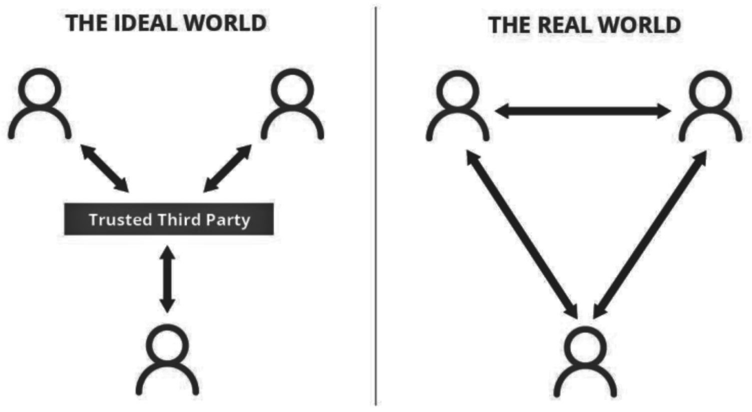

Figure 1: Incentive for MPC protocols

$$a = 34$$

Any MPC protocol should, stating informally, comply with the following requirements[2]:

1. Input Privacy: The information observable, when carrying out the protocol, should not leak out the private data held by the individual participants other than what is inevitably leaked from the output of the computation.
2. Correctness: An adversary (a.k.a a dishonest party) should not be able to force honest parties to output an incorrect answer via any means possible (supplying malicious data or deviating from protocol)

Note that MPC is a vital field for several applications like electronic voting, privacy preserving data mining, et cetera but we will focus on the applications pertaining to deep learning.

[1]Dalskov, A., Escudero, D., & Keller, M. (2020). Secure Evaluation of Quantized Neural Networks

[2]Yahuda, L. A Primer in Secure Multiparty Computation

3 Deconstructing the Problem

Before proceeding, for the sake of clarity, recall that the problem statement being explored is actually two fairly orthogonal subproblems:

1. quantizing neural networks for performance and mobile inference
2. facilitating secure inference (via MPC protocols) for these neural networks

The major cryptographic applications are rooted in the second sub-problem and one only needs to navigate the first one so as to not let it hinder the application (through means like efficiency, correctness and more) in the second one. Now that the objective is clearer, the discussion further ahead will be parted according to the two pursuits

4 Secure Inference

4.1 Canonical Perspective of Neural Networks

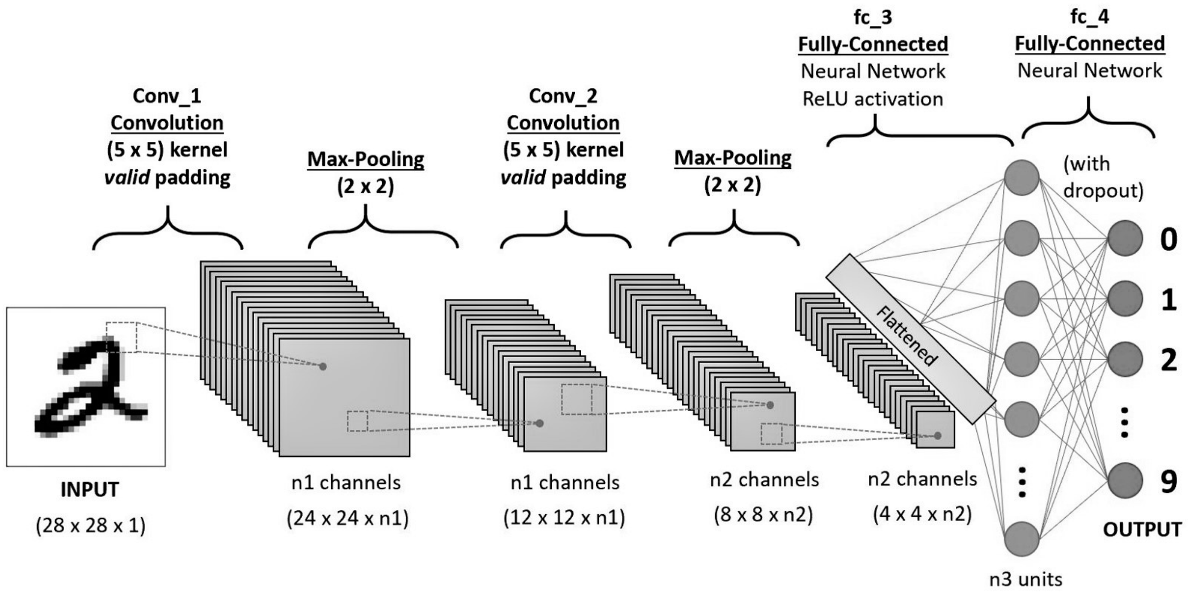

Figure 2: a typical CNN

Concentrating the discussion towards a specific instance: Convolutional Neural Networks (CNNs henceforth). Recall that these can be decomposed into simpler ones as follows:

Procedure (Layer)	Canonical Decomposition
Convolution	matrix multiplication
Pooling(Max)	a bunch of comparisons
Pooling(Weighted)	matrix multiplication
Fully Connected	matrix multiplication
Non-trivial Activations	approximable as additions and multiplications

Notice that, under the hood, matrix multiplication is simply a collections of additions and multiplications. Hence, if one can figure out how to securely add, multiply and compare numbers from different parties, one can also securely conduct inference for a CNN.

Pedagogically underscoring 2-party secure addition while directing to relevant references for multiplication and comparison:

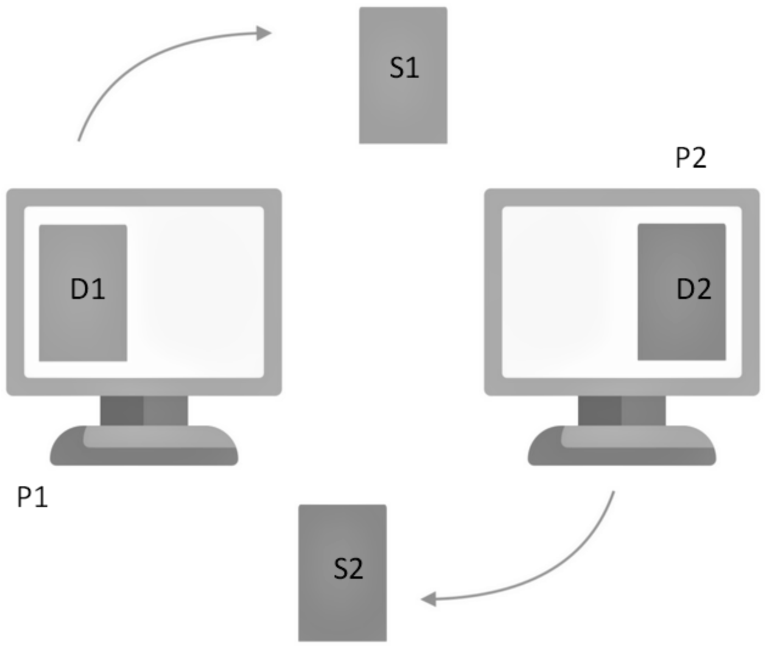

Figure 3: basic 2PC framework

4.2 Secure 2-Party Addition

The basic idea revolves around splitting the private data of a party into a secret that will be shared with others and a part that will not be shared. Finally, all the parties work with the shares and combine them to get the desired result without leaking the private part.

Realizing these ideas with secure addition now, the individual data item (datum) of a party (DS) is split into the private D and a share-able S (to be shared with the other party), with the parties being P_1 and P_2 and the subscripts being donated to the individuals accordingly.

The two parties (P_1 and P_2) want to securely compute $DS_1 + DS_2$. Now:

$$\begin{aligned} DS_1 + DS_2 &= (D_1 + S_1) + (D_2 + S_2) \\ &= (D_1 + S_2) + (D_2 + S_1) \end{aligned}$$

Now P_1 can solve $D_1 + S_2$ while P_2 solves the other half and upon combination, the correct result is output.

The splitter can be formulated as follows, the only constraint being DS remains private given S:

```
func splitter(DS):
    S <- get_rand() \\ randomly sourcing the share
    D <- DS - S     \\ the share doesn't leak any
    return D,S      \\ information about DS
```

Figure 4: splitter for secure addition

4.3 Other Secure 2-Party Primitives

Secure multiplication is a slightly more involved and the parties will need to communicate multiple times during one operation, while also needing techniques like masking with a helper.[3] Several efficient ways for comparison have also been researched into[4] and now that one is equipped with these primitives, the complex requirements can be easily formulated.

4.4 Approximating Complex Activation Functions

Here we can introduce two kinds of irregularities that need to be maneuvered around a bit to be implemented:

1. A piece-wise differentiable activation function (e.g. ReLU)
2. A inherently transcendental function (e.g. Sigmoid)

The first kind of irregularity can be tackled using a secure comparison black-box, allowing us to reduce the abstraction down into the second kind. Note that this also allows us to deal with the discontinuities in an activation function. Given the formulation has been reduced to multiple instances of the second kind, now one can employ Taylor's theorem to model the differentiable function in the definite range as a polynomial. Observe that a polynomial is representable by a bunch of multiplications and additions.

Do note that the preceding representations are not necessarily the ones to be used in implementations which formally draws upon approximation theory, and to be more specific: they are implemented using Chebyshev polynomials, a good treatment of which can be found in Chebyshev Polynomials in Numerical Analysis. Hence, given the secure computation primitives (addition, multiplication and comparison) are in place, one can approximate functions beyond the scope of practical needs of a neural network.

5 Quantization

Now that secure computation is dealt with on a lower level, quantization can be brought into the picture. The authors [5] explicitly delved into secure inference with quantization as that holds greater pragmatic importance as discussed before. Observing quantization to an 8-bit domain briefly. Most approaches[6] quantize the network layer by layer and this is a summary of how it's done:

1. realize the range the weights of that layer fall into - characterizing it by $[\alpha_{min}, \alpha_{max}]$
   - this is done so as to not have extremely dense and sparse spaces that would be encountered if one moved ahead with the same range assumption for all layers.
2. observe now that given this formulation, one has a set of maps
   - a many-to-one quantizer taking the weights from $\mathbb{R}$ to $\mathbb{Z}_{2^8}$
   - a one-to-one dequantizer dequantizing those transformed weights back into their representative weights in $\mathbb{R}$
   - note that these maps are completely determined by the range of the original weights

[3] `https://medium.com/dropoutlabs/secret-sharing-explained-acf092660d97`
[4] Rabbit: efficient Comparison for Secure Multi-Party Computation
[5] Dalskov, A., Escudero, D., & Keller, M. (2020). Secure Evaluation of Quantized Neural Networks
[6] Benoit J. et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference

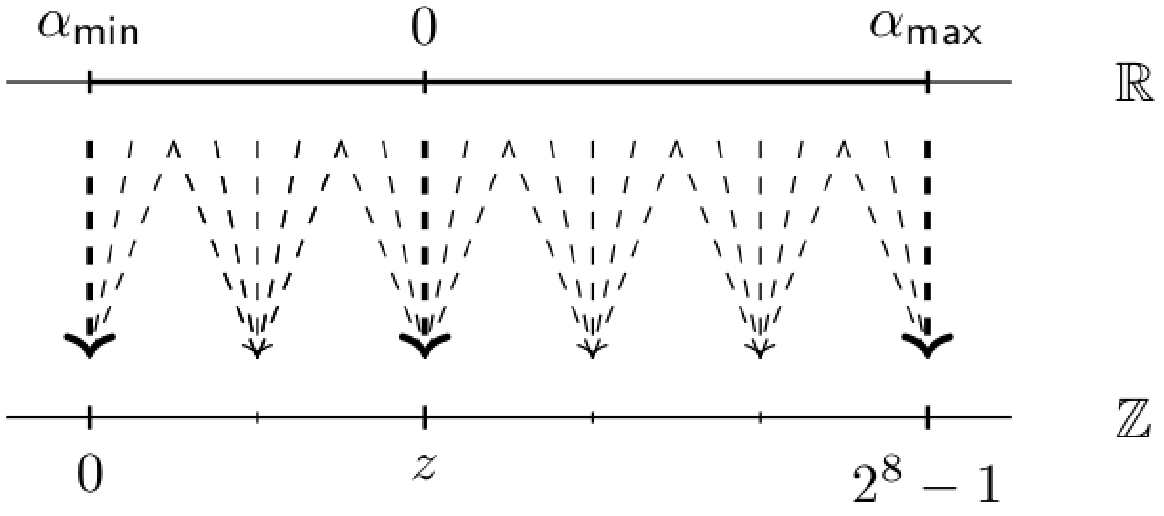

Figure 5: Quantization

6 Challenges and Contributions

Now that we've the primitives figured out, summarizing on the challenges encountered in this domain and the contributions of the principal paper.

6.1 Challenges

- Training a network and deploying it for secure inference is not currently facilitated by commonly used frameworks such as PyTorch, TensorFlow, et cetera
- A major issue arises if the quantization is done in a naive manner: even if the model had converged in the finer domain of floats, its quantized representation may not be able to perform at the same level if one does not work out the peculiarities
- Even if some frameworks [7] do manage to get the conversion compatible with common frameworks, they impose other constraints such as specific hardware requirements
- Most MPC protocols, are only applicable for too specialized of a neural network architecture, disqualifying them from the scrutiny of mainstream research pursuits which affects the rate of progress of the domain i.e. one needs to have model-agnostic protocols

6.2 Contributions

- Using the quantization technique briefly stated before in this report, the authors have abstracted the process of quantization in terms of a dot-product followed by a truncation: hence forwarding the cause of model-agnosticism
- An important bottleneck that was overcome by the authors was that of the truncation to be done during the quantization: by shifting to a probabilistic approach rather than a deterministic one, they've managed to maintain accuracy and efficiency of inference at the same time

[7] Nishant Kumar et al. Cryptflow: Secure tensorflow inference. Cryptology ePrint Archive, Report 2019/1049, 2019. `https://eprint.iacr.org/2019/1049`.

Table 1: Possibilities of Adversarial Conditions in MPC protocols

Breakdown of Adversarial Conditions		**Adversarial Nature**	
		Active	*Passive*
Majority	*Honest*	<50% malicious deviants	<50% malicious observers
	Dishonest	>50% malicious deviants	>50% malicious observers

7 Testing Protocols : Factors

Now that we have seen the basic 2PC framework, introducing the client server model for our specific case: The collection of P_i being the server cluster and the model and data owner being the clients. The clients

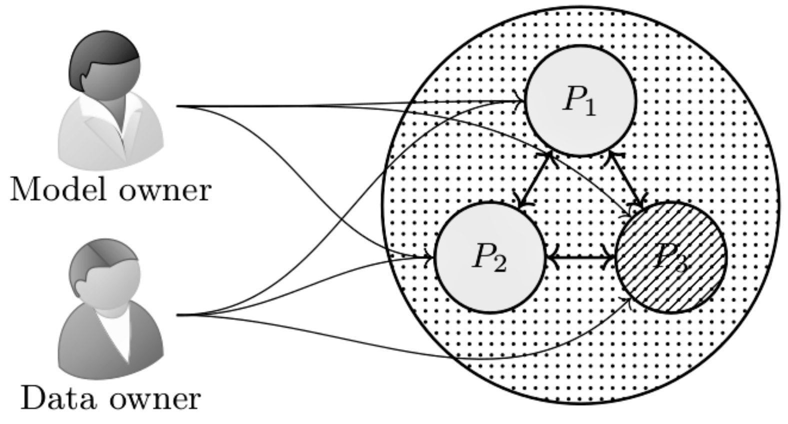

Figure 6: client server model for inference by a neural network: P_3 is dishonest

share the data to the servers which then execute the secure computation, receiving the results in return.

Most research will test out their proposed protocol against the adversarial conditions summarized in table 1, w.r.t to the following dimensions:

1. Adversarial Nature: The dishonest party can be active (deviates from protocol) or passive (only capable of observing the operations). The first case is a stronger, more practical formulation of an attacker. In most cases the best solution is for the honest parties to abort operations in case an active adversary successfully breaks protocol. This guarantees that if an answer is output, it is correct.

2. Honesty Majority: Given multiple parties are involved, performance of the protocol depends heavily on the fraction of honest and dishonest parties and hence, to account for practical scenarios, one must test out the protocol while varying this fraction.

The authors conducted a range of experiments for different extents of these threats along different dimensions. What is more important is that due to accessibility of the proposed protocol for common frameworks, these experiments serve as benchmarks and will be under scrutiny by a larger community than if the MPC protocol had too specific requirements. Finally, their major contribution lies in the domain of quantization rather than cryptography: they employed probabilistic truncation which allowed them to maintain efficiency and accuracy at the same time, hence promoting model-agnostic MPC protocols.

8 Possibilities

Finally, as an interesting expansion to secure inference, a lot of research is being put into Homomorphic Encryption [8], [9]. This allows the data owner to send in encrypted images to the model, which it processes without decrypting, sends it back, allowing the user to observe the results without leaking any data. Note that this is an extension to the static nature of MPC protocols which may be for online secure training of models rather than just offline inference.

9 References

- Dalskov, A., Escudero, D., & Keller, M. (2020). Secure Evaluation of Quantized Neural Networks
- Yahuda, L. A Primer in Secure Multiparty Computation
- Rabbit: efficient Comparison for Secure Multi-Party Computation
- Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. CoRR, abs/1712.05877, 2017.
- Nishant Kumar et al. Cryptflow: Secure tensorflow inference. Cryptology ePrint Archive, Report 2019/1049, 2019
- Viand, A., Jattke, P., & Hithnawi, A. (2021). SoK: fully Homomorphic Encryption Compilers.

[8]Viand, A., Jattke, P., & Hithnawi, A. (2021). SoK: fully Homomorphic Encryption Compilers.
[9]`https://medium.com/dropoutlabs/introducing-dropout-labs-d1b96f638ae2`