A comprehensive analysis of CrypTen for Neural Network inference

Maitreya Patel* Elliot Nester†

Abstract

With the recent advent of deep learning, more and more companies are adopting and providing deep learning-based services. However, most of these services require users to upload the data to companies' servers or companies have to share their proprietary models with users. Both of these approaches hinder the privacy of the users' data and companies' proprietary models. Recently, many attempts have been made to tackle this problem and provide two-way privacy. But, all published approaches are still time and communication costly, and they cannot be easily integrated with existing services. To remedy this problem Facebook introduced CrypTen, an open-source secure multi-party-based approach with significant scalability. In this work, we compare CrypTen with previous State-of-the-art (SOTA) methods to understand the overall impact and efficiency of CrypTen relative to various metrics. Analysis on standard industrial classification neural network architectures (i.e., ResNet32) suggests that CrypTen significantly outperforms the baselines by at least 94% in terms of total processing time (pre-processing + online) and it performs comparably during the online phase.

1 Introduction

Recently, because of the improvements in computational hardware like GPUs, neural network-based methodologies are being applied to countless problems. Industries are developing deep learning-based solutions and providing essential services as an endpoint. For example, the healthcare field is one of the most important industries in our lives and it can be greatly impacted by the progress of machine learning. However, such an application involves the transfer of patients' private information such as medical history, medications information, and diagnosis results. Because of these and many other problems, privacy must not be ignored while providing the customer their desired services. Most of the existing services can be categorized into two parts: 1) where the client shares their private data, and 2) where the company shares their proprietary model to the user. However, neither of these approaches is suitable as users' data should be private and the company may be running a proprietary model trained on several private datasets.

To tackle this problem, in the past, many cryptography-based approaches have been identified such as CryptoNets and Gazelle [1, 2]. Additionally, many solutions rely on secure enclaves provided by hardware companies, but these are not scalable and their computation power is highly limited. Therefore, in this study, we only consider cryptography inference-based methodologies such as Gazelle, Falcon, and Delphi [2, 4, 5]. These approaches are still very limited in terms of time and communication costs though, and they cannot be adopted easily by existing services. Methodologies like Gazelle take 232 seconds and transfer over 8GB of data during setup and online phases, while Delphi takes about 104 seconds and transfers 4GB of data for a single image classification task on ResNet32 [5]. Moreover, except for Delphi, none of the baselines supports GPU-based secure inference, which again shows the limitations of existing methodologies.

CrypTen is a secure multi-party computation based approach under development by Facebook [3]. CrypTen is developed on top of PyTorch to make the secure training/inference of neural networks more applicable to existing and future DL solutions. However, deep learning is moving ahead quickly and there are no proper comparisons available online for CrypTen with previous SOTA methods. This makes it difficult to figure out the usefulness of CrypTen on various deep learning methods. Therefore, in this study, we focus on performing a comprehensive analysis of CrypTen and making comparisons by testing the framework on similar scenarios to those used in published SOTA results.

We first implemented a COVID X-ray detection-based inference using CrypTen by replicating similar settings and environments as those used in baselines. Later, we compared CrypTen with baselines by using ResNet32 (460k parameters) and MiniONN (160K parameters) architectures by analyzing the costs of different layers and the overall time and communication costs. Our analysis suggests CrypTen significantly outperforms baselines, as it only requires about 6 seconds and 100MB of communication when tested on the ResNet32 architecture.

*CSE Master's 1st year
†CSE Master's (4+1) 2nd year

2 Problem Setup and Threat Model

In this section, we first focused on defining a CrypTen-based problem setup for secure computation and then focused on CrypTen's threat model.

2.1 Problem Setup

CrypTen relies on private addition and multiplication for all types of computations, so it defines scalable arithmetic and binary secret shares that have homomorphic properties. Most of the deep learning solutions can be divided into several layers which contain addition, multiplication, linear functions, non-linear activations, and various comparators. Moreover, linear layers can be further divided into addition and multiplication. While non-linear activations (i.e., sigmoid, softmax, and logloss) are replaced with standard approximation. Moreover, comparators are implemented by creating binary shares to compute sign bits and later answer is distributed using arithmetic shares.

2.2 Threat Model

CrypTen relies upon the semi-honest-based threat model, which is Honest-But-Curious (HBC). In this threat model, it is assumed that the attacker or adversary will follow the protocol but simultaneously try to learn all possible information in the process. In this threat model, there can be two different adversaries: 1) at the user level, and 2) at the company level. The company-level adversary should not learn anything about the user input except for the model weights. And user-level adversary should not learn anything except for the user's input, the output of the model, and only architecture details without weights.

3 Experimental Setup

In this section, we focused on different experiments carried out using CrypTen to compare it with baseline approaches.

3.1 System setup

One consideration is that all of the baselines were implemented using different computing architectures. For example, Falcon was tested on an Intel i5-7500 with 4 cores running at 3.4 GHz, Delphi was tested on an Intel Xeon 8000 series processor with 4 cores running at 3.0 GHz, and in our experiments, we tested CrypTen on an Intel i7-7th gen with 4 cores running at 3.4 GHz. As there is only a slight difference between base clock speed between Falcon/CrypTen and Delphi, we believe we can make fair, approximate comparisons.

3.2 Experimental Results

To analyze CrypTen's inference efficiency, we first trained ResNet32 and MiniONN architectures on a COVID X-ray dataset from Kaggle[1]. Here, we observed that class imbalance is 3:1 for a COVID: Normal patient. ResNet32 achieves 92.6% accuracy within the first 7 epochs, while MiniONN achieves 91.5% accuracy at the 30th epoch. Next, we compared the industrial standard ResNet32 model and toy MiniONN model for comparison on classification-based secure inference.

3.2.1 Execution time and communication analysis

We evaluated the performance of CrypTen on the Resnet32 model and compared it with Gazelle and Delphi. We further evaluated CrypTen on the MiniONN model and compare it with Gazelle, Falcon, and Delphi. Here, to ensure a fair comparison with Delphi, only CPU-based results are reported as done in Mishra *et. al.* Since the baselines evaluate their approaches on 3x32x32 sized images, we first downscaled the X-ray images for a fair comparison.

Table (1) shows the comparison with baselines of a ResNet32 architecture with 460k parameters. Here, it can be inferred CrypTen shows an overall improvement of at least 93% and performs comparably to Delphi in the online phase[2]. Similarly, Table (3) shows the baseline comparison of a MiniONN architecture with only 160k parameters. On the MiniONN architecture, CrypTen outperforms the baselines as well. In Table (2) the results for the setup and online time consumption for the baselines with various layer types are shown[3]. It can be observed that Falcon takes significantly less time compared to CrypTen and Gazelle, however, the

[1] `https://www.kaggle.com/tawsifurrahman/covid19-radiography-database`

[2] Falcon does not evaluate their performance on ResNet32.

[3] Delphi follows a similar strategy as Gazelle and they improve it using Neural Architecture Search to replace ReLU with quadratic activation.

Table 1: Execution time and communication comparison on ResNet32.

Model	Phase	Time (s)	Communication (MB)
Gazelle	Pre-processing	∼150	∼8192
	Online	∼82	∼560
Delphi	Pre-processing	∼100	∼2024
	Online	∼4	∼70
CrypTen	Pre-processing	∼1	-
	Online	∼6	∼76

Table 2: ReLU and MaxPooling time constraints.

Layer	# inputs	Time (ms) - Setup			Time (ms) - Online		
		Gazelle	**Falcon**	**CrypTen**	**Gazelle**	**Falcon**	**CrypTen**
ReLU	1000	89	9.82	243	15	4.2	147
	10000	551	96.2	230	137	43.2	160
MaxPool	1000	164	12.1	200	58	5.6	600
	10000	1413	100	240	513	45.5	610

setup time reported for CrypTen includes the time it took to load the model weights (even if ReLU does not have any weights) as CrypTen has automated model encryption, unlike the baselines. It also can be observed that the time requirements of Gazelle and Falcon increase linearly with an increase in the number of inputs, while CrypTen stays almost constant. Table (4) shows communication requirements for the different approaches. Here, CrypTen significantly outperforms the baselines in its overall requirements and for all methods, the communication costs increase linearly with an increase in the input data size. Further analysis and results are given in the appendix.

3.2.2 Application to regression

Another common use case of neural networks is to produce outputs for regression problems. In the realm of image processing, that output may be a style transfer, segmentation map, or quality-restored version of the input. We attempted to apply CrypTen to an image segmentation task using the U-Net architecture, as this represents a legitimate real-world use case where users may want to perform fast, intelligent segmentation of images without the service providers seeing their images. Image regression architectures can also be more complex than classification architectures due to the output often having the same dimensions as the input, allowing for further useful insights amid our classification results.

This particular architecture, like many similar architectures, utilizes upsampling layers in its convolutional network. Unfortunately, we discovered CrypTen does not yet support this operation type, revealing some major limitations of the framework. Because this layer type is fairly common in major SOTA regression networks, CrypTen is not yet ready to become a ubiquitous tool capable of solving all machine learning privacy issues. Perhaps after several years of further development, CrypTen will be ready to be tested in a serious regression experiment, but as of now, that is not the case.

4 Conclusions

In this study, we attempted to make a proper comparison between CrypTen for Neural Network Inference and SOTA methods to understand how it holds up in real-world scenarios. First, we implemented a CrypTen-based inference module on COVID X-ray detection by keeping the system configurations as close as possible to the baselines. We analyzed and compared CrypTen with Gazelle, Falcon, and Delphi on overall time and communication costs with layer-wise breakdown for standard architectures (i.e., ResNet32 and MiniONN). The results show CrypTen outperforms the baselines in terms of overall execution time and communication costs except in ReLU layers where CrypTen performs relatively poorly. Therefore, based on their usabilities and efficiencies, we came to the following ranking: CrypTen > Delphi > Falcon > Gazelle. Even though Delphi supports GPU computations as a batch-wise computation which further decreases the costs during the online phase, we rank it second because CrypTen is more usable and is a mostly automatized tool. However, CrypTen is still under development and does not support GPU computation and many important PyTorch functionalities used in various deep learning solutions.

References

[1] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210. PMLR, 2016.

[2] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1651–1669, 2018.

[3] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten. Crypten: Secure multi-party computation meets machine learning. In *Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning*, 2020.

[4] S. Li, K. Xue, B. Zhu, C. Ding, X. Gao, D. Wei, and T. Wan. Falcon: A fourier transform based approach for fast and secure convolutional neural network predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8705–8714, 2020.

[5] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa. Delphi: A cryptographic inference service for neural networks. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 2505–2522, 2020.

Appendices

A Secure multi-party computations for Neural Networks

As discussed previously, all the types of computations for Neural Networks can be divided upon addition and multiplication. Moreover, non-linear operations can be approximated to make them work in this setting. Following is a summarized overview of secure computation by CrypTen [3].

A.1 Secure addition

Each party p computes the addition of their shares of $[x]$ and $[y]$ by creating arithmetic shared variables. Later another arithmetic shared variable is created denoting their final summation $[z]_p$ as $[z]_p = [x]_p + [y]_p$.

A.2 Secure multiplication using Beaver triples

Secure multiplication uses beaver triples on top of arithmetic shared variables. Consider, three such variables $([a], [b], [c])$ with the goal of $c = ab$. Now, CrypTen creates another two such variables (i.e., ε, δ) as $[\varepsilon] = [x] - [a]$ and $[\delta] = [y] - [b]$, respectively. At last, final result of secure multiplication can be defined as $[x][y] = [c] + \varepsilon[b] + [a]\delta + \varepsilon\delta$.

A.3 Secure comparison

CrypTen calculates $[z < 0]$ by first creating binary secret shares then creating the arithmetic secret shares of the sign bit, $[b] = [z] >> (L-1)$. This allows CrypTen to support ReLU activation as $ReLU([x]) = [x][x > 0]$. Because of this CrypTen require less communication and time constraints. Also, this allows CrypTen to support such important activation functions in Deep Learning without affecting the results and efficiency too much.

B Extended results

This appendix section focuses on further analysis and results for CrypTen. Table (3) shows the comparisons between CrypTen and baselines on a MiniONN architecture. It can be observed that CrypTen outperforms when pre-processing and online costs are combined. However, Delphi has the least online costs as it incorporates Neural Architecture Search to replace ReLU with quadratic activation and supports GPU computations. Moreover, Table (3) shows the contradictory results of Gazelle from the Falcon paper. Mishra *et. al.* performed their experiments on MiniONN architecture and they took Gazelle's results as they were in the paper. However, Gazelle did not show the same performance on the MiniONN architecture, and so Mishra *et. al.* compared Falcon with Gazelle on different architectures. That said, Falcon's performance was dramatically better when we compared it with a proper architecture.

From Table (6) it can be observed that CrypTen significantly outperforms the SOTA when comparing CPU executions for different convolution layers. Although, when Delphi is executed on a GPU, it reduces time consumption by 98%. In the case of CrypTen, it still does not support GPU computations.

C Extending security setting

We closely follow examples provided by CrypTen for Neural Network inference. However, these examples (i.e., *v1*) don't reflect the true sense of security as an input data file and proprietary model weights are known to Bob and Alice. Therefore, we additionally create different version of implementation (i.e., *v2*) and it's results are shown in Table (5). Furthermore, it can be easily interpreted that given two parties-based secure implementations (i.e., *v1* and *v2*) don't work if the adversary doesn't follow the threat model. For example, if the user can learn the model weights shared with his/her server then s/he can easily find out the proprietary model details. Therefore, we further experiment with CrypTen by increasing the participating

Table 3: Execution time and communication time comparison on MiniONN.

Model	Phase	Time (s)	Communication (MB)
Gazelle	Pre-processing	∼90	∼4096
	Online	∼50	∼100
Falcon	Pre-processing	∼7	∼1200
	Online	∼3	∼265
Delphi	Pre-processing	∼40	∼80
	Online	∼0.5	∼10
CrypTen	Pre-processing	∼0.3	-
	Online	∼2	∼42

Table 4: ReLU and MaxPooling communication cost across different strategies.

Layer	# inputs	Communication (MB) - Setup			Communication (MB) - Online		
		Gazelle	Falcon	CrypTen	Gazelle	Falcon	CrypTen
ReLU	1000	5.43	1.95	-	1.68	0.01	0.232
	10000	54.3	19.2	-	16.8	0.11	2.32
MaxPool	1000	15.6	1.94	-	8.39	0.01	0.76
	10000	156	20	-	83.9	0.12	7.4

Table 5: CrypTen implementation results for more security based analysis.

Model	Implement Version	# of server	Time (s)		Communication (MB)	
			Setup	Online	Setup	Online
ResNet32	v1	2	1	5.9	-	76
	v2	2	2.6	3.2	-	76
	v2	4	4.6	38.9	-	222
MiniONN	v1	2	0.3	2	-	42
	v2	2	1.2	2.9	-	42
	v2	4	2.1	18.6	-	126

servers/users to 4. If we distribute the data across many servers then it makes it hard for the adversary to learn everything. However, this affects the performance as shown in Table (5).

D Prior Works

We analyze CrypTen with Gazelle, Falcon, and Delphi. Here, Gazelle and Delphi compute convolutions and fully connected layers with linearly-homomorphic encryption, which allows secure addition and multiplication. Moreover, both of them use garbled circuits to compute the non-linear layers (i.e., activation layers). However, Delphi proposes to divide Gazelle into pre-processing and online phases, which allows Delphi to support GPU computations during the online phase. Furthermore, to avoid using costly garbled circuits for ReLU, Delphi uses Neural Architecture Search (NAS) to replace ReLU with quadratic activation without losing accuracy. In the case of the Falcon, it designs Fourier transform-based methodology. Also, Falcon uses lattice-based homomorphic encryption instead. Furthermore, they decrease the cost of the garbled circuit by using *ADDGate*, *SUBGate*, *GTGate*, and *MUXGate*. Moreover, they further optimize the strategy to compute Maxpooling and ReLU if they occur consecutively. Please refer to the Fig. (1) and Fig. (2) for working flow of Gazelle and Delphi, respectively.

Table 6: Different layer size and kernel size based CNN comparisons.

Input	Kernel	Systems	Time (ms)		Communication (MB)	
			setup	online	setup	online
16x32x32	16x3x3	Gazelle	-	1236	-	10.5
		Delphi	1236	16	10.5	0.07
		CrypTen	238	20	-	0.03
		Delphi (GPU)	1236	**0.34**	10.5	0.07
32x16x16	32x3x3	Gezelle	-	1262	-	5.24
		Delphi	1262	16	5.24	0.02
		CrypTen	240	65	-	0.1
		Delphi (GPU)	1262	**0.24**	5.24	0.02
64x8x8	64x3x3	Gezelle	-	2662	-	5.24
		Delphi	2662	15.6	5.24	0.04
		CrypTen	245	75	-	0.3
		Delphi (GPU)	2662	**0.24**	5.24	0.04

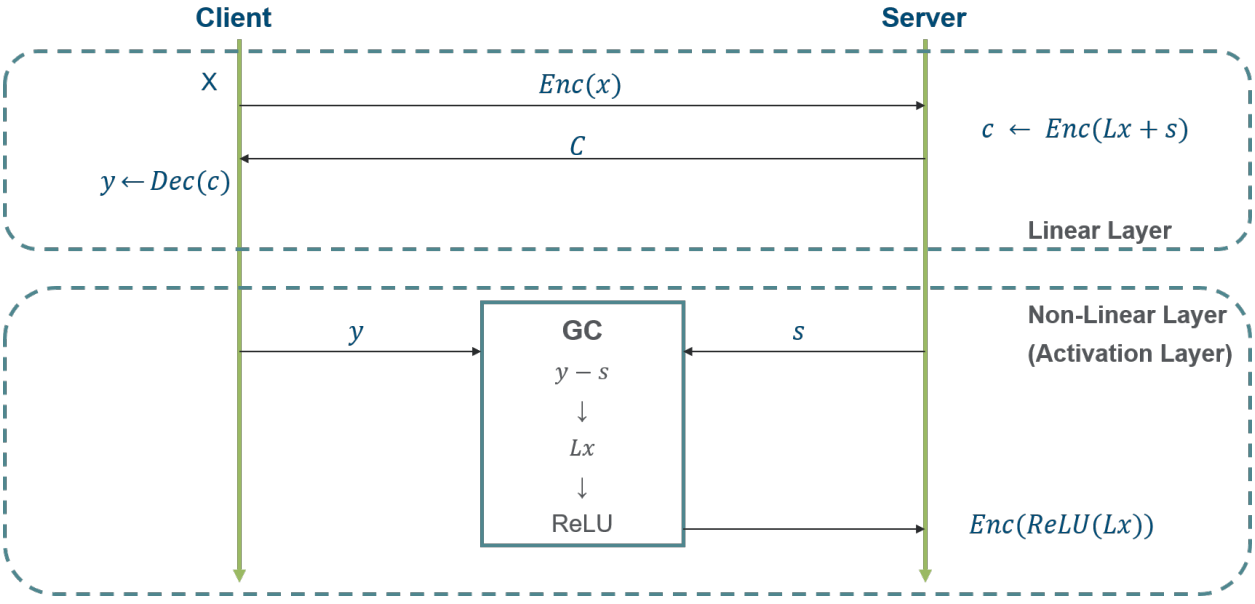

Figure 1: Gazelle workflow for secure NN inference.

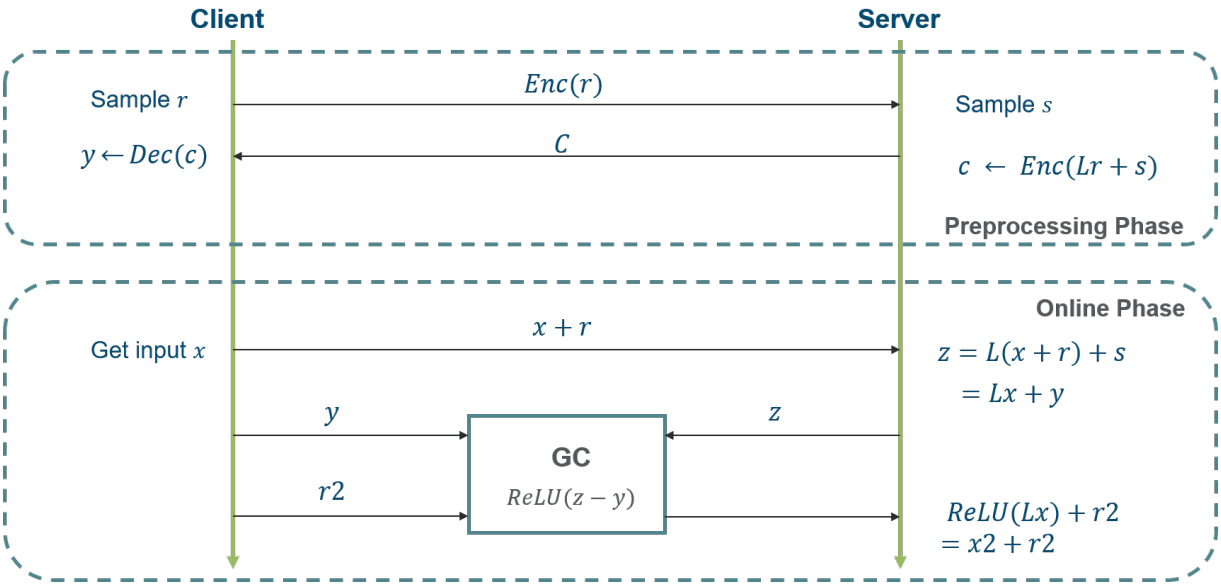

Figure 2: Delphi workflow for secure NN inference divided into pre-processing and online phase.