A Note on "Secure Quantized Training for Deep Learning"

Marcel Keller
marcel.keller@data61.csiro.au
CSIRO's Data61

Ke Sun
ke.sun@data61.csiro.au
CSIRO's Data61 & The Australian National University

August 11, 2023

Abstract

Keller and Sun (ICML'22) have found a gap in the accuracy between floating-point deep learning in cleartext and secure quantized deep learning using multi-party computation. We have discovered that this gap is caused by a bug in the implementation of max-pooling. In this note, we present updated figures to support this conclusion. We also add figures for another network on CIFAR-10.

1 Introduction

Multi-party computation (MPC) is a way to compute on distributed data without revealing it to other parties [Lin20]. In MPC, the common floating-point arithmetic is considerably slower than using quantization. Instead, Keller and Sun [KS22] represent a real number x as an integer $\lfloor x \cdot 2^f \rceil$, where $f > 0$ is a prescribed precision and $\lfloor \cdot \rceil$ means rounding to the nearest integer. They found no improvement in going beyond $f = 16$ (i.e. $f > 16$) for LeNet. They also established a gap in the accuracy between cleartext floating-point computation and quantized computation of 99.2%/99.0% for SGD and 99.3%/99.2% for AMSGrad. For an AlexNet-like network with CIFAR-10, the gap is 69%/65% with Adam. This is somewhat surprising for two reasons. First, any positive single-precision floating-point number can be represented with $f = 127$. Second, Keller and Sun use precise computation of exponentiation and division instead of rough approximation. Since then, we have identified a bug in the back-propagation of max-pooling in their implementation in MP-SPDZ [Kel20]. The bug had the effect that the gradient would always be back-propagated to one corner of the pooling window rather than the position of the maximum input

Table 1: LeNet Benchmarks in the three-party LAN setting with one corruption. Accuracy "N/A" means that the accuracy figures were not given or computed in a way that does not reflect the secure computation. (*) [WTB+21] only implemented their online phase.

	s/epoch	GB/ep.	Acc. (# ep.)	Precision (f)
[WGC19]	7,188	N/A	N/A	13
[WTB+21]*	1,412	162	N/A	13
[TKTW21]	1,036	534	94.0% (5)	20
[KVH+21]	10,940	N/A	96.7% (4)	16
Ours (SGD)	283	280	99.0% (5)	16
Ours (AMSGrad)	409	603	99.1% (5)	16

signal. Closing the gap between quantized and floating-point computation means that there is no need for more expensive computation in MPC in order to match the accuracy of cleartext computation. The updated implementation is available on the same repository as the previous version.[1]

2 Updated Figures

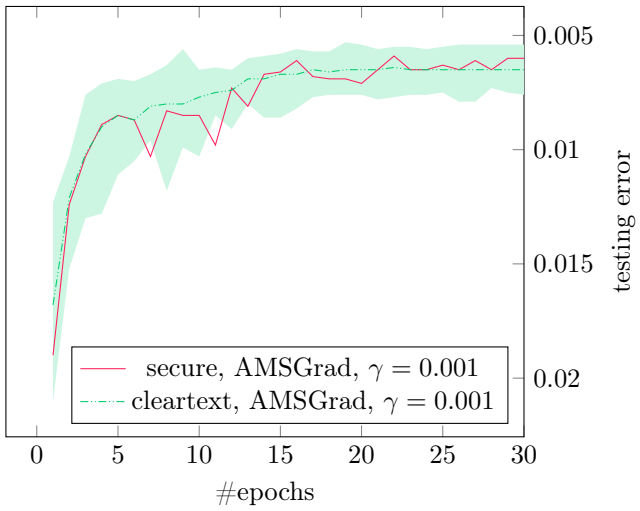

Figure 1: Comparison of cleartext training and secure training for LeNet with $f = 16$ and probabilistic truncation. γ is the learning rate, and the shaded area shows the range of 20 independent executions.

Figure 1 compares LeNet training on MNIST between cleartext floating-point computation and secure quantized computation, and Figure 2 does so for

[1] https://github.com/csiro-mlai/deep-mpc

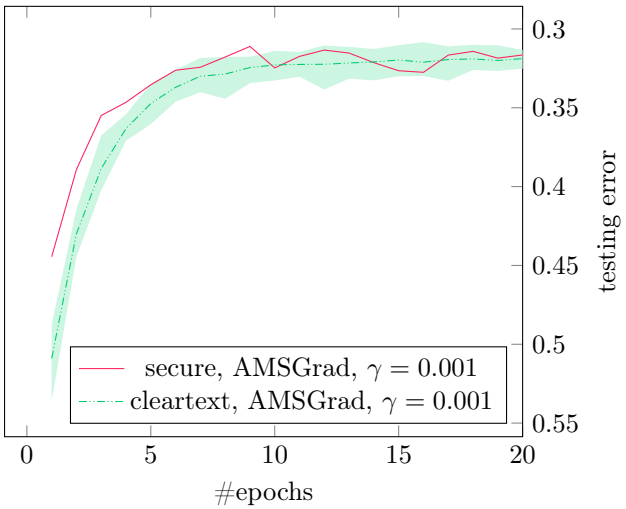

Figure 2: Comparison of cleartext training and secure training for FALCON's AlexNet-like network with $f = 16$ and probabilistic truncation. γ is the learning rate, and the shaded area shows the range of 20 executions.

Table 2: Time (seconds) and communication (GB) per epoch, accuracy after ten epochs, and fixed-point precision for CIFAR-10 training in the three-party LAN setting with one corruption. [TKTW21] do not train from scratch, which is why we do not include their accuracy figure. (*)[WTB+21] only implemented their online phase.

	s/epoch	GB/ep.	Acc. (# ep.)	Precision (f)
[WTB+21]*	3,156	80	N/A	13
[TKTW21]	1,137	535	N/A	20
Ours (SGD)	783	740	66.5%	16
Ours (AMSGrad)	1,630	3,167	67.6%	16

the CIFAR-10 and AlexNet-like network proposed by the authors of FALCON [WTB+21]. The hyperparameters were chosen to match [KS22]. $\gamma = 0.001$ is a common default choice for AMSGrad, and probabilistic truncation is the most efficient truncation method. Both figures show that there is not gap between the two, and the secret computation is within the range of 20 cleartext executions after 10 and 15 epochs, respectively. Tables 1 and 2 contain updated timing results corresponding to a part of Table 3 and Table 6 in [KS22], respectively. They show that the bugfix does not increase the complexity. Instead, optimizations to MP-SPDZ actually decrease the cost.

3 New Figures

The FALCON adaption of AlexNet reduces several convolutional layers to fully-connnected layers by the fact that the kernel is at least as large as the input channels. In Figure 3, we propose another AlexNet-like network, which has fewer parameters but results in better accuracy on CIFAR-10. Figure 4 shows that the one run of secure training is entirely within the range of 20 runs of cleartext training after five epochs.

```
nn.Sequential(
    nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=2),
    nn.ReLU(inplace=inplace),
    nn.MaxPool2d(kernel_size=2),
    nn.Conv2d(64, 96, kernel_size=3, padding=2),
    nn.ReLU(inplace=inplace),
    nn.MaxPool2d(kernel_size=2),
    nn.Conv2d(96, 96, kernel_size=3, padding=1),
    nn.ReLU(inplace=inplace),
    nn.Conv2d(96, 64, kernel_size=3, padding=1),
    nn.ReLU(inplace=inplace),
    nn.Conv2d(64, 64, kernel_size=3, padding=1),
    nn.ReLU(inplace=inplace),
    nn.MaxPool2d(kernel_size=3, stride=2)
    nn.Flatten(),
    nn.Linear(1024, 128),
    nn.ReLU(inplace=inplace),
    nn.Linear(128, 256),
    nn.ReLU(inplace=inplace),
    nn.Linear(256, num_classes),
)
```

Figure 3: Our AlexNet-like network for CIFAR-10. The network architecture is specified in PyTorch syntax.

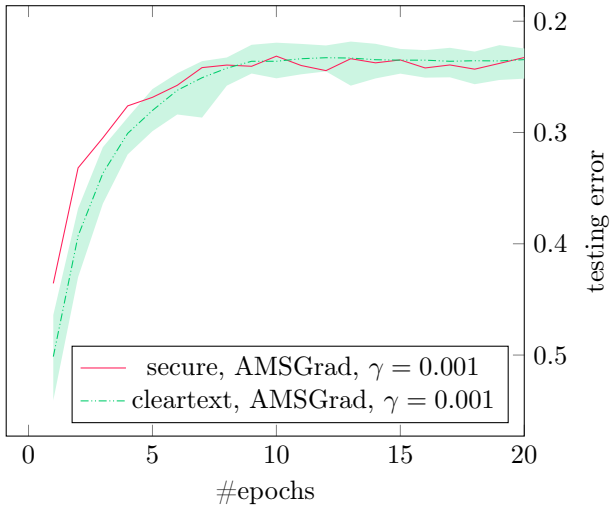

Figure 4: Comparison of cleartext training and secure training for our AlexNet-like network with $f = 16$ and probabilistic truncation. γ is the learning rate, and the shaded area shows the range of 20 executions.

References

[Kel20] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1575–1590. ACM Press, November 2020.

[KS22] Marcel Keller and Ke Sun. Secure quantized training for deep learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10912–10938. PMLR, 17–23 Jul 2022.

[KVH+21] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubhabrata Sengupta, Mark Ibrahim, and Laurens van der Maaten. CrypTen: Secure multi-party computation meets machine learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[Lin20] Yehuda Lindell. Secure multiparty computation. *Commun. ACM*, 64(1):86–96, dec 2020.

[TKTW21] Sijun Tan, Brian Knott, Yuan Tian, and David J. Wu. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *IEEE Symposium on Security and Privacy*, 2021.

[WGC19] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-party secure computation for neural network training. *PoPETs*, 2019(3):26–49, July 2019.

[WTB+21] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *PoPETs*, 2021(1):188–208, January 2021.