CRYPTO-ORIENTED NEURAL ARCHITECTURE DESIGN

Avital Shafran *Gil Segev* *Shmuel Peleg* *Yedid Hoshen*

School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel

ABSTRACT

Sending private data to Neural Network applications raises many privacy concerns. The cryptography community developed a variety of secure computation methods to address such privacy issues. As generic techniques for secure computation are typically prohibitively expensive, efforts focus on optimizing these cryptographic tools. Differently, we propose to optimize the design of crypto-oriented neural architectures, introducing a novel Partial Activation layer. The proposed layer is much faster for secure computation as it contains fewer non linear computations. Evaluating our method on three state-of-the-art architectures (SqueezeNet, ShuffleNetV2, and MobileNetV2) demonstrates significant improvement to the efficiency of secure inference on common evaluation metrics.

Index Terms— Secure Inference; NN Architecture;

1. INTRODUCTION

Deep neural networks are revolutionizing many applications, but wider use may be slowed down by privacy concerns. As an example, a hospital may wish to preserve privacy of its data when using external medical image diagnosis services. On the other hand, the diagnosis company may not be willing to share its neural network model with the hospital to safeguard its intellectual property. Such privacy conflicts could prevent hospitals from using neural network services for improving healthcare. The ability to evaluate neural network models on private data will allow the use of neural network services in privacy-sensitive applications.

The privacy challenge has attracted significant research in the cryptography community. Cryptographic tools were developed to convert any computation to secure computation, i.e. computation where the view of each involved party is guaranteed not to reveal any non-essential information on the inputs of the other parties. The deep learning setting consists of two parties, one providing the data and the other providing the neural network model. Secure computation is typically slower than non-secure computation and requires much higher networking bandwidth. Recently, various approaches were proposed for secure computation of neural networks. However, due to their computational complexity, these approaches have been limited to very small networks having little applicability.

Our Contribution. Instead of using existing architectures and optimizing the cryptographic protocols, we propose to design new neural network architectures that are crypto-oriented. As non-linear activations, such as ReLU, are very expensive for secure computation, we introduce *partial activation* layers having fewer non linearities. Each layer is split into two branches, and non-linear activation is applied only on one branch. As different activation layers have different effects on accuracy, ratio of removed activations should be carefully selected. For layers whose removal makes no significant impact on accuracy we can use a 0%-partial activation layer, i.e. removing the activation layer completely.

Using *partial activation* layers, we present new crypto-oriented architectures based on three popular (non crytpo-oriented) efficient neural network architectures: MobileNetV2 [1], ShuffleNetV2 [2] and SqueezeNet [3]. Our new architectures are significantly more efficient than their non-crypto-oriented counterparts, with a minor loss of accuracy.

2. BACKGROUND

2.1. Privacy-Preserving Machine Learning

In privacy-preserving machine-learning inference, a pre-trained neural network is transformed to process (possibly interactively) encrypted data. The network's output should also be encrypted, and only the data owner can decode it. This enables private data, such as medical records, to be used with services of external model providers.

Existing privacy-preserving inference methods use three cryptographic approaches, developed by the cryptography community in the context of secure computation: Homomorphic encryption, garbled circuits, and secret sharing.

Given a neural network N with depth k, it can be represented by a list of composed layers:

$$N(X) = F_k \circ F_{k-1} \circ ... \circ F_1(X) \qquad (1)$$

where F_i is the i^{th} layer of the network, and X is the input to

This research has been supported by the Israel ministry of Science and Technology, by the Israel Science foundation, and by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253).

	CIFAR-10 / 100				MNIST / FASHION			
Model	**Accuracy** CIFAR10 / 100	**Comm. (MB)**	**Rounds**	**Runtime (sec)**	**Accuracy** MNIST / FASHION	**Comm. (MB)**	**Rounds**	**Runtime (sec)**
Squeeze-orig	92.49 / 70.41	327.2	393	14.59	99.27 / 94.05	248.07	393	14.36
Squeeze-ours	91.87 / 69.7	149.59	232	9.03	99.08 / 93.29	66.77	152	6.34
Shuffle-orig	92.6 / 70.95	311.63	484	24.37	99.26 / 93.51	249.36	484	23.8
Shuffle-ours	92.5 / 70.07	157.63	294	14.88	99.23 / 93.4	104.3	294	11.4
Mobile-orig	94.49 / 74.8	1926.34	806	41.01	99.23 / 94.51	1517.37	806	38.41
Mobile-ours	93.44 / 72.61	403.52	296	17.11	99.25 / 94.29	250.42	296	16.12

Table 1. Comparison of performance on secure classification using a few known networks (SqueezeNet, ShuffleNetV2, and MobileNetV2), before and after our proposed crypto oriented modifications. Our modification provides substantial increase of efficiency with a minor reduction of accuracy. While the accuracy is noted separately for each dataset, the complexity measures for the two CIFAR datasets are almost the same, as well as for the MNIST and Fashion-MNIST datasets.

the network. Using the above cryptographic tools, each layer can be transformed into a privacy-preserving layer $\hat{F}_i$ such that given the encoding $\hat{X}$ of a private input X the output of:

$$\widehat{N(X)} = \hat{F}_k \circ \hat{F}_{k-1} \circ ... \circ \hat{F}_1(\hat{X}) \quad (2)$$

$\widehat{N(X)}$ is encrypted as well, and can be decoded only by the owner of X to compute $N(X)$. Due to the complexity of secure computation, the above approaches are practical mainly for simple computations.

Homomorphic encryption (HE) [4, 5] allows to compute an arbitrary function f on an encrypted input, without decryption or knowledge of the private key. HE was used in the CryptoNets system [6], but this approach significantly increased the overall inference time. Optimized methods based on HE appear in [7, 8, 9, 10, 11].

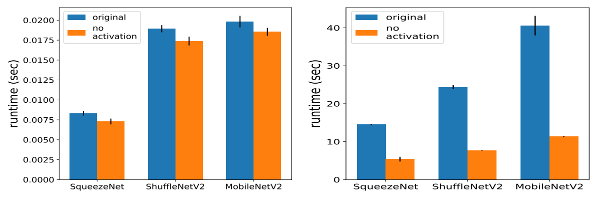
Garbled circuits can be roughly viewed as a one-time variant of HE. For two parties, A and B, where A holds a function f (corresponding to a single layer of the network) and B holds an input x, the function f is transformed by A into a garbled circuit that computes f on a single encoded input, provided by B [12, 13, 14]. More precise descriptions appear in [15, 16].

Secret sharing schemes [17, 18] provide the ability to share a secret between multiple parties, such that all parties can evaluate a function, e.g. a neural network layer, over their share [19, 20, 21, 22]. The result can be reconstructed by combining the shares of any "authorized" subset of parties.

2.2. Efficient Neural Network Architecture Design

Real world tasks require both accuracy and efficiency, sometimes under different constraints, e.g. hardware. Much work has focused on designing neural network architectures, optimally trading off accuracy and efficiency.

SqueezeNet [3] focused at reducing the number of model parameters. MobileNetV2 [1] utilizes depthwise separable convolution to improve efficiency, and proposed the efficient



(a) non-secure (b) secure

Fig. 1. Effects of removing all activation layers on the complexity of inference. (a) Negligible effect on non-secure inference. (b) A drastic reduction of complexity in secure inference. Comparison done on SqueezeNet, ShuffleNetV2 and MobileNetV2 using all datasets.

inverted residual with linear bottleneck block. ShuffleNetV2 [2] proposed guidelines for the design of efficient deep neural network architectures and utilizes pointwise group convolutions to reduce complexity and *channel shuffle* to help information flow across feature channels.

3. DESIGNING CRYPTO-ORIENTED NETWORKS

Our goal is to design neural networks that can be computed efficiently in a secure manner for providing privacy-preserving inference mechanisms. We propose a novel *partial activation* layer that exploits the trade-offs that come with the complexity of the cryptographic techniques enabling privacy-preserving inference.

In non-secure computation, the cost of affine operations like addition or multiplication is almost the same as the cost of non-linearities such as maximum or ReLU. Efficient network designs therefore try to limit the number and size of network layers, not taking into account the number of non-linearities.

The situation is different for privacy-preserving neural networks, as secure computation of non-linearities is much more expensive. Homomorphic encryption methods approximate ReLU with polynomials, higher polynomial degrees

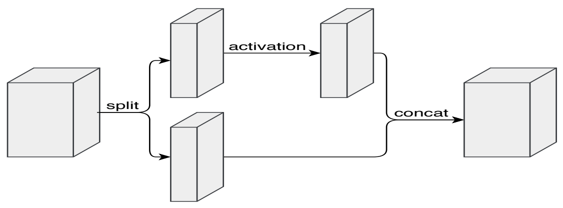


Fig. 2. In Partial Activation layers, the channels are split and activation is applied only to a subset of the channels, and not applied to the other channels.

are needed for better accuracy, increasing computational complexity. Garbled circuits and secret sharing methods present lighter-weight protocols, but have high communication and round complexities. As a result, the number of non-linearities is an important consideration in the design of efficient privacy-preserving networks.

Fig. 1 illustrates the remarkable runtime difference between secure and non-secure inference. We evaluate three popular architectures - SqueezeNet, ShuffleNetV2 and MobileNetV2 - on the CIFAR-10, CIFAR-100, MNIST and Fashion-MNIST datasets. We can see that in the secure case, removal of all ReLU activations results in more than a 60% runtime reduction, while in the non-secure case the reduction is only around 10%. This highlights that the number of non-linearities must be taken into account in crypto-oriented neural architecture design.

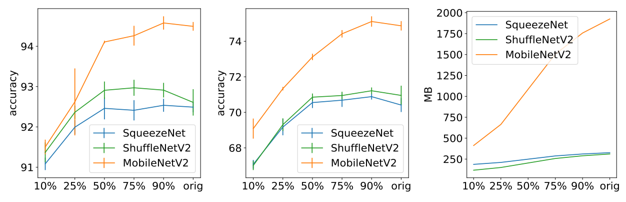
In the above, ReLU is only used as an illustration. This applies identically to all other non-linear activation layers such as Leaky-ReLU, ELU, SELU, ReLU6, although the exact numerical trade-offs may differ slightly.

Partial activation layers. In order to reduce the number of non-linear operations we propose a novel *partial activation* layer, illustrated in Fig. 2. *Partial activation* splits the channels into two branches, and non-linear activations are applied only on one branch. By using *partial activation* we can reduce the number of non-linear operations, while keeping the non-linearity of the model. Our experiments show that this operation results in attractive accuracy-efficiency trade-off, dependent on the amount of non-linear channels. It is also beneficial to remove complete activation layers where they do not improve the network accuracy. Linear layers were studied by [23, 24, 1] and have been shown to even have a positive effect over the accuracy.

4. EXPERIMENTS

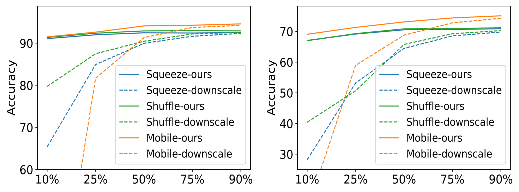
Experiments show that our crypto-oriented architectures have better trade-offs between efficiency and accuracy in privacy-preserving inference compared to standard architectures.

Efficiency evaluation metrics. The fundamental complexity measures for secure computations are the communication and the round complexities. Runtime varies with implementation, and therefore is used in only two cases: removing all activations in Fig. 1, and using all proposed optimizations in Table 1.



(a) CIFAR-10 (b) CIFAR-100 (c) Communication

Fig. 3. Comparison between different partial activation ratios on SqueezeNet, ShuffleNetV2, and MobileNetV2, in terms of accuracy (a-b) and communication complexity (c). Ratio of 50% is a good balance between accuracy and efficiency.



(a) CIFAR-10 (b) CIFAR-100

Fig. 4. Comparison between partial activation and downscaled network width (i.e. removing the no-activation branch). Results demonstrate that the channels with no ReLU activation contribute significantly to accuracy.

Implementation details. We examine privacy-preserving inference, and assume that networks are trained in the clear. Models are "encrypted" for inference to measure accuracy, runtime, round complexity and communication complexity on private data. We use the tf-encrypted framework [25] to convert trained neural networks to privacy-preserving networks. For runtime measurements we used an independent server for each party in the computation, each consisting of 30 CPUs and 20GB RAM.

We evaluated on the CIFAR-10 and CIFAR-100 datasets. Experiments were conducted on downscaled versions of three popular efficient architectures - SqueezeNet [3], ShuffleNetV2 [2] and MobileNetV2 [1].

4.1. Partial Activation

We tested different partial activation ratios between the channels in the non-linear branch and the total number of channels. Results are presented in Fig. 3. Activations on 50% of channels appears to be a good trade-off between efficiency and accuracy. Round complexity was not used in this comparison as we assume that element-wise non-linearities can all be computed in parallel, giving a constant round complexity for each layer regardless of the number of its non-linearities.

Model	**Accuracy** CIFAR-10 / 100	**Comm.** (MB)	**Rounds**
Squeeze-1	90.54 / 64.72	189.36	233
Squeeze-2	93.15 / 72.37	309.97	313
Squeeze-0.5-1	90.4 / 66.04	180.75	233
Squeeze-0.5-2	92.66 / 70.76	241.05	313
Squeeze-none	86.95 / 60.03	172.13	153
Squeeze-orig	92.49 / 70.41	327.2	393
Shuffle-1	92.83 / 71.02	219.01	294
Shuffle-2	92.9 / 71.37	188.86	324
Shuffle-0.5-1	92.5 / 70.07	157.63	294
Shuffle-0.5-2	92.19 / 69.53	142.55	324
Shuffle-none	83.26 / 46.8	95.25	134
Shuffle-orig	92.6 / 70.95	311.63	484
Mobile-1	93.92 / 74.35	1168.2	466
Mobile-2	94.13 / 74.17	1003.02	486
Mobile-0.5-1	93.66 / 72.77	706.54	466
Mobile-0.5-2	93.28 / 72.86	623.95	486
Mobile-none	78.45 / 51.45	244.88	146
Mobile-orig	94.49 / 74.8	1926.34	806

Table 2. Effects of the removal of activations in network blocks. Tested networks are built of blocks having two activation layers each. In $Network$-i we keep only the $i'th$ activation layer in each block, and remove the other activation layer. In $Network$-0.5-i we replace the $i'th$ activation layer with a 50% partial activation layer, and remove the other activation layer. We also show the original blocks and the removal of all activation layers in each block. The table shows that removing most of the activations in each block has minimal effect on accuracy but substantially increase speed.

Scaling down network width, i.e. reducing the number of channels in each layer (equivalent to dropping the no-activation branch), has also been tested. Fig. 4 shows that scaling down is inferior to partial activation on original width, demonstrating the importance of channels with no activation.

Activation removal Our tested networks consists of blocks, each block has two activation layers. Table 2 demonstrates that one activation layer in each block can be removed with a minimal loss of accuracy. Table 2 also shows that by also using 50%-partial activation on the remaining layer the communication and round complexities can be substantially improved with only a minor change in accuracy.

4.2. Alternative Non-Linearities

In addition to the removal of non-linearities, we investigated the cost of several commonly used non-linearities and propose more crypto-oriented alternatives.

Pooling. Previous works show that replacing max pooling with average pooling or strided convolutions has minimal effect on network accuracy. In secure inference max pooling is an expensive non-linear operation, while average pooling is much faster. The effect of using max pooling against average pooling is shown in Table 3. Average pooling is much more efficient than max pooling while not affecting accuracy.

Model	**Accuracy** CIFAR-10 / 100	**Comm.** (MB)	**Rounds**
Squeeze-Avg	91.74 / 68.69	234.96	312
Squeeze-Max	92.16 / 68.98	326.51	528
Shuffle-Avg	92.35 / 70.19	310.89	484
Shuffle-Max	92.92 / 70.39	400.92	781
Mobile-Avg	94.58 / 75.33	1925.42	806
Mobile-Max	93.40 / 73.87	2045.94	1022
Squeeze-ReLU	92.49 / 70.41	326.41	393
Squeeze-ReLU6	92.61 / 70.39	538.62	653
Shuffle-ReLU	92.61 / 70.95	310.89	484
Shuffle-ReLU6	92.76 / 70.42	550.04	854
Mobile-ReLU	94.28 / 74.93	1053.79	456
Mobile-ReLU6	94.49 / 74.87	1925.42	806

Table 3. Comparison between alternative non-linearities on SqueezeNet, ShuffleNetV2 and MobileNetV2, in terms of accuracy and communication and round complexities. Average pooling has similar accuracy to max pooling, but is much more efficient. Accuracy for both ReLU and ReLU6 is almost identical, while ReLU being less inefficient.

ReLU6. A common variant of ReLU is ReLU6 [26], which limits the activation value to a maximum of 6. The secure computation of ReLU6, having two comparisons, is double the cost of standard ReLU. In Table 3 we show that ReLU vs. ReLU6 has minimal effect on accuracy.

4.3. Crypto-Oriented Neural Architectures

We use our approach to optimize secure inference of the state-of-the-art architecture MobileNetV2 [1]. Replacing all activation layers in the inverted residual with linear bottleneck block with 50%-partial activation layers, removing the first activation layer completely and replacing the ReLU6 activation with ReLU, yields an improvement of 79% in communication complexity and 63.4% in round complexity.

Table 1 presents our results for the crypto-oriented versions of three state-of-the-art architectures: SqueezeNet [3], ShuffleNetV2 [2] and MobileNetV2 [1].

5. CONCLUSION

In order to increase efficiency of privacy-preserving secure neural network inference, we proposed to use partial activation layers, selective removal of complete activation layers, and avoiding the use of expensive non-linear variants. We demonstrate that efficiency is increased substantially with negligible loss of accuracy.

6. REFERENCES

[1] M. Sandler, A.w Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR'18*, 2018, pp. 4510–4520.

[2] N. Ma, X. Zhang, H. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *ECCV'18*, 2018, pp. 116–131.

[3] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5MB model size," *arXiv:1602.07360*, 2016.

[4] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*, Stanford University, 2009.

[5] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 13, 2014.

[6] 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 *ICML'16*, 2016, pp. 201–210.

[7] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv:1711.05189*, 2017.

[8] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network.," *IACR Cryptology ePrint Archive*, vol. 2017, pp. 35, 2017.

[9] A. Sanyal, M. Kusner, A. Gascon, and V. Kanade, "Tapas: Tricks to accelerate (encrypted) prediction as a service," in *ICML'18*, 2018, pp. 4490–4499.

[10] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster cryptonets: Leveraging sparsity for real-world encrypted inference," *arXiv:1811.09953*, 2018.

[11] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Int. Cryptology Conference*. Springer, 2018, pp. 483–512.

[12] B. Rouhani, M. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," in *Proc. Design Automation Conf. DAC'18*. ACM, 2018, pp. 1–6.

[13] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *USENIX Security Symposium*, 2018, pp. 1651–1669.

[14] M. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "Xonn: Xnor-based oblivious deep neural network inference.," *USENIX Security Symposium*, pp. 1501–1518, 2019.

[15] A. Yao, "How to generate and exchange secrets," in *FOCS*. IEEE, 1986, pp. 162–167.

[16] Yehuda Lindell and Benny Pinkas, "A proof of security of yao's protocol for two-party computation," *Journal of cryptology*, vol. 22, no. 2, pp. 161–188, 2009.

[17] A. Shamir, "How to share a secret," *Comm. of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[18] A. Beimel, "Secret-sharing schemes: A survey," in *Proc. of the 3rd Int. Workshop on Coding and Cryptology*, 2011, pp. 11–46.

[19] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE Symp on Security and Privacy (SP)*, 2017, pp. 19–38.

[20] J. Liu, . Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *SIGSAC Conf. on Computer and Communications Security*. ACM, 2017, pp. 619–631.

[21] M. Riazi, C. Weinert, O. Tkachenko, E. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. 2018 Asia Conf. on Computer and Communications Security*. ACM, 2018, pp. 707–721.

[22] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training," *Proc. on Privacy Enhancing Technologies*, vol. 2019, pp. 26–49, 2019.

[23] X. Dong, G. Kang, K. Zhan, and Y. Yang, "Eraserelu: A simple way to ease the training of deep convolution neural networks," *arXiv:1709.07634*, 2017.

[24] G. Zhao, Z. Zhang, H. Guan, P. Tang, and J. Wang, "Rethinking relu to train better cnns," in *ICPR'18*, 2018, pp. 603–608.

[25] "TF-Encrypted: Machine Learning on Encrypted Data in TensorFlow," `https://tf-encrypted.io/`, 2019.

[26] A. Krizhevsky, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, 2010.