GATENET: BRIDGING THE GAP BETWEEN BINARIZED NEURAL NETWORK AND FHE EVALUATION

Cheng Fu, Hanxian Huang
UC San Diego
{cfu,hah008}@eng.ucsd.edu

Xinyun Chen
UC Berkeley
xinyun.chen@berkeley.edu

Jishen Zhao
UC San Diego
jzhao@eng.ucsd.edu

ABSTRACT

With the advance of deep learning (DL) on sensitive classification tasks, recent works propose privacy-preserving DL evaluation methods to protect the client data privacy. However, DL models are also valuable as it is costly to collect training data. Fully homomorphic encryption (FHE) offers a promising security solution to preserve the privacy of both parties. Previous work finds that binarized neural networks (BNNs) are suitable for FHE which evaluates a function represented in logic gates. Yet, the computation costs of previous BNNs under FHE are very large. In this work, we first design evaluation circuits to enable any BNN inference under FHE. By leveraging a new BNN cost metric for FHE, called *gate ops*, we identify that the computation bottleneck of BNNs under FHE is the adder tree. Thus, we propose GateNet to reduce adder tree depth using group convolution which was originally designed to reduce multiplication. Also, as the non-linear functions are not the computation bottleneck, we apply a more advanced non-linear function to preserve task accuracy. Results show that GateNet can achieve 13.0×/28.5×/66.4× speedup over the state-of-the-art BNNs under FHE on MNIST/CIFAR-10/ImageNet datasets while keeping a high task accuracy (-0.9%/-1.2%/-4.1%). GateNet is the first work that guarantees the privacy of both model and client data on large dataset beyond MNIST.

1 INTRODUCTION

Deep Neural Networks (DNNs) have been widely used in many fields of applications such as image classification (He et al., 2016; Xie et al., 2019) and objective detection (Redmon et al., 2016). Many application domains (e.g., medical (Blecker et al., 2018), fraud detection (Ghosh & Reilly, 1994) require privacy and confidentiality in both the model vendors and clients data.

Among all possible methods for privacy-preserving DNN evaluation, such as Garble Circuit (GC) (Riazi et al., 2019) and differential privacy (DP) (Chase et al., 2017), Fully Homomorphic Encryption (FHE) is an ideal solution. Given encryptions $E(x)$ and $E(y)$, FHE allows the computation of function f and yields $E(f(x,y))$ without intermediate decryption of the ciphertext.

Many works have been proposed to accelerate the evaluation of DNN models on FHE data. The first mainstreams of work use a leveled SomeWhat HE (SWHE). These works are implemented using HElib (Halevi & Shoup, 2014) which allows very efficient ciphertext multiplications and additions. However, in FHE, noise accumulated during computation. To evaluate unbounded computation depth on the encrypted values, SWHE must apply *bootstrapping* to remove the noise in the result and this operation is extremely slow using HElib. To prevent noise accumulation, SWHE methods do not guarantee the privacy of model parameters, i.e., they compute the result from plaintext (model) and ciphertext (client data), which yields less noise to accommodate more computation steps. Another critical issue of SWHE is the non-linear functions[1]. Previous works (Chou et al., 2018; Gilad-Bachrach et al., 2016) use low-degree polynomials to approximate the non-linear functions which incurs large task accuracy loss. Other work (Juvekar et al., 2018) switches to other encryption protocols (e.g., Garble circuit) to do non-linear function which incurs many engineering difficulties.

[1] HELib Halevi & Shoup (2014) for SWHE is built on BGV protocol which only allows polynomial operations

Unlike SWHE, a recent new method called TFHE (Chillotti et al., 2020) provides a very efficient bootstrapping operation ($\sim 16ms$/per gate). As a downside, this has to be applied after every gate computation. The bootstrap means we can compute DNN without computation steps limitation like SWHE. Also, the model privacy can be guaranteed as we can do computations between ciphertexts. Previous works (Sanyal et al., 2018; Bourse et al., 2018) show that binary neural network can be easily adapted to this encryption protocol. Yet, these works only test on MNIST for proof-of-concept and some tricks proposed is not suitable for recent BNN advances.

In this paper, we first propose a new set of evaluation circuits to enable FHE inference for any modern BNN neural network. We find traditional BNN computation complexity metrics (i.e., BOPs (Martinez et al., 2020) or FLOPs) are not suitable for BNN inference under TFHE (Appendix D). As such, we define a new metric called *gate ops*, which is the required gate operations in a BNN inference using our proposed FHE evaluation circuits. Using *gate ops*, we identify that the computation bottleneck of BNN evaluation under TFHE is the addition or popcount operation (Fig. 1(a)). The number of gates in the adder tree for popcount grows exponentially to the length of input bits and takes up to 99.1% of the *gate ops* in XNOR-LeNet-5 (Fig. 1(b)). Inspired by this observation, we design a new BNN architecture called GateNet. GateNet leverages the group convolution (Zhang et al., 2018) to reduce the length of input bits for popcount. Also, because the non-linear circuits are not the bottleneck of FHE inference for BNN, GateNet incorporates a more advanced non-linear function (Liu et al., 2020b) to mitigate the accuracy loss compared to floating-point DNN. Our result shows that GateNet can yield 13.0×/28.5×/66.4× speedup compared to the state-of-the art BNNs under TFHE inference on MNIST/CIFAR-10/ImageNet while preserving a high task accuracy.

2 Privacy Guarantees

A common way for modern DNN evaluation is through machine learning as a service (MLaaS) provided by cloud servers as shown in Appendix C Fig. 4. In public-key encryption, the model vendors send the evaluation function f and encrypted model parameters $E(M, k_{pub})$ to cloud, the client i sends its data $E(x_i, k_{pub})$ to the cloud. By leveraging FHE, the cloud can compute $\square$ which satisfies $E(x_i) \square E(M) = E(f(x_i, M), k_{pub})$. After the cloud finishes the model inference, the client can decrypt the result using the private key k_{pri}: $D(E(f(x_i, M), k_{pub}), k_{pri})) = f(x_i, M)$.

During the evaluation, GateNet provides the following privacy guarantees: **(P1)** The cloud server (and any other party) does not know the client data. **(P2)** The cloud server (and any other party) does not know the model parameters except for the evaluation function f.

Leveraging BNN and TFHE, GateNet has computation advantages which are listed below: **(C1)** Require no third party in the computation. **(C2)** Require only two rounds of communications. The client and model vendors can stay offline during the model evaluation. **(C3)** Allow the server to update evaluation function f without communicating with clients. **(C4)** Apply no computation approximation. That means the computation result will not be affected by the noise accumulation in the encryption protocol. **(C5)** Support any non-linear layers. **(C6)** Report evaluation on CIFAR-10 and ImageNet to show the scalability of proposed method. The comparison of GateNet and previous methods regarding the above privacy guarantees and computation advantages are shown in Table 1.

Table 1: Comparison of privacy-preserving DL evaluation works in terms of the privacy guarantees and computation advantages.

Prior Work	Protocol	Privacy Guarantee		Computation Advantages					
		P1	P2	C1	C2	C3	C4	C5	C6
Cryptonet (Gilad-Bachrach et al., 2016), Faster CryptoNet (Chou et al., 2018)	HE	✓		✓	✓	✓	✓		
Gazella (Juvekar et al., 2018)	HE, GC	✓		✓			✓		
MiniONN (Liu et al., 2017), SecureML (Mohassel & Zhang, 2017)	HE, GC	✓		✓			✓		
XONN (Riazi et al., 2019), DeepSecure (Rouhani et al., 2018)	GC	✓		✓			✓	✓	
DINN (Bourse et al., 2018)	HE	✓		✓	✓	✓			
SHE (Lou & Jiang, 2019)	HE	✓		✓	✓	✓	✓	✓	✓
TAPAS (Sanyal et al., 2018)	HE	✓	✓	✓	✓	✓	✓	✓	
GateNet	HE	✓	✓	✓	✓	✓	✓	✓	✓

3 Method

Evaluation Circuits. BNNs (Courbariaux et al., 2016) apply binarized (+1/-1) activation and weights in convolution and convert the original multiplications into XNOR operations for efficient evaluation. Traditional BNNs use a sign function as the non-linear function between each convolution layer. Most recently, a more accurate BNN named ReActNet (Liu et al., 2020b) mitigates the accuracy loss using new activation functions (i.e., PRelu and Rsign) as shown in Fig. 2 (b, c). In this work, we design a set

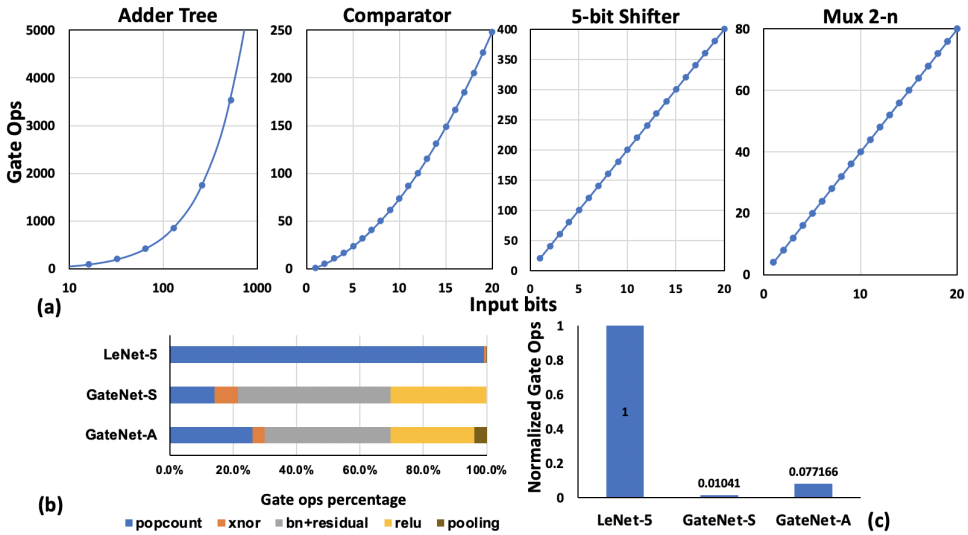

Figure 1: (a) Gate ops v.s. input bit length of adder tree/comparator/shifter/MUX 2-n. (b) A breakdown of gate ops in LeNet-5 and GateNet. (c) normalized gate ops comparison between LeNet-5 and GateNet.

of evaluation circuit (Fig. 2) for traditional sign functions and also recent PRelu/Rsign to enable any BNN inference using TFHE. By quantizing the β in the non-linear function and batch normalization in Fig. 2 to 2^n, integer multiplications can be fulfilled by using n-bit shifters. The circuit-level details and other evaluation circuits (e.g., traditional sign/pooling) are shown in Appendix A.

Identifying Computation Bottleneck Using Gate ops. As discussed in Sec. 1, traditional metrics (i.e., BOPs (The number of XNOR operations) or FLOPs (Floating-point operations) for evaluating BNN or DNN is not able to identify the bottleneck of BNN inference in FHE. In Floating-/Fixed-point DNN evaluation, the cost of multiplication dominates other computation operations. Yet, in the case of BNN for FHE, the multiplication can be efficiently implemented by using a low-cost XNOR gate while the popcount dominates the computation.

In this paper, we propose a new metric to evaluate BNN called *gate ops*. By building up the evaluation circuit for a given model architecture, we can compute the total number of gates to evaluate the BNN. *Gate ops* reveals the real latency for evaluation BNN under FHE. Breakdowns of gate ops for XNOR-LeNet-5[2] and GateNet is shown in Fig. 1 (b). The gate ops for popcount dominates other operations under TFHE due to the exponential size of the required adder tree (Fig. 1(a)). The key insight of designing GateNet is to reduce the length of input bits to the popcount. This will also reduce the logic gates used in non-linear and batch normalization (or affine) operations due to the lower bitwidth of the popcount result (Equations shown in Appendix A).

Design Method of GateNet. To reduce gate ops in traditional BNNs, we introduce *GateNet*, that minimizes the *gate ops* while keeping a high task accuracy. The building block of GateNet is shown in Appendix C Fig. 5. GateNet leverages group convolution in ShuffleNet (Ma et al., 2018; Zhang et al., 2018) to reduce the input bits to popcounts and in turn reduce the depth of adder trees in the BNN. Note that the original group convolution is used to reduce the overhead of multiplication in floating-point DNN. Here, the group convolution is applied for a completely different goal (e.g., reducing popcount overhead). The shuffle operations are essential to the task accuracy as it allows information to flow between groups. During GateNet's evaluation under TFHE, this can be fulfilled by simply changing the position of the ciphertext during evaluation. Also, because the non-linear functions are not the computation bottleneck in BNN under TFHE, GateNet applies more advanced non-linear functions (i.e., Psign and PRelu (Liu et al., 2020b)) to increase the task accuracy. Different from ShuffleNet building block, GateNet adds residual link inside each building block as suggested by (Liu et al., 2020a). The residual links can be fulfilled by using low bit-width adder.

For general BNNs, the first and last layers are critical to the task accuracy. We build a fully-binarized network (denoted as All or A) and a network where only the first (f)/last (l) layers are floating-point (denoted as Semi or $S_{f,l}$. S_f means only the first layer is floating-point).

[2] https://github.com/jiecaoyu/XNOR-Net-PyTorch/blob/master/MNIST/models/LeNet_5.py.

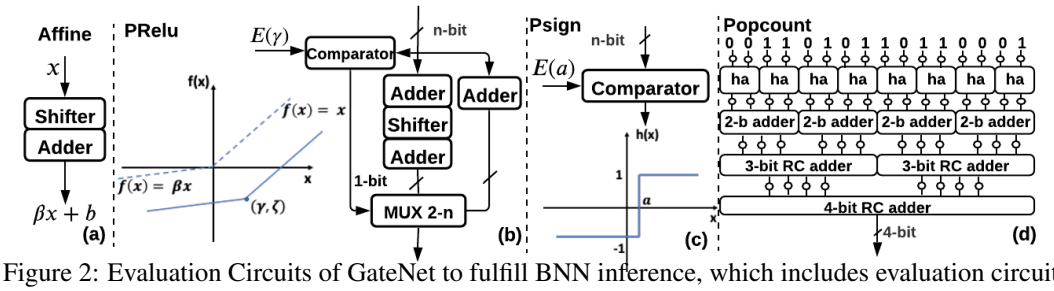

Figure 2: Evaluation Circuits of GateNet to fulfill BNN inference, which includes evaluation circuit for: (a) Batch normalization or affine function. (b) PRelu function. (c) Psign function. (d) Popcount operation. 'ha' in the adder tree denotes half-adder. 'MUX' refers to a multiplexer.

4 EXPERIMENTS

Experiment Setup and Training Details. We test the task performance of GateNet on three datasets (MNIST/CIFAR-10/ImageNet) to show its scalability. Besides the task accuracy, we also report the gate operations to prove its capability in reduce gate ops. Due to the fact that BNN is very easy to overfit (Zhu et al., 2019), we split the training process into three stages. 1) The activation is binarized and the weights are floating-point. The weight decay is 1e-5. 2) Then, we binarize both the weights and input activation (weights initialized from the first step). The weight decay is zero. 3) We further fine-tune the model by quantizing the β in each affine function and non-linear function to avoid integer multiplication. The learning rate and batch size settings for CIFAR-10/ImageNet follow the implementation of XNOR-Net and ReActNet (Liu et al., 2020b), respectively.

Results. The comparison of GateNet with previous BNNs methods for TFHE is shown in Table 2. For MNIST/CIFAR-10, GateNet can reduce the gate ops up to 92.3%/96.5% compared to binarized LeNet-5/ReActNet while achieving a high task performance (98.3%/84.6%). Previous work using BNN for TFHE (i.e., TAPAS), can not preserve the task accuracy and is not scalable to large datasets. The tricks proposed in TAPAS are orthogonal to our method. For ImageNet, GateNet can achieve 66/893× speedup over ReActNet/BENN while keep a high task accuracy (61.8%). Although the execution time of GateNet (∼4500/14000 hours) on a single 2-core CPU for ImageNet is not practical for commercial products, the execution of BNN can be easily paralleled across multiple CPUs and many GPU acceleration libraries (Dai & Sunar, 2015) for FHE are under construction.

Table 2: Comparison between GateNet with other BNN baselines under TFHE inference.

	Models	Type	Gate ops (Million)	popcount	XNOR	Other ops[†]	Projected Time (h)[*]	Accuracy [‡] (Top-1 %)
MNIST	XNOR-LeNet-5	A	953.26	944.16	5.69	3.40	4236.7	99.2
	TAPAS	A	825.71	745.80	79.62	0.30	3669.8	98.6
	GateNet-A	A	73.56	20.43	2.94	50.19	326.9	98.3
	GateNet-S	$S_{f,l}$	9.92	1.40	0.73	7.79	44.1	98.8
CIFAR-10	ReActNet (Bi-real) (Liu et al., 2020b)	$S_{f,l}$	68187.73	67462.87	610.27	114.58	303056.6	85.8
	DSQ (Gong et al., 2019)	$S_{f,l}$	68117.27	67462.87	610.27	44.12	302743.4	84.1
	DoReFaNet (Zhou et al., 2016)	$S_{f,l}$	68084.23	67462.87	610.27	11.08	302596.6	79.3
	FracBNN-1-bit (Zhang et al., 2021)	S_f	5296.66	5205.50	51.6096	39.55	23540.7	85.9
	GateNet-S-1.0×	$S_{f,l}$	882.13	669.36	54.69	158.08	3920.6	80.5
	GateNet-S-1.5×	$S_{f,l}$	2388.38	2028.19	123.07	237.12	10615.0	84.6
	GateNet-S-1.5×	S_f	3454.48	3076.75	137.21	240.51	15353.2	84.1
ImageNet	XNOR-AlexNet (Rastegari et al., 2016)	$S_{f,l}$	$2815.85_{\times 10^3}$	$2773.84_{\times 10^3}$	$28.32_{\times 10^3}$	$13.70_{\times 10^3}$	$12514.9_{\times 10^3}$	48.6
	BENN-SB-3 (Zhu et al., 2019)	$S_{f,l}$	$8447.56_{\times 10^3}$	$8321.51_{\times 10^3}$	$84.96_{\times 10^3}$	$41.10_{\times 10^3}$	$37544.7_{\times 10^3}$	53.6
	ReActNet (Bi-real) (Liu et al., 2020b)	$S_{f,l}$	$209.05_{\times 10^3}$	$206.78_{\times 10^3}$	$1.88_{\times 10^3}$	$0.40_{\times 10^3}$	$929.1_{\times 10^3}$	65.9
	Bi-RealNet-18 (Liu et al., 2020a)	$S_{f,l}$	$208.77_{\times 10^3}$	$206.78_{\times 10^3}$	$1.88_{\times 10^3}$	$0.12_{\times 10^3}$	$927.8_{\times 10^3}$	56.4
	GateNet-S-1.0×	$S_{f,l}$	$1.01_{\times 10^3}$	$0.45_{\times 10^3}$	$0.08_{\times 10^3}$	$0.48_{\times 10^3}$	$4.5_{\times 10^3}$	49.8
	GateNet-S-2.0×	$S_{f,l}$	$3.15_{\times 10^3}$	$1.94_{\times 10^3}$	$0.29_{\times 10^3}$	$0.92_{\times 10^3}$	$14.0_{\times 10^3}$	61.8

[*] We report the total execution time through linear projecting based on the evaluation time of average gate (16*ms* on a 2-core CPU).
[†] Other ops includes pooling / residual / Non-linear function / Batch Normalization.
[‡] The accuracy of other BNN methods do not quantize the β in affine functions.

5 CONCLUSIONS

In this work, we propose a set of new evaluation circuits function to enable any BNN inference under TFHE encyrption protocol. Our settings consider the privacy of both model parameters and client data. With a new proposed metric for BNN inference called *gate ops*, we identify the input bits to the popcount operations is the computation bottleneck during TFHE evaluation. As such, we build *GateNet* by leveraging group convolutions to reduce the adder tree depth and advanced non-linear functions to mitigate accuracy loss. Experiment results show that GateNet can greatly reduce the gate ops in TFHE inference while preserving a high task accuracy.

REFERENCES

Saul Blecker, David Sontag, Leora I Horwitz, Gilad Kuperman, Hannah Park, Alex Reyentovich, and Stuart D Katz. Early identification of patients with acute decompensated heart failure. *Journal of cardiac failure*, 24(6):357–362, 2018.

Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In Hovav Shacham and Alexandra Boldyreva (eds.), *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pp. 483–512. Springer, 2018. doi: 10.1007/978-3-319-96878-0_17. URL https://doi.org/10.1007/978-3-319-96878-0_17.

Melissa Chase, Ran Gilad-Bachrach, Kim Laine, Kristin E Lauter, and Peter Rindal. Private collaborative neural network learning. *IACR Cryptol. ePrint Arch.*, 2017:762, 2017.

Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

Edward Chou, Josh Beal, Daniel Levy, Serena Yeung, Albert Haque, and Li Fei-Fei. Faster cryptonets: Leveraging sparsity for real-world encrypted inference. *arXiv preprint arXiv:1811.09953*, 2018.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 3123–3131, Cambridge, MA, USA, 2015. MIT Press.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *In Advances in Neural Information Processing Systems.*, 2016.

Wei Dai and Berk Sunar. cuhe: A homomorphic encryption accelerator library. In *International Conference on Cryptography and Information Security in the Balkans*, pp. 169–186. Springer, 2015.

Sushmito Ghosh and Douglas L Reilly. Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pp. 621–630. IEEE, 1994.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pp. 201–210. PMLR, 2016.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4852–4861, 2019.

Shai Halevi and Victor Shoup. Algorithms in helib. Cryptology ePrint Archive, Report 2014/106, 2014. https://eprint.iacr.org/2014/106.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 1651–1669, 2018.

Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 619–631, 2017.

Zechun Liu, Wenhan Luo, Baoyuan Wu, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Binarizing deep network towards real-network performance. *International Journal of Computer Vision*, 128(1):202–219, 2020a.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision (ECCV)*, 2020b.

Qian Lou and Lei Jiang. She: A fast and accurate deep neural network for encrypted data. *Neural Information Processing Systems*, 2019.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. *arXiv preprint arXiv:2003.11535*, 2020.

Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38. IEEE, 2017.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. XONN: Xnor-based oblivious deep neural network inference. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1501–1518, Santa Clara, CA, August 2019. USENIX Association. ISBN 978-1-939133-04-5. URL https://www.usenix.org/conference/usenixsecurity19/presentation/riazi.

Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference*, pp. 1–6, 2018.

Amartya Sanyal, Matt Kusner, Adria Gascon, and Varun Kanade. TAPAS: Tricks to accelerate (encrypted) prediction as a service. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4490–4499, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/sanyal18a.html.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1284–1293, 2019.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

Yichi Zhang, Junhao Pan, Xinheng Liu, Hongzheng Chen, Deming Chen, and Zhiru Zhang. Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '21, pp. 171–182, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382182. doi: 10.1145/3431920.3439296. URL https://doi.org/10.1145/3431920.3439296.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Shilin Zhu, Xin Dong, and Hao Su. Binary ensemble neural network: More bits per network or more networks per bit? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4923–4932, 2019.

A CIRCUIT DETAILS

The details of a 4-bit comparator/ a 4-bit shifter / a 4-bit adder are shown in Fig. 6. The number of gates for each circuit can be computed as follows:

Comparator: Comparators are used in MaxPooling and PRelu/Psign operators. A 4-bit Comparator computes if $A > B$ is shown in Fig. 6(b). Suppose that an n-bit comparator requires c_n gates, we can derive that $c_n = c_{n-1} + n + 3$, with $c_1 = 1$.

Adder: The number of gates for a half adder and a full adder is 2 and 5 respectively. When the input bit-width is increased by 1, we need to add one more full adder (5 gates). Thus, we can derive the number of gates for an i bit adder is $5i - 3$.

Adder Tree: Considering the adder tree structure shown in Fig. 2(d), suppose the adder tree structure with 2^n half adders have a_n gates in total. We can derive that $a_n = 2a_{n-1} + 5(n+1) - 3 = 2a_{n-1} + 5n + 2$, with $a_1 = 2$. In this work, for x-bits adder tree input where $x \neq 2^k, k \in \mathbb{Z}$, we will round x to the 2^n where $2^{n-1} < x \leq 2^n$.

2-n MUX: A 2-n multiplexer (MUX) can do selection from two n-bit integers. It is used in Maxpooling and RRelu function. The circuit detail is shown in Fig. 6(d). For a 2-n MUX, the required number of gates to build the logic is $4 + 3n$.

p-q Shifter: A p-q shifter can shift a p bits integer q times. This module is used in PRelu/affine function and average pooling, As shown in Fig. 6(c), the required bit for an n-m shifter would be: $(4 + 3p)q$. In this work, we quantize the β to 2^n where $n \in [-3, 3]$, that means $n = 3$ our evaluation system.

The evaluation circuits for pooling and traditional sign function are shown in Fig. 3. For BNN (Courbariaux et al., 2015; 2016) with batch normalization ($f(x) = \beta x + b$) followed by a sign function (BN+sign), The Batch normalization can be removed: $x'_i = sign(x_i + T), T = \frac{b}{\beta}$.

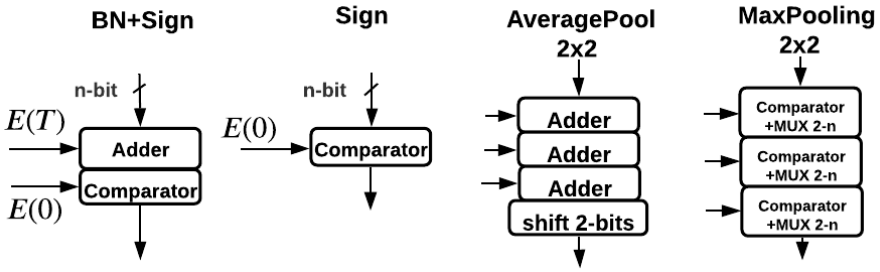

Figure 3: The evaluation circuits for (a) Batch normalization + sign function. (b) Sign function. (c) AveragePooling. (c) MaxPooling.

B MACHINE LEARNING AS A SERVICE (MLAAS).

An illustration of the MLaaS is shown in Fig. 4. Most previous works guarantee the client data privacy while assuming the model provider and cloud are the same party. Yet, in many scenarios, this is not true. For example, medical institutes / corporations have many training data to build a model. But they don't have enough cloud computing resources for MLaaS. In this work, we consider a more general usage of MLaaS that the model provider and cloud are different parties.

C GATENET BUILDING BLOCK AND ARCHITECTURE SETTING.

The channel and layer settings for MNIST follow the ShuffleNet-G1-1.0× in (Zhang et al., 2018) (also remove *stage 2* and *stage 3* while keeping only the *stage 1* layers). GateNet for CIFAR-10

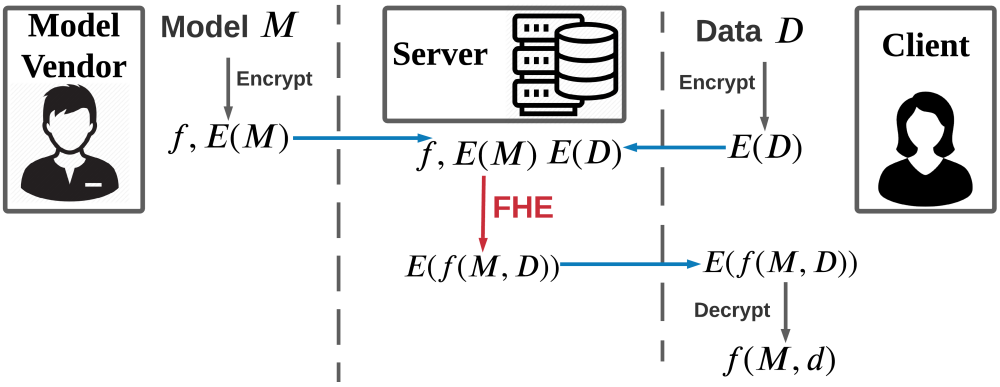

Figure 4: Overview of using GateNet for MLaaS.

follows the ShuffleNet-G3-1.0/1.5× settings (i.e., $g1, g2 = 3$ in Fig. 5). We remove the bottom maxpooling layer for CIFAR-10 and MNIST due to the small size the input images. GateNet for ImageNet follows the ShuffleNet-G3-1.0/2.0× settings in (Zhang et al., 2018). The building block of GateNet is shown in Figure 5.

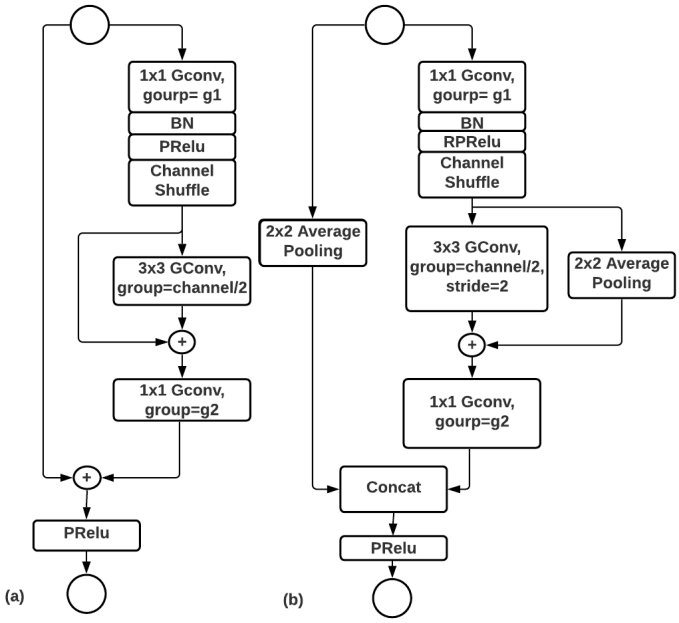

Figure 5: The building blocks of GateNet: (a) Normal block (b) Reduction block.

D WHY GATE OPS IS NOT SUITABLE FOR HARDWARE?

For FPGA design, the evaluation circuit will be synthesized into look-up tables (i.e., multiplexers) to fulfill the functionality of the circuit. Assuming an FPGA built on 6-3 MUXs, the 3-bit/3-bit XNOR has the same cost as a 6-bit popcount. However, in TFHE, the circuit represents by gates must be computed in a gate-level, and a 6-bit popcount is much more expensive than a 3-bit/3-bit XNOR operation.

For real hardware, such as CPU or GPU, the popcount and XNOR operations are compiled into instructions and executed on the hardware which is not relevant to the gate ops. The registers or flip-flops in real hardware allow a more efficient popcount design compared to the adder tree.

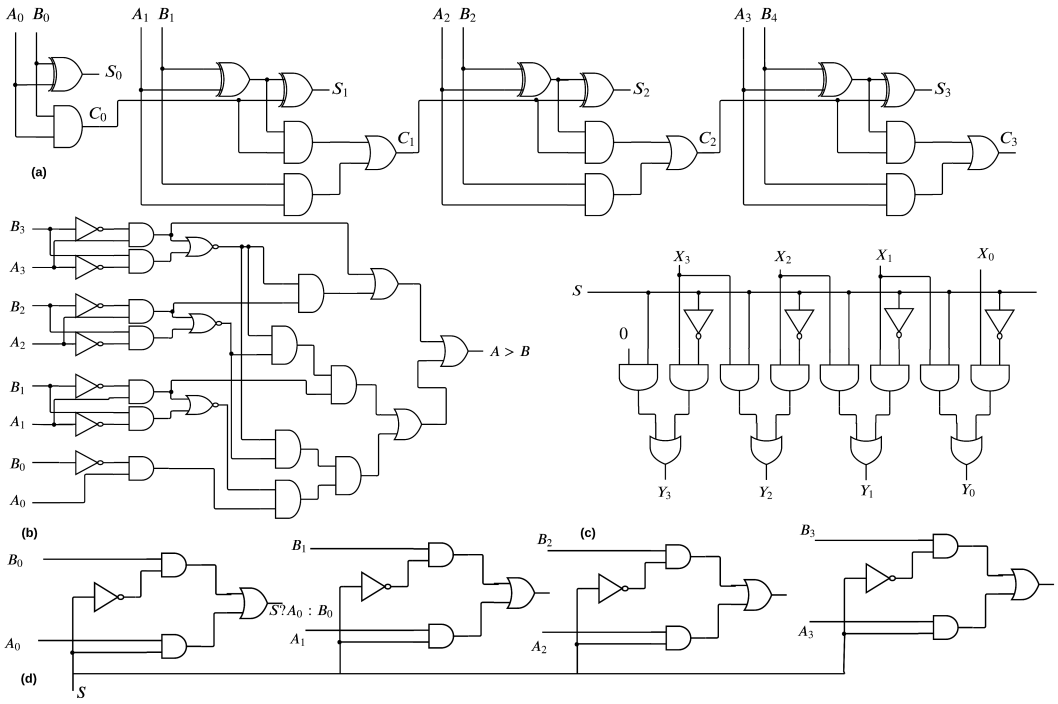

Figure 6: Circuit details of each module in Figure 2: (a) 4-bit adder (b) 4-bit comparator (c) 4-1 shifter (d) 2-4 MUX. Note that for larger bitwidth input, the user only need to cascade the repetitive logic.