Appendix for Paper 8410

Anonymous Author(s)
Affiliation
Address
email

1 Appendix

1.1 Open source

To replicate the training pipeline for the STGCN-3-256 model, please look into the attached folder entitled "NEURIPS-LinGCN-cleaned". For your convenience, we have provided the model and test code for the 4-STGCN-3-256 model in the aforementioned folder. Additionally, the test result logger file can be located within the directory at "NEURIPS-LinGCN-cleaned\work_dir\tmp\log.txt". These resources have been made accessible to promote clarity and ease of reproduction for those interested in building upon this work.

1.2 HE encoding with AMA format

Prior to encoding input data into polynomials, it is necessary to map the four-dimensional tensor $X \in R^{B\times C\times T\times J}$ to a one-dimensional vector in $R^{N/2}$ using the AMA format, as proposed in [1]. This transformation allows for more efficient execution of STGCN forward-computation in the HE domain. Below, we present the definition of the Vec function employed to map tensor X to a vector in $R^{N/2}$:

$$\begin{aligned} Vec(X) = y_j &= (y_{0,j}, \ldots, y_{i,j}, \ldots, y_{N/2,j}) \in R^{N/2} \\ \textit{s.t. } y_{i,j} &= X_{((i \bmod T)\%B)\times(i \bmod B\cdot T)\times(i\ \%\ T)\times j} \\ & j \in J \end{aligned} \tag{1}$$

Following the mapping process, the vectors y_j are encoded into polynomials with degree N and subsequently encrypted into ciphertext ct_j, as detailed in [2]. In this study, when N is set to 2^{16}, all tensors, including intermediate tensors, can be encrypted and packed into 25 ciphertexts, which corresponds to the number of nodes. For cases where $N = 2^{15}$ (2^{14}) the number of ciphertexts is 50(100). By selecting an appropriate value for N, the encryption and packing processes can be optimized to maintain performance and efficiency.

1.3 HE Setting Details

In Table 1, we furnish comprehensive details regarding the HE inference parameters. Specifically, i-STGCN-3 denotes a 3-layer STGCN model with i effective non-linear layers, while i-STGCN-6 signifies a 6-layer STGCN model with i effective non-linear layers. In this context, N represents the polynomial degree, and Q corresponds to the coefficient modulus.

To guarantee computation precision utilizing a one-time rescale operation, we assign the scale factor p for both ciphertext and plaintext to 2^{33}. This allocation results in a reduction of the current Q of ciphertext by p bits. This setup ensures that the overall performance and accuracy conform to the desired criteria while capitalizing on the security and resilience advantages conferred by HE.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Table 1: HE parameter settings in detail.

Model	Encryption Parameters				Mult Level
	N	Q	p	q_0	
6-STGCN-3	32768	509	33	47	14
5-STGCN-3	32768	476	33	47	13
4-STGCN-3	32768	443	33	47	12
3-STGCN-3	16384	410	33	47	11
2-STGCN-3	16384	377	33	47	10
1-STGCN-3	16384	344	33	47	9
12-STGCN-6	65536	932	33	41	27
11-STGCN-6	65536	899	33	41	26
7-STGCN-6	32768	767	33	41	22
5-STGCN-6	32768	701	33	41	20
4-STGCN-6	32768	668	33	41	19
3-STGCN-6	32768	635	33	41	18
2-STGCN-6	32768	602	33	41	17
1-STGCN-6	32768	569	33	41	16

Table 2: Comparison of latency breakdown between the non-reduced model with optimized model.

Model	HE Operators latency (s)				Total Latency (s)	Speedup ($\times$)
	Rot	PMult	Add	CMult		
6-STGCN-3-128	1336.25	378.25	99.65	37.45	1851.60	-
2-STGCN-3-128	392.21	266.13	68.90	14.31	741.55	2.50
6-STGCN-3-256	2641.09	1508.19	397.17	74.90	4621.36	-
2-STGCN-3-256	777.68	1062.21	274.96	28.63	2143.47	2.16
12-STGCN-6-256	18955.09	1545.09	396.23	275.39	21171.80	-
2-STGCN-6-256	4090.08	1006.79	244.19	115.05	5456.12	3.88

1.4 HE inference on GCNConv and Temporal-Conv Layer

Upon obtaining the AMA-packed ciphertexts ct_j, the adjacency matrix multiplication $A \cdot X$ can be decomposed into a series of plaintext multiplications, ***PMult***, in the HE domain. This decomposition accelerates HE-inference without necessitating rotations. Furthermore, the subsequent temporal convolution is performed solely on the temporal dimension T, utilizing 1×9 kernels.

$$ct'_k = A \cdot X = \sum_{i=1}^{m} ct_k A_i = \sum_{i=1}^{m} \sum_{k=1}^{J} PMult(ct_{i_k}, a_{i_k k}) \tag{2}$$

The AMA-packed ciphertexts allow for natural temporal convolution by single-node ciphertext ct_j, facilitating independent computation. This approach results in a ReLU-reduction design through structural pruning of ReLUs. The primary constraint to consider in this context is ensuring that the level consumption of each ciphertext remains equal prior to the GCNConv layer (node aggregation).

1.5 Further Detail of Operator Fusion

During the HE-inference process, employing weight fusion conserves the multiplicative depth, consequently reducing the ciphertext level budget. For instance, batch normalization, defined by an affine transformation $a'x + b'$, and a polynomial activation function, defined by $(ax + b)^2 + c$, can be readily fused into the corresponding temporal convolution layer $wx + b''$ with the function $w(a(a'x+b')+b)+b'' = (w \cdot a \cdot a')x + ab' + wb + b''$. As a result, three consecutive multiplications are consolidated into a single multiplication (pre-computing $w \cdot a \cdot a'$), thereby reducing the level consumption of ciphertext from 4 to 2 (1×9 convolution, batch normalization, and polynomial activation).

Analogous to the temporal-convolutional layer, the same fusion strategies can be applied to the polynomial activation and batch normalization of the GCNConv layer. Utilizing AMA-packed

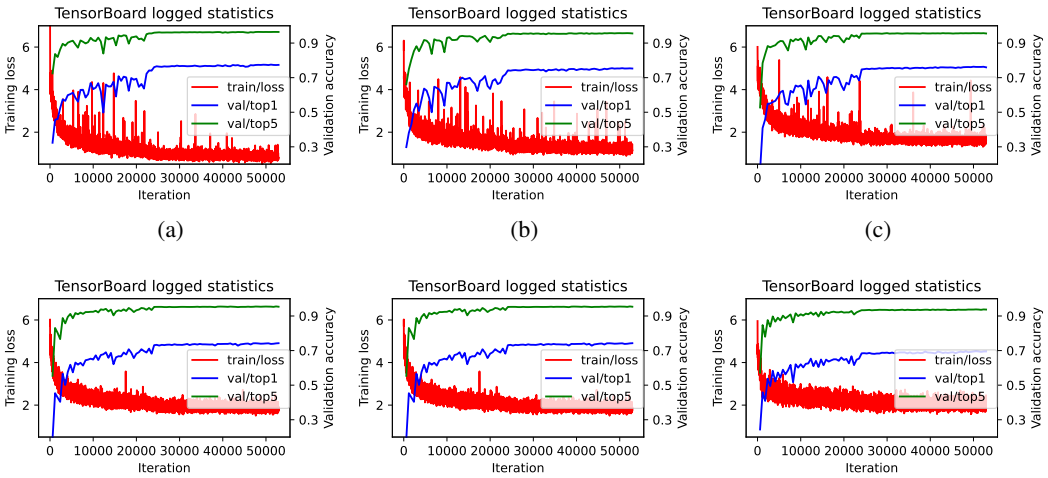

Figure 1: Polynomial replacement training curves for (a) 6-STGCN-3-128 (b) 5-STGCN-3-128 (c) 4-STGCN-3-128 (d) 3-STGCN-3-128 (e) 2-STGCN-3-128 (f) 1-STGCN-3-128

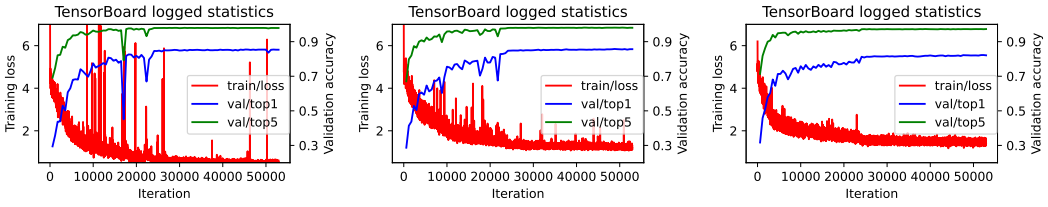

Figure 2: Polynomial replacement training curves for (a) 12-STGCN-6-256 (b) 4-STGCN-6-256 (c) 2-STGCN-6-256

ciphertexts, the node aggregation in GCNConv is translated as depicted in Equation 2, where each ciphertext carries out scalar multiplication with the plaintext of matrix elements $a_{i_k,k}$. Consequently, these plaintext matrix elements $a_{i_k,k}$ are fused with the primary 1×1 convolutional kernels to conserve the multiplicative level, reducing the total level consumption of the GCNConv layer from 4 to 2 (1×1 convolution, adjacency matrix multiplication, batch normalization, and polynomial activation).

1.6 Operator latency breakdown

Table 2 presents a comprehensive operator latency breakdown encompassing Rot, PMult, Add, and CMult operations. The designation i-STGCN-3-128 refers to an STGCN-3-128 model with i residual non-linear layers. As indicated in the table, the non-linear reduction contributes to a significant reduction in latency. By leveraging a smaller polynomial degree N, the overall latency experiences substantial improvement.

1.7 More Training Details and Insight

In this section, we present the training curves for the STGCN-3-256 model, which employs 6 to 1 effective second-order polynomial (non-linear) layers. Figures 1(a) through Figure 1(f) depict the training curve progression. During the training process, we utilized mixed-precision training for the polynomial model, which led to occasional instability in some iterations, as evidenced by spikes in the loss values. Nevertheless, the training process demonstrated rapid recovery following such loss spikes.

As demonstrated in training curve, a smaller number of second-order polynomial (non-linear) layers contribute to a more stable training process and facilitate smoother convergence. This finding explains the enhanced performance of the STGCN-6-256 model, which features a reduced number of non-linear layers, as compared to the full-polynomial model baseline.

To substantiate our hypothesis, we plot the polynomial replacement training curve for the STGCN-6-256 model in Figure 2. The training curves for 12 effective non-linear layers (12-STGCN-6-256), four effective non-linear layers (4-STGCN-6-256), and two effective non-linear layers (2-STGCN-6-256) are presented. As the number of non-linear layers increases, the model achieves greater expressivity; however, the polynomial replacement process becomes increasingly unstable. Consequently, for the STGCN-6-256 model with only 4 non-linear layers, a more stable replacement process facilitates better convergence, ultimately leading to improved accuracy performance.

References

[1] Ran Ran, Wei Wang, Quan Gang, Jieming Yin, Nuo Xu, and Wujie Wen. Cryptogcn: Fast and scalable homomorphically encrypted graph convolutional network inference. *Advances in Neural Information Processing Systems*, 35:37676–37689, 2022.

[2] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rns variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography*, pages 347–368. Springer, 2018.