Homomorphic Inference of Quantized CNNs via FHE16

1st Dohyuk Kim
Department of Electronic Engineering
Hanyang University
Seoul, Korea
dohyuk1000@hanyang.ac.kr

2nd Youngjun Kim
SW Development Group
Walllnut
Seoul, Korea
june0888@hanyang.ac.kr

3rd Seunghwan Lee
Department of Electronic Engineering
Hanyang University
Seoul, Korea
kr3951@hanyang.ac.kr

4th Dong-joon Shin
Department of Electronic Engineering
Hanyang University
Seoul, Korea
djshin@hanyang.ac.kr

***Abstract*—Fully Homomorphic Encryption (FHE) enables deep neural network inference on encrypted inputs, providing strong privacy guarantees for machine-learning-as-a-service. In this work, we study CNN inference on MNIST using FHE16, a TFHE-based bitwise FHE framework that lowers gate-bootstrapping cost by introducing 16-bit arithmetic and adopting efficient multi-input gate evaluation. We construct an end-to-end pipeline that trains a compact CNN in plaintext, quantizes weights and activations to integer form compatible with FHE16, and compiles the network into a Boolean circuit executed over encrypted bits. The resulting homomorphic inference relies on core FHE16 primitives for multi-operand accumulation and ciphertext–plaintext arithmetic, allowing exact evaluation of the discretized model without polynomial approximations.**

Experiments on an Intel Xeon Gold 6240R CPU show that the plaintext floating-point model achieves 81.82% accuracy, while the quantized integer model with scale factor 64 attains 80.47%. Encrypted inference over 1,000 MNIST test samples reaches 78.00% accuracy. The average end-to-end latency is $434.204\,\mathrm{s}$ per image, dominated by the convolution stage. These results demonstrate the feasibility of TFHE-based bitwise CNN inference under FHE16 and highlight convolutional bootstrapping as the primary performance bottleneck.

***Index Terms*—Fully Homomorphic Encryption, FHE16, Quantized DNN, Public Key Cryptosystem**

I. Introduction

Machine-learning-as-a-service (MLaaS) has become widespread, yet outsourcing inference to a remote provider raises privacy concerns because user inputs may be exposed to the server [1], [2]. Fully Homomorphic Encryption (FHE) addresses this problem by enabling inference directly over ciphertexts and returning encrypted predictions that only the client can decrypt [3], [4]. A major obstacle to practical FHE-based inference is efficiency, since homomorphic operations accumulate noise and require costly bootstrapping to support deep computations [4].

This research was supported by IRC

Existing homomorphic DNN inference has mainly adopted arithmetic FHE schemes such as BGV/BFV [5]–[7] and CKKS [8], [9]. While these schemes achieve high throughput using ciphertext packing, their bootstrapping procedures rely on large bootstrapping keys and incur substantial computational cost, which is a bottleneck for deeper networks. In contrast, TFHE-style bitwise FHE [10]–[12] evaluates Boolean circuits with relatively lightweight gate bootstrapping, shifting the performance focus from multiplicative depth to circuit size.

In this work, we evaluate a compact CNN on MNIST [13], [14] under FHE16, a TFHE-based low-cost bootstrappable system that reduces gate-bootstrapping overhead by introducing 16-bit arithmetic and efficient multi-input gate evaluation [15], [17]. We present an end-to-end pipeline from plaintext training to integer quantization and encrypted Boolean inference. In addition, we provide an explicit accounting of FHE16 primitive usage and a layerwise runtime breakdown, which together identify the dominant bottlenecks of TFHE-based CNN inference in practice.

II. Background

This section reviews the cryptographic primitives and design choices that underpin our encrypted CNN inference pipeline [13], [14]. We first summarize the role of bootstrapping in FHE, then contrast arithmetic FHE schemes with TFHE-based bitwise FHE schemes, and finally describe the specific optimizations introduced by FHE16 that motivate this work.

FHE enables evaluation of functions over encrypted data by providing homomorphic addition and multiplication. A fundamental limitation of all practical FHE constructions is ciphertext noise: each homomorphic operation increases the noise embedded in a ciphertext, and once this noise grows beyond a correctness bound, decryption fails. Bootstrapping resolves this problem by homomorphically refreshing ciphertexts through a decryption-related procedure, thereby reducing

noise and restoring the ability to continue homomorphic computation. As a result, bootstrapping is the key technique that enables homomorphic evaluation of circuits of unbounded depth, but its cost often dominates end-to-end performance.

Arithmetic FHE schemes such as BGV/BFV [5]–[7] and CKKS [8], [9] evaluate polynomial arithmetic over integer rings or approximate real/complex fields. The main advantage of them for machine learning is packing: many plaintext values can be embedded into a single ciphertext, allowing SIMD-style batch evaluation and hence achieving high throughput [18]. However, the same rich ciphertext structure makes bootstrapping expensive. In particular, refreshing packed ciphertexts requires large bootstrapping keys and a computationally expensive bootstrapping procedure, which becomes a bottleneck for deep networks or models requiring frequent refresh. Consequently, arithmetic FHE inference is applicable to shallow circuits or requires careful parameter tuning and activation approximations to avoid excessive bootstrapping.

TFHE-style bitwise FHE follows a different paradigm. Instead of packing many values into one ciphertext, TFHE-based FHE schemes encrypt individual bits in lightweight LWE ciphertexts and evaluate Boolean gates directly. Their defining feature is gate bootstrapping [11], [12], [16], which refreshes ciphertexts at the gate level with substantially smaller bootstrapping keys and lower per-refresh cost than arithmetic bootstrapping. This permits arbitrary Boolean circuit depth, while shifting the performance focus from multiplicative depth to circuit size. In other words, TFHE-style inference is most efficient when the target computation can be expressed with a small number of Boolean gates, even if the logical depth is large.

FHE16 builds on the TFHE bootstrapping framework and introduces two optimizations that are especially relevant for DNN inference [15], [17]. First, FHE16 incorporates 16-bit integer arithmetic in the gate-bootstrapping procedure, which significantly reduces the computational cost of each refresh operation compared with conventional TFHE implementations. Second, while standard TFHE circuits are naturally expressed as compositions of two-input gates, FHE16 supports efficient evaluation of multi-input gate operations. This capability reduces the number of parallel bootstrapping calls required to implement prefix and reduction circuits on encrypted bits. As a consequence, multi-operand arithmetic and comparison-based selection can be realized with fewer bootstraps than would be needed in a strictly binary-gate TFHE circuit. These properties make FHE16 a suitable backend for Boolean CNN inference, where convolutions and pooling require repeated multi-operand accumulations under encryption.

For the full cryptographic construction, parameter sets, and security analysis of FHE16, we refer the reader to the original reference paper [15].

III. Homomorphic Implementation of CNN

We train a compact CNN on MNIST in plaintext. The network consists of two 3×3 convolutional layers with average pooling [24], followed by two fully connected layers [23] that output ten logits. This architecture is deliberately made small in order to control Boolean circuit size under gate bootstrapping [11], [12], [16].

After the training is completed, we quantize weights and activations to 32-bit integers [19], [20]. Weights are stored in two's complement form, while activations are represented as unsigned values. ReLU activations [21], [22] are replaced by a thresholded sign operation applied after quantization. This replacement produces a discrete nonlinearity that is straightforward to implement as a Boolean comparison. The quantized network therefore becomes an integer circuit composed of dot products, additions, shifts, and comparisons.

To execute this quantized model with FHE16, we translate each integer operation into its bit-level Boolean circuit. Convolution and dense layers reduce to repeated 32-bit multiplications and multi-operand additions. Multiplication is implemented using ciphertext–plaintext `SMULLCONSTANT` operations, while additions rely on the `ADD3` reduction strategy to minimize the number of bootstrapped steps required to sum many operands. Pooling over 2×2 windows is computed as the sum of four encrypted values followed by a right shift by two bits, which corresponds to division by four and incurs only minor circuit overhead. The ReLU nonlinearity is realized homomorphically using the `MAX` primitive, and the final prediction is obtained argmax over decrypted logits.

In the encrypted inference protocol, the client encodes each MNIST pixel into a 32-bit integer and encrypts all bits with the FHE16 public parameters. The server evaluates the compiled Boolean CNN on ciphertexts and returns an encrypted prediction, which the client decrypts.

IV. Analysis of Homomorphic Operation Cost

Our CNN is intentionally made compact to make Boolean inference under FHE16 tractable [13]–[15]. The whole architecture outline is shown in Fig 1. Each MNIST input is a grayscale image of size 28×28, and we encrypt every pixel independently. Consequently, a single inference begins with $784 = 28 \times 28$ ciphertext encryptions. The encrypted image is then processed by a single convolution layer followed by a ReLU nonlinearity, a sum-pooling layer, and a fully connected output layer.

In the convolution layer, we apply 3×3 kernels with stride 3. Using three output channels (equivalently, three filters) allows the model to extract multiple local patterns while keeping circuit size manageable. With no padding, this configuration produces feature maps of spatial size 9×9 per filter, yielding an encrypted activation tensor of size $3\times9\times9$. Because FHE16 natively supports only integer arithmetic, smooth nonlinearities such as sigmoid are not directly evaluable. We therefore employ ReLU as the activation during training and realize it homomorphically via the $\max(0, x)$ primitive at inference time. It stabilizes optimization in plaintext (by alleviating vanishing gradients) and provides a discrete nonlinearity that aligns well with bitwise encrypted computation.

After activation, we apply 3×3 sum pooling [24] to each channel. Since pooling aggregates non-overlapping 3×3

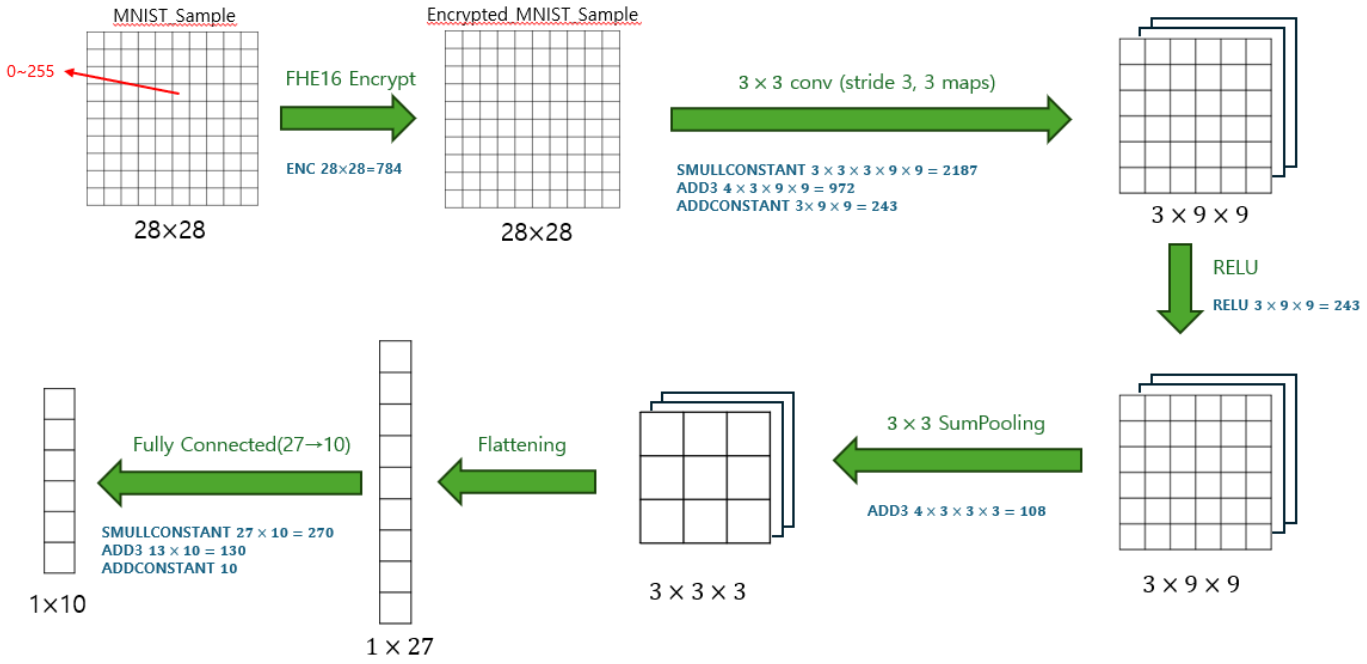

Fig. 1. The FHE16-based CNN Architecture.

windows, the spatial resolution is reduced from 9×9 to 3×3, resulting in an encrypted tensor of size $3 \times 3 \times 3$. The pooled tensor is flattened into a 27-dimensional vector, which feeds directly into the classifier. Notably, our encrypted model omits an additional hidden layer to avoid extra homomorphic multiplications and accumulations. Instead, the 27 pooled features are connected directly to ten output nodes, each representing one of ten classes $\{0, \ldots, 9\}$.

We now detail the homomorphic-cost profile of this architecture in terms of the core FHE16 primitives. Each convolution output requires computing an inner product over a 3×3 receptive field. Thus, nine ciphertext–plaintext multiplications are needed per each convolution output, implemented with `SMULLCONSTANT`. Because the convolution layer produces $3 \times 9 \times 9$ outputs, the total number of ciphertext–plaintext multiplications in the convolution layer is

$$3 \cdot 3 \cdot 3 \cdot 9 \cdot 9 \;=\; 2187.$$

The nine multiplication outputs of each convolution filtering must then be summed. Using the `ADD3` reduction strategy, summing nine operands requires four `ADD3` calls (three to reduce nine to three operands, and one more to reduce three to one). Therefore, the total number of `ADD3` executions in the convolution layer is

$$4 \cdot 3 \cdot 9 \cdot 9 \;=\; 972.$$

Also, bias addition is performed once per each convolution output via `ADDCONSTANT`, leading to

$$3 \cdot 9 \cdot 9 \;=\; 243$$

`ADDCONSTANT` calls. Finally, ReLU is evaluated elementwise using $\max(0, x)$, so we apply one homomorphic `Max` for each activation and the total number of `Max` operations is

$$3 \cdot 9 \cdot 9 \;=\; 243.$$

The subsequent 3×3 sum-pooling layer has the same arithmetic structure as the convolutional accumulation, except that it aggregates raw activations without multiplication. Each pooled output sums nine ciphertexts and therefore requires four `ADD3` calls. Since the pooling output has size $3 \times 3 \times 3$, the pooling layer performs

$$4 \cdot 3 \cdot 3 \cdot 3 \;=\; 108$$

`ADD3` executions in total.

In the fully connected layer [23], each of the ten class logits is computed as the weighted sum of the 27 pooled features. For each class, we perform 27 ciphertext–plaintext multiplications, requiring

$$27 \cdot 10 \;=\; 270$$

`SMULLCONSTANT` operations. The 27 products per class are accumulated using `ADD3` in a tree-like fashion: $27 \to 9 \to 3 \to 1$, which requires total $13 (= 9 + 3 + 1)$ `ADD3` executions per class. Hence, the fully connected layer uses

$$13 \cdot 10 \;=\; 130$$

`ADD3` operations. Each logit further adds a bias term via one `ADDCONSTANT`, yielding

$$10$$

`ADDCONSTANT` executions.

TABLE I
EXPERIMENTAL ENVIRONMENT AND ACCURACY UNDER DIFFERENT EVALUATION MODES.

Item	Value
CPU	Intel Xeon Gold 6240R (2.40 GHz)
Cores / Threads	24 / 48
Architecture	x86_64
Dataset	MNIST test set
# Test samples (FHE)	1,000
Plain (float) accuracy	81.82%
Plain (int, scale=64) accuracy	80.47%
FHE16 accuracy	78.00%

TABLE II
AVERAGE RUNTIME PER MNIST IMAGE UNDER FHE16 INFERENCE. PERCENTAGES ARE COMPUTED WITH RESPECT TO THE TOTAL LATENCY.

Stage	Time (s)	Share (%)
Encryption	117.895	27.15
Convolution	259.000	59.65
ReLU (MAX)	3.512	0.81
SumPool	13.023	3.00
Flatten	0.000	0.00
Fully Connected	39.944	9.20
Argmax + Decryption	0.001	< 0.01
Total	**434.204**	**100.00**

Overall, these counts show that the dominant homomorphic workload arises from encrypted convolution, where the combination of per-window multiplications and bootstrapped additions drives gate volume and latency. The pooling and fully connected stages contribute comparatively fewer bootstrapped operations, consistent with the compact design goals of our FHE16-based CNN.

V. EXPERIMENTS

A. Experimental Setup

As summarized in Table I, all experiments were executed on a single Intel Xeon Gold 6240R CPU running at 2.40 GHz (x86_64, 24 physical cores, 48 threads). We evaluate a compact CNN, which was trained on MNIST and subsequently deployed under FHE16 for encrypted inference. The network is intentionally made shallow and low-width to keep the Boolean circuit size tractable under bitwise gate bootstrapping, which prioritizes circuit volume over depth. The baseline plaintext model with floating-point weights achieves an accuracy of 81.82%. To match the integer-only computation supported by FHE16, we quantize the trained parameters using a scale factor of 64 and represent activations in the same low-bitwidth integer domain. The resulting integer-weight model retains 80.47% accuracy in plaintext and serves as a reference for all homomorphic evaluations reported in this section. Unless otherwise stated, all runtimes are measured on a per-image basis and averaged over 1,000 MNIST test samples.

B. End-to-End Accuracy

Homomorphic inference was conducted on 1,000 MNIST test samples (Table I), and the encrypted accuracy is reported as the average over these runs. It is shown that FHE16-based CNN inference achieves 78.00% accuracy. Since FHE16 evaluates discretized Boolean circuit exactly, any remaining gap with respect to quantized plaintext inference does not arise from homomorphic approximation error. Instead, the difference is attributable to circuit-level implementation constraints, such as fixed bit-width saturation/overflow behavior, the specific rounding and rescaling schedule induced by integer-only execution, and the exact MAX-based realization of ReLU. These effects, combined with the compact architecture (stride-3 downsampling, sum pooling, and a hidden-layer-free classifier), explain the modest accuracy drop observed in encrypted inference.

C. Runtime Breakdown

We measure end-to-end latency per encrypted image, including encryption on the client side and homomorphic evaluation on the server side. Table II summarizes the average time per stage. The total mean latency is 434.204 s per image, highlighting the substantial overhead of bitwise bootstrapped inference. Convolution dominates the runtime, accounting for nearly 60% of the total cost, while encryption itself contributes about 27%. Nonlinear activation and pooling are comparatively minor, and flattening is free in our implementation because it corresponds only to a permutation of ciphertext references. Argmax and decryption are negligible relative to bootstrapped circuit evaluation.

These measurements align with the primitive-count analysis in Section IV (where convolution incurs the largest volume of `SMULLCONSTANT` and multi-operand accumulation operations), confirming that convolution filtering is the primary driver of bootstrapping workload. In contrast, pooling and the final linear classifier require significantly fewer bootstrapped operations. Consequently, any end-to-end acceleration of TFHE-family CNN inference must focus on the convolution stage, either by reducing its gate/primitive count through architectural co-design and circuit optimization, or by accelerating bootstrapping via parallelism or hardware support.

VI. CONCLUSION

We investigated privacy-preserving CNN inference on MNIST using FHE16, a TFHE-based bitwise fully homomorphic encryption framework optimized through 16-bit arithmetic and efficient multi-input gate evaluation. Our work established a complete pipeline that starts with plaintext training, proceeds through integer quantization compatible with FHE16, and culminates in Boolean encrypted inference. This demonstrates that practical CNN inference can be realized under a purely bitwise, gate-bootstrapping setting without relying on polynomial activation approximations, thereby preserving the exactness of discrete nonlinear operations once the model is quantized.

The empirical study shows that the encrypted model retains most of the accuracy of its quantized plaintext counterpart, confirming that the dominant accuracy degradation stems from discretization rather than errors introduced during homomorphic evaluation. In terms of performance, our experiments show the key characteristic of TFHE-style inference: the

overall latency is governed primarily by the cost and frequency of bootstrapped bit-level operations, especially those arising from convolution filtering.

Taken together, these results highlight both the promise and the current limitations of bitwise FHE for DNN inference. Note that FHE16 materially improves the efficiency of multi-operand operations, making compact CNNs feasible on real hardware. However, scaling to larger datasets or deeper architectures will require further co-design between network structure and Boolean circuit cost. Future work should emphasize stronger discretized training (e.g., binary/ternary weights), circuit-minimizing accumulation strategies for convolutions, and parallel or hardware-accelerated bootstrapping. Advancing in these directions is essential for turning TFHE-based inference from a proof of feasibility into a practical MLaaS deployment.

REFERENCES

[1] R. Hamza, A. Hassan, M. B. Bashir, S. M. Alqhtani, T. M. Tawfeeg and A.Yousif, "Towards secure big data analysis via fully homomorphic encryption algorithms," Entropy 24.4 (2022): 519.

[2] R. Hamza and M. S. Dao, "Research on privacy-preserving techniques in the era of the 5G applications," Virtual Real. Intell. Hardw. 4.3 (2022): 210-222.

[3] N. M. Hijazi, M. Aloqaily, M. Guizani, B. Ouni and F. Karray, "Secure federated learning with fully homomorphic encryption for iot communications," IEEE Internet of Things Journal 11.3 (2023): 4289-4300.

[4] C. Gentry, "Fully homomorphic encryption using ideal lattices," Proceedings of the forty-first annual ACM symposium on Theory of computing. 2009.

[5] Z. Brakerski, C. Gentry and V. Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping," ACM Transactions on Computation Theory (TOCT) 6.3 (2014): 1-36.

[6] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," Annual cryptology conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[7] S. Carpov, M. Izabachène and V. Mollimard, "New techniques for multi-value input homomorphic evaluation and applications," Cryptographers' Track at the RSA Conference. Cham: Springer International Publishing, 2019.

[8] J. H. Cheon, K. Han, A. Kim and Y. Song, "Bootstrapping for approximate homomorphic encryption," Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer International Publishing, 2018.

[9] J. H. Cheon, A. Kim, M. Kim and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," In International Conference on the Theory and Application of Cryptology and Information Security, pages409-437." (2017).

[10] C. Bonte, I. Iliashenko, J. Park, H. V. Pereira and N. P. Smart, "FINAL: faster FHE instantiated with NTRU and LWE," International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer Nature Switzerland, 2022.

[11] I. Chillotti, N. Gama, M. Georgieva and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," international conference on the theory and application of cryptology and information security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.https://codeocean.com/capsule/4989235/tree

[12] I. Chillotti, N. Gama, M. Georgieva and M. Izabachène, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer International Publishing, 2017.

[13] S. S. Kadam, A. C. Adamuthe and A. B. Patil "CNN model for image classification on MNIST and fashion-MNIST dataset," Journal of scientific research 64.2 (2020): 374-384.

[14] R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional neural network (CNN) for image detection and recognition," 2018 first international conference on secure cyber computing and communication (ICSCCC). IEEE, 2018.

[15] S. Lee, D. Kim and D. J. Shin, "Fast, Compact and Hardware-Friendly Bootstrapping in less than 3ms Using Multiple Instruction Multiple Ciphertext," Cryptology ePrint Archive (2024).

[16] L. Ducas and D. Micciancio, "FHEW: bootstrapping homomorphic encryption in less than a second," Annual international conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[17] D. Kim, S. Kim, S. Lee and D. J. Shin, "Low-Latency Fully Homomorphic Arithmetic Using Parallel Prefix Group Circuit with Primitive Gate Bootstrapping," Cryptology ePrint Archive (2025).

[18] S. Halevi and V. Shoup. "Algorithms in helib," Annual Cryptology Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[19] S. Siddegowda, M. Fournarakis, M. Nagel, T. Blankevoort, C. Patel and A. Khobare, "Neural network quantization with ai model efficiency toolkit (aimet)," arXiv preprint arXiv:2201.08442 (2022).

[20] Y. Zhou, S. M. Moosavi-Dezfooli, N. M. Cheung and P. Frossard, "Adaptive quantization for deep neural network," Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.

[21] G. Lin, and W. Shen, "Research on convolutional neural network based on improved Relu piecewise activation function," Procedia computer science 131 (2018): 977-984.

[22] A. K. Dubey and V. Jain, "Comparative study of convolution neural network's relu and leaky-relu activation functions," Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018. Singapore: Springer Singapore, 2019. 873-880.

[23] S. S. Basha, S. R. Dubey, V. Pulabaigari and S. Mukherjee, "Impact of fully connected layers on performance of convolutional neural networks for image classification," Neurocomputing 378 (2020): 112-119.

[24] H. Gholamalinezhad and H. Khosravi, "Pooling methods in deep neural networks, a review," arXiv preprint arXiv:2009.07485 (2020).