Perceptual Encryption-based Privacy-Preserving Deep Learning in Internet of Things Applications

2022 13th International Conference on Information and Communication Technology Convergence (ICTC) | 978-1-6654-9939-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICTC55196.2022.9952589

Ijaz Ahmad[1] and Seokjoo Shin[*]
Department of Computer Engineering
Chosun University
Gwangju, South Korea
[1]ahmadijaz@chosun.kr, [*]sjshin@chosun.ac.kr (Corresponding author)

***Abstract*—In the Internet of Things (IoT) ecosystem, cloud computations are widely utilized to train a deep learning model on public datasets. The trained model can then be deployed on edge devices as an inference engine to facilitate various IoT applications in real-time. For consistent accurate predictions, the model needs to be retrained on the most recent data. This necessitates the data share between cloud servers and edge devices. However, the data acquired by IoT end devices usually consists of sensitive information and sharing them with cloud services provider results in users' privacy issues. In addition, the exchange of large volume of data requires high bandwidth. This study proposes an extension of block-based perceptual encryption (PE) algorithm to enable DL computations in encryption domain in order to protect users' privacy. For this purpose, four different extensions of baseline PE methods were analyzed in terms of compression savings, encryption efficiency and privacy-preserving DL performance. The analyses have shown that when global contents are preserved in each color channel, the extended PE method offers better security and preserves compression savings with only 3% drop in accuracy as compared to baseline method.**

Keywords—cloud computing, edge computing, perceptual encryption, privacy-preserving deep learning

I. INTRODUCTION

Given the recent success of Artificial Intelligence (AI) in various fields, most of the Internet of Things (IoT) applications are AI driven. The AI algorithms require data collected from the real world to provide a problem's solution. In the IoT ecosystem, the data comes from the deployed IoT nodes. Due to the resource-constrained nature of IoT devices, the traditional approach is to transmit the acquired data to a remotely located cloud server that runs an AI algorithm [1]. The data is processed there and the results are transmitted back to the origin of query. Cloud computation provides a solution to processing and data storage demands. However, the main limitation of centralized solution comes from the communication overhead, which makes them inefficient for real-time applications [1], [2]. In the recent years, edge computing is emerging as an alternative solution that offloads computations from the clouds to devices located at the edge of an IoT network. Edge devices are closer to the data sources than cloud servers are, thus significantly reduces the bandwidth consumption and network latency [1]–[3].

AI systems have two phases: 1) training phase, in which the algorithms learn how to solve a problem on available data and 2) inferencing phase, in which the trained algorithms are deployed to solve real world problems related to the domain they were previously trained on. Training AI models is compute-intensive task and its demand often exceeds the capability of resource-constrained edge devices [3]. On the other hand, inferencing requires less computational power; therefore, offloading inferencing computations to edge devices can provide timely solutions to IoT time sensitive applications [1], [3]. In traditional cloud-edge paradigm, deep learning (DL) models are trained on public datasets using cloud computations, whereas edge devices are only used for inferencing. For consistent accurate predictions, the model needs to be retrained on the most recent data available in IoT network. This necessitates the data share between cloud servers and edge devices. However, the data acquired by IoT end devices usually consists of sensitive information and sharing them with cloud services provider results in users' privacy concerns [1]. In addition, transmission between IoT edge devices and the cloud requires high bandwidth and makes the data vulnerable to being leaked. These dual requirements can be satisfied by compression and encryption processes [4]. The compression is a process of reducing the size of the data and can be carried out either in lossless or lossy mode. The lossless mode offers better image quality while lossy mode trades image quality for better compression savings. On the hand, encryption makes the data unintelligible and can only be recovered by using a secret key that the sender and receivers have agreed upon [4]. The traditional number theory and chaos theory based encryption algorithms are suitable for protecting the data during transmission and/or storage. However, running these algorithms on resource-constrained IoT devices is not efficient [5].

Overall, efficient edge-cloud collaboration to enable AI-based IoT applications have the following requirements: 1) the trained models should be periodically retrained on the most recent data. 2) When the data from edge devices is transferred to the cloud, bandwidth efficiency and security should be guaranteed. 3) The data should be encrypted in such a way that it can allow computations without the need of decryption.

In this study, we have first extended applications of perceptual encryption (PE) to meet the aforementioned requirements of IoT applications. Specifically, we have considered block-based PE (BPE) method proposed in [6] for color image encryption. The encryption algorithm consists of four steps: blocks permutation, blocks rotation and inversion, negative and positive transformation and color-channel shuffling. The steps are computationally inexpensive, which makes their implementation suitable on resource-constrained devices. Different form full image encryption, PE only hides the perceptual information of an image while preserves its intrinsic properties such as pixels correlation on a block level [7]. Based on this observation, BPE can be used for privacy-preserving DL (PPDL). However, a special consideration on the BPE security efficiency is required as the method was originally designed to provide bandwidth/storage efficiency and security during image transmission and/or storage. For better compression savings, larger block size is used which may make the algorithm vulnerable to ciphertext-only attack. Therefore, this study proposed four different extensions of

BPE method and analyzed their performance in-terms of compression savings, security and PPDL accuracy.

The rest of the paper is summarized as: Section II presents an overview of PE methods and proposed extension of PE methods, Section III presents analysis results and Section IV concludes the paper.

II. Perceptual Encryption Methods

A perceptual encryption (PE) makes an image unintelligible by disrupting its certain properties such as correlation among adjacent pixels and/or redundancy in an image. The PE can be carried out on a block level in order to preserve intrinsic properties of the image to enable certain computations in the encryption domain. A block-based PE algorithm consists geometric and color transformations performed on a block level as illustrated in Fig. 1. The geometric transformations change a block position and its orientation, whereas color transformation modifies pixels values in a block. For visual analysis, cipher images obtained from different PE methods are shown in Fig. 2. The conventional and proposed PE methods are described below.

A. Block-based PE Method (BPE)

An example BPE generated cipher image is shown in Fig. 1. (b). No visual information of the original image is visible in the cipher image. For an image I with $H \times W \times C$ pixels, divided into n blocks each with $B_h \times B_w$ elements, the block-based PE method proposed in [6] can be described as:

Step 1. Shuffle the blocks positions by using a randomly generated secret key K_1.

Step 2. Rotate and invert each block by using a random key K_2in order to change their orientations. Each entry of key K_2 represents a different combination of rotation and inversion direction.

Step 3. Randomly apply negative-positive transformation to each block by using a uniformly distributed key K_3. The transformation is a piecewise function that changes a pixel value p_j to $\check{p}_j$ in a block as:

$$\check{p}_j = \begin{cases} p_j, & K_3 = 0 \\ 255 - p_j, & K_3 = 1 \end{cases} \quad (1)$$

Step 4. Randomly shuffle the blocks in the three-color channels by using key K_4.

The same encryption keys K_i for steps $i = 1,2,3$ are used in each color channel. Processing each color channel independently can improve encryption efficiency as demonstrated in [8]. The original image can be recovered by performing the above steps in a reverse order. The block choice is dependent on the desired application. For example, to compress the cipher images obtained in Step (d) with the JPEG algorithm [9], a block size of 16×16 should be used.

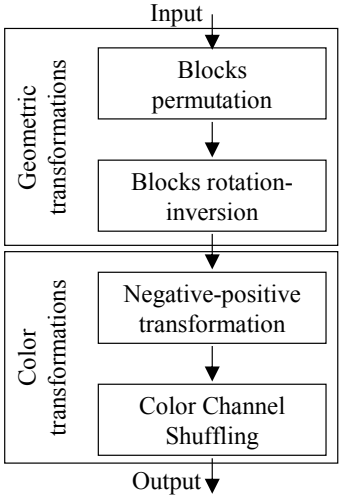

Fig. 1. Block-based perceptual encryption steps.

For BPE methods, there is a tradeoff relation between encryption and compression efficiency. For better security, large number of blocks are required that can be achieved by using smaller blocks size. However, the compression algorithm constraints the allowable smallest block size. In addition, the use of larger block size and leaving color intensity distribution unaltered make the BPE method vulnerable to ciphertext-only attacks [8]. Therefore, the current study proposes extended block-based PE method to mitigate these limitations.

B. Proposed Extended BPE Method (EBPE)

The main idea of proposed method is to benefit from smaller block size on a sub-block level as suggested in [7], [10]. Based on this principal, there are several possible extensions of conventional BPE methods as described below.

1) EBPE–1

An example EBPE–1 encrypted image is shown in Fig. 2. (c). Compare to BPE cipher image, the EBPE–1 has better scrambled image information. For an image I with $H \times W \times C$ pixels, divided into n blocks each with $B_h \times B_w$ elements, the cipher image is generated as

Step 1. Shuffle the blocks positions by using a randomly generated secret key K_1^1.

Step 2. Divide each block into sub-blocks of size $B_{h'} \times B_{w'}$ such that each block has l blocks.

Step 3. Shuffle the sub-blocks positions within a block by using a randomly generated key K_2^1.

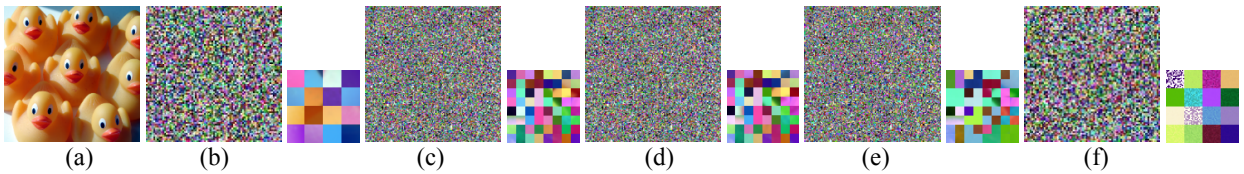

Fig. 2. Example image from the dataset. (a) is original image and (b) – (f) are its cipher images obtained from BPE and extended EBPE 1 – 4, respectively. The bottom right corner in each cipher image is magnified and shown on its right side.

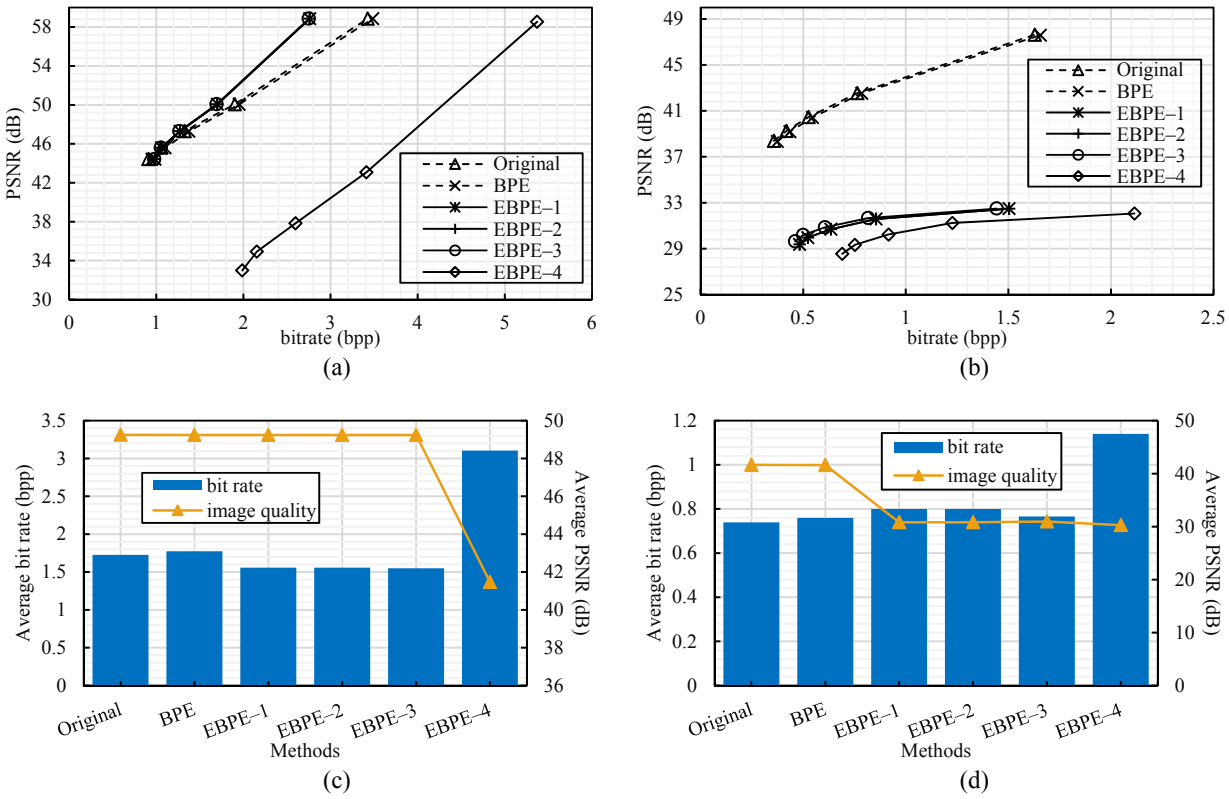

Fig. 3. Compression savings and recovered image quality analysis of plain and cipher images compression. The JPEG compression is performed without and with chroma sub-sampling in (a) and (b), respectively. (c) and (d) are the average compression savings and image quality for (a) and (b), respectively.

Step 4. Rotate and invert each sub-block by using a random key K_3^1 in order to change their orientations.

Step 5. Randomly apply negative-positive transformation to each sub-block using a uniformly distributed key K_4^1 as in (1).

Step 6. Randomly shuffle the sub-blocks in the three-color channels by using key K_5^1.

Note that performing Steps 4 – 6 on an entire block does not result in new transformations; however, only reverses the transformations of the sub-blocks. For example, changing orientation of an entire block will scramble the positions of sub-blocks in that block, which has already been performed in Step 2. Similarly, performing color transformations on an entire block will reverse Step 5 and 6 transformations on the sub-blocks.

2) EBPE–2

The EBPE–2 extension omits the sub-block scrambling as performed by EBPE–1 while performing the rest of the steps on a sub-block level. An example EBPE–2 cipher image is shown in Fig. 2. (d). For an image I with $H \times W \times C$ pixels, divided into n blocks each with $B_h \times B_w$ elements, perform the following steps:

Step 1. Shuffle the blocks positions by using a randomly generated secret key K_1^2.

Step 2. Divide each block into sub-blocks of size $B_{h'} \times B_{w'}$ such that each block has l blocks.

Step 3. Rotate and invert each sub-block by using a random key K_2^2 in order to change their orientations.

Step 4. Randomly apply negative-positive transformation to each sub-block using a uniformly distributed key K_3^2 as in (1).

Step 5. Randomly shuffle the sub-blocks in the three-color channels by using key K_4^2.

3) EBPE–3

This extension preserves the same information in each color channel; therefore, the color channel shuffling step is performed on a block level instead of sub-block. An example of EBPE–3 cipher image is shown in Fig. 2. (e). It can be seen in that color information in each channel is preserved on a block level. For an image I with $H \times W \times C$ pixels, divided into n blocks each with $B_h \times B_w$ elements, perform the following steps:

Step 1. Shuffle the blocks positions by using a randomly generated secret key K_1^3.

Step 2. Randomly shuffle the blocks in the three-color channels by using key K_2^3.

Step 3. Divide each block into sub-blocks of size $B_{h'} \times B_{w'}$ such that each block has l blocks.

Step 4. Shuffle the sub-blocks positions within a block by using a randomly generated key K_3^3.

Step 5. Rotate and invert each sub-block by using a random key K_4^3 in order to change their orientations.

Step 6. Randomly apply negative-positive transformation to each sub-block using a uniformly distributed key K_5^3 as in (1).

4) EBPE–4

In the previous extensions, the local contents of the image are preserved on a sub-block level. This extension disrupts contents of a block on the smallest possible unit that is pixels in the block, to achieve better security. An example of EBPE–4 cipher image is shown in Fig. 2. (f). For an image I with $H \times W \times C$ pixels, divided into n blocks each with $B_h \times B_w$ elements, perform the following steps:

Step 1. Shuffle the blocks positions by using a randomly generated secret key K_1^4.

Step 2. Scramble each pixel position in the block by using a randomly generated key K_2^4.

Step 3. Randomly apply negative-positive transformation to each block by using a uniformly distributed key K_3^4 as in (1).

Step 4. Randomly shuffle the blocks in the three-color channels by using key K_4^4.

Note that changing orientation of an entire block does not result in new transformations. For example, rotating a block will only change the positions of pixels in that block, which has already been performed in Step 2. Therefore, block rotation-inversion step is omitted from this extension.

III. Results and Discussion

This section first presents compression and encryption performance of conventional BPE [6] and proposed EBPE methods. The analysis were carried out on Tecnick sampling dataset [11], which consists of 120 color images of 1200×1200 dimensions. For the compression of cipher images, the JPEG standard algorithm [9] was used with and without chroma sub-sampling. In the quantization step, the standard luminance and chrominance quantization tables were used. In the second set of experiments, we have analyzed classification performance of a DL model on the plain and cipher images obtained from different PE methods.

A. Compression Analysis

For the compression efficiency analysis, Fig. 2. shows the rate distortion curves (RD) for bitrate savings (bpp) against image quality in terms of peak-signal-to-noise ratio (PSNR) (dB). The RD curves are for the JPEG quality factors of $Q_f = \{80, 85, 90, 95, 100\}$. The JPEG compression was carried out without and with chroma-subsampling in Fig. 3. (a) and (b), respectively. The block size is 16×16 for BPE and entire block processing of proposed EBPE, while 8×8 is used for sub-block processing of proposed EBPE. The plain image compression is named as 'Original'. The average compression savings and quality differences of the methods are shown in Fig. 3. (c) and (d), with and without chroma sub-sampling. When the images are compressed without sub-sampling, the difference in bit rate and image quality is almost negligible for BPE, EBPE–1, EBPE–2, and EBPE-3 compared to plain images compression. However, EBPE–4 drastically decreased the compression savings and image quality. When the compression is carried out with sub-sampling, the image quality of proposed EBPE methods is almost reduced by 11 dB. However, the compression savings remain almost same except for EBPE–4. Overall, the results shows that sub-block processing of the extensions EBPE–1, EBPE–2, and EBPE–3 has negligible effect on compression savings; however, when high quality images are required then the methods are not adequate.

For visual analysis the recovered images from different methods are shown in Fig. 4. When the images are compressed without chroma sub-sampling then no visible distortion can be seen in the recovered images across all methods. However, when the JPEG algorithm is implemented with chroma sub-sampling, then the sub-block processing results in block artifacts visible in the recovered images. For EBPE 1 – 3, the distortion appeared to be line on the borders of the blocks, whereas for EBPE–4 the whole block appearance is changed.

B. Encryption Analysis

This section presents robustness of PE methods against brute-force attack in terms of key space. The security efficiency of symmetric encryption algorithms has direct relation with the algorithms key size that is, larger the key size, the more secure the scheme is [5]. Each step in the PE algorithm has a random key and its size depends on the number of blocks the image has been divided. For an image I with $H \times W \times C$ pixels, divided into blocks of size $B_h \times B_w$ pixels, the number of blocks n are given as

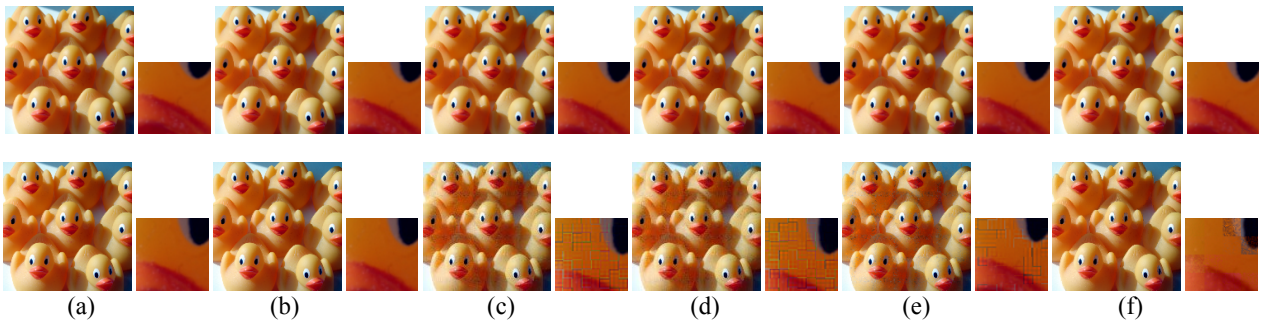

Fig. 4. Visual analysis of decoded images of Fig. 2. (a). The JPEG compression is performed in lossless mode (row 1) and lossy mode (row 2). (a) are recovered by decompressing the plain compressed image while (b) – (f) are recovered from the compressed cipher images for BPE and EBPE 1 – 4, respectively. The center duck left eye in each recovered image is magnified and shown on its right side.

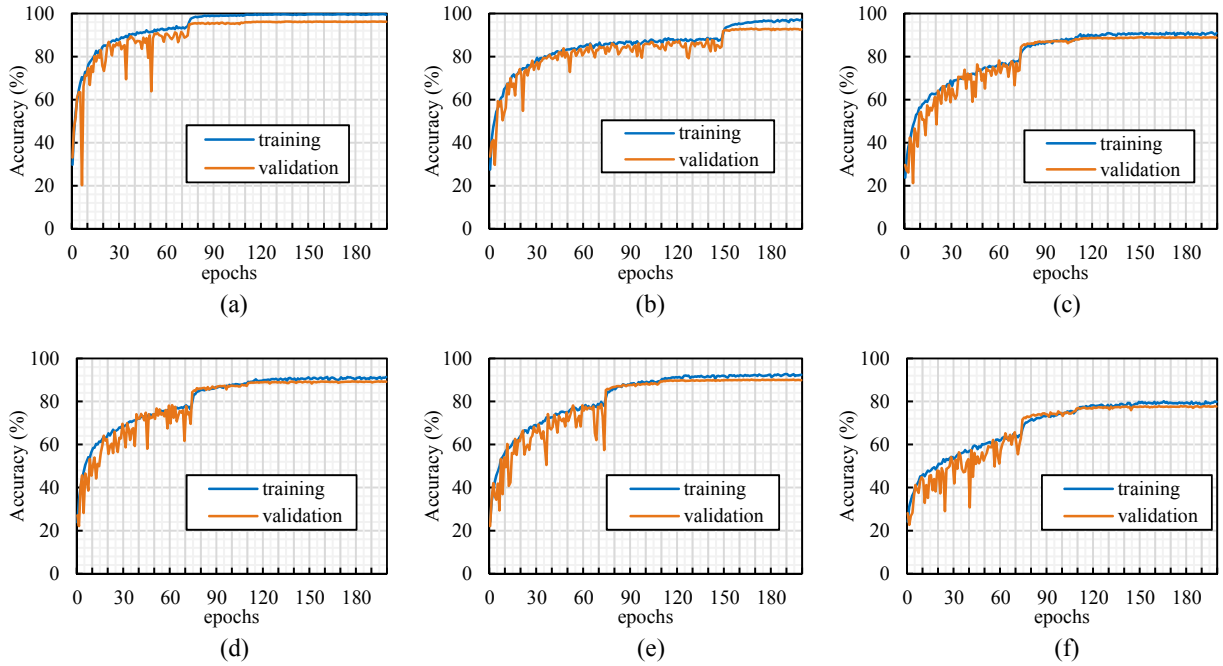

Fig. 5. Training and validation accuracy of the DL model for the CIFAR10 dataset. (a) is performance of the model on plain images while (b) – (f) is its privacy-preserving classification performance on the cipher images obtained from BPE and EBPE 1 – 4 methods, respectively.

$$n = \frac{H}{B_h} \times \frac{W}{B_w}, \tag{2}$$

and when the blocks are divided into smaller blocks of size $B_{h'} \times B_{w'}$ pixels, then the number of sub-blocks m in the image is

$$m = \frac{H}{B_{h'}} \times \frac{W}{B_{w'}}, \tag{3}$$

and the number of sub-blocks l in a block is given as

$$l = \frac{B_h}{B_{h'}} \times \frac{B_w}{B_{w'}}. \tag{4}$$

For PE methods, the keyspace can be derived as a product of key sizes selected in each encryption step. For example, the keyspace $K_{[6]}$ of conventional BPE is given as

$$\begin{aligned} K_{[6]} &= K_1 \cdot K_2 \cdot K_3 \cdot K_4 \\ &= n! \cdot 8^n \cdot 2^n \cdot 3!^n. \end{aligned} \tag{5}$$

The proposed EBPE methods incorporate sub-block processing for better security as opposed to BPE that only performs block level processing. The security efficiency is because of the fact that when the recovery of sub-block transformations (for example, orientation and color transformations) does not necessarily result in the correct appearance of their entire block [7]. For EBPE–1, the keyspace $K_{[EBPE-1]}$ is given as

$$\begin{aligned} K_{[EBPE-1]} &= K_1^1 \cdot K_2^1 \cdot K_3^1 \cdot K_4^1 \cdot K_5^1 \\ &= n! \cdot (n \cdot l!) \cdot (8^n \cdot 8^m) \cdot (2^n \cdot 2^m) \cdot (3!^n \cdot 3!^m). \end{aligned} \tag{6}$$

Note that sub-block permutation key $K_2^1 \neq m!$ as the sub-blocks positions in each block (that is, $l!$) can be recovered independent of other blocks. The product term $(n \cdot l!)$ shows that the process is repeated for each block in the image. Similarly, the product terms in keys K_3^1, K_4^1, and K_5^1 indicate that once the sub-block transformations are recovered, then the entire block transformations should be recovered as well. When the sub-blocks permutation is omitted then the keyspace $K_{[EBPE-2]}$ for EBPE–2 is given as

$$\begin{aligned} K_{[EBPE-2]} &= K_1^2 \cdot K_2^2 \cdot K_3^2 \cdot K_4^2 \\ &= n! \cdot (8^n \cdot 8^m) \cdot (2^n \cdot 2^m) \cdot (3!^n \cdot 3!^m). \end{aligned} \tag{7}$$

For EBPE–3, the keyspace $K_{[EBPE-3]}$ is given as

$$\begin{aligned} K_{[EBPE-3]} &= K_1^3 \cdot K_2^3 \cdot K_3^3 \cdot K_4^3 \cdot K_5^3 \\ &= n! \cdot 3!^n \cdot (n \cdot l!) \cdot (8^n \cdot 8^m) \cdot (2^n \cdot 2^m). \end{aligned} \tag{8}$$

For EBPE–4, the keyspace $K_{[EBPE-4]}$ is given as

$$\begin{aligned} K_{[EBPE-4]} &= K_1^4 \cdot K_2^4 \cdot K_3^4 \cdot K_4^4 \\ &= n! \cdot (n \cdot (h \times w)!) \cdot (2^n \cdot 2^m) \cdot (3!^n \cdot 3!^m). \end{aligned} \tag{9}$$

Based on (5) – (9) it can be seen that the proposed EBPE method has larger keyspace than BPE; therefore, can resist brute-force attacks. Among the proposed extensions, EBPE–4 has the largest keyspace.

C. *Privacy-Preserving Classification Analysis*

For privacy-preserving classification analysis, we have implemented the PyramidNet model proposed in [12]. The model was 110 layers in depth with a widening factor of α=270. The ShakeDrop regularization [13] was utilized for better performance. The model was trained for 200 epochs using Stochastic Gradient Descent (SGD) with the Nesterov

TABLE I. Privacy-preserving image classification analysis of DL model in terms of accuracy (%).

Methods	Training	Test	Epochs
Original	99.58	96.30	136
BPE	96.35	92.74	139
EBPE–1	91.03	89.04	151
EBPE–2	90.89	89.33	177
EBPE–3	92.06	90.16	167
EBPE–4	79.08	77.85	139

accelerated gradient and momentum method. The momentum of 0.9, weight decay of 0.0001 and batch size of 512 were used during training. The initial learning rate was set to 0.1, which was then decayed by a factor of 0.1 at 75, 110 and 150 epochs. The dataset used was Cifar10 dataset [14], which consists of 50K and 10K training and test images. The images were uniformly distributed among 10 classes. During training, random crop and flip were used as augmentation methods. Fig. 5 shows the model's accuracy on plain and cipher images obtained from various PE methods. Table 1 summarizes best accuracies and the number of epochs required by the model to converge to them. Then, accuracy is determined on the test dataset. The model has achieved better accuracy on BPE cipher images as they preserves image local contents on a larger block size. The accuracy drop is only 3.5% as compared to the model tested on plain images. Among the extended BPE methods, EBPE 1–3 have a negligible difference in their performance. However, they resulted in additional 2.5% drop in the test accuracy. The most secure EBPE–4 has drastically reduced accuracy of the model.

D. Discussion

The main limitation of BPE is the security inefficiency resulted from the block choice. The keyspace analysis presented in Section III. B showed that extended BPE methods that incorporated sub-block processing improved the security of PE methods. The compression friendly extensions EBPE 1–3 methods preserved compression savings and image quality when no chroma subsampling is applied. However, when chroma subsampling is used, then the recovered image quality degrades. The compression savings gained because of the sub-sampling can be traded for better image quality, which may improve the model's accuracy. Overall, the extension of BPE that preserves global contents in each color channel (for example, EBPE–3) meets the requirements of edge-cloud efficient collaboration to enable AI-based IoT applications.

IV. Conclusion and Future Work

This study proposed extension of block-based PE method to realize AI enabled IoT applications. The proposed method provided bandwidth efficiency and security during data transmission from edge devices to the cloud for high performance AI models. In addition, it enabled privacy-preserving DL computations in order to protect privacy-sensitive information. The analysis have shown that EBPE can enable efficient edge-cloud collaboration in IoT ecosystem.

The current study showed that the accuracy of convolution-based model degrades as encryption steps become more complex. Therefore, considering PE methods with Vision Transformer (ViT) model [15] for image classification task can be an interesting future work. The motivation is that ViT model utilizes the Transformer architecture with self-attention to sequences of image patches. Therefore, each block of the cipher image can be used as an image patch of the ViT models.

Acknowledgment

This research is supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07048338).

References

[1] K.-H. Le, K.-H. Le-Minh, and H.-T. Thai, "BrainyEdge: An AI-enabled framework for IoT edge computing," *ICT Express*, p. S2405959521001727, Dec. 2021, doi: 10.1016/j.icte.2021.12.007.

[2] P. Fraga-Lamas, S. I. Lopes, and T. M. Fernández-Caramés, "Green IoT and Edge AI as Key Technological Enablers for a Sustainable Digital Transition towards a Smart Circular Economy: An Industry 5.0 Use Case," *Sensors*, vol. 21, no. 17, p. 5745, Aug. 2021, doi: 10.3390/s21175745.

[3] M. Merenda, C. Porcaro, and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," *Sensors*, vol. 20, no. 9, p. 2533, Apr. 2020, doi: 10.3390/s20092533.

[4] I. Ahmad and S. Shin, "A novel hybrid image encryption–compression scheme by combining chaos theory and number theory," *Signal Processing: Image Communication*, vol. 98, p. 116418, Oct. 2021, doi: 10.1016/j.image.2021.116418.

[5] M. Suárez-Albela, P. Fraga-Lamas, and T. Fernández-Caramés, "A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices," *Sensors*, vol. 18, no. 11, p. 3868, Nov. 2018, doi: 10.3390/s18113868.

[6] K. Kurihara, S. Shiota, and H. Kiya, "An encryption-then-compression system for JPEG standard," in *2015 Picture Coding Symposium (PCS)*, Cairns, Australia, May 2015, pp. 119–123. doi: 10.1109/PCS.2015.7170059.

[7] I. Ahmad and S. Shin, "A Perceptual Encryption-Based Image Communication System for Deep Learning-Based Tuberculosis Diagnosis Using Healthcare Cloud Services," *Electronics*, vol. 11, no. 16, p. 2514, Aug. 2022, doi: 10.3390/electronics11162514.

[8] I. Ahmad, E. Kim, S.-S. Hwang, and S. Shin, "Privacy-Preserving Surveillance for Smart Cities," in *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, Barcelona, Spain, Jul. 2022, pp. 301–306. doi: 10.1109/ICUFN55119.2022.9829680.

[9] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consumer Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992, doi: 10.1109/30.125072.

[10] I. Ahmad and S. Shin, "Encryption-then-Compression System for Cloud-based Medical Image Services," in *2022 International Conference on Information Networking (ICOIN)*, Jeju-si, Korea, Republic of, Jan. 2022, pp. 30–33. doi: 10.1109/ICOIN53446.2022.9687214.

[11] N. Asuni and A. Giachetti, "TESTIMAGES: a Large-scale Archive for Testing Visual Devices and Basic Image Processing Algorithms," *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, p. 8 pages, 2014, doi: 10.2312/STAG.20141242.

[12] D. Han, J. Kim, and J. Kim, "Deep Pyramidal Residual Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, Jul. 2017, pp. 6307–6315. doi: 10.1109/CVPR.2017.668.

[13] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, "Shakedrop Regularization for Deep Residual Learning," *IEEE Access*, vol. 7, pp. 186126–186136, 2019, doi: 10.1109/ACCESS.2019.2960566.

[14] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," p. 60.

[15] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," 2020, doi: 10.48550/ARXIV.2010.11929.