Partial Gated Feedback Recurrent Neural Network for Data Compression Type Classification

HYEWON SONG, BEOM KWON, HOON YOO, AND SANGHOON LEE, (Senior Member, IEEE)

1Department of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea
2Department of Electronics, Sangmyung University, Seoul 03016, South Korea
Corresponding author: Sanghoon Lee (slee@yonsei.ac.kr)

This work was supported by the research fund of Signal Intelligence Research Center supervised by Defense Acquisition Program Administration and Agency for Defense Development of Korea.

ABSTRACT Owing to the widespread use of digital devices such as mobile phones and tablet PCs that are capable of easily viewing contents, the number of digital crimes committed using these digital devices has increased. One of the most common digital crimes is to hide the header information of the compressed data, which makes the user’s data unusable. It is difficult to restore original data without the header because header contains the compression type. In this paper, we propose a Partial Gated Feedback Recurrent Neural Network (PGF-RNN) for the identification of lossless compression algorithms. We modify the gated recurrent units to improve the correlation of layers by grouping the fully-connected layers to effectively determine the characteristics of the compressed data. We emphasize on the temporal features, which consider a wide range of data, and spatial features from fully-connected layers to extract the feature vectors of each compression type. To improve the performance of the proposed PGF-RNN, we apply post-processing that considers the frequency of bit sequences on some compression types with similar compressed data. The proposed method is evaluated on 31 well-known lossless compression algorithms of the Association for Computational Linguistics dataset. The average top 1 accuracy of the proposed method is 92.63%.

INDEX TERMS Compression type classification, deep learning, gated recurrent unit, lossless compression, partial gated feedback recurrent neural network, recurrent neural network.

I. INTRODUCTION

As the use of digital devices has increased, data exchange has become more convenient, and various types of information can now be efficiently stored and transmitted by data compression [1]–[3]. However, digital crimes are also increasing daily owing to an increase in the data exchange. One of the most common digital crimes is to hide the header information of the compressed data, rendering the data unusable until a financial demand is met. This is effective because the header contains information regarding the compression type, and thus the compressed data cannot be used when the header is modified or removed.

In digital crimes, users are often unable to restore their data because the data header is eliminated, and there is no clear solution for restoring the data header. When the header is modified or removed, restoring the original data is nearly impossible because it is extremely difficult to determine the compression type. If a method is proposed to reduce the number of candidate compression types for compressed data, the compressed data could potentially be restored even without the header of the compressed data. Therefore, we propose a compression type classification method that uses only the overall characteristics of compressed data.

In previous studies [4]–[8], the proposed methods used feature vectors of the compressed data by extracting various hand-crafted features by conducting tests, such as a frequency test and runs test. While these methods can be applied easily, there is a limitation in expressing the data because it is difficult to increase the dimensionality of the feature vector using only the hand-crafted features. To overcome this problem, some studies have suggested methods that use deep learning to represent the characteristics of the compressed data, rather than extracting hand-crafted features.

Lossless compression algorithms compress data using a prestored index for a specific data sequence or using 6 of
different length depending on the frequency of a sequence’s appearance. Because of these characteristics of lossless compression, it is important to understand the data entirely to determine the frequently used text sequences. Additionally, generating features that can represent each compression type is difficult because it depends upon various parameters, such as a prestored index and sign of a substituted sequence. Recently, a convolutional neural network (CNN)-based method of distinguishing dictionary-based compression algorithms has been proposed [9]. In general, it is difficult to apply to text data that cannot be controlled by the scale (length), unlike image or video, because the input size of the CNN is fixed. To overcome this limitation, a spatial pyramid pooling (SPP) layer is used for extracting a fixed-sized features vector of each text data. However, using a CNN that partially extracts features from entire bit sequences is not an efficient method. Since lossless compression algorithms check the entire data when indexing for compression, the characteristics of compression types cannot be analyzed properly if the feature extraction is performed partially as CNN. Although the method uses various filters of SPP layer to extract fixed-sized feature vectors, it cannot represent the characteristics of each compression type by sliding a fixed-sized window. The average accuracy of CNN-based method as low as 80% proves this.

To know the characteristics of each lossless compression type, it is necessary to consider the entirety of the text data, rather than only a specific part of the data. Therefore, we apply a recurrent neural network (RNN), which is generally used for voice or text data. Unlike CNN, RNN considers the temporal dimension, enabling it to obtain compression type characteristics considering a wider area of data. However, in the case of a conventional RNN, only temporal features of data can be reflected, rather than including spatial features, because its structure has one-to-one matching for the connection between hidden states in each timestep. In other words, it is possible to find the relationship between the values within a certain interval of data, but it is difficult to determine the overall relationship of the data.

In this paper, we extract features that account for the relationship between bit sequences instead of extracting local features. Therefore, we propose a Partial Gated Feedback Recurrent Neural Network (PGF-RNN) that reflects the characteristics of lossless compression algorithms, such as structural features, indexing methods, and the organization of bitstreams. By extracting the temporal features of bitstreams using RNN, we handle the features of lossless compression algorithms effectively. In addition, the PGF-RNN has a structure in which several hidden states of each layer are grouped and fully-connected to hidden states of the next layer. By enhancing the correlation between layers through the fully-connected hidden states of each group, we generate a feature vector optimized for the classification of lossless compression algorithms by extracting both the temporal and spatial features of compressed text data. Finally, the compressed data are analyzed in bit units after performing PGF-RNN to increase the classification accuracy of the proposed method. For some entropy-based lossless compression types, such as Shannon, Shannon Fano Elias, and Huffman, it is insufficient to analyze using only the form of bit sequences because the encoded data are generated similarly. Therefore, the overall performance of the proposed method is improved by reclassifying misclassified data through a frequency test of input bit sequences. The main contributions of this paper are summarized as follows:

1) An RNN-based architecture, which extracts the temporal features of compressed text data in consideration with the characteristics of lossless compression algorithms, enables the improved performance of compression type classification;
2) The architecture of PGF-RNN, which groups the hidden states of each timestep and fully-connects them to the group of next timestep, helps to generate temporal and spatial features of the compressed data to more effectively determine the characteristics of the compression algorithms;
3) To demonstrate the validity and performance of PGF-RNN, we present a comparison of compression type classification accuracy between the gated recurrent unit (GRU)-based method and PGF-RNN and an analysis of the feature vectors.

The rest of the paper is organized as follows. In Section II, we present an overview of lossless compression algorithms and two basic methods for PGF-RNN. In Section III, the proposed PGF-RNN and post-processing are described. In Section IV, the analysis of the proposed architecture and performance comparison between the proposed and existing models are given. Conclusions are presented in Section V.

II. RELATED WORK

A. LOSSLESS COMPRESSION ALGORITHMS

There are two types of compression methods, i.e., lossy compression and lossless compression [10]–[13]. Lossy compression has the advantage of a high compression rate, but it has the disadvantage of being unusable for voice or text data because they are sensitive to data loss. In comparison with lossy compression, lossless compression has a low compression rate but has the advantage of no data loss during compression. Usually, the compression methods used to generally transmit data are .7z or .zip, which are examples of lossless compression. There is no difficulty in restoring images or videos to a state that is comparable to the original because the data lost from images or videos are not critical even if lossy compression is used. However, in the case of text data, it is difficult to estimate some lost data through surrounding values. Thus, since compressed text data can be restored differently from the original data, lossless compression is predominantly used in everyday life.

There are two types of lossless compression methods, i.e., entropy- and dictionary-based compression. Entropy-based compression is a method of compressing continuous data values by defining them as a shortcode for
reducing the length of data. Entropy specifies the amount of essential information that can be considered redundant and compressed to determine the lowest bit value of data occurring during communicating or writing of the given information. Then, it is proved that the limit of compression is defined as entropy to prevent the loss of original information. In other words, irrespective of how ideal the compression method is, the data size cannot be smaller than its entropy. Entropy-based compression has compression types that are affiliated with Shannon [14], Huffman [15], [16], and Golomb [17]. One of the general compression types in entropy-based compression is Shannon encoding. Shannon encoding follows the mechanism below.

1) Find the probability of each symbol and use them to find the code length and sum of probabilities.
2) Convert the sum of probabilities to binary and assign each symbol following a code length.

A symbol $S_i$ is the $i^{th}$ sequence of the data, and the probability $p_i$ of each symbol can be obtained as the number of bits of the entire data divided by the number of appearances of each symbol. The code length for each symbol can be calculated as follows:

$$l_i = \lceil -\log_2 p_i \rceil,$$

(1)

where $l_i$ represents the code length for each symbol and $\lceil x \rceil$ denotes a ceiling function that rounds $x$ up to the next integer value. Therefore, the method uses shorter code lengths for symbols that are more frequent over all the data. Table 1 shows an example of Shannon encoding. If there are 6 symbols and their probabilities $p_i$ are 0.36, 0.18, 0.18, 0.12, 0.09, 0.07, the code length of each symbol can be obtained using (1). For obtaining the binary values of each symbol as shown in the fifth column of Table 1, $p_i$ is listed in ascending order and the cumulative sum of $p_i$ is calculated. From the obtained binary values, the binary code is decided by cutting off the decimal point as the code length of each symbol.

In contrast, dictionary-based compression is a method of indexing a specific character sequence rather than consecutive or overlapped characters and then converting the entire bitstream based on predefined indices. Typically, algorithms using the dictionary-based compression method are kinds of Lempel-Ziv algorithms. Among the various Lempel-Ziv series, we describe the process of encoding and decoding of using LZ77 as shown in Fig. 1. LZ77 is an internal dictionary-based compression algorithm proposed in 1977 by Lempel and Ziv [1]. LZ77 compresses data by indexing the overlapping parts in a predefined sliding window to the dictionary and then replacing repeated sequences with the relative positions of the existing text sequences. Sliding windows comprise a search buffer that is used as a dictionary and look-ahead buffer that performs encoding using the search buffer. The output of the encoded data consists of three parts, i.e., an offset indicating the relative position of the overlapping text sequences, length of the overlapping string, and codeword. The encoding process of LZ77 is as follows:

1) Set the size of the search buffer and look-ahead buffer.
2) Determine if there is any redundancy between look-ahead buffer and search buffer.
3) If redundancy exists between the look-ahead buffer and search buffer, the position of the duplicated sequence is represented by the offset and length values. Then, the character after the duplicated part is set as the codeword. In this case, the offset is a value indicating the first part of the duplicated string, and the length indicates the length of the duplicated string.
4) If there is no redundancy with the search buffer, the offset and length values are set as 0, and the codeword is the first character of the look-ahead buffer.
5) The offset, length, and codeword determined by steps 3) and 4) are stored as tuples. Then, the sliding window is moved by the length of the encoded data.
6) By repeating steps 2) to 5), the encoding is performed.

The encoding processes of the two previous compression algorithms are those of Shannon and LZ77, which are the most representative compression algorithms in entropy and dictionary-based compression, respectively. The entropy-based compression algorithm is a method of making codes according to the frequency of the bits, whereas a dictionary-based compression algorithm is a method of generating codes according to a dictionary which depends on bit alignments. The two lossless compression schemes have different compression schemes, and thus the characteristics

| Symbol | $p_i$ | $l_i$ | Sum from $p_i$ to I-1 | Sum to $p_i$ binary | Code |
|--------|------|------|----------------------|-------------------|------|
| $S_1$ | 0.36 | 2    | 0                    | 0.00000...        | 00   |
| $S_2$ | 0.18 | 3    | 0.36                 | 0.01011...        | 010  |
| $S_3$ | 0.18 | 3    | 0.54                 | 0.10001...        | 100  |
| $S_4$ | 0.12 | 3    | 0.72                 | 0.10111...        | 101  |
| $S_5$ | 0.09 | 4    | 0.84                 | 0.11010...        | 1101 |
| $S_6$ | 0.07 | 4    | 0.93                 | 0.11101...        | 1110 |

**TABLE 1.** Example of Shannon coding. The code length $l_i$ of each symbol can be obtained from the probability $p_i$ by (1).
of the compressed data are also different. Fig. 2 shows the visualization of the feature vectors of Shannon and LZ77 obtained through GRU for arbitrary compressed text data. It is an image of all values obtained from hidden states in each layer. The x- and y-axis represents the number of hidden states in each layer and the number of layers in the GRU, respectively. This shows the difference between the two different compression algorithms. As shown in Fig. 2, in the case of Shannon, most of the hidden states of each layer are activated as indicated by the red box. In contrast, in the case of LZ77, only a few hidden states of each layer are activated, as indicated by the yellow box, and are gradually deactivated. When checking feature vectors of other compression algorithms except for Shannon and LZ77 as shown in Fig. 2, there are a big difference between the feature vectors of entropy- and dictionary-based compression algorithm. The parameters of the entropy-based compression algorithm are determined by checking through the overall data sequences. In contrast, the dictionary-based compression algorithm only needs to know the index stored in the dictionary, which means that it only needs to activate in front of the layers. Therefore, deep learning-based lossless compression type classification is possible because the feature vector for each compression algorithm differs due to the characteristics of the compression algorithm. In total, we used 31 well-known lossless compression algorithms, comprising 16 dictionary- and 15 entropy-based compression algorithms, as shown in Table 2.

### B. RECURRENT NEURAL NETWORK

RNN is a kind of neural network that forms a circular structure by connecting hidden states using directional edges. It is mainly used for sequential data, such as speech recognition, speech enhancement, and text classification, because the structure of the network can accept data regardless of the length of the data sequence [18], [19]. For an explanation of RNN, let $h_t$ be the hidden state at the $t^{th}$ timestep. Then, $h_t$ can be represented as follows:

$$ h_t = \phi(W_h h_{t-1} + U x_t), $$

where $x_t$ is input at the $t^{th}$ timestep, $h_{t-1}$ is hidden state at the $(t-1)^{th}$ timestep, $\phi$ is usually a logistic sigmoid function or a hyperbolic tangent function, $W$ is weight matrix suggesting how much information is obtained from $h_{t-1}$, and $U$ is the weight matrix suggesting how much information is obtained from $x_t$. RNN learns the data characteristics by determining the given data by a timestep. Although RNN has the advantage of understanding the flow of the data, the vanishing gradient problem may occur, meaning the gradient value gradually decreases during back-propagation as the number of $h$ increases. GRU has been proposed to avoid this problem.

### C. GATED RECURRENT UNITS

GRU was proposed by Bengio et al. in 2014, and evolved from RNN [20], [21]. For reducing the power of gate computation, the GRU structure is lighter than that of long short-term memory, which is also used for solving the gradient vanishing problem [22]. The value of the hidden state at the $t^{th}$ timestep of the GRU is expressed as:

$$ h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t. $$

The update gate $z_t$ provides the information for updating the current state by reflecting the ratio of previous and current information. $z_t$ is calculated using $h_{t-1}$ and $x_t$:

$$ z_t = \sigma(W_z x_t + U_z h_{t-1}), $$

where $W_z$ and $U_z$ are the weights which represent the extent of reflection of $x_t$ and $h_{t-1}$ at update gate $z_t$, and $\sigma$ is a
the five layers, layers in a group between two timesteps, and where the superscript \(i\) to the timestep to the hidden states. Gate \(g\) between timesteps can be expressed as:

\[
\tilde{h}_t = \tanh(W_t x_t + r_t \odot U_t h_{t-1}),
\]

where \(\odot\) is the Hadamard product [23]. The main difference from (2) is that the GRU has a reset gate \(r\) to determine about how much the past information is reflected in the current state. \(r\) is calculated using \(h_{t-1}\) and \(x_t\):

\[
r_t = \sigma(W_r x_t + U_r h_{t-1}),
\]

where \(W_r\) and \(U_r\) are the weights that represent the extent of reflection of \(x_t\) and \(h_{t-1}\) at reset gate \(r_t\), respectively.

### III. PARTIAL GATED FEEDBACK RECURRENT NEURAL NETWORK

#### A. THE STRUCTURE OF PGF-RNN

PGF-RNN for lossless compression type classification is based on GRU. PGF-RNN improves the conventional GRU by grouping fully-connected layers between timesteps [24]. Fig. 3 shows the structural difference between the conventional GRU and the proposed PGF-RNN. As shown in Fig. 3(a), conventional GRU takes the value only from \(h_{t-1}\) and reflects the value in \(h_t\). However, PGF-RNN transmits the information of several hidden states from the \((t-1)\)th timestep to the \(t\)th timestep using the partial fully-connection of hidden states. Gate \(g^{i\to j}\) reflecting fully-connected layers between timesteps can be expressed as:

\[
g^{i\to j} = \sigma(W^{i\to j}_g h^{i-1}_t + U^{i\to j}_g \sum_{s=1}^{H} g^{s\to j} U^{s\to j} h^{j}_{t-1}),
\]

where the superscript \(i\to j\) denotes the interaction between the \(i\)th layer at the \((t-1)\)th timestep and \(j\)th layer at the \(t\)th timestep, superscript \(s\to j\) denotes the fully-connected layers in a group between two timesteps, and \(h^{j}_{t-1}\) contains five layers, \(h_{t-1}, \ldots, h_{t-4}\). In PGF-RNN, it is used as a successive fully-connected structure of five hidden states at the \((t-1)\)th timestep and five hidden states at the \(t\)th timestep for improving the layer to layer interaction and the time interaction as shown in Fig. 3(b). In a conventional GRU, the information from the \((t-1)\)th timestep is simply obtained through \(U h_{t-1}\) as in (5). However, in the PGF-RNN, \(\tilde{h}_t\), which reflects \(g^{i\to j}\), is calculated as:

\[
\tilde{h}_t = \tanh(W^{i\to j}_g h^{i-1}_t + r_t \odot U^i h_{t-1}),
\]

where \(H\) is the number of hidden layers and \(\tilde{h}_t\) is the new candidate memory of the content of the \(j\)th layer at the \(t\)th timestep. The amount of information obtained from the group of hidden states of the previous timestep can be obtained by applying \(g^{i\to j}\). \(g^{i\to j}\) represents the connection between timesteps to \(\tilde{U}\), which means the extent of reflection of \(x_t\) in (5). Therefore, a wide range of data can be understood for the extraction of the characteristics of input data by sharing the information about a plurality of layers using PGF-RNN.

The specific parameters used to implement PGF-RNN for lossless compression type classification are as follows. The number of hidden states in each timestep is 125, and the number of cells (i.e., the number of timesteps) is set to 50 to learn the information at each timestep sufficiently. Compressed text data are converted to decimal after separating data by 8 bits. The reason why the bitstream is divided into decimal codes is that it is difficult to know the characteristics of the compression algorithms because the variation between bits is constant when binary values are used. The compressed text data converted to decimal are normalized and are finally used as the input of the network, as shown in Fig. 4. The dimension of \(X_t\) is the value resulting from dividing the total length of the decimal value converted from the compressed text data.
Fig. 4. Generation of input in PGF-RNN.

Fig. 5. Cross-correlation matrix of CNN-based, GRU-based and PGF-RNN. Each axis represents the index of compression types.

by the number of cells, which is 50. It means that PGF-RNN accepts text data of arbitrary length. We also used the 1 × 125 vector, which is the concatenated output \( Y_t \), as the input of the fully-connected layer. The class label for each data is a 1 × 31 dimensional one-hot vector. The probability of classifying compression types for the data is obtained and we finally determine the compression type with the highest probability.

B. THE FEATURES USED FOR PGF-RNN

The proposed PGF-RNN improves the performance of compression type classification by extracting feature vectors differently for each compression type, unlike the conventional CNN- and GRU-based methods. The feature vector for the compression type can be analyzed for two aspects: temporal feature and spatial feature. The temporal feature is not considered to be obtained by simply referring to some surrounding values such as CNN, but instead considers the sequence of text data by obtaining the characteristics of a wider region. The spatial feature results from the structural difference between the GRU-based method and PGF-RNN. The GRU-based method has one-to-one connections between the hidden states of each layer, but in the proposed PGF-RNN, it has partially fully-connected. It can be said that PGF-RNN gives more consideration to the correlation between layers. By analyzing both aspects, we show the structural validity of PGF-RNN.

1) TEMPORAL FEATURES

The temporal feature is a feature vector that considers the overall values of the compressed data by extracting the characteristics referring to a wider region. Fig. 5 shows the cross-correlation matrix whose values are the average of cross-correlation between feature vectors of each compression algorithm learned by the CNN-based method, GRU-based method, and PGF-RNN. In the cross-correlation matrix, a lighter color indicates that two feature vectors are similar, whereas a darker color indicates that the correlation of two feature vectors is lower. The diagonal line in the matrix is always bright because it represents a correlation value between the same feature vectors. In other words, it is ideal that the cross-correlation matrix has only a bright diagonal line. The difference in performance according to the usage of temporal features is shown by identifying the correlation between feature vectors of each compression algorithm through the cross-correlation matrix. As shown in Fig. 5(a), there are many brightly colored areas in addition to the diagonal line in the case of determining compression algorithms using the CNN-based method. This means that the CNN-based method is not good for compression type classification because the feature vectors of each compression algorithm are not distinguishable. In Fig. 5(b), the feature vectors of some compression types have high correlations that show brighter colors than the CNN-based method. However, as shown in Fig. 5(c), the correlation of feature vectors of each compression type is significantly lower using PGF-RNN than that using the GRU-based method. This result shows that the features of lossless compression algorithms are effectively extracted by considering both temporal and spatial features by exchanging information between layers through partially gated feedback.

2) SPATIAL FEATURES

As shown in Fig. 3, the structure of the conventional GRU can show the association between bit sequences because hidden states in the same order are connected between layers. However, there is a limitation to the association between neighboring sequences because it uses a concatenation structure. In contrast, PGF-RNN has a structure in which several hidden states belonging to the same layer are assembled in one group and are fully-connected with groups of other layers. As shown in Fig. 6, the final outputs of RNN are obtained by passing through several concatenated cells. We analyze the cross-correlation of the output \( Y_i \) at the \( i^{th} \) cell \( (i \in [1, \cdots , N]) \) that can be obtained from each cell to take advantage of the structural advantages of PGF-RNN using fully-connected layers. Fig. 7 is a cross-correlation matrix between the conventional GRU and the output of each cell of the PGF-RNN. The x and y axes of each matrix represent the number of each compression type defined in Table 2. Additionally, the
cross-correlation matrix is calculated by the mean values of cross-correlation between the \( \{Y_1, Y_2, \ldots, Y_N\} \) obtained from each data of each class. As shown in Figs. 7(a) and 7(b), the two cross-correlation matrices make a distinct difference. Higher the correlation between \( Y_i \) is indicated with red, whereas lower correlation is indicated with blue. Additionally, the larger the red area around the diagonal line, the greater the number of cells affected by one cell. In the case of PGF-RNN, the red region is wider around the diagonal line than that in the GRU-based method. Fig. 7(c) shows the difference of correlation matrix values of Figs. 7(a) and 7(b). This clearly shows the performance difference between the GRU-based method and PGF-RNN. The more the diagonal line goes away, the higher the difference of correlation values is. In other words, it can be seen that the PGF-RNN extracts the characteristics of the compression type by considering the front and back relationship of the compressed data more widely than the conventional GRU by connecting being fully-connected between each timestep.

\section*{C. POST-PROCESSING}

Entropy-based compression types generate the code of each symbol \( S_i \) for encoding according to the frequency of symbols in the given data. When the functions \( F_i \) and \( I_i \) are similar, which are the functions to help \( p_i \) to make a binary form and determine the length of the code, the encoding results are similar. Fig. 8 is an example of encoding with Shannon and Shannon Fano Elias, which are the basic methods of entropy-based lossless compression types. As shown in Fig. 8, the codes of Shannon and Shanno Fano Elias are similar because they have similar functions \( F_i \) and \( I_i \). For this reason, the classification accuracies of some entropy-based compression types are very low. In particular, the top 1 classification accuracies of Shannon and Shannon Fano Elias are more than 40% lower than those of other compression types. Shannon is most often misclassified as Shannon Fano Elias and Shannon Fano Elias is often confused as Shannon. In addition, both compression types are misclassified as Huffman. Huffman creates a code tree by combining the two symbols with the smallest probability and assigns code to each symbol, as shown in Fig. 9. Huffman has a different encoding method from Shannon and Shannon Fano Elias. However, there is a high probability that the symbol code is generated similarly, as shown in Fig. 9(b), because the entropy-based compression type has a characteristic that the symbol with the smallest probability has the longest code.

Therefore, high-performance compression type classification cannot be performed using only deep learning-based PGF-RNN for some misclassified compression types. In this paper, post-processing is used for reducing the misclassification to reclassify some blind data after primary classification with PGF-RNN. The overall flowchart of compression type classification with post-processing is shown in Fig. 10. Firstly, the compressed text data \( T \) is estimated its class \( C \) using the PGF-RNN. If the class \( C \) is not in \( N = \{\text{Shannon, Shannon Fano Elias, Huffman}\} \), \( T \) is reclassified. If \( C \) is in \( N \), then the data text \( T \) is reclassified by the Frequency test. The frequency test makes the data be reclassified based on the ratio \( R \) of bit 0 in the entire bit sequences which are generated as shown in Fig. 4. Reclassified class \( C'(T) \) is defined as follows:

\[
C'(T) = \begin{cases} 
N_1, & R \leq 7 \\
N_2, & 7 < R \leq 11 \\
N_3, & R > 11 
\end{cases}
\]
FIGURE 10. Flowchart of compression type classification.

We use the average values obtained from other text databases as the standard for dividing the three compression types. The top 1 classification accuracies of misclassified compression types $N$ have increased by more than 8% on average using post-processing. This result is shown in Table 5.

IV. EXPERIMENT
A. ENVIRONMENT
We used the Association for Computational Linguistics dataset with more than 100,000 text files to evaluate the performance of our proposed compression type classification method. Using the open-source code of each lossless compression algorithm, all the text files were compressed individually with 31 lossless compression algorithms. The total number of training data is 1,240,000 text files (40,000 text files per compression type), and the number of testing data is 155,000 text files (5,000 text files per compression type). The batch size is 620 (20 files for each compression type) and the learning rate is 0.001.

B. THE PERFORMANCE OF PGF-RNN
The experiments were conducted in two directions to evaluate the proposed method. One is to show the superiority of feature vectors obtained through PGF-RNN. We analyze the results of the support vector machine (SVM) and k-nearest neighbor (KNN), which are trained using feature vectors from the CNN-based method [9] and PGF-RNN. The other experiment is to measure the top 1 and top 3 accuracies of the compression type classification of PGF-RNN. It shows the performance of the proposed method as a compression type classifier by comparing it with the GRU-based method.

1) THE SUPERIORITY OF PGF-RNN’S FEATURES
We proved the superiority of the feature vectors obtained from our proposed method using SVM and KNN algorithms. The feature vectors for training SVM and KNN are obtained from the last layer of each method. By applying the two basic classification algorithms to each feature vector and comparing their accuracy, we checked whether the feature vectors obtained through each machine learning method are better than the feature vectors obtained through the CNN-based method [9] and PGF-RNN. The other experiment is to measure the top 1 and top 3 accuracies of the compression type classification of PGF-RNN. It shows the performance of the proposed method as a compression type classifier by comparing it with the GRU-based method.
FIGURE 11. Confusion matrix heatmaps of the CNN-based method and our proposed method. The number in the heatmap represents each compression type as mentioned in Table 3.

TABLE 4. Classification accuracy of GRU-based and proposed method.

| No. | Type          | GRU-based  | Proposed  |
|-----|---------------|------------|-----------|
|     |               | Top 1    | Top 3    | Top 1  | Top 3  |
| 1   | Adaptive Huffman | 96.67%  | 99.19%  | 96.60%  | 99.70% |
| 2   | Arithmetic Coding | 97.83%  | 99.49%  | 97.02%  | 99.36% |
| 3   | BPE           | 99.49%  | 99.90%  | 99.58%  | 99.92% |
| 4   | DEFLATE       | 89.14%  | 96.21%  | 94.14%  | 98.98% |
| 5   | Exp-Golomb    | 99.44%  | 99.95%  | 99.60%  | 100.00% |
| 6   | Fibonacci     | 99.95%  | 100.00% | 100.00% | 100.00% |
| 7   | Gamma         | 96.41%  | 99.95%  | 98.14%  | 99.98% |
| 8   | Golomb        | 97.32%  | 99.19%  | 98.76%  | 99.52% |
| 9   | Huffman       | 70.40%  | 86.01%  | 93.26%  | 98.72% |
| 10  | Levenshtein   | 100.00% | 100.00% | 99.94%  | 100.00% |
| 11  | LZ4           | 83.13%  | 98.89%  | 98.10%  | 99.88% |
| 12  | LZ77          | 81.67%  | 97.78%  | 85.84%  | 98.56% |
| 13  | LZ78          | 90.81%  | 98.89%  | 97.36%  | 99.92% |
| 14  | LZJB          | 43.43%  | 91.92%  | 75.72%  | 99.72% |
| 15  | LZMA          | 98.99%  | 99.90%  | 96.96%  | 99.94% |
| 16  | LZO           | 79.95%  | 98.74%  | 92.14%  | 99.56% |
| 17  | LZRW          | 98.59%  | 99.85%  | 99.06%  | 99.96% |
| 18  | LZS           | 57.68%  | 97.22%  | 80.00%  | 97.58% |
| 19  | LZSS          | 74.39%  | 97.68%  | 84.32%  | 97.14% |
| 20  | LZW           | 64.19%  | 96.72%  | 84.04%  | 98.78% |
| 21  | LZX           | 97.78%  | 99.60%  | 98.58%  | 99.70% |
| 22  | Modified Huffman (Black) | 100.00% | 100.00% | 100.00% | 100.00% |
| 23  | Modified Huffman (White) | 99.95%  | 100.00% | 99.98%  | 100.00% |
| 24  | RLE           | 94.24%  | 99.39%  | 95.09%  | 99.36% |
| 25  | Shannon       | 52.22%  | 96.82%  | 63.36%  | 98.90% |
| 26  | Shannon Fano  | 88.74%  | 96.41%  | 91.82%  | 95.58% |
| 27  | Shannon Fano Elias | 37.27%  | 99.09%  | 58.02%  | 98.96% |
| 28  | SNAPPY        | 93.59%  | 98.84%  | 97.30%  | 99.54% |
| 29  | Tunstall      | 98.74%  | 99.90%  | 98.56%  | 99.76% |
| 30  | Unary         | 100.00% | 100.00% | 100.00% | 100.00% |
| 31  | ZStandard     | 95.05%  | 99.55%  | 97.68%  | 99.78% |
|     | Average       | 86.36%  | 98.29%  | 92.63%  | 99.32% |

2) RESULTS OF COMPRESSION TYPE CLASSIFICATION

We proved the performance of the proposed method by comparing the classification accuracy of the proposed method with that of the GRU-based method, which is a base of our method. Furthermore, we proved the post-processing efficiency by showing the average top 1 accuracies of some entropy-based lossless compression types with and without post-processing.

Table 4 shows the average accuracy of classification for each compression type using the GRU-based method and the proposed method. The average top 1 accuracy of the proposed method is 92.63%, whereas that of the GRU-based method is 86.36%, which is more than 6% lower than that of the proposed method. The classification accuracies of

suitable for lossless compression type classification, i.e., how well each feature vector reflects the characteristics of the compression type [25]. Table 3 shows the top 1 accuracy of classification by learning SVM and KNN using the feature vector of the last layer of two methods. Since the classification accuracy of each compression type is similar between the CNN-based and proposed methods, the characteristics of the compression type reflected by the feature vector of the two methods are similar. However, the accuracies of some compression types such as Arithmetic coding (2), LZ77 (12), LZSS (19), SNAPPY (28), and Tunstall (29) of SVM and KNN based on the proposed method are more than 10% higher than those of the CNN-based method. It can be interpreted that the CNN-based method does not reflect the characteristics of various compression types, unlike the proposed method. Additionally, it can be seen that the average classification accuracy of the two methods differs by more than 10%. To determine the characteristics of the compression type, it is important to understand the relationship between bit sequences. The proposed method has a structure that increases the correlation between each timestep. It means that this structure is better for understanding the characteristics of the compression type by learning about the data.

Fig. 11 shows the classification results of the CNN-based and our proposed method as confusion matrix heatmaps. A confusion matrix, also known as an error matrix, shows performance for statistical classification. Each row of the matrix represents the predicted class and each column of the matrix represents the actual class. If the matching ratio between the predicted class and the actual class is high, the diagonal line of the confusion matrix looks clearer. In contrast, if the probability that the predicted class is not similar to the actual class, the non-diagonal parts become more visible. As shown in Fig. 11(a), although most of the compression types look to be properly classified, compression types, which have the numbers 1, 2, or 25, are not properly classified by comparing with the confusion matrix heatmap of the proposed method of Fig. 11(b). Additionally, there are more misclassified data using the CNN-based method than using the proposed method including the type numbers 1, 2, and 25. The average accuracy of both methods can also show this analysis of confusion matrix heatmaps.
some compression types such as numbers 9, 14, 16, and 19 (Huffman, LZJB, LZO, and LZSS) have significantly increased. This demonstrates that PGF-RNN represents the characteristics of various compression types by simultaneously considering the temporal and spatial features. The average top 3 accuracy of the proposed method is also higher than the result of the GRU-based method. Even though the top 1 accuracies of some compression types are significantly lower than other compression types, the top 3 accuracies of most lossless compression types are 95% or higher. This shows that PGF-RNN can be effectively used to determine the compression type for blind compressed data.

For some entropy-based compression types that have low average top 1 accuracy, we apply the post-processing after classifying the data through PGF-RNN, as shown in Fig. 10. Among several compression algorithms with low average top 1 accuracy, we performed post-processing on Huffman (9), Shannon (25), and Shannon Fano Elias (27). Table 5 shows the average top 1 accuracy with and without post-processing for three compression types. As shown in Table 5, post-processing has a good performance on all three compression types. The accuracies of three compression types have increased by more than 5%. By improving the top 1 accuracies of the three compression types through post-processing, the average top 1 accuracy of the 31 compression types tested also increased from 91.78% to 92.63%. In the case of entropy-based compression, it is important to determine the overall bit pattern from compressed data. However, there is a limitation in identifying the characteristics of the algorithm as a pattern of data, especially for a similar type of compression algorithm. Therefore, through the additional post-processing using bit frequency information, the overall performance of the proposed method is improved by making up for the parts represented by a deep learning-based method.

### V. CONCLUSION AND FUTURE WORK

We propose PGF-RNN for lossless compression type classification. We performed classification by modifying the conventional GRU to use a model that considers the timestep due to the characteristics of the lossless compression algorithm that compresses data by scanning the whole bit sequences. PGF-RNN has a structure that utilizes information between layers to enhance the interaction of hidden states by sharing the weights of layers. This structure overcomes the limitations of GRU, which only temporal characteristics of data, by also using the spatial features of compressed data.

For enhancing the accuracies of some misclassified compression types, we apply post-processing which considers the frequency of bit sequences. Our proposed compression type classification method has good performance on lossless compression type classification. We expect that the proposed method will be useful for finding digital forensics evidence.

### REFERENCES

[1] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977.

[2] A. K. Jain, “Image data compression: A review,” *Proc. IEEE*, vol. 69, no. 3, pp. 349–389, Mar. 1981.

[3] M. J. Weinberger, A. Lempke, and J. Ziv, “A sequential algorithm for the universal coding of finite memory sources,” *IEEE Trans. Inf. Theory*, vol. 38, no. 3, pp. 1002–1014, May 1992.

[4] B. Kwon, M. Gong, and S. Lee, “Novel error detection algorithm for LZSS compressed data,” *IEEE Access*, vol. 5, pp. 8940–8947, 2017.

[5] B. Kwon and S. Lee, “Error detection algorithm for Lempel-Ziv-77 compressed data,” *J. Commun. Netw.*, vol. 21, no. 2, pp. 100–112, Apr. 2019.

[6] B. Kwon, M. Gong, J. Huh, and S. Lee, “Identification and restoration of LZ77 compressed data using a machine learning approach,” in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Ann. Summit Conf. (APSIPA ASC)*, Honolulu, HI, USA, Nov. 2018, pp. 178–181.

[7] B. Kwon, M. Gong, and S. Lee, “EDA-78: A novel error detection algorithm for Lempel-Ziv-78 compressed data,” *Wireless Pers. Commun.*, vol. 111, pp. 1–13, Nov. 2019.

[8] B. Kwon, H. Song, and S. Lee, “Accurate blind Lempel-Ziv-77 parameter estimation via 1-D to 2-D data conversion over convolutional neural network,” *IEEE Access*, vol. 8, pp. 43965–43979, 2020.

[9] H. Song, B. Kwon, S. Lee, and S. Lee, “Dictionary based compression type classification using a CNN architecture,” in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Ann. Summit Conf. (APSIPA ASC)*, Nov. 2019, pp. 245–248.

[10] M. Nelson and J. Gailly, *The Data Compression Book*. New York, NY, USA: M&T, 1996.

[11] K. Sayood, *Lossless Compression Handbook*. Berkeley, CA, USA: Elsevier, 2002.

[12] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Commun. ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987.

[13] S. Kaur and M. Sukhjeet, “Entropy coding and different coding techniques,” *J. Netw. Commun. Emerg. Technol.*, vol. 6, no. 5, pp. 4–7, 2016.

[14] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.

[15] M. Sharma, “Compression using Huffman coding,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 10, no. 5, pp. 133–141, May 2010.

[16] J. S. Vitter, “Design and analysis of dynamic Huffman codes,” *J. ACM*, vol. 34, no. 4, pp. 825–845, Oct. 1987.

[17] R. Gallager and D. van Voorhis, “Optimal source codes for geometrically distributed integer alphabets (corresp.),” *IEEE Trans. Inf. Theory*, vol. 21, no. 2, pp. 228–230, Mar. 1975.

[18] I. Sutskever, O. Vinyals, and Q. Le, “Sequence to sequence learning with neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2014, pp. 3104–3112.

[19] A. Graves, “Generating sequences with recurrent neural networks,” 2013, arXiv:1308.0850. [Online]. Available: http://arxiv.org/abs/1308.0850

[20] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” 2014, arXiv:1406.1078. [Online]. Available: http://arxiv.org/abs/1406.1078

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” 2014, arXiv:1412.3555. [Online]. Available: http://arxiv.org/abs/1412.3555

[22] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
HYEWON SONG received the B.S. degree from Hongik University, Seoul, South Korea, in 2015. She is currently pursuing the Ph.D. degree with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul. Her main research interests include computer vision and machine learning.

BEOM KWON was born in Seoul, South Korea, in 1989. He received the B.S. degree from Soongsil University, Seoul, in 2012, and the M.S. and Ph.D. degrees from Yonsei University, Seoul, in 2018, all in electrical and electronic engineering. From March 2018 to September 2019, he was a Senior Researcher with the Agency for Defense Development (ADD), Daejeon, South Korea. Since October 2019, he has been a Senior Engineer with Samsung Electronics Company, Ltd., Suwon, South Korea. His research interests include wireless communication networks, computer vision, and machine learning.

HOON YOO received the B.S., M.S., and Ph.D. degrees in electronic communications engineering from Hanyang University, Seoul, South Korea, in 1997, 1999, and 2003, respectively. From 2003 to 2005, he was with Samsung Electronics, South Korea where he was involved in the world’s first development of the satellite digital multimedia broadcasting (DMB) and the terrestrial DMB on mobile phones. He is currently a Full Professor with the Department of Electronics, Sangmyung University, Seoul. His research interests include 3-D integral imaging, image processing and recognition, and digital signal processing.

SANGHOON LEE (Senior Member, IEEE) received the B.S. degree from Yonsei University, Seoul, South Korea, in 1989, the M.S. degree from the Korea Advanced Institute of Science and Technology, South Korea, in 1991, and the Ph.D. degree from The University of Texas at Austin, USA, in 2000. From 1991 to 1996, he was with Korea Telecom, South Korea. From 1999 to 2002, he was with Lucent Technologies, Murray Hill, NJ, USA. In 2003, he joined the EE Department, Yonsei University, as a Faculty Member, where he is currently serving as a full-time Professor. His current research interests include image/video processing, computer vision, and graphics. He received the 2015 Yonsei Academic Award from Yonsei University, the 2012 Special Service Award from the IEEE Broadcast Technology Society, and the 2013 Special Service Award from the IEEE Signal Processing Society. He was the General Chair for the 2013 IEEE IVMSP Workshop and served as a Steering Committee Member for the IEEE and APSIPA conferences. He has been serving as the Chair for the IEEE P3333.1 Quality Assessment Working Group, since 2011. He was the IV Moss Technical Committee Chair for APSIPA, from 2018 to 2019. He is also a Board of Governors Member of APSIPA. He was the IEEE IVMSP Technical Committee Member, from 2014 to 2019. He has also been the IEEE MMSP Technical Committee Member, since 2016. He served as the Editor for the Journal of Communications and Networks, from 2009 to 2015, and as a special issue Guest Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, in 2013. He was an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, from 2010 to 2014. He served as an Associate Editor, from 2014 to 2018. He is also a Senior Area Editor of the IEEE SIGNAL PROCESSING LETTERS.