Received 16 November 2023, accepted 27 November 2023, date of publication 7 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3340441

RESEARCH ARTICLE

ByteRCNN: Enhancing File Fragment Type Identification With Recurrent and Convolutional Neural Networks

KRISTIAN SKRAČIĆ, (Senior Member, IEEE), JURAJ PETROVIĆ, (Senior Member, IEEE), AND PREDRAG PALE, (Senior Member, IEEE)
Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia
Corresponding author: Juraj Petrović (juraj.petrovic@fer.hr)

This work was supported by the Advanced Computing Service provided by the University of Zagreb University Computing Centre (SRCE).

ABSTRACT File fragment type identification is an important step in file carving and data recovery. Machine learning techniques, especially neural networks, have been utilized for this problem, some with very promising results. This paper presents a novel neural network architecture for identifying file fragment types using a combination of byte embeddings as well as recurrent and convolutional elements. The corresponding classification model, ByteRCNN, has been trained on the publicly available file fragment FiFTy dataset and evaluated in closed-set and open-set recognition settings using FiFTy and other available file fragment datasets. Evaluation results have demonstrated that ByteRCNN can compete with state-of-the-art models described in literature in terms of classification accuracy, with 71.1% average accuracy on 512-byte fragments and 83.9% average accuracy on 4,096-byte fragments from the FiFTy dataset. When evaluated on other publicly available datasets in closed-set and open-set recognition settings, ByteRCNN performs similarly or slightly better than the FiFTy classification model. Obtained results overall suggest that ByteRCNN is a competitive file fragment classification model, but they also reveal that there is still plenty of space for further improving file type identification methods using more complex datasets or in open-set recognition settings. ByteRCNN is publicly available at https://github.com/kristian-fer/ByteRCNN.

INDEX TERMS File fragment type identification, file fragment classification, byte embeddings, recurrent neural network, convolutional neural network, open-set recognition, memory forensics, carving.

I. INTRODUCTION

With an increase in data storage usage, the need to accurately identify the file type based on file fragments has become increasingly important in digital forensics and law enforcement investigations. File fragment type identification [1], also known as *file fragment classification* [2], *file type recognition*, or *file type detection* [3], refers to the process of identifying and categorizing partial or incomplete files found on digital storage media. File fragmentation occurs when files are stored on a disk in non-contiguous clusters, resulting in a file being broken down into smaller parts – *fragments* or *blocks*. These fragments can be scattered across the disk, making it difficult to recover and connect them. However, even if an entire file cannot be recovered, the identification of the file type for individual fragments can provide valuable information for investigations [4]. The identification of file fragment types, therefore, is a crucial step in digital forensics and criminal investigations as it provides valuable insight into the contents of digital devices and can assist in uncovering potential evidence.

The goal of file fragment type identification is to classify an unknown sequence of bytes into one of the predefined file types, in the absence of any additional side information. In this context it is important to stress the difference between *file type identification* and *data type identification*. Data type identification is more granular and encompasses complex combinations where one data type is embedded into another, for example, when images are inserted into PDF documents or base64-encoded images are inserted into HTML textual

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

files. Creating a data type dataset is significantly more complex and requires a specialised carving logic which may not be supported for most data types currently in use [1]. Therefore, most file fragment type identification research is focused on file type identification since it relies on the type of file that the fragment originally belonged to.

The research described in this paper is focused on file fragment type identification and advancing its current state of the art through several contributions. First, we argue and demonstrate how competitive results in file fragment type identification can be achieved by implementing a neural network with its architecture inspired by sequence processing models that are typically used in natural language processing. This way, we utilize temporal information stored in the sequence of bytes of a file fragment. A comprehensive comparison of the proposed model (ByteRCNN) with other state-of-the-art file type identification models is provided in the paper in terms of accuracy, but also in terms of the inference time and the number of parameters. Classification accuracy is evaluated on the FiFTy scenario #1 dataset [1].

Second, we evaluate the proposed model on other publicly available datasets [2], [3], [4], [5] which have by now remained ignored by other researchers. Those datasets provide valuable data for model evaluation. Since the FiFTy classification model is currently the only publicly available state-of-the-art file fragment type identification model trained and evaluated on the FiFTy scenario #1 dataset, we compare ByteRCNN performance on datasets [2], [3], [4], [5] to the FiFTy classification model.

Third, we address the issue of open-set recognition [6], which is most relevant for the file type identification problem, but also currently unaddressed in related research. We utilize fragments belonging to file types included in datasets [2], [3], [4], [5] that are not included in the FiFTy scenario #1 dataset to evaluate our open-set recognition accuracy.Again, since the FiFTy classification model is currently the only publicly available state-of-the-art file fragment type identification model trained and evaluated on the FiFTy scenario #1 dataset, we compare open-set recognition performance of ByteRCNN to the FiFTy classification model.

Finally, we make our model implementation publicly available at [GitHub, after publication] to facilitate future model comparisons and advancement of file fragment type identification.

The remainder of this paper is organized as follows. The related research section introduces state-of-the-art datasets and approaches to file fragment classification. The proposed model section describes the proposed ByteRCNN model in terms of its structure, inputs and outputs. In the experimental evaluation section, ByteRCNN is compared to state-of-the-art models for file fragment type identification performance in closed-set and open-set conditions. Finally, in the conclusion section we provide concluding remarks on the proposed model, its results as well as possible future research directions.

II. RELATED RESEARCH

A. FILE FRAGMENT DATASETS

A substantial portion of previously published work related to file fragment type identification relied on limited private datasets or the publicly available GovDocs dataset [7], also referred to as *govdocs1*. Having a common dataset is critical to training and comparing performance of different machine learning models. While the use of private datasets renders the replication of results impossible, a number of published file fragment type identification models have employed the GovDocs corpus [7], [8], [9], [10]. The GovDocs dataset consists of an imbalanced collection of publicly available files of different types, of which 20 file types constitute 99.3% of the dataset, while the remaining 0.7% of the dataset is comprised of 43 file types. This introduces the problem of choosing the file types and fragments to be included in model training and evaluation. Researchers using GovDocs typically chose to include different subsets of file types and different ways of generating fragments from those files [11], [12]. Since GovDocs lacks some file types which are prevalent today (for example, DOCX, PPTX, and XLSX), some authors use GovDocs-based datasets supplemented with other relevant file types [10] or they use completely customized datasets for specialized applications such as video [13] or image [14], [15], [16], [17] fragment classification and file carving.

In contrast to GovDocs, the FiFTy dataset [1] contains pre-extracted fragments divided into several subsets or scenarios (see **TABLE 1**), thereby facilitating a fair comparison between file fragment type identification outcomes. The FiFTy main dataset (scenario #1) offers a publicly available dataset comprising 75 distinct file types. This dataset is, to the best of the authors' knowledge, the most comprehensive and extensive labelled set of file fragments available to the public. The authors compiled the dataset by collecting public files from Internet sources as well as their own collections. To ensure a balanced dataset suitable for classification, the authors sampled 102,400 fragments, both 512-byte and 4,096-byte, for each file type. The fragments were then randomly shuffled to distribute the file types evenly among the training (80%), validation (10%), and hold-out testing (10%) subsets. Aside from the main dataset (scenario #1), the FiFTy dataset includes five additional specialized scenarios that focus on a specific subset of included file types. **TABLE 1** outlines the details of each scenario. Most of the five additional scenarios are use-case specific and focus on various multimedia file types. Scenario #1, which comprises all 75 file types, is currently the most comprehensive publicly available file fragment dataset. The FiFTy dataset was utilized in training the ByteRCNN model, which was subsequently evaluated against state-of-the-art approaches.

Four other file fragment datasets have recently been published to assist in file fragment classification research. They are focused on the classification of audio file formats [2], video file formats [3], textual file formats [4], and image file formats [5]. All four of those datasets contain

1,024-byte fragments belonging to a variety of included file types. A summary of those four datasets is provided in **Table 2**.

TABLE 1. **FiFTy - file fragment datasets and scenarios (adapted from [1]).**

Scenario	Description
#1 All 75 classes	The main dataset which contains all 75 file types.
#2 Use-specific	Files are grouped into 11 classes which describe their use. For example, RAW, bitmap, video, archive, executable, office, audio, misc, etc.
#3 Photos and videos	Only considers file types that are photos or videos.
#4 Coarse photo	Separate classes for different photographic types.
#5 Specialised JPEG	Only contains 2 classes, JPEG and all others are grouped into "other".
#6 Camera specialised JPEG	Only contains 2 classes, JPEG and all others are grouped into "other". The difference between scenario #5 is that the files chosen for the "other" label mimic files that would be found on an SD card from digital cameras.

TABLE 2. **File fragment datasets [2], [3], [4], [5].**

Dataset description	File types
Audio files [2] • 20 audio file formats with different compression settings • 1,008 fragments of 1,024 bytes per file type • 20,160 fragments in total	AMR, AWB, AAC, AIFF, CVSD, FLAC, GSM-FR, LBC, Microsoft ADPCM, MP3, PCM, WMA, ALAW, ULAW, G726, G729, Microsoft GSM, OGG, OPUS, SPEEX
Video files [3] • 10 video file formats, 20 different pairs of video format and codec • 30,000 fragments of 1,024 bytes per file type • 300,000 fragments in total	3GP, AVI, ASF, FLV, MKV, MOV, MP4, WEBM, OGV, RMVB
Textual files [4] • 5 textual file formats in three different languages • 4,500 fragments of 1,024 bytes per file type • 22,500 fragments in total	DOC, DOCX, PDF, RTF, TXT
Image files [5] • 10 image file formats with different compression settings • 2,560 fragments of 1,024 bytes per file type • 25,600 fragments in total	BMP, BPG, FLIF, GIF, JPEG, JPEG2000, JPEGXR, PNG, TIFF, WEBP

While some of the datasets [2], [3], [4], [5] have been used in validating file fragment classification models [18], most state-of-the-art research seems not to be utilizing their potential and is focused only on using the FiFTy dataset [19], [20], [21], [22].

B. APPROACHES TO FILE FRAGMENT TYPE IDENTIFICATION

In recent years, the field of file fragment type identification has received significant attention due to its increasing importance for digital forensics, cybersecurity and content-based file analysis. Different approaches utilizing machine learning algorithms have been applied to this problem in order to automate the process and increase identification accuracy.

Early approaches to file fragment type identification mostly relied on hand-engineered features including statistical measures, byte frequency distribution and n-grams. N-Grams refer to frequencies of occurrence of different combinations of n bytes in the file fragment. The use of byte frequency distribution was initially proposed in [23] where the average classification accuracy reported was 27.5%. The same research describes experiments with byte frequency cross-correlation which achieves 45.83% average classification accuracy on a private dataset of 30 file types used to train the models. Veenman [24] used linear discriminant analysis to classify 28 file type fragments using byte frequency distribution and the Kolmogorov complexity measure. The model achieved 45% average accuracy. Li et al. [25] used support vector machines to address the challenging task of classifying file fragments belonging to high entropy file types. Their approach also relied on byte histograms and achieved promising results of 81.50% average accuracy, yet only focused on 4,096-byte fragments belonging to one of four file types: DLL, PDF, JPG and MP3.

Sceadan [26] is one of the first significant milestones in file fragment type identification. Its authors used support vector machines (SVM) to classify file fragments using unigram and bigram (1-gram and 2-gram) features and achieved the classification accuracy of 73.8% across 38 different file types. N-Grams or n-gram-derived features remain frequently used in the newer approaches [27], [28] and can also be used to perform a detailed comparison of different classification approaches. For example, Seste et al. [29] perform a detailed comparison of support-vector machines and neural networks applied for identifying file fragment types, by focusing on the n-gram analysis as a feature for the two different classifiers.

More recently, Wang et al. [10] used the sparse coding approach to extract features from 512-byte fragments belonging to a total of 18 most common file types. Their dataset was based on the GovDocs dataset and supplemented with OOXML file types (DOCX, XLSX, PPTX) due to their relevance today. Using a combination of unigrams, bigrams and sparse n-gram features, they achieved a maximum average accuracy of 61.31%.

Another recent milestone in identifying file fragment types, FiFTy [1], has achieved excellent results without the need to rely on hand-engineered fragment features. The corresponding model is capable of classifying file fragments belonging to any of the 75 different file types with average accuracy of 65.6% when using 512-byte blocks, and 77.5% when using 4,096-byte blocks. As already mentioned, FiFTy provides six separate datasets or scenarios with a different number of included file types. Each scenario is classified with a different model. However, all six models share a similar architecture which is based on 1-D convolutional neural networks (CNNs) which take blocks of raw bytes as input and embed them into a trainable latent space. Shifting individual bytes into a latent space was inspired by the current state-of-the-art natural language processing models where words, or sub-words, are embedded into a common latent space before being sent through a neural network [30], [31], [32], [33], [34]. The use of byte embeddings instead of 1-hot encoding or hand-crafted features such as input is, arguably, one of the key insights offered by the FiFTy research paper. This allows the original input sequence to be captured in a latent space. Overall, FiFTy has achieved the best classification results in the field on the most comprehensive reported dataset and has also provided future research with a dataset to allow for fair and relevant comparisons.

Other researchers have sometimes reported higher classification accuracy scores compared to FiFTy, yet usually on significantly smaller datasets. Innovative approaches to file fragment type identification described in such research have made valuable contributions to the field, yet they have made the comparison of results more difficult. Haque and Tozal [9], for example, explored using the word2vec algorithm on byte sequences. Word2vec is another prominent algorithm used in natural language processing for transforming words into comparable vectors based on their co-occurrence. The authors achieved 72% classification accuracy only by using the averaged byte embedding vectors and a kNN classifier on a dataset of 35 file types. Chen et al. [7] converted 4,096-byte file fragments into 64×64 pixel greyscale images. A deep 2-D convolutional neural network (CNN) was trained to distinguish between 16 different file types that were sampled from the GovDocs dataset. The model achieved 70.9% accuracy. Interestingly, the same model achieved very high accuracy (92%) on the compressed GZ file type which is usually classified quite poorly by most existing models, even FiFTy (13.2%). However, the model ignored other popular compressed archive files such as 7Z and ZIP, which could explain its high score, and achieved rather low accuracy for textual files and office documents (DOC, DOCX and PPT). Hiester [35] explored the usage of feedforward, recurrent and convolutional networks as file fragment classifiers. While using only 4 different file types, he obtained the classification accuracy of 98%, thus revealing potentially promising research directions that should be further explored.

The file fragment classification issue has gained a lot of attention in recent years. Some of the research based on convolutional neural networks managed to obtain results better than the original FiFTy classification models using the FiFTy dataset. Saaim et al. [19] succeeded in that task by using depthwise separable convolutions (DSCNN) for file fragment type identification and achieved excellent results of 78.45% and 65.89% average accuracy on scenario #1 of the FiFTy dataset for 4,096-byte and 512-byte fragment size respectively. Ghaleb et al. [20] experimented with several convolutional neural network architectures and managed to outperform the original FiFTy classifiers in 4 out of 6 FiFTy scenarios for 512-byte fragments, and in 1 out of 6 FiFTy scenarios for 4,096 byte fragments (DSC-SE model [20]). Zhu et al. [21] managed to obtain results that are similar to or in some cases better than the results of the original FiFTy scenario #1 classifier by using convolutional neural networks to learn higher level representations of file fragments as well as by using a long short-term memory network (LSTM) to classify them. Finally, Liu et al. [22] managed to outperform FiFTy classifiers in nearly all FiFTY scenarios by interpreting file fragments as 2-dimensional grey-scale images. Currently, to the best of the authors' knowledge, these results are the highest in the field.

C. OPEN-SET RECOGNITION

To the best of our knowledge, all published results related to the issue of file fragment classification pertain to the problem of *closed-set recognition*, where the unknown fragment is assumed to belong to one of the known classes. In *open-set recognition*, on the other hand, no such assumption is made, and the classifier has to be able to infer whether a fragment belongs to one of the known classes or to an unknown class [6]. In forensic practice or data recovery this situation is significantly more realistic due to a variety of file types used in practice.

There are many open-set recognition approaches, often tailored to a specific context of classification problems they are used for [36]. Among them, OpenMax (*Open set Recognition with Maximum Softmax Probability*) [37] is one of the more commonly used ones. OpenMax is used for detecting data instances that lie beyond the confines of the training data distribution. It operates as a post-processing step following the traditional *softmax* classification layer in a neural network model and it introduces several critical parameters, including the alpha parameter, thresholds, and the definition of unknown classes. The alpha parameter (α) controls the distribution of OpenMax scores for known classes, impacting the decision boundary between the known and unknown classes. A higher alpha value makes it more challenging for an input to be classified as a known class. Thresholds are employed to determine the likelihood at which an input is classified as a known or unknown class. Additionally, the model requires the softmax scores generated by the neural network, which represent the model's confidence in class assignments. OpenMax has been used in this paper to evaluate the developed classification model in open-set classification conditions.

III. PROPOSED MODEL – ByteRCNN

A. MODEL DESIGN

The file fragment type identification model proposed in this paper, ByteRCNN (**Figure 1**), draws inspiration from recent breakthroughs in natural language processing that incorporate a combination of recurrent and convolutional neural

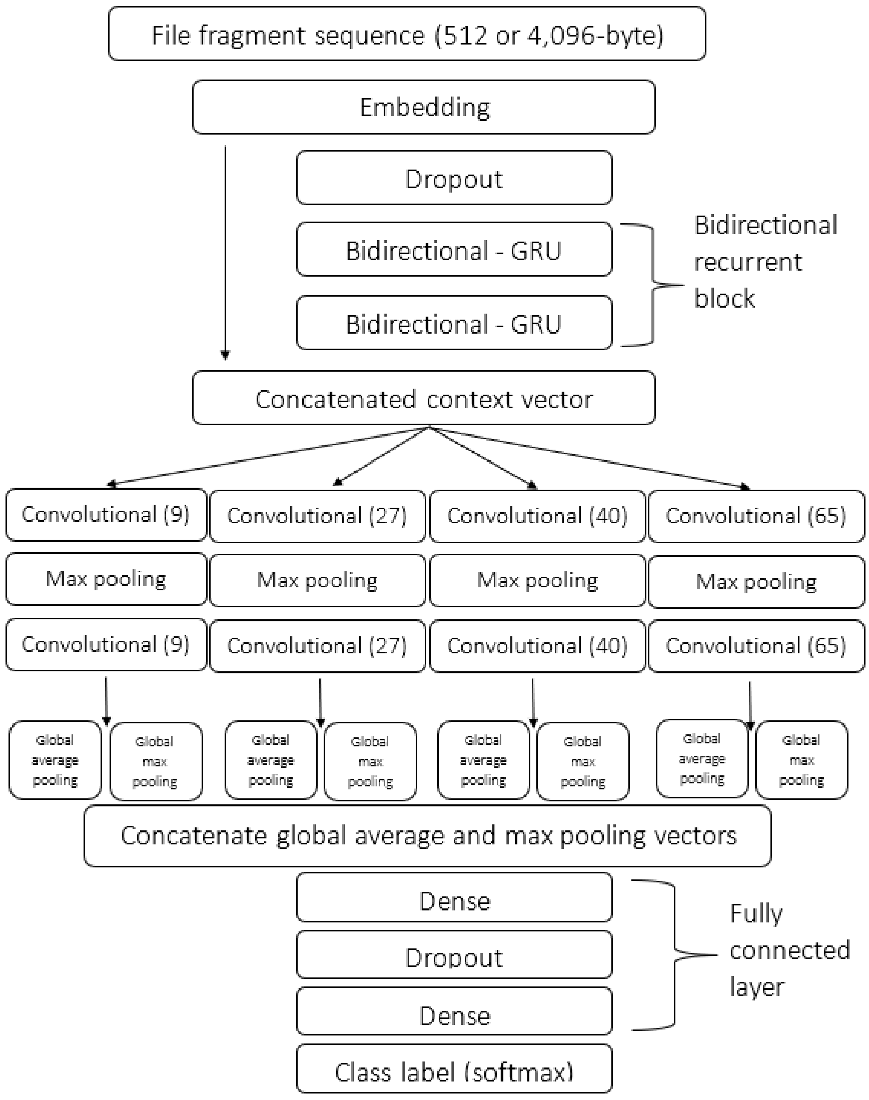

FIGURE 1. **ByteRCNN model.**

networks. Such networks have shown great potential for advancing text classification. Convolutional neural networks, renowned for their success in computer vision, have demonstrated considerable efficacy in text classification, notably exemplified by the FiFTy model discussed in the previous section. Nonetheless, they are incapable of capturing the temporal dynamics of sequential data. To address this shortcoming, recurrent neural networks (RNNs) were designed to encode temporal information in sequential data, such as sentences.

The proposed model concept involves, in the first place, utilizing an RNN to extract significant sequences from the input, followed by employing a CNN to eliminate the bias arising from later elements in the sequence overshadowing the earlier ones, thus extracting the most meaningful sequences. In the context of file fragment type identification, meaningful sequences could correspond to a particular combination of bytes that frequently occur in specific file types. However, some elements in the sequence, such as padding or null bytes, may negatively impact the classification outcome since they are shared across multiple file types and do not contribute to differentiation. Furthermore, due to a substantial number of distinct file types, it is conceivable that the relevant sequences that uniquely identify a fragment may vary in length. Consequently, using only one type of CNN may not suffice to identify the relevant sequences across all file types. Therefore, we posit that model width may be more significant than its depth.

In contrast to manually crafted features such as byte n-grams, which merely capture the frequency of byte occurrences without considering their sequence relative to other file types, an RNN possesses the capacity to discern significant sequences from file fragment bytes. This feature extraction capability significantly enhances the discriminatory power of the model, allowing for superior classification performance.

TABLE 3. Comparison of classification accuracy of ByteRCNN with other research using FiFTy dataset scenarios.

Scenario	**#1**		**#2**		**#3**		**#4**		**#5**		**#6**	
#of Classes	75		11		25		5		2		2	
Sequence size (bytes)	512	4,096	512	4,096	512	4,096	512	4,096	512	4,096	512	4,096
FiFTy [1]	65.6	77.5	78.9	89.8	87.9	94.6	90.2	94.1	99.0	99.2	99.3	**99.6**
DSCNN [19]	65.9	78.5	74.8	85.7	80.8	93.1	87.1	94.2	98.9	99.3	98.8	**99.6**
DSC-SE [20]	66.3	79.3	75.0	87.1	80.8	93.1	87.3	94.6	99.0	**99.4**	98.7	**99.6**
ResNet18 [22]	71.0	82.1	**90.4**	**94.2**	**93.5**	**96.8**	**93.6**	**96.1**	**99.2**	99.3	99.2	99.4
CNN-LSTM [21]	66.5	78.6	/	/	/	/	/	/	/	/	/	/
Ours (ByteRCNN)	**71.1**	**83.9**	87.5	93.1	91.0	96.5	92.0	95.4	99.0	99.3	**99.5**	99.5

B. MODEL ARCHITECTURE

The architecture of our proposed model is illustrated in FIG. 1. The ensuing section describes its components, parameter search space, and the parameter combination leading to the optimal model performance.

The model comprises an embedding layer that is situated immediately after the input layer, which translates each byte into a high-dimensional latent space. The dimensions of the embedding layer are contingent on the sequence length and the number of unique values. In this case it is the number of distinct bytes, which is a constant value of 256. The sequence length is determined by the length of the fragment and prior research typically employs sequences of 512 bytes or 4,096 bytes. Finally, the embedding layer is characterized by the length of the vector in the resulting latent space. The FiFTy model explores various embedding dimensions, including 16, 32, 48 and 64. In FiFTy, different scenarios were established using distinct embedding dimensions. Our investigation suggests that modifying the embedding dimensions does not significantly impact the performance of the proposed model. Therefore, we opt for the smallest embedding dimension, namely a length of 16, which reduces the model size and enhances its processing speed.

After the embedding layer we place a dropout layer which helps reduce overfitting [38], [39]. We conducted simple parameter tuning for the best dropout probability value. The optimal value was 10%.

Following the dropout layer, we place two consecutive bidirectional recurrent blocks. For the recurrent block we chose the gated recurrent unit (GRU) [40]. In our experiments we tried using the Long Short-Term Memory (LSTM) cells, but the results were slightly lower when compared to using GRU as the recurrent cell. The use of the bidirectional recurrent block was motivated by recent NLP advances [41] which used separate inputs as right and left context embedders. In essence, processing each word in our model entailed analysing its surrounding context on both left and right. To simplify this input procedure, we employed a bidirectional recurrent block, which duplicates the recurrent layer within the network, resulting in two parallel layers that process the input sequence from left to right and right to left. The GRU cell, which reconstructs the sequence, was evaluated with different values for the number of units, ranging from 32 to 128. Our experimental findings indicate that the optimal number of units for the GRU cells is 64. Further increasing of the number of units did not appear to improve model performance. Parameter tuning was performed using the HyperBand approach [42]. The optimization objective was set to increase the overall accuracy, and the maximum number of epochs was set to 100 with a reduction factor of three.

The model has two consecutive bidirectional GRU layers, as shown in FIG. 1, which share the same number of GRU cells. Adding multiple bidirectional GRU layers to a sequence classification model can improve its accuracy by allowing the model to capture more complex patterns and dependencies in the input sequence data. Each layer of a bidirectional GRU captures different abstraction levels in the input sequence. The first layer might capture low-level features, while subsequent layers may capture higher-level patterns and dependencies. This hierarchical feature extraction enables the model to learn more informative representations of the data, which can improve classification performance. Stacking multiple layers of the bidirectional GRU introduces non-linearity to the model. Complex patterns in sequence data are often non-linear, and by adding more layers the model can learn increasingly complex mappings from input to output, which can lead to better classification performance. Additionally, stacking bidirectional GRU layers can help capture these long-term dependencies by allowing information to flow bidirectionally through multiple layers. In general, increasing the number of layers in a neural network boosts its capacity to fit the training data. By adding more layers, we effectively create an ensemble of models within a single architecture. Each layer provides a different perspective on the data, and combining these perspectives can improve the overall performance.

Once the recurrent blocks had extracted crucial sequences from the input, we concatenated the embedding layer and the output of the second recurrent block to create a combined vector. This approach was used to augment the correlation between the original input sequence and the recurrent layers' interpretation of it. To identify the relevant patterns that may exist in certain file types, we explored the concatenated vector using various processing parameters to account for the differences in the pattern length. This approach involved constructing several parallel channels with a one-dimensional convolutional layer followed by a max pooling layer and another one-dimensional convolutional layer. To optimize the performance of each convolutional layer, we tested different filter values, ultimately determining that 64 filters produced the best results.

We used four parallel channels with different kernel sizes for the convolutional layers in each channel. By using different kernel sizes, the model extracts patterns of different length. This approach was used in NLP for text classification [43]. The model was fine-tuned based on evaluating its performance using different kernel sizes in the parallel channels, and the best performance was achieved by using the kernel sizes of 9, 27, 40 and 65.

The output of each of the parallel channels is processed using global max and average pooling. The resulting vectors are concatenated together to create one vector which is the input to the fully connected layer. The configuration of the layer was achieved through a straightforward parameter optimization process, wherein we experimented with varying numbers of layers and dense units. Ultimately, the optimal architecture was determined to be a two-layer dense neural network, comprising 1,024 units in the first layer, followed by 512 units in the second layer. Adding more layers did not improve the overall classification accuracy. In order to prevent overfitting, a dropout layer with a probability of 10% was introduced between the dense layers. The final output layer is a softmax function with dimensions equal to the number of classes.

IV. EXPERIMENTAL EVALUATION

A. PERFORMANCE EVALUATION ON FIFTY DATASET

The proposed model was implemented in the Tensorflow framework version 2.10 and all evaluation experiments were run on NVIDIA A100 with 40GB of memory running CUDA version 11.6. The evaluation of the model was performed using the FiFTy dataset for all six defined scenarios. In each of the six FiFTy scenarios, the corresponding dataset was split into the 80:10:10 ratio for training, testing and validation respectively. The classification results were compared with the results reported in [1], [19], [20], [21], and [22] since those models provide currently the best published results, and they were trained and evaluated on the FiFTy dataset, thereby enabling a fair comparison as well as reproducibility.

As shown in **Table 3**, the ByteRCNN model outperforms models [1], [19], [20], [21] in all FiFTy scenarios except in two cases: in scenario #5 for 4,096-byte fragments, in which all five compared models perform within the 99.3±0.1% accuracy range, and in scenario #6 for 4,096-byte fragments, in which all five compared models perform within the 99.5±0.1% accuracy range. The ByteRCNN performance is second only to ResNet18 [22],which generally performs better than ByteRCNN, except in case of scenario #1 and scenario #6. ByteRCNN achieves the best average accuracy among all compared models in scenario #1 (71.1% for 512-byte fragments and 83.9% for 4,096-byte fragments) and scenario #5 (99.5% for 512-byte fragments). It is worth noting that the ByteRCNN model performs very well across all six scenarios, thus avoiding the need to train or use multiple models, and that it has the highest performance among all compared models in case of scenario #1 which consist of the highest number of classes (75) among all scenarios.

Table 4 shows accuracy breakdown for each of the 75 classes included in FiFTy scenario #1 for ByteRCNN, CNN-LSTM [21], and FiFTy [1], as researchers in [19], [20], and [22] did not disclose per-class classification accuracies of their models. When compared to FiFTy and CNN-LSTM, ByteRCNN provides the best classification accuracy for 65 out of 75 classes for 512-byte and 46 out of 75 classes for 4,096-byte fragments. As it was expected, the longer input proved to have more data which the recurrent elements could use to extract deeper patterns of the file type. By looking at **Table 4** we also notice that certain file types consistently achieve lower accuracy scores. Those are typically file archives (BZ2, XZ), executable binaries (EXE), and video files (MOV, MP4, AVI), all achieving accuracy below 20%. We still see improvements in classifying some of these file types using ByteRCNN.

The difficulty in classifying archives is that they often contain other file types. For example, given that we have several images and textual documents compressed in a single archive, when we attempt to extract their fragments from the compressed archive, we are effectively taking the compressed versions of the original files - images and text. In most cases image file formats already have some type of compression, so that traditional compression methods such as ZIP, 7Z, and BZ2 will not have much effect on the overall file fragment byte sequence compared to the original. Text, on the other hand, will be compressed much more efficiently, thereby significantly changing the underlying fragment byte sequence. In addition, due to the fact that compression archives work similarly, they are often confused for one another or for the file types they contain. Most notably, the ZIP file type is most often confused for the GZ file type and for the DOCX file type. ZIP and GZ file types work in a very similar way, so it is not surprising that they are confused for one another. The DOCX file type is part of the Microsoft OOXML standard, which is a compressed archive itself and very similar to the ZIP archive.

Executable binaries from various platforms, including EXE, RPM, PKG, DEB, and MSI, also exhibit low classification accuracy. These file types, commonly found on Windows and Linux operating systems, contain executable code as well as other file formats such as images and text. Due to their embedding of various file types, they are often mistaken for both executable files and compressed archives.

Video files contain long segments of compressed data. Different video containers can support same codecs making it difficult to correctly predict a codec based on a fragment of a video file's content. Additionally, data compression can make it difficult to differentiate that fragment from fragments belonging to other compressed file types like images compressed archives. Among video files, MOV and MP4 file types have the lowest classification accuracy among all three

TABLE 4. Classification results per file type for FiFTy [1], CNN-LSTM [21], and our model (ByteRCNN).

File type	FiFTy	CNN-LSTM	Ours	FiFTy	CNN-LSTM	Ours
	512			4,096		
ARW	97.8	96.4	**98.0**	98.9	98.3	**99.4**
CR2	86.7	87.6	**94.3**	94.8	**95.6**	12.5
DNG	82.7	80.0	**89.2**	96.1	**97.0**	78.9
GPR	99.2	99.4	**99.8**	**99.9**	**99.9**	98.3
NEF	87.7	86.4	**92.5**	95.3	**95.6**	67.6
NRW	96.9	95.0	**97.0**	97.7	98.0	**99.6**
ORF	86.3	87.4	**92.8**	96.3	96.9	**99.7**
PEF	95.1	94.3	**97.1**	99.1	**99.2**	98.0
RAF	98.3	98.6	**99.2**	87.1	89.4	**91.4**
RW2	96.5	93.8	**96.8**	97.9	**98.0**	96.4
3FR	99.6	**99.8**	99.6	99.5	99.6	**99.7**
JPG	83.5	87.0	**90.4**	86.3	90.3	**99.4**
TIFF	96.1	95.6	**96.6**	**99.0**	97.4	48.7
HEIC	31.7	**36.2**	5.6	49.5	46.5	**62.0**
BMP	98.0	**98.1**	**98.1**	98.4	**99.6**	88.4
GIF	93.5	**93.6**	93.5	99.1	**99.3**	38.9
PNG	67.2	74.2	**76.4**	88.0	85.4	**97.9**
AI	23.3	25.0	**39.1**	57.7	55.4	**93.1**
EPS	98.7	**98.8**	**98.8**	95.8	**99.0**	66.4
PSD	95.3	95.4	**96.2**	95.9	96.0	**99.0**
MOV	6.1	**15.2**	14.0	18.5	16.5	**97.8**
MP4	1.6	**12.3**	5.6	71.8	68.5	**98.9**
3GP	85.6	86.0	**90.2**	98.1	**98.9**	55.5
AVI	10.9	14.8	**16.8**	67.1	**70.1**	52.4
MKV	88	88.0	**92.1**	98.4	**99.3**	87.4
OGV	57.3	65.0	**81.7**	94.9	94.9	**98.7**
WEBM	39.8	**42.7**	42.5	78.1	79.5	**99.8**
APK	29.9	31.2	**46.7**	49.7	47.4	**57.4**
JAR	41.2	44.3	**54.2**	73.8	**74.0**	13.7
MSI	10.1	9.8	**22.0**	60.1	57.5	**99.8**
DMG	18.2	16.4	**27.3**	19.4	24.8	**94.3**
7Z	0	7.6	**28.2**	0.1	23.2	**87.4**
BZ2	13.9	11.6	**17.0**	80.6	82.6	**100.0**
DEB	**13.8**	12.7	12.2	0.8	2.9	**76.9**
GZ	13.2	15.0	**23.5**	42.1	55.1	**100.0**
PKG	4.5	7.3	**17.5**	62.7	57.5	**96.8**
RAR	24.0	23.6	**27.6**	46.6	39.4	**99.0**
RPM	13.6	12.5	**15.4**	21.9	22.4	**99.8**
XZ	0	**6.1**	4.5	12.1	9.5	**99.4**
ZIP	13.8	15.1	**24.1**	34.2	38.5	**99.6**
EXE	0.3	6.8	**9.8**	2.7	4.9	**14.1**
MACH-O	92.6	94.2	**95.1**	95	96.0	**99.3**
ELF	85.9	87.6	**88.9**	86.4	92.8	**96.9**
DLL	91.1	92.2	**94.0**	91.7	**94.5**	61.3
DOC	87.2	89.0	**92.4**	88.9	90.0	**96.6**
DOCX	19.4	17.7	**23.5**	60.6	63.4	**98.5**
KEY	44.1	44.3	**56.9**	43.1	49.8	**98.7**
PPT	41.3	38.2	**58.3**	53.5	55.6	**95.3**
PPTX	44.4	41.3	**59.6**	36	37.5	**98.1**
XLS	99.3	**99.5**	**99.5**	99.3	99.5	**99.7**
XLSX	95.2	96.9	**98.4**	97.2	**98.0**	84.7
DJVU	**24.6**	22.8	**24.6**	30.4	40.8	**98.5**
EPUB	20.7	24.8	**33.3**	79.7	**80.0**	49.2
MOBI	72.4	73.0	**73.1**	73.7	74.7	**87.2**
PDF	23.1	22.2	**28.9**	45.8	46.7	**61.3**
MD	97.1	97.4	**98.7**	97.4	**99.9**	56.5
RTF	99.7	99.7	**99.9**	99.8	**100**	98.9
TXT	93.7	93.4	**95.9**	93	93.2	**97.9**
TEX	97.6	**97.9**	97.6	98.3	**98.9**	44.2
JSON	99.5	99.6	**99.9**	99.8	**99.9**	51.5
HTML	97.5	97.6	**98.6**	99.6	99.5	**99.8**
XML	**100**	**100**	**100**	**100**	**100**	97.7
LOG	99.9	**100**	**100**	99.9	99.9	**100.0**
CSV	99.6	99.6	**99.8**	**99.7**	99.6	99.5
AIFF	99.4	**99.7**	**99.7**	98.8	**99.0**	95.2
FLAC	**74.2**	70.4	65.8	97.9	95.6	**99.7**
M4A	94.3	97.3	**98.9**	98.2	**98.3**	96.8
MP3	94.4	95.0	**98.2**	98.9	**99.0**	97.4
OGG	90.4	90.3	**95.8**	95.5	**96.0**	87.9
WAV	**100**	**100**	**100**	98.7	98.8	**100**
WMA	98.2	98.7	**99.2**	**99.9**	**99.9**	**99.9**
PCAP	51	49.3	**59.9**	95.4	**95.5**	**95.5**
TTF	98.1	98.7	**99.3**	99.3	**99.4**	**100**
DWG	96.2	96.3	**96.7**	96.9	**98.8**	20.2
SQLITE	98.6	99.4	**99.6**	**99.8**	98.5	62.1
Total wins	4	15	**65**	6	26	**46**

classification models compared in the Table 4. The MOV file type only achieves 6.1% and 18.5% accuracy when using the FiFTy model for 512-byte and 4,096-byte fragments respectively, while CNN-LSTM performs slightly better with 15.2% and 16.5% accuracy for 512-byte and 4,096-byte fragments respectively. In ByteRCNN the MOV file type is frequently confused for HEIC and DJVU file types which are used to store images, as well as for compressed archives that likely contained multimedia data that could not be compressed efficiently, thus introducing a bias into the training dataset. Interestingly, the MP4 file type displays considerable improvements when switching from 512-byte to 4,096-byte fragments as input to the classifiers. The FiFTy model achieves 1.6% and 71.8% accuracy, CNN_LSTM achieves 12.3% and 68.5% accuracy, whereas ByteRCNN achieves 5.6% and 98.9% accuracy, all on 512-byte and 4,096-byte fragments, respectively. All three models exhibit significant improvements when using the longer 4,096-byte fragments. It is evident across all tests that the longer sequences of 4,096-bytes yield better results, as more patterns emerge and are learned by the model.

What is also notable across all models, including ByteRCNN, is that there is a lot of semantic overlap between several classes, which ultimately lowers the classifiers' ability to produce good results. As described earlier in this section, compressed archives contain any combination of file types possible. Most multimedia file types cannot be compressed further as they already have their own compression (such as PNG and JPEG). Therefore, producing very similar byte sequences for compressed archives (such as ZIP and GZ) and multimedia files (such as MOV, JPEG, and PNG). There is currently no way of knowing which file type the compressed

TABLE 5. Performance of ByteRCNN (ours) and FiFTy scenario #1 model on 512-byte fragments from datasets [2], [3], [4], [5] closed-set recognition.

Audio file formats [2]			Image file formats [3]			Textual file formats [4]			Video file formats [5]		
File type	FiFTy	Ours	File type	FiFTy	Ours	File type	FiFTy	Ours	File type	FiFTy	Ours
AIFF	18.67	25.5	JPG	13.4	6.2	DOCX	3.6	12.3	AVI	5.28	7.26
MP3	24.6	15.2	GIF	93.8	86.1	RTF	25.3	11.71	MOV	0.9	3.5
OGG	39.4	33.7	PNG	3.25	4.3	DOC	24.69	50.2	3GP	47.4	43.2
FLAC	17.5	12.2	BMP	6.7	6.1	TXT	8.4	10.7	OGV	47.1	61.0
WMA	21.2	20.3				PDF	29.2	19.9	MP4	0.6	1.3
									MKV	2.7	0.9
									WEBM	12.8	15.0
Overall*	26.5	22.7		21.1	17.9		18.1	20.2		11.8	12.7

* Since datasets [2]–[5] are not balanced, the overall average accuracy is different from the average of per-class accuracies

TABLE 6. Comparison between FiFTy and our model in terms of inference time and model Parameters.

Model	Block size	# Params	Speed [ms/block]	GFLOPS
FiFTy (Scenario #1)	512	289,995	2	0.11
	4,096	449,867	7	0.52
Ours (ByteRCNN)	512	5,500,000	34	0.61
	4,096	5,500,000	300	10.53

fragment contained since the FiFTy dataset does not make this information available. A similar case can be made for other complex file types such as DOCX and PDF. Those files may also contain images and text and they are compressed.

B. PERFORMANCE EVALUATION ON OTHER DATASETS

The ByteRCNN model was also validated on datasets [2], [3], [4], [5]. Since those datasets contain 1,024-byte fragments, each of those fragments was randomly subsampled to obtain a 512-byte sub-fragment. This validation was performed only on fragments belonging to classes from datasets [2], [3], [4], [5] that ByteRCNN was trained on, i.e. file types that are among the 75 file types included in the FiFTy scenario #1 dataset. The classification results for the file types in closed-set recognition settings are presented in **Table 5** and are compared to the classification results of the same data using the FiFTy model trained on scenario #1. This comparison was performed only by using the FiFTy scenario #1 model since the models described in related research [19], [20], [21], [22] are not publicly available.

The results presented in **Table 5** suggest that datasets [2], [3], [4], [5] provide fragment examples that both FiFty and ByteRCNN find challenging to classify, even though they have been trained on those file types. ByteRCNN performs, on average, very similar to FiFTy classifier on each of the four datasets. Classification accuracy for each of those datasets is between 11.8% and 26.5%, which is significantly lower than the results achieved on the FiFTy scenario #1 dataset in **Table 3**. Both models achieve the worst results on video file formats, which is likely due to the high entropy of video compression codecs, but also due to the fact that different video file types can support the same codecs. As indicated

TABLE 7. Average accuracy for open-set recognition of FiFTy and ByteRCNN models on the FiFTY scenario #1 dataset.

Model	Sequence size (bytes)	Closed-set average accuracy	Open-set average accuracy
FiFTy	512	65.6	58.1
	4,096	77.5	68.5
Ours (ByteRCNN)	512	69.5	68.5
	4,096	81.6	71.4

in the results and discussion related to **Table 3**, these results overall suggest that classification accuracy is significantly lower when there is greater diversity in the properties of sampled files' content, even if they belong to known file types. This diversity is, in this case, introduced by using different codecs and compression settings in video and audio files, by using different compression settings in image files as well as different languages and scripts in textual files.

C. INFERENCE TIME EVALUATION

The comparison between the FiFTy classifier for scenario #1 and ByteRCNN in terms of inference time and the number of model parameters is presented in **Table 6**. The results suggest that ByteRCNN has an approximately 10 to 20 times higher number of parameters and approximately 10 to 40 times slower inference time. This is not surprising since as the number of layers increases, the number of parameters in CNN models increases exponentially, which negatively affects the training and inference time [19]. While a direct comparison would require running experiments on the same hardware,

TABLE 8. Performance of ByteRCNN (ours) and FiFTy scenario #1 model on 512-byte fragments from datasets [2], [3], [4], [5] open-set recognition.

Audio file formats [2]			Image file formats [3]			Textual file formats [4]**			Video file formats [5]		
File type	FiFTy	Ours	File type	FiFTy	Ours	File type	FiFTy	Ours	File type	FiFTy	Ours
AIFF	15.9	3.0	JPG	10.9	5.8	DOCX	3.6	6.7	AVI	4.2	0.8
MP3	19.9	9.8	GIF	73.1	21.6	RTF	23.0	4.5	MOV	0.9	3.5
OGG	32.0	10.6	PNG	3.0	4.1	DOC	19.5	10.3	3GP	38.0	14.9
FLAC	15.2	9.0	BMP	5.6	2.8	TXT	7.0	3.0	OGV	38.1	21.2
WMA	17.3	8.1	*Unknown*	4.5	3.3	PDF	24.2	4.1	MP4	0.6	0.6
Unknown	11.3	28.3							MKV	2.3	0.8
									WEBM	11.0	6.3
									Unknown	14.4	37.2
Overall*	14.7	21.9		9.5	4.8		15.3	5.6		10.2	9.16

* Since datasets [2]–[5] are not balanced, the overall average accuracy is different from the average of per-class accuracies
** Since all classes from [4] are included in the FiFTy dataset, there is no *Unknown* class in this case

ByteRCNN is significantly slower that the FiFTy scenario #1 model and is therefore also slower than [19] and [20].

D. OPEN-SET RECOGNITION

ByteRCNN was next validated in open-set recognition settings on the FiFTy dataset for scenario #1 and on datasets [2], [3], [4], [5]. This comparison was performed only by using the FiFTy scenario #1 model since models described in related research [19], [20], [21], [22] are not publicly available.

To use those two models in the open-set recognition context, we conducted a detailed study of parameter optimization in the OpenMax framework, specifically to improve accuracy when recognizing fragments belonging to unknown classes. This involved carefully adjusting alpha and threshold parameters for each model. By making these adjustments, our goal was to fully utilize the OpenMax framework and to enhance the classifiers' ability to distinguish fragments belonging to unknown classes. Every model has a different probability distribution on the final softmax layer, so the confidence scores on which the OpenMax is based will vary for each model. Alpha coefficient values were examined in the range from 0.1 to 1 with 0.1 resolution and in the range from 1 to 75 (the number of classes) with the resolution of 1. The best results were achieved for $\alpha = 1$. Increasing α from 1 to 75 did not provide an improvement in classification outcomes, as already noted in [37]. The threshold values were examined in the range from 0.1 to 1 with 0.1 resolution. Based on the analysis results, we finally used the alpha coefficient $\alpha = 1$ and the threshold of 0.2 for the FiFTy model as well as the alpha coefficient $\alpha = 0.8$ and the threshold of 0.8 for ByteRCNN.

The classification results of open-set recognition of FiFTy scenario #1 dataset for FiFTy scenario #1 model and ByteRCNN are presented in **Table 7**. As expected, the average accuracy of both models is now slightly lower than the closed-set recognition results presented in **Table 3**. This difference in average accuracy is the result of classifiers wrongly assigning the unknown class to some fragments in the FiFTy scenario #1 dataset.

The classification results for open-set recognition of the FiFTy scenario #1 model and ByteRCNN achieved on datasets [2], [3], [4], [5] are presented in **Table 8**. Overall, the average accuracy for recognizing the *unknown* class ranges from 3.3 to 37.2% for ByteRCNN and from 4.5% to 14.4% for the FiFTy scenario #1 model. Still, those results are achieved on datasets that proved to be challenging for classification, even in closed-set conditions (see **Table 5**). The challenge here remains to both improve classification accuracy for known classes as well as to improve the accuracy of detecting samples belonging to the *unknown* class.

E. ABLATION STUDY

As stated in Section III/B MODEL ARCHITECTURE, we used the HyperBand tuner to find the optimal parameters for the model. Interestingly, we found that a larger size for the embedding dimension neither greatly improved nor degraded model performance. We hypothesized that this was due to the relatively low cardinality of the input vocabulary which consists of only 256 possible values for bytes. Given that each byte can, in theory, be seen close to any other byte, a larger embedding space would not necessarily bring better representations on the single byte level. While increasing the vector size may lead to better representation, there are diminishing returns. Once the vector size has become sufficiently large, the additional benefit in performance starts to plateau or even decrease. Instead, the main gain is achieved by using the entire sequence utilizing bidirectional GRU blocks and subsequent CNN blocks. To verify that these elements truly bring the highest gains, an ablation study was performed.

Table 9 summarises the results obtained by the study and confirms that using two consecutive, bidirectional GRU blocks achieves higher accuracy, while higher embedding dimensions do not bring any significant gains. The table

TABLE 9. Ablation study results.

Number of consecutive bidirectional blocks	Embedding dimensions	Scenario #1 accuracy
1	16	80.03
1	32	80.21
1	64	80.77
2	16	83.51
2	32	83.59
2	64	83.72

shows results for the 4,096-byte model since it has the highest likelihood of leveraging the increased parameter space given the longer inputs.

V. CONCLUSION

This paper describes a novel machine learning model, ByteRCNN, which is based on byte embeddings as well as convolutional and recurrent elements for file fragment type identification. Through training and evaluation on the FiFTy scenario #1 dataset, we demonstrate that ByteRCNN performs similarly and in some cases better than state-of-the-art models in terms of classification accuracy (71.1% average accuracy on 512-byte fragments and 83.9% average accuracy on 4,096-byte fragments from the FiFTy scenario #1 dataset), while consisting of more parameters and being slower than some of them [1], [19], [20]. Notably, we achieved this result with a single model across different settings, whereas other relevant research relied on different models.

ByteRCNN evaluation on other datasets, however, revealed that there is still a lot of room for improving classification accuracy in the field of file fragment type identification, in the context of both open-set and closed-set recognition. Since no other state-of-the-art classification models are publicly available, we support this conclusion by evaluating the FiFTy scenario #1 model on the same datasets, showing that the classification results on datasets [2], [3], [4], [5] are significantly lower compared to the classification results on the FiFTy scenario #1 dataset. Fragments coming from media files with different codec and compression settings seem to be challenging for classification in closed-set and open-set conditions.

The obtained results suggest possible directions for future research. Since a number of efficient file fragment type identification models have been described in literature, it would be wasteful if their implementations were not published. Publishing their implementations would enable the obtaining of their per-class accuracies to identify their individual strengths and weaknesses, and even using them in parallel to increase the overall classification results. Open-set recognition also needs to be addressed in future research to take into account more challenging, yet more realistic file fragment type identification settings. Finally, more research should be dedicated to the available or new datasets containing fragments from diverse files belonging to the same file type. We hope that ByteRCNN and the results published in this paper will help direct that research.

REFERENCES

[1] G. Mittal, P. Korus, and N. Memon, "FiFTy: Large-scale file fragment type identification using convolutional neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 28–41, 2021, doi: 10.1109/TIFS.2020.3004266.

[2] A. Khodadadi and M. Teimouri, "Dataset for file fragment classification of audio file formats," *BMC Res. Notes*, vol. 12, no. 1, p. 819, Dec. 2019, doi: 10.1186/s13104-019-4856-1.

[3] N. Sadeghi, M. Fahiminia, and M. Teimouri, "Dataset for file fragment classification of video file formats," *BMC Res. Notes*, vol. 13, no. 1, p. 213, Apr. 2020, doi: 10.1186/s13104-020-05037-x.

[4] F. Mansouri Hanis and M. Teimouri, "Dataset for file fragment classification of textual file formats," *BMC Res. Notes*, vol. 12, no. 1, p. 801, Dec. 2019, doi: 10.1186/s13104-019-4837-4.

[5] R. Fakouri and M. Teimouri, "Dataset for file fragment classification of image file formats," *BMC Res. Notes*, vol. 12, no. 1, p. 774, Nov. 2019, doi: 10.1186/s13104-019-4812-0.

[6] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1757–1772, Jul. 2013, doi: 10.1109/TPAMI.2012.256.

[7] Q. Chen, Q. Liao, Z. L. Jiang, J. Fang, S. Yiu, G. Xi, R. Li, Z. Yi, X. Wang, L. C. K. Hui, D. Liu, and E. Zhang, "File fragment classification using grayscale image conversion and deep learning in digital forensics," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 140–147, doi: 10.1109/SPW.2018.00029.

[8] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digit. Invest.*, vol. 6, pp. 2–11, Sep. 2009, doi: 10.1016/j.diin.2009.06.016.

[9] M. E. Haque and M. E. Tozal, "Byte embeddings for file fragment classification," *Future Gener. Comput. Syst.*, vol. 127, pp. 448–461, Feb. 2022, doi: 10.1016/j.future.2021.09.019.

[10] F. Wang, T.-T. Quach, J. Wheeler, J. B. Aimone, and C. D. James, "Sparse coding for n-Gram feature extraction and training for file fragment classification," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2553–2562, Oct. 2018, doi: 10.1109/TIFS.2018.2823697.

[11] E. Mina and S. Jalili, "File fragment type classification by bag-of-visual-words," *ISC Int. J. Inf. Secur.*, vol. 13, no. 2, pp. 101–116, 2021, doi: 10.22042/isecure.2021.243876.570.

[12] M. Masoumi, A. Keshavarz, and R. Fotohi, "File fragment recognition based on content and statistical features," *Multimedia Tools Appl.*, vol. 80, no. 12, pp. 18859–18874, May 2021, doi: 10.1007/s11042-021-10681-x.

[13] E. Altinisik and H. T. Sencar, "Automatic generation of H.264 parameter sets to recover video file fragments," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4857–4868, 2021, doi: 10.1109/TIFS.2021.3118876.

[14] E. Uzun and H. T. Sencar, "JPG *Scraper*: An advanced carver for JPEG files," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1846–1857, 2020, doi: 10.1109/TIFS.2019.2953382.

[15] E. Durmus, P. Korus, and N. Memon, "Every shred helps: Assembling evidence from orphaned JPEG fragments," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2372–2386, Sep. 2019, doi: 10.1109/TIFS.2019.2897912.

[16] J. De Bock and P. De Smet, "JPGcarve: An advanced tool for automated recovery of fragmented JPEG files," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 19–34, Jan. 2016, doi: 10.1109/TIFS.2015.2475238.

[17] Z. Seyedghorban and M. Teimouri, "Anomaly detection in file fragment classification of image file formats," in *Proc. 11th Int. Conf. Comput. Eng. Knowl. (ICCKE)*, Oct. 2021, pp. 248–253, doi: 10.1109/ICCKE54056.2021.9721457.

[18] K. Skračić, J. Petrović, and P. Pale, "Classification of low- and high-entropy file fragments using randomness measures and discrete Fourier transform coefficients," *Vietnam J. Comput. Sci.*, pp. 1–30, Jul. 202. [Online]. Available: https://www.worldscientific.com/doi/10.1142/S2196888823500070, doi: 10.1142/s2196888823500070.

[19] K. M. Saaim, M. Felemban, S. Alsaleh, and A. Almulhem, "Light-weight file fragments classification using depthwise separable convolutions," *IFIP Adv. Inf. Commun. Technol.*, vol. 648, pp. 196–211, Jun. 2022, doi: 10.1007/978-3-031-06975-8_12.

[20] M. Ghaleb, K. Saaim, M. Felemban, S. Al-Saleh, and A. Al-Mulhem, "File fragment type classification using light-weight convolutional neural networks," 2023, *arXiv:2305.00656*.

[21] N. Zhu, Y. Liu, K. Wang, and C. Ma, "File fragment type identification based on CNN and LSTM," in *Proc. 7th Int. Conf. Digit. Signal Process.* New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 16–22, doi: 10.1145/3585542.3585545.

[22] W. Liu, Y. Wang, K. Wu, K.-H. Yap, and L.-P. Chau, "A byte sequence is worth an image: CNN for file fragment classification using bit shift and n-Gram embeddings," in *Proc. IEEE 5th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2023, pp. 1–5, doi: 10.1109/AICAS57966.2023.10168636.

[23] M. McDaniel and M. H. Heydari, "Content based file type detection algorithms," in *Proc. 36th Annu. Hawaii Int. Conf. Syst. Sci.*, Jan. 2003. [Online]. Available: https://www.researchgate.net/publication/221182341_Content_Based_File_Type_Detection_Algorithms, doi: 10.1109/hicss.2003.1174905.

[24] C. J. Veenman, "Statistical disk cluster classification for file carving," in *Proc. 3rd Int. Symp. Inf. Assurance Secur.*, Aug. 2007, pp. 393–398, doi: 10.1109/ias.2007.75.

[25] Q. Li, A. Ong, P. Suganthan, and V. Thing, "A novel support vector machine approach to high entropy data fragment classification," in *Proc. South Afr. Inf. Secur. Multi-Conf. (SAISMC)*, Jan. 2010, pp. 236–247.

[26] N. L. Beebe, L. A. Maddox, L. Liu, and M. Sun, "Sceadan: Using concatenated n-Gram vectors for improved file and data type classification," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 9, pp. 1519–1530, Sep. 2013, doi: 10.1109/TIFS.2013.2274728.

[27] M. Teimouri, Z. Seyedghorban, and F. Amirjani, "Fragments-Expert: A graphical user interface MATLAB toolbox for classification of file fragments," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 9, p. e6154, May 2021, doi: 10.1002/cpe.6154.

[28] K. Vulinovic, L. Ivkovic, J. Petrovic, K. Skracic, and P. Pale, "Neural networks for file fragment classification," in *Proc. 42nd Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2019, pp. 1194–1198, doi: 10.23919/MIPRO.2019.8756878.

[29] J. Sester, D. Hayes, M. Scanlon, and N.-A. Le-Khac, "A comparative study of support vector machine and neural networks for file type identification using n-Gram analysis," *Forensic Sci. Int., Digit. Invest.*, vol. 36, Apr. 2021, Art. no. 301121, doi: 10.1016/j.fsidi.2021.301121.

[30] M. R. Vivek and P. Chandran, "Analysis of subword based word representations case study: Fasttext Malayalam," in *Proc. IEEE 19th India Council Int. Conf. (INDICON)*, Nov. 2022, pp. 1–6, doi: 10.1109/INDICON56171.2022.10040147.

[31] S. Nazir, M. Asif, S. A. Sahi, S. Ahmad, Y. Y. Ghadi, and M. H. Aziz, "Toward the development of large-scale word embedding for low-resourced language," *IEEE Access*, vol. 10, pp. 54091–54097, 2022, doi: 10.1109/ACCESS.2022.3173259.

[32] S. Dlamini, E. Jembere, and A. Pillay, "Evaluation of word and subword embeddings for isiZulu on semantic relatedness and word sense disambiguation tasks," in *Proc. Int. Conf. Artif. Intell., Big Data, Comput. Data Commun. Syst. (icABCD)*, Aug. 2020, pp. 1–6, doi: 10.1109/icABCD49160.2020.9183836.

[33] A. Akdemir, T. Shibuya, and T. Güngör, "Subword contextual embeddings for languages with rich morphology," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2020, pp. 994–1001, doi: 10.1109/ICMLA51294.2020.00161.

[34] S. Wang, W. Zhou, and C. Jiang, "A survey of word embeddings based on deep learning," *Computing*, vol. 102, no. 3, pp. 717–740, Mar. 2020, doi: 10.1007/s00607-019-00768-7.

[35] L. Hiester. (May 2018). *File Fragment Classification Using Neural Networks with Lossless Representations*. Honors Theses, Undergrad. [Online]. Available: https://dc.etsu.edu/honors/454

[36] A. Mahdavi and M. Carvalho, "A survey on open set recognition," in *Proc. IEEE 4th Int. Conf. Artif. Intell. Knowl. Eng. (AIKE)*. Washington, DC, USA: IEEE Computer Society, Dec. 2021, pp. 37–44, doi: 10.1109/AIKE52691.2021.00013.

[37] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1563–1572, doi: 10.1109/CVPR.2016.173.

[38] B. Ko, H.-G. Kim, and H.-J. Choi, "Controlled dropout: A different dropout for improving training speed on deep neural network," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 972–977, doi: 10.1109/SMC.2017.8122736.

[39] K. Sanjar, A. Rehman, A. Paul, and K. JeongHong, "Weight dropout for preventing neural networks from overfitting," in *Proc. 8th Int. Conf. Orange Technol. (ICOT)*, Dec. 2020, pp. 1–4, doi: 10.1109/ICOT51877.2020.9468799.

[40] 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," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734, doi: 10.3115/v1/d14-1179.

[41] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proc. 29th AAAI Conf. Artif. Intell.* Austin, TX, USA: AAAI Press, Jan. 2015, pp. 2267–2273.

[42] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 185, pp. 1–52, Apr. 2018.

[43] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751, doi: 10.3115/v1/d14-1181.

KRISTIAN SKRAČIĆ (Senior Member, IEEE) was born in Zagreb, Croatia, in 1988. He received the B.S., M.S., and Ph.D. degrees in computing from the Faculty of Electrical Engineering and Computing (FER), University of Zagreb, Croatia, in 2018.

He is currently a Research Associate with the FER, University of Zagreb. He is the author of several scientific articles. His research interests include computer forensics and the application of computer science and machine learning in industrial applications.

JURAJ PETROVIĆ (Senior Member, IEEE) was born in Zagreb, Croatia, in 1986. He received the B.S. degree in computing, the M.S. degree in information and communication technology, and the Ph.D. degree in computing from the Faculty of Electrical Engineering and Computing (FER), University of Zagreb, in 2017.

He is currently an Assistant Professor with the FER, University of Zagreb. He is the coauthor of more than 25 scientific articles. His research interests include the application of technology in learning and assessment, information security, computer forensics, and digital signal processing.

Dr. Petrović is an active IEEE volunteer and the Vice President of the E25 Education Chapter of the IEEE Croatia.

PREDRAG PALE (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering and the Ph.D. degree in computer science from the Faculty of Electrical Engineering and Computing (FER), University of Zagreb.

He is currently an Associate Professor with the FER, University of Zagreb, and among other subjects teaches computer forensics and digital file forensics. In addition, he creates and runs courses on cybersecurity topics for national and international ICT experts, middle and top management, educators, parents, and the general public.

Dr. Pale is a member of the American Academy of Forensic Science and an expert court witness. He received the Distinguished Chapter Leadership Award from the IEEE Education Society. He also received the medal of The Order of Croatian Star with the Effigy of Ruer Bošković, the State Award for Science, and the Croatian IEEE Chapter Outstanding Engineering Contribution Award. He is listed in the Croatian Encyclopedia. He was a keynote speaker on information warfare for NATO's top generals at the "Open Road 99" Conference. He is often teaching at NATO and RACVIAC workshops.