

Size-agnostic file fragment classification via fixed byte slicing

Hyeongsik Kim¹  | Sisung Liu²  | Je Hyeong Hong³ 

¹Department of Artificial Intelligence Semiconductor Engineering, Hanyang University, Seoul, Republic of Korea

²Department of Artificial Intelligence, Hanyang University, Seoul, Republic of Korea

³Department of Electronic Engineering, Hanyang University, Seoul, Republic of Korea

Correspondence

Je Hyeong Hong, Department of Electronic Engineering, Hanyang University, Seoul, Republic of Korea.
Email: jhh37@hanyang.ac.kr

Funding information

This work was supported by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) grant funded by the Korea government (Defense Acquisition Program Administration [DAPA]) (No. KRIT-CT-22-021, Space Signal Intelligence Research Laboratory, 2022).

Abstract

File fragment classification is an essential task of identifying the file type given an incomplete binary file fragment. Despite their importance, existing deep-learning methods face two key limitations: (1) extensive hyperparameter tuning for varying fragment sizes and (2) considerably degraded inference performance when tested on fragment sizes different from those used during training. To address these challenges, we propose the byte-segment convolution and attention network (ByteSCAN) that introduces a fixed-slicing approach. Fixed slicing divides the binary data into overlapping segments of uniform size, enabling consistent feature extraction with a moderately deep convolutional neural network (CNN) and efficient feature integration through a self-attention layer. Experiments on two public datasets show that ByteSCAN outperforms previously published methods in terms of competitive training time and efficient inference while also demonstrating adaptability to varying fragment sizes. The code is available at <https://github.com/SpatialAILab/ByteSCAN>.

KEYWORDS

convolutional neural network, file fragment classification, fixed slicing, self-attention

1 | INTRODUCTION

File fragment classification (FFC) is a key technology in digital forensics and cybersecurity that involves the process of identifying types of incomplete file fragments. This enables the identification of the intrinsic nature of the data, even in situations where file system metadata are absent. FFC is extensively applied in file recovery [1], malware detection [2], and data integrity verification and plays a crucial role in digital evidence collection and strengthening cybersecurity.

Conventional FFC approaches rely on manually crafted features combined with classical machine-learning models [3–6]. However, the advent of deep learning has advanced the field. Mittal and others [7] demonstrated the effectiveness of convolutional neural networks (CNNs) over previous methods, and Saaime and others [8] improved the efficiency using the GoogLeNet [9] architecture. Liu and others [10] further enhanced the performance through bit-shifting data augmentation. Recently, RNN models, such as ByteRCNN [11] and transformer-based approaches [12–14], have achieved state-of-the-art results.

Despite recent advancements, the existing methods still face limitations owing to the unique characteristics

Hyeongsik Kim and Sisung Liu contributed equally to this work.

This is an Open Access article distributed under the term of Korea Open Government License (KOGL) Type 4: Source Indication + Commercial Use Prohibition + Change Prohibition (<http://www.kogl.or.kr/info/licenseTypeEn.do>).

1225-6463/\$ © 2026 ETRI

of file fragments. Unlike images, file fragments consist of bytes with distinct semantic meanings that make resizing impractical. Consequently, models trained on a specific fragment size often experience severe performance degradation when tested on fragments of different and unseen sizes—in some cases, they fail to perform inferences altogether. In a real-world digital forensic investigation in which file fragments of unpredictable and varying sizes are the norm, this limitation is a critical flaw that undermines the reliability of the analysis. Separate training models for different fragment sizes have been attempted; however, this approach substantially increases the overall training cost. Moreover, when an unseen fragment size that is not included in the training data appears, additional training is often required or leads to a considerable drop in accuracy. In addition, models such as XMP exhibit high sensitivity to hyperparameter settings (for example, kernel size and number of layers) depending on the fragment size, which further increases the burden of tuning and computational costs, as listed in Table 1.

Motivated by these issues, we propose ByteSCAN, which is a simple, yet efficient model designed to address these challenges. The core of our proposal is fixed slicing, which divides any incoming file fragment into fixed-size overlapping segments. This strategy ensures a consistent input size at the segment level, rendering the model insensitive to the overall size of the original fragment. Each segment is then processed using a moderately deep CNN for local feature extraction, followed by a shallow self-attention mechanism to integrate the global context. This optimized design not only achieves robust performance across various fragment sizes but also simplifies the training process by reducing the need for complex hyperparameter tuning.

Through extensive experiments, we demonstrate ByteSCAN's effectiveness, achieving state-of-the-art

results on two public datasets while noticeably reducing training costs. In addition, we present an ablation study that highlights the contribution of each component to our model.

Our key contributions are the following:

- We adopt a fixed-slicing approach to ensure consistent input data sizes, removing the need for extensive hyperparameter tuning while ensuring robust feature extraction across varying fragment sizes.
- We propose an efficient architecture that utilizes fixed slicing, as previously mentioned, combining pretrained CNNs for feature extraction on the fixed-size segments with a self-attention layer for global context, enabling parallel processing.
- Experimental results showing ByteSCAN's robustness to varying fragment sizes, achieving state-of-the-art performance on public datasets with competitive training time.

2 | PROPOSED METHOD

2.1 | Motivation

Inspired by the Vision Transformer (ViT) [15]'s patch-based self-attention for context capture, ByteSCAN addresses the high computational costs and long training times associated with ViT's deep-attention layers.

To address these issues in the context of FFC, we were inspired by a Detection Transformer (DETR) [16] that applies a deep CNN backbone to capture the local context in the images. We adapted this approach by using "fixed slicing" to split file data into smaller segments and utilized a deep CNN backbone to extract rich local features. These locally extracted features were then combined using self-attention, similar to the approach in ViT

TABLE 1 Comparison of recent and state-of-the-art file fragment classification approaches.

Model	Key characteristics	Limitations
XMP [13]	• Transformer with multiscale cross-attention • Captures long-range dependencies between pairs of bytes	• Slower inference due to large Transformer • Sensitive to hyperparameters
JSANet [14]	• Lightweight byte/channel/sector self-attention • Models intrasector byte importance & inter-sector context	• Needs retraining when fragment size changes
ByteResNet/ByteFormer [12]	• 1D/2D byte representation via 2D augmentation • Leverages image based augmentation tricks	• Needs retraining when fragment size changes
ByteSCAN (ours)	• Byte-segment local feature extraction & integration • Fixed slicing $\Rightarrow$ robust to fragment size • No size-specific hyperparameter tuning	• More parameters than JSANet/ByteResNet

and DETR, allowing us to capture the global context while reducing the computational overhead.

Furthermore, transformer-based models often require additional hyperparameter tuning based on input size. ByteSCAN mitigates this dependency using a fixed-slicing technique and an architecture composed of a CNN and a self-attention layer, achieving robust performance across diverse file sizes.

2.2 | Training strategy

In our approach, the learning process consists of two stages: backbone pretraining and attention layer training. Figure 1 shows an overview of the proposed framework.

In the first stage, a CNN backbone is pretrained on the cropped segments $d \in \mathbb{R}^l$ from each input file fragment $D \in \mathbb{R}^L$. L is the total length of the input data, and l is the size of each segment. To enable the CNN backbone to handle varying segment positions and avoid overfitting, it learns the generalized features from these cropped segments. This ensured robust feature extraction and effective generalization across different file fragment sizes without performance loss.

In the second stage, the pretrained backbone is frozen, and the self-attention layer is trained. The entire input file fragment D was divided into multiple segments to ensure that all parts were used for training. The backbone extracts features independently from each segment,

and the self-attention layer integrates these features to model the global context.

2.3 | Byte slicing and attention-based feature integration

To implement this segmentation, we apply herein a fixed-slicing mechanism that divides the input data D into overlapping segments. The selection of the segment size l and the overlap S are critical to the model's effectiveness. We chose a segment size of $l = 512$ bytes because it is large enough to capture meaningful local patterns and structural motifs within a file fragment yet small enough to be processed efficiently by the CNN backbone. An overlap of $S = 256$ bytes was implemented to ensure that important byte sequences were not truncated at the segment boundaries and to help the model learn the continuity between adjacent segments. This process resulted in N segments, where $N = (L - l) / S + 1$.

Each segment d_i is independently passed through a frozen CNN backbone G , which is pretrained on segments of size l . The CNN extracts a local feature vector $\mathbf{f}_i$ for each segment, $\mathbf{f}_i = G(d_i)$, where $\mathbf{f}_i \in \mathbb{R}^{\frac{1}{16} \times C}$. The symbol C is the number of feature channels. This process captures meaningful local structures within each segment and effectively creates a sequence of rich feature representations.

The features extracted from all the segments are then gathered to form a matrix of row-stacked features

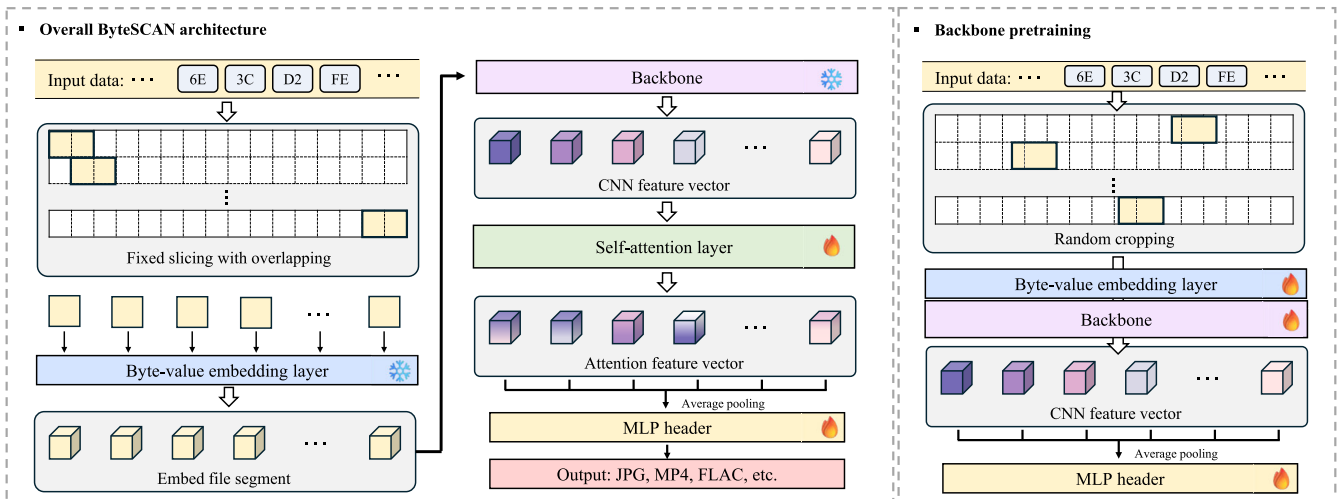


FIGURE 1 An overview of our ByteSCAN architecture. The input file fragments are divided into overlapping segments via fixed slicing and passed through a byte-value embedding layer. A deep CNN backbone extracts local feature vectors from each segment. These feature vectors are then processed by a shallow self-attention layer to capture global relationships between segments. The final classification is performed through an MLP header. For backbone pretraining, input data are randomly cropped into smaller segments for feature extraction, ensuring the model generalizes across different file fragment sizes.

$F := [\mathbf{f}_1, \dots, \mathbf{f}_{N/16}]^T$, which serves as the input to a shallow self-attention layer. This layer is designed to model the global context by capturing the dependencies between different segments. It applies linear transformations to F to generate query (Q), key (K), and value (V) representations and subsequently computes the attention output A as

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, A \in \mathbb{R}^{N \times C}, \quad (1)$$

where a softmax operation was performed for each row. The self-attention layer outputs A , which represents the N segments as context-aware row vectors. We then apply global average pooling across the row dimension to reduce A to a single feature vector $\mathbf{a} \in \mathbb{R}^C$. This consolidated vector, which encapsulates both local and global information, is fed into a final fully connected layer that produces classification scores $s \in \mathbb{R}^M$, where M is the number of file types.

To train the model, the following standard cross-entropy loss function was used:

$$\mathcal{L} = - \sum_{j=1}^M y_j \log\left(\frac{\exp(s_j)}{\sum_{l=1}^M \exp(s_l)}\right), \quad (2)$$

where s_j is the j th element of s and y_j is the corresponding ground truth label (in one-hot encoding) for the j th file type. By combining detailed local feature extraction with a self-attention mechanism, the model learns to capture the intricate relationships among different segments and understand the overall global context of the given input data. This integrated architecture ensures a robust and consistently efficient performance across various input data sizes.

3 | EXPERIMENTAL RESULTS AND DISCUSSION

3.1 | Implementation details

In our implementation, we set the training data D to a size of $L = 4096$ bytes, which is the standard fragment size extensively adopted in previous studies [7,8,13]; this size corresponds to the typical cluster size in modern file systems (for example, NTFS), providing a realistic experimental setup. To demonstrate the applicability of our method, we used segments of length $l = 512$ bytes with an overlap of $S = 256$ bytes. We employed ResNet18-1D as the CNN backbone, and our model incorporated a single

attention layer with eight heads to combine CNN features. The key architectural choices for this backbone, such as the primary kernel size (set to 27) and the CNN output dimensionality fed into the attention layer (set to 512), were guided by empirical evidence and successful configurations reported in other publications, such as FiFTy, ByteRCNN, and XMP.

All the experiments were conducted using a single NVIDIA A6000 graphics processing unit. Training utilized the Adam optimizer with an initial learning rate of 10^{-3} , and the batch size was set to 128. We employed the ReduceLROnPlateau scheduler and early stopping with the maximum number of epochs set to 100. For the 1024- or 2048-byte settings, we applied the same hyperparameters that were optimized for the 4096-byte settings. Table 2 summarizes the detailed architecture and training configuration of ByteSCAN.

3.2 | Datasets

3.2.1 | FFT-75 [7]

This study utilized the FFT-75 dataset designed for the FFC. This dataset comprises 75 file types, each with 102 400 samples, and provides data having the sizes of 512 and 4096 bytes. To evaluate the model's ability to generalize across different fragment sizes, we randomly cropped 4096-byte samples into datasets comprising 512, 1024, and 2048 bytes. Our study focuses on Scenario 1, the most comprehensive and challenging task, which treats all 75 file types as classes. Table 3 lists detailed file extensions in each category.

TABLE 2 Architecture and training details of ByteSCAN.

Parameter	Value
Data size	All
Embedding size	256
CNN backbone	ResNet18-1D
Network depth	8 (CNN) + 1 (attention)
CNN kernel size	27
CNN output channels	512
Optimizer	Adam
Initial learning rate	10^{-3}
Batch size	128
Scheduler	ReduceLROnPlateau
Early stopping	Enabled (patience: 11)
Max epochs	100

3.2.2 | GovDocs [18]

The GovDocs dataset includes approximately 1 M files composed of 63 file types. We selected 13 file types (CSV, GIF, GZ, HTML, JPG, LOG, XLS, PPT, DOC, RTF, PDF, PNG, and TXT) that overlapped with FFT-75 and contained sufficient data. GovDocs contains full files; therefore, we cropped the 4096-byte data into segments of 512, 1024, and 2048 bytes to maintain consistency.

3.3 | Comparison against other methods

We compared our method with existing FFC methods using two experiments. The first experiment

TABLE 3 File categories and representative extensions.

Category	File extensions
Archive (Arc)	apk, jar, msi, dmg, 7z, bz2, deb, gz, pkg, rar, rpm, xz, zip
Audio (Aud)	aiff, flac, m4a, mp3, ogg, wav, wma
Bitmap (Bmp)	jpg, tiff, heic, bmp, gif, png
Executable (Exe)	exe, mach-o, elf, dll
Human-readable (Hum)	md, rtf, txt, tex, json, html, xml, log, csv
Office (Off)	doc, docx, key, ppt, pptx, xls, xlsx
Published (Pub)	djvu, epub, mobi, pdf
Raw	arw, cr2, dng, gpr, nef, nrw, orf, pef, raf, rw2, 3fr
Vector (Vec)	ai, eps, psd
Video (Vid)	mov, mp4, 3gp, avi, mkv, ogv, webm
Miscellaneous (Mis)	pcap, ttf, dwg, sqlite

evaluated the performance across various input sizes using input-specific training, whereas the second tested the size-agnostic capability of models pretrained on 4096-byte inputs.

In the first experiment (Table 4), ByteSCAN achieved state-of-the-art performance across most input sizes in both datasets except for the 512-byte input. For 512-byte inputs, ByteSCAN demonstrated a highly competitive performance, achieving second and third place in each dataset with only marginal differences compared with the top-performing model.

In addition to its high accuracy, ByteSCAN demonstrated practical advantages in terms of training efficiency (Figure 2). Compared with the second-best performing model, XMP (82.9%), the training of ByteSCAN is considerably faster (approximately 75 vs. 125 h) and achieves a higher accuracy (83.3%). More importantly, unlike models such as JSANet, ByteResNet, and ByteFormer that require layer-specific modifications and complete retraining for each new input size, ByteSCAN maintains a consistent hyperparameters across all input dimensions, and can handle multiple sizes with a single trained model (Table 5). This eliminates the need for size-specific retraining, substantially reducing the practical training costs in real-world deployments where varying fragment sizes are common. This architectural advantage results in a reliable generalization across the entire range of evaluated input sizes.

Table 5 shows the performance of the model pretrained on the 4096-byte setting when tested across various input sizes, with JSANet, ByteResNet, and ByteFormer excluded because they cannot perform inferences on input sizes other than 4096 bytes, owing to their fixed input size constraint. ByteSCAN demonstrated size-agnostic capability across all input sizes by achieving an accuracy of 64.9% for 512-byte inputs, considerably outperforming other

TABLE 4 Comparison of rank-1 file classification accuracy (%) on the FFT-75 and GovDocs datasets.

Method	Parameters (M)	Architecture	FFT-75 (S#1)				GovDocs			
			512	1024	2048	4096	512	1024	2048	4096
FiFTy [7]	0.44	CNN	65.2	71.3	76.8	77.4	84.6	86.8	88.5	89.6
DSCNN [8]	0.09	CNN	64.6	69.8	74.4	78.1	84.8	87.4	89.1	90.4
ByteRCNN [11]	4.95	CNN + RNN	68.7	72.4	75.7	82.4	85.3	<u>88.0</u>	89.8	<u>91.5</u>
XMP [13]	77.23	Transformer	70.9	73.4	78.3	<u>82.9</u>	<u>86.0</u>	<u>88.0</u>	<u>90.3</u>	90.5
JSANet [14]	0.97	Transformer	69.2	71.3	77.5	82.6	86.1	87.9	89.2	90.7
ByteResNet [12]	11.78	CNN	68.8	73.3	77.7	82.1	83.3	85.8	87.4	88.4
ByteFormer [12]	31.25	Poolformer	72.5	<u>76.2</u>	<u>79.1</u>	81.9	85.1	86.5	87.5	88.4
ResNet1D [17] (backbone)	34.07	CNN	67.7	73.4	78.1	82.3	85.1	87.3	89.2	90.5
ByteSCAN (ours)	34.11	CNN + self-attention	<u>72.0</u>	76.3	80.1	83.3	85.3	90.2	91.5	92.7

Note: For the FFT-75, the 512-byte data were also cropped from 4096-byte files to maintain consistency with the 1024- and 2048-byte data. Bold and underlined values indicate the best and second-best accuracy in each column, respectively.

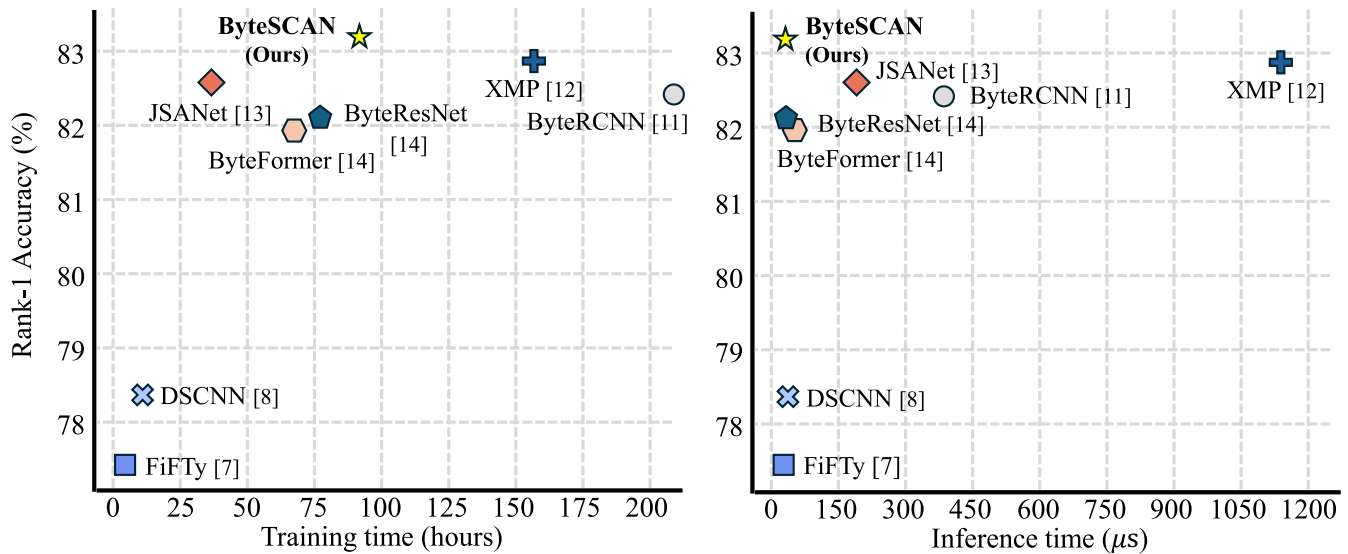


FIGURE 2 Training and inference times against file classification accuracy across baseline and state-of-the-art algorithms in file fragment classification on the 4096-byte setting in the FFT-75 [7] Scenario #1. Note that FiFTy and XMP require additional training time for hyperparameter tuning that was not accounted (please refer to the descriptions listed above). Conversely, the training time for the byte-segment convolution and attention network (ByteSCAN) includes backbone pretraining.

TABLE 5 Generalization performance of models trained on 4096-byte fragments.

Method	FFT-75 (S#1)		
	512	1024	2048
FiFTy [7]	48.3 (−16.9)	60.3 (−11.0)	69.0 (−7.8)
DSCNN [8]	46.3 (−18.3)	59.5 (−10.3)	68.0 (−6.4)
ByteRCNN [11]	38.5 (−30.2)	57.3 (−15.1)	72.9 (−2.8)
XMP [13]	49.4 (−21.5)	64.7 (−8.7)	<u>76.3</u> (−2.0)
ResNet1D [17] (backbone)	<u>52.8</u> (−14.9)	<u>64.9</u> (−8.5)	73.0 (−5.1)
ByteSCAN (ours)	64.9 (−7.1)	73.3 (−3.0)	79.0 (−1.1)

Note: Each model was trained only on 4096-byte fragments and then evaluated on smaller, unseen fragment sizes (512, 1024, and 2048 bytes) without any fine-tuning. Negative values indicate the decrease in accuracy compared with models trained on data of the same size (see Table 4). JSANet [14], ByteResNet [12], and ByteFormer [12] are excluded due to fixed input size constraints. Bold and underlined values indicate the best and second-best generalization performance for each fragment size, respectively.

models and approaching the accuracy achieved by FiFTy when specifically trained on 512-byte inputs (Table 4). The performance gap between ByteSCAN and the other models increased as the input size decreased, indicating its robustness to input size variations.

To evaluate the classification capabilities of the ByteSCAN model, we grouped the 75 labels from FFT-75 into broader semantic categories according to [7], as listed in Table 3. Each category contained multiple file extensions that shared similar structural and semantic characteristics.

Table 6 lists the classification accuracies for these categories. ByteSCAN achieved the highest accuracy in five out of eleven categories—Archive, Audio, Raw, Vector, and Miscellaneous—and maintained a highly competitive performance across all others. The strong performance in the Archive and Raw categories was

particularly noteworthy. Archive files (for example, zip and 7z) often contain complex compressed data structures, whereas raw image files (for example, cr2 and nef) are characterized by low-level sensor data patterns. ByteSCAN's success in these areas suggests that our CNN-based feature extraction from fixed-size segments is exceptionally effective in identifying intricate, low-level byte patterns that other models may overlook.

3.4 | Category-wise performance analysis

To evaluate the classification capabilities of the ByteSCAN model, we grouped the 75 labels from FFT-75 into broader semantic categories according to [7], as listed in

TABLE 6 Accuracy (%) across categories on FFT-75 (4096 bytes).

Cat.	FiTy	DSCNN	ByteRCNN	XMP	JSANet	ByteResNet	ByteFormer	ByteSCAN
Arc	66.7	69.5	<u>82.3</u>	77.7	78.2	78.9	78.3	82.5
Aud	97.7	97.2	98.5	98.6	98.4	<u>98.7</u>	<u>98.7</u>	99.0
Bmp	83.8	80.2	83.1	85.4	86.2	86.0	<u>86.1</u>	84.8
Exe	73.9	74.1	75.6	77.2	<u>76.9</u>	76.4	76.2	76.3
Hum	99.1	99.2	99.4	99.6	99.4	99.4	99.3	<u>99.5</u>
Off	81.8	81.1	90.4	93.4	91.4	90.0	89.0	<u>92.0</u>
Pub	58.0	65.7	71.9	73.8	<u>73.4</u>	70.3	68.7	72.2
Raw	97.9	98.4	<u>99.2</u>	<u>99.2</u>	<u>99.2</u>	99.1	<u>99.2</u>	99.6
Vec	83.3	87.9	<u>94.9</u>	<u>94.9</u>	94.4	92.3	92.5	95.6
Vid	85.5	84.9	81.2	<u>82.3</u>	79.5	79.2	80.2	82.1
Mis	97.3	98.3	<u>98.9</u>	99.0	98.8	<u>98.9</u>	99.0	99.0

Note: Bold and underlined values indicate the best and second-best accuracy on the FFT-75 dataset for each category, respectively.

Table 3. Each category contained multiple file extensions that shared similar structural and semantic characteristics.

Table 6 lists the classification accuracies for these categories. ByteSCAN achieved the highest accuracy in five out of eleven categories—Archive, Audio, Raw, Vector, and Miscellaneous—and maintained a highly competitive performance across all the others. The strong performance in the Archive and Raw categories was particularly noteworthy. Archive files (for example, zip and 7z) often contain complex compressed data structures, whereas raw image files (for example, cr2 and nef) are characterized by low-level sensor data patterns. ByteSCAN's success in these areas suggests that our CNN-based feature extraction from fixed-size segments is exceptionally effective in identifying intricate, low-level byte patterns that other models may overlook.

Among all the categories, Published and Executable methods are the most challenging, which is a well-known difficulty in byte-level file classification. These categories predominantly contain compressed or encoded files with high information entropy [13], as evidenced by the performance range across all the baseline methods (Published: 58.0% to 73.8%, Executable: 73.9% to 77.2%). Despite these inherently challenging cases, ByteSCAN maintained a competitive performance while achieving the highest overall accuracy (83.3%).

3.5 | Ablation study

To validate the effectiveness of the design choices, we conducted an ablation study on the FFT-75 dataset. Table 7 lists the results.

TABLE 7 Ablation study results on FFT-75 focusing on the different components of ByteSCAN.

Tested methods	1024	2048	4096
ByteSCAN (ours)	76.3	80.1	83.3
Without (w/o) Attention (soft voting)	76.1	79.2	81.5
w/o Random cropping (fixed)	72.1	76.5	81.0
w/o Overlap (512-byte segments)	76.4	79.9	83.1
w/o Overlap (256-byte segments)	74.8	78.5	81.9

Note: Bold values indicate the best performance for each input fragment size among the ablation variants.

3.5.1 | Attention versus ensemble

We first evaluated the contribution of the self-attention layer by replacing it with a simple soft voting ensemble of the predictions from each segment. The full ByteSCAN model outperformed the soft voting variant by a substantial margin (equal to 1.8%) for 4096-byte fragments (83.3% vs. 81.5%). This result underscores the fact that merely aggregating independent predictions is insufficient. The performance improvement from the self-attention layer demonstrates that its ability to model the global context and intersegment relationships is crucial for understanding the overall structure of the file.

3.5.2 | Random versus fixed cropping

The impact of our pretraining strategy was assessed by comparing against a model pretrained with fixed, non-random cropping. The use of random cropping yielded a notable performance increase of 2.3% for 4096-byte fragments (83.3% vs. 81.0%). This improvement stems from

TABLE 8 Comparison of generalization capability.

Test size (bytes)	Training data size		
	Specific	4096 bytes	Mixed
512	72.0	64.9	71.4
1024	76.3	73.3	75.9
2048	80.1	79.0	79.6
4096	83.3	83.3	82.4

Note: “Mixed” and “Specific” denote models trained on mixed sizes and the target size only, respectively.

the random cropping’s role as a powerful data augmentation technique; exposing the model to more diverse data segments prevents overfitting to byte patterns at specific locations and encourages the learning of more generalizable features.

3.5.3 | Overlapping versus non-overlapping segments

We analyzed the impact of segment configuration on the feature extraction performance, focusing on the effects of segment overlapping and the influence of segment size versus number.

To analyze the effects of overlapping, the 512-byte segmentation method with 256-byte overlap (512 overlapping) was compared with the 512-byte nonoverlapping segmentation method (512 nonoverlapping). While the overlapping method performed better, the nonoverlapping method showed a marginal 0.1% improvement with an input of 1024, likely owing to excessive overlap for this smaller data size.

To investigate the influence of the segment size and number, the 512 overlapping method (approximately 15 segments) was contrasted with the 256 nonoverlapping method (16 segments). The larger sized 512 overlapping method showed higher performance across all input sizes, indicating that the segment size impacts performance more than the segment number.

3.5.4 | Impact of mixed-size training

Finally, we investigated whether explicitly training on variable fragment sizes could further enhance the model’s adaptability beyond the inherent capabilities of fixed slicing. We compared our standard model (trained solely on 4096-byte fragments) with a unified model trained on a mixed dataset comprising $L \in \{512, 1024, 2048, 4096\}$ bytes. As summarized in Table 8, the mixed-size training strategy yields performance gains for smaller fragments without requiring

architectural changes. Most notably, the accuracy of 512-byte inputs improved by 6.5% (from 64.9% to 71.4%), reaching a level comparable with that of a size-specific model trained exclusively on 512-byte data (72.0%). Despite a minor drop (0.9%) at 4096 bytes, this strategy fostered robust feature learning, effectively reducing the performance gap across different sizes.

4 | CONCLUSION

We addressed the limitations of recent FFC models, such as their sensitivity to hyperparameters and lack of robustness to varying input sizes. To overcome these issues, we proposed ByteSCAN, a lightweight and efficient model that combines fixed slicing, a moderately deep CNN, and a self-attention layer. ByteSCAN achieves robust performance across various file sizes without requiring extensive hyperparameter tuning while maintaining competitive training and inference times.

Although ByteSCAN demonstrates strong performance, several directions remain for future research. First, exploring model optimization techniques can reduce the parameter count while maintaining performance, creating a more lightweight version. Second, investigating adaptive slicing mechanisms could potentially offer additional performance gains beyond our fixed-slicing approach. Finally, to validate the generality of the model more thoroughly, we plan to extend our evaluation to other scenarios of the FFT-75 dataset and other public datasets in future studies. Through this study, we hope to elucidate the development of more advanced and practical model architectures for classifying binary data.

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

ORCID

Hyeongsik Kim  <https://orcid.org/0009-0001-7040-0152>

Sisung Liu  <https://orcid.org/0009-0003-3306-9047>

Je Hyeong Hong  <https://orcid.org/0000-0003-2797-553X>

REFERENCES

1. X. Lin, *File carving, Introductory computer forensics: a hands-on practical approach*, Springer, Cham, 2018, pp. 211–233.
2. S. J. Stolfo, K. Wang, and W.-J. Li, *Fileprint analysis for malware detection*, Association for Computing Machinery (ACM), 2005.
3. 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 Secur.* **8** (2013), no. 9, 1519–1530.
4. S. Fitzgerald, G. Mathews, C. Morris, and O. Zhulyn, *Using NLP techniques for file fragment classification*, *Dig. Investig.* **9** (2012), S44–S49.

5. Q. Li, A. Ong, P. Suganthan, and V. Thing, *A novel support vector machine approach to high entropy data fragment classification*, (South African Information Security Multi-Conference, Port Elizabeth, South Africa), 2011, pp. 236–247.
6. N. Zheng, J. Wang, T. Wu, and M. Xu, *A fragment classification method depending on data type*, (IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Communi.; Dependable Auton. Secure Comput.; Pervasive Intell. Comput., Liverpool, UK), IEEE, 2015, pp. 1948–1953.
7. G. Mittal, P. Korus, and N. Memon, *FiFTy: large-scale file fragment type identification using convolutional neural networks*, IEEE Trans. Inf. Forensics Secur. **16** (2020), 28–41.
8. K. M. Saaim, M. Felemban, S. Alsaleh, and A. Almulhem, *Light-weight file fragments classification using depthwise separable convolutions*, (IFIP Int. Conf. ICT Syst. Secur. Privacy Prot., Copenhagen, Denmark), Springer, Cham, 2022, pp. 196–211.
9. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, (Proc. IEEE Conf. Comput. Vision Pattern Recognit., Boston, MA, USA), IEEE, 2015, pp. 1–9.
10. 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*, (IEEE Int. Conf. Artif. Intell. Circuits Syst., Hangzhou, China), 2023.
11. K. Skračić, J. Petrović, and P. Pale, *ByteRCNN: enhancing file fragment type identification with recurrent and convolutional neural networks*, IEEE Access **11** (2023), 138176–138187.
12. W. Liu, K. Wu, T. Liu, Y. Wang, K.-H. Yap, and L.-P. Chau, *ByteNet: rethinking multimedia file fragment classification through visual perspectives*, IEEE Trans. Multimedia **27** (2024), 1305–1319.
13. J. G. Park, S. Liu, and J. H. Hong, *XMP: a cross-attention multi-scale performer for file fragment classification*, (ICASSP 2024—2024 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), Seoul, Rep. of Korea), 2024, pp. 4505–4509.
14. Y. Wang, W. Liu, K. Wu, K.-H. Yap, and L.-P. Chau, *Intra- and inter-sector contextual information fusion with joint self-attention for file fragment classification*, Knowled.-Based Syst. **291** (2024), 111565.
15. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Mindere, G. Heigole, S. Gelly, J. Uszkoreit, and N. Houlsby, *An image is worth 16x16 words: transformers for image recognition at scale*, (Int. Conf. Learn. Representations), 2021.
16. N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-end object detection with transformers*, (Eur. Conf. Comput. Vision, Glasgow, United Kingdom), Springer, Cham, 2020, pp. 213–229.
17. K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, (Proc. IEEE Conf. Comput. Vision Pattern Recognit., Las Vegas, NV, USA), 2016, pp. 770–778.
18. S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, *Bringing science to digital forensics with standardized forensic corpora*, Dig. Invest. **6** (2009), S2–S11.

AUTHOR BIOGRAPHIES



Hyeongsik Kim received a B.Sc. degree in Electronics and Electrical Engineering from Dankook University, Yongin, Republic of Korea, in 2024. Currently, he is pursuing an M.S. degree in the Department of Artificial Intelligence Semiconductor Engineering at Hanyang University.



Sisung Liu received a B.Sc. degrees in Applied Physics from Hanyang University in 2023. He is currently pursuing a Ph.D. in the Department of Artificial Intelligence at Hanyang University, Seoul, Republic of Korea. His research interests include information forensics and neural geometry estimations.



Je Hyeong Hong received B.A. and M.Eng. degrees in Engineering (Electrical and Information Sciences) from the University of Cambridge, UK, in 2011, and a Ph.D. degree in Engineering (Computer Vision) from the University of Cambridge, UK, in 2018, after collaborating with Microsoft and Toshiba Research Europe. He has served as an alternative military service in Republic of Korea as a postdoctoral researcher at the Korea Institute of Science and Technology (KIST) from 2018 to 2021. Currently, Je Hyeong is an assistant professor in the Department of Electronic Engineering at Hanyang University in Republic of Korea, a position he has held since 2021. He has published several papers at prestigious conferences such as CVPR, ECCV, ICCV, and ICASSP. His main research interests include computer vision, machine learning, and optimization.

How to cite this article: H. Kim, S. Liu, and J. Hyeong Hong, *Size-agnostic file fragment classification via fixed byte slicing*, ETRI Journal **48** (2026), 747–755, DOI [10.4218/etrij.2025-0314](https://doi.org/10.4218/etrij.2025-0314)