IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

File Fragment Type Classification using Light-Weight Convolutional Neural Networks

MUHAMAD FELEMBAN [1,2] (Member, IEEE), MUSTAFA GHALEB[1], KUNWAR SAAIM [3], SALEH AL-SALEH[1], AHMAD ALMULHEM[1,2]

[1]Computer Engineering Department, KFUPM, Dhahran, Saudi Arabia (e-mail: {mustafa.ghaleb,mfelemban,ahmadsm,g202212920, salehs}@kfupm.edu.sa)
[2]Interdisciplinary Research Center for Intelligent Secure Systems, KFUPM, Dhahran, Saudi Arabia
[3]Department of Computing Science, University of Alberta, Canada (e-mail: ksaaim@ualberta.ca)

Corresponding author: Muhamad Felemban (e-mail: mfelemban@kfupm.edu.sa).

The authors would like to acknowledge the Interdisciplinary Research Center for Intelligent Secure Systems at KFUPM.

ABSTRACT In digital forensics, file carving is used to extract files without relying on the underlying file system metadata. This process can be challenging if the file is fragmented. Therefore, it is important first to identify the type of file fragment. There exist several techniques to identify the type of file fragments without relying on metadata, for example, using headers and footers to identify the fragment type. Recently, convolutional neural network (CNN) models have been used to build classification models to achieve this task. Existing models for file fragment type classification often require significant computational resources due to their large number of parameters, leading to slower inference times and higher memory consumption. To address these challenges, we propose light-weight file fragment type classification models based on separable CNNs that maintain comparable accuracy while reducing computational demands. Our proposed light-weight file fragment type classification model leverages depthwise separable convolutions to improve the efficiency of feature extraction while reducing computational overhead. This approach leads to improved classification performance by focusing on the most relevant features within file fragments, achieving comparable accuracy to state-of-the-art models with significantly fewer parameters. The evaluation results demonstrate the model's effectiveness, with a 79% accuracy on the FFT-75 dataset using nearly 100K parameters and 164M FLOPs —representing a 4x reduction in model size and a 6x improvement in speed over the best-performing existing classifier. Our results demonstrate that these light-weight models are effective for real-time digital forensic applications where computational efficiency is critical.

INDEX TERMS Digital Forensics, File carving, File fragment type classification, Depthwise-Separable Convolution

I. INTRODUCTION

DIGITAL forensics is a field of forensic science that deals with the extraction, analysis, and investigation of digital artifacts extracted from the varieties of digital devices used nowadays. The objective is to collect, preserve, and analyze digital evidence that can be used in legal proceedings. Digital forensics has become increasingly important with the increased reliance on digital devices and systems. Several digital forensic tools have been proposed to facilitate this process [1]. Digital Forensics and Incident Response (DFIR) teams use these tools to investigate, analyze, and report cyberattack incidents in enterprises.

It is common practice for attackers to perform a disk erasure to remove any evidence that incriminates them after successful attacks [2]. Therefore, the first step in DFIR is to recover wiped files. However, recovery methods that use file system metadata are deemed ineffective in such scenarios. To overcome this challenge, digital investigators use file carving to reconstruct files based on their content [3], [4]. The aim of file carving process is to recover erased and damaged data files from blocks of raw binary data without using metadata. There are several file carving techniques for fully or partially reconstructing files using a variety of techniques, including header/footer matching [5], probabilistic measures [6], and n-gram analysis [7].

With the absence of metadata from the file system, it is

challenging to infer the type of carved file. This is even more challenging when file carving is performed on fragmented files. As a result, two tasks need to be resolved in file fragment type classification: selecting a candidate sequence of file blocks and classifying the fragment type [4]. The latter task is called file fragment type classification. There exist several approaches for classifying file fragment types in the literature [8]–[10]. For example, *Sceadan* tool classifies file fragments according to the content [8]. Recently, deep learning has been used to identify the type of file fragment. Gray-scale [9] and FiFTy [10] tools illustrate how Convolution Neural Networks (CNNs) can be efficiently used for file fragment type classification.

A major drawback of CNNs is the exponential increase in the number of parameters as the depth of the model increases [11]. While increasing the depth provides better accuracy, such an increase can lead to increased training and inference time. Therefore, it is important to find a balance between inference time and accuracy by optimizing the CNN structure. To overcome this challenge, researchers proposed a variation of CNN that reduces the number of parameters called Depthwise Separable Convolution (DSC) [12]–[14]. In our previous paper [15], we addressed this gap by proposing a classification of file fragment types based on DSC. Other recent advancements in CNN architectures have led to the integration of Squeeze-and-Excitation (SE) blocks with CNN [16]. The objective of SE blocks is to improve the representational power of a network by explicitly modeling the interdependencies between the channels of its convolutional features.

In this paper, we propose two new models for file fragment type classification: Depthwise Separable Convolutional with Squeeze-and-Excitation (DSC-SE) and Modified Depthwise Separable Convolutional (M-DSC) based on the model proposed in [15]. By embedding SE blocks in a depthwise separable convolutional network, the DSC-SE model can selectively emphasize informative features while reducing the overall parameter count, making it both efficient and effective for file fragment type classification tasks. The M-DSC model is a modification of the original DSC network. While the standard DSC is effective in reducing computational complexity by decomposing the convolution operation into depthwise and pointwise convolutions, the model struggles to capture complex spatial patterns. The M-DSC addresses these limitations by introducing specific architectural changes, including altering activation functions and incorporating group normalization to improve the network's ability to capture fine-grained details in file fragment type classification. The accurate classification of file fragments is particularly critical in digital forensics, where reconstruction of files from fragmented digital evidence can be vital for data recovery and cybercrime investigations. In this context, forensic analysts often rely on the ability to piece together incomplete data, making the efficiency and accuracy of classification models paramount. The application of advanced CNN architectures, such as DSC-SE and M-DSC, directly contributes to this need by improving both the precision and speed of classification tasks, thus providing valuable tools for security and forensic professionals.

We show that the proposed models reduce the number of parameters compared to CNN. Therefore, the process of file fragment type classification can be carried out without the need of specialized hardware, e.g., GPUs and TPUs. To evaluate the performance of our models, we compared the results of the proposed models with FiFTy [10] and a baseline Recurrent Neural Networks (RNN) [17]. We have trained the models using FFT-75 dataset [10], which includes six scenarios. The results of Scenario 1 show that our model achieves an accuracy that is comparable to FiFTy using 6.3x less floating-point operations (FLOPs) on 4096 bytes fragment and 87x less FLOPs on 512 bytes fragments. Furthermore, our models achieve 79.27% accuracy with around 100K parameters and 164M FLOPs, while FiFTy achieves 77.04% accuracy with more than 400K parameters and 1 GFLOPs. Finally, we show that our models outperform FiFty in terms of inference time (milliseconds per block) by being 15x faster on GPU than FiFTy and 105x faster than baseline RNN.

The remainder of this paper is organized as follows. In Section 2, we provide the necessary background for this work. In Section 3, we present our proposed models. In Section 4, we discuss the experiments and the results of the evaluation. In Section 5, we discuss the state-of-the-art of file fragment type classification. Finally, we draw conclusions from our findings and suggest potential directions for future research in Section 6.

II. RELATED WORK

Several techniques have been proposed for identifying and classifying file fragment types. The techniques range from basic methods using magic numbers and file headers [18] to more advanced methods using machine learning and deep learning [8], [10]. Typically, file carving methods can be broadly categorized into three categories: 1) statistical methods, 2) machine learning methods, and 3) deep learning methods.

Statistical methods. In [19], the authors introduced *Oscar* to determine the probable file type of binary data fragment. *Oscar* develops vector models based on the Byte Frequency Distribution (BFD) of 4096 byte fragments of different file types. The vector distance between the mean and standard deviation of the segment to be classified and the centroids is computed. The segment is classified as a modeling file if the distance falls below a threshold value. Another notable open source tool for file carving is *Sceadan* that exploits a wide range of statistical features such as unigrams, bigrams, entropy, mean byte value, and longest streak, among others [8]. Ten distinct input vectors are produced from these statistical features, subsequently divided into four sets: unigrams, bi-grams, all global features aside from n-grams, and a subset of global features. These sets are used as inputs to the classification models. In [20], the authors proposed a method to identify the type of file based on its content. To reduce

the computation time for identification, two techniques are employed. First, a subset of features is chosen based on the frequency of occurrence. Second, file blocks of 100 byte size are sampled. It was observed that this classifier performed well on low-entropy file fragments (like plain-text files, uncompressed images, etc.) but failed on high-entropy file fragments (like compressed files, binary executables, etc.).

Machine learning methods. In [21], the authors proposed a Support Vector Machine (SVM) model that is trained using a feature vector based on the databyte histogram. SVM is used for binary classification, where one class is fixed as JPEG and the other class varies, including DLL, EXE, PDF, and MP3. Similarly, Fitzgerald et al. [22] introduced an SVM model for fragment classification, in which file fragments are treated as a bag of bytes represented by a feature vector encompassing unigram and bigram counts, along with statistical measures such as entropy. In [23], an SVM-based approach was proposed that used BFD or histogram and entropy as feature vectors. In [24], the authors proposed a hierarchical combination of Principal Component Analysis (PCA) and Multi-layer Perceptron (MLP) for feature extraction. These extracted features were supplied to the classifier. PCA selected N_1 number of features from BFD of the raw features. These N_1 characteristics were used to train an auto-associative neural network that extracts N_2 features ($N_2 < N_1$). In [25], the authors proposed a hierarchical classification approach for file fragment type classification utilizing SVM as base classifiers. The study found that the features used were not sufficient for complex file types, as they failed to characterize the membership relationship between the file fragment and its parent class. The authors recommend refining the hierarchy branch for complex files and engineering features that better depict the membership relationship of a file fragment with its parent file type. In [26], the authors proposed *byte2vec*, a novel feature generation model that extends the *word2vec* concept to map file fragments to dense vector representations. *byte2vec* generates vector representations utilizing the Skip-gram model, and the k-Nearest Neighbors (kNN) classifier is trained on these representations to identify fragment types. *byte2vec* corpus model works for various block sizes and file types. Using a publicly available dataset, the authors combine *byte2vec* with the kNN algorithm (*byte2vec*+kNN) for feature extraction and classification.

Deep learning methods. CNNs have been widely adopted in the field of file fragment type classification due to their ability to automatically learn hierarchical feature representations from raw data. In [27], the authors proposed a one-layer convolution with multiple kernel sizes. The input to the model is a binary representation of the raw bytes, which is fed into an embedding layer to convert the binary representation to a dense vector. In [9], the authors converted the 4096-byte file fragment to a 64x63 *gray-scale* image. The authors argue that the data fragments from different files would result in to different texture features in the gray-scale. The gray-scale images are then fed to a deep CNN network with several convolutions and max-pool layers followed by dense classification. In [28], a comparison between feed-forward, recurrent, and CNNs with the binary representation of file fragments was conducted using several file types including JPEG, GIF, XML, and CSV. The results show that recurrent networks provided better results compared to other models. In [10], the authors proposed a CNN-based file fragment type identification, called *FiFTy*. The authors created a dataset specifically for file fragment type classification. Moreover, a compact neural network was developed that used trainable embedding space and convolutional neural networks. In [29], authors proposed a techique to convert raw bytes to images, i.e., *byte2image*, and apply CNN to identify file fragment types using the converted images. However, these traditional CNN models tend to be computationally intensive, requiring substantial hardware resources for both training and inference. This makes them less suitable for real-time applications or environments with limited computational power. To address the computational inefficiency of standard CNNs, researchers have proposed depthwise separable convolutions for file fragment type classification, which significantly reduce the number of parameters and computational cost by decomposing standard convolutions into depthwise and pointwise operations [15]. While depthwise separable convolutions have proven effective in reducing computational overhead, they come with their own set of limitations. Specifically, they may struggle to capture complex spatial patterns and long-range dependencies within the data, leading to potential loss of critical information during the classification process. The Squeeze-and-Excitation (SE) block, introduced by [16], offers a solution to the challenge of capturing complex feature interactions. The SE block works by recalibrating channel-wise feature responses, effectively emphasizing the most informative features while suppressing less useful ones. This technique has been integrated into various CNN architectures to enhance their representational capacity without a significant increase in computational complexity.

Recent advancements in file fragment type classification have seen the introduction of Transformer-based models, such as XMP [30], which leverage multi-scale self and cross-attentions between CNN features extracted from byte n-grams of binary data. This approach has demonstrated state-of-the-art accuracy in various scenarios, showcasing the effectiveness of Transformer architectures for this task. Another recent approach in file fragment type classification is the joint self-attention network (JSANet) [31], which integrates intra- and inter-sector contextual information to enhance feature representation. This model significantly improves classification accuracy, particularly on datasets that reflect real-world file fragmentation. More recently, Skračić et al. proposed ByteRCNN that combines recurrent and convolutional layers to enhance file fragment type identification [32]. This hybrid model has shown competitive performance, particularly in scenarios involving varying fragment sizes. However, these techniques often require significant com-

putational resources, making them less practical for real-time applications or environments with limited resources. In addition, their increased complexity can lead to higher inference times, which may not be suitable for all use cases.

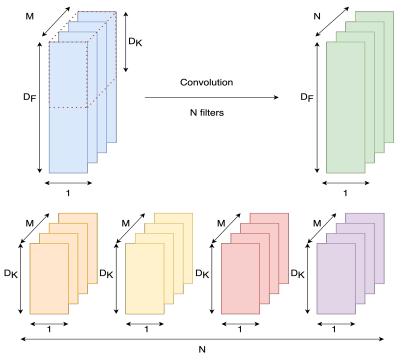
Our proposed models, DSC-SE and M-DSC, are designed to address the specific limitations identified in the related literature. The DSC-SE model integrates SE blocks into a depthwise separable convolutional architecture, enhancing its ability to recalibrate features dynamically while maintaining a light-weight structure. This combination allows the model to balance computational efficiency with the need for complex feature representation. On the other hand, M-DSC introduces further modifications, such as the use of ReLU activation and Group Normalization, to improve the model's generalization capabilities and its ability to handle diverse datasets effectively. These innovations make our models particularly well suited for the challenges of file fragment type classification in digital forensics, where both accuracy and efficiency are paramount. Compared to the existing neural network-based classifier [9], [10], [27], our model achieves better accuracy on the largest number of file types, as discussed in the experiment section.

III. PRELIMINARIES

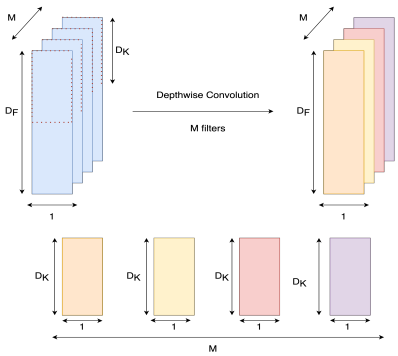
Typically, CNN contains several convolution layers that can extract features from the inputs and predict an output based on the extracted features [33]. CNN is widely used for object detection and image classification. In general, kernel filters are designed to be smaller than the input to allow for the sharing of kernel weights across input dimensions. In file fragment type classification, the input to the CNN is a vector of 4096 or 512 dimensions, equivalent to fragments of 4KB or 512 bytes, respectively. The input is then transmitted to an embedding layer that converts the input vector into a continuous (4096,32) or (512,32) dimensional vector, depending on the fragment size. The output of the embedding layer is fed into a convolution layer with kernels of any size. Several convolution layers can be stacked [10].

The separable convolution model was first introduced by Sifre and Mallat [34] to reduce the number of CNN parameters without severely affecting the accuracy of the model. Since then, several models based on separable convolution have been proposed, including Xception-net [35], Mobilenets [12], Efficient [36], and Shufflenet [37]. The convolution kernel in depthwise separable convolution is split into two parts: a depthwise convolution and a 1×1 pointwise convolution. Such splitting of convolution kernel reduces the number of parameters and hence the computation time required for training and inference. Figure 1 illustrates the difference in the convolution kernel between standard, depthwise, and pointwise convolution.

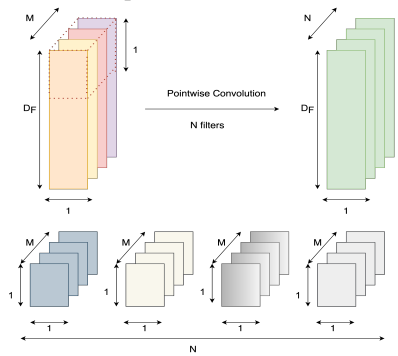
In particular, the standard one-dimensional convolution operation, shown in Figure 1a, takes the $D_{\mathrm{F}} \times 1 \times M$ feature maps as input and produces the $D_{\mathrm{F}} \times 1 \times N$ feature maps, where D_F is the length of the input feature, M is the number of input channels, and N is the number of output channels.



(a) Standard Convolution



(b) Depthwise Convolution



(c) Pointwise Convolution

FIGURE 1: Standard convolution in (a) is factorized into two layers: Depthwise Convolution in (b) and pointwise in (c).

In a convolutional network, the number of parameters with kernel size K is $D_{\mathrm{K}} \times 1 \times N \times M$. Subsequently, the total computation cost is

$$D_{\mathrm{F}} \cdot N \cdot M \cdot D_{\mathrm{K}} \tag{1}$$

On the other hand, depthwise separable convolution factorizes the convolution layer into two layers: depthwise convolution, depicted in Figure 1b, and pointwise convolution, depicted in Figure 1c. The depthwise convolution layer with one filter per input channel has $D_{\mathrm{K}} \times 1 \times M$ parameters with K sized kernel and M number of channels. Therefore, depthwise convolution has a computation cost of:

$$D_{\mathrm{F}} \cdot M \cdot D_{\mathrm{K}} \tag{2}$$

On the other hand, pointwise convolution uses a linear combination of filtered feature maps with the help of 1×1 convolution. The cost of 1×1 convolution is the same as the standard convolution having a kernel size of 1, i.e., $D_F \times N \times M$. Consequently, the total computation cost of the depthwise separable convolution is:

$$D_F \cdot M \cdot D_K + D_F \cdot N \cdot M \tag{3}$$

As a result, the total reduction in the computation cost when using depthwise separable convolution is:

$$\frac{D_F \cdot M \cdot D_K + D_F \cdot N \cdot M}{D_F \cdot N \cdot M \cdot D_K} \tag{4}$$

$$= \frac{1}{N} + \frac{1}{D_K} \tag{5}$$

The value of N ranges from 32 to 1024 and D_K for one-dimensional convolution can be between 9 and 27. Therefore, the reduction in computation cost is between 85% and 95%. In this regard, several models have been created using depthwise separable convolution for the purpose of image classification. Such models achieve high accuracy while still allowing for a rapid inference time [12], [35], [38].

IV. LIGHT-WEIGHT CNNS ARCHITECTURE

In this section, we describe the proposed architectures of the light-weight CNNs models for file fragment type classification. Previous file carving techniques are mainly based on statistical feature extraction [19], SVMs [8], CNN [10], RNN [28], and Fully Connected Neural Networks (FCNN) [39]. While RNNs are exponentially slow [40], FCNNs and CNNs require a deep model to achieve good performance. However, our objective is to reduce the inference time of file fragment type classification while maintaining high accuracy. In this regard, several models have been developed based on depthwise separable convolution for image classification tasks that achieved a fast inference time while maintaining high accuracy [12], [35], [38]. Compared to FCNNs, CNNs, and RNNs, depthwise separable convolution typically requires less computation time and fewer parameters [11], [12], [41]. Therefore, in this paper, we opt to use depthwise separable convolution to develop a light-weight CNN model for file fragment type classification. The methodology of this paper is illustrated in Figure 2. This diagram provides an overview of the various phases, including byte embeddings, application of light-weight convolutional models, and final classification, facilitating a better understanding of the proposed approach.

Figure 3 illustrates the overall architectures of the proposed models. All models consist of an embedding layer and a standard 1D convolution block. The embedding layer transforms a file fragment byte ranging from 0 to 255 into a dense continuous vector of 32 dimensions. The embedding layers are used to compress the input feature dimension into a smaller space. Note that without the embedding layer, each byte value would be represented by a 256-dimensional sparse vector. With multiple layers of standard and depthwise separable convolutions, features are automatically derived by

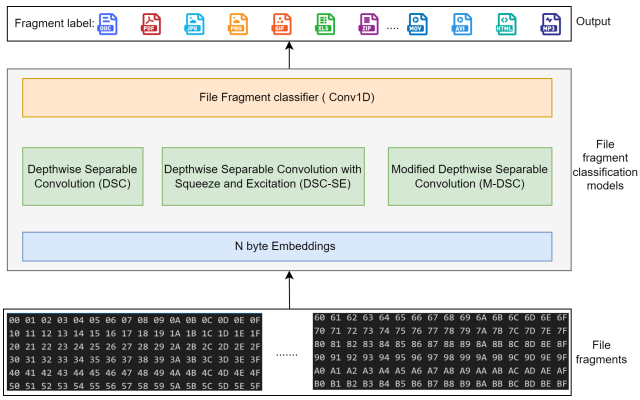


FIGURE 2: General Flow of light-weight CNN-Based File Fragment Classification

non-linearly transforming inputs. In order to extract essential features automatically for classification, the dense vector representation is then passed to a standard 1D convolution block and multiple depthwise separable inception blocks or squeeze-and-excitation blocks. To obtain the final features for classification, the output of the last depthwise separable convolution block is averaged along the spatial dimension. In the following sections, we provide more details about each model.

A. DEPTHWISE SEPARABLE CONVOLUTION (DSC)

The DSC model begins with an embedding layer that transforms the input file fragment into a continuous 32-dimensional vector. This vector is then passed through a 1D convolution layer with a filter size of 19. The output of the 1D convolution layer is passed to a Hardswish activation function, which adds nonlinearity to the model. The Hardswish function [42] is as follows.

$$Hardswish(x) = \begin{cases} 0 & if x \leq -3, \\ x & if x \geq +3, \\ \frac{x \cdot (x+3)}{6} & otherwise \end{cases} \tag{6}$$

The output is then processed by three inception blocks, which are composed of multiple parallel convolutional and pooling layers, designed to capture both local and global information from the input. After the inception blocks, the output is passed to an average pooling layer to reduce the spatial dimension, followed by a final 1D convolutional layer that performs feature map reduction and generates the final prediction. The class probabilities are determined from the feature vectors using 1×1 convolution followed by the *softmax* activation function [43].

Inception Block. Figure 4 depicts the architecture of a typical inception block. The previous layer's output is passed to three parallel depthwise separable convolutions and one 1×1 convolution. In our model, the kernel size of the depthwise separable convolution layer is 11, 19, and 27, respectively, with strides of 1, 1, 1, and 4, respectively. Kernel sizes

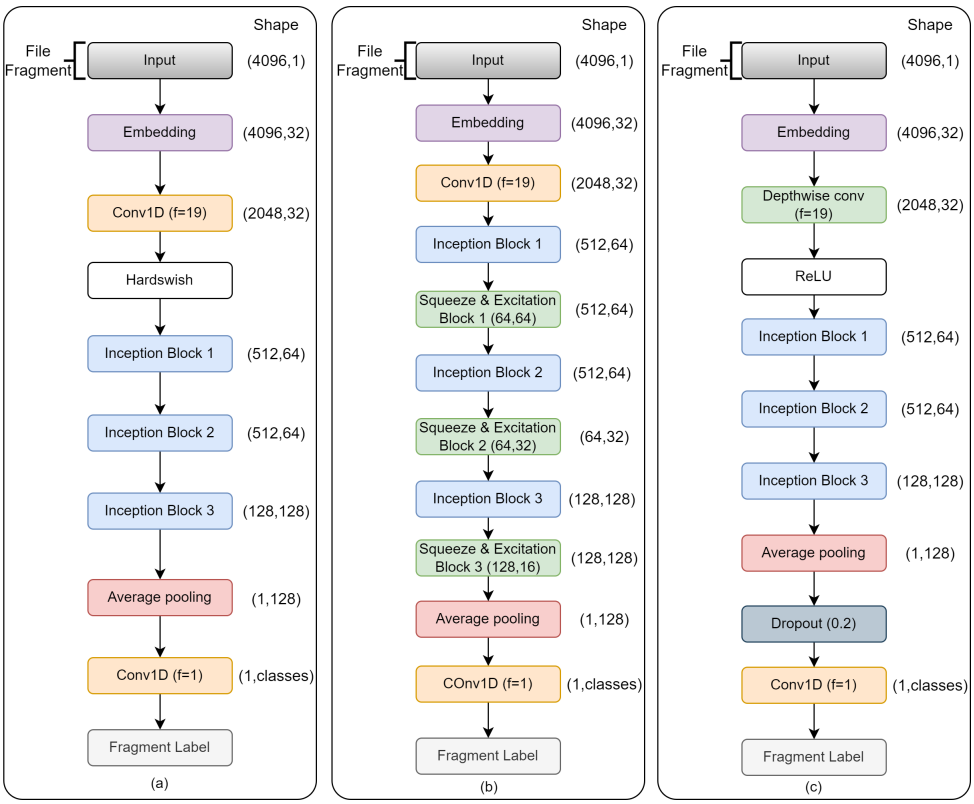


FIGURE 3: Network Architectures. a) DSC, b) DSC-SE, and c) M-DSC

were chosen empirically based on previous state-of-the-art network [10]. A batch normalization layer follows each of the convolution layer. The outputs of all the depthwise separable convolution layer are summed and then sub-sampled using a max-pooling layer with size 4. The output of the 1×1 convolution branch is directly added to the output of the max-pooling layer. If the number of the input and output channels is the same, then the 1×1 convolution layer is replaced with the direct addition of the output of max-pooling layer.

While the base DSC model can be considered successful in reducing computational complexity, it presents certain limitations when handling complex feature interactions. This is primarily due to the nature of depthwise separable convolutions, which fall short in tasks that require fine-grained feature discrimination. The DSC model's ability to recalibrate features dynamically is limited, as it lacks the mechanism to adjust the importance of different channels, which can lead to underperformance in more intricate classification tasks. In addition, while the use of Hardswish activation and Batch Normalization in DSC is effective in many cases, these components may not always deliver optimal results across diverse datasets, where the need for more flexible activation and normalization strategies becomes apparent. To address these challenges, we introduce the DSC-SE and M-DSC models.

B. DEPTHWISE SEPARABLE CONVOLUTION WITH SQUEEZE AND EXCITATION (DSC-SE)

DSC-SE is based on DSC with the addition of SE blocks [16] after each inception block. The rest of the architecture remains similar to the DSC model, with the same sequence of Inception Blocks and final classification layers. SE block is a type of attention mechanism to improve CNNs performance. SE block allows the network to focus on the most important features in each channel of the feature maps, which is useful for tasks such as image classification, segmentation, and object detection. SE block works by first squeezing the spatial information from the feature maps into a single channel-wise global descriptor and then using a fully connected layer to calculate a set of channel-wise weights. These weights are then element-wise multiplied with the feature maps. This, in turn, assigns higher weights to the channels that contain the most important information for the task. Recently, SE blocks have been integrated into the inception architecture, resulting in improved performance on various tasks [16]. The core principle behind SE blocks is to give greater importance to features that contribute more to the final prediction. Moreover, SE blocks are computationally efficient and easy to implement. Figure 5 shows the structure of the DSC-SE in detail.

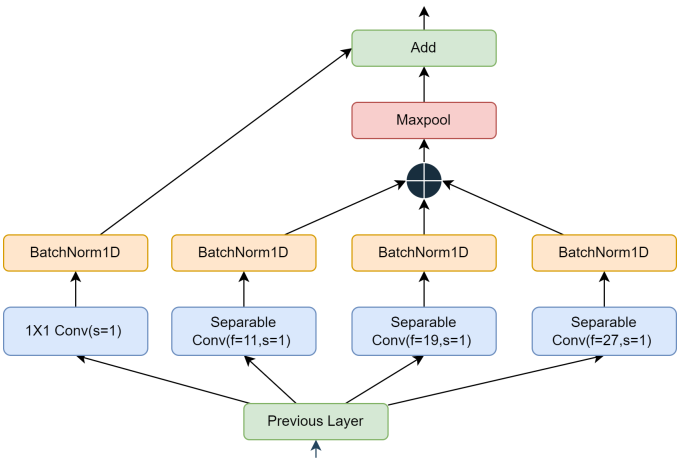


FIGURE 4: Inception Block

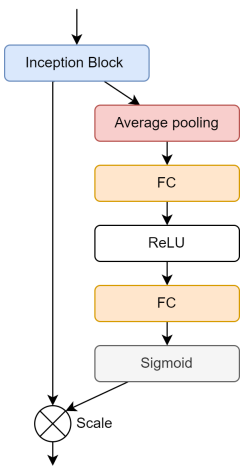


FIGURE 5: Squeeze-and-Excitation Block

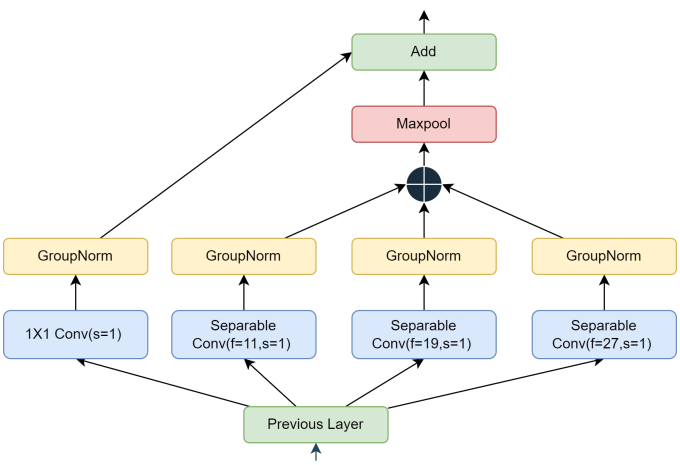


FIGURE 6: Modified Inception Block

C. MODIFIED DEPTHWISE SEPARABLE CONVOLUTION (M-DSC)

M-DSC architecture is also based on DSC but with several modifications. Instead of the first 1D convolutional layer, M-DSC uses a depthwise convolutional layer that performs convolutions independently on each input channel, which further reduces the number of parameters.. The Hardswish activation is replaced with ReLU to simplify the activation landscape. In addition, we used group normalization instead of batch normalization to normalize the activation of multiple channels at once [44], [45]. Group Normalization improves the model's generalization across varied data distributions, addressing the challenges of complex spatial patterns that DSC alone may struggle with. The modified inception block used in our proposed models is shown in Figure 6. This results in a reduction of the memory and computation overhead of batch normalization. Finally, after the average pooling and before the final 1D convolutional layer, a dropout layer is added to prevent overfitting by randomly setting a fraction of the activation to zero during training.

D. DISCUSSION

In summary, the three architectures differ in the way of handling activation functions, normalization, and feature recalibration, with the objective to improve the performance and efficiency of the model. DSC-SE adds SE blocks to the inception blocks to improve feature recalibration, while M-DSC replaces the 1D convolutional layer with a depthwise convolutional layer, replaces Hardswish with ReLU, and introduces group normalization and dropout to improve performance and efficiency. The key advantages of the proposed architectures compared to the state-of-the-art work are summarized as follows.

- **Parameter Efficiency**: Our models are designed to significantly reduce the number of parameters through the use of depthwise separable convolutions, making them more efficient compared to traditional CNNs.
- **Computational Optimization**: By focusing on lightweight architectures, the models are optimized for faster inference, suitable for environments with limited computational resources.
- **Improved Feature Re-calibration**: The DSC-SE model integrates Squeeze-and-Excitation (SE) blocks, which enhance the model's ability to focus on the most relevant features, improving overall classification accuracy without adding significant computational overhead.
- **Enhanced Accuracy with Fewer Resources**: Despite their reduced complexity, the proposed models maintain a high level of accuracy, achieving performance comparable to state-of-the-art models like FiFTy, but with a fraction of the parameters and FLOPs.
- **Adaptation to Fragment Sizes**: The proposed models are capable of handling different fragment sizes (e.g., 4096-byte and 512-byte fragments) effectively, making them versatile across various file fragment type classification tasks.
- **Robust Performance on Realistic Datasets**: The models have been evaluated using the FFT-75 dataset, demonstrating their robustness and practical applicability in realistic digital forensic scenarios.

TABLE 1: Grouping of different file types

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

These advantages not only differentiate our proposed models from traditional approaches but also provide a robust foundation for efficient file fragment type classification in real-world applications. In the following sections, we detail our experimental setup and present a comprehensive evaluation of the models, demonstrating how these architectural choices contribute to improved accuracy and reduced inference time.

V. PERFORMANCE EVALUATION

In this section, we present the results of evaluating the performance of our proposed model against baseline models based on RNN and FiFTy [10].

A. DATASET

To evaluate the performance of our models, we used the dataset provided by Mittal et al. in [10], which contains a balanced number of files per class. Other datasets, e.g., [46], are highly imbalanced with 20 files-types comprising 99.3% of the dataset and remaining 0.7% belonging to 43 file types. The dataset used is composed of 75 types of files (Table 1) that are organized into 6 different scenarios. The Scenarios are summarized in Table 2.

B. BASELINE MODEL

We implemented a variation of RNN, i.e., Long Short Term Memory (LSTM) based model, as a baseline model for performance comparison based on the work in [28]. It was observed that the low number of learnable parameters hindered their effectiveness in classifying 75 types of files. The model specifications are as follows. File fragment byte sequences are fed into a 32-dimensional embedding layer followed by bidirectional and unidirectional LSTM layers. The LSTM layer has 128 neurons in each of its two layers for 512-byte fragments and 128 and 256 neurons in its bidirectional and unidirectional layers for 4 KB fragments, respectively. A softmax dense layer was used to produce the output labels.

C. EXPERIMENTAL SETUP

All of our experiments were conducted on a dual Intel Xeon CPU E5-2620 CPUs @ 2.40 GHz (12 physical cores, 24 logical cores), 192 GB RAM, and a single Nvidia Titan X GPU, running on Ubuntu 20.04 Operating System. Pytorch 1.5.0 was used for neural network design. We used automated hyper-parameter tuning for learning rate, optimizer choice, and activation function using TPE [47] implemented through Optuna [48]. We found out that Adam optimizer [49] and Hardswish [42] are the best optimizer and activation function, respectively. We did not perform parameter tuning of convolution kernel size. Instead, we used the kernel sizes reported in [10] that proved to be effective. The kernel sizes for different branches of the inception block were taken as 11,19, and 27.

TABLE 2: Summary of scenarios in Fifty dataset [10]

Scenario	Number of classes	Description
All	75 classes	Each file type is considered as a sperate class.
User-specific	11 classes	file types are grouped into 11 classes according to the their use.
Media Carver - Phhotos & Videos	25 classes	File types tagged as: bitmap, RAW, or vidoe are considered as seperate classes; all remaining types as consdered as one group.
Coarse Photo Carver	5 classes	Sperate classes for difference photographic types: JPEG, RAW, vidoes, and bitmap; all remaining file types are consdered as one class.
Specilized JPEG Carver	2 classes	JPEG images is one class; all remaining file types are consdered as one class.
Camera-Specialized JPEG Carver	2 classes	Similar to scnario 2, excpet that the "other" group mimics files found in camera SD cards.

The accuracy for different networks was calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative. To achieve higher accuracy from smaller models, we pretrained our network on corresponding fragment size dataset. We tried to leverage the performance of transfer learning in gaining higher accuracy [50]. In particular, to develop a 4096-byte fragment, the model was pretrained on 512-byte fragment dataset and vice-versa. We found that 6-8% accuracy was increased when pretraining was done using 512-byte fragment data for developing the 4096-byte fragment model. However, similar performance was not achieved when pretraining was done 4096-byte fragment data for the 512-byte fragment model.

D. RESULTS

In general, our proposed models outperform FiFTy in inference time with no significant loss of accuracy. The results are summarized in Table 3. In particular, we observe that our models are 25x and 660x faster than FiFTy and RNN, respectively, in terms of inference time for scenario 1 (75 file types) when running on 1 GB data of 4096-byte fragments.

TABLE 3: Comparison of results of five models on all 75 file types

Model	Neural Network	Block Size	# Params	Accuracy	Speed[ms/block]†	Speed [min/GB]†
DSC	Depthwise Separable CNN	4096	103,083	78.45	**2.567**	**0.055**
		512	103,083	65.89	3.531	0.382
DSC-SE	Depthwise Separable CNN with SE	4096	105,515	**79.27**	3.908	0.064
		512	105,515	66.33	3.473	0.471
M-DSC	Modified Depthwise Separable CNN	4096	**85,291**	78.68	2.606	0.057
		512	**85,291**	64.03	**2.776**	**0.378**
FiFTy	1-D CNN	4096	449,867	77.04	38.189	1.366
		512	289,995	65.66	38.67	3.052
Baseline RNN	LSTM	4096	717,643	70.51	268.58	36.375
		512	379,851	**67.5**	126.54	33.431

† Computed on Nvidia Titan X

TABLE 4: Comparison between FiFTy and our models in terms of inference time and model Parameters

Scenario	Fragment Size	#param				Inf. Time (CPU)[ms/block]				Inf. Time (GPU)[ms/block]			
		DSC	DSC-SE	M-DSC	FiFTy	DSC	DSC-SE	M-DSC	FiFTy	DSC	DSC-SE	M-DSC	FiFTy
1	4096	103,083	105,515	**85,291**	449,867	18.595	23.291	**17.338**	121.476	**2.567**	3.908	2.606	38.189
	512	103,083	105,515	**85,291**	289,995	6.958	8.216	**6.279**	75.551	3.531	3.473	**2.776**	38.673
2	4096	94,827	97,259	**77,035**	597,259	22.587	18.023	**17.214**	92.324	2.763	3.300	**2.610**	29.826
	512	94,827	97,259	**77,035**	269,323	7.827	6.657	**6.346**	89.344	3.239	3.504	**2.910**	29.809
3	4096	96,633	99,065	**78,841**	453,529	**17.068**	18.860	17.361	102.808	3.284	3.182	**2.632**	37.286
	512	96,633	99,065	**78,841**	690,073	**6.066**	6.613	6.300	99.710	3.078	3.499	**2.708**	35.361
4	4096	94,053	96,485	**76,261**	684,485	**17.149**	20.477	17.537	100.117	**2.524**	3.197	2.608	40.176
	512	94,053	96,485	**76,261**	474,885	**6.029**	6.533	6.285	79.346	**2.620**	3.258	2.677	35.965
5	4096	93,666	96,098	**75,847**	138,386	22.427	**16.362**	17.474	99.855	**2.524**	3.362	2.769	39.262
	512	93,666	96,098	**75,847**	336,770	**5.967**	6.327	6.275	79.831	**2.581**	3.259	2.873	42.413
6	4096	93,666	96,098	**75,847**	666,242	18.418	23.000	**17.279**	98.536	**2.530**	3.389	2.641	34.931
	512	93,666	96,098	**75,847**	242,114	**6.133**	8.373	6.398	89.104	**2.612**	3.274	2.702	38.982

Using 512-byte fragments, we observe that FiFTy and RNNs are 8x and 87x, respectively, slower than our models when running on 1 GB of data. Furthermore, It was observed that RNN model performs well with 512-byte fragments due to their ability to handle short sequences. In contrast, when the fragment size increases to 4096-bytes, the performance deteriorates due to the vanishing gradient problem of RNNs [51]. While RNN model has higher accuracy on the 512-byte fragments, RNN is not practical because it takes relatively longer inference time.

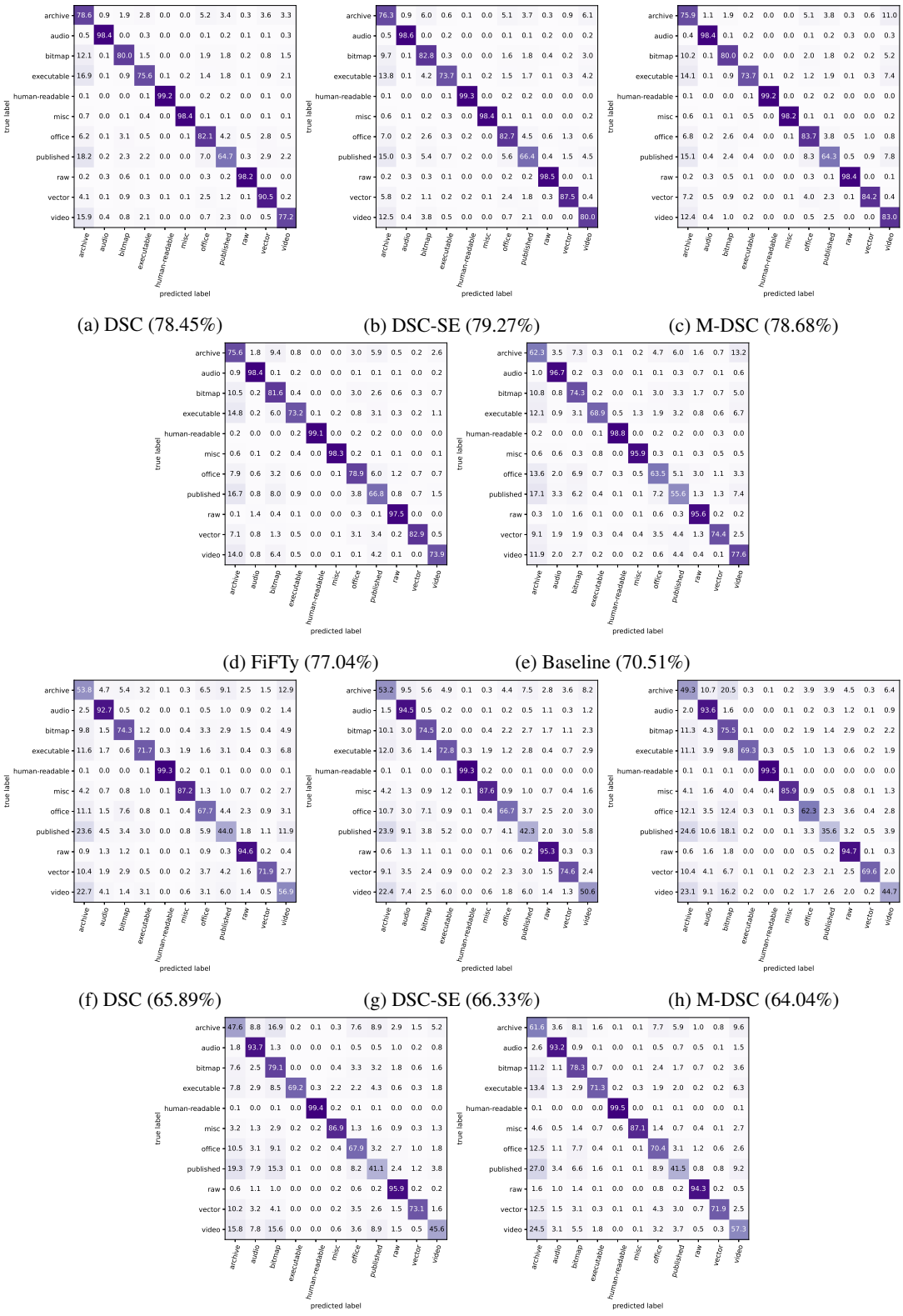
Number of parameters and inference time. Table 4 summarizes the number of neural network parameters in the proposed models compared to FiFTy. For all six scenarios and both fragment sizes, our models have far fewer parameters than FiFTy. In addition, Table 4 compares the inference times for each of the six scenarios on both the GPU and CPU. Our models outperform FiFTy by a large margin on GPU and CPU in term of inference time for all six scenarios. We observed a 5x reduction in inference time for 4096-byte file fragments and a 9x reduction for 512-byte file fragments compared to FiFTy on CPU. For GPU, the time reduction is more than 9x for both fragment types.

Number of FLOPS. To provide hardware-independent metrics, we calculated the floating-point operations in our models and FiFTy. In Table 5, we provide the results of the comparison. Compared to FiFTy, our models have 6.3x fewer FLOPS for 4096-byte fragments and 87x fewer for 512-byte fragments. In Scenario 1, for example, with a fragment size of 4096-bytes, DSC has 164.88, DSC-SE has 164.96, and M-DSC has 89.07 MFLOPs, while FiFTy has 1047.59 MFLOPs. These results demonstrate the effectiveness of our proposed models, as they require fewer computations to make predictions while still maintaining high accuracy. This makes our models more suitable for real-world applications where computational resources are limited.

TABLE 5: Comparison between FiFTy and our models for floating point operations (Mega FLOPs)

Scenario	Fragment size	FLOPs			
		DSC	DSC-SE	M-DSC	FiFTy
1	4096	164.88	164.96	**89.07**	1047.59
	512	20.63	20.64	**11.15**	1801.71
2	4096	164.86	164.95	**89.05**	1327.90
	512	20.61	20.63	**11.13**	918.06
3	4096	164.86	164.95	**89.05**	647.78
	512	20.61	20.63	**11.14**	3579.57
4	4096	164.86	164.95	**89.05**	2378.51
	512	20.61	20.63	**11.13**	1576.71
5	4096	164.86	164.95	**89.05**	488.37
	512	20.61	20.63	**11.13**	2330.48
6	4096	164.86	164.95	**89.05**	1126.00
	512	20.61	20.63	**11.13**	611.30

Accuracy. The confusion matrix of file types grouped by use-cases for Scenario 1 is plotted in Figure 7 (the grouping is listed in Table 1). Most of the misclassification occurred in the archived file group because of other file types embedded in them. The accuracy of the proposed models and FiFTy is listed in Table 6. While FiFTy achieves comparable accuracy, our light-weight models offer a significant advantage in inference time and resource efficiency, making them more suitable for real-time applications.



(a) DSC (78.45%) (b) DSC-SE (79.27%) (c) M-DSC (78.68%)

(d) FiFTy (77.04%) (e) Baseline (70.51%)

(f) DSC (65.89%) (g) DSC-SE (66.33%) (h) M-DSC (64.04%)

(i) FiFTy (65.66%) (j) Baseline (67.5%)

FIGURE 7: Confusion matrices for Our models, FiFTy [10] and Baseline RNN model on 4 KB fragments in (a)-(e) and 512 bytes in (f)-(j). Due to large number of classes(75), classes belonging to same super class were clustered into one.

TABLE 6: Comparison between FiFTy and our models in terms of accuracy

Scenario	#of files	Fragment Size	Accuracy DSC	DSC-SE	M-DSC	FiFTy
1	75	4096	78.45	**79.27**	78.68	77.04
		512	65.89	**66.33**	64.04	65.66
2	11	4096	85.7	87.10	85.33	**89.91**
		512	75.84	74.99	72.18	**78.97**
3	25	4096	93.06	93.32	91.87	**94.64**
		512	80.79	80.79	78.53	**87.97**
4	5	4096	94.17	**94.61**	92.66	94.03
		512	87.14	87.32	83.41	**90.30**
5	2	4096	99.28	**99.37**	99.24	99.12
		512	98.94	98.96	98.94	**99.07**
6	2	4096	99.59	**99.69**	99.62	99.59
		512	98.76	98.65	98.87	**99.23**

E. ABLATION STUDY

This section presents an ablation study to evaluate the impact of SE blocks on the performance of a depthwise separable convolutional network for the classification of file fragment types using the Fifty dataset. We incrementally introduced SE blocks into the network to assess their effect on the accuracy of the model and the number of parameters. The baseline model comprises a depthwise separable convolutional network without any SE blocks, serving as the control setup. We progressively introduced SE blocks into the network through four experiments: 1) *Baseline*: model without any SE blocks, 2) *First SE block only*: model with an SE block after the first inception block, 3) *First and second SE blocks*: model with SE blocks after the first and second inception blocks, and 4) *Full SE incorporation*: model with SE blocks after each of the three inception blocks. The introduction of SE blocks aims to leverage their capacity for adaptively recalibrating channel-wise feature responses, potentially enhancing the model's ability to emphasize informative features for classification. Initially, our baseline model, which lacked any SE blocks, achieved an accuracy of 78.45% with 103,083 parameters. Incorporating only a single SE block slightly improved accuracy to 78. 59%, with a minor increase in parameter count to 103,211. The addition of a second SE block further enhanced the model accuracy to 78. 82%, although with a slightly higher parameter count of 105,259. The greatest improvement was observed when the SE blocks were fully incorporated after each of the three inception blocks, culminating in the highest accuracy of 79.27% with 105,515 parameters. Table 7 shows the results of the ablation study.

These results highlight the advantage of introducing SE blocks in improving classification accuracy through the adaptive recalibration of features, which allows for more effective feature extraction. The gradual improvements underscore the benefit of recalibration at various network depths. Despite the modest increase in parameters, the improvement in accuracy justifies the inclusion of SE blocks to strike a favorable balance between complexity and performance. In summary, the ablation study confirms the effectiveness of SE blocks in augmenting the representational capacity of convolutional networks for complex tasks like file fragment type classification. This underscores the importance of feature recalibration mechanisms in enhancing deep learning architectures, contributing to the broader understanding of designing more efficient and accurate models.

TABLE 7: Ablation study results

Model SE	Scenario #1 Accuracy	# Parameters
Baseline (Without SE)	78.45%	103,083
Adding one SE Block	78.59%	103,211
Adding two SE Blocks	78.82%	105,259
DSC-SE model (3 SEs)	79.27%	105,515

F. DISCUSSION

The results show that the proposed models have similar inference times with minimal variance between runs. Our models are optimized for efficient inference using cutting-edge techniques to minimize computational overhead and reduce latency. As a result, we were able to achieve reduced inference times compared to state-of-the-art models while maintaining relatively high accuracy. This demonstrates the effectiveness of our proposed models in real-world applications where fast and efficient inference is crucial.

Similarly to previous work [9], [10], we observe that files with high entropy are difficult to classify because there is no statistical trace that the convolutional kernel can extract. Moreover, many files are container types that contain other files as embedded objects, e.g., *pdf* files that can contain embedded *jpg* images. As a result, classifiers behave erratically. Finally, similar file types with different format, e.g., *ppt*, *pptx*, and *key*, are misclassified among themselves. However, *doc* and *docx* are not affected by this as *docx* uses *XML* whereas *doc* is stored as binary.

VI. CONCLUSION

In this paper, we proposed light-weight file fragment type classification models utilizing depthwise separable CNNs. The objective is to develop a model with better inference time compared to the state-of-the-art models without compromising on accuracy. Specifically, we succeeded in crafting classification models with approximately 100K parameters. The models were evaluated using the FFT-75 dataset, which includes different scenarios. The evaluation results indicated that the proposed models perform faster than the state-of-the-art file fragment type classification model for all scenarios without significant accuracy degradation. Various enhancements can be used to increase the classification accuracy for classes with high misclassification rates. Without hardware constraints, the search for neural architectures [52], [53] proves to be the optimal approach to devise an architecture for data-specific models. Furthermore, for straightforward scenarios such as classifying JPEGs against other file types, redundant connections in the neural network can be eliminated using distillation in the network [54]. In future work, we plan to evaluate the performance of our models on imbalanced datasets to further validate their robustness and

effectiveness in real-world applications, ensuring they can handle diverse and challenging scenarios.

ACKNOWLEDGMENT

The authors acknowledge the support from the Interdisciplinary Research Center for Intelligent Secure Systems at KFUPM.

REFERENCES

[1] M. Rafique and M. Khan, "Exploring static and live digital forensics: Methods, practices and tools," International Journal of Scientific & Engineering Research, vol. 4, no. 10, pp. 1048–1056, 2013.

[2] D. Bennett, "The challenges facing computer forensics investigators in obtaining information from mobile devices for use in criminal investigations," Information Security Journal: A Global Perspective, vol. 21, no. 3, pp. 159–168, 2012.

[3] S. L. Garfinkel, "Carving contiguous and fragmented files with fast object validation," digital investigation, vol. 4, pp. 2–12, 2007.

[4] X. Lin, "File carving," in Introductory Computer Forensics. Springer, 2018, pp. 211–233.

[5] G. G. Richard III and V. Roussev, "Scalpel: A frugal, high performance file carver." in DFRWS. Citeseer, 2005.

[6] K. Alghafli, C. Y. Yeun, and E. Damiani, "Techniques for measuring the probability of adjacency between carved video fragments: The vidcarve approach," IEEE Transactions on Sustainable Computing, 2019.

[7] 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 Transactions on Information Forensics and Security, vol. 13, no. 10, pp. 2553–2562, 2018.

[8] 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 Transactions on Information Forensics and Security, vol. 8, no. 9, pp. 1519–1530, 2013.

[9] Q. Chen, L. C. Hui, D. Liu, E. Zhang, Q. Liao, Z. L. Jiang, J. Fang, S. Yiu, G. Xi, and R. e. a. Li, "File fragment classification using grayscale image conversion and deep learning in digital forensics," 2018 IEEE Security and Privacy Workshops (SPW), 2018.

[10] G. Mittal, P. Korus, and N. Memon, "Fifty: large-scale file fragment type identification using convolutional neural networks," IEEE Transactions on Information Forensics and Security, vol. 16, pp. 28–41, 2020.

[11] K. Zhang, K. Cheng, J. Li, and Y. Peng, "A channel pruning algorithm based on depth-wise separable convolution unit," IEEE Access, vol. 7, pp. 173 294–173 309, 2019.

[12] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 04 2017.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[14] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in International conference on machine learning. PMLR, 2019, pp. 6105–6114.

[15] K. M. Saaim, M. Felemban, S. Alsaleh, and A. Almulhem, "Light-weight file fragments classification using depthwise separable convolutions," in ICT Systems Security and Privacy Protection: 37th IFIP TC 11 International Conference, SEC 2022, Copenhagen, Denmark, June 13–15, 2022, Proceedings. Springer, 2022, pp. 196–211.

[16] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] I. F. Darwin. (2008) Libmagic. [Online]. Available: ftp://ftp.astron.com/pub/file

[19] M. Karresand and N. Shahmehri, "Oscar - file type identification of binary data in disk clusters and ram pages," in SEC, 2006.

[20] I. Ahmed, K.-S. Lhee, H.-J. Shin, and M.-P. Hong, "Fast content-based file type identification," Advances in Digital Forensics VII IFIP Advances in Information and Communication Technology, p. 65–75, 2011.

[21] Q. Li, A. Ong, P. Suganthan, and V. Thing, "A novel support vector machine approach to high entropy data fragment classification," 01 2010.

[22] S. Fitzgerald, G. Mathews, C. Morris, and O. Zhulyn, "Using nlp techniques for file fragment classification," Digital Investigation, vol. 9, 2012.

[23] N. Zheng, J. Wang, T. Wu, and M. Xu, "A fragment classification method depending on data type," 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 1948–1953, 2015.

[24] M. C. Amirani, M. Toorani, and S. Mihandoost, "Feature-based type identification of file fragments," Security and Communication Networks, vol. 6, pp. 115–128, 2013.

[25] M. Bhatt, A. Mishra, M. W. U. Kabir, S. Blake-Gatto, R. Rajendra, M. T. Hoque, and I. Ahmed, "Hierarchy-based file fragment classification," Machine Learning and Knowledge Extraction, vol. 2, no. 3, pp. 216–232, 2020.

[26] M. E. Haque and M. E. Tozal, "Byte embeddings for file fragment classification," Future Generation Computer Systems, vol. 127, pp. 448–461, 2022.

[27] Y. Wang, Z. Su, and D. Song, "File fragment type identification with convolutional neural networks," Proceedings of the 2018 International Conference on Machine Learning Technologies - ICMLT 18, 2018.

[28] L. Hiester, "File fragment classification using neural networks with lossless representations," 2018.

[29] 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 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, 2023, pp. 1–5.

[30] J. G. Park, S. Liu, and J. H. Hong, "Xmp: A cross-attention multi-scale performer for file fragment classification," in ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2024, pp. 4505–4509.

[31] 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," Knowledge-Based Systems, vol. 291, p. 111565, 2024.

[32] K. Skračić, J. Petrović, and P. Pale, "Bytercnn: Enhancing file fragment type identification with recurrent and convolutional neural networks," IEEE access, 2023.

[33] K. Fukushima, "A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biol. Cybern., vol. 36, pp. 193–202, 1980.

[34] L. SIfre and S. Mallat, "Rigid-motion scattering for texture classification," 03 2014.

[35] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 07 2017, pp. 1800–1807.

[36] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295–2329, 2017.

[37] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856.

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6 2018.

[39] K. Vulinović, L. Ivković, J. Petrović, K. Skračić, and P. Pale, "Neural networks for file fragment classification," in 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2019, pp. 1194–1198.

[40] A. Samanta, I. Hatai, and A. K. Mal, "A survey on hardware accelerator design of deep learning for edge devices," Wireless Personal Communications, pp. 1–46, 2024.

[41] B. Peng, E. Alcaide, Q. G. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. N. Chung, L. Derczynski et al., "Rwkv: Reinventing rnns for the transformer era," in The 2023 Conference on Empirical Methods in Natural Language Processing, 2023.

[42] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., "Searching for mobilenetv3," in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1314–1324.

[43] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax

activation function," in Proc. 8th Aust. Conf. on the Neural Networks, Melbourne, vol. 181. Citeseer, 1997, p. 185.
[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in International conference on machine learning. pmlr, 2015, pp. 448–456.
[45] Y. Wu and K. He, "Group normalization," in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 3–19.
[46] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," Digital Investigation, vol. 6, 2009.
[47] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in Advances in Neural Information Processing Systems 24, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2546–2554.
[48] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[50] L. Y. Pratt, "Discriminability-based transfer between neural networks," in Advances in neural information processing systems, 1993, pp. 204–211.
[51] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 6, no. 02, pp. 107–116, 1998.
[52] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.
[53] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in International conference on machine learning. PMLR, 2018, pp. 4095–4104.
[54] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," in International Conference on Learning Representations, 2020.

MUSTAFA GHALEB is a Postdoctoral Research Fellow at the Interdisciplinary Research Center for Intelligent Secure Systems (IRC-ISS) at King Fahd University of Petroleum and Minerals (KFUPM) in Saudi Arabia. He obtained his M.Sc. and Ph.D. degrees in Computer Science from KFUPM. Ghaleb's research interests include cybersecurity, Internet of Things (IoT), distributed computing, NLP, trust modeling, and deep learning applications in various domains.

MUHAMAD FELEMBAN is an Assistant Professor in the Computer Engineering department at KFUPM. He is also the director of the Interdiscipinary Research Center for Intelligent Secure Systems (IRC-ISS). He obtained his PhD in Electrical and Computer Engineering from Purdue University in 2018, MSc degree in Computer Science from KAUST in 2011, and BSc in Computer Engineering from KFUPM in 2008. His research interests include cybersecurity, data privacy, and quantum computing.

KUNWAR SAAIM earned his bachelor's degree in computer engineering from Aligarh Muslim University in 2021. Currently, he is pursuing a master's degree in computing science at the University of Alberta. His primary area of research interest lies in the field of deep learning.

SALEH AL-SALEH received B.Sc degree (Honos.) in Computer Science from King Fahd University of Petroleum and Minerals (KFUPM), in 2016, the M.A.Sc degree in Computer Engineering from King Fahd University of Petroleum and Minerals (KFUPM) in 2019. He is currently a Lecturer in the Computer Engineering Department at KFUPM. His research interests include Digital Design, FPGA, Embedded Systems, and Computer Architecture.

AHMAD ALMULHEM is an assistant professor in the Computer Engineering Department, KFUPM, Saudi Arabia. His research interests span various aspects of computer security and digital forensics including threat modeling, user authentication, cyber-physical security, and applications of machine learning and AI techniques in these fields.