Camera-Model Identification Using Encoding and Container Characteristics of Video Files

Enes Altinisik, Hüsrev Taha Sencar

Abstract—We introduce a new method for camera-model identification. Our approach combines two independent aspects of video file generation corresponding to video coding and media data encapsulation. To this end, a joint representation of the overall file metadata is developed and used in conjunction with a two-level hierarchical classification method. At the first level, our method groups videos into metaclasses considering several abstractions that represent high-level structural properties of file metadata. This is followed by a more nuanced classification of classes that comprise each metaclass. The method is evaluated on more than 20K videos obtained by combining four public video datasets. Tests show that a balanced accuracy of 91% is achieved in correctly identifying the class of a video among 119 video classes. This corresponds to an improvement of 6.5% over the conventional approach based on video file encapsulation characteristics. Furthermore, we investigate a setting relevant to forensic file recovery operations where file metadata cannot be located or are missing but video data is partially available. By estimating a partial list of encoding parameters from coded video data, we demonstrate that an identification accuracy of 57% can be achieved in camera-model identification in the absence of any other file metadata.

I. INTRODUCTION

An essential requisite for media forensics is the ability to track the provenance of multimedia content. Such a capability may answer several questions about the origin of an image or video with differing levels of specificity. At the one end, the focus may be on identifying whether a given media is camera-captured or a deepfake, i.e., synthesized by a deep neural network. At the other end, it may be about attributing the media to the particular device that generated it. In between, the inquiry may concern some class-level characteristics of the media at hand such as the brand and model of the acquiring camera, the social media platform it is downloaded from, or the deep learning algorithm used for generation.

Forensic methods proposed to address different instances of the attribution problem mainly adopt a content-based analysis approach to uncover relevant forensic traces. Among these, source camera attribution based on a sensor’s photo-response non-uniformity (PRNU) is now very well established. The most important factor underlying the effectiveness of this method is the multi-dimensional nature of the PRNU, which depends on the number of photosensitive elements (i.e., pixels) in a sensor, thereby yielding a highly discriminative signal. Considerable success is also reported in source attribution settings that involve a classification problem with fewer classes such as detecting deepfake media and provenance tracing in online social networks.

In contrast to these successes, reliably identifying the camera model that produced a media remains a challenge. Essentially, camera-model attribution requires the extraction of distinctive characteristics that can distinguish a large number of source classes. In this regard, characterizing specifics of individual processing steps applied in-camera and during post-processing (such as color filter array configuration, demosaicing algorithm, lens-related distortions) or the better performing data-driven modeling methods do not yield sufficient discriminative information to distinguish between many camera models. Improving the accuracy of this task crucially depends on identifying and combining several such model-dependent characteristics.

In source camera-model identification, an alternative approach to content-based analysis exploits the representation aspect of image and video data. Multimedia data is almost always stored and transferred in some compressed file format, therefore it is accompanied by rich metadata and information that govern its organization to ensure successful reconstruction. Moreover, due to model-specific variations in implementation of formatting standards and the choice in parameters, attributes of a video file are expected to vary across camera models while remaining largely invariant among media captured by the same camera model. Combined with the fact that such information is readily available, without the need for a complicated extraction procedure, file structure and the associated metadata potentially offer a high discrimination capability.

Videos typically include a large amount of audio-visual data with strict rendering requirements, and video coding is a quite sophisticated process involving many parameters. Therefore, components and structure of a video file provide a suitable basis for camera-model identification. This was initially observed by Gloe et al. who examined videos in several file formats generated by many cameras and editing tools. Based on this observation, the subsequent work focused on file container formats and exploited the fact that file data is packed in a sequence of compartments, known as boxes in MPEG 4-based file formats and resources in AVI format, to identify camera model/brand and detect video tampering.

In , Song et al. proposed using the sequence of AVI metadata fields in a video as a signature of a camera model. The following research mainly focused on developing a representation for video file metadata. For this, Iuliani et al. proposed using an ordered listing of MP4 file metadata fields and their values as a representative feature vector. Yang et al. modified this initial representation by disregarding order information and decoupling fields from their values. To improve the robustness of the representation against insertion and deletion of fields, Gelbing et al. introduced an alternative enumeration of fields. Based on a similar repre-
sentation, Xiang et al. [26] proposed another improvement in how non-categorical values are expressed and performed linear discriminant analysis to further reduce the dimensionality of the feature vector.

Although these methods utilize the rich metadata describing the structure of the container file format, the parameter values that identify the encoding and decoding setting of a video have not been methodically utilized for camera-model identification. Since the coder-decoder (codec) used for (de)compression and the file format governing the arrangement of coded data can be selected independently, the use of codec parameters provides a complementary capability to existing approaches that do not apply to container formats such as Transport Stream (TS) and Matroska Multimedia Container (MKV). According to the Video Developer Survey 2019 [27], 91% of the developers use H.264 video coding standard. In concert with this, analysis of more than 100K user-uploaded videos revealed that 99.6% of those were encoded using H.264 format [28]. Thus, the use of H.264 coding parameters should be expected to improve achievable distinguishability of camera models.

Motivated by this, in this work, we introduce the use of H.264 video sequence headers that contain sets of codec parameters for the source attribution task. We examine how the parameter sets used by the H.264 codec vary within and across camera models. We then demonstrate the complementary nature of encapsulation and coding characteristics by combining them. To obtain a joint representation we combine these characteristics in a file-metadata tree, and introduce a hierarchical method that first exploits topological properties of the resulting tree representation followed by the use of field and parameter values to provide a more scalable approach to camera-model classification. Our method is validated on the largest dataset used in any study of this nature, which includes 20,153 videos obtained by combining four public datasets, including the VISION [29], ACID [30], SOCRatES [31] and EVA-7K [24] datasets. Our proposed method yielded a balanced accuracy of 91% in distinguishing videos generated by 112 camera models, 3 video editing tools, and 4 social media platforms. Results show that incorporation of encoding parameters as part of file metadata yields an overall improvement of 6.5% in accuracy.

The use of codec parameters in the identification of source camera models also allows this capability to be extended to partially available videos. This corresponds to an investigative setting encountered frequently in the process of recovering files from storage devices during which file metadata is largely unavailable. In such cases, some encoding parameters can be determined through analysis of coded frame data as demonstrated in [28]. Hence, parameters estimated from incomplete video files can also be used to infer the camera model. Our results show that even in this limited setting camera models can be identified with a balanced accuracy of 57%.

The rest of the paper is organized as follows: In the next section, we describe the video generation process from the perspective of H.264 encoding and MP4 file encapsulation due to their prevalence. The distinctiveness of coding parameters is analyzed in Sec. III. Our representation for file metadata incorporating both aspects of video files is introduced in Sec. IV and details of the two-level hierarchical classification method are described in Sec. V. The results for identification accuracy on the combined dataset are provided in Sec. VI considering several test settings. Finally, we discuss our findings in Sec. VII and present our conclusions in Sec. VIII.

II. VIDEO FILE GENERATION

A video file is generated in a camera at the last processing stage of the acquisition pipeline. At this stage, the main objective is to reduce the size of frames created by the earlier processing stages and package the coded data before it can be saved. A video encoder performs one part of this task by removing the temporal and spatial redundancy in and between successive frames. Several video coding standards with improved coding performance have been introduced over time, such as MPEG-1, MPEG-2 (H.262), MPEG-4 (H.263), MPEG-4 AVC (H.264), and HEVC (H.265). In addition, several alternative encoding formats that are inspired by H.264 and H.265 standards have been developed independently, including VP8/9, VC, and AV1.

The coded video frames are then combined with other essential data, including coded audio segments, encoding parameters, subtitles, etc., in a container file. File container formats are essentially optimized for different use cases, i.e., streaming or playback, and support a range of audio and video codecs. For playback videos, the ISO/IEC 14496-12 (MPEG-4 Part 12) standard specifies a general media file format [32], and many widely used file formats, such as MP4, 3GP, and MOV, are based on it. Besides, there are several open and proprietary container formats such as MKV and AVI. For video streaming, the MPEG Transport Stream (TS) standard specifies a container format for encapsulating packetized data. This forms the basis of the most commonly used streaming formats such as HTTP Live Streaming (HLS) and MPEG-DASH [27].

Among the existing file encoding formats, H.264 is by far the most widely used codec today [27], [28]. In terms of file container formats, despite a great diversity of options, mobile phones predominantly use file formats that extend over the MPEG-4 standard [33]. Since smartphones serve as the de facto camera, methods proposed for the characterization of video files mainly focused on MP4 files. Therefore, we examine the overall structure and content of H.264 encoded videos contained in MP4-like file formats more closely.

1) H.264 Coding Parameters: H.264 encoding follows a block-based coding approach where compression of picture blocks involves the key steps of prediction, (in-loop) de-blocking filtering, transformation, quantization, and entropy coding [34]. These processing steps are governed by a set of parameters that are dynamically determined by the encoder to attain a target compression bitrate and picture quality. The same parameters are required at the decoder to recreate the video. Therefore, they are stored along with the coded data. The H.264 video coding parameters can be grouped into three sets:

**Sequence Parameter Set (SPS):** The SPS includes 38 parameters and applies to a sequence of pictures that are encoded in an interdependent manner. Some of these parameters specify
general characteristics of the video, such as resolution, bitrate, and frame rate, so that the decoder can ensure it has sufficient resources to handle the video bitstream. Another subset includes parameters that define the bit widths of several variables needed for parsing the bitstream. A third subset of parameters provides values for variables such as the width and height of video pictures and the number of reference frames that must be kept in memory during decoding. The remaining subsets serve as sequence metadata and contain information like SPS ID (as multiple SPSs can be present) and a bit flag to denote whether video usability information parameters are included as part of the header.

**Picture Parameter Set (PPS):** A PPS contains 25 parameters that specify the entropy coding method, motion prediction mode, baseline quantization values, and deblocking filter settings. Every picture in a sequence may be accompanied with a separate PPS header. The PPS, when combined with the SPS, provides all the information needed to reconstruct a picture in a standalone manner.

**Video Usability Information (VUI):** The VUI parameters are optional and are used in the post-decoding stage to prepare the resulting video sequence for output and display. This may include up to 32 parameters and may provide supplemental information about the aspect ratio of coded pictures, the original format of the source video and its color space, picture output timing to ensure correct play-out speed of the decoded video sequence, and potential bitstream restrictions that help the decoder pick the right computational configuration for decoding. When they are present, VUI parameters appear as part of the SPS.

2) **Structure of MP4-like Files:** In MP4 and other similar container file formats based on the ISO/IEC 14496 Part 12 standard, the basic data structure is referred to as a box or an atom following the earlier-defined QuickTime file format. Accordingly, each box is identified by a four byte ASCII code showing its type which is preceded by a four-byte value indicating the size of the box. The boxes in a file are stored in a hierarchical order allowing each box to contain further sub-boxes as well as data values organized in attribute fields. Hence the structure of data contained within MP4-like files can be represented as a labeled-tree where internal nodes include boxes labeled by their type and leaf nodes labeled by field-value attributes. Many different types of boxes appear at the top of the hierarchy, most notably the ftype, moov, mdat boxes. Among these, the moov box is the most important and has several children nodes containing information about the movie header, audio and video tracks, and references to raw data. In terms of the size of contained data, the mdat box takes most of the file size as it contains the coded media data.

3) **Encapsulation of Coding Parameters:** In MP4-like file containers, the accv box nested under the moov box stores a variety of information on the coded bitstream. Among these, the AVC Decoder Config Record contains the needed SPS and PPS headers. The headers are stored as bit strings which the decoder parses into parameters to initialize itself prior to decoding the video bit stream. Our examination revealed that the commonly used parser in extracting MP4 file metadata does not extract this information. Regardless of a parser’s behavior, however, expressing encoding parameters in a combined manner as field-attribute values suffers from two main limitations. First, this is a sub-optimal way of combining otherwise orthogonal sets of features that relate to coding and packaging aspects. Therefore, videos encoded using H.264 but encapsulated into other formats, such as transport streams, cannot be characterized. Second, and more critically, storing the whole SPS and PPS as a bit string essentially corresponds to combining up to 95 parameters, all of which may independently serve as features, into a single feature. As a consequence of this, a change in any of the parameter values will result in a completely new value in the accv box field. Therefore, the difference between camera models cannot be accurately distinguished.

### III. Distinctiveness of Encoding Parameters

We explore the distinctiveness of encoding parameters used by different camera models during video coding. To this objective, we examined 19,472 H.264 coded videos in the ACID [30], VISION [29], SOC RatES [31], and EVA-7K [24] datasets. These videos are divided into 116 classes that include 109 camera models, three video editing tools, and four social media platforms. In all these videos, we identified a total of 535 distinct encoding settings involving 124 unique SPS, 53 unique PPS, and 407 unique VUI headers. In total, only 18 video classes are observed to not include the optional VUI parameters. By comparing values of parameters extracted from these videos, we evaluate the degree of distinctiveness exhibited at the class level.

A. **Within-Video Variation**

The H.264 standard allows an encoder to vary its parameters during coding of a video sequence. Therefore, we first examine how variant SPS and PPS are over the duration of a video. Our analysis revealed that except for one class of videos, all videos are encoded using fixed sets of parameters. That is, a video stream is encoded using the same SPS and PPS specific to that class. This can be attributed to the fact that video coding must be performed in real-time with limited computational resources without much room for optimization. For the Sony Cyber Shot DSC-WX350 camera, however, each video is determined to use four different PPSs in their coding. This essentially indicates that in practice a video file can be characterized by the SPS and PPS used in its generation.

B. **Within-Class Variation**

Next, we examine the variation of encoding parameters within and across video classes. To this objective, we create a dictionary by collating all headers associated with each video class. This allows us to determine how many distinct headers are used by each video class and reveal the overlap of headers among video classes. Such a dictionary is created separately for SPS, PPS, and VUI parameter sets, and considering the combination of the three that defines the overall encoding setting of a video.
Figure 1 displays a measure of within-class variability in terms of the count of video classes with a specified number of distinct headers used in their generation. Accordingly, 93 video classes in our dataset are generated using fixed SPS, VUI, and PPS headers. This finding does not imply that encoding settings are unique to those classes but rather it shows that these video classes don’t exhibit any within-class variation. All the classes that involve more than eight encoding settings are associated with either editing tools or social media platforms. Among these, videos sourced from YouTube and Facebook comprise the most diverse classes. This is most likely due to these platforms trying to preserve the quality of the original videos by using a compatible encoding setting when re-encoding them. As for the two video editing tools, Avidemux and FFmpeg, they were configured to carry over the original codec parameters of the source video to the edited version.

It is also determined that the VUI parameter set shows higher within-class variation with several classes having more than 30 VUI parameter sets. Since VUI parameters appear as part of the SPS, this finding implies that treating the SPS string found in the accv box as a feature value by itself results in the creation of many features for the same video class. This emphasizes the need for decoupling encoding parameters to better differentiate video classes.

When examined at the parameter level, the within-class variation is mostly due to a small number of parameters. Out of the 95 parameters involved in video coding, 34 parameters exhibit no variation in any of the 119 video classes, i.e., for each video class these parameters take class-specific but fixed values. Figure 2 shows the number of classes where each of the remaining 61 parameters exhibited some variation. Accordingly, the parameter that varies most across all video classes is the level_idc parameter which indicates the quality of a video and shows within-class variation in 18 video classes. This is followed by four frame resolution-related parameters that take different values in 16 to 17 video classes. Overall, we determine that 79 parameters (those excluding the first 16 in Fig. 2) show variation only in 10 or fewer classes. Since parameters that exhibit high within-class variation cannot be intrinsic to a video class, these 79 parameters will be more distinctive in identifying video classes.

C. Between-Class Variation

To evaluate the between-class variation, we examined the overlap of headers among video classes. Figures 3(a) and (b) display corresponding confusion matrices where each row shows the relation between a video class and others in terms of the intersection of the common SPS, VUI, PPS dictionary items as well as their unique combination. Due to the symmetric nature of a confusion matrix, in each figure, the area above and below the diagonal is associated with a different parameter set. In these figures, the intensity at a particular row and column is indicative of the number of overlapping header entries between corresponding video classes. These confusion matrices essentially demonstrate that encoding settings are not unique to each video class but each class overlaps only with a small number of classes with very few video classes getting confused with a large number of classes. This indicates that video classes can be clustered based on their encoding settings.

IV. REPRESENTATIONS FOR FILE METADATA

In the MP4 file format, information is organized in a hierarchy of boxes that either contain other boxes or data organized in attribute fields. Since boxes may appear in any order, have different boxes nested under them, and vary in their values, their organization and content serve as a means for characterizing the source device that encapsulated a media file. Several studies have proposed ways of interpreting this tree-structured metadata to obtain a feature representation. In common, this is realized by extracting all paths of a metadata tree starting at the root node and ending at leaves [23]–[26], [36]. By combining labels of all nodes (i.e., box types) along a path with the field-value pairs at the leaf nodes, a
A downside of the above representation is that it yields very high-dimensional, sparse vectors as each distinct path receives its own dimension. Xiang et al. [26] proposed an alternative representation to lower the dimensionality of the feature vector. For this, root-to-leaf paths are incorporated with paths extending to each node in the tree, i.e., root-to-node paths, while encoding order information only in the labels of the media track (trak) boxes. Unlike previous approaches that map each distinct path to a separate feature, this approach uses the number of occurrences for each distinct path entry as a feature. In addition, root-to-leaf paths are discriminated depending on the type of box field, i.e., whether they take categorical or non-categorical values with the latter ones assigned their value as features, such as width, height, bitrate, rather than their number of occurrences. To further reduce the dimension of the resulting feature vector, correlated features are eliminated through a feature selection procedure, and linear discriminant analysis (LDA) is used to ultimately obtain a two-dimensional feature vector.

An important consideration here is the parsing of metadata from the video files. The flexibility of the MP4 file format allowed manufacturers to accommodate changes in the technology and to address new use cases. However, such manufacturer-specific additions also introduce a complication that prevents the design of a universal parser. In this regard, the MP4 Parser library [35] has been a commonly used tool for this purpose [23], [25]. It is determined in [26] that this tool fails to correctly extract the information in the list (metadata item list) box due to its non-standard usage. Similarly, it is observed that video editing tools incorporate another layer of metadata that requires further interpretation. Building a comprehensive parser is not trivial as it requires knowledge of manufacturer-level variations in the metadata. Therefore, in this study, we use two publicly available parsing libraries [35].

V. PROPOSED METHOD

As encoding and encapsulation characteristics of a file represent two independent aspects, they are complementary in nature. Therefore, our method combines both sources of information for more accurate camera-model identification. To this end, we first introduce a joint representation for file metadata, named file-metadata tree. Then, we use a two-level hierarchical framework for classification as a stratified learning approach will provide better scalability in meeting the challenge of performing camera-model identification in the presence of a large number of camera models. At the first-level, we evaluate topological properties of the file-metadata tree to perform a coarse classification considering three abstractions. Then at the second level, we map the file-metadata representation to a feature vector in order to identify the camera model.

A. File-Metadata Tree

To obtain a coherent representation, the file structure information and encoding parameters are combined in a file-metadata tree. This tree consists of two subtrees whose nodes are annotated with labels from a relatively large alphabet comprising box types and parameter names and whose leaf nodes correspond to field and parameter values. The first subtree comprises the encapsulation-related metadata. In fact, the hierarchical nature of the MP4 file format readily lends itself to such a representation. Accordingly, the internal nodes of this subtree correspond to boxes and field attributes reflecting the order they appear in the file. The value of each field attribute is then added to this subtree as a child node descending from the node denoted by the field name.

In contrast, encoding parameters comprise three, ordered feature vectors of varying lengths. To organize the encoding parameters as a subtree, the SPS, PPS, and VUI are connected to a root node as its child nodes and labeled accordingly. Each of these three nodes is then appended as many child nodes as the number of parameters they contain and in the fixed order

![Image of file-metadata tree](image-url)
they appear in SPS and PPS headers. These second-level nodes are labeled by the name of the parameters they correspond to. Similar to field-value attributes, each parameter node is appended a leaf node containing its value. The complete file-metadata tree is then generated by combining the two subtrees as shown in Fig. 5.

When extracting file metadata, we used the publicly available software modules. In the case of H.264 coded files, the encoding parameters are extracted using the H264 BitStream toolbox.[38] For MP4 files, these include the MP4 Parser library[35] and the GPAC library[37] which will be respectively referred to as MP4 Parser and GPAC. Our examination revealed that MP4 Parser yielded some errors when processing MOV formatted files, failing to read some of the fields, such as SPS and PPS strings. As reported in [26], some box fields contain information that reflect a manufacturer-specific usage (such as the ilst box); neither parser supports such non-standard coding of information. Nevertheless, we observed that both parsers are able to extract different information and one is not superior over the other. Therefore, we used both parsers as part of our method.

Finally, a part of extracted file metadata is specific to the contained media and varies from one file to another. Previous work has already identified several fields that cannot be used for attribution to the camera-model. In our approach, we also excluded all fields that are explicitly related to the video data such as duration, data size values, and several types of entry count values. For others, we rely on our classification method's ability to weed out irrelevant information.

B. Learning at First Level

The design of our method is mainly motivated by the observation that the topology of a file-metadata tree does not vary significantly across camera-models and, in fact, remains very similar or the same for many video classes. Hence by first identifying metaclasses that group together several video classes, this step facilitates better learning of the nuances in variation in field and parameter values between similar video classes. In a way this can be viewed as analogous to performing brand identification first. This approach, however, further exploits the fact that different brands or editing tools may exhibit similar encapsulation and encoding behavior. At this level, we consider three different levels of abstraction for the file-metadata tree. For this purpose, the whole file-metadata tree with the exclusion of leaf nodes containing values of box fields and encoding parameters is used.

1) Tree Topology: This representation disregards the order and labels of nodes in a tree to perform a coarse characterization. That is, only the shape of a file-metadata tree is taken into consideration to detect different structures of trees. For this, we treat the file-metadata tree as an undirected graph and use a graph representation learning approach. A graph embedding essentially transforms a whole graph into a low-dimensional vector to allow for the classification of graphs. We considered several graph embedding techniques for our task.[39][41] Our tests revealed that these approaches perform very similarly. (We defer further discussion of this to Sec. VII) Based on our evaluation, we selected the local degree profile (LDP) [39] representation scheme which yields a fixed embedding for the same graph, thereby avoiding the need for error-prone clustering. This representation essentially characterizes each node and its 1-hop neighbors through five degree statistics. The empirical distributions obtained by aggregating these statistics over all nodes in a graph are then concatenated into a feature vector. This resulting representation is used as a basis for differentiating file-metadata trees, where each distinct embedding is treated as a separate metaclass.

2) Tree Topology with Node Labels: A more complete characterization requires also incorporating the labels of nodes in a tree. To obtain such a representation, we considered three graph embedding methods that can utilize node labels, including FEATHER-G[43], GeoScattering[44], and Graph2Vec[45]. To better capture the relationship between node labels, the textual node labels are also converted into a vector representation. For this purpose, a distributed vector representation based on the Word2Vec[46] CBOW model is trained by treating each root-to-leaf path comprising a sequence of node labels as a sentence[2]. These graph embeddings, however, yield slightly different representations for the same graph. Therefore, resulting embeddings need to be clustered together to identify metaclasses. For this purpose, the K-means clustering method is used, and the number of clusters is decided by using the Elbow method and Silhouette score. In our evaluation, we determined that GeoScattering embeddings perform slightly better. (Related aspects will be further discussed in Sec. VII).

3) Exact Tree Match: The last abstraction also takes into account the order of boxes in addition to tree topology and node labels by performing exact tree match. Essentially, the order of boxes in MP4-like files can be captured by invoking a depth-first search (DFS) from the leftmost node of the corresponding file-metadata tree. To exploit this, DFS-ordered node labels obtained after removal of the leaves of a file-metadata tree are organized as a sequence and cryptographically hashed to produce a value. Each hash value is then treated as an index to a separate metaclass.

1All graph embeddings are created by the Karate Club[42] software package using their default hyperparameter settings of the 1.2 release.
2The Word2Vec representations are generated using default settings of Gensim[47] implementation by setting the embedding size to 10.
In all the three settings, an important concern is the comprehensive coverage of metaclasses. Since some of the video classes have fewer video instances than others, it is likely that during training our method will not generate descriptors for all metaclasses. As a result of this incomplete view, corresponding metaclasses for some video files may not be identified. To account for such cases, we incorporate an unknown metaclass to each clustering setting at the first level. Hence during testing, if a file-metadata tree cannot be structurally matched to any of the known metaclasses, it is assigned to the unknown class.

C. Classification at Second Level

After file-metadata trees are grouped into metaclasses based on their structural properties, a more refined classification is performed. At this level, feature and parameter values are used to build separate machine learning models for each metaclass. This again requires mapping the information contained in file-metadata trees to a feature representation. For this, we use the track and type-aware representation introduced in [26] that creates a feature vector with each dimension holding occurrence frequency of root-to-node path descriptions and values of non-categorical values in the file-metadata tree as it yields more compact feature vectors. The resulting feature vectors are used in conjunction with a decision tree learner to perform multi-class classification.

D. Tackling Partial Video Files

When a video file is partially available, a condition encountered frequently during recovery of data from disks, memory, and SSDs using file carving tools, file metadata will be largely inaccessible. Especially for file formats that are mainly designed for video playback (as opposed to streaming), this is a major problem because metadata information comprise only a very small part of the overall file size, and they may not be located during file carving.

To make such video file fragments playable, [28] introduced a method for estimating encoding parameters through a search over parameter space by incorporating decoding error messages. Although the space of parameter values defined by the H.264 standard is extremely large, many parameters vary slightly across camera models and for several parameters choosing the highest or lowest categorical setting will not curtail decoding. This work essentially showed that a large number of videos can be decoded without having access to actual parameters by only determining a core set of 10 parameters in the SPS and PPS. Crucially, five of these parameters relate to how the bitstream is parsed and the rest determine the entropy coding mode, frame width, and base-quantization level.

Given this capability, we also investigate how reliably the source camera-model of a video can be identified using a subset of coding parameters. With this objective, we apply the method in [28] to coded frame data extracted from videos to estimate their coding parameters with one modification. In its operation, this method does not identify frame height as a core parameter because an incorrect setting causes the decoded frame to be either truncated or elongated through content repetition. To improve the distinguishability, we also incorporate the frame height information. For this, we changed the default value for frame height to the largest possible value which inadvertently caused interpolation of the last line of 8×8 picture blocks. The height is then estimated by determining the repetition point through the correlation of block content. The resulting 11 parameter values are then treated as a feature vector and used for model building directly.

VI. RESULTS

We now examine the effectiveness of the proposed source camera-model identification method. Our evaluation assumes a closed-set identification scenario. For this, we merged together VISION, ACID, SOCRatES, and EVA-7K datasets to obtain a large pool of video data. The former three datasets include videos captured by smartphones and digital cameras whereas the last one includes videos of various provenance downloaded from different social media platforms and edited by several tools. This resulted in 20,153 videos from 112 camera models of over 28 brands, 3 editing tools, and 4 social media platforms with 10-1,400 videos per video class. Our examination showed that except for those associated with eight camera models, all videos are captured in H.264-MP4 format. Out of these camera models, five models used H.264 encoding, one used the MP4 file container format, and the remaining two used neither as summarized in Table I. Since the resulting dataset is significantly unbalanced, in our performance evaluation we used the balanced-accuracy metric which evaluates the success of distinguishing each camera model at an equal footing.

| Camera Model | Datasets | Type of Container | Codec | Identifier in Header |
|--------------|----------|-------------------|-------|----------------------|
| Panasonic HC-V180 | ACID | MTS | H.264 | ✓ |
| JVC Everio R | ACID | MTS | H.264 | ✓ |
| Fujifilm Finepix X800 | ACID | AVI | mjpeg | ✓ |
| Sony Cyber-shot DSC-WX350 | ACID | AVI | mjpeg | ✓ |
| Nikon Coolpix X3100 | ACID | AVI | mjpeg | ✓ |
| Panasonic FX400 | ACID | MTS | H.264 | ✓ |
| Sony NEX VG10 | SOCRatES | MTS | H.264 | ✓ |
| Asus Zenfone 5 | SOCRatES | MIP4 | mjpeg | ✓ |

A. Comparison of Coding and Encapsulation Characteristics

We first quantify the level of distinguishability provided by encoding and encapsulation characteristics individually and together. In our evaluation, we use a flat classification approach similar to that proposed by earlier work [24], [25]. For this, we consider two file-metadata tree representations to determine the achievable identification accuracy, which will be used as a baseline. The first is the sparse vector representation based on

3In disagreement with its description, the SOCRatES dataset did not include videos captured by the following models: Wiko Rainbow Up 4G, LG G4, Apple iPhone 5, Sony Xperia E3, and Meizu M3 Note. In addition, videos associated with the Motorola Moto G could not be played. The dataset, however, included another instance of the same model.

We also excluded Adobe Premiere and ExiTool videos in the EVA-7K dataset from our combined dataset as both editing tools hardcode source information within the Extensible Metadata Platform (XMP) headers.
root-to-leaf path descriptions that combines two feature vectors [24], one obtained using an unordered list of path entries with only field names and the other using an unordered list including both fields and values (see Fig. 3). The second is the track and (field) type-aware representation introduced in [26] without feature selection and LDA-based feature dimension reduction. When using this representation all codec parameters are treated as non-categorical variables. To also assess the impact of metadata parsers, file-metadata trees are generated for all the videos using two public parser libraries [35], [37]. Feature vectors are then used to train a decision tree (DT) classifier with balanced class weight.

Table II provides balanced accuracy values obtained after five-fold cross validation with 100 runs to reduce any sampling bias. The first column denotes the library used for parsing MP4 metadata, the second one shows the method used for turning file-metadata trees into feature vectors, and the third column displays the source of metadata, i.e., whether based on container, codec, or the combination of the two. The balanced accuracy results obtained in identifying 119 video classes involving 20,153 videos are given in the fourth column. Tests are also repeated considering only MP4 formatted videos, which reduced our combined dataset to 112 classes with 18,456 videos, as presented in the last column.

Results show that when compared in isolation, codec parameters provide less distinguishability (line 9) than MP4 file metadata (lines 1, 3, 5, and 7). This can be attributed to the lower dimensional nature of codec parameters compared to the rich information contained in the MP4 file metadata. However, when combined together, the complementary nature of both information sources is revealed as in all test settings combined features yielded the best accuracy (lines 2, 4, 6, and 8). More specifically, the highest balanced accuracy achieved by either sparse [24] or track and type-aware [26] feature representation of container characteristics in the overall dataset is 84.3% (line 5). Whereas in the combined setting, balanced accuracy reaches 90.22% (line 4) with an improvement of 6% in accuracy. Even when limiting the analysis to only MP4 formatted videos, classification accuracy improves by 1.4%, from 89.02% (line 5) to 90.42% (line 4). Overall, these results indicate that the encoding aspect of a video file provides improved discrimination capability.

Our findings also demonstrate that the more compact track and type-aware feature representation, in general, yields better performance than the sparse representation. Further, results show that the metadata extracted by both parsers are indeed different. Comparatively, the GPAC parser is more effective for both feature representations. In the combined setting, however, MP4 Parser is found to yield the best performance. One factor contributing to this result may be the incorporation of SPS and PPS strings extracted by the GPAC parser (which are disregarded by MP4 Parser) with individual parameter values which might be confusing the DT classifier.

Table III presents five-fold cross validation results averaged over 50 randomizations of the training and testing datasets (lines 1-3). That is, in each run, we used 80% of the videos to identify metaclasses and to train several DT classifiers, while the remaining 20% were used to measure the balanced identification accuracy of the overall system. The first column of the table shows the method used for clustering file-metadata trees in the first layer, and the following three columns present results of the first-level classification obtained on the whole dataset. The total number of metaclasses generated by each abstraction is given in the second column. The relatively low number of metaclasses created by the three abstractions validates our intuition that file-metadata trees are in fact structurally very similar and do not exhibit strong variation among videos. The percentage of file-metadata trees assigned to the unknown class (third column) further verifies that structures associated with each metaclass are quite invariant. It is determined that except for the second abstraction, on average only 0.1% of videos are assigned to the unknown class. (When clustering GeoScattering embeddings, we used an Euclidean distance of 3.5 as a threshold for assigning a video to the unknown class.) The error in assigning a file-metadata tree to an incorrect metaclass at the first level is given in the fourth column, and it is found to be extremely small in all cases.

Table IV presents five-fold cross validation results averaged over 50 randomizations of the training and testing datasets (lines 1-3). That is, in each run, we used 80% of the videos to identify metaclasses and to train several DT classifiers, while the remaining 20% were used to measure the balanced identification accuracy of the overall system. The first column of the table shows the method used for clustering file-metadata trees in the first layer, and the following three columns present results of the first-level classification obtained on the whole dataset. The total number of metaclasses generated by each abstraction is given in the second column. The relatively low number of metaclasses created by the three abstractions validates our intuition that file-metadata trees are in fact structurally very similar and do not exhibit strong variation among videos. The percentage of file-metadata trees assigned to the unknown class (third column) further verifies that structures associated with each metaclass are quite invariant. It is determined that except for the second abstraction, on average only 0.1% of videos are assigned to the unknown class. (When clustering GeoScattering embeddings, we used an Euclidean distance of 3.5 as a threshold for assigning a video to the unknown class.) The error in assigning a file-metadata tree to an incorrect metaclass at the first level is given in the fourth column, and it is found to be extremely small in all cases.

Camera-model identification accuracy values obtained at the second level using the three abstractions are given in the last two columns considering both the whole dataset (119 classes)
and its reduced version (112 classes). Overall, results show that the three abstractions yield similar performance, but the use of tree-hashing at the first level yields the best performance. Compared to the best results obtained using flat classification (Table II), as repeated in lines 4 & 5, it is determined that the hierarchical classification method provides an additional 0.6% improvement in balanced accuracy, from 90.2% to 90.8%.

### Table III

**Camera-Model Identification Accuracy with Two-Level Hierarchical Classifier**

| Two-Level Hierarchical Classification Method | Balanced Accuracy | Incorrect Classification (%) | All 1 MP4 |
|--------------------------------------------|-------------------|-------------------------------|----------|
| LDP                                        | 84.30             | 164                           | 90.84    | 91.12 |
| GeoScattering                              | 90.22             | 90.42                         | 90.84    | 91.14 |
| Flat (Leaf Only)                           | 84.50             | 90.74                         | 90.85    | 91.10 |
| Flat (Leaf Only and Codec)                 | 84.50             | 90.74                         | 90.85    | 91.10 |

### C. Comparison with Other Methods

We compare our proposed method with the approaches of three methods that utilize file metadata for video source identification [24]–[26]. Since public implementations for all these methods were not available, we implemented them to the best of our ability and applied them to our context. In [24], Yang et al. focused on the video integrity verification problem while proposing an improvement over [23] which used the likelihood ratio of ordered root-to-leaf paths (including field-value pairs) to perform brand identification and integrity verification. This work reported that the use of the sparse feature representation obtained by combining the above path entries with unordered root-to-leaf paths with only field names in conjunction with a DT classifier yields higher accuracy. This method uses the public MP4 Parser library, and its feature extractors are available. In this regard, the results given in the first line of Table II (77.8% and 82.0%) correspond to achievable accuracy by this method.

Gelbing et al. [25] introduced the insertion/deletion resistant ordering of root-to-node path elements and used a weighting-based matching to evaluate the similarity of file metadata across brands and unique devices. Accordingly, the weight for each path entry is determined based on the extent to which they contribute to the identification of each model. The authors also proposed blocking a list of boxes because they cannot be used for attribution. Our implementation, excluding those boxes, yielded an area under the curve (AUC) of 91.1% for device identification on the VISION dataset which is very close to the 90.3% value reported in the paper. However, the computation of the AUC metric allows matching a video close to the 90.3% value found in the paper. When implementing this method, we used MP4 Parser to obtain the track and type-aware feature vector as it yielded better results than the GPAC parser. To determine the impact of using the public parser (as opposed to the deployed custom parser) on performance, we validated our implementation on a closed-set brand-attribution setting involving the VISION dataset as considered in the paper. This yielded an identification accuracy of 99.77% which is very close to the 99.8% value found in the paper. When performing feature selection and LDA, we considered all clusters of size up to 3K features which at the best case yielded a balanced accuracy of 74.5% on the overall dataset. We believe reducing the feature vector size drastically may not be appropriate when the number of video classes is large. Therefore, we used track and type-aware feature vectors as part of a DT classifier, without dimension reduction, which resulted in 84.1% balanced accuracy as given in Table II (line 7) implying that a DT is more effective in identifying critical features.

Overall these results demonstrate that the incorporation of encoding parameters with the conventional MP4 based file metadata in a multi-level classification approach improves the achievable camera-model identification accuracy by almost 6.5%.

### D. Exclusion of User Adjustable Settings

Many cameras as well as camera apps of smartphones allow users to choose among a set of resolutions, frames-per-second, and a compression quality when capturing a video. Since these are user set parameters, comprehensive coverage of all possible settings associated with a camera model may not be possible. Therefore, we investigate the extent to which these settings affect the identification accuracy. For this purpose 7 parameters in SPS and 24 fields in MP4 boxes are excluded when mapping file-metadata trees to feature vectors. Repeating the same training and test procedure described in Sec. VI-B corresponding identification accuracy values in the third row of Table III are, respectively, obtained as 90.85% and 91.10%. The fact that almost the same performance could be obtained indicates that video file metadata contains rich model-specific information and that this diversity can compensate for the lack of considered user adjustable settings.

### E. Partial Video Files

This test setting concerns the extreme case where only a minimal subset of encoding parameters are available for camera-model identification. For this, the automatic header

---

1It is reported that in this test scenario the method achieves 99% accuracy in identifying two brands while getting 100% accuracy on all others. This corresponds to a balanced brand classification accuracy of 99.8%.
generation method of [28] is used to estimate the values of the 10 core parameters along with the picture height from original (unedited) videos captured by a camera. We must note here that changing the values of encoding parameters does not necessarily cause a decoding failure. As an example, a video encoded with the frame cropping flag set to 0 can still be decoded when the cropping offset is set to 0 while the cropping flag is set to 1. Therefore, the estimated parameters that allow decoding of the video data may not necessarily be the same as those in the original file. In fact, 6 of the 11 parameters, including the base quantization value, cropping (flag and offset), picture order count, number of frames, and height, can potentially take alternative values without a failure in decoding.

We used the first coded frame of 12,892 H.264 video captured by 109 camera models to estimate the 11 parameters that allowed successful decoding of the picture. The comparison of estimated parameter values from all videos with their original values revealed that around 80% of cropping and quantization parameters are the same as their original values. Similarly, around 70% of the values corresponding to picture order count and the number of frames parameters were found to be the same. In the case of picture height, the average error across all videos is measured to be 34.5 pixels with 61% of the height values having an error less than 10 pixels.

These estimated parameters are then used as feature vectors to build a DT classifier which resulted in a balanced accuracy of 57.2%. Although this marks a considerable drop from the results of Table [III] (from 91% to 57%), it is surprising to determine that only 11 estimated parameters can achieve this level of discrimination among 109 camera-models. To further test the distinctiveness of these parameter values, we developed another classifier for brand-level identification by combining together videos of all models of a brand as a single class. For this task, the achievable accuracy is found to be 81%. Similar to the previous setting, we also performed the test by disregarding four user-adjustable parameters that relate to frame height and width. As expected, this caused a further decrease in performance, and 39.7% and 69.2% balanced accuracy results were obtained for both scenarios.

VII. DISCUSSION

In the presence of thousands of camera models, the most important challenge for camera-model identification methods is scalability. This requires a comprehensive approach that combines several sources of information, including those obtained from photos and videos through both content and file-metadata analysis. Camera-model identification has long been the goal of content analysis methods, more specifically in the context of photographic images. Accordingly, the highest performance has so far been reported by data-driven methods that use deep learning frameworks to learn features or as part of an end-to-end classification approach [19], [29], [48], [49]. In this respect, when tested on images of 54 camera models in DRESDEN and VISION image datasets, the method of [29] achieved 83% classification accuracy. In comparison to these results, our findings show that file-metadata based camera-model identification provides an important complement to existing content-based analysis methods.

At its core, our approach views encoding and encapsulation as two complementary and independent aspects. Therefore, other representations for container-based file metadata can be easily incorporated with the codec parameters to perform the hierarchical classification. Similarly, the use of more advanced parsers that can extract non-standard, manufacturer-specific information will further improve the reported accuracy values as our method builds on these capabilities.

Application of neural network models. As an alternative to our current approach, we also considered recurrent neural networks to exploit the sequential nature of the file metadata. Since in its original form, file metadata appears as a sequence of boxes along with their fields and values, we also applied GRU/LSTM [50], [51] models by posing the problem as a sequence classification task. Our results showed that these sequential models could not sufficiently capture the inherent tree-structure of the data and yield inferior results. This may simply be attributed to the limited amount of video data available in some video classes (10 videos), and the effectiveness of the DT classifier in identifying clear decision boundaries for our mostly categorical features.

We further tested some of the unsupervised graph learning methods on our dataset. These whole graph embedding approaches allow embedding of file-metadata trees (including all branch nodes and leaf nodes containing values) to a vector space which can then be used for creating a classification model. In this regard, the well-known Graph2Vec [45] and GL2Vec [52] graph representations, coupled with the KNN classifier, achieved relatively low balanced accuracy values of around 67%. In contrast, more recently introduced FEATHER-G and GeoScattering graph embedding methods achieved a performance slightly above 85%. This finding indicates that in the presence of larger datasets, graph learning methods have strong applicability to the model-identification task.

We also examined the distinctiveness inherent to the shape of file-metadata trees by disregarding the node labels and values. For this purpose, we performed topology based classification of file-metadata trees using LDP [59], NOG [40], Slq-VNGE, and Slq-LSD [41] graph embeddings which overall yielded balanced accuracy values in the range of 45-56% with LDP and Slq performing the best.

Best achievable performance on our dataset. To better understand the room for improvement on our dataset while using the two parsing tools and codec parameters, we compared the extracted metadata from all videos in a pair-wise manner. This side-by-side comparison excluded any metadata related to video content that were eliminated during previous tests. Over 119 video classes, we identified 10 tuples of video classes, including nine pairs and one triplet, that have exactly the same features. This essentially indicates that a confusion among 21 video classes is unavoidable, and only 10 of these video classes can be correctly distinguished. In other words, when all remaining videos are matched correctly to their classes, the best achievable balanced accuracy will be limited to 90.8% with only 108 out of the 119 classes being identified accurately. In the case of MP4 formatted videos, the best
achievable accuracy would be 91.1% due to collision among 10 video classes out of the 112. Overall, these results show that our proposed method already achieves a perfect discrimination on this dataset. It can be noticed that our accuracy values are marginally better than these performance bounds. The slight difference of 0.04% is essentially due to the randomness in how data is partitioned into folds and the resulting minute variations in class weights which cause different classes of each confused tuple to be selected in different folds.

**Robustness issue.** Another important issue is the robustness of file-metadata-based characteristics to video editing or tampering. Essentially, any video file can be repackaged in one of several video file formats without ever modifying the video content, thereby removing the majority of encapsulation-related characteristics. In this regard, results of [24], [26], [53] indicate that existing tools and systems do not exhibit such behavior, and in most cases preserve most of the metadata in the original file after editing. Ultimately, however, modifying the encoding parameters will require re-encoding of the original video. That is, a change in the encoding parameters will introduce double-compression artifacts which can possibly be identified through content-based analysis methods. This essentially provides a mechanism to determine whether a given set of encoding parameters should be used for camera-model identification.

**Difficulty of datasets.** Facilitating further research in this area requires more comprehensive media sets that contain a larger variety of camera models. In this regard, our collection includes the largest dataset used in such a study. When evaluated individually, our method yields 91%, 97%, 87%, and 100% balanced accuracy values, respectively, on VISION, ACID, SOCRateS, and EVA-7K datasets. These results show that among the four datasets EVA-7K and ACID pose the least difficulty to video-class identification. In contrast, SOCRateS is the most challenging dataset primarily due to the large number of classes it includes and few number of video samples in each class.

**REFERENCES**

[1] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, “Deepfakes and beyond: A survey of face manipulation and fake detection,” Information Fusion, vol. 64, pp. 131–148, 2020.

[2] L. Verdoliva, “Media forensics and deepfakes: an overview,” IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 910–932, 2020.

[3] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, “Determining image origin and integrity using sensor noise,” IEEE Transactions on information forensics and security, vol. 3, no. 1, pp. 74–90, 2008.

[4] A. E. Dirik, H. T. Sencar, and N. Memon, “Digital single lens reflex camera identification from traces of sensor dust,” IEEE Transactions on Information Forensics and Security, vol. 3, no. 3, pp. 530–532, 2008.

[5] A. T. Ho and S. Li, “Forensic camera model identification,” 2015.

[6] P. R. M. Júnior, L. Bondi, P. Bestagini, S. Tubaro, and A. Rocha, “An in-depth study on open-set camera model identification,” IEEE Access, vol. 7, pp. 180713–180726, 2019.

[7] I. Amerini, T. Uricchio, and R. Caldelli, “Tracing images back to their social network of origin: A cnn-based approach,” in 2017 IEEE Workshop on Information Forensics and Security (WIFS). IEEE, 2017, pp. 1–6.

[8] I. Amerini, R. Caldelli, A. Del Mastio, A. Di Fuccia, C. Molinari, and A. P. Rizzo, “Dealing with video source identification in social networks,” signal processing: Image communication, vol. 57, pp. 1–7, 2017.

[9] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, “Do gans leave artificial fingerprints?” in 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR). IEEE, 2019, pp. 506–511.

[10] N. Yu, L. S. Davis, and M. Fritz, “Attributing fake images to gans: Learning and analyzing gan fingerprints,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7556–7566.

[11] M. Kirchner, “Efficient estimation of cfa pattern configuration in digital camera images,” in Media Forensics and Security II, vol. 7541. International Society for Optics and Photonics, 2010, p. 754111.

[12] C.-H. Choi, J.-H. Choi, and H.-K. Lee, “Cfa pattern identification of digital cameras using intermediate value counting,” in Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security, 2011, pp. 21–26.

[13] S. Bayram, H. T. Sencar, N. Memon, and I. Avicbas, “Improvements on source camera-model identification based on cfa interpolation,” Proc. of WG, vol. 11, no. 9, 2006.

[14] A. Swaminathan, M. Wu, and K. R. Liu, “Nonintrusive component forensics of visual sensors using output images,” IEEE Transactions on Information Forensics and Security, vol. 2, no. 1, pp. 91–106, 2007.

[15] C. Chen and M. C. Stamm, “Camera model identification framework using an ensemble of demosaicing features,” in 2015 IEEE international workshop on information forensics and security (WIFS). IEEE, 2015, pp. 1–6.

[16] K. San Choi, E. Y. Lam, and K. K. Wong, “Automatic source camera identification using the intrinsic lens radial distortion,” Optics express, vol. 14, no. 24, pp. 11 551–11 565, 2006.

[17] L. T. Van, S. Emmanuel, and M. S. Kankanahalli, “Identifying source cell phone using chromatic aberration,” in 2007 IEEE International Conference on Multimedia and Expo. IEEE, 2007, pp. 883–886.

[18] J. Yu, S. Craver, and E. Li, “Toward the identification of dslr lenses by chromatic aberration,” in Media Watermarking, Security, and Forensics III, vol. 7880. International Society for Optics and Photonics, 2011, p. 788010.

[19] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, “First steps toward camera model identification with convolutional neural networks,” IEEE Signal Processing Letters, vol. 24, no. 3, pp. 259–263, 2016.

[20] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, “A study of co-occurrence based local features for camera model identification,” Multimedia Tools and Applications, vol. 76, no. 4, pp. 4765–4781, 2017.

[21] T. Gloe, A. Fischer, and M. Kirchner, “Forensic analysis of video file formats,” Digital Investigation, vol. 11, pp. S68–S76, 2014.

[22] J. Song, K. Lee, W. Y. Lee, and H. Lee, “Integrity verification of the ordered data structures in manipulated video content,” Digital Investigation, vol. 18, pp. 1–7, 2016.

[23] M. Iuliani, D. Shullani, M. Fontani, S. Meucci, and A. Piva, “A video forensic framework for the unsupervised analysis of mp4-like file containers,” IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 635–645, 2018.
[24] D. Shullani, M. Iuliani, D. Shullani, R. Ni, Y. Zhao, and A. Piva, “Efficient video integrity analysis through container characterization,” IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 947–954, 2020.

[25] E. Gelbinger, L. Würsching, S. Znadzinski, and M. Steinebach, “Video source identification from mp4 data based on field values in atom/box attributes,” Electronic Imaging, vol. 2021, no. 4, pp. 331–1, 2021.

[26] Z. Xiang, J. Horváth, S. Baireddy, P. Bestagini, S. Tubaro, and E. J. Delp, “Forensic analysis of video files using metadata,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1042–1051.

[27] “Video developer survey 2019,” https://go.bitmovin.com/video-developer-survey-2019 (Accessed on 08/29/2021).

[28] E. Altinisik and H. T. Sencar, “Automatic generation of h. 264 parameter sets to recover video file fragments,” IEEE Transactions on information forensics and security, 2021.

[29] D. Shullani, M. Fontani, M. Iuliani, O. Al Shaya, and A. Piva, “Vision: a video and image dataset for source identification,” EURASIP Journal on Information Security, vol. 2017, no. 1, pp. 1–16, 2017.

[30] B. C. Hosler, X. Zhao, O. Mayer, C. Chen, J. A. Shackleford, and M. C. Stamm, “The video authentication and camera identification database: A new database for video forensics,” IEEE Access, vol. 7, pp. 76937–76948, 2019.

[31] C. Galdi, F. Hartung, and J.-L. Dugelay, “Socrates: A database of realistic data for source camera recognition on smartphones.” in ICPRAM, 2019, pp. 648–655.

[32] “Iso - is0iec 14496-12:2008 - information technology - coding of audio-visual objects - part 12: Iso base media file format,” https://www.iso.org/standard/51533.html.

[33] C. Q. Huamán, A. L. S. Orozco, and L. J. G. Villalba, “Authentica- tion and integrity of smartphone videos through multimedia container structure analysis,” Future Generation Computer Systems, vol. 108, pp. 15–33, 2020.

[34] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Transactions on circuits and systems for video technology, vol. 13, no. 7, pp. 560–576, 2003.

[35] “sannies/mp4parser: A java api to read, write and create mp4 files,” https://github.com/sannies/mp4parser (Accessed on 10/23/2021).

[36] R. R. López, E. A. Luengo, A. L. S. Orozco, and L. J. G. Villalba, “Digital video source identification based on container’s structure analysis,” IEEE Access, vol. 8, pp. 36 363–36 375, 2020.

[37] “Gpac nightly builds — gpac,” https://gpac.wp.imt.fr/downloads/gpac-nightly-builds/ (Accessed on 10/23/2021).

[38] “Github - aizvorski/h264bitstream,” https://github.com/aizvorski/h264bitstream/ (Accessed on 04/07/2021).

[39] C. Cai and Y. Wang, “A simple yet effective baseline for non-attributed graph classification,” in Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM ‘20). ACM, 2020, p. 3125–3132.

[40] B. Rozemberczki, O. Kiss, and R. Sarkar, “Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs,” in Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM ‘20). ACM, 2020, p. 3125–3132.

[41] B. Rozemberczki and R. Sarkar, “Characteristics functions on graphs: Birds of a feather, from statistical descriptors to parametric models,” in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1325–1334.

[42] F. Gao, G. Wolf, and M. Hirn, “Geometric scattering for graph data analysis,” in International Conference on Machine Learning. PMLR, 2019, pp. 2122–2131.

[43] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaswal, “graph2vec: Learning distributed representations of graphs,” arXiv preprint arXiv:1707.05005, 2017.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[45] R. Rehurek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[46] D. Güera, F. Zhu, S. K. Yarlagadda, S. Tubaro, P. Bestagini, and E. J. Delp, “Reliability map estimation for cnn-based camera model attribu-