Forensic Analysis of Video Files Using Metadata

Ziyue Xiang* János Horváth* Sriram Baireddy*
Paolo Bestagini† Stefano Tubaro† Edward J. Delp*
* Video and Image Processing Lab (VIPER), School of Electrical and Computer Engineering,
Purdue University, West Lafayette, Indiana, USA
† Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy

Abstract

The unprecedented ease and ability to manipulate video content has led to a rapid spread of manipulated media. The availability of video editing tools greatly increased in recent years, allowing one to easily generate photo-realistic alterations. Such manipulations can leave traces in the metadata embedded in video files. This metadata information can be used to determine video manipulations, brand of video recording device, the type of video editing tool, and other important evidence. In this paper, we focus on the metadata contained in the popular MP4 video wrapper/container. We describe our method for metadata extractor that uses the MP4's tree structure. Our approach for analyzing the video metadata produces a more compact representation. We will describe how we construct features from the metadata and then use dimensionality reduction and nearest neighbor classification for forensic analysis of a video file. Our approach allows one to visually inspect the distribution of metadata features and make decisions. The experimental results confirm that the performance of our approach surpasses other methods.

1. Introduction

The proliferation of easy-to-use video manipulation tools has placed unprecedented power in the hands of individuals. Recently, an Indian politician used deepfake technology to rally more voters [22]. In the original video the politician delivered his message in English; it was convincingly altered to show him speaking in local dialect. Media manipulation methods are also used as tools of criticism and to undermine the reputation of politicians [12]. Such manipulated videos can now be easily generated to bolster disinformation campaigns and sway the public opinion on critical issues.

A wide variety of tools for video forensic analysis have been developed [26]. These tools can be used to attribute a video to the originating device, to reconstruct the past video compression history, and even to detect video manipulations. The most popular video manipulation detection techniques focus on inconsistencies and artifacts in the pixel domain [6, 24, 27, 29]. As video manipulation detection methods become more sophisticated, video editing techniques continue to improve, leading to a situation where manipulated videos are becoming practically indistinguishable from real videos [7, 8, 14, 28, 36, 38]. For this reason, detection techniques exploiting pixel-level analysis may fail, while methods that do not use pixel data will increasingly gain importance.

Video forensic techniques not exploiting pixel analysis typically work due to the presence of "metadata" [15, 20]. This is additional embedded information that every video file contains. The metadata are used for video decoding [21] and indicating other information such as the date, time, and location of the video when created. Because video editing tools tend to cause large structural changes in metadata, it is difficult for one to alter a video file without leaving any metadata traces. Therefore, metadata can serve as strong evidence in video forensics tasks.

In this paper, we leverage the seminal work presented in [20, 39] to investigate the use of metadata for video forensic analysis of the MP4 and MOV video formats, which are among the most popular video wrappers/containers. The MP4 format is used by numerous Android devices, social networks, and digital video cameras [13, 33, 41]. MOV format is mostly used by Apple devices and is derived from the same ISO standard as MP4 [18]. The design of the MP4 format is based on MOV [2]. The two formats can be parsed in a similar manner, thus we will refer to MP4 containers hereinafter even if MOV containers are considered. As a result, our approach can analyze a large number of videos in the real world.

In our work, we examine the results of using the metadata in MP4 files for video forensic scenarios, extending the work presented in [39]. More specifically, we describe a metadata extraction method and improve the feature representation format to make metadata-based forensic feature vectors more compact. We employed feature selection techniques to boost the quality of the feature vectors. Finally, we reduced the dimensionality of the feature vectors to two, which allows visualization and classification in 2D space. We show that

these feature vectors can be used for a wide variety of video forensic tasks, from source attribution to tampering detection. Compared to other work, our proposed approach can generate 2D feature scatter plots and decision boundary graphs for many video forensics tasks. This feature enables us to gain insights into the distribution of MP4 metadata and make interpretable decisions.

Our experimental results show that many video forensics problems on standard datasets can be solved reliably by looking only at metadata. We also discovered that videos uploaded to specific social networks (e.g., TikTok, WeiBo) present altered metadata, which invalidates metadata-based forensics methods. This is one of the limitations of our techniques and will be addressed in future research.

2. Related Work

Many techniques have been described to determine whether some part of a video has been tampered or not [26]. Most of these methods were developed to detect manipulations in the pixel domain and do not use metadata information. Compared to pixel-level analysis, metadata-based methods possess unique advantages. The size of metadata is significantly smaller than pixel data, which enables the analysis of large datasets in short amounts of time. Most video manipulation software do not allow users to alter metadata directly [15, 20]. Consequently, metadata has a higher degree of opacity than pixel data, which makes metadata-based media forensics more reliable and its corresponding attacks more challenging.

Most existing work focuses on the metadata in MP4-like video containers, which maintain data in a tree structure. In [20] and [39], the authors design features based on symbolic representation of MP4's tree structure, which are processed by a statistical decision framework and decision trees, respectively. The authors report high performance for both video authentication and attribution tasks. Güera *et al*. [15] extract video metadata with the `ffprobe` utility and then do video authentication with an ensemble classifier.

More low-level metadata-related information can be found by looking into video compression traces. Video compression methods typically leave series of traces related to the way the video frames are compressed. This information is not easy to modify, thus acting as a metadata-like feature. As an example, most contemporary video encoders compress frame sequences in a structures known as a Group of Pictures (GOP), where one frame can be defined using contents of other frames in order to save space [30]. The dependency between frames within or across different GOPs may provide evidence for video manipulation. Due to the complexity of video codecs, a number of techniques have been proposed for various settings of a codec where specific video encoding features are turned on or off. Vázquez-Padín *et al*. [37] provide a detailed explanation of the video encoding process and GOP

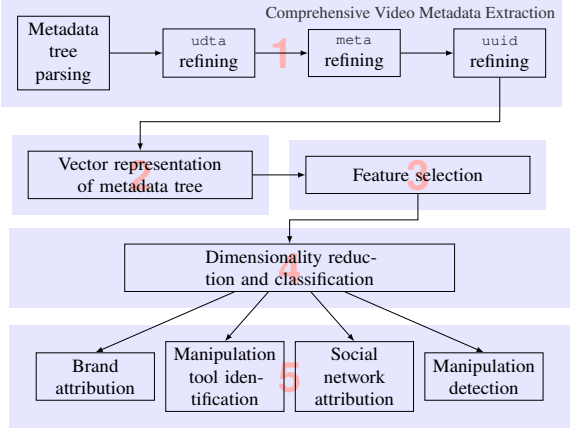

Figure 1: The structure of our proposed metadata forensic analysis technique.

structure. They propose a video authentication method that generalizes across multiple video encoding settings. Yao *et al*. [40] discuss the detection of double compression when an advanced video encoding feature called adaptive GOP is enabled.

3. Proposed Approach

3.1. An Overview of Our Approach

Video metadata captures multiple aspects of the history of a video file. In this paper we propose a framework that exploits an MP4 video file's metadata to solve multiple video forensics problems, including brand attribution, video editing tool identification, social network attribution, and manipulation detection. Our method can also be easily adapted to other forensics applications.

The structure of our proposed framework is illustrated in Figure 1. As will be discussed in Section 3.2, the MP4 format manages data using a tree structure. First we extract the metadata from MP4 files while preserving their relationships in the tree structure. The MP4 standard is around twenty years old, it contains numerous vendor-specific nuances that require separate parsing strategies. The metadata tree needs to go through several refining stages, which increase the granularity of the extracted information. In the next step, the tree representation of metadata is converted into a numeric feature vector, which can be easily processed by machine learning methods. Our feature representation scheme is based upon [39]. We improve the handling of media tracks and metadata fields that take on continuous values inside the tree. The resulting feature vectors preserve more characteristics of the videos, yet they tend to also be more compact. In the last stage, we use these features with a classifier based on the selected forensic application. In the following we provide additional details about each step of our proposed framework.

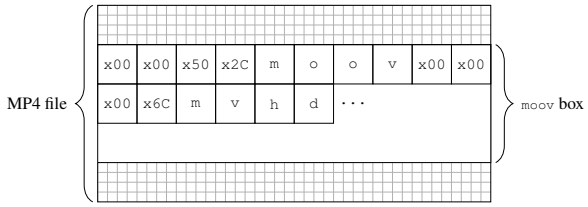

Figure 2: Illustration of the MP4 file format, where each cell represents one byte. An MP4 file is made up of a series of *boxes*. Every box has an 8-byte header, where the first 4 bytes store the size of the box in bytes (including the header) as a 32-bit big-endian integer, and the last 4 bytes store the name of the box. The content of a box is either child boxes or binary data. Binary data from different boxes may require distinct decoding methods.

3.2. Video Metadata

The first step in our approach consists of parsing metadata from video files. Digital video files are designed to handle multimodal data such as video, audio, and subtitles. The standards of these data modalities also shifts as technology advances. For example, since MP4's introduction in 2001, the mainstream video codec has changed from MPEG-2 to H.264, and may change to H.265 in the near future [17, 34, 35]. The metadata surrounding these various data modalities and standards are inserted into a video file in distinct ways. In this paper, we use the word *comprehensive* to describe a video metadata extraction scheme that is capable of parsing metadata across most data modalities and encoding specifications.

Figure 2 shows the basic structure of an MP4 file [19]. An MP4 file is composed by a number of *boxes*. Each box starts with a header that specifies the box size (in byte) and name. The size helps the user to locate box boundaries, and the name defines the purpose of the box. The content of a box can either be a child box or some encoded binary data. Thus, an MP4 file is effectively a tree structure, where information is stored on leaf nodes. Given MP4's tree structure, we can capture metadata information at two levels: (1) the structure of the tree; and (2) the interpreted values of binary data contained in leaf nodes. Therefore, the job of a metadata extractor is to traverse the MP4 tree, attain the tree's structure, filter non-metadata leaf nodes (e.g., nodes that contain video compressed pixel information) and interpret values on relevant leaves. As shown in Figure 3, the output of a metadata extractor can be represented without any loss by a collection of human-readable strings.

Our metadata extractor focuses on vendor-specific non-standard MP4 metadata boxes that are significant for forensic purposes. More precisely, we determine that `udta`, `uuid`, `meta`, and `ilst` boxes are likely to carry vital information for video forensics. We next discuss our strategies to refine the parsing process.

STRING 1
`moov`
MEANING
The `moov` box is present in MP4 file.

STRING 2
`moov/mvhd`
MEANING
The `mvhd` box is present in MP4 file. It is a child box of `moov`. One can also imply that the `moov` node itself must not contain any binary data.

STRING 3
`moov/mvhd/@duration=1546737`
MEANING
The `mvhd` box is a leaf node in the tree; it contains a field called `duration`, whose value is 1546737.

STRING 4
`moov/udta/@`[A9]`mod=`[00][09]`&`[81]`iPhone 5c`
MEANING
The `udta` box is a leaf node in the tree; it contains a field called [A9]`mod`, whose value is [00][09]`&`[81]`iPhone 5c`.

Figure 3: Examples of representing MP4 metadata tree with strings. Node paths are separated with '/', the values of leaf nodes are prefixed with '@', non-ASCII and unprintable characters are shown as hexadecimal codes surrounded by black frames. The metadata tree of any MP4 file can be portrayed by a collection of such strings.

3.2.1 Parsing `ilst` Data

`ilst` ("metadata item list") boxes in MP4 files are used to store vendor-specific metadata [3]. Generally speaking, `ilst` boxes are container boxes whose child boxes carry metadata items as key-value pairs. The names of `ilst`'s children (i.e., the *keys*) would start with [A9] (equivalent to character '©'). A list of frequently used `ilst` keys is shown in Table 1. One can see that the content of the `ilst` box is particularly important for forensic analysis, for it often contains information about the manufacturer of the video capturing device, the encoder version, and the location and time of the capture.

It is difficult to parse the `ilst` box because various device manufacturers employ vastly different approaches when using it. Below, we report some interesting variants we found during our experiments:

- `ilst`'s child boxes directly placed in `moov/udta`

  In some old Apple devices (e.g., iPhone 4, iPhone 5c, iPad 2), the child boxes of `ilst` are placed directly in `moov/udta` box.
- `ilst` boxes in `moov/meta`

  As its name suggests, the `meta` box is used to store metadata.

Table 1: A list of common keys in `ilst` boxes [4].

Key	Description
[A9]`mod`	camera model
[A9]`too`	name and version of encoding tool
[A9]`nam`	title of the content
[A9]`swr`	name and version number of creation software
[A9]`swf`	name of creator
[A9]`day`	timestamp of the video
[A9]`xyz`	geographic location of the video

In this case, the `meta` box behaves similarly as other standard boxes, which means it can be parsed normally. As the MP4 parser traverses the box, it will eventually reach `ilst` and its children.

- `ilst` boxes in `moov/udta/meta` and `moov/trak/meta`

  When `meta` boxes appear in `udta` and `trak` boxes, they deviate from standard boxes. More specifically, 4 extra bytes are inserted right after the `meta` header to store information [25]. These types of `meta` boxes cannot be parsed by the MP4 parser, normally because the program will see these 4 bytes as the size of next box, which will lead to corrupted results.

Our comprehensive metadata extractor is able to distinguish between these three scenarios and process MP4 video files correctly by fine-tuning the parsing of `udta` and `meta` boxes.

3.2.2 Parsing XML Data

We concluded that many video files contain XML data after inspecting numerous video files, especially those edited by ExifTool and Adobe Premiere Pro. These tools make use of the Extensible Metadata Platform (XMP) to enhance metadata management, which introduces a large amount of metadata inside an MP4 file's `uuid` and `udta` boxes in the form of XML. In Figure 4, we show two XML examples extracted from MP4 containers. It can be seen that these XML data can potentially contain a large amount of information, which includes type of manipulation, original value before manipulation, and even traces to locate the person that applied the manipulation. It is vital for our metadata extractor to have the ability to handle XML data inside MP4 files.

To avoid over-complicating the extracted metadata tree, we discard the tree structure of XML elements and flatten them into a collection of key-value pairs. If there is a collision between key names, the older value will be overwritten by the newer one, which indicates that only the last occurring value of each key is preserved. Despite the fact that some information is lost by doing so, the data we have extracted is likely to be more than enough for most automated video forensic applications.

3.3. Track and Type Aware Feature

The second step of our approach consists of turning the parsed metadata into feature vectors. Most machine learning methods use vectors as the input. The string representation of metadata trees generated in the previous step needs to be transformed into feature vectors before being used by machine learning methods.

Our feature representation technique is shown in Figure 5. For feature vectors to contain sufficient information of the MP4 tree, they need to include two levels of details: structure of the tree and value of metadata fields. Metadata can be either categorical or continuous numerical fields. Considering categorical values, we assign each node and metadata key-value pair in the MP4 tree an element in the feature vector, which counts the number of occurrences of that node or pair. This strategy preserves information about the MP4 tree in the feature vector to a great extent. Considering numerical values, creating a new element for each of these key-value pairs will render the feature vectors large and redundant. We decide to insert the values into the feature vectors directly.

From Figure 3, we know that string representation can be put into two categories: (1) strings that indicate the presence of a node, with node path separated by '/'; and (2) strings that show the key-value pair stored in a node, with node path separated by '/' and key-value separated by '='. Since an MP4 file can be seen as a collection of such strings, the feature transformation process can be viewed as a mapping from a given collection of strings S to a vector $\boldsymbol{v}$. For the discussion below, we use $\boldsymbol{v}_{[l]}$ to denote the l-th element of $\boldsymbol{v}$.

In order to construct a mapping $S \rightarrow \boldsymbol{v}$, we need to consider the set of all possible strings Ω. We denote the set of all category (1) strings and category (2) strings by $C_{(1)}$ and $C_{(2)}$, respectively. By definition, $C_{(1)}$ and $C_{(2)}$ form a partition of Ω. We assume that both $C_{(1)}$ and $C_{(2)}$ are ordered so that we can query each element by its index. Let us denote the l-th elements of $C_{(1)}$ and $C_{(2)}$ by $C_{(1)}[l]$ and $C_{(2)}[l]$, respectively.

Each category (1) string corresponds to an element in $\boldsymbol{v}$. For the i-th category (1) string, we denote the index of the corresponding vector element in $\boldsymbol{v}$ by $\chi_{(1)}(i)$. The value corresponding to this element is given by

$$\boldsymbol{v}_{[\chi_{(1)}(i)]} = \text{number of times } C_{(1)}[i] \text{ occurrs in } S, \quad \forall i \in \left\{1, 2, \ldots, \left|C_{(1)}\right|\right\}. \quad (1)$$

We treat each media track (`trak`) segment in the metadata strings in a different way. In the MP4 file format, each `trak` node contains information about an individual track managed by the file container. We observed that the number of tracks and content of the tracks remain consistent among devices of the same model. The structure of an MP4 file can be better preserved if we distinguish different tracks rather than merging them. This is achieved by assigning each track a track number in the metadata extraction phase. For example, the first track will be `moov/trak1`, and the second track will be `moov/trak2`. As a result, the child nodes and key-value pairs of each track will be separated, which effectively makes the feature vectors *track-aware*.

For category (2) strings that represent key-value pairs stored in a node, we applied a slightly different transformation strategy. We observed that some fields in MP4 files are essentially *continuous* (e.g., `@avgBitrate`, `@width`). Despite the fact that most MP4 metadata fields are discrete, assigning each combination of these continuous key-value pairs a new

```
<x:xmpmeta xmlns:x=3D"adobe:ns:meta/" x:xmptk=3D"Adobe XMP Core
↪  5.6-c145 79=
.163499, 2018/08/13-16:40:22        ">
 <rdf:RDF xmlns:rdf=3D"http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about=3D""
   xmp:CreateDate=3D"2019-11-19T22:47:45+01:00"
   xmp:ModifyDate=3D"2019-11-19T22:47:45+01:00"
   xmp:MetadataDate=3D"2019-11-19T22:47:45+01:00"
   xmp:CreatorTool=3D"Adobe Premiere Pro CC 2019.0 (Windows)"
   dc:format=3D"H.264">
   <creatorAtom:windowsAtom
    creatorAtom:extension=3D".prproj"
    creatorAtom:invocationFlags=3D"/L"
    creatorAtom:uncProjectPath=3D"\\?\C:\Users\pengpeng\Documents\Ado
   ↪  be\Pre=
miere Pro\13.0\premcut.prproj"/>
  </rdf:RDF>
</x:xmpmeta>
```


(a) Excerpt of XML data from a video processed by Adobe Premiere Pro. It clearly contains multiple important timestamps, the software name and version, and even the path to the Premiere project.


```
<x:xmpmeta xmlns:x=3D'adobe:ns:meta/' x:xmptk=3D'Image::ExifTool
↪  11.37'>
<rdf:RDF xmlns:rdf=3D'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>

 <rdf:Description rdf:about=3D''
  xmlns:exif=3D'http://ns.adobe.com/exif/1.0/'>
  <exif:DateTimeOriginal>1986-11-05T12:00:00</exif:DateTimeOriginal>
 </rdf:Description>

 <rdf:Description rdf:about=3D''
  xmlns:xmp=3D'http://ns.adobe.com/xap/1.0/'>
  <xmp:CreateDate>1986-11-05T12:00:00</xmp:CreateDate>
  <xmp:ModifyDate>1986-11-05T12:00:00</xmp:ModifyDate>
 </rdf:Description>
</rdf:RDF>
</x:xmpmeta>
```


(b) XML data from a video modified by ExifTool. It can be seen that the version of ExifTool is 11.37; the presence of `exif:DateTimeOriginal` implies the date of the video is modified.

Figure 4: Examples of XML data in MP4 video containers.

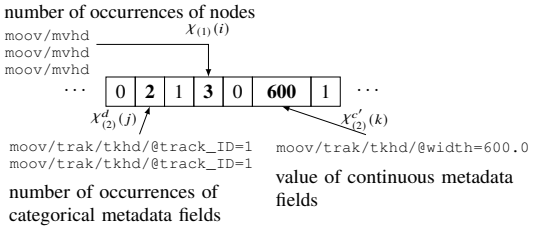

Figure 5: Illustration of the vector representation of MP4 metadata. The χ functions help determine the corresponding element of a node or a metadata field in the feature vector.

element in $\boldsymbol{v}$ will still result in large and redundant feature vectors. We continue to subdivide $C_{(2)}$ based on the *type* of each field, where the set of strings that have discrete fields is denoted by $C^{d}_{(2)}$, and the set of strings that have continuous fields is denoted by $C^{c}_{(2)}$. For strings that belong to $C^{d}_{(2)}$, the transformation scheme is similar to that of category (1) strings. Let the vector element index of the j-th string in $C^{d}_{(2)}$ be $\chi^{d}_{(2)}(j)$, then the value corresponding to the element is given by

$$\boldsymbol{v}\left[\chi^{d}_{(2)}(j)\right] = \text{number of times } C^{d}_{(2)}[j] \text{ occurs in } S, \quad \forall j \in \left\{1, 2, \ldots, \left|C^{d}_{(2)}\right|\right\}. \quad (2)$$

As for strings that belong to $C^{c}_{(2)}$, we first discard the values in the strings to form a set of distinct keys $C^{c'}_{(2)}$, and then put the values in $\boldsymbol{v}$ directly. For the k-th string in $C^{c'}_{(2)}$, the index of the corresponding vector element in $\boldsymbol{v}$ is $\chi^{c'}_{(2)}(j)$, and the value of the element is

$$\boldsymbol{v}\left[\chi^{c'}_{(2)}(k)\right] = \begin{cases} \text{the value of key } C^{c'}_{(2)}[k] \text{ in } S \\ 0 \text{ (if the key } C^{c'}_{(2)}[k] \text{ is not in } S) \end{cases}, \quad \forall j \in \left\{1, 2, \ldots, \left|C^{c'}_{(2)}\right|\right\}. \quad (3)$$

It can be seen that the dimensionality of $\boldsymbol{v}$ is given by

$$\dim(\boldsymbol{v}) = \left|C_{(1)}\right| + \left|C^{d}_{(2)}\right| + \left|C^{c'}_{(2)}\right|. \quad (4)$$

In general, the actual value of the index functions χ can be arbitrary as long as all values of $\chi_{(1)}(i)$, $\chi^{d}_{(2)}(j)$, and $\chi^{c'}_{(2)}(k)$ form a valid partition of integers in the range $[1, \dim(\boldsymbol{v})]$.

By using different representation strategies for discrete and continuous fields (i.e., being *type-aware*), the resulting feature vectors are more compact and suited to machine learning techniques.

3.4. Feature Selection

Our third step consists of reducing the set of selected features by discarding redundant features. Based on the feature extraction scheme above, it can be observed that some elements in the feature vector are significantly correlated. For example, the presence of string `moov/mvhd/@duration=1546737` in a collection S extracted from a valid MP4 file must lead to the presence of `moov/mvhd` in S. Therefore, feature selection can reduce redundancy within the feature vectors.

In the feature selection step, we reduce the redundancy among features that are strongly correlated. Since only a small proportion of elements in the feature vector $\boldsymbol{v}$ are inserted from continuous fields, most elements in $\boldsymbol{v}$ correspond to the number of occurrences of a node or a field value. If two features in $\boldsymbol{v}$ are negatively correlated, then it often means the presence of an MP4 entity implies the absence of another MP4 entity. In forensic scenarios, presence is much stronger evidence than absence. Therefore, if we only focus on features that are positively correlated, then we can select features of higher forensic significance.

Given a set of feature vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N$, we can compute the corresponding correlation matrix $\boldsymbol{R}$, where $\boldsymbol{R}_{ij}$ is equal to the correlation coefficient between i-th and j-th feature in the set. Then, we set all negative elements in $\boldsymbol{R}$ to zero, which results in matrix $\boldsymbol{R}^{+}$. That is, negative correlation is ignored. Because all elements in $\boldsymbol{R}^{+}$ are within the range $[0, 1]$, the matrix $\boldsymbol{R}^{+}$ can be seen as an affinity matrix between $\dim(\boldsymbol{v})$ vertices in a graph, where an affinity value of 0 indicates no connection and an affinity value of 0 indicates strongest connection. This allow us to use spectral clustering with α clusters on $\boldsymbol{R}^{+}$, which assigns multiple strongly correlated features into the same cluster [31]. Then, we select clusters with more than β features. For

each selected cluster, we retain only one feature at random. Here, $\alpha > \beta > 0$ are hyperparameters. This feature selection step helps improve the quality of feature vectors.

3.5. Dimensionality Reduction and Classification

In the last step, depending on the the video forensics problem, we use the feature vectors for classification in two ways.

Multi-class problems When the classification problem is multi-class, we use linear discriminant analysis (LDA) [11] to drastically reduce the dimensionality of feature vectors to 2 dimensions. LDA is a supervised dimensionality reduction technique. For a classification problem of K classes, LDA generates an optimal feature representation in a subspace that has dimensionality of at most $K - 1$. The new features in the subspace are optimal in the sense that the variance between projected class centroids are maximized. For multi-class classification problems, we always reduce the dimensionality of the feature vector to 2. After dimensionality reduction, we use a nearest neighbor classifier that uses the distance between the query sample and λ nearest samples to make a decision, where λ is a hyperparameter. Each nearest sample is weighted by their distance to the query sample. The use of the dimensionality reduction and nearest neighbor classifier lead to concise and straightforward decision rules in 2D space, which can be interpreted and analyzed by human experts easily.

Two-class problems When the classification problem is two-class ($K = 2$), LDA can only generate one-dimensional features. Our experiments have shown that 1D features are insufficient to represent the underlying complexity of video forensics problems. As a result, for binary classification problems, we use a decision tree classifier without dimensionality reduction.

4. Experiments and Results

In this section we report all the details of the experiments and comment on the results.

We study the effectiveness of our approach using the following datasets:

- VISION [32]: the VISION dataset contains 629 pristine MP4/MOV videos captured by 35 different Apple, Android and Microsoft mobile devices.
- EVA-7K [39]: the EVA-7K dataset contains approximately 7000 videos captured by various mobile devices, uploaded to distinct social networks, and edited by different video manipulation tools. The authors took a subset of videos from the VISION dataset and edited them with a number of video editing tools. Then, they uploaded both the original videos and edited videos to four social networks, namely YouTube, Facebook, TikTok and WeiBo. The videos were subsequently downloaded back. The EVA-7K dataset is made up of the pristine videos, edited videos, and downloaded videos.

The VISION dataset is used for device attribution experiments, while all other experiments are conducted on the larger EVA-7K dataset.

We demonstrate the effectiveness of our approach in four video forensic scenarios. In all of the experiments below that use LDA and nearest neighbor classification, we choose $\alpha = 300$, $\beta = 4$, and $\lambda = 5$. For each of the scenarios below, unless indicated otherwise, the dataset is split into two halves with stratified sampling for training and validation, respectively. The metadata nodes and fields that are excluded during metadata extraction, as well as the list of continuous features in the vector representation step, are provided in supplementary materials. We mainly compare the performance of our method to [39], where the EVA-7K dataset is described. For brand attribution, because it is conducted on the VISION dataset, we select [20] as the baseline. The experiment results show that our approach achieves high performance evaluation metrics in all four scenarios.

4.1. Scenario 1: Brand Attribution

Brand attribution consists of identifying the brand of the device used to capture the video file under analysis. We examined brand attribution experiments in two scenarios. In the first experiment, we assume the analyst has access to all possible devices at training time (i.e., close-set scenario). In the second experiment, we assume that a specific device may be unknown at training time (i.e., blind scenario).

Close-set scenario In Table 2, we show the F1-score comparison between our approach and previous work. Our method almost perfectly classifies the VISION dataset, with only one Apple device being misclassified as Samsung. Because our framework is capable of extracting and analyzing more metadata, the performance of our method is better compared to the baseline, especially for brands like Huawei, LG, and Xiaomi. The 2D feature distribution and decision boundary for each label are shown in Figure 6, from which we can determine the metadata similarity between brands and the metadata "variance" for each brand. Visualizations as shown in Figure 6 generated by our method can aid an analyst in making a more interpretable and reliable decision. If a new video under examination is projected close to the LG region, one can assume it is unlikely for that video to come from a Samsung device.

Blind scenario We also examined device attribution with an unknown device using our approach. Let us consider a specific example where device ID 20 (an Apple device) is the unknown device. In the training phase, we use the entire VISION dataset except for samples from device ID 20. In order to classify this device that is unknown to the classifier, we project its features in 2D space and plot them

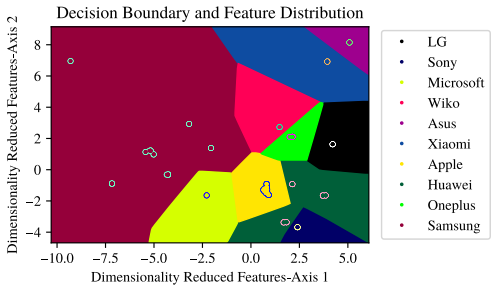

Figure 6: 2D feature distribution and classification boundary for device attribution scenario.

Table 2: F1-score comparison of device attribution scenario.

Device Attribution		
Brand	Iuliani, *et al.* [20]	Our Approach
Apple	1.00	0.99
Asus	1.00	1.00
Huawei	0.87	1.00
LG	0.94	1.00
Oneplus	0.93	1.00
Samsung	0.93	0.99
Wiko	0.65	1.00
Xiaomi	0.74	1.00

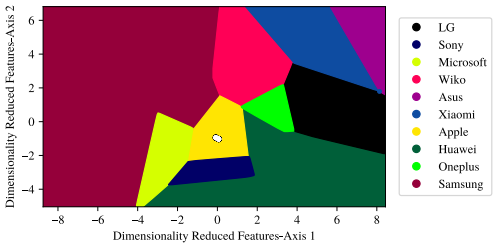

Figure 7: Example of blind device attribution scenario. The projected samples from the unknown device are shown with white markers with black contour.

in the decision boundary plot shown in Figure 7. It can be seen that all samples lie in Apple's decision region, which indicates that the unknown samples are classified correctly.

4.2. Scenario 2: Manipulation Tool Identification

The goal of this scenario is to determine the video editing tool used to manipulate a given video file. We considered native video files from the acquisition devices as non-edited, and compare them with video files edited using Avidemux [5], Exiftool [16], ffmpeg [10], Kdenlive [23] and Premiere [1] as reported in [39].

In Table 3, we compare the F1-score of our method to previous work. Our method achieves a higher average F1-score compared to the previous work. The 2D feature distribution and decision boundary for this scenario are shown in Figure 8.

4.3. Scenario 3: Social Network Attribution

In our social network attribution scenario, we classify the source social network of the video files. If a video file is not downloaded from a social network or the video file comes from an unknown social network, it will be classified

Table 3: F1-score comparison of manipulation tool identification scenario.

Manipulation Tool Identification		
Tool	Yang, *et al.* [39]	Our Approach
Native	0.97	1.00
Avidemux	0.99	0.98
Exiftool	0.98	1.00
ffmpeg	0.94	1.00
Kdenlive	0.95	1.00
Premiere	1.00	0.99
Average	0.97	0.99

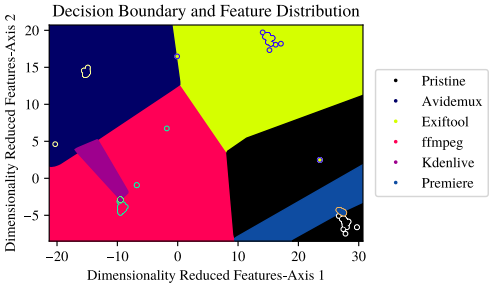

Figure 8: 2D feature distribution and classification boundary for manipulation tool identification scenario.

as "other". The F1-scores in this scenario are shown in Table 4; the 2D feature distribution and decision boundary for this scenario are shown in Figure 9. For this task, our approach achieves high average F1-score of 0.99. The high performance also implies that each social network leaves a unique fingerprint on its videos.

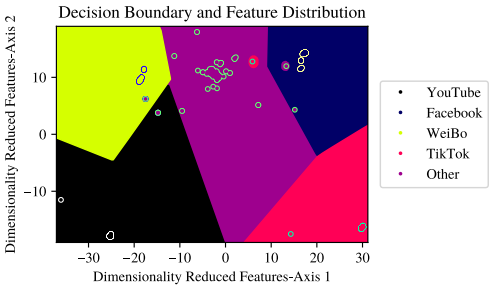

Figure 9: 2D feature distribution and classification boundary for social network attribution scenario.

Table 4: F1-score comparison of social network attribution scenario.

Social Network Attribution		
Social Network	Yang, *et al.* [39]	Our Approach
YouTube	1.00	0.99
Facebook	1.00	1.00
WeiBo	0.99	0.99
TikTok	1.00	1.00
Other	-	0.99

4.4. Scenario 4: Manipulation Detection

There are two sub-tasks within this scenario. As discussed above, for binary classification scenarios such as manipulation detection, LDA can only generate one-dimensional features. It limits the features' expression power after dimensionality reduction, which leads to inferior classification

performance. Therefore, for binary classification problems, we prefer using a decision tree classifier without dimensionality reduction.

Manipulation detection on videos files from social networks In this task, we detect manipulated videos given the fact that both pristine and edited videos are uploaded to social networks first. We compare the performance of using our features with different classification strategies to [39], and the results are shown in Table 5. Using the EVA-7K dataset, the manipulation detection problem is unbalanced because there are much more edited videos than original ones (edited:pristine≈9:1), meaning more samples with positive labels. In this case, the true positive rate (TPR) and the true negative rate (TNR) reflect a classifier's performance more objectively than F1-score. It can be seen that our features combined with decision tree classifier achieve higher TNR for videos from all four social networks. When we use LDA and nearest neighbor classifier for this scenario, the classifier completely fails for videos from TikTok and WeiBo. It is likely because LDA can only generate one-dimensional features, which do not possess enough degrees of freedom to represent the complexity of this problem. Thus, the decision tree classifier is preferred for this scenario.

From Table 5, we also have a glimpse of how each social network process uploaded videos. For WeiBo and TikTok videos, conducting further metadata-based forensic analysis becomes unreliable, which indicates they may have significantly altered videos uploaded to their platforms. YouTube videos can be classified perfectly, which implies that they apply minimum modification to videos' metadata.

Table 5: Performance evaluation metrics comparison between our approach and previous work. TPR and TNR stand for True Positive Rate and True Negative Rate, respectively. The accuracy score has been balanced.

Social Network Manipulation Detection

		Yang, *et al.* [39]	Our Features+ Decision Tree	Our Features+ LDA & NNC
Facebook	Accuracy	0.76	0.84	0.62
	TNR	0.40	0.87	0.30
	TPR	0.86	0.82	0.95
TikTok	Accuracy	0.80	0.69	0.50
	TNR	0.51	0.94	0.00
	TPR	0.75	0.43	1.00
Weibo	Accuracy	0.79	0.63	0.50
	TNR	0.45	0.57	0.00
	TPR	0.82	0.68	1.00
YouTube	Accuracy	0.60	1.00	1.00
	TNR	0.36	1.00	1.00
	TPR	0.74	1.00	1.00

Manipulation detection on local videos In this task, we classify pristine videos and edited videos that are not exchanged via social network. To mimic the real world classification scenario, we employ a similar leave-one-out validation strategy as introduced in [39]. This approach takes the video files from one device model out as the validation set at a time. Since there is only one Microsoft device among 35 models, it is discarded in this scenario, as described in [39]. Because the Microsoft device either belongs to the training set or validation set, we are left with no samples to validate or no data to train. The mean balanced accuracy comparison of the 34-fold cross validation is shown in Table 6. Our approach achieves higher performance compared to previous works.

Table 6: Comparison of our method with previous works. The balanced accuracy is averaged over 34 folds.

Manipulation Detection on Local Videos

	Balanced Accuracy
Güera, *et al.* [15]	0.67
Iuliani, *et al.* [20]	0.85
Yang, *et al.* [39]	0.98
Our Features + Decision Tree	0.99

5. Conclusion

In this paper, we proposed a video forensics approach for MP4 video files based on metadata embedded in video files. Our improved metadata extraction and feature representation scheme allows one to represent more metadata information in a compact feature vector. We use feature selection, dimensionality reduction, and nearest neighbor classification techniques to form interpretable and reliable decision rules for multiple video forensics scenarios. Our approach achieves better performance than other methods.

The performance of our method in many of the scenarios indicates that we need to increase our video forensics dataset to include more difficult cases. Our research also exposed the limitation of metadata-based forensics methods, namely its failure to analyze videos from specific social networks such as TikTok and WeiBo. This is a significant disadvantage compared to pixel-based methods. In the future, we plan to continue exploring the potential of metadata-based video forensics by adding the ability to parse more manufacturer-specific data models (e.g., Canon's CNTH tags [9]) and by looking into lower-level metadata in the distribution of audio/video samples as well as location of key frames in the video stream. We hope that metadata-based video forensics methods can be proved to be reliable in more forensic scenarios.

6. Acknowledgment

This material is based on research sponsored by DARPA and Air Force Research Laboratory (AFRL) under agreement number FA8750-20-2-1004. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA and Air Force Research Laboratory (AFRL) or the U.S. Government. Address all correspondence to Edward J. Delp, ace@ecn.purdue.edu.

References

[1] Adobe Inc. Professional video editor & video maker - Adobe Premiere Pro. https://www.adobe.com/products/premiere.html. 7

[2] Apple Inc. Classic version of the quicktime file format specification. https://developer.apple.com/standards/classic-quicktime/. 1

[3] Apple Inc. Quicktime file format specification–metadata. https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/Metadata/Metadata.html. 3

[4] Apple Inc. Quicktime file format specification–movie atoms. https://developer.apple.com/library/archive/documentation/QuickTime/QTFF/QTFFChap2/qtff2.html. 3

[5] Avidemux contributors. Avidemux. http://avidemux.sourceforge.net/. 7

[6] Sevinc Bayram, Husrev Taha Sencar, and Nasir Memon. Video copy detection based on source device characteristics: A complementary approach to content-based methods. *Proceedings of the ACM International Conference on Multimedia Information Retrieval*, pages 435 – 442, October 2008. Vancouver, British Columbia, Canada. 1

[7] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, H. Anderson, H. Roff, G. C. Allen, J. Steinhardt, C. Flynn, S. O hEigeartaigh, S. Beard, H. Belfield, S. Farquhar, C. Lyle, R. Crootof, O. Evans, M. Page, J. Bryson, R. Yampolskiy, and D. Amode. The malicious use of artificial intelligence : Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*, February 2018. 1

[8] Committee On Ethics. Intentional use of audio-visual distortions and deep fakes. https://ethics.house.gov/campaign-activity-pink-sheets/intentional-use-audio-visual-distortions-deep-fakes. 1

[9] Exiftool contributors. Canon Tags. https://exiftool.org/TagNames/Canon.html. 8

[10] FFmpeg contributors. FFmpeg. https://www.ffmpeg.org/. 7

[11] J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer, New York, USA, 2009. 6

[12] R. T. Garcia. Deepfakes are being used to puncture politicians bluster. https://ffwd.medium.com/deepfakes-are-being-used-to-puncture-\politicians-bluster-e4bb4473841. 1

[13] Google LLC. Android developers: Supported media formats. https://developer.android.com/guide/topics/media/media-formats. 1

[14] S. Grobman. Mcafee labs 2020 threats predictions report. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-labs-2020-threats-predictions-report/. 1

[15] D. Güera, S. Baireddy, P. Bestagini, S. Tubaro, and E. J. Delp. We need no pixels: Video manipulation detection using stream descriptors. *arXiv preprint arXiv:1906.08743*, June 2019. 1, 2, 8

[16] P. Harvey. ExifTool by Phil Harvey. https://exiftool.org/. 7

[17] ISO. ISO/IEC 13818-1:2019 - Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems. https://www.iso.org/standard/75928.html. 3

[18] ISO. ISO/IEC 14496-12:2020 - Information technology — Coding of audio-visual objects — Part 12: ISO base media file format. https://www.iso.org/standard/74428.html. 1

[19] ISO. ISO/IEC 14496-14:2020 - Information technology — Coding of audio-visual objects — Part 14: MP4 file format. https://www.iso.org/standard/79110.html. 3

[20] M. Iuliani, D. Shullani, M. Fontani, S. Meucci, and A. Piva. A video forensic framework for the unsupervised analysis of mp4-like file container. *IEEE Transactions on Information Forensics and Security*, 14(3):635–645, March 2019. 1, 2, 6, 7, 8

[21] Keith Jack. *Chapter 13 - MPEG-2*. Newnes, Burlington, MA, 2007. 1

[22] C. Jee. An indian politician is using deepfake technology to win new voters. https://www.technologyreview.com/2020/02/19/868173/an-indian-politician-is-using\-deepfakes-to-try-and-win-voters/. 1

[23] Kdenlive contributors. Kdenlive. https://kdenlive.org/en/. 7

[24] M. Koopman, A. Macarulla Rodriguez, and Z. Geradts. Detection of deepfake video manipulation. *Proceedings of the Irish Machine Vision and Image Processing Conference*, pages 133–136, August 2018. Belfast, United Kingdom. 1

[25] Leo van Stee. On date, time, location and other metadata in mp4/mov files. https://leo-van-stee.github.io/. 4

[26] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro. An overview on video forensics. *APSIPA Transactions on Signal and Information Processing*, 1:e2, 2012. 1, 2

[27] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen. Multi-task learning for detecting and segmenting manipulated facial images and videos. *Proceedings of the IEEE International Conference on Biometrics*, September 2019. Tampa, Florida. 1

[28] D. Ruiz. Deepfakes laws and proposals flood us. https://blog.malwarebytes.com/artificial-intelligence/2020/01/deepfakes-laws-and-proposals-flood-us/. 1

[29] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan. Recurrent convolutional strategies for face manipulation detection in videos. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. Long Beach, California, USA. 1

[30] R. Schafer and T. Sikora. Digital video coding standards and their role in video communications. *Proceedings of the IEEE*, 83(6):907–924, June 1995. 2

[31] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, August 2000. 5

[32] D. Shullani, M. Fontani, M. Iuliani, O. Alshaya, and A. Piva. Vision: a video and image dataset for source identification. *EURASIP Journal on Information Security*, 2017:15, October 2017. 6

[33] Sony Corporation. FDR-AX40 Specifications. https://www.sony.co.in/electronics/handycam-camcorders/fdr-ax40/specifications. 1

[34] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, December 2012. 3

[35] G. J. Sullivan and T. Wiegand. Video compression - from concepts to the H.264/AVC standard. *Proceedings of the IEEE*, 93(1):18–31, January 2005. 3

[36] J. Umawing. The face of tomorrows cybercrime: Deepfake ransomware explained. https://www.terabitweb.com/2020/06/26/the-face-of-tomorrows-cybercrime-deepfake-ransomware-explained/. 1

[37] D. Vázquez-Padín, M. Fontani, D. Shullani, F. Pérez-González, A. Piva, and M. Barni. Video integrity verification and gop size estimation via generalized variation of prediction footprint. *IEEE Transactions on Information Forensics and Security*, 15:1815–1830, 2020. 2

[38] D. Webb. Avatarify: Create real time deepfakes for video calls. https://ccm.net/faq/64681-avatarify-video-call-deepfakes. 1

[39] P. Yang, D. Baracchi, 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*, 14(5):947–954, August 2020. 1, 2, 6, 7, 8

[40] H. Yao, R. Ni, and Y. Zhao. Double compression detection for h. 264 videos with adaptive gop structure. *Multimedia Tools and Applications*, 79(9):5789–5806, March 2020. 2

[41] A. York. Always up-to-date guide to social media video specs. https://sproutsocial.com/insights/social-media-video-specs-guide/. 1