Detección de Post-Procesamiento en Vídeos Digitales de Dispositivos Móviles mediante el Análisis de la Estructura del Contenedor Multimedia

Carlos Quinto Huamán, Daniel Povedano Álvarez, Ana Lucila Sandoval Orozco, and Luis Javier García Villalba, *Member, IEEE*

***Resumen*—Technological innovations have increased the usefulness of mobile devices and a mobile phone is the most useful tool for any daily task since, in addition to providing access to a wide variety of digital content (instant messaging applications, social networks, services, tv, and so on), it allows the generation of multimedia content (videos and images) taking advantage of the advantages offered by integrated cameras. It is increasingly common for users of these devices to share these videos on different platforms to express themselves, without knowing that this content is exposed to any manipulation, compromising its authenticity and integrity. Likewise, videos shared on social networks and through instant messaging applications go through filtering and compression processes to reduce their size, facilitate their transfer, and optimize storage on their platforms. The result of these transformations leaves a distinctive pattern in the multimedia content of the social network used. This work presents a forensic method to identify the characteristic fingerprints that each social network and instant messaging application leaves in the MOV and MP4 videos shared across their platforms. This method is based on the extraction and analysis of the structure of multimedia containers, and the use of supervised machine learning algorithms.**

***Palabras claves*—Container Structure Analysis, Forensics Analysis, Multimedia Container, Social Media Detection, Supervised Classification Techniques.**

I. Introducción

Los teléfonos móviles se han convertido en una herramienta fundamental para las personas, debido que ofrece realizar múltiples actividades con un único dispositivo, como acceder a Internet, enviar correos electrónicos, usar la cámara integrada para capturar contenido multimedia, usar aplicaciones que solucionan operaciones que antes demandaba mucho más tiempo (transacciones bancarias, solicitar citas médicas, compras online, etc). Esto lo convierte en uno de los dispositivos más demandados en los últimos años y se prevé un crecimiento en los próximos. Según la investigación de Cisco [1], en 2022, el tráfico IP global alcanzará los 396 Exabytes mensuales (4,8 Zettabytes anuales), los usuarios de Internet aumentarán a 4.800 millones de los 3.400 millones del 2017 y habrá 28.500 millones de conexiones de dispositivos personales fijos y móviles de los 18.000 millones del 2017. Asimismo, el tráfico de vídeo IP representará un 82 % del tráfico global IP, comparado con el 75 % que se alcanzó en 2017.

C. Quinto Huamán, D. Povedano Álvarez, A. L. Sandoval Orozco and L. J. García Villalba. Grupo de Análisis, Seguridad y Sistemas (GASS), Departamento de Ingeniería del Software e Inteligencia Artificial, Facultad de Informática, Despacho 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040 Madrid, España. e-mail: {cquinto, dpovedano}@ucm.es, {asandoval, javiergv}@fdi.ucm.es.

Un reciente estudio de Hootsuite [2] indica que actualmente un usuario medio de Internet pasa más de 6 horas y media en línea cada día, lo que significa que la comunidad digital del mundo pasará más de 1.200 millones de años utilizando Internet en 2019. También señalan que este año existe alrededor de 3.484 millones de usuarios activos de redes sociales, que representa un 9 % más que en 2018. De estos, 3.256 millones acceden a estas a través de los teléfonos móviles. También indican que la red social Facebook es la más popular con 2.120 millones de usuarios activos mensualmente en todo el mundo, seguida por Youtube con 1.900 millones y por WhatsApp con 1.500 millones. En cuanto aplicaciones de mensajería instantánea WhatsApp es la preferida por 133 países del mundo y Facebook messenger en 75 países.

Es evidente que la tecnología provee de múltiples beneficios a la cotidianidad de la sociedad actual, sin embargo, estos beneficios se pueden convertir en un puente o conexión para que personas malintencionadas aprovechen los recursos que se encuentran sin la protección debida y realicen algún tipo de fraude o falsificación. Los vídeos capturados por dispositivos móviles no están exentos a este tipo de amenazas, porque es común compartir este tipo de ficheros por redes sociales, dispositivos de almacenamiento, incluso por la pérdida del dispositivo móvil. Por lo anterior, los vídeos están cada vez más propensos a ser manipulados y ser presentados como pruebas digitales en ámbitos procesales para evadir responsabilidades sobre acciones delictivas como pornografía infantil, tráfico de personas, etc. En este sentido, es necesario investigar sobre diferentes métodos para verificar la autenticidad e integridad de un vídeo. En esta investigación, se presenta un método para la detección de post-procesamiento de vídeos compartidos por las plataformas de redes sociales y aplicaciones de mensajería instantánea basado en el análisis de la estructura del contenedor multimedia combinado con el uso de algoritmos de aprendizaje supervisado.

Este trabajo está estructurado en 6 secciones, siendo la primera la presente introducción. En la Sección II se describe brevemente algunos conceptos sobre contenedores multimedia. La Sección III estudia los trabajos relacionados a la identificación de la fuente de vídeos utilizando la estructura de contenedores multimedia. En la Sección IV se presenta el método propuesto. En la Sección V se describen los

https://doi.org/10.12804/si9789587844337.05

experimentos y resultados. Por último en la Sección VI se presentan las conclusiones del presente trabajo.

II. Contenedores Multimedia

El contenedor multimedia es un formato de archivo que contiene varios tipos de datos como el flujo de datos de vídeo y audio, previamente comprimidos por un determinado códec, metadatos, además de otros elementos como subtítulos. Para encapsular o envolver estos datos los fabricantes siguen un formato preestablecido en las especificaciones técnicas de estándares internacionales. En [3] [4] indican que un contenedor multimedia está constituido por una pista de vídeo y una de audio. Ambas pistas son comprimidas por un códec propietario de cada fabricante.

En [5] se indica que la compresión del vídeo se consigue mediante la eliminación de la redundancia temporal que existe entre los fotogramas que componen una secuencia, de esta forma sólo se obtienen componentes necesarios para una reproducción fiel de los datos. Existen diferentes estándares de compresión de vídeos, pero en la actualidad los más usados por los dispositivos móviles son dos: 1) H264/AVC o MPEG-4 Parte 10 [6] y 2) H265/HEVC o MPEG-H Parte 2 [7], ambos desarrollados por el ITU-T *International Telecommunication Union* (ITU) y ISO/IEC *International Organization for Standardization* (ISO), *International Electrotechnical Commission* (IEC). Los tres contenedores multimedia más usados en la actualidad son: 1) Contenedor MP4 del estándar *Moving Picture Experts Group* (MPEG)-4 parte 14. 2) Contenedor 3GP que cumple las especificaciones MPEG-4 parte 14. Los fabricantes cuyo sistema operativo es Android [8], utilizan principalmente los dos primeros contenedores. 3) Contenedor MOV del estándar *QuickTime*, desarrollado por *Apple* [9]. Este último contenedor es usado por los dispositivos cuyo sistema operativo es iOS [10]. No obstante, existen otros contenedores multimedia como *Audio Interleave de Microsoft* (AVI) [11] y *Matroska* (MKV) [12] que son usados por dispositivos más específicos. En este trabajo se utilizan vídeos con contenedores multimedia MOV y MP4, debido a su alta presencia en los principales fabricantes de dispositivos móviles. Los contenedores multimedia de vídeos están estructurados por átomos o cajas que sirven de soporte fundamental para el almacenamiento de toda la información necesaria para la reproducción del vídeo [13]. Un átomo contiene una cabecera, seguido de los datos del propio átomo y estos átomos están organizados de forma jerárquica, es decir un átomo puede contener otros átomos y así sucesivamente, pero cuando un átomo no contiene otros átomos se le denomina átomo hoja o *leaf atom* [6] [9]. La cabecera de un átomo contiene generalmente los campos size y type, excepcionalmente el campo extended size. Adicionalmente los átomos tienen uno o más etiquetas con sus respectivos valores. El campo type se traduce en un entero sin signo de 32 bits, interpretado como un código *American Standard Code for Information Interchange* (ASCII) de cuatro caracteres. El campo size se traduce en un entero de 32 bits, interpretado como un código de cuatro caracteres. El tamaño real de un átomo no puede ser menor de 8 bytes dado que siempre contienen los campos type y size. En este sentido, el átomo se representa como: */moov/*; el átomo que contiene otro átomo hijo como: */moov/mvhd/*; el átomo con etiqueta y valor como: */moov/mvhd/version,* ***value****: 0))*. Este conjunto de características se denomina estructura del contenedor multimedia. Cada red social y aplicación de mensajería instantánea incrusta una estructura propia al momento de compartir los vídeos; esto se debe a que ejecutan un proceso de re-compresión sobre los vídeos.

III. Trabajos Relacionados

Los últimos años el aprendizaje automático o *Machine Learning* (ML) ha contribuido notablemente a la resolución de diversos tipos de tareas, como la gestión de riesgos financieros, detección precoz del cáncer, seguridad de datos, la educación, campañas electorales, ciencia forense, etc. Según [14] ML es un método de análisis de datos que automatiza la construcción de modelos analíticos y está basado a cuatro tipos de aprendizaje: supervisado, semi-supervisado, no supervisado y de refuerzo.

En este contexto, la literatura nos provee trabajos relacionados a la detección de la fuente de vídeos utilizando técnicas de ML, pero en su gran mayoría están orientados a explotar características tradicionales como el ruido del sensor [15] [16] [17] [18] [19] [20] [21], existiendo pocas investigaciones que utilizan la estructura de contenedores multimedia para estos fines.

No obstante, este número reducido de trabajos que explotan las estructura de contenedores multimedia con fines de detección de la fuente, se centran en vídeos con formatos AVI, siendo muy limitado para contenedores MOV y MP4. Asimismo, existen investigaciones que analizan la estructura del contenedor multimedia de un número reducido de redes sociales y aplicaciones de mensajería instantánea.

En [18], se realiza un estudio de los tipos características que son objeto de análisis forense en dispositivos móviles. El problema fundamental de este enfoque es que los diferentes modelos de las cámaras digitales usan componentes de un número reducido de fabricantes y que los algoritmos que usan para la generación de las imágenes y vídeos también son muy similares entre modelos de la misma marca.

En [22] se realiza una comparación minuciosa de los principales grupos de técnicas de identificación de la fuente de adquisición. Estas se dividen en cinco grupos y están basadas en: metadatos, características de la imagen, defectos de la matriz *Color Filter Array* (CFA) e interpolación cromática, imperfecciones del sensor y las transformadas wavelet.

En [23] los autores implementan una técnica para verificar la integridad de vídeos con formato AVI, generados por grabadores de datos de eventos (*Video Event Data Recorders* (VEDRS)). Realizaron el análisis de la estructura de 296 vídeos originales que posteriormente fueron editados por 5 programas de edición. Los resultados del análisis demostraron que los editores cambian notablemente la estructura y los valores de los metadatos con respecto los originales. Cada programa de edición incrusta una estructura específica que ayuda al analista forense a detectar si un vídeo ha sufrido algún tipo de manipulación.

En [24] se realizó un análisis de las estructuras de vídeos con formato AVI y MP4, agrupados en 19 modelos de cámaras digitales, 14 modelos de teléfonos móviles y 6 programas de edición. Después de analizar los vídeos originales los

autores determinaron que las estructuras de cada tipo de contenedor no está estrictamente definida como se especifica en los estándares. Se encontraron diferencias considerables entre vídeos generados por dichos dispositivos. Asimismo, los vídeos AVI después de ser manipulados con los programas de edición, cambiaron la estructura interna incluyendo los valores de los metadatos, características esenciales para saber el origen de los vídeos.

En [25], los autores implementaron un método no supervisado para verificar la integridad de vídeos basado en la disimilitud entre vídeos originales y editados. Asimismo, desarrollaron un método para identificar la fuente de adquisición del vídeo mediante el análisis de los contenedores. Para lograr este objetivo utilizaron la librería *MP4Parser*[26], obteniendo ficheros de lenguaje de marcado extensible (*Extensible Markup Language* (XML)) para un posterior análisis, logrando buenos resultados en sus experimentaciones, sosteniendo que la solución utiliza un mínimo de recurso computacional a comparación de otras alternativas.

IV. Método Propuesto

En este trabajo se propone un método forense para la identificación de redes sociales y aplicaciones de mensajería instantánea en vídeos con formato MOV y MP4, utilizando como característica principal la estructura de los contenedores multimedia. La metodología y procedimientos que se han seguido para evaluar y seleccionar el mejor modelo de ML supervisado se detalla en la Figura 1.

IV-A. Extracción de Átomos

El proceso de extracción de características es una parte fundamental en la aplicación de cualquier técnica de aprendizaje automático. Para este trabajo se ha desarrollado un algoritmo de extracción de átomos, que se encarga de obtener un conjunto de características (átomos, etiquetas, valores y orden de aparición) de los vídeos con formato MOV y MP4, para ser analizadas y posteriormente encontrar un patrón que permita distinguir una red social y aplicación de mensajería instantánea. La implementación del algoritmo está basado al uso de funciones recursivas. Primero, se obtiene byte inicial del átomo; el tamaño del átomo (4 bytes); y el tipo de átomo (4 bytes) que está representado por una cadena de caracteres. Segundo, se extrae el átomo leído y se asigna un orden relativo a cada átomo, con la finalidad de organizar eficientemente las estructuras y evitar eliminar átomos que poseen similar nombre. Tercero, se localiza y extrae los átomos hijos conjuntamente con sus etiquetas y valores; este proceso se repetirá mientras el átomo contenga otros átomos. En caso que el átomo extraído no es ubicado en la lista de átomos previamente implementados, dicho átomo se registra como átomo desconocido. Una vez finalizado el proceso, se almacena la información en un DataFrame.

IV-B. Pre-Procesamiento

Las características que provienen de la extracción de átomos están estructuradas en un DataFrame con los siguientes campos: *Path-file-name*, que es la ruta del vídeo seguido por el nombre; *Path-origin*, es la ruta del vídeo; *Class-label*, es la identificación de cada red social y aplicación de mensajería; *File-name*, es el nombre del vídeo; *Marker*, es la marca del dispositivo móvil; *Model*, es el modelo del dispositivo móvil; *PathOrder-tag* es el conjunto de átomos con su respectivo orden relativo, seguido de la etiqueta con su respectivo valor; *ValueReading-orders*, es el orden absoluto de cada átomo, también llamado orden de aparición.

Dado un vídeo (X), contiene un conjunto de átomos $(a_1,....a_n)$, representado por */ftyp-1/*. También, estos átomos pueden contener otros átomos y etiquetas: *a*=$((a_1), w_1....(a_n), w_n)$ representado por */ftyp-1/majorBrands*. Asimismo, estas etiquetas tienen valores: */ftyp-1/majorBrands: @mp42*. En este sentido, el *PathOrder-tag* contiene 2 tipos de características: Secuencia de átomos *(/ftyp-1/)* y secuencia de átomos seguido de las etiquetas *(/ftyp-1/majorBrands)*.

Para el presente trabajo se utiliza solo se utiliza los campos *Path-file-name*, *PathOrder-tag*, *Class-label*, y por ello se elimina las columnas que no son utilizadas. Seguidamente se asigna un identificador a cada clase y se agrupa la estructura de cada vídeo, representado por un vector con N *PathOrder-tag* con igual cardinalidad. Esto se lleva a cabo utilizando la presencia o ausencia de los *PathOrder-tag*, asignando la variable binaria 1 en caso de presencia y 0 en caso de ausencia. Después, se obtiene el conjunto de características de entrenamiento $Train_{features}$ y sus respectivas etiquetas. Finalmente, se divide el $Train_{features}$ 2 partes: 80 % (Conjunto de entrenamiento) y 20 % (Conjunto de test).

IV-C. Oversampling y Reducción de Dimensiones

Un problema común en el uso de técnicas de ML es el desbalanceo entre clases.

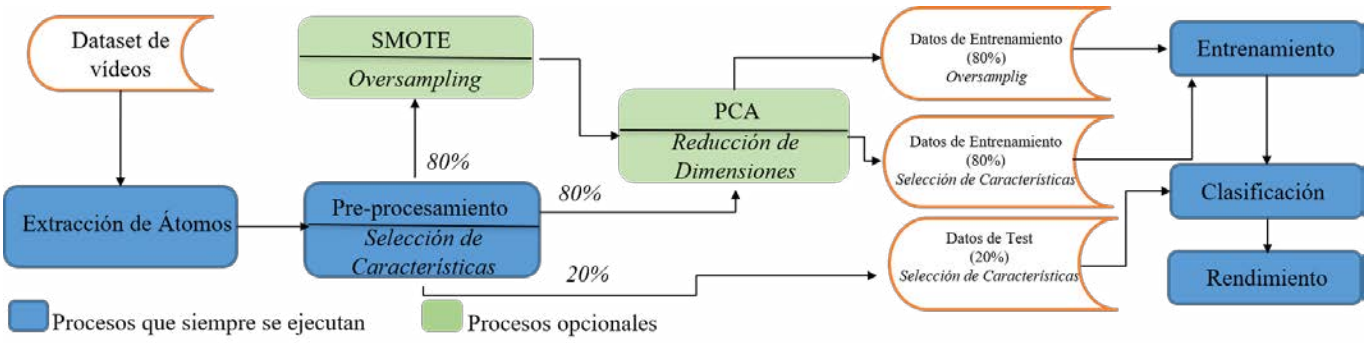

Figura 1: Pipeline del método propuesto

Si bien es cierto, la cantidad de vídeos utilizados para cada clase es el mismo, se ha considerado este paso como opcional para evaluar el funcionamiento del modelo bajo éstas condiciones y así solventar el impacto que tiene sobre la decisión final de los clasificadores.

En este sentido, se hace uso de la técnica muy utilizada en la literatura denominado *Syntethic Minority OverSampling* (SMOTE) [27], que genera o incrementa el número de instancias de las clases minoritarias de forma sintética [28] [29], en este contexto cada instancia es un vector por cada vídeo con N *PathOrder-tag*.

Otro problema frecuente es contar con características menos importantes y que no aportan significativamente a la predicción. A esta problemática se suma un mayor tiempo de procesamiento y mayor uso de recursos. Para mitigar esta dificultad, opcionalmente se hace uso de la técnica de Análisis de Componentes Principales o *Principal Component Analysis* (PCA), que ayuda a identificar y eliminar *PathOrder-tag* irrelevantes para mejorar el rendimiento computacional y entender mejor la evaluación del modelo y sus resultados. Cabe señalar, que a ésta técnica se le antepone la técnica SMOTE a fin de evitar causar un sobreajuste o *Overfitting* al modelo [30].

IV-D. Training

La fase de entrenamiento es fundamental ya que determina el rendimiento de los algoritmos y de esta forma elegir el mejor modelo predictivo para realizar la clasificación. Es común utilizar un buen porcentaje de características, dejando una porción para evaluar el rendimiento y capacidad de generalizar nuevas características que no han sido contemplados en la fase de entrenamiento. En nuestro caso, como se detalló en la Sección IV-B se destina un 80 % de *PathOrder-tag* para el entrenamiento (Conjunto de entrenamiento) y un 20 % para evaluar el modelo (Conjunto de Test).

Una vez definido el porcentaje de los datos de entrenamiento, el siguiente paso es seleccionar los hiper-parámetros de los modelos de ML y esto depende del cada algoritmo que se utilizará. Existen varios procedimientos para obtener hiper-parámetros que ofrezcan un buen rendimiento del algoritmo de clasificación [31]. En este trabajo se utilizó la técnica búsqueda exhaustiva o *grid search* [28]. Para ello, se construye los *pipelines* para cada *ensemble classification algorithm*: *Logistic Regression* (LR), *Support Vector Machine* (SVM), *Decision Tree* (DT), *Linear Discriminant Analysis* (LDA), *Gaussian Naive Bayes* (GNB), *KNeighbors* (KNN), y para cada *Non-ensemble classification algorithm*: *Random Forest* (RF), *Gradient Boosting Classifier* (GBC), *Extreme Gradient Boosting* (XGB) [29] [32]. El conjunto de hiper-parámetros seleccionados para cada uno de los algoritmos propuestos, aplicando la validación cruzada o *cross-validation* con *k=5* se detalla a continuación:

- ***Logistic Regression (LR)***: *clf_C: 0.1, clf_penalty: l2, clf_solver: liblinear*.
- ***Support Vector Machine (SVM)***: *clf_C: 0.1, clf_kernel: linear*.
- ***Decision Tree (DT)***: *clf_min_samples_split: 10, clf_min_samples_leaf: 1, clf_max_depth: 18*.
- ***Linear Discriminant Analysis (LDA)***: *clf_solver: svd*
- ***Gaussian Naive Bayes (GNB)***: *clf_var_smoothing: 1e-08*.
- ***KNeighbors (KNN)***: *clf_weights: distance, clf_n_neighbors: 15, clf_metric: cosine*.
- ***Random Forest (RF)***: *clf_n_estimators: 1200, clf_min_samples_split: 10, clf_min_samples_leaf: 4, clf_max_features: sqrt, clf_max_depth: 10, clf_bootstrap: True*.
- ***Gradient Boosting Classifier (GBC)***: *clf_subsample: 1.0, clf_n_estimators: 100, clf_min_samples_split: 0.9, clf_min_samples_leaf: 0.2090*.
- ***Extreme Gradient Boosting (XGB)***: *clf_subsample: 0.6, clf_min_child_weight: 5, clf_max_depth: 4, clf_gamma: 2, clf_colsample_bytree: 0.8*.

IV-E. Testing

En esta fase se hace uso del conjunto de test, es decir el 20 % de *PathOrder-tag* separado en la Sección IV-B, con el objetivo de evaluar la capacidad de clasificación del métodos utilizados. Asimismo, se valida si los métodos se adecuan a un entorno real. El resultado de la evaluación será la elección del modelo que mejor se comporta con el tipo de características que se provee. Para ello, se utiliza 4 métricas de rendimiento que se presentan en a siguiente sección.

IV-F. Métricas de Rendimiento

Como se mencionó en la Sección III, en la actualidad, no existe investigaciones relacionadas a la clasificación supervisada de redes sociales y aplicaciones de mensajería instantánea que utilicen la estructura del contenedor multimedia, y que sirvan como referencia para elegir las mejores métricas. En este contexto, se consideró utilizar 4 métricas para la evaluación del rendimiento de soluciones de identificación de la fuente de adquisición en vídeos *(Accuracy, F1-Score-micro, F1-Score-macro, Log Loss)* [29]. Considerando que se desea clasificar cada muestra (vector con N variables) con su respectiva clase, se denomina verdaderos positivos *(TP)* a las muestras que se clasificaron como positivos correctamente, verdaderos negativos *(TN)* a las muestras que se clasificaron como negativos correctamente, falsos positivos *(FP)* a la predicción no clasificada en la clase de interés, falsos negativos *(FN)* a la predicción no clasificada correctamente en la clase de no interés. Previamente a la evaluación de las métricas propuestas, se evaluó dos métricas de soporte: *Recall*, que es la tasa de verdaderos positivos, es decir nos indica la capacidad del método para clasificar las muestras en la clase correcta; *Precision*, nos indica la proporción de muestras que nuestro método ha clasificado en una determinada clase. En los siguientes párrafos se describen brevemente las métricas utilizadas:

- **Accuracy**. Esta métrica es la más simple de calcular y es el porcentaje total de elementos clasificados correctamente, se define como:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- **F1-Score**. Es la media armónica de *Precision* y *Recall*. Mientras el valor se acerque más a 1, el rendimiento del sistema de clasificación será más óptimo [33]. En

este trabajo, ésta métrica se desglosa en 2, de acuerdo al tipo de configuración del cálculo de la media: 1) F1-macro (Average:macro) que calcula la métrica independientemente para cada clase y luego toma el promedio, es decír, trata a todas las clases por igual; 2) F1-micro (Average:micro) agrega las contribuciones de todas las clases para calcular la métrica promedio. En un sistema de clasificación multiclase, es preferible utilizar esta métrica si se sospecha que puede haber un desequilibrio de clase, es decir puede tener muchos más vídeos de una clase que de otras clases. La métrica de define como:

$$\textit{F1-Score} = 2 * \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} \tag{2}$$

- **Log-loss**. Mide el rendimiento de un modelo de clasificación donde la entrada de predicción es un valor de probabilidad entre 0 y 1. El objetivo de nuestros modelos es minimizar este valor. Un modelo perfecto tendría una pérdida logarítmica de 0, se define como:

$$\textit{Log-loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \tag{3}$$

Donde: N es el número de muestras o instancias, M es el número de etiquetas posibles, y_{ij} toma el valor de 1 para la clase correcta y 0 para otras clases, y_{ij} es la probabilidad asignada para esa clase.

Después de haber detallado el método propuesto, se realiza la evaluación de cada uno de los algoritmos seleccionados en la Sección IV-D, con la finalidad de seleccionar un modelo robusto y eficiente. En la Tabla I se observa el rendimiento de los algoritmos para la identificación de redes sociales y aplicaciones de mensajería instantánea utilizando PCA y descartando SMOTE. Se descartó el uso de SMOTE, debido a que los resultados obtenidos en experimentaciones previas son similares a la Tabla I, demostrando que el dataset de vídeos está balanceado.

Tabla I: Evaluación del rendimiento de los modelos candidatos

Modelo	Accuracy	F1-micro	F1-macro	Log loss
LR	0.975891	0.975891	0.972057	0.063861
RF	0.975891	0.975891	0.972057	0.063479
DT	0.975552	0.975552	0.971684	0.078106
SVM	0.975891	0.975891	0.972057	0.074281
KNN	0.975891	0.975891	0.972057	0.067543
LDA	0.927674	0.927674	0.910590	0.760697
GNB	0.975891	0.975891	0.972057	0.832684
GBC	0.707640	0.707640	0.626514	0.656345
XGB	0.975891	0.975891	0.972057	0.075127

En general, se observa un rendimiento muy alto de todos los algoritmos, excepto GBC, que alcanza un accuracy y F1-micro de 0.707640 y F1-macro de 0.626514. No obstante, LDA, GNB y GBC alcanzan valores log-loss cercanos a 1, lo que significa que el rendimiento en la clasificación de estos algoritmos no son tan perfectos. El modelo con mejor rendimiento es RF, alcanzando excelentes resultados en las 4 métricas. En este sentido, el algoritmo RF, será utilizado en las experimentaciones del presente trabajo.

V. Experimentos y Resultados

Para evaluar el método propuesto se han realizado dos grupos de experimentos: un análisis de la estructura de contenedores multimedia y, posteriormente, la identificación de redes sociales y aplicaciones de mensajería instantánea utilizando el modelo de ML seleccionado.

V-A. Descripción del Dataset

La preparación del conjunto de datos es un proceso muy importante para este trabajo ya que es un factor que determina la calidad de resultados que se desea alcanzar. En la literatura, existen muy pocos conjuntos de datos de vídeos generados por dispositivos móviles que hayan sido compartidos por redes sociales y aplicaciones de mensajería instantánea. Tener un conjunto de datos de vídeos organizado, robusto, balanceado y sobre todo actualizado es todo un reto. Por ese motivo, se ha generado un conjunto de datos heterogéneo y suficientemente grande, para evaluar el método propuesto en escenarios reales y así obtener resultados efectivos. Las Tablas II y III conforman el dataset generado para los experimentos. En la Tabla II se presenta el conjunto de datos que está compuesto por 270 vídeos originales, 10 vídeos por cada modelo. Los vídeos fueron capturados con las opciones de las cámaras por defecto (p. ej. registro de ubicación desactivado). Posteriormente, los vídeos de la Tabla II fueron compartidos a través de 8 redes sociales (2160 vídeos resultantes) y 3 aplicaciones de mensajería instantánea (810 vídeos resultantes), haciendo un total de 2970 vídeos que serán analizados. En la Tabla III se detalla las redes sociales y aplicaciones de mensajería instantánea que se utilizaron con sus respectivas configuraciones de subida y descarga de los vídeos.

Tabla II: Características de los vídeos originales del dataset

Marca	Modelo	Id Modelo	S.O.	Resolución	Codec
Apple	Ipad 2	D01	iOS 9.3.5	1280 x 720p	H.264
	Ipad Air	D02	iOS 11.3	1920 x 1080p	H.264
	Iphone 5	D03	iOS 7.0.4	1920 x 1080p	H.264
	Iphone 5S	D04	iOS 9.2	1920 x 1080p	H.264
	Iphone 6	D05	iOS 8.4	1920 x 1080p	H.264
	Iphone 7	D06	iOS 11.2.6	3840 x 2160p	H.264
	Iphone 8 Plus	D07	iOS 11.2.5	1920 x 1080p	H.265
	Iphone X	D08	iOS 11.4.1	1920 x 1080p	H.264
	Iphone XS Max	D09	iOS 12.1.0	3840 x 2160p	H.264
Huawei	Ascend	D10	Android	1280 x 720p	H.264
	P9	D11	Android	1920 x 1080p	H.264
	P10	D12	Android	1920 x 1080p	H.264
LG	Nexus 5	D13	Android	1920 x 1080p	H.264
	G6	D14	Android	640x480p	H.264
Microsofot	Lumia 640 LTE	D15	Windows Phone	1920 x 1080p	H.264
Motorola	Moto G2	D16	Android	1280 x 720p	H.264
	Nexus 6	D17	Android	1920 x 1080p	H.264
One Plus	A0001	D18	Android	1920 x 1080p	H.264
Samsung	Galaxy A6	D19	Android	1920 x 1080p	H.264
	Galaxy S5	D20	Android	3840 x 2160p	H.264
	Galaxy S7	D21	Android	1920 x 1080p	H.264
	Galaxy S9 Plus	D22	Android	3840 x 2160p	H.265
	Galaxy J5 2016	D23	Android	1920 x 1080p	H.264
	Galaxy Tab A	D24	Android	1280 x 720p	H.264
Xiaomi	Mi3	D25	Android	1280 x 720p	H.264
	Redmi Note 5	D26	Android	1920 x 1080p	H.264
	PocoPhone	D27	Android	3840 x 2160p	H.264

V-B. Análisis de Contenedores Multimedia

Este análisis se realiza para observar el comportamiento de las redes sociales y aplicaciones de mensajería instantánea a la hora de subir y descargar vídeos a través de sus plataformas. Asimismo, conocer de manera general cual es la estructura que inserta cada herramienta a los contenedores multimedia. En este sentido, se realiza la comparación de la estructura del

Tabla III: Configuración del proceso de generación del dataset de redes sociales y aplicaciones de mensajería instantánea

Red Social	Versión	Proceso de Subida	Proceso de Descarga
Facebook HD	Website	Max 4gb, 240min	Firefox (Inspect element)
Facebook SD	Website	Max 4gb, 240min	Firefox (Inspect element)
Youtube	Website	Max 128gb,12hrs	Youtube studio beta
Flickr	Website	Max 1gb	Website(save as)
Linkedin	Website	Max 6gb, min 75kb	Website(save as)
Instagram	Website	Max 10 min, ratio 9:16	Firefox (Inspect element)
Twitter	Website	Max 500mb, 2.20 min	Twitervideodownloader
Tumblr	Website	Max 100mb	Firefox (Inspect element)
Aplicación	**Versión**	**Características**	
Facebook Msn	255.0.0.13.113	-	
WhatsApp	2.19.20	-	
Telegram	5.7.1	Max 1.5GB	

contenedor de un vídeo original, generado por un teléfono móvil de marca Apple y modelo Ipad Air, con la estructura del mismo vídeo luego de haber sido compartido a través de Facebook HD, Youtube, WhatsApp, Linkedin, Telegram. Ver Figura 2.

En vídeo original de marca Apple y modelo Ipad Air, tiene 4 átomos raíz *(ftyp-1, wide-2, mdat-3, moov-4)*. Los valores de las etiquetas *@majorBrands y @compatibleBrand* son *qt y qt* respectivamente. La ruta para obtener el valor de la etiqueta *@timeScale* es */ftyp-1/wide-2/mdat-3/moov-4/mvhd-1/@timeScale:600*. El átomo *moov-4* contiene 4 átomos *trak*, 1 para la pista de vídeo, 1 para la písta de audio y 2 exclusivamente para metadatos. La estructura del contenedor esta compuesto por 651 características.

En vídeo de Facebook HD, tiene 4 átomos raíz *(ftyp-1, moov-2, free-3, mdat-4)*. Los valores de las etiquetas *@majorBrands y @compatibleBrand* son *isom y isomiso2avc1mp41* respectivamente. La ruta para obtener el valor de la etiqueta *@timeScale* es */ftyp-1/moov-2/mvhd-1/@timeScale:1000*. El átomo *moov-2* contiene 2 átomos *trak*, 1 para la pista de vídeo, 1 para la písta de audio. La estructura del contenedor esta compuesto por 267 características.

En vídeo de Youtube, tiene 3 átomos raíz *(ftyp-1, moov-2, mdat-3)*. Los valores de las etiquetas *@majorBrands y @compatibleBrand* son *mp42 y isommp42* respectivamente. La ruta para obtener el valor de la etiqueta *@timeScale* es */ftyp-1/moov-2/mvhd-1/@timeScale:1000*. El átomo *moov-2* contiene 2 átomos *trak*, 1 para la pista de vídeo, 1 para la písta de audio. La estructura del contenedor está compuesto por 250 características.

En vídeo de WhatsApp, tiene 4 átomos raíz *(ftyp-1, beam-2, moov-3, mdat-4)*. Los valores de las etiquetas *@majorBrands y @compatibleBrand* son *mp42 y mp41mp42isom* respectivamente. La ruta para obtener el valor de la etiqueta *@timeScale* es */ftyp-1/beam-2/moov-3/mvhd-1/@timeScale:44100*. El átomo *moov-3* contiene 2 átomos *trak*, 1 para la pista de vídeo, 1 para la písta de audio. La estructura del contenedor esta compuesto por 237 características.

En vídeo de Linkedin, tiene 4 átomos raíz *(ftyp-1, moov-2, free-3, mdat-4)*. Los valores de las etiquetas *@majorBrands y @compatibleBrand* son *isom y isomiso2avc1mp41* respectivamente. La ruta para obtener el valor de la etiqueta *@timeScale* es */ftyp-1/moov-2/mvhd-1/@timeScale:1000*. El átomo *moov-2* contiene 2 átomos *trak*, 1 para la pista de vídeo, 1 para la písta de audio. La estructura del contenedor está compuesto por 258 características.

En vídeo de Telegram, tiene 3 átomos raíz *(ftyp-1, mdat-2, moov-3)*. Los valores de las etiquetas *@majorBrands y @compatibleBrand* son *mp42 y mp41mp42isom* respectivamente. La ruta para obtener el valor de la etiqueta *@timeScale* es */ftyp-1/mdat-2/moov-3/mvhd-1/@timeScale:44100*. El átomo *moov-3* contiene 2 átomos *trak*, 1 para la pista de vídeo, 1 para la písta de audio. La estructura del contenedor está compuesto por 251 características.

Cabe señalar que después de analizar los vídeos de la red social Flickr, se ha confirmado que la plataforma no realiza ningún tipo de re-compresión y tampoco cambia la estructura del contenedor multimedia, manteniendo las mismas características que el vídeo original. En resumen, este análisis ha permitido demostrar demostrar que cada red social y aplicación de mensajería instantánea inserta una estructura diferente. Estas diferencias se derivan del orden de aparición de los átomos y valores asignados cada etiquetas.

V-C. Identificación de Redes Sociales y Aplicaciones de Mensajería Instantánea

En este experimento se utilizaron los 2970 videos compartidos en 8 redes sociales y 3 aplicaciones de mensajería instantanea, repartidos en 2376 videos para train y 594 para test. En la Tabla IV se observa que WhatsApp, Facebook HD, Facebook Msn, Telegram, Youtube, Flickr, Linkedin, Twitter y Tumblr se clasifican al 100 %. Sin embargo la red social Facebook SD alcanza un 0.98 % de acierto, confundiéndose un 0.02 % con Tumblr. Finalmente, Instagram alcanza un 0.69 % de acierto, confundiéndose un 0.31 % con Facebook Msn. En la Figura 3, se observa un resumen del factor de impacto de los *PathOrder-tag* o características que tienen sobre el modelo a la hora de clasificar una clase. Los *PathOrder-tag* que contienen el atomo *wide-2* tienen mayor impacto para definir que la clase sea Flickr (vídeos originales).

Tabla IV: Matriz de confusión para redes sociales y aplicaciones de mensajería instantánea

Social media	WhatsApp	FacebookHD	FacebookSD	FacebookMsn	Telegram	Youtube	Flickr	Linkedin	Instagram	Twitter	Tumblr
WhatsApp	**1**	-	-	-	-	-	-	-	-	-	-
FacebookHD	-	**1**	-	-	-	-	-	-	-	-	-
FacebookSD	-	-	**0.98**	-	-	-	-	-	-	-	0.02
FacebookMsn	-	-	-	**1**	-	-	-	-	-	-	-
Telegram	-	-	-	-	**1**	-	-	-	-	-	-
Youtube	-	-	-	-	-	**1**	-	-	-	-	-
Flickr	-	-	-	-	-	-	**1**	-	-	-	-
Linkedin	-	-	-	-	-	-	-	**1**	-	-	-
Instagram	-	-	-	0.31	-	-	-	-	**0.69**	-	-
Twitter	-	-	-	-	-	-	-	-	-	**1**	-
Tumblr	-	-	-	-	-	-	-	-	-	-	**1**

Iphone Ipad Air

- ftyp-1
  - @majorBrands: qt
  - @minorVersion: 0
  - @compatibleBrand : qt
- wide-2
- mdat-3
- moov-4
  - mvhd-1
    - @version: 0
    - @flags: 0
    - @reserved: 0
    - @timeScale: 600
    - @posterTime: 0
    - @currentTime: 0
    - @nextTrackId: 5
    - @previewTime: 0
    - ...
  - trak-1
    - tkhd-1
      - @version: 0
      - @flags: 0
      - ...
  - trak-2 ...
  - trak-3 ...
  - trak-4 ...
- Total structure = 651

Facebook HD

- ftyp-1
  - @majorBrands: isom
  - @minorVersion: 512
  - @compatibleBrand: isomiso2avc1mp41
- moov-2
  - mvhd-1
    - version: 0
    - @flags: 0
    - @reserved: 0
    - @timeScale: 1000
    - @posterTime: 0
    - @currentTime: 0
    - @nextTrackId: 3
    - @previewTime: 0
    - ...
  - trak-1
    - tkhd-1
      - @version: 0
      - @flags: 3
      - ...
  - trak-2
- free-3
- mdat-4
- Total structure = 267

Youtube

- ftyp-1
  - @majorBrands: mp42
  - @minorVersion: 0
  - @compatibleBrand: isommp42
- moov-2
  - mvhd-1
    - version: 0
    - @flags: 0
    - @reserved: 0
    - @timeScale: 1000
    - @posterTime: 0
    - @currentTime: 0
    - @nextTrackId: 3
    - @previewTime: 3
    - ...
  - trak-1
    - tkhd-1
      - @version: 1
      - @flags: 3
      - ...
  - trak-2
- mdat-3
- Total structure = 250

WhatsApp

- ftyp-1
  - @majorBrands: mp42
  - @minorVersion: 1
  - @compatibleBrand: mp41mp42isom
- beam-2
- moov-3
  - mvhd-1
    - version: 0
    - @flags: 0
    - @reserved: 0
    - @timeScale: 44100
    - @posterTime: 0
    - @currentTime: 0
    - @nextTrackId: 3
    - @previewTime: 0
    - ...
  - trak-1
    - tkhd-1
      - @version: 0
      - @flags: 1
      - ...
  - trak-2
- mdat-4
- Total structure = 237

Linkedin

- ftyp-1
  - @majorBrands: isom
  - @minorVersion: 512
  - @compatibleBrand: isomiso2avc1mp41
- moov-2
  - mvhd-1
    - version: 0
    - @flags: 0
    - @reserved: 0
    - @timeScale: 1000
    - @posterTime: 0
    - @currentTime: 0
    - @nextTrackId: 3
    - @previewTime: 0
    - ...
  - trak-1
    - tkhd-1
      - @version: 0
      - @flags: 3
      - ...
  - trak-2
- free-3
- mdat-4
- Total structure = 258

Telegram

- ftyp-1
  - @majorBrands: mp42
  - @minorVersion: 1
  - @compatibleBrand: mp41mp42isom
- mdat-2
- moov-3
  - mvhd-1
    - version: 0
    - @flags: 0
    - @reserved: 0
    - @timeScale: 44100
    - @posterTime: 0
    - @currentTime: 0
    - @nextTrackId: 3
    - @previewTime: 0
    - ...
  - trak-1
    - tkhd-1
      - @version: 0
      - @flags: 1
      - ...
  - trak-2
- Total structure = 251

Figura 2: Comparación de la estructura de contenedores multimedia

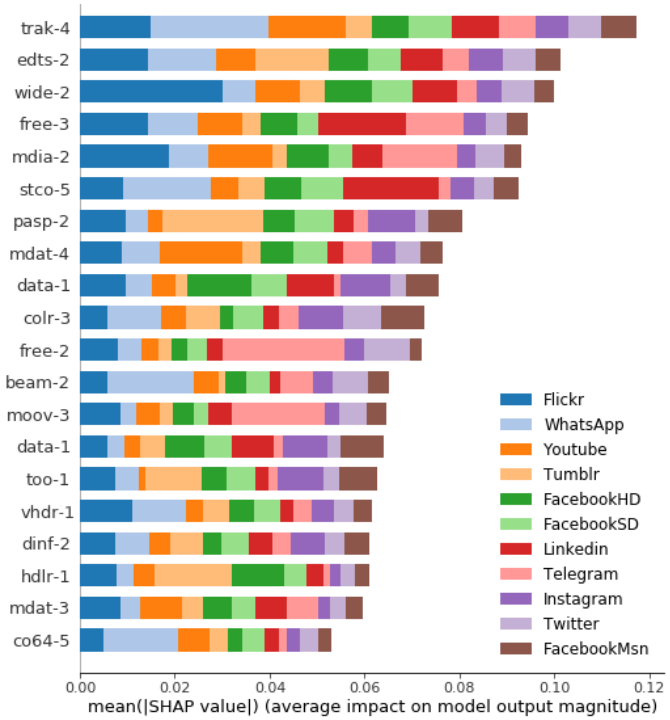

Figura 3: Factor de impacto de *PathOrder-tag*

Los *PathOrder-tag* que contienen el átomo *trak-4* tienen gran impacto para que el modelo clasifique la muestra como WhatsApp. Este experimento nos muestra que el modelo clasifica notablemente las redes sociales. No obstante, recordar que la red social Flickr clasifica perfectamente pero realmente esta clasificando vídeos originales.

En la Figura 4, se examina la predicción individual de los puntos de datos *(PathOrder-tag)* por redes sociales y aplicaciones de mensajería instantánea utilizando el paquete de Python Eli5 [34]. En la Figura 4(a) se examina Facebook Msn, Instagram y Flickr. Se observa que el *PathOrder-tag /moov-2/udta-4/meta-1/ilst-2/too-1/data-1/* tiene mayor peso de contribución y es el más influyente para predecir la aplicación de mensajería Facebook Msn (+0.056) y la red social Instagram (+0.050). No obstante, el conjunto de *PathOrder-tag* positivos de Facebook Msn, alcanza una probabilidad del 0.753 % para que el resultado de la predicción sea ésta aplicación, mientras que el conjunto de *PathOrder-tag* positivos de Instagram, alcanza una probabilidad de 0.244 % para que el resultado de predicción se incline a ésta red social. Asimismo, se observa que el conjunto de *PathOrder-tag* positivos de Flickr logra obtener solo un 0.002 % de probabilidad para que la predicción sea ésta red social (vídeo original).

y=FacebookMsn (probability **0.753**) top features			**y=Instagram** (probability **0.244**) top features			**y=Flickr** (probability **0.002**) top features		
Contribution?	**Feature**	**Value**	**Contribution?**	**Feature**	**Value**	**Contribution?**	**Feature**	**Value**
+0.074	<BIAS>	1.000	+0.084	<BIAS>	1.000	+0.138	<BIAS>	1.000
+0.056	/moov-2/udta-4/meta-1/ilst-2/too-1/data-1/	-0.854	+0.050	/moov-2/udta-4/meta-1/ilst-2/too-1/data-1/	-0.854		*... 6 more positive ...*	
+0.034	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	-3.103	+0.028	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	-3.103		*... 33 more negative ...*	
+0.029	/moov-2/trak-3/mdia-2/mdhd-1/	-0.626	+0.020	/moov-2/trak-3/mdia-2/mdhd-1/	-0.626	-0.009	/moov-2/meta-3/ilst-3/	0.002
+0.027	/moov-2/trak-2/mdia-3/minf-3/stbl-3/stsd-1/avc1-1/pasp-2/	-0.145	+0.016	/moov-2/udta-4/meta-1/ilst-2/too-3/data-1/	-0.058	-0.020	/moov-2/udta-4/meta-1/ilst-2/too-1/data-1/	-0.854
	... 39 more positive ...			*... 20 more positive ...*		-0.022	/moov-2/trak-3/mdia-2/mdhd-1/	-0.626
				... 19 more negative ...		-0.041	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	-3.103

(a) Facebook Msn, Instagram, Flickr

y=Instagram (probability **0.998**) top features			**y=Flickr** (probability **0.001**) top features			**y=FacebookMsn** (probability **0.001**) top features		
Contribution?	**Feature**	**Value**	**Contribution?**	**Feature**	**Value**	**Contribution?**	**Feature**	**Value**
+0.084	<BIAS>	1.000	+0.138	<BIAS>	1.000	+0.074	<BIAS>	1.000
+0.065	/moov-2/udta-4/meta-1/ilst-2/too-1/data-1/	-0.448		*... 7 more positive ...*		+0.039	/moov-2/udta-4/meta-1/ilst-2/too-1/data-1/	-0.448
+0.042	/moov-2/trak-2/mdia-3/minf-3/stbl-3/stsd-1/avc1-1/pasp-2/	-0.497		*... 32 more negative ...*		+0.021	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	-3.089
+0.040	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	-3.089	-0.009	/moov-2/meta-3/ilst-3/	-0.003	+0.013	/moov-2/trak-3/mdia-2/mdhd-1/	-0.610
+0.034	/moov-2/trak-3/mdia-2/mdhd-1/	-0.610	-0.019	/moov-2/udta-4/meta-1/ilst-2/too-1/data-1/	-0.448		*... 11 more positive ...*	
	... 39 more positive ...		-0.021	/moov-2/trak-3/mdia-2/mdhd-1/	-0.610		*... 28 more negative ...*	
			-0.040	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	-3.089	-0.012	/moov-4/trak-2/mdia-4/minf-3/stbl-4/stsd-1/hvc1-1/	0.000

(b) Instagram, Flickr, Facebook Msn)

y=WhatsApp (probability **0.999**) top features			**y=Flickr** (probability **0.001**) top features			**y=YouTube** (probability **0.000**) top features		
Contribution?	**Feature**	**Value**	**Contribution?**	**Feature**	**Value**	**Contribution?**	**Feature**	**Value**
+0.234	/beam-2/	2.923	+0.138	<BIAS>	1.000	+0.084	<BIAS>	1.000
+0.137	<BIAS>	1.000	+0.021	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	3.651		*... 19 more positive ...*	
+0.105	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	3.651		*... 8 more positive ...*			*... 18 more negative ...*	
+0.066	/moov-3/trak-4/mdia-2/hdlr-2/	-0.157		*... 31 more negative ...*		-0.013	/moov-2/trak-3/mdia-2/mdhd-1/	-1.346
+0.047	/moov-2/trak-3/mdia-2/mdhd-1/	-1.346	-0.018	/moov-2/trak-3/mdia-4/minf-3/stbl-4/sdtp-4/	-0.000	-0.019	/moov-2/trak-3/mdia-2/hdlr-2/	-0.011
	... 39 more positive ...		-0.035	/moov-2/trak-3/mdia-2/mdhd-1/	-1.346	-0.023	/beam-2/	2.923
			-0.050	/beam-2/	2.923	-0.029	/moov-2/trak-2/mdia-3/minf-3/dinf-2/dref-1/	3.651

(c) WhatsApp, Flickr, Youtube

Figura 4: Predicción individual de puntos de datos en Redes sociales y aplicaciones de mensajería.

La interpretación de las demás figuras se realiza de forma similar a la anterior. En resumen, una o más clases pueden tener el mismo *PathOrder-tag* con un cierto grado de influencia, pero también es importante la probabilidad que alcanza el conjunto de *PathOrder-tag* positivos, ya que de ambos factores depende una clasificación eficiente de los vídeos con su respectiva clase (Redes sociales y aplicaciones de mensajería instantánea).

VI. Conclusiones

En este trabajo se presenta un método de detección de post-procesamientos en vídeos compartidos a través de redes sociales y aplicaciones de mensajería instantánea que analiza los cambios presentes en la estructura de contenedores multimedia. Para la implementación del método propuesto, en primer lugar, se generó un conjunto de datos de vídeos y posteriormente fueron compartidos a través de las redes sociales y aplicaciones de mensajería instantánea más usados en la actualidad. En segundo lugar, se realizó la extracción del conjunto de características (*PathOrder-tag*) del 100 % de vídeos con el algoritmo de extracción de átomos. En tercer lugar, se construyó un modelo de machine learning supervisado a partir de la evaluación de 4 métricas obtenidas por 9 algoritmos de clasificación, siendo *Random Forest (RF)* el algoritmo que obtuvo el mejor rendimiento. En el análisis de los contenedores multimedia se demostró que cada red social y aplicación de mensajería inserta una estructura propia a los contenedores multimedia, excepto Flick, ya que no realiza un proceso de re-compresión sobre vídeo al momento de ser compartido por su plataforma. Las diferencias detectadas en cuanto a número de átomos es considerable (Facebook HD: 267, Youtube: 250, WhatsApp: 237, Linkedin: 258, Telegram: 251), y en cuanto a valores es mucho más evidente.

En el experimento de identificación de redes sociales y aplicaciones de mensajería instantánea se alcanzó el 100 % de acierto para todas las clases, excepto Instagram que se confunde un 0.31 % con Facebook Msn y Facebook SD se confunde un 0.02 % con Tumblr.

En lineas generales, el método propuesto utilizando la estructura de contenedores multimedia es eficiente y robusto para detectar las redes sociales y aplicaciones de mensajería instantánea en vídeos con formato MP4 y MOV.

Agradecimientos

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700326. Website: http://ramses2020.eu. This paper has also received funding from THEIA (Techniques for Integrity and authentication of multimedia files of mobile devices) UCM project (FEI-EU-19-04).

Referencias

[1] CISCO, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper," February 2019.

[2] HOOTSUITE, " Global Digital Reports 2019," pp. 1–221, February 2019.

[3] J. Kaur and N. Sharma, "Survey on the General Concepts of MPEG Moving Picture Experts Group," *Paripex: Indian Journal of Research*, vol. 5, no. 2, pp. 252–255, February 2016.

[4] B. G. Haskell, P. A., and N. A. N., *Digital Video: An Introduction to MPEG-2 Digital Multimedia Standards*. Orlando, FL, USA: Springer US, 2007.

[5] S. Dhanani and M. Parker, *Digital Video Processing for Engineers: A Foundation for Embedded Systems Design*. Newton, MA, USA: Newnes, 2012.

[6] I. T. Union, "Advanced Video Coding for Generic Audiovisual Services H.264," 2016. [Online]. Available: http://www.itu.int/

[7] International Telecommunication Union, "High Efficiency Video Coding," 2018. [Online]. Available: http://www.itu.int/

[8] Google, "Android-OS," 2019. [Online]. Available: https://www.android.com/

[9] Q. F. Format, "QuickTime File Format Specification," 2016. [Online]. Available: https://developer.apple.com

[10] Apple Inc., "iOS," 2019. [Online]. Available: www.apple.com/es/ios

[11] Microsoft, "AVI RIFF File," 1992. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/directshow/avi-riff-file-reference

[12] Matroska, "Matroska Specification," 2002. [Online]. Available: https://www.matroska.org/technical/specs/index.html

[13] Tengku Mohd T. S. and Halimah Badioze Z. and Hsinchun C. and Shalini R. U. and Sung Hyon M., "Digital libraries: Technology and management of indigenous knowledge for global access," in *Proceedings of the 6th International Conference on Asian Digital Libraries*. Springer, Berlin, Heidelberg, December 2003, pp. 76–83.

[14] F. Camastra and A. Vinciarelli, *Machine Learning for Audio, Image and Video Analysis*.

[15] J. Lukas, J. Fridrich, and M. Goljan, "Digital Camera Identification from Sensor Pattern Noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, June 2006.

[16] C. Li, "Source Camera Identification Using Enhanced Sensor Pattern Noise," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 280–287, June 2010.

[17] Z. J. Geradts, J. Bijhold, M. Kieft, K. Kurosawa, K. Kuroki, and N. Saitoh, "Methods for Identification of Images Acquired with Digital Cameras," in *Proceedings on Enabling Technologies for Law Enforcement and Security*, Boston, Massachusetts, USA, February 2001.

[18] Van Lanh, T. and Chong, K. S. and Emmanuel, S. and Kankanhalli, M. S., "A Survey on Digital Camera Image Forensic Methods," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, Beijing, July 2007, pp. 16–19.

[19] Costa, F. O. and Eckmann, M. and Scheirer, W. J. and Rocha, A., "Open Set Source Camera Attribution," in *Proceedings of the 25th Conference on Graphics, Patterns and Images*, Ouro Preto, Brazil, August 2012, pp. 71–78.

[20] Li, J. and Ma, B. and Wang, C., "Extraction of PRNU Noise From Partly Decoded Video," *Journal of Visual Communication and Image Representation*, vol. 57, pp. 183–191, November 2018.

[21] I. Amerini, R. Caldelli, A. Del Castillo, 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, September 2017.

[22] A. Sandoval Orozco, D. Arenas González, J. Rosales Corripio, L. García Villalba, and J. C. Hernandez-Castro, "Techniques for Source Camera Identification," in *Proceedings of the 6th International Conference on Information Technology*, Amman, Jordan, May 2013, pp. 1–9.

[23] J. Song, K. Lee, W. Y. Lee, and L. H., "Integrity Verification of the Ordered Data Structures in Manipulated Video Content," *Digital Investigation*, vol. 18, no. C, pp. 1–7, Septiembre 2016.

[24] Gloe, T. and Fisher, A. and Kirchner, M., "Forensic Analysis of Video File Formats," in *Proceedings of the First Annual DFRWS Europe*, Munster, Germany, May 2014, pp. 68–76.

[25] M. Iuliani, D. Shullani, M. Fontani, M. S., and A. Piva, "A video forensic framework for the unsupervised analysis of mp4-like file container," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 635–645, March 2018.

[26] Annies, S., "MP4 Parser." [Online]. Available: https://github.com/sannies/mp4parser

[27] S. Maldonado, J. Lopez, and C. Vairetti, "An alternative smote oversampling strategy for high-dimensional datasets," *Applied Soft Computing*, vol. 76, pp. 380–389, 2019.

[28] D. Freeman and C. Chio, *Machine Learning and Security*. Boston, USA: O'Reilly Media, February 2018.

[29] KAGGLE, "KAGGLE: Online Community of Data Scientists and Machine Learners." [Online]. Available: https://www.kaggle.com/

[30] G. Rebala, A. Ravi, and S. Churiwala, *An Introduction to Machine Learning*, 1st ed. Switzerland: Springer, January 2019.

[31] D. Sarkar, R. Bali, and T. Sharma, *Practical Machine Learning with Python*, 2nd ed. Massachusetts, USA: Apress, December 2017.
[32] S. Rachka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow*, 2nd ed. Birmingham, United Kingdom: Packt Publishing, September 2017.
[33] S. Guido and A. C. Muller, *Introduction to Machine Learning with Python*. USA: O'Reilly Media, December 2017.
[34] ELI5, "ELI5 for Python," 2019. [Online]. Available: https://eli5.readthedocs.io/en/latest/overview.html

Carlos Quinto Huamán received his Computer Science degree in 2012 at Universidad Inca Garcilaso de la Vega in Lima (Perú) and a M.Sc. degree in Computer Science in 2016 from the Universidad Complutense de Madrid (Spain). He is currently a Ph.D. student in the Department of Software Engineering and Artificial Intelligence of the Faculty of Computer Science and Engineering at the Universidad Complutense de Madrid (UCM) and member of the Complutense Research Group GASS (Group of Analysis, Security and Systems, http://gass.ucm.es). His research interests are: computer forensics, cybersecurity, electronic warfare and cyberdefense.

Daniel Povedano Álvarez received his Computer Science Engineering degree in 2017 at Universidad Complutense of Madrid. He is currently a M.Sc. student of Data Science in Universidad Complutense de Madrid and member of the Complutense Research Group GASS (Group of Analysis, Security and Systems, http://gass.ucm.es). His research interests are: computer forensics, data science, cybersecurity, artificial intelligence.

Ana Lucila Sandoval Orozco was born in Chivolo, Magdalena, Colombia in 1976. She received a Computer Science Engineering degree from the Universidad Autónoma del Caribe (Colombia) in 2001. She holds a Specialization Course in Computer Networks (2006) from the Universidad del Norte (Colombia), and holds a M.Sc. in Research in Computer Science (2009) and a Ph.D. in Computer Science (2014), both from the Universidad Complutense de Madrid (Spain). She is currently a postdoctoral researcher and member of the Research Group GASS (Group of Analysis, Security and Systems, http://gass.ucm.es) at Universidad Complutense de Madrid (Spain). Her main research interests are coding theory, information security and its applications.

Luis Javier García Villalba received a Telecommunication Engineering degree from the Universidad de Málaga (Spain) in 1993 and holds a Ph.D. in Computer Science (1999) from the Universidad Politécnica de Madrid (Spain). Visiting Scholar at COSIC (Computer Security and Industrial Cryptography, Department of Electrical Engineering, Faculty of Engineering, Katholieke Universiteit Leuven, Belgium) in 2000 and Visiting Scientist at IBM Research Division (IBM Almaden Research Center, San Jose, CA, USA) in 2001 and 2002, he is currently Associate Professor of the Department of Software Engineering and Artificial Intelligence at the Universidad Complutense de Madrid (UCM) and Head of Complutense Research Group GASS (Group of Analysis, Security and Systems) which is located in the Faculty of Computer Science and Engineering at the UCM Campus. His professional experience includes the management of both national and international research projects and both public (Spanish Ministry of R&D, Spanish Ministry of Defence, Horizon 2020 - European Commission, . . .) and private financing (Hitachi, IBM, Nokia, Safelayer Secure Communications, TB Solutions Security, . . .). Author or co-author of numerous international publications is editor or guest editor of numerous journals such as Entropy MPDI, Future Generation Computer Systems (FGCS), Future Internet MDPI, IEEE Latin America Transactions, IET Communications (IET-COM), IET Networks (IET-NET), IET Wireless Sensor Systems (IET-WSS), International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), International Journal of Multimedia and Ubiquitous Engineering (IJMUE), Journal of Supercomputing, Sensors MDPI, etc.