Video tracking for marketing applications

Mohamed Allouche

INSTITUT POLYTECHNIQUE DE PARIS

TELECOM SudParis

IP PARIS

NNT : 2024IPPAS033

Thèse de doctorat

Video tracking for marketing applications

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Telecom SudParis

École doctorale n°626 de
l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Informatique

Thèse présentée par

Mohamed Allouche

Composition du jury :

M. Titus Zaharia Professeur, Télécom SudParis	Président du jury
Mme. Amel Benazza Professeur, SupCom Tunis, Tunisie	Rapporteur
M. Chaker Larabi Professeur, Université de Poitiers	Rapporteur
M. William Puech Professeur, Université de Montpellier	Examinateur
Mme. Federica Battisti MdC, University of Padova, Italie	Examinateur
M. Jean Le Feuvre Directeur d'Etudes, Télécom Paris	Examinateur
M. Mihai Mitrea Professeur, Télécom SudParis	Directeur de thèse
M. Laurent Breboin CEO, VIDMIZER	Invité
M. Erwan Huhardeaux CIO, VIDMIZER	Invité

Mohamed Allouche was affiliated with VIDMIZER (https://cutz.cloud/) during his PhD program. VIDMIZER provided anonymized databases as well as precious industrial feedback for our study. Yet, the present research study was conducted in the absence of any commercial or financial constraint that could be considered as bias in the results here presented.

Acknowledgements

This thesis becomes a reality with the support and help of many people to whom I would like to express my sincere thanks and acknowledgment.

Firstly, my deep gratitude goes to my thesis director, Prof. Mihai Mitrea for his warm welcome my first day at Telecom SudParis. I would like to express my appreciation for his trust and for seeing in me a future PhD. I would also like to thank him his valuable guidance, timely suggestions, infinite motivation and support throughout not my PhD thesis and master internship. I can surely say that I wouldn't be the person who I am today without Prof Mihai Mitrea in multiple levels.

My deep gratitude also goes to the distinguished members of my defense committee, and particularly to the two reviewers, Prof. Amel Benazza and Prof. Chaker Larabi, for their precious feedback and enriching comments that contributed to the final version of this manuscript.

I am also grateful to Prof. Jenny Benois-Pineau and Prof. Azeddine Beghdadi, the members of my CSI committee, for their professional constructive feedback and thoughtful contributions to my work during my PhD progress.

I would like to thank all the VIDMIZER teams for the great moments we spent together and for the learning opportunities, they offered me.

My colleagues Marina Ljubojevic, Carl De Sousa Trias, Alexendre Moreaux, and Mateo Zoughebi deserve a special mention: I thank them for their availability during this thesis, the deep enriching discussion and work collaborations. An extra special thanks to Carl who introduced me to running and coached me throughout my first races.

I would like to thank Mrs. Evelyne Taroni for her proactive attitude and valuable administrative help during my master and PhD at the ARTEMIS department.

In addition, I like to thank the entire ARTEMIS team, former and present members that I have met.

I am mostly fortunate to have the opportunity to acknowledge gratitude to the people who mean the most to me. My parents Abd Alkarim and Mouna, who raised me, taught me, and supported me all throughout my life: their selfless love, care, and sacrifices shaped my life.

I like to deeply thank my brother Ahmed and my sister Mariem for their motivational discussions and emotional support.

For my friends and for all those who have touched my life in any way since I started this thesis, I am grateful for all what they have done.

Résumé

Au cours des dernières décennies, la production et la consommation du contenu vidéo a considérablement augmenté et il est communément admis que 80 % du trafic Internet est constitué par de vidéos. Dans ce cadre, la distribution des vidéos publicitaires est encore dominée par le contenu payant (c'est-à-dire le contenu créé par une agence média qui paie un annonceur pour distribuer ce contenu). Cependant, le contenu *vidéo organique* progresse lentement mais sûrement. Le terme « contenu organique » fait référence à un contenu dont la création et/ou la distribution n'est pas payante. Dans la plupart des cas, il s'agit d'un contenu créé par l'utilisateur, avec une valeur publicitaire implicite, ou d'un contenu publicitaire distribué par un utilisateur sur un réseau social. En pratique, un tel contenu est directement produit dans un format compressé (par exemple AVC - Advanced Video Coding, HEVC - High efficiency Video Coding ou VVC - Versatile Video Coding) et est souvent partagé par d'autres utilisateurs, sur le même réseau social ou sur des réseaux sociaux différents, créant ainsi une chaîne virtuelle de distribution qui est étudiée par les experts en marketing.

Une telle application peut être modélisée par au moins deux cadres scientifiques différents, à savoir la blockchain et l'empreinte (*fingerprinting*) vidéo. D'une part, si l'on considère d'abord les problèmes de distribution, la blockchain semble être une solution attrayante, car elle prévoit une solution sécurisée, décentralisée et transparente pour suivre les changements de tout actif numérique. Alors que la blockchain a déjà prouvé son efficacité dans une grande variété d'applications de distribution de contenu, ses applications liées au multimédia restent rares et soulèvent des contradictions conceptuelles entre les limitations des ressources de calcul/stockage disponibles dans la blockchain et la grande quantité de données et les opérations complexes que le traitement vidéo exige. D'autre part, si l'on considère d'abord les questions relatives au contenu multimédia, chaque étape de la distribution peut être considérée comme une opération de quasi-doublonnage (*near-duplicate content*). Ainsi, le suivi d'une vidéo organique peut être assuré par le fingerprinting vidéo qui regroupe les efforts de recherche consacrés à l'identification des versions dupliquées et/ou répliquées d'une séquence vidéo dans un ensemble de données vidéo de référence. Alors que le suivi du contenu vidéo dans le domaine non compressé est un domaine de recherche riche, le fingerprinting vidéo dans le domaine compressé est encore sous-exploré.

La présente thèse étudie la possibilité de tracer un contenu vidéo compressé, dans le contexte de sa propagation spontanée et incontrôlée dans un réseau distribué, à travers trois aspects principaux :

- le suivi vidéo au moyen de solutions basées sur la blockchain, malgré la grande quantité de données et les exigences de calcul des applications vidéo, *a priori* incompatibles avec les solutions blockchain actuelles,
- le fingerprinting vidéo dans le domaine compressé, même si la compression vidéo est censée exclure la redondance visuelle qui permet de retrouver le contenu vidéo,
- les synergies applicatives entre la blockchain et le fingerprinting vidéo.

Les principaux résultats consistent en la conception, la spécification et la mise en œuvre de :

- COLLATE – une architecture de répartition du calcul et du stockage *on-chain* / *off-chain*, qui permet d'étendre de manière abstraite les ressources informatiques limitées de n'importe quelle blockchain par des ressources informatiques à usage général ;
- COMMON – Compressed dOMain Marketing videO fiNgerprinting, qui démontre la possibilité de modéliser des empreintes vidéo compressées dans un cadre d'apprentissage profond ;
- BIDDING – BlockchaIn-baseD viDeo fINgerprintinG, un pipeline de traitement de bout en bout qui permet de coupler l'empreinte vidéo à la solution d'équilibrage de charge de la blockchain.

Abstract

The last decades have seen video production and consumption rise significantly: TV/cinematography, social networking, digital marketing, and video surveillance incrementally and cumulatively turned video content into the predilection type of data to be exchanged, stored, and processed. It is thus commonly considered that 80% of the Internet traffic is video, and intensive and holistic efforts for devising lossy video compression solutions are carried out to reach the trade-off between video data size and their visual quality.

Under this framework, marketing videos are still dominated by the paid content (that is, content created by the advertiser that pays an announcer for distributing that content). Yet, *organic video* content is slowly but surely advancing. In a nutshell, the term organic content refers to a content whose creation and/or distribution is not paid. In most cases, it is a user-created content with implicit advertising value, or some advertising content distributed by a user on a social network. In practice, such a content is directly produced by the user devices in compressed format (*e.g.* the AVC – Advanced Video Coding, HEVC – High efficiency Video Coding or VVC – Versatile Video Coding) and is often shared by other users, on the same or on different social networks, thus creating a virtual chain distribution that is studied by marketing experts.

Such an application can be modeled by at least two different scientific methodological and technical frameworks, namely *blockchain* and *video fingerprinting*. On the one hand, should we first consider the distribution issues, blockchain seems an appealing solution, as it makes provisions for a secure, decentralized, and transparent solution to track changes of any digital asset. While blockchain already proved its effectiveness in a large variety of content distribution applications, its multimedia related applications stay scarce and rise conceptual contradictions between the strictly limited computing/storage resources available in blockchain and the large amount of data representing the video content as well as the complex operations video processing requires. On the other hand, should we first consider the multimedia content issues, each step of distribution can be considered as a *near-duplication operation*. Thus, the tracking of organic video can be ensured by video fingerprinting that regroups research efforts devoted to identifying duplicated and/or replicated versions of a given video sequence in a reference video dataset. While tracking video content in uncompressed domain is a rich research field, compressed domain video fingerprinting is still underexplored.

The present thesis studies the possibility of tracking compressed video content, in the context of its uncontrolled, spontaneous propagation into a distributed network, while specifically addressing:

- video tracking by means of blockchain-based solutions, despite the large amount of data and the computation requirements of video applications, *a priori* incompatible with nowadays blockchain solutions
- effective compressed domain video fingerprinting, even though video compression is supposed to exclude the very visual redundancy that allows video content to be retrieved.
- applicative synergies between blockchain and fingerprinting frameworks.

The main results consist in the conception, specification and implementation of:

- COLLATE, an on-Chain Off-chain Load baLancing ArchiTecturE, thus making it possible for the intimately constrained computing, storage and software resources of any blockchain to be abstractly extended by general-purpose computing machine resources;
- COMMON – Compressed dOMain Marketing videO fiNgerprinting, demonstrating the possibility of modelling compressed video fingerprint under deep learning framework
- BIDDING – BlockchaIn-baseD viDeo fINgerprintinG, an end-to-end processing pipeline coupling compressed domain video fingerprinting to the blockchain load balancing solution.

Table of contents

Table of figures

Table of tables

Abbreviation list

ABI	Smart Contract Connector Unit
AI	Artificial Intelligence
API	Application Programming Interface
ARM	Advanced RISC Machines
AUC	Area Under the Curve
AV1	Alliance for open media Video 1
AVC	Advanced Video Coding
BIDDING	BlockchaIn-baseD viDeo fINgerprintinG
CABAC	Context-adaptive binary arithmetic coding
Cb	Chroma blue
CDVS	Compact Descriptors for Visual Search
CMS	Content Management Systems
CNN	Convolutional Neural Networks
COLLATE	Chain Off-chain Load baLancing ArchiTecturE
COMMON	Compressed dOMain Marketing videO fiNgerprinting
CPU	Central Processing Unit
CRBM	Conditional Restricted Boltzmann Machine
Cr	Chroma red
CRF	Constant Rate Factor
CRM	Customer Relationship Management
CS-LBP	Center-symmetric Local Binary Patterns
CTU	Coding Tree Units
DApp	Decentralized Applications
DASH	Dynamic Adaptive Streaming over HTTP
DCT	Discrete Cosine Transform
DL	Deep Learning
DML	Deep metric learning
DNN	Deep Neural Networks
DRM	Digital Rights Management
DST	Discrete Sine Transform
DWT	Discrete Wavelet Transform
ERC	Ethereum Request for Comment
EVM	Ethereum Virtual Machine
GOP	Group Of Pictures
GPU	Graphics processing unit
HEVC	High Efficiency Video Coding
HOG	Histogram of Oriented Gradients
IntraC	Intra prediction mode Chroma
IntraY	Intra prediction mode Luma
IoMT	Internet of Media Things
IPR	Intellectual Property Rights
JSON	JavaScript Object Notation
JWT	JSON Web Tokens
KNN	k-Nearest Neighbors
KPI	Key Performance Indicator
LSTM	Long short-term memory
ML	Machine Learning
MPAI	Moving Picture, Audio and Data Coding by Artificial Intelligence
MPEG	Moving Picture Experts Group
NFT	Non-fungible tokens
NIP	Nested Invariance Pooling

NN	Neural Networks
OCCE	Off-Chain Connector Environment
OCCU	Off-Chain Connector Unit
ORB	Oriented FAST and Rotated BRIEF
PNC	Non Player Character
PoA	Proof of Authority
PoC	Proof of Concept
PoS	Proof of Stake
Pow	Proof of Work
PKI	Public Key Infrastructure
RAM	Random-Access Memory
REST API	Representational State Transfer
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SCCU	Smart Contract Connector Unit
SHA256	Secure Hash Algorithm 256-bit
SIFT	Scale-Invariant Feature Transform
SSL/TLS	Secure Sockets Layer/Transport Layer Security
SURF	Speeded Up Robust Features
TIRI	Temporal Informative Representative Image
URL	Uniform Resource Locator
VBR	Variable Bit-Rate
VVC	Versatile Video Coding
Y	Luma

Chapter I. Overview

The present thesis deals with video content tracking for marketing applications. This chapter identifies the underlying context, the current-day conceptual and methodological limitations, as well as our main contributions. The dissemination activity is also mentioned.

I.1. Context

The last decades have seen video production and consumption rise significantly: TV/cinematography, social networking, digital marketing, and video surveillance incrementally and cumulatively turned video content into the predilection type of data to be exchanged, stored, and processed. It is thus commonly considered that 80% of the Internet traffic is video [CIS18], and intensive and holistic efforts for devising lossy video compression solutions are carried out to reach the trade-off between video data size and their visual quality [OTH20]. As a valiant attempt in reducing the processing complexity, compressed domain video processing techniques are sometimes considered [AMM18; BEN10; HAS14; HEI15; MAN07; SHA11; SHA13].

Video has also become one of the most powerful tools in the digital marketer's arsenal, as it offers the ability to efficiently and seamlessly convey marketing messages, as brought forth in [STA24a] and illustrated in Figure 1.

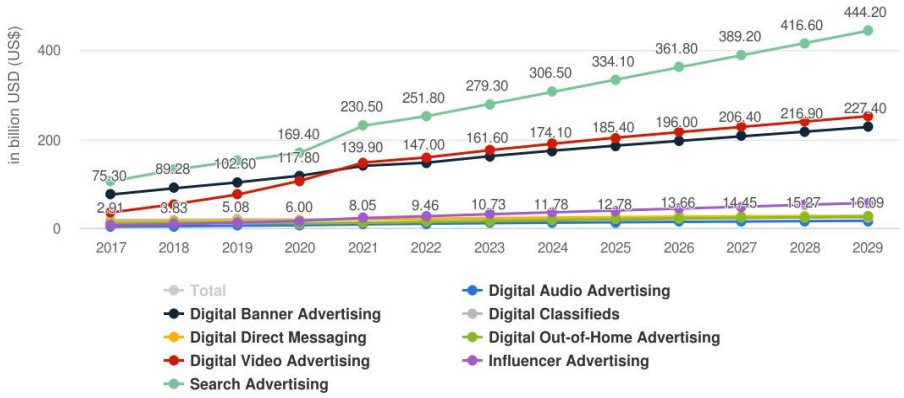


Figure 1: Worldwide advertising spending in billions of USD, between 2017 and 2028 (source: [STA24a])

While historically, video marketing was led by the traditional TV advertising, in the last few years, digital video advertising is taking over, as illustrated in Figure 2. This shift can be explained by the continues growth of the time spent on the different social platforms averaging 147 minutes per day in 2024 compared to 90 minutes in 2012 [STA24d], as well as the social networks penetration across all regions and generations, with an expected 6.05 billion users in 2028, compered to 2.73 billion users back in 2017 [STA24c].

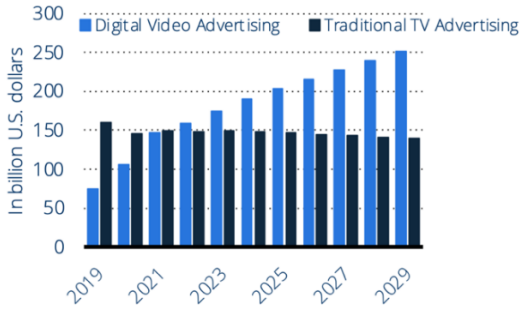


Figure 2: Revenue in the TV & Video Advertising market for different segments Worldwide from 2019 to 2029 (in billion U.S. dollars) (source: [STA24b])

While the digital marketing is still dominated by the paid content (that is, content created by the advertiser and the distribution is paid for, in order to appear for a specific group of users for a specific number of times), *organic video* content is slowly but surely advancing. Organic video marketing refers to the use of video content that reaches audiences naturally, without any paid promotion. It could be in the form of tutorials, behind-the-scenes footage, customer testimonials, or user-generated content, offering a more authentic connection between brands and clients. This implicit method of marketing thrives on platforms where users could share, engage and edit the content.

The organic video content is most likely to be directly produced by the user (or, at least, nonprofessional) devices in one of the widely used compressed format (*e.g.* the AVC – Advanced Video Coding, HEVC – High efficiency Video Coding or VVC – Versatile Video Coding) and is often shared by other users, on the same or on different social networks, thus creating a virtual distribution chain that can be studied by marketing experts, as illustrated in Figure 3.

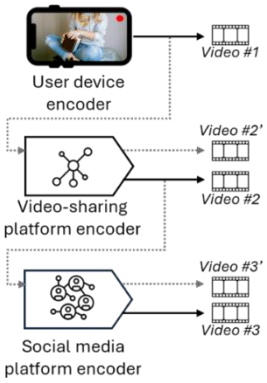


Figure 3: Organic video content distribution: some user-created content, with advertising value for a brand that is not controlling that content, can be shared on video platforms and/or social networks

With all those new technological trends in the digital marketing, the term MarTech saw the light. MarTech is a blend of **Mar**keting and **Tech**nology that was coined to define the integration of cutting-edge digital tools and marketing strategies in the modern business landscape as illustrated in Figure 4. It represents the integration of technology and software platforms in the planning, execution, and evaluation of marketing strategies. This interdisciplinary field encompasses a broad spectrum of tools and systems aimed at enhancing marketing effectiveness and operational efficiency. It spans across multiple domains, including but not restricted to Customer Relationship Management (CRM), Content Management Systems (CMS), social media management, email marketing, analytics, and automation platforms.

MarTech solutions are mainly data-centric applications that focus on the collection, analysis, and visualization of a tremendous amount of data gathered from the social media and video streaming platforms. MarTech tools offer a deeper understanding of consumer behavior, preferences, and can identify emerging trends. Analytics are a vital component of the MarTech ecosystem, offering tools for measuring Key Performance Indicators (KPIs) and assessing campaign success.

Figure 4: MarTech main application scopes (Source: [DAS23])

The raise of the organic video dictates the need for new solutions capable to track the propagation of this type of new content throughout all the social networks. Such an application can be modeled by at least two different scientific, methodological and technical frameworks, namely *blockchain* and video *fingerprinting*.

On the one hand, should we first consider the distribution issues, *blockchain* seems an appealing solution, as it makes provisions for a secure, decentralized, and transparent solution to track changes of any digital asset. While blockchain already proved its effectiveness in a large variety of content distribution applications, from cold chain monitoring [BAD18] to cryptofinance [DOS22], its multimedia related applications stay scarce [QUR20] and rise conceptual contradictions between the strictly limited computing/storage resources available in blockchain and the large amount of data representing the video content as well as the complex operations video processing requires. On the other hand, should we first consider the multimedia content issues, each step of distribution can be considered as a *near-duplication*[1] operation. Thus, the tracking of organic video can be ensured by video *fingerprinting* (also referred to as *content- based copy detection*, *near duplicate detection* or *semantic fingerprinting*) that regroups research efforts devoted to identifying duplicated and/or replicated versions of a given video sequence (query) in a reference video dataset [DOU08, JIA16]. While tracking video content in uncompressed domain is a rich research field, compressed domain video fingerprinting is still underexplored [JIA16, ALL22, CHE24].

The present thesis studies the possibility of tracking advertising compressed video content, in the context of its uncontrolled, spontaneous propagation into a distributed network. Specifically, it explores the possibilities of achieving:

[1] In this manuscript, near-duplicated content is defined according to [BEN10] as *identical or approximately identical videos close to the exact duplicate of each other, but different in file formats, encoding parameters, photometric variations (color, lighting changes), editing operations (caption, logo and border insertion), different lengths, and certain modifications (frames add/remove)*.

- **video tracking by means of blockchain-based solutions, despite the large amount of data and the computation requirements of video applications, *a priori* incompatible with nowadays blockchain solutions**
- **effective compressed domain video fingerprinting, even though video compression is supposed to exclude the very visual redundancy that allows video content to be retrieved.**
- **applicative synergies between blockchain and fingerprinting frameworks.**

I.2. Blockchain-based video tracking

Blockchain is a distributed ledger technology that offers secure, decentralized, and transparent data exchange [CHR16, TSC16, MON19, WAN19, ZHO20]. Each block in the chain records a set of transactions or data entries, and these blocks are cryptographically linked through a process where each block includes the unique hash of the previous one. One main property of this type of hashing is its bit-sensitivity, meaning that even the slightest change to the input, as small as a single bit, results in a completely different hash output. This way, the hash ensures the integrity of the entire chain, breaking the link with the succeeding block in case of alterations. This structure forms an immutable sequence of blocks, as illustrated in Figure 5, where any data tampering would require consensus from the entire network to be accepted. The cryptographic linkage not only secures the blockchain but also preserves its transparency, as all participants can verify and trust that the recorded data has not been altered. As a result, blockchain provides a robust and tamper-resistant system for securely managing data across decentralized environments.

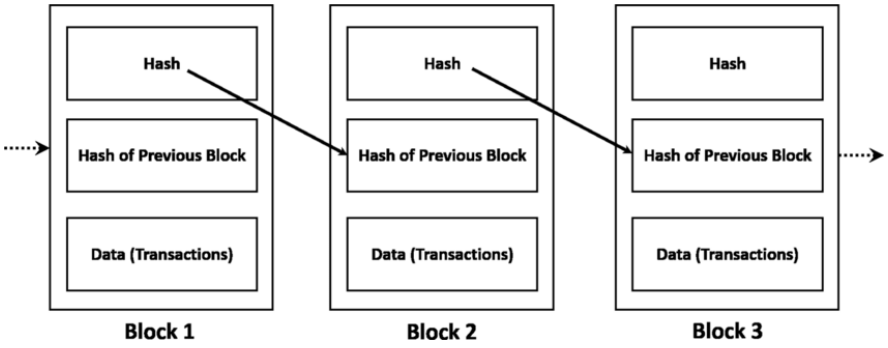


Figure 5:Simplified diagram of the cryptographical link between blocks (source [BOS19])

In the context of video tracking, blockchain's primary advantage is its ability to provide an immutable record of content ownership, distribution, and usage. Each video can be assigned a unique identifier or hash that is stored on the blockchain, ensuring that the video's origin and subsequent changes in ownership or distribution are recorded and accountable. Specifically for marketing applications, blockchain can be used to track the distribution of promotional videos across various platforms: the video identification information stored in blockchain allows marketers to monitor where the content under investigation has been distributed, viewed, *etc*. This also allow for any illicit duplication or redistribution of the video to be spotted out, as for instance, alcohol-related organic video content for teenagers.

While blockchain offers various advantages for video tracking, this is not without challenges.

First, the very content authentication in blockchain is based on digital hashing functions that produce a fixed-length digest output for any given input (*e.g.* 256 bits for SHA256 function in use today [LIU23]) for any content to be identified. The bit sensitivity of the hash makes it highly secure for content authentication but cannot cope to the applicative constraints of visual content distribution, where transcoding, resizing, cropping, *etc*.

occurring during distribution produce entirely new hash values, rendering it ineffective for tracking modified media as shown in Figure 6 for the simplified case of images.

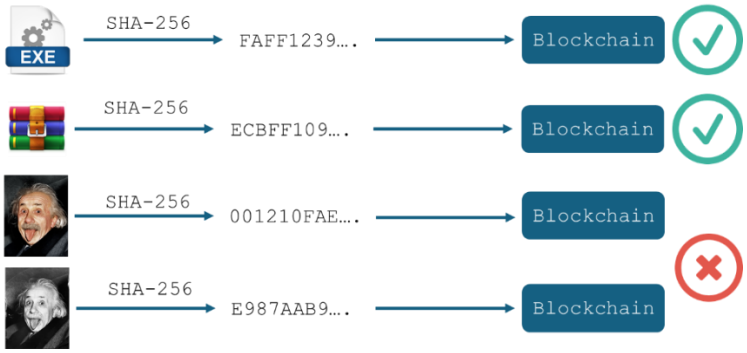


Figure 6: Blockchain solutions are design to detect any modification in the recorded content, regardless of its impact in the content semantics

Secondly, blockchain technologies impose strong computational constraints to ensure its well-functioning. Every operation, whether it is financial transaction, data insertion, or Smart Contract execution, must be verified and validated across a network of nodes. This decentralized validation guaranties the transparency and the immutability along the network. However, these advantages impose the computational limitation, especially when complex applications, such as the video tracking, are considered. Each blockchain platform has adopted strict algorithmic and architectural design rules in order to establish the balance between security, scalability and cost-efficiency. For instance, Ethereum, one of the most popular blockchain in use today [WAN19; CHE21] and the second largest blockchain network by market capitalization in 2024, requires all its Smart Contracts to be written in Solidity, a programming language tailored for the Ethereum Virtual Machine (EVM) [GIT24a]. By design, Solidity restricts many of the features that can be found in traditional programming languages like C, JAVA, and Python to reduce the excessive computational load on the network. For instance, it does not allow floating-point arithmetic that is computationally expensive. The use of fixed-point arithmetic can solve this design choice, but it presents its own limitations in term of precision, range, and code readability and maintenance. Another main aspect of Solidity's design is the restriction on certain arithmetic operations like the division. Although division is technically possible in Solidity, it is discouraged due to a drastic increase of the usage cost and gas costs (gas is the unit of complexity work needed to execute an operation in the blockchain) which directly translates into expensive transactions. Like Ethereum, Tezos is also a prominent blockchain platform that faces the same computational challenges through its proprietary programming language named Liquidity [GIT24b]. Liquidity follows a functional programming paradigm that limits the complexity of operations that can be executed *on-chain* by restricting resource-intensive processes, such as deep recursion or extensive loops. It is designed to ensure that Smart Contracts remain lightweight and cost-effective. The functional paradigm, while efficient for the blockchain integrity, may not be well suited for sophisticated applications that require iterative processes or complex data structures.

Thirdly, the decentralized application (DApp) scalability is frequently considered among the most significant challenges. As the number of transactions on a blockchain increases, the computational effort required validating and processing the new blocks increases as well. This high demand can lead to delays, particularly in the case of video tracking applications coming across with millions of views and interactions in near real-time: each view, like, comment, or share would represent a new transaction that must be recorded on the blockchain! The high volume of interactions can overwhelm the blockchain, resulting in bottlenecks and repetitive delays in block mining process. This makes it difficult to maintain an accurate, real-time ledger for applications such as copyright enforcement or content monetization.

Finally, storing the video fingerprints can become a significant functional issue, due to the blockchain's reliance on gas fees, calculated based on the computational work and the size of the data being stored. For instance, gas fees in Ethereum are correlated with the amount of data being processed and transacted, meaning that as fingerprint size grows, so does the cost to store it. Even though video fingerprints are much smaller than full video sequences, their accumulation over time in a large-scale system becomes prohibitive. Moreover, storing the video fingerprints *on-chain* could face block size limitations since Blockchain blocks can only handle a finite amount of data, and transactions that include larger amounts of data.

On this research item, the thesis contributions consist in:

- ***methodological level*: conception, specification and implementation of COLLATE, an *on-Chain Off-chain Load baLancing ArchiTecturE*, thus making it possible for the intimately constrained computing, storage and software resources of any blockchain to be abstractly extended by general-purpose computing machine resources;**
- ***experimental level*: specifying an experimental testbed and carrying out the underling experiments for achieving the proof-of-concepts for visual content tracking on lightweight computing resources (namely, an ARM multiprocessor embedded platform, integrated into a Raspberry Pi), with illustrations for Ethereum and Tezos.**

I.3. Video Fingerprinting

Video fingerprints are compact, unique digital identity representations that are obtained by analyzing key features from a video sequence. Such fingerprints are designed to identify video sequences reliably, even if the video has undergone a variety of transformations such as resizing, re-encoding, or modifications in contrast, white balance or color. Unlike metadata or watermarks, which are external identifiers, fingerprints are inherent to the video content itself and are extracted from its spatial, temporal, or visual characteristics as illustrated in Figure 7.

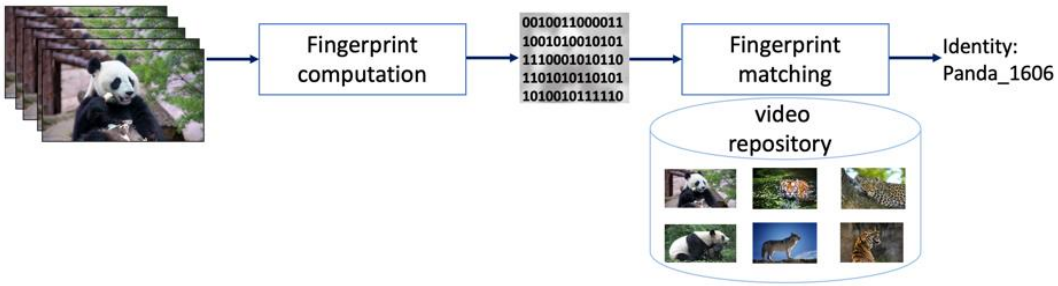


Figure 7: Video fingerprinting concept

Video fingerprint purpose is to offer a mechanism for visual content identification without needing to store the entire video sequence. This is particularly important in contexts such as copyright protection, Digital Rights Management (DRM), video content distribution platforms or fake news detection. One of the core strengths of video fingerprinting lies in its ability to match transformed versions of the same video, offering flexibility when content is shared across platforms that compress or alter the file in various ways.

With a two-decade history, video fingerprinting is now a mature research field, with approaches targeting a joint functional optimization of the fingerprinting extraction and retrieving procedures. To this end, various methodological frameworks, from information theory to machine learning (including deep learning) are explored [JIA16; ALL22; CHE24]. Despite their conceptual and applicative varieties, they all operate (at least partially) at the pixel level, after the stream decoding[2]. In early fingerprinting systems, visual features such as edges, corners, or motion patterns are extracted directly from the pixel data. These features are then optimized to shape a compact representation that can be stored and later used for retrieval and comparison. The challenge here is to find a balance between generating fingerprints that are both unique (able to distinguish between different videos) and robust (able to match transformed versions of the same video).

The fingerprinting conventional methods involve two core stages: fingerprint extraction and fingerprint retrieval. The extraction stage starts by video pre-processing, such as frame resizing, frame dropping, key-frame detection, and color modifications, which prepare the video for feature extraction. Local features can be derived using techniques like HOG (Histogram of Oriented Gradients), ORB (Oriented FAST and Rotated BRIEF),

[2] This state-of-the art situation is different for video indexing, where previous attempts exploiting the compressed-domain information can be encountered [BEN10].

time alignment operations (such as time origin synchronization and jitter cancellation) to ensure consistency across different video versions. The extracted information is then compared using similarity measures like Hamming distance, Euclidean norms, correlation coefficients, or maximum a posteriori probability.

The deep learning-based methods leverage on the neural networks for implicitly learning the visual salient features of the content and for subsequently classify the queries in the corresponding classes. To this end, a large variety of models are considered, individually or combined. Just for illustration, spatial information can be addressed by AlexNet, VGG, ResNet, while temporal information by structures based on Long short-term memory (LSTM): Siamese LSTM, BiLSTM, ...

As video content is predominantly recorded, stored, and transmitted in compressed formats, achieving fingerprinting directly at the level of compressed domain offers significant computational advantages, particularly when working with large-scale video datasets.

To our knowledge, one of the earliest studies exploring compressed domain fingerprinting introduced a method that computes fingerprints based on both pixel-domain and MPEG-2 stream information [CHO05]. Specifically, this method combines frame color histograms, ORB descriptors, and motion vector normalized histograms. Each of these components is individually matched using appropriate criteria, and the overall decision is reached by fusing the results from multiple features using a weighted additive voting model. This approach allows for a hybrid representation of both spatial and temporal video characteristics while still leveraging the compressed domain's efficiency.

Since then, a few other studies have advanced similar ideas [JIA16; ALL22], yet the field remains relatively underexplored. It is worth noticing that the interest in identifying compressed video streams extends beyond traditional fingerprinting use cases. For example, identifying and tracking video streams delivered on specific platforms, such as YouTube or Netflix, has become an emerging area of research, given the growing demand for securing video content across different delivery channels [AFA22].

While compressed domain video fingerprinting offers promising computational advantages, several technical challenges must be addressed. One of the primary obstacles is the inherent loss of visual information that occurs during video compression. Lossy video encoding algorithms, such as AVC, HEVC or VVC, aim to eliminate redundant data from the stream representation to reduce stream size. This contradicts the very fingerprinting principle according to which visual data redundancy is explored for tracking the content. Consequently, designing robust fingerprinting techniques that can extract sufficient relevant information from compressed streams without compromising accuracy is a key research focus particularly as video compression becomes more aggressive to optimize bandwidth.

Another challenge is the integration of compressed domain fingerprinting with deep learning-based methods. The same kind of conceptual contradiction also arise here, as deep learning models typically exploit data redundancy to learn effective feature representations, whereas video encoders are designed to minimize redundancy for compression purposes. Bridging this gap requires innovative architectures and learning

strategies that can operate effectively within the constraints of compressed data while still maintaining the discriminatory power required for fingerprinting tasks.

The thesis contributions consist in a comprehensive methodological and experimental study on the possibility of achieving effective compressed domain video fingerprinting:

- ***at the methodological level*, both the fingerprinting extraction and fingerprinting matching are investigated, thus obtaining *COMMON – Compressed dOMain Marketing videO fiNgerprinting*:**
  - **the optimal steam syntax elements *a priori* likely to represent the fingerprint are studied and identified,**
  - **the possibility of modelling compressed modeling video fingerprint under conventional DL frameworks is studied and the underlying end-to-end processing pipe-line are advanced by reconsidering and extended current-days DL architectures;**
- ***at the experimental level*, the prof of concepts for marketing video content fingerprinting is achieved:**
  - **an experimental testbed is specified: 164 original marketing video sequences, 100 near-duplicated modifications of 10 typologies applied to each original sequence, 5 encoding formats (AVC, HEVC, VVC, VP9 and AV1) for any original and near-duplicated sequence,**
  - **experiments are performed and *Accuracy*, *Precision* and *Recall* values larger than 90% are reported,**
  - **methodological invariance with respect to the encoding format is brought forth.**

I.4. Blockchain-fingerprinting applicative synergies

Traditional video fingerprinting systems rely on centralized databases, where the digital signatures of video content are stored and verified. However, centralized architectures present several vulnerabilities that can cause trust loss of the entire system. Firstly, centralized databases are prone to hacking. If a malicious actor manages to gain access to fingerprints database, they can potentially alter or delete critical information, compromising the integrity of the retrieval system. Secondly, tampering with the database can result in unapproved changes to content verification mechanisms, undermining the authenticity of the video fingerprints. Additionally, data loss is also a concern, due to system failures, corruption, or mismanagement of storage facilities. Centralized databases are vulnerable to hardware malfunctions, natural disasters, or even human error, where entire databases can become permanently inaccessible leading to the loss all stored video fingerprints.

As earlier explained, blockchain have a decentralized architecture, that eliminates the single point of failure risks and the need for a central authority, making it more resistant to different type of attacks as long as each one of active nodes of the network holds a copy of the blockchain, ensuring that there is no single point of failure. This decentralization not only makes blockchains more resilient to classic attacks but also ensures that the data is duplicated across multiple locations around the globe**,** providing a much higher level of fault tolerance. Furthermore, blockchain's consensus mechanisms ensure that any changes to the data require approval from the majority of participants. This guarantees the immutability of the stored video fingerprints, offering a much higher level of data integrity than centralized systems can provide. Additionally, blockchain eliminates the need for a central authority to manage the system. This eliminates the potential risk for abuse of power since all nodes having equal control over the ledger and nullify the risk of single-point governance failure.

Consequently, the possibility of establishing functional synergies between video fingerprinting and blockchains has also been studied in the present thesis.

The thesis contribution consists in:

- ***at the methodological level*, the definition of *BIDDING – BlockchaIn-baseD viDeo fINgerprintinG,* an end-to-end processing pipeline for coupling compressed domain video fingerprinting to the blockchain load balancing solution.**
- ***at the experimental level*, the proof of concepts for the effectiveness of the solutions, obtained on the same database, and in both PC and embedded blockchain (Ethereum, Tezos) environments.**

I.5. Summary

The main thesis contributions are regrouped in Table 1 and synoptically presented in Figure 8.

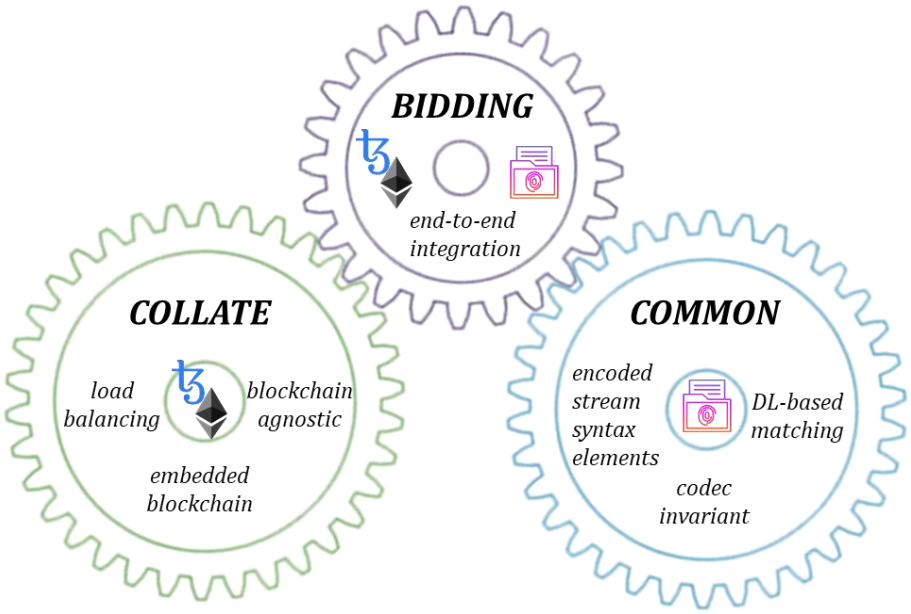


Figure 8: Synopsis of thesis's contributions: (1) COLLATE, an on-Chain Off-chain Load balancing, (2) COMMON - Compressed dOMain Marketing videO fingerprinting and (3) BIDDING – BlockchaIn-baseD viDeo fINgerprintinG

Note that all the AI-related experiments are carried out by considering TensorFlow frameworks, namely TensorFlow v2.10.0 for server-oriented setups and tflite-runtime v2.14.0 for embedded setups.

The dissemination activities cover journal and conference papers, as well as ISO standard contributions.

Journal papers:

- Allouche, M., Mitrea, M., 2022. Video fingerprinting: Past, present, and future. Front. Signal Process. 2, 984169. https://doi.org/10.3389/frsip.2022.984169
- Allouche, M., Mitrea, M., Moreaux, A., Kim, S.-K., 2021. Automatic Smart Contract generation for Internet of Media Things. ICT Express 7, 274–277. https://doi.org/10.1016/j.icte.2021.08.009
- Allouche, M., Frikha, T., Mitrea, M., Memmi, G., Chaabane, F., 2021. Lightweight Blockchain Processing. Case Study: Scanned Document Tracking on Tezos Blockchain. Applied Sciences 11, 7169. https://doi.org/10.3390/app11157169

Conference papers:

- Allouche, M., Mitrea, M., De Sousa Trias, C., HEVC Compressed Video Fingerprinting, accepted for IS&T Electronic Imaging, 2025.
- Allouche, M., Colle, E., Zoughebi, M., De Sousa Trias, C., Mitrea, M., Green video encoder identification, accepted for IS&T Electronic Imaging, 2025.
- Allouche, M., Ljubojevic, M., Mitrea, M., 2021. Visual document tracking and blockchain technologies in mobile world. Electronic Imaging 33, 1–7.

- Allouche, M., Mitrea, M., De Sousa Trias, C., 2023. HEVC fingerprinting: conventional vs. ML methods. Case study: organic video content. TAIMA 2023.

Contributions for ISO standards

31 technical contributions for ISO/IEC JTC 1 SC29, reflected in the specification of a new standard ISO/IEC 21000-23 Smart Contract for Media and in a new edition of an already existing standard, ISO/IEC 21000-22 MPEG User Description.

Table 1: Marketing video tracking issues, challenges, pain points, and thesis contributions

Issues	Challenges	State of the art pain points	Thesis contributions

1 Blockchain based video tracking	Ensure video identification in blockchain environments	1. State of the art blockchains considers bit sensitive hash functions (SHA256)
		2. Blockchain computing and storage environments are currently limitative
			Main result:
			• **methodological: COLLATE, an *on-Chain Off-chain Load baLancing ArchiTecturE***
			  • on chain/off chain load balancer capable of seamlessly operate high complexity work load in any decentralized application (DApp)
			  • Integration of multiple blockchain nodes in an embedded device
			• **experimental: proof-of-concepts for visual content tracking on lightweight computing resources**
			
			Dissemination & outcomes:
			• [ALL21a],[ALL21b]
			• ISO standardization contributions

	Develop blockchain agnostic architecture that allows bidirectional data exchange between different blockchain environments and general purpose computing environments	The amount of data that can be exchanged between blockchain components is limited, restricting the versatility of applicative workflows	
	Efficient usage of blockchain on embedded devices	Blockchain is well known of its huge energy consumption and the need of high computational powers for block generation	

2 Compressed domain video fingerprinting	Define the fingerprint to be extracted from the compressed stream syntax elements	Compressed domain content is *a priori* likely to eliminate the visual redundancy that would allow the visual content to be identified	**Main result:**
			• **methodological: COMMON – Compressed dOMain Marketing videO fingerprinting:**
			  • the optimal steam syntax elements *a priori* likely to represent the fingerprint are studied and identified,
			  • the possibility of modelling compressed modeling video fingerprint under conventional DL frameworks is shown
			• **experimental: an experimental testbed is specified:**
			  • Syntax element parser for MPEG-4 AVC and HEVC codecs capable of extracting the Luma and Chroma components as well as the prediction modes from all the macroblocks

	Identify DL structures able to match fingerprints in the compressed domain	Usual DL methods benefits from the information redundancy to learn patterns; no prior study on the performance of those algorithms with the data extracted from compressed domain exists	
	Proof of concept for the compressed domain fingerprinting system.	An efficient fingerprint method should ensure the: - unicity - robustness - efficiency	

			• 164 original marketing video sequences, 100 near-duplicated modifications of 10 typologies applied to each original sequence, 5 encoding formats (AVC, HEVC, VVC, VP9 and AV1) for any original and near-duplicated sequence, • experiments are performed and *Accuracy*, *Precision* and *Recall* values larger than 90% are reported, • methodological invariance with respect to the encoding format is brought forth. **Dissemination & outcomes:** • [ALL22] • [ALL23], [ALL25a], [ALL25b]
3. Blockchain-fingerprinting applicative synergies	Develop a zero-trust system for video tracking	Both blockchain and video fingerprinting have interesting features that can help solve the video tracking problem but they were proposed as separate solution	**Main result:** • **methodological**: ***BIDDING – BlockchaIn-baseD viDeo fINgerprintinG***, end-to-end processing pipeline for coupling compressed domain video fingerprinting to blockchain • **experimental**: in both PC and embedded blockchain (Ethereum, Tezos) environments. **Dissemination & outcomes:** • [ALL21c] • ISO standardization contribution

Chapter II. State of the art

This chapter presents the background works of the thesis and identifies the thesis main research directions.

.0

The state-of-the-art section will be structured according to the two main content distribution frameworks this thesis deals with, namely blockchain and fingerprinting.

II.1. Blockchain

II.1.A. Fundamentals

This section introduces basic blockchain concepts, acting as a foundation for further discussion within the paper. It doesn't exhaustively cover all aspects but provides key definitions necessary for understanding subsequent content. The definitions in this section are based on [NAK08; CHR16; TSC16; MON19; WAN19; ZHO20].

The word blockchain was coined in 2008, in financial context by Satoshi Nakamoto when introducing Bitcoin [NAK08]. Blockchain builds on the idea of peer-to-peer, zero-trust networks and provides a universal data set that every user can trust, even though they might not know neither trust each other. In other words, it provides a shared and trusted ledger of transactions, where immutable and encrypted copies of information are stored on every active node in the network.

The structure of blockchain technology is meticulously designed to ensure security, transparency, and efficiency. It is composed of following primary elements:

- **Transactions** represent the fundamental actions carried out by participants within the blockchain network. For example, in the context of Bitcoin [NAK08], a transaction involves the transfer of cryptocurrency between two parties. Each transaction is digitally signed by the sender, ensuring the authenticity and non-repudiation of the transaction.
- **Blocks** serve as the containers for these transactions. A block in the blockchain does not just store a single transaction; it aggregates several transactions that are validated and bundled together during a specific time period. For instance, a Bitcoin block contains a list of recent transactions, along with a timestamp and a reference (*via* a hash) to the previous block, as illustrated in Figure 5. This method of linking blocks ensures that each subsequent block reinforces the validation of the previous block and thereby the entire blockchain.
- **Chain of Blocks** refers to the linked sequence of blocks, commonly known as the blockchain. Each block is connected to its predecessors and successors via a cryptographic hash contained in the block header as illustrated in Figure 5. This hash links each block to its previous block, creating a chain that is resistant to data modification. For instance, if an attacker attempts to alter a transaction in a previously confirmed block, the hash of the altered block would change, thus invalidating every subsequent block in the chain due to mismatched hashes. This chain reaction ensures the integrity of the blockchain's history.

The classic blockchain transaction process between Alice and Bob can be articulated in six distinct stages, forming a coherent workflow that ensures both transparency and security:

- Initiation of Transaction: The process begins with Alice initiating a funds transfer to Bob, and is automatically and transparently performed. The identity of each participant is masked using pseudonyms, providing a layer of security and privacy within the network.
- Transaction Encoding and Recording: Following the initiation, the transaction is encoded and subsequently recorded online alongside with other transactions, forming a new block. This aggregation is the preliminary step towards integration into the larger blockchain.
- Verification by Network Participants: all network participants rigorously verify the integrity and validity of the block's transactions. This verification utilizes advanced cryptographic techniques, ensuring that the system is impervious to fraud. Each participant has the capability to review all transactions, both historical and current, enhancing the transparency of the process.
- Block Validation by Miners: Once the transactions within a block are approved, network members known as miners undertake the responsibility of validating the block. This step maintains the blockchain's integrity, as altering any record would require a simultaneous and collaborative effort by the majority of miners, a scenario highly unlikely due to the decentralized nature of the network.
- Block Addition to Blockchain: Post-validation, the block is timestamped and officially added to the blockchain. This blockchain is accessible to all users and serves as a permanent and immutable ledger of all transactions, ensuring a reliable record that withstands any attempt at alteration.
- Completion of Transaction: The final stage sees Bob receiving the transaction initiated by Alice. This transaction is now indelibly recorded on the blockchain, visible and verifiable by all network participants.

This workflow demonstrates the robustness of blockchain technology, where security, transparency, and immutability are of prime importance. The use of asymmetric cryptography (also known as public key cryptography – *PKI*) ensures that each transaction is securely encrypted and only accessible to intended parties, thus maintaining privacy while being transparent and verifiable by all network participants. This secure and transparent aspect reflects the blockchain's potential to revolutionize digital transactions, making it a foundational technology for modern digital interaction.

II.1.B. Consensus

Blockchain is a linked list of Blocks where each block contains a set of transactions. Each blockchain has a specific technique to create these blocks and orchestrate their insertion into the chain (processes known as mining or baking), as illustrated in Table 2. The most popular technique is the proof of work (PoW) used in blockchains like Bitcoin and Ethereum. Several other types of consensus algorithms are proposed in the literature such

as the Proof of Stake (PoS), or the Proof of Authority (PoA); they are considered by themselves or in conjunction with some optimization mechanisms like the case of Tezos, Cardano, Solana, VeChain, or GoChain, to mention but a few.

Table 2: Comparison on between PoW, PoS, and PoA consensus algorithms

Feature	Proof of Work (PoW)	Proof of Stake (PoS)	Proof of Authority (PoA)
Core Mechanism	All users can be miners and solve complex cryptographic puzzles to validate transactions.	Validators are chosen based on the amount of tokens they stake.	Pre-selected validators validate transactions.
Resource Dependence	High computational power and electricity consumption.	Relies on the possession of a large stake in the network.	Relies on the identity and reputation of validators.
Security	Secured by computational work, which makes attacks costly.	Security is based on economic stakes and penalties.	Security is based on trustworthiness of the authorized validators.
Speed and Efficiency	Slower transaction validation and higher energy use.	Faster validation and reduced energy consumption compared to PoW.	High efficiency and speed, ideal for private networks.
Decentralization	100% decentralized, anyone with high computational resources being able to participate.	Less decentralized, based on the amount of tokens the user possess.	Low decentralization, few validators being concerned.

II.1.C. Smart Contracts

Smart Contracts are self-executing contracts with the terms of the agreement directly written into lines of code that can adapt to various type of industries. The concept was advanced by Nick Szabo in 1994 [SZA94], long before blockchain, as a way to automate legal contract execution in digital environments. With the advent of blockchain platforms like Ethereum, Smart Contracts have found a practical application, as they can operate in a trustless environment and be executed automatically when a set of predefined conditions are met. This automation not only reduces the need for intermediaries and system administrators but also decreases the likelihood of fraud and disputes, ensuring that all parties adhere to the contract terms without bias or error.

A Smart Contract can be illustrated as a software installed on the Blockchain that automatically executes a pre-programmed contractual commitment. It is not a legal document in itself, but it automates the execution of a contractual commitment.

Once they are deployed, Smart Contracts are immutable and public, meaning that it cannot be altered, and its execution history is accessible for verification by all network participants. This feature has been pivotal in sectors such as supply chain management, where it ensures compliance and allows product tracking from production to delivery [BAD18].

The structure of the Smart Contract lifecycle is presented in Figure 9. Initially, an *Agreement phase* occurs where two or more parties identify a mutual opportunity and set the terms and conditions of the contract. Following this agreement, the Smart Contract *Coding phase* begins. In this stage, the agreed-upon terms are translated into a blockchain-compatible programming language such as Solidity for Ethereum-based contracts or

Liquidity for Tezos-based contracts. The coding process is the most critical as it directly influences the contract's functionality and security, requiring developers to implement robust error-checking and validation mechanisms to prevent exploits and unintended operations.

After the finalization of the code development, the code should be compiled into Michelson for Tezos or Bytecode for Ethereum. The Solidity code results in two files, the Bytecode which is the executable part and Application Binary Interface (ABI) which defines how to interact with contracts, both from outside the network and from other contract [ABI24].

Next, the *Encryption and Deployment phase* takes place. The contract code is encrypted, and then deployed onto the blockchain. Upon deployment, the contract resides in an dormant (idle) state within the blockchain environment, waiting for its triggering conditions to be met. *The Execution phase* is initiated when the predefined conditions are fulfilled, such as the achievement of a specific date, the receipt of a payment… When activated, the Smart Contract executes the encoded instructions without the need for further human intervention. Finally, the *Network Update phase* solidifies the contract's actions within the blockchain. The results of the contract's execution are immutably recorded on the blockchain and synchronized across all nodes within the network. This update ensures that every participant maintains an updated and consistent view of the ledger, reflecting the new state post-execution. The contract returns to its dormant statement waiting for its next execution.

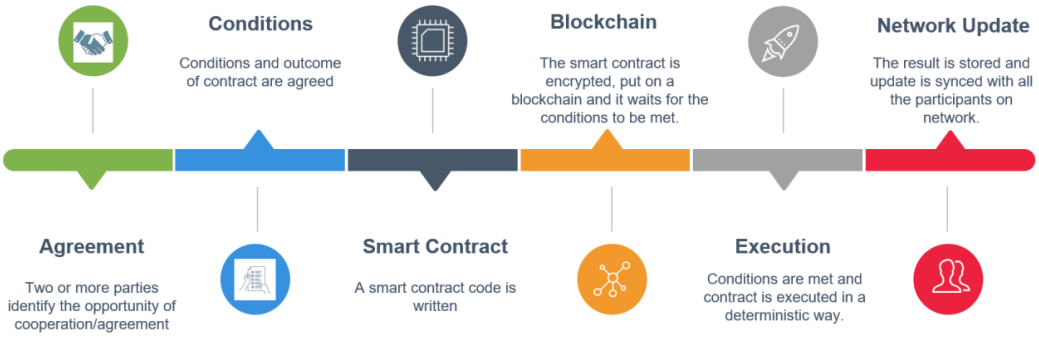


Figure 9: Smart Contract life cycle (source: [HOW24])

II.1.D. Tokens

Tokens are abstract digital entities, created on blockchain and offering the possibility to represent any type of real-world assets (*e.g.* from art to real estates or IPR [SCH21]). Unlike primary cryptocurrencies like Ether or Tez, which are native to their specific blockchains, tokens are developed using standards set by the development and industrial communities.

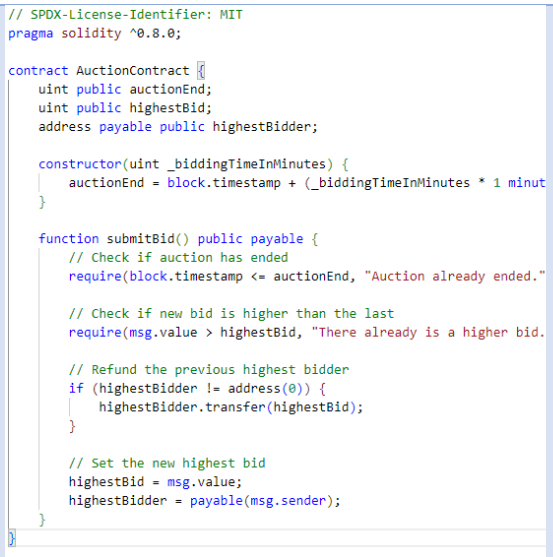
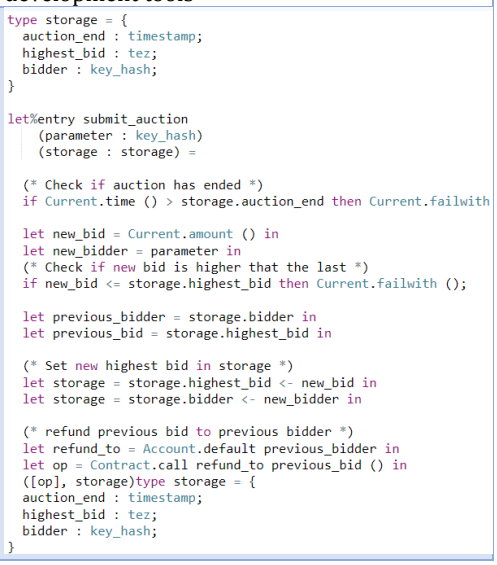
According to [ETH24c], two types of tokens are used in Smart Contracts, namely *non-fungible* and *fungible* tokens. Non-fungible tokens are unique—one token represents one piece of an asset or one user, whereas fungible tokens are identical and uniform.

Ethereum communities are the first to adapt a standardization aspect to tokens by the implementation of token standards such as ERC-20 for fungible tokens and ERC-721 for non-fungible tokens (NFTs) [ETH24a], which have pioneered the development of a vast array of decentralized applications (DApps). These tokens can serve multifarious purposes ranging from digital currencies to representation of physical assets. Similar to Ethereum, Tezos supports fungible and non-fungible tokens through its FA1.2 and FA2 standards, which ensure high interoperability and flexibility for developers [OPE24a].

II.1.E. Blockchain dependency: Tezos vs. Ethereum

While the presentation before tries to bring forth the common principles, note that blockchain solutions are developed to optimize specific issues and very little (if any) provision is made for interoperability. Hence, Table 3 is meant to show the differences and complementarities between the two platforms considered in the present thesis, namely Ethereum and Tezos.

Table 3: Comparison of basic concepts in Ethereum and Tezos

Technical Aspect	Ethereum	Tezos
Design philosophy	Decentralized applications and DeFi platform	Self-amending crypto-ledger designed to avoid hard forks
Consensus mechanism	Proof of Work (PoW), known for high energy consumption	Liquid Proof of Stake (LPoS) reduces energy use and allows decentralized staking
Transaction speed	15-30 transactions per second	Approximately 40 transactions per second, scalable with network upgrades
Scalability solutions	Limited; primarily network upgrades and EIPs without fundamental changes to scalability	On-chain governance facilitates seamless and continuous upgrades
Network upgrade mechanism	Ethereum Improvement Proposals (EIPs) for upgrades	Protocol upgrades are voted on and implemented by token holders
Security features	Smart Contract audits, EVM for executing code	Formal verification of contracts and on-chain governance for security patches
Smart Contract language	Solidity, Vyper [ETH24b]	Liquidity, LIGO, SmartPy, Archetype [LAN24]
Developer ecosystem	Extensive, with a large number of developers and tools	Growing, supported by an increasing range of development tools
Code example		

II.2. Fingerprinting

The fingerprinting state of the art was studied in [ALL22], from which the present thesis borrows its main elements.

II.2.A. Definition

Video fingerprints is best understood by drawing an analogy to the human fingerprints as illustrated in Figure 10. The patterns of dermal ridges on the human fingertips are natural identifiers for humans. Although they convey very little information compared to the entire human, human fingerprints can uniquely identify a person even if the person changes haircuts, clothes, or wears a wig or a disguise, [IDR97; GAR16; JIA16; ALL22].

Analogously, video fingerprints are video identifiers. The fingerprints must be able to uniquely identify visual contents even if it goes under multiple set of transformations. The transformations a video can undergo will be further referred to as *modifications*, *distortions*, or *attacks*. The resulting video which is transformed, modified, distorted or attacked will be denoted as a *copy*, a *replica* or a *near-duplicated* video.

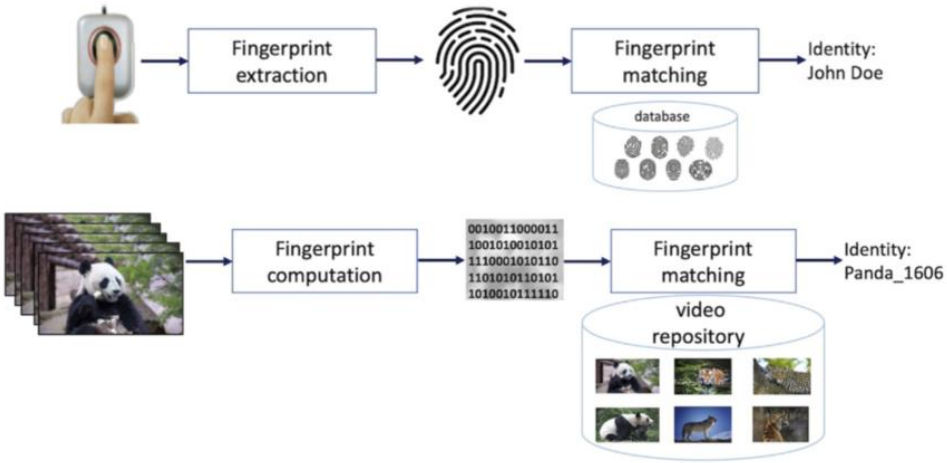


Figure 10: Human fingerprinting versus video fingerprinting

A video fingerprint is a compact, distinctive digital representation of a video that allows its unique identification across a variety of contexts, even when the video undergoes transformations such as resizing, compression, or cropping. Unlike metadata-based methods or watermarking techniques, which rely on external information insertion, video fingerprinting extracts features directly from the content itself. These features may include key visual patterns, motion sequences, or color distributions. Video fingerprints must remain stable across the application specific distortions to ensure robust recognition of both original and modified versions of the sequence.

By focusing on characteristics already present in the clip, video fingerprints provide a reliable mechanism for video identification, particularly useful for content tracking, copyright protection, and duplicate detection across diverse platforms. They ensure that the video's identity remains verifiable despite changes, making it resilient against intentional or unintentional alterations.

II.2.B. Applicative scope

The applicative scope of video fingerprint can be identified through synergies and complementarities with video indexing [IDR97] and video watermarking [DIG08].

Video indexing could be historically identify as the first framework for content-based searching and retrieval, tracing its roots back to efforts in the late 1990s [IDR97; BUR10]. Assuming a video repository, the goal of video indexing is to identify all the video sequences that are visually related to a query. For instance, assuming the query is a video showing some Panda bears and the repository consist of some wild animal sequences, a video indexing solution searches for all sequences in the repository that contain Panda bears, as well as sequences containing the same type of background, as illustrated in Figure 11. To this end, salient features (referred to as descriptors) are extracted from the query and compared to the descriptors of all the sequences in that repository. The visual descriptors capture the visual essence of each sequence, such as color distributions, motion patterns, edges, or object shapes. Once extracted, the descriptors are compared using a pre-defined similarity measure that quantifies the visual proximity between two sequences. A pre-calibrated threshold is often set to determine when two video descriptors are considered a match, allowing the system to efficiently search large video datasets.

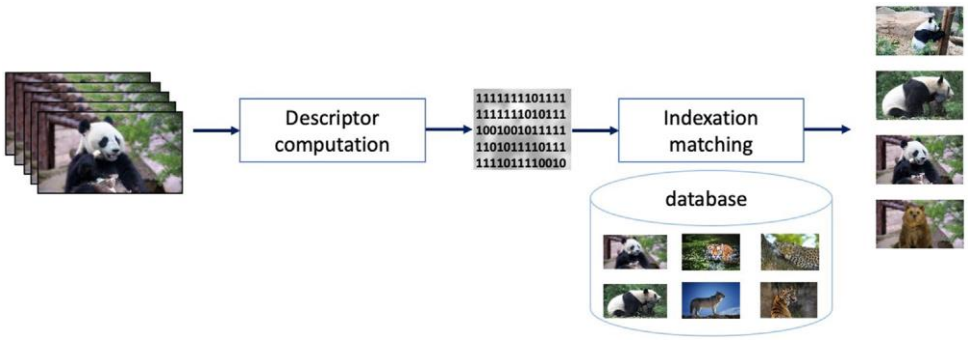


Figure 11: Video indexing principle: a binary descriptor is extracted from a query video to retrieve any other related visual content in the dataset

Digital watermarking [DIG08] deals with the identification of any modified version of video content, Figure 12. In this case, assuming again a video sequence representing some Panda bears is displayed on a screen and that the screen content is recorded by an external camera, the original content should be identifiable from the camera recorded version. To this end, some extra information (referred to as mark or watermark) is imperceptibly inserted (or, as a synonym, embedded) into the video content prior to its release

(distribution, storage, display, ...). By detecting the watermark in a potentially modified version of the watermarked video content, the original content shall be unambiguously identified. Of course, the watermark shall not be recovered from any unmarked content (be it visually related to the original content or not). This technique ensures that even altered versions of a video can easily be traced back to their original source. One of the primary use cases includes copyright protection, where media owners and producers can identify unauthorized distribution by tracking the watermark in pirated copies. Watermarking techniques can address the challenges related to broadcast monitoring, rights management, content management but only under the condition that the content is already watermarked, (*i.e.* the additional information is inserted in the multimedia content before its distribution).

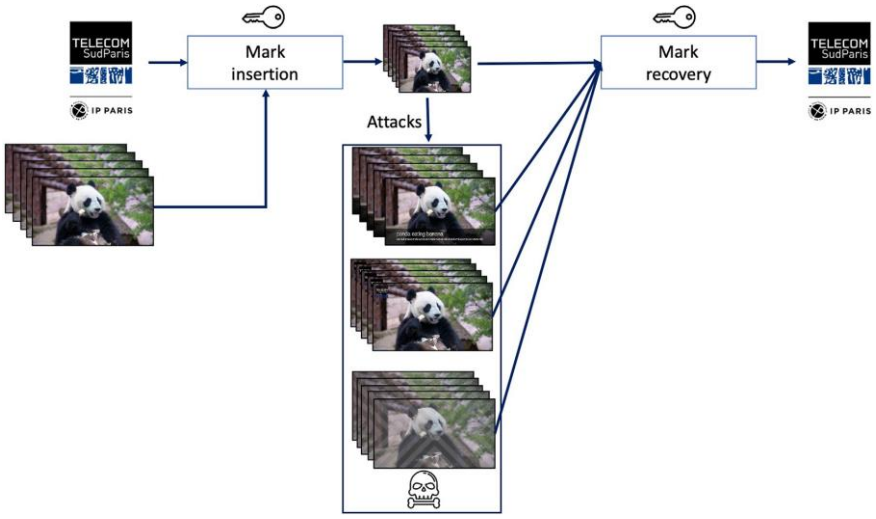


Figure 12: Video watermarking principle: a binary watermark is imperceptibly inserted (embedded) in the video sequence; this way, the watermarked sequence can be subsequently identified even when its content is modified (maliciously or not)

Video fingerprinting also deals with identifying slightly modified (replicated, or near duplicated) content, yet its approach is different with respect to both indexing and watermarking, as illustrated in Figure 7 and Figure 13. Coming back to the previous two examples, video fingerprinting shall also track a near-duplicated video sequence (*e.g.*, a screen recorded Panda sequence) back to its original (*e.g.*, the Panda original sequence) that is stored in a video repository. Yet, unlike indexing, any other sequence, even visually related to it (*e.g.*, the same Panda bear at a different time of the day and/or in different postures) shall not be detected as identical. To this end, some salient information (referred to as *fingerprint* or *perceptual hash*) is extracted from the query video sequence (note that this information is not previously inserted in the content, as in case of watermarking sequences). By comparing (according to a similarity measure and a preestablished threshold) the query fingerprint to the reference sequence fingerprints, a decision on the visual identity between the video sequences shall be made.

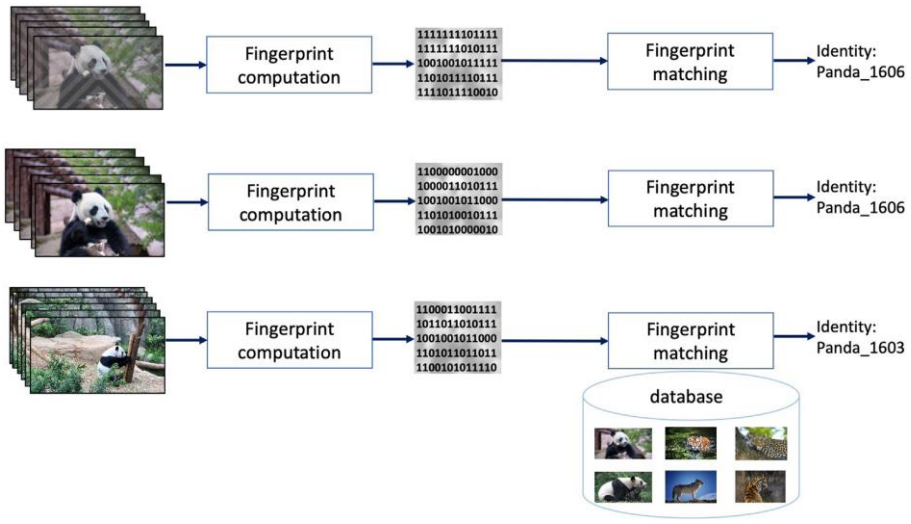


Figure 13: Video fingerprinting principle: a binary descriptor extracted from a query video (fingerprint) can unambiguously identify all the near-duplicated versions of that content.

By comparing among them these three methodological frameworks, it can be noted that:

- Indexing and fingerprinting share the concept of tracking content *via* information directly extracted from that content (that is, both indexing and fingerprinting are passive tracking technique). However, while fingerprinting tracks the content instance, indexing rather focuses on identifying related or similar semantic content families. From an application standpoint, fingerprinting emphasizes unicity by tracking exact copies, whereas indexing accommodates wide range of semantic variations.
- Watermarking and fingerprinting share the possibility of tracking both an original content and its replicas modified under a given level of accepted distortion. However watermarking requires the insertion of additional information (that is, watermarking is an active tracking technique), whereas fingerprinting solely exploits information extracted from the very content to be tracked. This makes watermarking more adaptable for active tracking, while fingerprinting operates without altering the content offering passive tracking.

A fingerprinting methodology often requires the following three main properties:

- First, the *unicity* (or *uniqueness*) property assumes that different contents (*i.e.*, content that is neither the query nor one of its near-duplicated versions) result in different fingerprints (in the sense of the similarity measure and of its related threshold).
- Secondly, the *robustness* property relates to the possibility of identifying as similar sequences that are near-duplicated. The transformations a video can undergo will be further referred to as modifications, distortions, or attacks. The video that is obtained through transformations, modifications, distortions, or attacks will be denoted as a copy, a replica video, a near duplicated video or an attacked video. While these terms are conceptually similar, fine distinction among them can be made for some specific applicative fields.

- Finally, the *dataset search efficiency* property ensure that the computation of the fingerprints and the matching procedure is optimized offering low, application dependent computation time. The dataset search efficiency is assessed by the average computation time needed to identify a query in the context of a considered video fingerprinting use case (that is, execution time on a given processing environment and on a given repository).

Nowadays, any fingerprinting method can usually be separated to the two main steps in a generic fingerprinting computing pipeline: fingerprinting extraction (that is, spatio-temporal salient information extraction) and fingerprinting matching (that is, comparing salient information extracted from two different video sequences). These two basic steps are, in their turn, composed of several sub-steps [DOU08; LEE08; XIN09]. On the one hand, the fingerprint extraction generally includes video pre-processing (*e.g.,* frame resizing, letterboxing removal, frame dropping or key-frame detection), global feature extraction, local feature extraction, local/global feature description, temporal information retrieval, and the means for accelerating the search in the dataset (inversed file, *etc*.). On the other hand, the fingerprint detection process generally includes some time-alignment operations (*e.g.,* jitter cancelation, time origin synchronization), followed by information matching.

II.2.C. Evaluation framework

State of the art methods consider fingerprinting as a binary problem, whose performances are empirically evaluated according to the four types of situations that can be encountered during the fingerprinting matching:

- False positive (*fp*): also referred to as *false alarms*, the system erroneously retrieved a reference video as a copy of the input.
- False negative (*fn*): also referred to as *missed detections*, the system erroneously did not retrieve a reference video which is a copy of the input.
- True positive (*tp*): the system correctly retrieved a reference video that is a copy of the input.
- True negative (*tn*): the system correctly did not retrieve a reference video which was not a copy of the input.

The measures above are formalized into performance indicators, as follows:

- To evaluate the *database search efficiency* property, the average processing time required by the fingerprinting system to identify the query video within the database and to output the result is considered.
- To evaluate the *uniqueness* property, two evaluation criteria are considered in the literature: the *probability of false alarm (Pfa)* and the *precision rate (Prec)* defined as:

$$P_{fa} = \frac{fp}{fp + fn + tp + tn}$$

$$Prec = \frac{tp}{tp + fp}$$

- To evaluate the *robustness* property, two evaluation criteria are considered in the literature: the probability of missed detection (Pmd) and the recall rate (Rec) defined as:

$$P_{md} = \frac{fn}{tp + tn + fn + fp}$$

$$Rec = \frac{tp}{tp + fn}$$

- To evaluate the trade-off between *uniqueness* and *robustness*, the *Accuracy* and the *F1 score* are computed.

$$Accuracy = \frac{tp + tn}{tp + tn + fn + fp}$$

The *F1* score is computed only in the case of correctly identified copies, as the harmonic mean of *Prec* (precision) and *Rec* (recall) rates and defined as:

$$F1 = \frac{2 \times Prec \times Rec}{Prec + Rec}$$

An efficient fingerprinting method (featuring both unicity and robustness) should jointly ensure low values for P_{fa} and P_{md} while having *Prec* and *Rec* values close to 1. The actual thresholds for these entities depend on the specific use case.

Although *Prec* and *Rec* are two measures commonly used in the evaluation of any information retrieval system, they are not statistical measures, as they do not consider the *tn* (true negative) results. Hence, to comprehensively present the properties of a system, P_{fa} and P_{md} should also be considered. In practice, several other derived and/or complementary performance indicators can be considered, such as the *F1 score*, the *ROC* (Receiver Operating Characteristic), the *AUC* (Area Under the Curve), or the *mAP* (mean Average Precision).

II.2.D. Information theory based fingerprinting methods

1. Main directions

As a common ground, these methods stem from image processing, machine learning, and information theory concepts and leverage the fingerprinting extraction on three incremental levels [GAR16].

First, in an attempt to get to frame aspect distortion invariance, the fingerprinting is extracted from derived representations such as 2D-DWT (2D Discrete Wavelet Transform) coefficients [GAR16], 3D-DCT (3D Discrete Cosine Transform) coefficients [COS06], pixel differences between consecutive frames, temporal ordinal measure of average intensity blocks in successive frames [HAM01], visual attention regions [XIN09], quantized block

motion vectors, ordinal ranking of average grey level of frame blocks, quantized compact Fourier–Mellin transform coefficients, ordinal histograms of frames [CHA05], color layout descriptor, ...

Secondly, frame content distortion invariance can be achieved by the complementary between global features incorporating geometric information (*e.g.*, centroid of gradient orientations of keyframes [LEE08] or invariant moments of frames edge representation) and local features based on interest points (corner features, Hessian-Affine, Harris points, Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF)) generally described under the Bag of Visual Words (BoVW) framework [DOU08; JIA11].

Thirdly, video format distortion invariance is generally handled by using a large variety of additional synchronization mechanisms, pair designed with the feature selection, from synchronization block, based on wavelet coefficients to K-Nearest Neighbors matching [LAW07] of interest points or Viterbi-like algorithms [WEI11].

These main directions as well as their mutual combinations serve as basis for a large variety of studies, as presented in Figure 14 that is structured in three layers, shaped as hemicycles:

- the outer blue layer relates to local feature description, exemplified through MPEG-CDVS (Compact Descriptors for Visual Search), ORB (Oriented Fast and Rotated BRIEF), SURF, Transformed domains, SIFT, CS-LBP (Center-symmetric Local Binary Patterns), and HOG (Histogram of Oriented Gradient).
- The middle grey layer relates to global features, exemplified through luminance component, color histograms, and BoVW.
- The inner blue layer relates to the temporal features, exemplified through luminance spectrogram, motion vectors, histogram correlation, optical flow, and TIRI (Temporal Informative Representative Image).

The order of the classes in each hemicycle is chosen to allow for a better visual representation of the synergies among them. The studies represented in grey-shadowed rectangles correspond to conventional methods while the studies represented in white rectangles correspond to NN-based method that also include conventional modules.

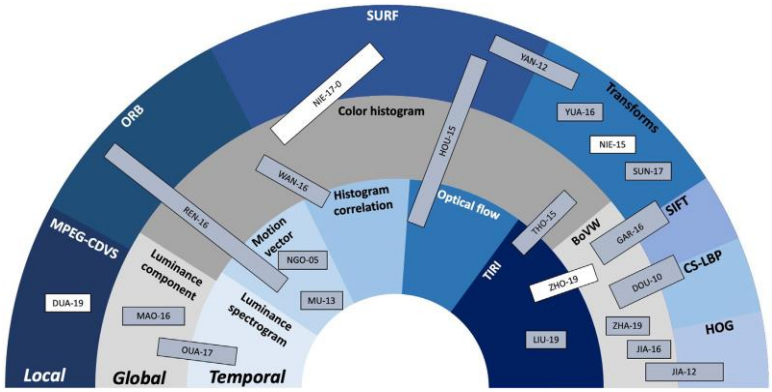


Figure 14: Conventional fingerprinting method synopsis: the hemicycles (areas) related to the local, global, and temporal features are located at the outer, middle, and inner parts of the figure, respectively. Inside each hemicycle, examples of state-of-the-art solutions are presented. Conventional methods are presented in gray-shadowed rectangles while NN-based methods that also include conventional modules are represented in white rectangles.

2. Methods overview

Structural modeling and temporal analysis form the backbone of several advanced fingerprinting techniques. For instance, [TAN09] employs a network flow model to integrate visual similarity and temporal consistency, effectively transforming frame matching into a search for maximal paths. Similarly, [CHO05] and [WAN16] emphasize the temporal aspects of video content. [CHO05] models videos as temporal graphs to identify salient features through motion vector analysis, while [WAN16] structures video sequences around key frames to utilize color correlation histograms, focusing on the sequential nature of frames for enhanced temporal context. Additionally, [SUN17] adopts a contourlet Hidden Markov Tree model, capturing multidirectional and multiscale information, linking coefficient states across the video structure for a robust analysis.

Feature extraction is another pivotal area where techniques such as those in [DOU10], [YAN12], and [JIA12] focus on capturing detailed local and global visual features. These studies use advanced methods like SURF points and histograms of oriented gradients combined with relative mean intensity to extract distinctive features from video frames. The approach not only aids in creating robust fingerprints but also adapts well to various video modifications, enhancing the robustness of the fingerprinting process. Furthermore, [HOU15] and [THO15] extend the feature extraction by integrating multiple features including color histograms and optical flow, ensuring that both motion and color data contribute to the fingerprint.

Dimensionality reduction and efficient matching strategies are crucial for handling large datasets and ensuring quick retrieval times. [LIU19] introduces the rHash method, which simplifies the fingerprint matching process by reducing frame rates and resolution, followed by binary operations to streamline the fingerprint structure. Techniques like the Double Optimal Projection in [NIE15] and the clustering methods in [MAO16] further emphasize reducing fingerprint dimensionality, which helps in managing the vast amounts of data involved in video processing. Moreover, [YUA16] leverages the Shearlet transform to handle both low and high frequency aspects, obtaining optimized fingerprints for quick matching processes.

3. Discussion

The previous section brings to light that the fingerprinting conventional methods form a fragmented landscape. While the general methodological framework is unitary, each study ambitions to take a different applicative challenge, from searching of strongly modified sequences in reduced-size video datasets to reducing the complexity and the execution time of the fingerprint matching.

These research efforts can be timely illustrated in Figure 15. This figure covers the 2009—2019 time span and presents the key conceptual ideas (the dark-blue, left block) as well as the methodological enablers in fingerprinting extraction (the blue, right-upper block) and matching (the light-blue, right-lower block). For a specific year, the information presented in Figure 14 may correspond to several references.

Figure 14 and Figure 15 shows that the state-of-the-art is versatile enough to pragmatically offer solutions to specific applicative fields, without being able to provide the ultimate fingerprinting method.

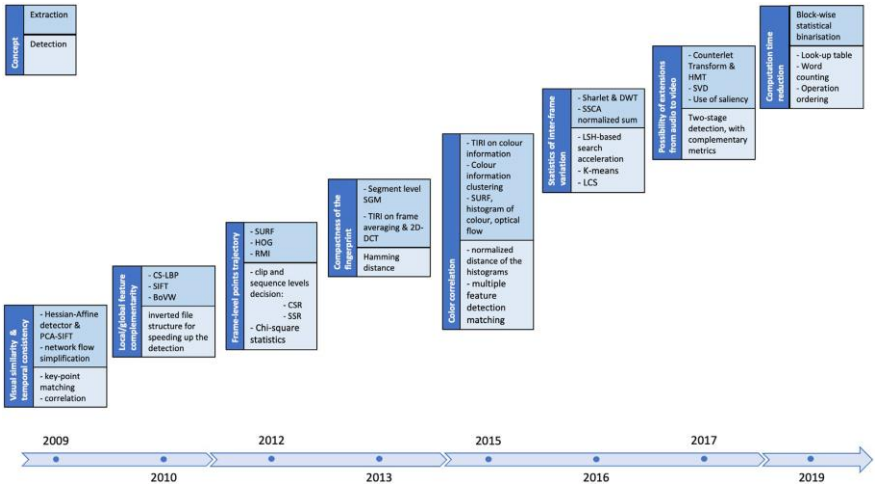


Figure 15: Incremental evolution of the conventional methods.

II.2.E. Neural Network based methods

1. Main directions

The class of NN-based video fingerprinting methods is incremental with respect to the conventional fingerprinting methods presented in Section II.2.D. It inherits its basic conceptual workflow: pre-processing video sequence, extracting spatial and temporal information, eventually aggregating them into various derived representations (be they binary or not).

However, NN-based video fingerprinting methods rely (at least partially) on various types of NN, from AlexNet [KRI12] and ResNet [HE16] to CapsNet [SAB17] and LSTM [HOC97], sometimes requiring specifically designed architectures [ZHI18]. Yet, such an approach does not exclude the usage of partial conventional solutions in conjunction with NN, *e.g.*, BoVW can be considered as an aggregation tool of visual features extracted by Convolutional Neural Networks (CNN) [ZHA19]. Moreover, the matching algorithm generally comes across with the NN considered in the extraction phase.

These main directions will be illustrated by a selection of 20 studies, published since 2016. The relationship among and between them is depicted in Figure 16, that is also structured in three hemicycles (as Figure 14), yet their meanings are slightly different:

- The outer blue layer corresponds to the spatial features, exemplified through: CRBM(Conditional Restricted Boltzmann Machine), ResNet, NIP (Nested Invariance Pooling), VGGNet, AlexNet, GoogleNet, new structures designed to the fingerprinting purpose, RetinaNet, and Tracked HetConv-MK (heterogeneous convolutional multi-kernel).

- The middle gray layer corresponds to temporal features, exemplified through: weight correlation, LSTM (Long- Short Term Memory), SiameseLSTM (Siamese LSTM), Deep Metric Learning, and BiLSTM (bidirectional LSTM).
- The inner blue layer corresponds to spatial-temporal features, exemplified through: 3D-ResNet50, and CapsNets structures.

The order of the classes in each hemicycle is again chosen to allow for a better visual representation of the synergies among them.

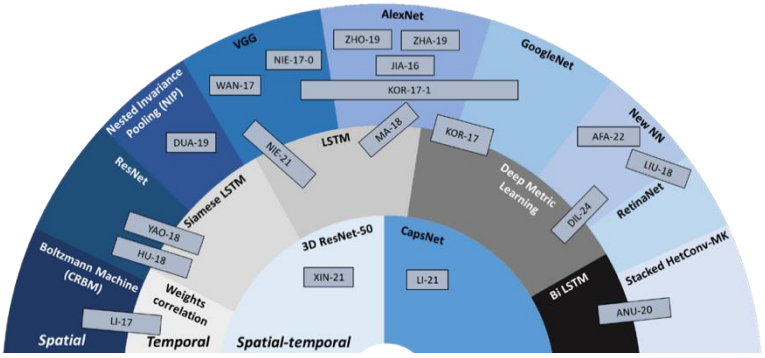


Figure 16: NN-based fingerprinting method synopsis: the hemicycles (areas) related to the spatial, temporal and spatial-temporal features are located at the outer, middle, and inner parts of the figure, respectively. Inside each hemicycle, examples of state-of-the-art solutions are presented

2. Methods overview

The state-of-the-art shows that, as a rule, CNNs are innovatively integrated with traditional computer vision techniques to optimize feature extraction processes.

For instance, [JIA16] and [ZHO19] apply CNNs in conjunction with systems like the Bag of Visual Words (BoVW) to improve the organization and analysis of extracted features. These studies utilize architectures such as AlexNet not only to capture spatial and temporal features from video streams but also to process these features through layers that enhance their utility for tasks like video copy detection and content retrieval. Similarly, [ZHA19] exploits CNNs for extracting robust feature vectors, which are then utilized in a BoVW framework for efficient retrieval of top-k video clips, showcasing the adaptability of CNNs to various aspects of video processing.

Further exploring the depth of CNN activations, studies like [KOR17; LIU18; DIL24] focus on utilizing intermediate outputs from CNN layers. [KOR17] employs these activations to create frame-level histograms which are aggregated into video-level histograms, enhancing the detail and richness of the information captured for video retrieval. On the other hand, [LIU18] takes a spatial fingerprinting approach, leveraging CNN-detected visual objects which are then binarized and processed through intricate thresholding operations to create distinct fingerprints for each video. Moreover, the features [DIL24] considered as fingerprints are derived from both convolutional and fully connected layers of a pre-trained VGG-16 model. The method propose a triple loss framework coupled with a homemade DNN for fingerprint matching.

The integration of CNNs with recurrent neural network (RNN) architectures also marks a significant trend, as illustrated by [HU18] and [YAO18]. These studies combine the spatial feature extraction capabilities of CNNs with the temporal processing power of RNNs, particularly using Long Short-Term Memory (LSTM) units. This combination allows for a comprehensive analysis that captures both the immediate visual details and the dynamic content changes over time, thereby enhancing the accuracy and reliability of the video detection process.

Emerging neural network technologies like Capsule Neural Networks (CapsNet) have also been explored, as in [XIN21], which uses CapsNet to capture hierarchical feature relationships more effectively. This approach is especially beneficial in scenarios where video data may exhibit significant variations in appearance and motion, providing a robust framework that maintains high accuracy despite such challenges.

Deep metric learning (DML) has found its place in enhancing the discriminative capabilities of video fingerprinting models. For example, [KOR17b] introduces a triplet-based network that refines feature comparison and matching, focusing on optimizing the loss functions to better differentiate between similar and dissimilar videos, thereby improving the overall precision of the system. A similar triple loss function coupled with a custom made DNN was presented by [DIL24]

Hybrid models that combine convolutional approaches with hashing techniques to create compact yet effective fingerprints of video content also show substantial advancements. [ANU20; LI21; XIN21; ZHA21] illustrate this approach by integrating multi-kernel convolutional filters with bidirectional LSTM networks or quadruplet fully connected CNN structures to produce binary codes that effectively summarize video content for quick retrieval and matching.

Methods using compressed video streams are also studied. [PRO24] presents a near-duplicate video retrieval method using the compressed representation by partially decode the video and extraction the residual elements as well as the motion vectors as fingerprints when a visual transformer model is used for matching. On the other hand, [AFA22] develops a method for identifying YouTube videos in network traffic by utilizing a fingerprinting technique that addresses inconsistencies in Variable Bit-Rate (VBR) streaming, followed by a custom CNN model for the matching process.

3. Discussion

A global retrospective view on the investigated NN-based methods is presented in Figure 17 that is paired designed with Figure 15. It originates in 2016 and presents, for each analyzed year, the key conceptual ideas (the dark-blue, left block) as well as the methodological enablers in fingerprinting (the blue, right block). Note that in this case the fingerprint extraction and matching are merged (as they are tightly coupled).

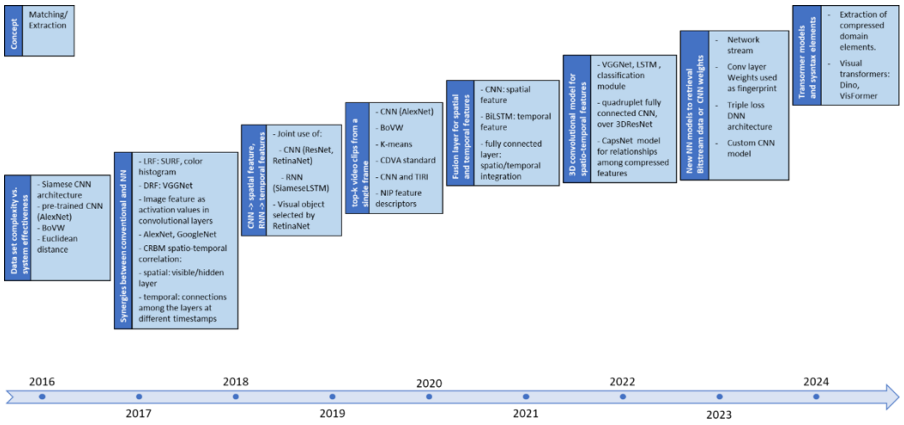


Figure 17: Evolution of the NN methods

The rapid evolution of neural network-based fingerprinting, highlighted as an emerging research field since 2016, significantly expands upon the methodological framework established by conventional fingerprinting approaches. Fingerprint extraction gradually shifted from considering NN solution at an individual level (*e.g.*, spatial or temporal features) to holistic, 3D neural networks capable of capturing integrated spatio-temporal features effectively. Intermediate solutions that blend neural network techniques with conventional image processing tools, such as SURF, TIRI, or BoVW, also demonstrate significant progress, bridging the gap between traditional methodologies and modern computational capabilities. These hybrid models, which integrate deep learning with traditional hashing or encoding strategies, address the complex challenges of managing large-scale video datasets, enhancing detection precision and recall, and improving the efficiency of video retrieval systems. By continuously advancing the integration and optimization of diverse analytical techniques, these innovative approaches bring to light their capabilities to solve the scalability, precision, and operational efficiency of video fingerprinting challenges.

II.2.F. Towards compressed domain fingerprinting

The conventional video encoding/decoding scheme involves several stages to efficiently compress/decompress a video stream, as presented in Figure 18. Initially, the video undergoes pre-processing to optimize the data for encoding. Following this, the video is partitioned into segments called *slices* that can be independently processed, enhancing parallel processing capabilities and overall efficiency. The prediction stage utilizes spatial (within the same frame) and temporal (across different frames) elements to predict video content, significantly reducing redundancy before transformation. The residual signals are then transformed by the *DCT* or *DST* functions in order to compact the residual energy into a low-frequency components. These coefficients are then quantized, reducing precision and removing less visible details to the human eye. The entropy coding stage further compresses the data by applying the *Context-adaptive binary arithmetic coding* (CABAC) lossless data compression method. Finally, the bitstream generation packages the compressed data into a formatted bitstream more adapted to storage or transmission.

The decoding process is paired designed: except for the *Quantization*, all other operations are one-to-one functions.

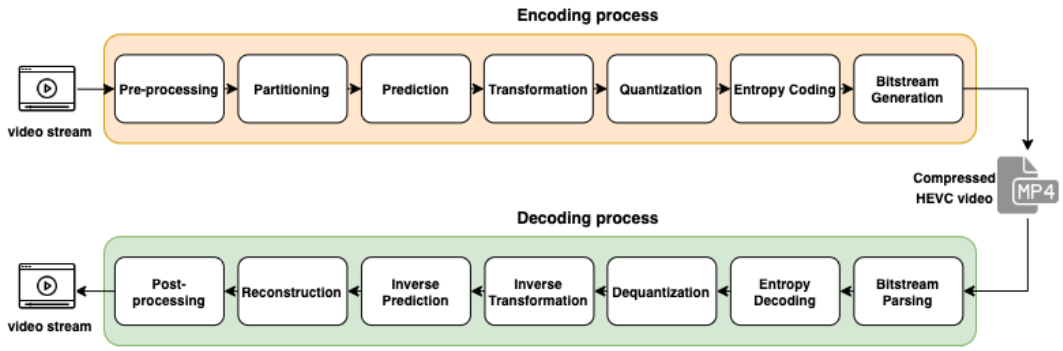


Figure 18: Main HEVC steps in video encoding/decoding

As video content is preponderantly recorded, stored, and transmitted in compressed formats, fingerprints extracted directly from the compressed stream will beneficially eliminate the need for decoding operations. While early studies [NGO05], [LI13] already considered MPEG motion vectors as a partial information in fingerprinting applications, [REN16] can be considered as an incremental step: the fingerprinting computation combines features from the decompressed domain and features from MPEG-2 stream level. The fingerprint is a combination of the color histograms, ORB descriptors and motion vector normalized histogram. The video fingerprint is the set of key frame fingerprints. The matching procedure is individually performed at the level of the three components and the overall decision is achieved through fusing decisions made on multiple features by a weighted additive voting model. [PRO24] tackled the compressed domain challenge by using the motion vectors and the residuals extracted from each GOP composing a HEVC video. It is stated that this method allows reducing the computation duration twice compared to uncompressed domain methods. For the matching process, the authors studied different transformers and concluded that DINO_vits16 and VisFormer_small models are both capable to efficiently retrieve near duplicate videos.

Another way to solve the compressed domain video tracking is to focus on the network stream generated once the video is being streamed. [SCH17] presents a method for fingerprinting video streams, using distinct packet burst patterns related to the content, where the very burst of information delivered to the end user serve as the fingerprint. A specific CNN model is then used for the matching step. Inspired by these results, the study in [AFA22; LI18] further investigates the capabilities of the data collected by a *Middleman* present in the network, by extracting the information from the Wi-Fi traffic.

II.3. Summary

This state-of-the-art study not only brings to light the current day limitations of the two basic technologies of relevance for the thesis, namely blockchain and fingerprinting, but also put them in the perspective of their mutual integration, as follows:

On the one hand, when putting the blockchains in the perspective of supporting fingerprinting-like operations, multifold inner limitations are encountered, and answers to open questions are expected:

- How can the bit sensitive hash functions (*e.g.* SHA256) cope with the plurality of digital representations for the same visual content?
- How the formal limitations in the amount of data that can be exchanged between blockchain components are reflected in the versatility of applicative workflows that go beyond simple transactions?
- While the energy consumption generally associated to blockchain functioning and the need of high computational powers for block generation are curiously accepted in digital banking, can it still be accepted in other applicative fields?

On the other hand, when putting the fingerprinting solutions in the perspective of effective compressed stream processing, the state of the art becomes scarce, with some ad-hoc solutions, rather pragmatic then addressing the problem at the conceptual level. Consequently, open questions expect accurate answers in this field:

- While very little (if any) visual redundancy is expected to still be present in compressed stream syntax elements, how can near-duplicated visual content be identified as similar?
- While DL methods benefit from redundancy to learn patterns, how can the fingerprints be clustered?
- While fingerprints are expected to reach a trade-off between uniqueness and robustness, are classifiers an effective trend in compressed domain fingerprinting?
- While sophisticated DL techniques seem promising for compressed domain fingerprinting, would them still applicative effective?

The thesis structure is also oriented according to the findings in the state-of-the-art analysis. First, in Chapter III On-chain / off-chain processing, the blockchain computing environment will be revisited in order to allow effective load balancing between on-chain and off-chain components, thus allowing complex workflow to be backboned by blockchains. Secondly, in Chapter IV Compressed domain video fingerprinting, the possibility of achieving compressed domain video fingerprinting will be demonstrated based on ML and DL approaches. Finally, the Chapter V Blockchain-fingerprinting applicative synergies shows the effective coupling of the two content tracking paradigms the present thesis deals with.

Chapter III. On-chain / off-chain processing

This chapter will present an on-chain / off-chain load balancing methodological framework, making it possible for synergies between the blockchain main properties and general purposes computing systems to be established. Direct applicative instantiations will be also hinted to.

When looking forward towards ensuring complex computation operations on a blockchain, a twofold deadlock is met. On the one hand, enriching the current day blockchain workflows (*on-chain*) with larger programming functionalities and computing resources is forbidden by the blockchain costs (gas). On the other hand, benefiting from conventional, general purposes computing environment (*off-chain*) is intrinsically limited in security, immutability and trust properties. Faced to this problem, the present thesis identifies the optimal trade-off between blockchains and general purposes computing environment, by establishing the architectural framework for ensuring the zero-trust *on-chain / off-chain* load balancing and by demonstrating its effectiveness on a use-case related to video recording.

To this end, the present section designs COLLATE, *on-Chain Off-chain Load baLancing ArchiTecturE*, in Section III.1, then presents its open source code in Section III.2, before demonstrating its effectiveness on an ISO/IEC 23093 Internet of Media Things (IoMT) video camera use cases in Section III.3.

III.1. COLLATE

The landscape of blockchain technology continues to evolve, offering opportunities for innovation across various sectors. As blockchain networks and applications complexity grow, efficiently managing the computational load becomes a main challenge. This section presents the concept of *on-chain* and *off-chain* load balancing as a strategic solution to this challenge, ensuring that new distributed applications (DApps) with high complexity can be effectively integrated to blockchain environments, while reaching the trade-off between performance and security.

In the realm of distributed ledger, a contrast is made between the *on-chain* and *off-chain* operations, each serving distinctive roles within DApps. *On-chain* operations occur directly on the blockchain and are prized for their security and transparency. Each *on-chain* action is validated and recorded on the network's ledger, ensuring an immutable and transparent record that upholds the integrity and the well-functioning of the blockchain.

On the contrary, *off-chain* mechanisms create a supplementary environment designed to manage tasks and computation outside the main blockchain. This approach could be particularly beneficial for processing complex computations that are beyond the capacity of the main ledger, thereby enabling a broader spectrum of decentralized solutions. *Off-chain* solutions can significantly enhance the scalability and efficiency of blockchain applications by handling data-intensive or complex tasks that would otherwise bog down the blockchain with excessive processing demands and high transaction fees.

To face such an antagonism, it is essential to follow a load balancing strategy as early as in the design phase of any new DApps. This load balancing step involves partitioning the application into two distinct logical blocks, each tailored to meet specific needs:

- *High Sensitivity Block*: This segment of the application leverages the inherent security and transparency of the blockchain. It is ideal for functions that require strict security constraints, such as asset transactions and royalty distributions. By running these operations *on-chain*, the application ensures that all actions are transparent and tamper-proof, enriching the trust and reliability properties of the overall decentralized application.
- *High Complexity Block*: This segment is suited for the application tasks that demand extensive computational resources or complex computation, beyond the capabilities of typical Smart Contract languages (*i.e.* operations that require long processing times or the use of high-performance computing resources like GPUs). *Off-chain* mechanisms can efficiently handle such tasks by exploiting external systems or specialized hardware, thereby preserving the overall blockchain's performance and speed as well as the application specifications.

From the conceptual point of view, the orchestration in the joint execution of *High Sensitivity Block* and *High Complexity Block* requires the conception, design and integration of new type of component, further referred to as `Load Manager`. In other words, the `Load Manager` ensures the *on-chain / off-chain* balancing and the strict respect of functional rules of blockchains. The underlying logical workflow we advanced [ALL21] and is explained here after.

To this end, the `Load Manager` should secure a robust communication with both the blockchain and the off-chain computing environment, and the corresponding two connectors are illustrated in Figure 19.

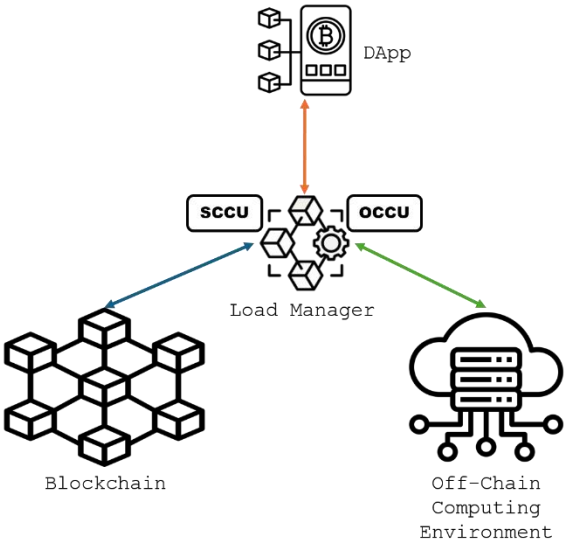


Figure 19: Generic On-chain/off-chain load balancing architecture: the communication between the load balancer and the DApp are presented in orange, the on-chain interactions are presented in blue, and the off-chain actions are presented in green

The communication with the blockchain is ensured by the *Smart Contract Connector Unit* (SCCU), which is directly linked to the blockchain, enabling it to execute transactions and

receive real-time blockchain updates. This connection is essential for remaining synchronized with the blockchain's state and to react to issues or changes as they arise.

For the communication with the *Off-Chain Computing Environment* (OCCE), the *Off-Chain Connector* Unit (OCCU) module is introduced. OCCU ensure a secure communication network *via* private network initialization and end-to-end encryption system between the `Load Manager` and OCCE.

The functioning of the on-chain / off-chain load balancing will be presented by considering the its three main stages, namely Initialization, Processing and Finalization.

III.1.A. Initialization

1. Smart Contract initialization

Smart Contract development is a critical phase for blockchain applications and both Tezos and Ethereum blockchains offer distinct approaches to the development and deployment of Smart Contracts, based on their distinct execution environments. This section shows how Smart Contract are initialized in Tezos and Ethereum, as illustrated in Figure 20, explaining both the similarities and differences between them.

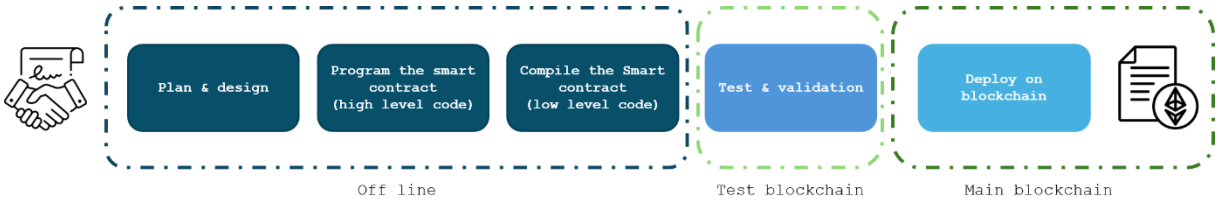


Figure 20: Conventional Smart Contract workflow

- *Plan and design* step forms the basis of Smart Contract development on both Tezos and Ethereum. This step requires a deep understanding in the legal field as well as the application sector. First, all the conditions and terms agreed on between all the parties should be listed and illustrated in the form of logic conditions (often structured as a series of `if...then...` statements). Second, the state management, interaction protocols, and the token standards (if needed) should be agreed on.
- *Program the Smart Contract* step focus on the translation of the conditions into high-level code that encapsulates the business logic of the Smart Contract, detailing transaction mechanisms, conditional executions, and data management procedures. Programming Smart Contracts on Ethereum often relays on Solidity language [GIT24a] while Tezos allows developers to use several officially supported languages namely Liquidity, SmartPy, and LIGO [LAN24].
- *Compile the Smart Contract* step transforms the high-level code into a blockchain-executable format. Ethereum compiles Solidity into bytecode executed by the Ethereum Virtual Machine (EVM). Tezos, on the other hand, compiles its high-level languages into Michelson code for execution on its own virtual machine.

- *Test and validation* is a compulsory step, given the significant risks raised by the immutability of the deployed Smart Contracts. Both blockchains stress the importance of rigorous testing and auditing, employing unit, integration, and system tests. These tests are typically conducted on `testnets` offering the same particularities as the `mainnet` but running on mock currency, eliminating financial risk during the testing phase. Additionally, Tezos enhances the reliability of its Smart Contracts through the integration of formal verification tools, which provide a mathematical guarantee that the contracts will function as intended.
- *Deployment* step sends the compiled Smart Contract to the blockchain where it becomes alive and public. The deployment step in both blockchains requires gas fees, although the cost may vary significantly depending on the network congestion and computational complexity of the Smart Contract.

2. `Load Manager` initialization

The `Load Manager` orchestrates the work be done by *on-chain* and *off-chain* resources in order to answer the DApp requests. This module functions within a robust environment that integrates various components to ensure seamless operations and enhanced security. To ensure the `Load Manager` is set to effectively handle its responsibilities, its initialization involves several key stages, as illustrated in Figure 21:

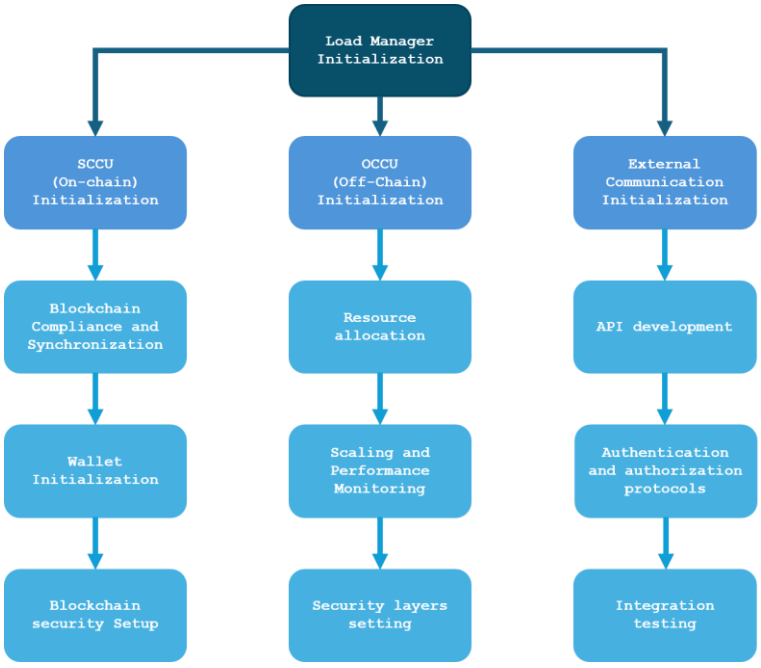


Figure 21: Simplified workflow for Load Manager initialization and configuration

- *On-chain related initialization* step configures the module to comply with blockchain protocols and settings. This includes setting up Smart Contract interfaces, initializing blockchain listeners, and setting transaction rules. Next, the synchronization with the blockchain network takes place. During this phase, the module is updated with the last blocks states, ensuring accurate and real-time

transaction validations and Smart Contract executions. Robust security protocols are also established during this step by implementing cryptographic procedures, such as digital signatures and hash functions, to secure transaction data and sensitive information. Finally, the blockchain wallets are created and set with an initial amount of asset to guaranty the execution of tests.

- *Off-Chain Related Initialization* step starts by setting up and configuring OCCU to handle computationally intensive tasks. To this end, the `Load Manager` determines the DApp needs related to the *off-chain* timely computation and storage resources. Security and high availability are of a high concern for the *off-chain* part. Security includes setting up encrypted channels for data exchange between the blockchain and external servers, such as SSL/TLS protocols [KRA13]. Additionally, enhanced isolation is achieved through the creation of private networks, providing an extra layer of protection and control over the data environment. The load balancing configuration, scaling strategies, and failure management are designed to ensure performance stability and high availability. Continuous performance monitoring tools (*e.g.* CPU and RAM usage) complement this setup.
- *External Communication Initialization* step covers the development and integration of Application Programming Interfaces (APIs) that coordinate the communication between the `Load Manager` and the DApp. Those APIs are associated with robust authentication and authorization protocols such as OAuth, API keys, or JWT (JSON Web Tokens) ensuring that only authorized DApps can initiate actions or access data managed by the `Load Manager`. Finally, the system conducts comprehensive integration testing to ensure that all communication interfaces work seamlessly with the DApps.

3. Off-Chain Computing Environment initialization

The *Off-Chain Computing Environment* (OCCE) is tailored to the specific needs of applications and may include a variety of powerful computing resources such as CPUs, GPUs, Application-Specific Integrated Circuits (ASICs), or Field Programmable Gate Arrays (FPGAs). OCCE serves as a complement to blockchain by executing the computationally intense algorithms, which are far beyond the capability of standard blockchain technologies and/or by ensuring gas-fee exempted storage.

The initialization of the OCCE shared numerous aspect with classic servers. It is configured to manage high-load processes and provide computational power where needed. This configuration ensures that the OCCE can handle large-scale computations seamlessly by the adaptation of automatic scaling mechanisms. At this point, all the algorithms and operations running on the OCCE are pre-programmed and all participating parties have agreed on their functioning. This agreement is significant for maintaining trust and transparency among all stakeholders and users, thus aligning with the decentralized nature of blockchain technologies. Complementary measurements can be implemented in the core of the algorithms to further insure the system trust like AI model watermarking [TRI24]. The specification of disaster recovery plans should also be agreed on in advance.

To this end, the continuous monitoring[2] tools already mentioned are installed as well. Other than the computing capabilities, advanced security protocols are deployed to protect against different type of attacks and potential security breaches.

III.1.B. On-chain / off-chain processing

Once the system is initialized, it is autonomously functioning, without the need for any further external intervention. This self-effectiveness is highlighted in the lifecycle of a DApp request, from its request to the receiving of the expected results, as illustrated in Figure 22:

1. *DApp request*: The processing is started by DApp, that, based on user interactions such as clicks or command inputs, sends a request to the `Load Manager`. This request specifies the nature of the task, which may involve data retrieval, computation, or transaction execution.
2. *Transaction via SCCU*: The load manager analyses the request and define the *on-chain* part and the *off-chain* part. Then it initiates the transactions with the blockchain using the SCCU, executing either asset transactions or instructions within a Smart Contract. The SCCU ensures that these transactions are constructed, signed, and broadcast correctly. The transaction construction includes defining the core transaction details such as the sender, recipient, amount, gas limits, and nonce, while also formatting inputs to comply with the application's binary interface (ABI) specifications.
3. *Blockchain Processing and Event Detection*: once the transaction is submitted, the blockchain network processes it. The `Load Balancer`, actively monitoring the network, detects events such as transaction confirmations or Smart Contract triggers. This event-driven approach allows the system to responds dynamically to changes or results from the blockchain.
4. *Data Parsing and Job Creation via OCCU*: After detecting the relevant blockchain events, the `Load Manager` parses the transaction confirmations or Smart Contract state changes and events. Using the OCCU, it formulates job requests for the OCCE modules.
5. *Job Execution by OCCE*: The OCCE receives the job added to a queue and executes it as soon as one of the resources is available.
6. *Result Handling*: Upon completion of the *off-chain* computation, the results are encapsulated and securely transmitted back to the `Load Manager`.
7. *Completion Confirmation to DApp*: Finally, the `Load Manager` confirms the successful completion of the entire process to the DApp and sends back the obtained results as well as the confirmation of the blockchain transaction, allowing the DApp to update accordingly and inform the user.

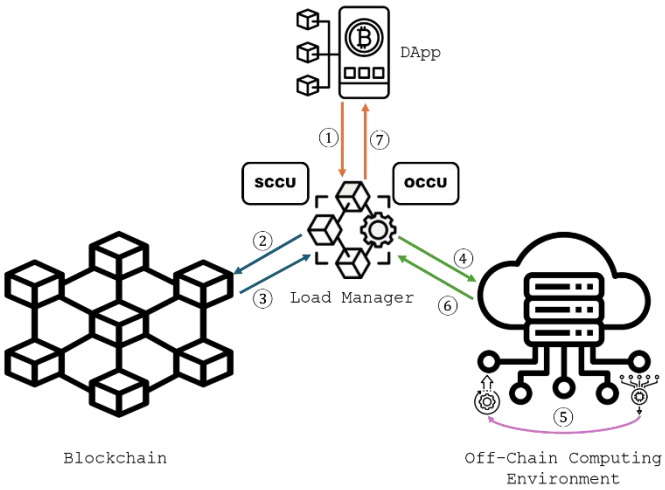


Figure 22: Interaction scenario between the DApp and the load manager

III.1.C. Finalization

In order to finalize the *On-chain / Off-chain* load balancing system, a structured and methodical approach is of a great importance to follow a specific order in shutting down the componence.

1. The *Off-Chain Computing Environment* is the first element to turn off. A cool down system is required for this step: any existing jobs should be allowed to finish and return the result, while any new specific computing job should be denied. This process can take from seconds up to several minutes (the maximum duration can be set by the system administrator) based on the number and nature of the processes executed on the environment. All the processes in Idle state will shut down directly.
2. Once all tasks in the OCCE are completed, the OCCU can now be safely decommissioned by releasing all the resources that were dedicated to *Off-chain* computations.
3. After a stabilization period where the unperformed jobs get reimbursed, the communication channels between the Load Manager and the DApp are closed. This turns the system inaccessible for any further request from the user.
4. As the Smart Contracts deployed by the `Load Manager` could still have assets attached to them, the system administrator makes a series of transaction to the Smart Contracts to direct all the available asset to a secure wallet.
5. The SCCU is next to be turned down as it is the end of the functional life of the Smart Contracts, and no more communication with be blockchain is required.
6. The final step is to shut down the Load Manager module. This should be straightforward and easy task since all its communication with the three parties has been ended.
7. In the case of using a private blockchain that served only for this application, this is turned down by disconnecting and turning off all associated nodes.

III.1.D. Potential usage

COLLATE has been conceived and design with the specific purpose of keeping its workflow at a conceptual level, thus ensuring independence with both the blockchain structure and the specific DApp.

The previous sections show that at least up to some extent, two different blockchain infrastructures (Ethereum and Tezos) can equally accommodate such a solution, assuming the specific needs related to initialization and development of Smart Contract are properly delt with.

From thee workflow standpoint, each step is evaluated based on its requirement for security, cost, speed, and complexity to determine the most suitable processing environment. By well-balancing *on-chain* and *off-chain* operations, the system maintains the blockchain properties while addressing its limitations, resulting in robust, scalable, and efficient DApps capable of supporting complex and dynamic use cases across various sectors as illustrated in Table 4.

Table 4 Examples of use cases that can benefit from the on-chain/off-chain load balancing

Blockchain application	*On-chain*	*Off-chain*
Financial investment, *e.g.* [ZHA22]	- Risk level agreement - Asset transactions - Management fees	- Real time market tracking - Risk assessment algorithm - Market predictions
Cold Chain Management, *e.g.* [BAD18]	- Provenance tracking - Temperature and storage condition monitoring - Quality certifications - Product spoilage alert (error)	- Route optimization - Supermarket inventory prediction
Healthcare, *e.g.* [AL-N24]	- Permission for patient record access - Allergy and disease logs - Drug prescription	- MRI scan analysis - NN model for brain tumor detection - Genomic data analysis
Gaming, *e.g.* [PFE20]	- In-game purchases (*e.g.* Gems) - In-game item ownership transfer (sell, trade, ...) - Service subscription	- Real-time multiplayer gameplay - AI-driven non-player character (NPC) actions
Media and copyright, *e.g.* [MOR23b]	- Royalty distribution - Licensing and content usage rights (personal, commercial, distribution, ...)	- Near-duplicate tracking - Content recommendation system - QoS optimization

III.2. Implementation

The following section presents the development grounds for the *on-chain/off-chain* module, ensuring optimal load balancing and efficient resource utilization.

The workflow, synoptically presented in Figure 23, is structured into distinct modules to streamline development and ensure robust system performance upon deployment.

Different modules may require the expertise of different engineers and developers varying from system architect for planning and designing the different modules, application specific developer (*i.e.* a neural network classifier requires a data scientist meanwhile a high-resolution medical imaging requires an embedded systems engineer).

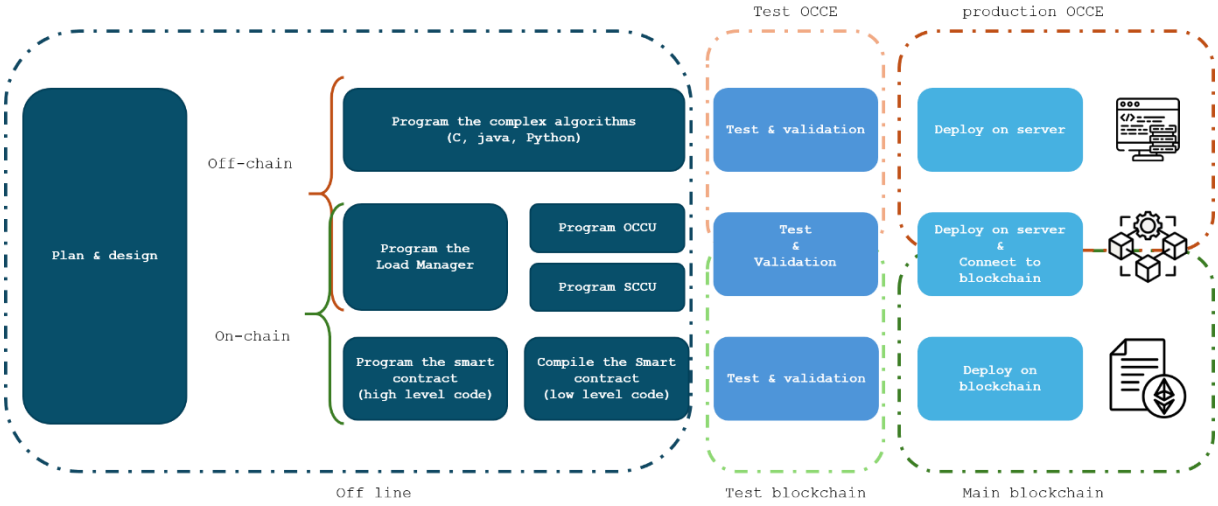


Figure 23: On-chain/off-chain load balancing architecture development workflow

III.2.A. Plan and Design

This foundational phase involves strategic planning and detailed design, setting the stage for subsequent development efforts. The project's scope and requirements are defined, focusing on the overall project challenges and possible issues. This includes selecting the *off-chain* computation needs and the well-suited blockchain network understanding the transaction types, Smart Contract capabilities offered by each blockchain.

III.2.B. Programming

The programming phase of the *on-chain / off-chain* load balancing architecture is broken down to three major sections:

- The Smart Contracts and tokens are developed to ensure the applicative continuity yet to benefit from the security and immutability aspects within the blockchain. For applications that run on multiple blockchains, it is important to ensure that all the functions have the same declaration, same input and same output, which will facilitate the interaction with the SCCU. To further facilitate the interaction

between the Smart Contracts and the SCCU, `Events` are implemented in the Smart Contract functions to trigger the targeted conditions, as illustrated in Figure 24. On the other end on the system, Figure 25 shows how the events are handled in the `Load Manager`. Similar to the `Events`, alerts with clear and details messages are returned in case of an important condition is not satisfied as shown in Figure 26 and Figure 27. A security aspect in developing these Smart Contracts is to have a function that allow to securely managing the authorization aspects (*i.e.* list of SCCU addresses have the permission to transact Smart Contract functions that the public blockchain users cannot). The additional step of compiling the high-level code in order to obtain blockchain executable code. The Solidity code results in two files, the *Bytecode* which is the executable part but also the Application Binary Interface (ABI) The import and initialization of the Smart Contract *via* the ABI is illustrated by Figure 28.

- The `Load Manager` is composed of three main blocks. First, the central controller, determining whether tasks should be executed *on-chain* or *off-chain* based on predefined criteria such as transaction cost and execution speed. Second, the SCCU is a piece of software (*e.g.* developed in Python3 in the thesis) to interface with the Ethereum and Tezos blockchains, handling transaction submissions and Smart Contract interactions. It uses the Web3 library [MET24] for Ethereum, Figure 28, and PyTezos library [PYT24] for Tezos to facilitate communications. Third, the OCCU is developed to manage and interact with the OCCE, securing the communications and collecting continuous health checks. A monitoring dashboard for the OCCE metrics can be coupled with OCCU.
- The complex algorithms are developed using conventional programming languages like C/C++, JAVA, or Python based on the task requirements.

```
//events
    event startMPEGV ( address indexed _from, string _value );
    event startVideo ( address indexed _from, string _value );
    event startImage ( address indexed _from, string _value );
    function getMPEGVCamera(string memory tid) public payable returns(string memory){
    buyTokens(7);
    emit startMPEGV(msg.sender, tid);
    return "tokens sent to buyer" ;
}
```

Figure 24: Snapshot of the event declaration and emission in Solidity

```
def handle_event(event):
    # the event is of type dict
    logging.info(f"Author: {event['args']['_from']}, token ID: {event['args']['tid']}")

# Create a filter for catching the latest events
event_filter = contract.events.ValueChanged.createFilter(fromBlock='latest')

while True: # this should run on a seperate thread
    for event in event_filter.get_new_entries():
        handle_event(event)
    Web3.eth.waitForBlock(Web3.eth.blockNumber + 1, timeout=60) # block wait
```

Figure 25: Python Code for Listening to Ethereum Events

```
function main(string memory clauseId, string memory location) public payable {
    require(clauses[clauseId].exists, "clause id is not known");
    clause memory cl = clauses[clauseId];
    require((block.timestamp >= cl.afterDate) && (block.timestamp <= cl.afterDate), "check temporal interval");
    require(containsLocation(clauseId, location), "This item is not available in your country");
    if (msg.value < cl.cost) {
      revert("Not enough money");
    }
}
```

Figure 26: Failure condition in Solidity

```
let%entry main (clauseId , location) storage = begin
  let cl = match Map.find clauseId storage.clauses with
    | None -> failwith ("clause id is not known",clauseId)
    | Some x -> x ;
  in
  if ((Current.time () < cl.afterDate) or (Current.time () > cl.beforeDate)) then failwith ("temporal iterval is " , cl.afterDate ,"and", cl.beforeDate);
  if not ( Set.mem location cl.spatialLocation ) then
    failwith "This item is not available in your country";
  if Current.amount () < cl.cost then failwith "Not enough money";
  if cl.runs < 1 then failwith "maximum runs exceeded";
  let newruns = cl.runs - 1 in
  let cl = cl.runs <- newruns in
  let storage = storage.clauses <- Map.update clauseId (Some cl) storage.clauses in
  ([],storage)
end
```

Figure 27: Failure condition in Liquidity

```
HTTP_PROVIDER_URL = "http://127.0.0.1:8545"

def connect_blockchain(provider_url):
    """Connect to Ethereum blockchain"""
    web3_instance = Web3(Web3.HTTPProvider(provider_url))# Initialize instance
    if web3_instance.isConnected():# Check if the connection is successful
        logging.info("Successfully connected")
        return web3_instance, True
    else:
        logging.error("Failed to connect")
        return None, False

def load_contract(file_path, web3_instance):
    """Load a contract given a file path and a Web3 instance"""
    try:
        with open(file_path, 'r') as f:
            datastore = json.load(f)
        abi = datastore["abi"] # parse the json to get the ABI
        contract_address = datastore["contract_address"] # get the address
        return web3_instance.eth.contract(address=contract_address, abi=abi)
    except Exception as e:
        logging.error(f"Failed to load or parse contract file {file_path}: {str(e)}")
        sys.exit(1)

def main(contract_file, token_file):
    # Connect to Ethereum client
    web3_instance, is_connected = connect_blockchain(HTTP_PROVIDER_URL)
    if not is_connected:
        logging.error("Unable to connect to the blockchain..")
        sys.exit(1)
    web3_instance.eth.defaultAccount = web3_instance.eth.accounts[1]
    # Load the contract
    contract = load_contract(contract_file, web3_instance)
    logging.info(f"Contract loaded at address: {contract.address}")
    # Load the token
    token = load_contract(token_file, web3_instance)
    logging.info(f"Token loaded at address: {token.address}")
```

Figure 28: Python code to import Smart Contract and token addresses and ABIs after being deployed

III.2.C. Test and validation

This testing phase starts with validating the three developed sections followed with testing the overall application integration:

- *Testing the Smart Contracts* encompasses a series of validations to confirm that they operate as intended. This process begins with unit testing, where each function of the Smart Contract is tested in isolation to early detect any issue. Integration tests follow, where the Smart Contracts are deployed on `testnets` to audit the way they would function in a live environment. The integration tests help identifying any issue interacting with the blockchain platforms or unexpected behaviors under different conditions. The main objective is to ensure that the Smart Contracts are secure, performant, and free from vulnerabilities that could be exploited once deployed which could be fetal to all the system since malfunctioning contracts cannot be fixed or deleted from the blockchain.
- *Testing the* `Load Manager`, as the central controller of the system, requires attentive testing to validate its ability correctly routing the tasks between *on-chain* and *off-chain* processes as intended. Functional testing and performance testing are performed to evaluate the `Load Manager` efficiency under various load scenarios to ensure that it can handle real-world operating conditions. Stress tests are applied to gauge its robustness and recovery from extreme conditions, ensuring the Load Manager remains stable and reliable.
- *Testing the Off-Chain Program* relates to validating the complex algorithms focus on computational accuracy, ensuring that the algorithms perform the expected calculations correctly and efficiently. Security testing is performed to safeguard the *off-chain* components from potential cyber-attacks, ensuring that all the resources are secure from unauthorized access or manipulation.
- *End-to-End System Testing* represents the end-to-end system testing. It evaluates the complete system performance, usability, and reliability validation. It includes user acceptance testing to ensure the system meets all specified requirements and is ready for live deployment. Realistic scenarios that replicate typical DApp actions are employed to see how the system manages and executes these within the live operational environment. This final phase of testing helps guarantee that the system is fully prepared for deployment and operation, facing real-world applications at scale.

III.2.D. Deployment

The deployment phase of the *on-chain / off-chain* load balancing workflow is transitioning all components into a live operational state. Although the development and testing phases of the system components could be done in any order, the deployment phase should follow a precise order: (1) Of-Chain Computing Environment and the Load Manager core application and Off-Chain Connector Unit, (2) Smart Contract Connector Unit and the blockchain specific initializations, (3) the deployment of the communication channels with the DApp.

III.3. Use case demonstration: celebs identification in live IoMT video camera

III.3.A. Celebs identification as an on-chain / off-chain deployment

The ISO/IEC 23093 series, also known as MPEG-IoMT (Internet of Media Things), outlines an architecture that includes APIs and data flowing between various Media Things (MThings) such as *MCameras*, *MMicrophones*, *MDisplays*, *MAnalysers*, or *MStorages*. [ISO20; ISO22; MIT19] This standard facilitates the design, orchestration, and operation of these devices across a broad range of tasks, including acquisition, rendering, processing, and storage of multimedia content. Additionally, the IoMT standards incorporate provisions for integrating blockchain technologies with MThings. Specifically, the standard defines APIs and mechanisms for transactions, enabling the use of digital currencies or fiat money to pay MThings for their services, as the example of the *MCamera* use case that will be presented in this section.

The use of a live video camera in conjunction with a blockchain has been studied in [ALL21b].

The sequence diagram presented by Figure 29 illustrates the interaction flow between a decentralized application (DApp), a blockchain, an IoMT device represented by a Media Camera (MCamera) integrating a Smart Contract (presented in Figure 30), a `Load Manager`, and an artificial intelligence (AI) off-chain application. This use case is part of an IoMT system where various components work together to stream a live football event with players performance analytics. Here's a breakdown of the steps depicted in the diagram:

1. `GetVideoCostPerMinute`: The decentralized application (DApp) initiates a request to the *MCamera* to determine the cost of a football live stream per minute, specifying the cryptocurrency or token types that will be used for payment. the Solidity implementation of this method is shown in Figure 31.
2. `GetWalletAddress`: Upon the DApp request, the *Mcamera* responds by providing a wallet address specific to the transaction type. The address is derived from Smart Contracts deployed on the blockchain enhancing the transparency and reliability of the system.
3. `SendToken`: The DApp sends the required tokens to the wallet address received in the previous step. This transaction is executed on the blockchain, leveraging its secure environment to ensure that tokens are transferred without the risk of fraud or loss.
4. `CheckTransactionCompletion`: After initiating the token transfer, the DApp continually checks the blockchain to verify if the transaction has been successfully completed using the *transactionID* automatically generating when initiating the transaction.

5. `GetVideoURL`: Once the transaction is confirmed, the DApp requests the video URL from the *MCamera* passing the *transactionID* as a proof of payment.
6. `CheckTransactionCompletion`: This step ensures that the transaction related to the video stream request are completed and confirmed.
7. `GetPlayerID`: The DApp uses the video timestamp to request a player identity from the *MCamera*. This step is essential for correlating specific video frames with the correct player, facilitating accurate performance analysis.
8. `FaceRecognition`: The `Load Manager` associate to the *MCamera* transfer the DApp request to the off-chain AI Application who processes the video to perform face recognition and associates the recognized face with a player ID before sending back the results to the *MCamera*. The *MCamera* finally returns the *playerID* to the DApp.
9. `GetPlayerPerformance`: The DApp requests the performance data of the player using their *playerID* and an *eventID* from the *MCamera*. Note that, a payment process is also associated to this step, but it was not illustrated to simplify the sequence diagram.
10. `AnalyzePerformance`: The AI Application analyzes the player's performance using complex algorithms to assess metrics like speed, passes accuracy, distance run during the match and overall efficiency. This analysis is performed *off-chain* due to its computational intensity and sends the results back to the *MCamera* that returns it to the DApp for further use
11. `PlaceBet`: This step if not a part of the IoMT use case, it was added to show how the DApp could communicate with different entities over the same blockchain resulting in a complex and branched systems with multi actors.

Throughout this process, the *MCamera* interactions between the blockchain and the *off-chain* components demonstrate a dynamic use case in which blockchain technology ensures secure transactions and data integrity, while the *off-chain* components handle computationally intensive tasks like video analysis and face recognition. This architecture efficiently divides the workload between on-chain and *off-chain* processes, optimizing both security and performance in an IoMT environment.

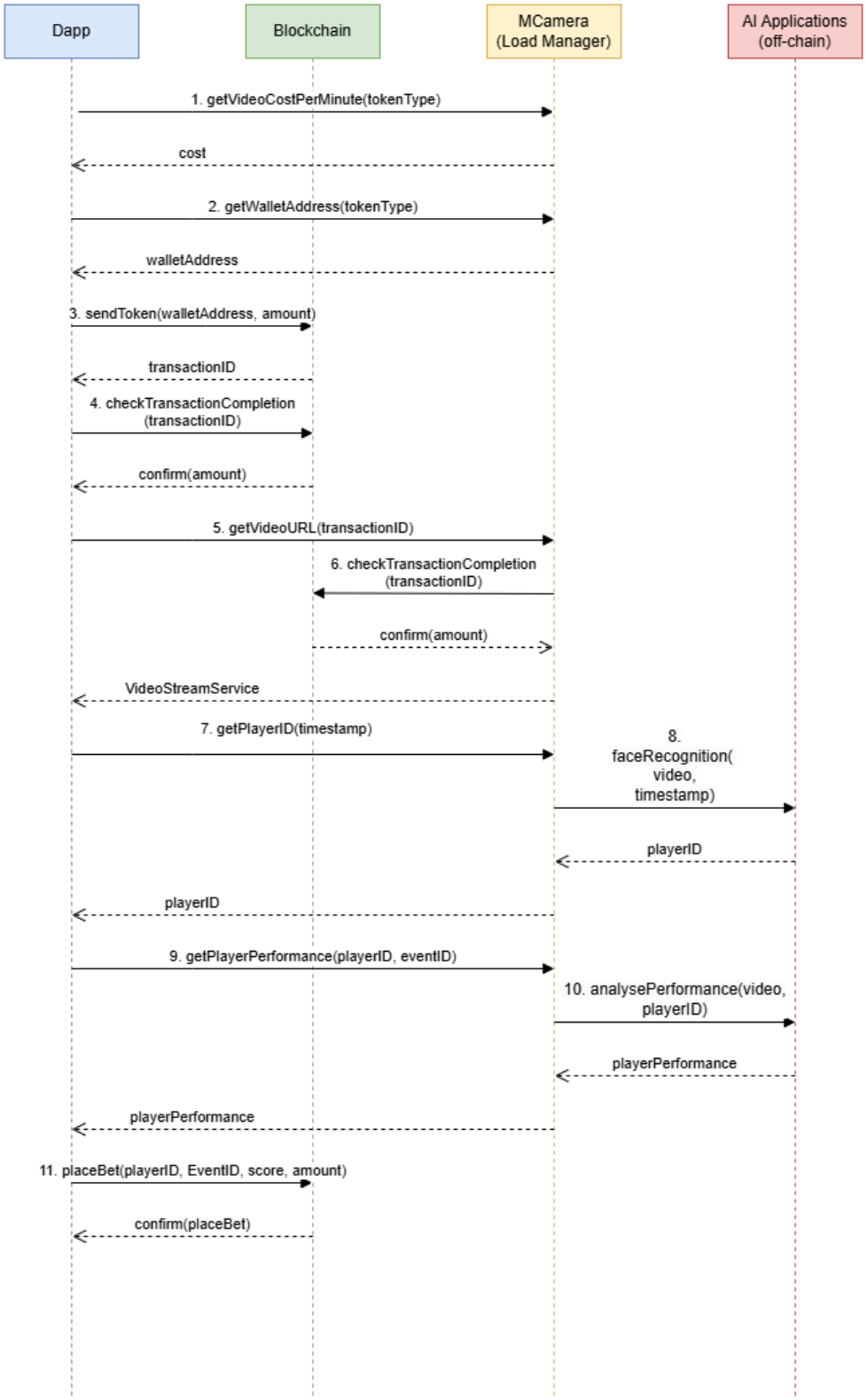


Figure 29: Sequence diagram for an IoMT use case involving DApp, Blockchain, MCamera, AI off-chain App

```
//events
event startMPEGV(address indexed _from, string _value);
event startVideo(address indexed _from, string _value);
event startImage(address indexed _from, string _value);
// constructors
constructor(
    string memory _id,
    string memory _serverIPAddress,
    int _serverPort
) {
    id = _id;
    serverIPAddress = _serverIPAddress;
    serverPort = _serverPort;
}
// internal Functions
function multiply(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require(y == 0 || (z = x * y) / y == x, "error miltiply");
}

function buyTokens(uint256 _numberOfTokens) internal {
    require(
        mCamFT.balanceOf(address(this)) >= _numberOfTokens,
        "require2: All tokens are sold out"
    );
    require(
        mCamFT.transfer(msg.sender, _numberOfTokens),
        "require3: unable to send token"
    );
}
```

Figure 30: MCamera Smart Contract initialization

```
function getMPEGVCamera_Cost(
    int8 tokenType,
    string memory tokenName
) public view returns (int256) {
    if (tokenType == 0) {
        if (MPEGVCameraCryptocurrencyCost[tokenName] == 0) {
            return -1;
        } else {
            return MPEGVCameraCryptocurrencyCost[tokenName];
        }
    } else if (tokenType == 1) {
        if (MPEGVCameraLegalTenderCost[tokenName] == 0) {
            return -1;
        } else {
            return MPEGVCameraLegalTenderCost[tokenName];
        }
    }
    return -1;
}
```

Figure 31: Smart Contract code to get the MCamera cost per minute

III.3.B. Experimental illustrations

This subsection illustrates the experiments related to the *on-chain / off-chain* deployment on Ethereum platforms. Similar results are obtained for the Tezos platform.

The user-friendly management of the Ethereum DApp through a visual interface is illustrated in Figure 32. The user has thus access to the main Smart Contract functions presented in Figure 29.

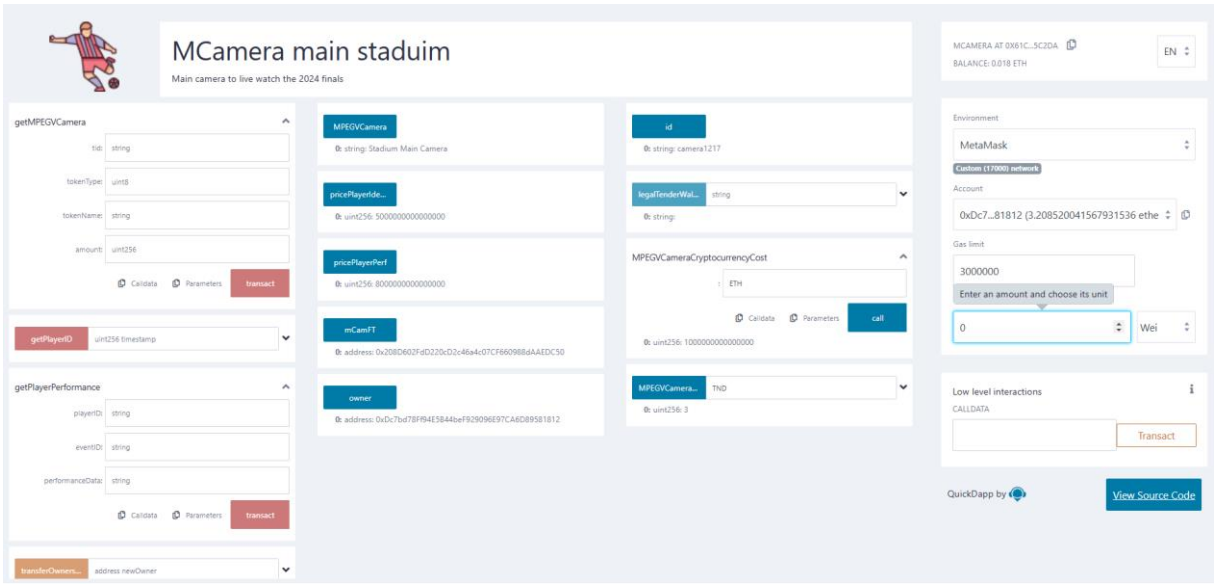


Figure 32: Visual interface for the Ethereum DApp

The on-chain functioning is illustrated through Figure 33 that shows the joint monetization of the services, according to the preestablished costs managed by the Smart Contract, and the blockchain transactional fees, imposed by the blockchain governance.

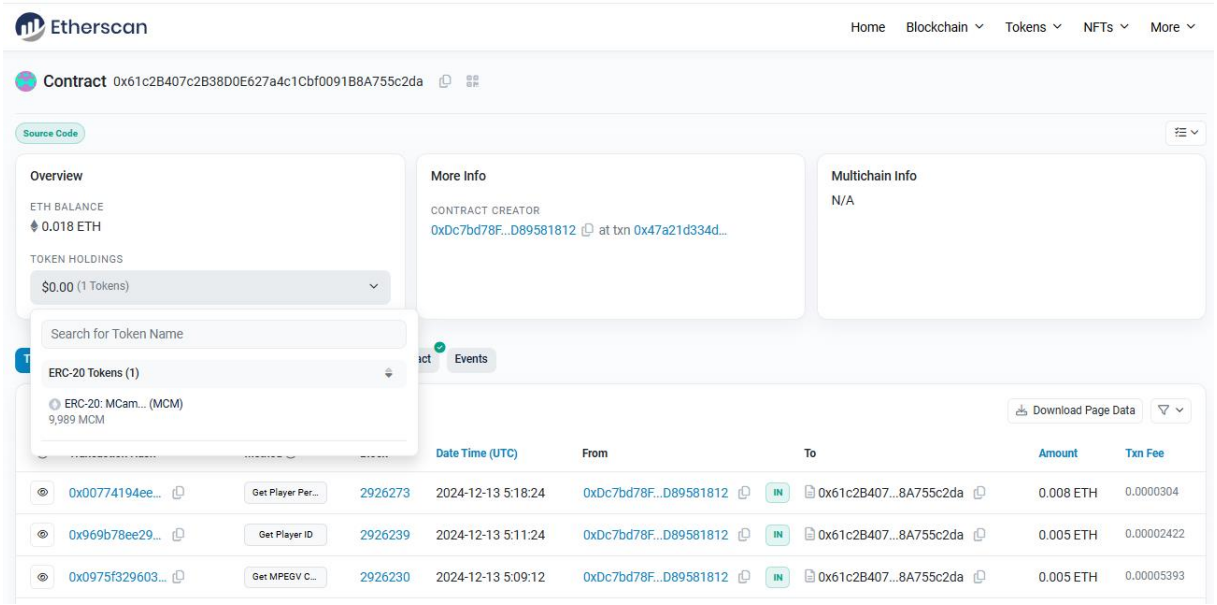


Figure 33: Service monetization and blockchain fees

The orchestration of the events triggering *off-chain* actions and the underlying token management (purchase and transfer) are illustrated in Figure 34 and Figure 35, respectively.

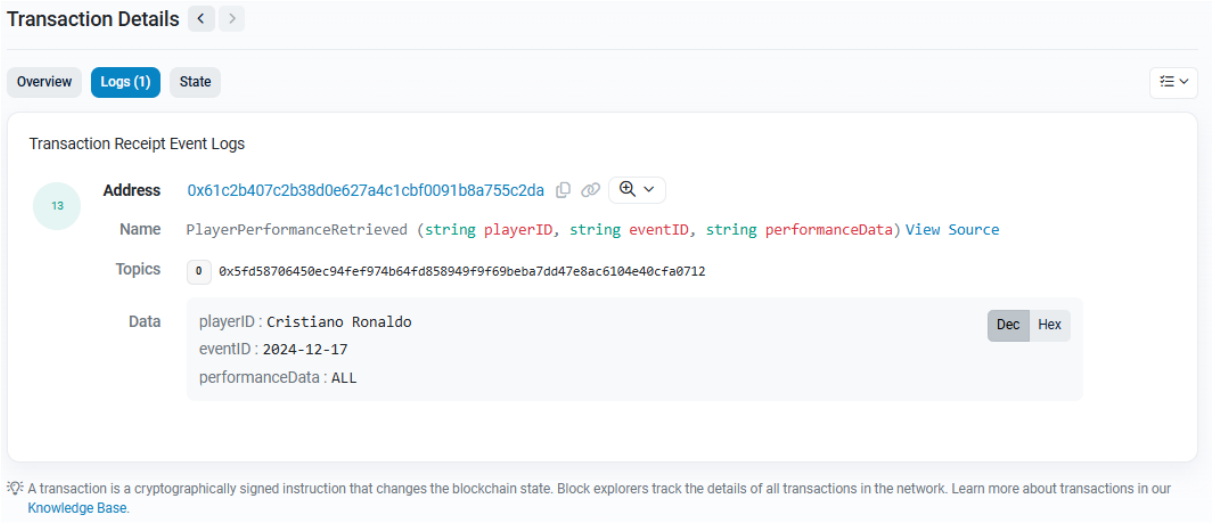


Figure 34: *On-chain* event triggering *off-chain* execution

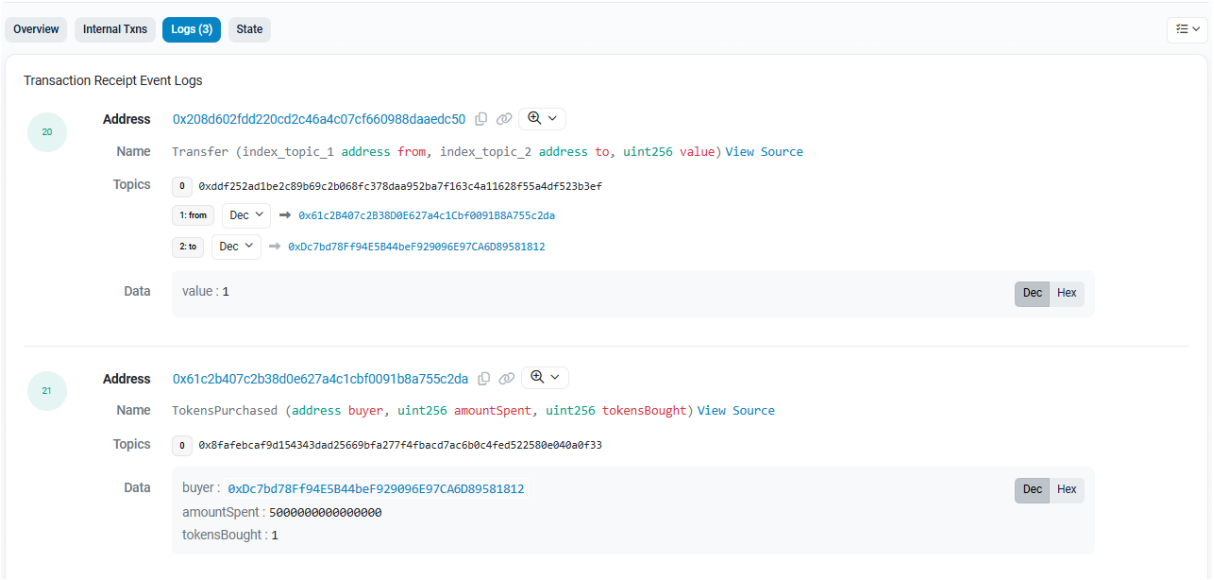


Figure 35: *On-chain* event triggering token management

III.4. Summary

The present Chapter starts from the conceptual contradiction between the blockchain restricted capabilities (computing, storage, exchanging) and their potential benefits for multimedia processing. To reach a trade off in this respect, COLLATE, *on-Chain Off-chain Load baLancing ArchiTecturE* is conceived, designed, open source implemented and experimentally validated.

COLLATE results in the functional optimization of blockchain applications, ensuring that each task is processed in the most appropriate environment. This strategic partitioning not only enhances the functionality and efficiency of DApps but also helps maintain the scalability of the blockchain infrastructure, making it more adaptable to a variety of advanced uses in an increasingly digital world. For instance, the Figure 36 illustrates the improvements COLLATE provides compared to state-of-the-art solutions with maximum gas fees of 55 603 for Ethereum and 2 100 for Tezos, that are significantly lower than the upper technical limits set at 30M and 1M, respectively.

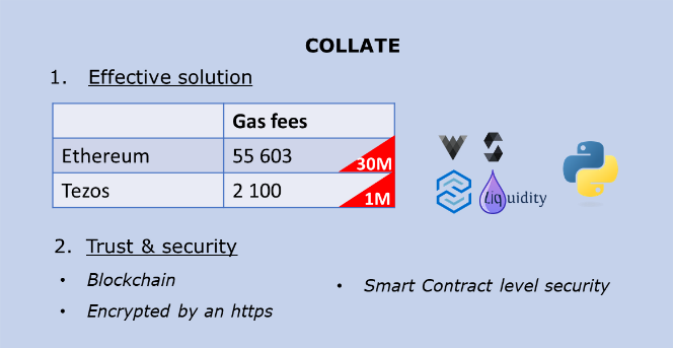


Figure 36: COLLATE effectiveness

We emphasize one key aspect related to the COLLATE security. Of course, when multiple processing environments are orchestrated, the data communication between the entities is intuitively considered as a security failure point, and particular attention was paid in this respect, as previously explained. Firstly, the data exchanged through the OCCU are encrypted by an https mechanism: in this way, the off-chain virtual machine is controlled only by trusted entities. Secondly, specific security constraints are considered during Smart Contract programming, in order to ensure the integrity and the non-repudiation of all accepted requests; implicitly, an attacker that was able to pass the https level of security is thereby referred from rewriting the *on-chain* data. Thus, the Smart Contract will record whatever actions trigger its entry points. Moreover, if an outcome in the contract depends on at least one user argument, the Smart Contract will not accept any additional user-supplied inputs until the off-chain process finishes the in-progress tasks.

This architecture considers the general blockchain principles, while the code presented in this manuscript correspond to Tezos and Ethereum.

Note that in the present thesis, we focused on the Smart Contract workflow; the issues related to tokens and to blockchain interoperability being addressed in [MOR23a].

As a final remark, note that to the best of our knowledge, COLLATE is the first of its kind solution for virtually extending blockchain infrastructures with computing, storage and programming facilities. It can be considered as an incremental level over the current-day ORACLE solutions [ETH24d] and [DOC24] that allows for external data to be input to Smart Contracts, without being able to make provisions for computation or storage resources related to those data.

Chapter IV. Compressed domain video fingerprinting

This chapter brings forth a comprehensive study about whether and how video fingerprinting can be achieved out of processing compressed stream elements. It encompasses aspects related to stream syntax element identification and extraction, to an ML based PoC as well as to DL based functional solution.

As the world increasingly turns to digital video use in all life aspects [CIS18], the need for efficient video fingerprinting becomes more and more important. Most of the studies reviewed in the state-of-the-art Section II.2 necessitate decoding the video streams prior to fingerprinting, a process that is both computationally expensive and time-consuming.

The current thesis explores the feasibility of compressed domain fingerprinting that offers the potential to bypass the decompression stage, thus promising significant gains in processing speed and resource efficiency. Yet, deep learning compressed domain fingerprinting reveals contradiction both at the conceptual and operational levels.

At a conceptual level, compressed domain video fingerprinting face three main challenges:

- How to extract the content in the compressed stream level while remaining robust against various encoding parameters and transcoding?
- How to represent a family of near-duplicate content by a unique fingerprint if no more redundancy is left at the level of compressed streams?
- How ML/DL models can be trained on data that does not have either spatial or temporal redundancy?

On the operational front, the main question relates to the very possibility of fitting video fingerprinting under the ML frameworks. On the one hand, the robustness property of the fingerprinting method dictates the possibility to track near-duplicate content, thus pointing to a conventional classification problem. Yet, conventional classifiers are *a priori* likely to contradict the fingerprinting uniqueness property.

Also on the operation front, note that ML and DL solutions are known to be complex from the computation point of view, so another *a priori* question relates to whether the computational overhead associated to cutting-edge classifiers (*e.g.* transformers) would not, in fine, contradict the very goal of fast, efficient fingerprinting.

When trying to mitigate these tensions, we followed a bottom-up approach. First, Section IV.1 Stream syntax elements and visual tracking discusses the compressed domain stream syntax elements and their potential usefulness for video fingerprinting applications. Secondly, Section IV.2 ML-based proof of concepts presents the proof of concepts for compressed domain video fingerprinting through a basic ML approach, namely decision trees. Then, the compressed domain fingerprinting problem is stated and solved in its general form under the DL framework in Section IV.3 COMMON. The issues related to scaling up towards real-life applications are discussed in Section IV.4 COMMON at work, while a retrospective view on the method is provided in Section IV.5 Summary.

Note that each step was designed to simplify the system's structure, focusing on maximizing efficiency and effectiveness.

IV.1. Stream syntax elements and visual tracking

IV.1.A. From human vision to stream syntax elements

The nowadays video encoding landscape features a dual trend. On the one hand, from the research perspective, the last decades have seen Advanced Video Coding AVC (formally referred to as H.264, or MPEG-4 Part 10) being replaced by High Efficiency Video Coding HEVC (formally referred to as H.265, or MPEG-H Part 2) and by Versatile Video Coding VVC (formally referred to as H.266 or MPEG-I Part 3). Yet, from the industrial perspective, the current day solutions mainly consider MPEG-4 AVC with HEVC expected to take over in the upcoming years [STE24]. To cope with this duality, the manuscript will present the main syntax elements related to HEVC; yet, the experiments are carried out for both MPEG-4 AVC and HEVC (only HEVC being included in the manuscript). Moreover, VVC will be studied as a robustness criterion.

HEVC significantly enhances the efficiency of video compression, offering a reduction in data requirements up to 50% compared to AVC while maintaining equivalent video quality. Alternatively, it can provide significantly improved video quality at the same bit rate, particularly for high-resolution videos. However, these enhanced compression performances come with increased complexity in both encoding and decoding processes [COR12]. The enhancements in HEVC, such as more flexible block partitioning, bigger blocks and larger variety of coding modes, demand higher computational resources, which can challenge its implementation particularly for real-time applications or with limited resources platforms, like single board computers, for instance (*e.g.* Raspberry Pi).

In most ways, HEVC is an improved, expanded version of AVC, employing similar encoding concepts and processes. The video encoding can be broken down to four key operations. First, video frames are divided into Groups of Pictures (GoP), each is composed of an initial *I* frame and followed by a sequence of *P* and *B* frames. In this step, each frame is partitioned into blocks. The encoder then predicts the similarities among the blocks in a frame and among the frames in the same GoP, transforms the prediction errors, Quantizes the resulting coefficients obtained and applies entropy Coding to further compress the video. The decoding process, presented in Figure 37, simply reverses main steps to reconstruct the video from the compressed file.

The complexity of decoding is associated with all video content analysis solutions since the vast majority of applications are processing fully-decoded video content as illustrated in the state of the art (Section II.2) where features are often extracted in the pixel level. Therefore, in order to reduce the fingerprinting algorithm complexity, the challenge is to extract the fingerprints from representations as close as possible to the binary compressed stream.

Consequently, the present study goes further and takes the challenge to investigate the possibility of ensuring fingerprint extraction closer to the compressed stream to ensure the trade-off between the decoding operation complexity and the level of redundancy still existing in the compressed stream. From this point of view, the decision to extract the fingerprint after the Inverse quantization and before the Inverse transformation was made, as illustrated in Figure 37. This decision is not only a strategic choice to optimize

complexity but also aligns with established practices in compressed-domain video watermarking, as evidenced by prior research [HAS14]. Extracting the fingerprint at this level is supposedly capable of eliminating the impact of Quantization parameter attacks making the overall system more robust.

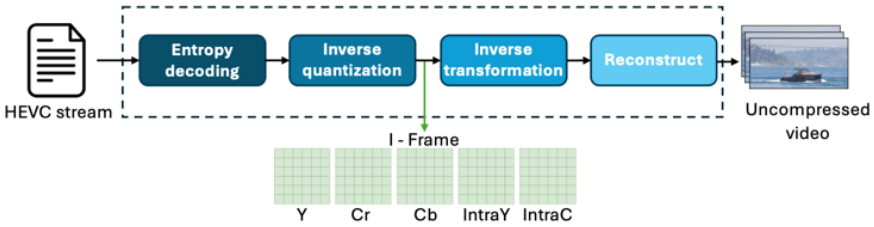


Figure 37: HEVC decoding process and fingerprint extraction

In order to further tailor the approach to the unique challenges of video fingerprinting, it is important to differentiate between videos that are semantically related but distinct, the method avoids using information from inter-frames. Instead, the focus is shifted to *I*-frames, which are self-contained frames that hold all the necessary visual information. This choice is also strategic because *I*-frames are less affected by the common variations introduced through repeated re-encoding, making the fingerprinting process more stable and reliable across different postings of the video. The exclusive selection of *I*-frame information reduces the size of the fingerprint and elements the needs to have an extra complex task to extract the key frames.

When determining which specific information to extract from an *I*-frame for video fingerprinting, the luma and chroma coefficients, as well as their intra prediction modes are *a priori* likely to be considered.

At this level, we shall consider 5 complete matrices: Luminance (Y), Chrominance red (Cr), Chrominance blue (Cb), intra prediction luminance (IntraY), and intra prediction Chrominance (IntraC).

The relationship between human vision and such stream syntax elements is illustrated in Figure 38, Figure 39, Figure 40 and Figure 41.

Figure 38 illustrates one *I* frame from which three macroblocks will be investigated: a top-left one (illustrated in red), a center one (illustrated in yellow) and a right-bottom one (illustrated in green). These three macroblocks have been selected as they correspond to three types of visual content: natural uniform (the block in red), natural with details (the block in yellow) and text (the block in green). When comparing among them Figure 39, Figure 40 and Figure 41, it can be noticed that the value and the distribution of stream syntax elements intrinsically depends on the type of content they are corresponding to.

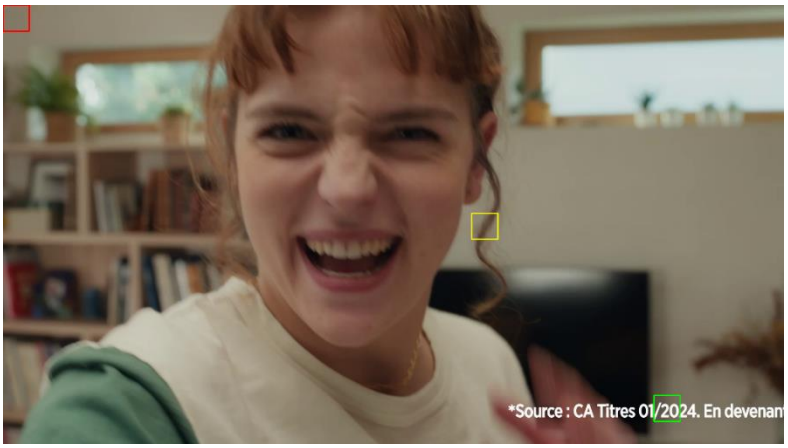


Figure 38 Representation of one *I*-frame, from which three macroblocks will be investigated: a top-left one (in red), a center one (in yellow) and a right-bottom (in green).

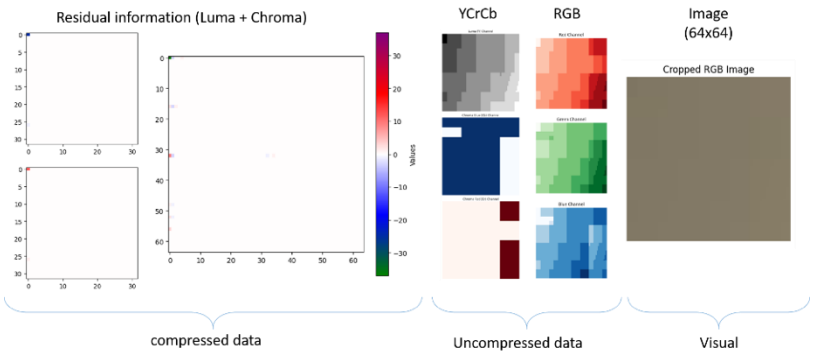


Figure 39: Different representations for the macroblock in red (top-left)

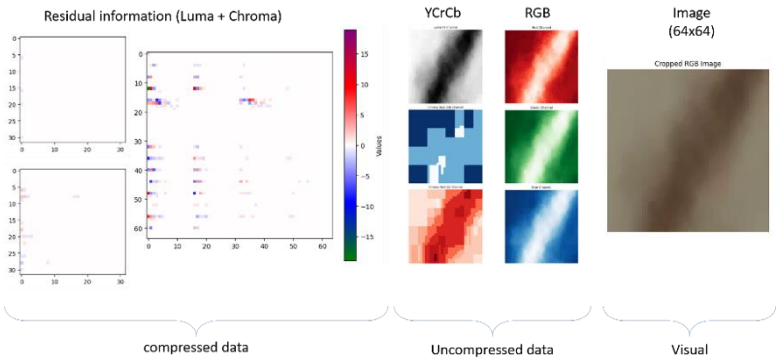


Figure 40: Different representations for the macroblock in yellow (center)

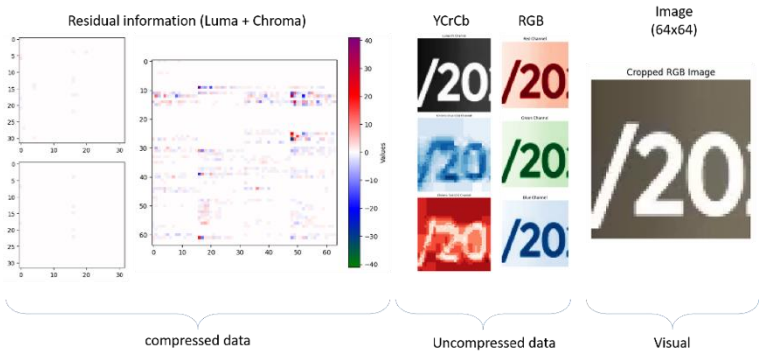


Figure 41: Different representations for the macroblock in green (bottom-right)

IV.1.B. Extracting stream syntax elements

In order to extract the syntax elements, a parser was developed. Among the four decoding main steps, only the Entropy decoding and the inverse quantization are required, as illustrated in Figure 37.

The parser is integrated in the open source HEVC reference software [GIT24c]. The reference software offers different applications, but during this study, only the `TAppDecoder` and the libraries communicating with it are updated.

The parser exclusively extracts the *I*-frame and save the residual syntax elements, which are composed of the luma and chroma residuals. These residuals represent the difference or error between predicted and actual pixel values in a compressed video, highlighting changes in brightness and color that are not captured in prediction compression phase.

Alongside with the Luma and Chroma residuals, the Intra prediction modes are also recorded. The obtained information includes the method of prediction (*e.g.*, spatial or temporal prediction), its associated prediction parameters, and the prediction error signal (referred to as the residual signal).

For each frame, the parsed syntax elements are collected and grouped by type. Then, the syntax elements from all the video frames are collected and stored. The output of the parser is recorded in a collection of TXT files to facilitate future use on the extracted syntax elements.

The parser developed in the thesis is available in open source at [GIT24e] and its functioning is illustrated by Figure 42, Figure 43 and Figure 44.

```
TAppDecoder::decode()
└── TDecTop::decode()
    └── TDecSlice::decodeSlice()
        └── TDecCu::decodeCtu()
            └── TDecCu::extractFingerprint(pCtu,fileName,iframeNb, isLastCtuOfSliceSegment)
```

Figure 42: Flow of function calls leading to the parser

The Figure 42 illustrates stream syntax elements flow, as implemented in the `TAppDecoder` method. The primary function, `TAppDecoder::decode()`, serves as the initial entry point, followed by `TDecTop::decode()`, which orchestrates the stream decoding operations. To process video slices, `TDecSlice::decodeSlice()` is invoked, wherein the decoding is further refined by `TDecCu::decodeCtu()`. This function processes the Coding Tree Units (CTUs). At this stage, `TDecCu::extractFingerprint()` checks if the CTU type correspond to an *I* frame, as illustrated in Figure 43, then extracts the syntax elements from each CTU, subsequently concatenating them into a single output file per frame for each element type.

In cases where luma and chroma share similar intra prediction modes, HEVC includes the `DM_CHROMA_IDX` flag to facilitate this linkage. However, for more precise intra prediction

chroma mode tracing, the chroma modes are associated with their modes rather than relying on that flag, as shown in Figure 44, ensuring that the chroma mode predictions are as accurate as possible, thus capturing the true characteristics of the video data.

```
if (curSlice->getSliceType() == I_SLICE)
{

  int *pLumaResidual = pCtu->getCoeff(ComponentID::COMPONENT_Y);
  int *pCrResidual = pCtu->getCoeff(ComponentID::COMPONENT_Cr);
  int *pCbResidual = pCtu->getCoeff(ComponentID::COMPONENT_Cb);
  int iPosX = pCtu->getCUPelX();
  int iPosY = pCtu->getCUPelY();

  string baseName = getBaseFileNameFromURL(fileName);
```

Figure 43: Check if the slice is I frame

```
UInt numPartitions = pCtu->getTotalNumPart(); // Total number of partitions in the CTU
for (UInt uiAbsPartIdx = 0; uiAbsPartIdx < numPartitions; uiAbsPartIdx++)
{
  {
    UChar intraDirLuma = pCtu->getIntraDir(CHANNEL_TYPE_LUMA, uiAbsPartIdx);
    outIntraLuma << static_cast<unsigned int>(intraDirLuma) << " ";
    UChar intraDirChroma = pCtu->getIntraDir(CHANNEL_TYPE_CHROMA, uiAbsPartIdx);
    if (intraDirChroma == DM_CHROMA_IDX)
    {
      outIntraChroma << static_cast<unsigned int>(intraDirLuma) << " ";
    }
    else
    {
      outIntraChroma << static_cast<unsigned int>(intraDirChroma) << " ";
    }
  }
}
```

Figure 44: Extraction of intra prediction modes for the Chroma component

IV.2. ML-based proof of concepts

Conventional video fingerprinting methods, operating in uncompressed domain, make intensive usage of basic ML solutions like KNN for instance, for clustering descriptors like SIFT and SURF, as detailed in Section II.2.

Consequently, as we expect the compressed domain video fingerprinting to be a more complex problem, we shall start our study by considering binary trees, conventionally considered as more performant than KNN [HAS18].

Decision tree is a non-parametric supervised learning ML model used for classification and regression tasks [SCI24]. The prediction process is composed of a sequence of comparisons of the input's features with pre-learned threshold values, as illustrated in Figure 45. Starting from the top node commonly referred to as the root node, and going downward towards the leaves, in each decision node the result of the comparison determines if the input goes left or right in the tree (generally left means that the condition was fulfilled, right means otherwise). When the query reaches a leaf (an end node) the decision is made. Although decision trees are recognized for their straightforward, interpretable nature, they are not without limitations, being prone to overfitting, especially if they grow deep without constraints [SCI24].

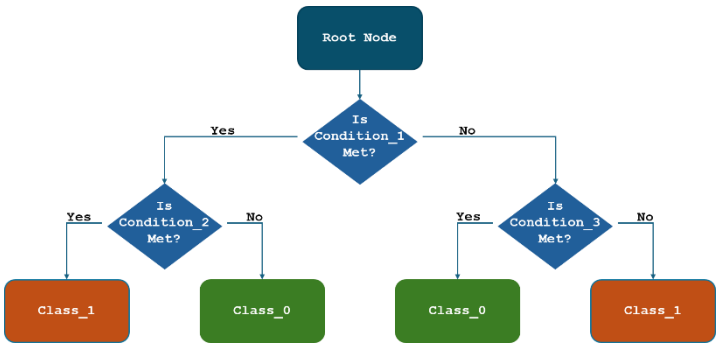


Figure 45: Binary decision tree principle

IV.2.A. Fingerprint extraction

As even the basic binary decision tree requires a series of comparisons, the full set of syntax elements extracted in Section IV.1 (Y, Cr, Cb, IntraY and IntraC) cannot be directly used as fingerprinting.

To reach a practical feasibility of the ML based fingerprinting, a two-steps preprocessing phase is considered.

First, all zero-value entries are removed from the data representation, thus reducing the fingerprint's size and drastically improve processing speed.

Secondly, the IntraY and IntraC information is skipped, and the Y, Cr, Cb values are represented by their statistics. This way, for each frame, and for the Y, Cr, and Cb

components, the fingerprinting is presented as the number of non-zero elements, the mean, the standard deviation, the minimum and the maximum values.

When combined, such frame-level fingerprints are concatenated to result in the entire video's fingerprint.

IV.2.B. Fingerprint matching

Building on the foundational principles of decision trees, the method we advance for fingerprinting matching leverages multiple binary decision trees to enhance the accuracy of the results, as illustrated in Figure 46.

The matching block is composed of *N* individual decision trees, where *N* corresponds to the total number of the original videos present in the dataset. Each binary decision tree is trained to identify one specific video, making decisions at each node. These decisions evaluate each frame fingerprint, directing the data down different branches of the tree based on the outcomes (Yes or No) of these evaluations.

The decision-making criteria at each node are predetermined during the specific tree's training phase that involves selecting the important features to be examined and setting thresholds for each. As the fingerprint navigates through the paths of a specific tree, it reaches the leaf nodes where a final classification is made on whether the query is a near-duplicate of that particular video or not.

Such a process is repeated along all the trees, which leads to a list of potential candidates to the video.

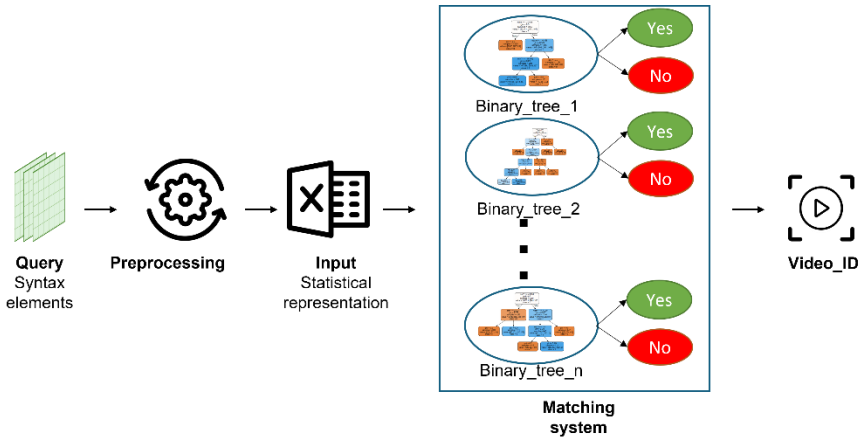


Figure 46: Binary decision tree-based method presentation

IV.2.C. Experimental results

1. Database specification

A comprehensive testbed comprising original video sequences and their altered duplicates are considered in the experiments. The corpus consists of 865 original video sequences of organic video, provided by VIDMIZER.

Each video sequence length varies between 6 sec. and 65 sec. In the case of longer videos, several chunks are extracted without having interlaying frames. For each original video sequence, we generate eight near-duplicate sequences through a series of systematic transformations. These transformations are designed to simulate common alterations that might occur in real-world scenarios as illustrated in Figure 47 and include:

- **Picture-in-Picture (PiP)**: This involves embedding a smaller version of a different video or a logo all along the original video duration, mimicking scenarios where additional content is overlaid onto the primary video.
- **Blurring:** To simulate the impact of loss of video details, three distinct blurring settings are applied. Each setting varies the intensity and spread of the blur effect, presenting the degradation that might occur during video transmission or from encoding artifacts.
- **Brightness Adjustment:** the attack alters the brightness of the video in three incremental steps. This transformation tests the fingerprint's sensitivity to changes in illumination and can be considered similar to some social media filters.
- **Text Insertion:** Overlaying text onto video frames simulates common modifications such as commenting or subtitling.

Each of these 8 transformations is applied with in conjunction with noise addition and re-encoding.

Figure 47: Database sample and attacks example; the commercial logo on the right-up corner was blurred in this illustration to avoid any potential conflict of interest

2. Result illustrations

The decision tree is based on the scikit-library [SCI24] with the 'gini' criterion, a maximal depth of *D*, the minimal number of samples to create a node equal to 2, and weights on the two classes to balance the decisions.

The fingerprinting detection is stated as a binary decision problem. The query extracted and formatted from any sequence is successively process by each and every decision tree.

The dataset is split in training and testing subset, on an 80% - 20% basis.

The process is illustrated in Figure 48, Figure 49 and Figure 50. Figure 48 illustrates the simplest case, namely a single video sequence, solely the *Y* component and $D = 3$. Figure 49 goes one step further in complexity, as it illustrates, for the same video sequence, the case in which both luminance and chrominance components are considered, and $D = 5$. Figure 50 resumes the case presented in Figure 49 for $D = 10$.

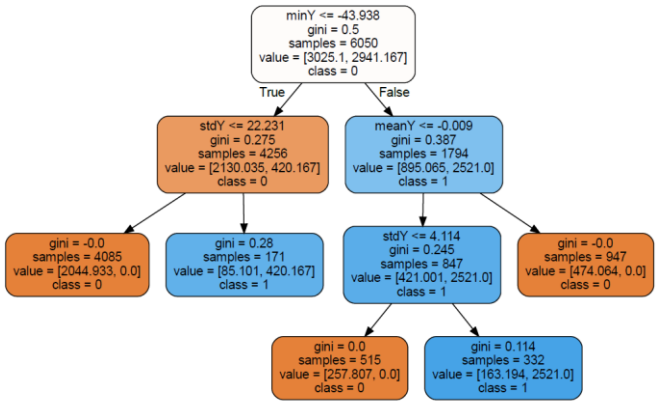


Figure 48 Example of the decision tree classification using only Luma syntax element; $D = 3$.

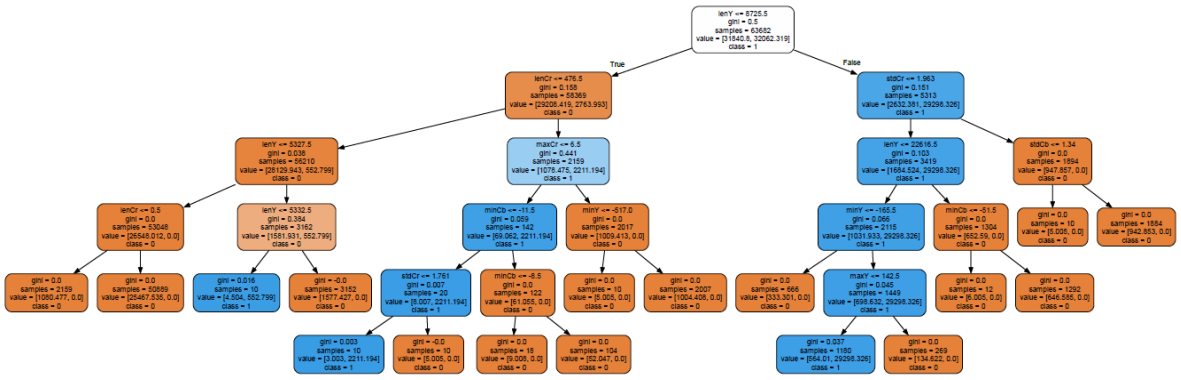


Figure 49: Example of the decision tree classification using Luma and Chroma syntax elements; $D = 5$.

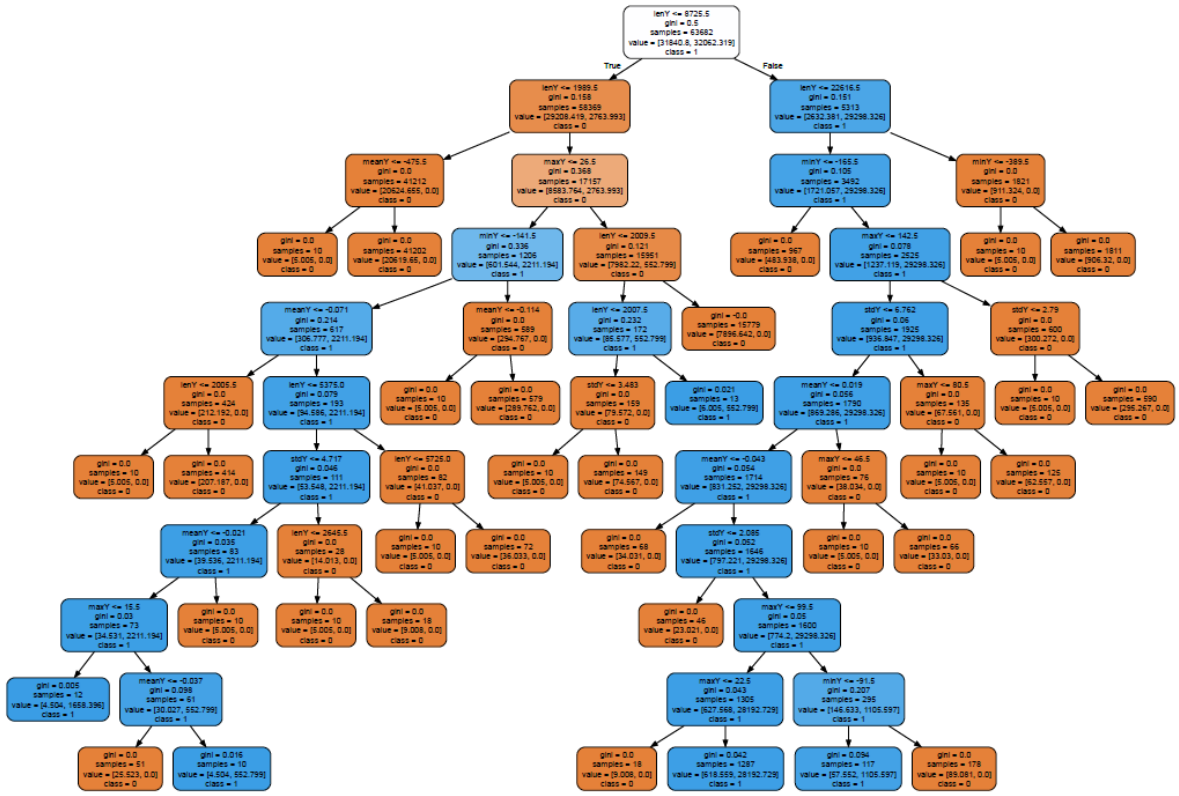


Figure 50: Example of the decision tree classification using Luma and Chroma syntax elements; D = 10.

The quantitative results obtained by computing the mean performances of all decision trees on the entire test dataset (original and attacked sequences) are presented in Table 5. They present the performance metrics of the matching method across different tree depths, using either Y component alone or combined Y, Cr, Cb components revealing clear patterns in *Accuracy*, *Precision*, *Recall*, F1 score, probability of false alarm (P_{fa}), and probability of missed detection (P_{md}) as the depth of the tree increases.

Table 5: Result for the decision tree matching method

	Depth	*Accuracy*	*Precision*	*Recall*	F1 score	P_{fa}	P_{md}
Y	3	0.82	0.99	0.82	0.90	0.17	1.9e-04
	5	0.87	0.99	0.87	0.93	0.12	1.9e-04
	10	0.94	0.99	0.94	0.96	0.05	3.7e-04
	15	0.97	0.99	0.97	0.98	0.02	5.7e-04
	20	0.98	0.99	0.98	0.99	0.01	7.7e-04
Y, Cr, Cb	3	0.88	0.99	0.88	0.93	0.11	1.3e-04
	5	0.92	0.99	0.92	0.95	0.07	1.5e-04
	10	0.96	0.99	0.96	0.98	0.03	2.6e-04
	15	0.98	0.99	0.98	0.98	0.01	3.6e-04
	20	0.98	0.99	0.98	0.99	0.01	4.6e-04

For both configurations, as the tree's depth increases from 3 to 20, a notable improvement in the *Accuracy*, *Recall*, and F1 score (for a preserved 0.998 value of *Precision*) has been obtained.

The same variation in *D* value is also reflected in better error probabilities: P_{fa} decreases from 0.170 to 0.011 in configuration using only the luma residual elements, and from 0.118 to 0.011 in the configuration using all three syntax elements residual; P_{md} remains low across all depths.

However, there are limits when increasing the *D* value, as it makes the solution more sensitive to overfitting [SCI24], especially when the depth exceeds the number of features which is 5 for the configuration using only luma, and 15 for the luma and chroma.

The values in Table 5 also show that both luminance and chrominance information are required for fingerprinting applications: integrating both luma (Y) and chroma (Cr, Cb) components tends to improve the accuracy of video fingerprinting models, as the increase of the number and of the diversity of features allows the construction of a more accurate model and a richer set of information for the model to learn from.

Note that the results presented in Table 5 are obtained in a highly unbalanced context: only 8 elements in the "near-duplicated" class and around 7500 in the complementary class and show a significant number of false positives.

When going deeper into a qualitative evaluation of these false positive, it was pointed out that they are mainly brought by content belonging to the same initial sequence (before it being chunked in less than 1min) or by content semantically close (yet different) from the query, as illustrated in Figure 51.

Figure 51: Example of false positives induced by the ML based method. The query sequence (at the left) results in false positive belonging to the same content but at different time moments (in the middle) or by visually related contents moments (at the right)

IV.2.D. Summary

The values presented in Table 5 can be considered as a successful PoC for our approach: it is thus demonstrated that video fingerprinting can be achieved in compressed domain.

In other words, it is demonstrated that conventional ML solutions, like binary trees, can learn features derived from compressed stream syntax elements.

However, note that the applicative performances of such solutions are intrinsically limited by three inner binary trees mechanisms:

- the computational limits impose the use of statistics computed over the stream syntax elements instead of the stream syntax elements *per-se*,
- the depth of the trees that is *a priori* expected to increase the performances is limited by an overfitting side effect,
- the memory footprint prohibitively grows with the number of video sequences to be fingerprinted.

Hence, when putting these results in the perspective of a practical application, several shortcomings are encountered. First, the performances in terms of false positives are close to 10% and this is mainly due to frame-level visual similarity. Note that the inherent complexity of binary trees makes complicated the use of Intra prediction information. Secondly, the approach is inherently bound to binary decisions and is conceptually unscalable, at least when imposing a pre-established complexity threshold.

Consequently, the results presented in this section should be considered just as door open towards future investigations that will be presented here-after.

IV.3. COMMON

To overcome the challenges presented in the Section IV.2, the COMMON is introduced using DL models to solve the compressed fingerprinting challenge.

IV.3.A. Fingerprint extraction

While using CNNs such as ResNet and MobileNet, the pre-processing of the input media is essential to optimize model performance. Traditional methodologies in the uncompressed domains generally manipulate data in the RGB color space, which exhibit uniform statistical properties conducive to straightforward processing. In contrast, syntax elements within the compressed domain present sparse distributions, predominantly characterized by zero-value coefficients as illustrated in Section IV.1.A. Thus, the standard image resizing techniques such as Nearest Neighbor, Bicubic, and Bilinear interpolations are not suitable for the data extracted in the compressed domain as they would amplify the sparsity, complicating the feature matching process.

In order to better accommodate the data characteristics of compressed video streams, a different resizing strategy is used to conduct this study, based on the method proposed by [GAR16]. A sorting algorithm that prioritizes elements according to the magnitude of their luma coefficients is implemented. This process involves sorting the frame elements in descending order based on their absolute values and extracting the top *N* coefficients. This method preserves the most significant data points that contribute to the overall video content definition as illustrated in Figure 52 and Figure 53.

Furthermore, for chroma residual elements and intra prediction modes, the selection is directly tied to their corresponding luma positions. Specifically, chroma and intra prediction elements located at the same spatial coordinates as the top *N* luma coefficients are also selected, regardless of their individual values. This ensures that the chroma and intra predictions are consistent with the significant luma elements, preserving the integrity of the video's color and texture information.

Figure 52: RGB color space resizing algorithms

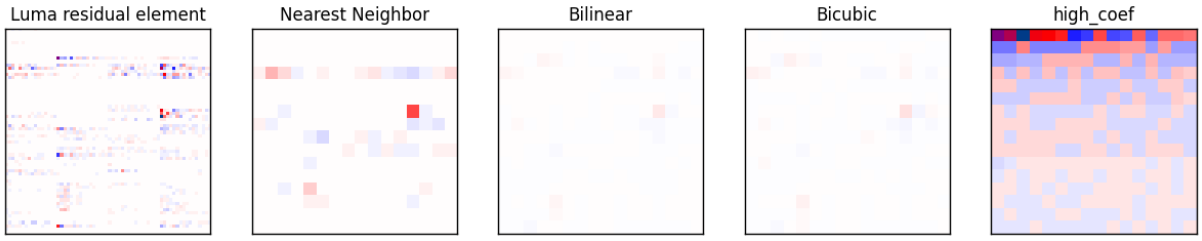


Figure 53: Residual elements resizing algorithms

This strategy is applied to the Y components, and the related Cr, Cb and prediction components are selected so as to represent the fingerprint.

IV.3.B. Fingerprint matching

In our study, we consider a DL structure composed a backbone and three additional layers, as illustrated in Figure 54.

Candidate models for the backbone components are the widest used convolutional Neural Network (CNN), namely the ResNet family [HE16] (ResNet18, ResNet50, ResNet101) and MobileNetV3small [HOW19]. Note that following our initial objective of reducing the end-to-end fingerprinting complexity, we avoid more sophisticated classification solutions, like transforms, for instance.

As usual in classification applications, a dense classification layer is positioned at the output (denoted by O3 in Figure 54), to map the previously obtained features to the final output classes. This backbone should be completed by additional layers meant to solve the two previous identified issues.

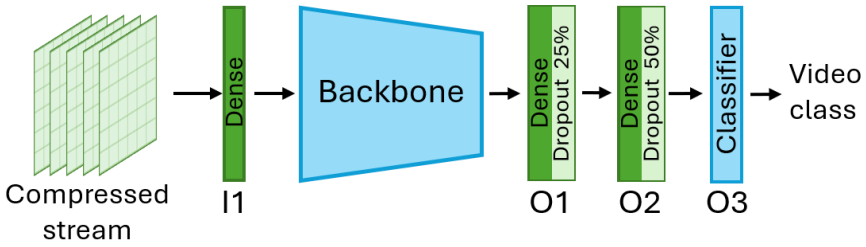


Figure 54: DL-based fingerprinting model: 1 input and 2 output layers (in green) are considered around the backbone and the final Classifier layer (in blue).

On the one hand, prior to the backbone, an input pre-processing layer, denoted by I1, is added. I1 is dense and is expected to serve for the weighting of the heterogeneous information included in the fingerprint (Y, Cr, Cb, and their corresponding intra prediction modes) to be learned. Note that in uncompressed domain fingerprinting, such a layer is not required, as the backbone is already designed for weighting the RGB components of a pixel.

On the other hand, two post-processing layers (denoted by O1 and O2 in Figure 54), are included in the model. O1 and O2 are both dense, with dropout rates of 25% and 50%, respectively. These two layers are added for two complementary reasons. First, as the backbone is fed with a combination of compressed stream syntax elements, its task is more complex to be learned than the conventional (pixel) classification task for which it was designed. Secondly, the fingerprint matching task should find the balance between the uniqueness and robustness properties, that translates on a trade-off between correctly retrieving near-duplicated contents and avoiding semantically related yet different contents.

IV.3.C. Experiment results

1. Database specification

Uncompressed domain video fingerprinting solutions benefit from already available databases. Specifically, conventional methods can be benchmarked on challenge databases (*e.g.* TRECVID) while deep-learning solutions on general purposes computer vision datasets, *e.g.* YLI-MED [BER15] or Youtube-8M [ABU16]. However, such databases cannot be considered in our study, as they are either already encoded with legacy encoders (like MPEG-2 or MPEG-4 AVC) or presented as tensors obtained from some specific key-frames. Consequently, our experimental study starts by organizing the reference content to be processed, and to this end, we shall consider two complementary databases.

The UVG database [LIU13] is available in raw format. In the present study, it is encoded by the VideoLAN implementation of HEVC [VID24]. UVG is composed of 16 natural 3840x2160 video sequences, with variable lengths ranging from 2.5 to 12 sec. The encoding process provides an average of 3.2 I frame per video (min. 2, max. 12). A total of 4983 frames is generated from this dataset. Note that UVG is designed as a generic end-to-end video encoding benchmarking database, with no direct relevance for the organic video applications. The organic video content is represented in our study by home-made database, referred to as VID and composed of 164 video excerpts, 1920x1080, from advertising content provided by an industrial company. This content was edited so as to not include duplicated/reused content in different sequences. VID sequences durations range from 5 to 180 sec. and they contain an average of 6.5 I frame per video (min. 1, max. 41). A total of 117769 frames are included in this dataset.

These two reference databases are subsequently subjected to a set of 10 different near-duplicated transformations, as follows. Firstly, 5 luminance/colorimetry modifications are applied: brightness, contrast, Gamma, hue, and saturation modifications. For each individual type of modification, 10 different relative increasing/decreasing parameters of maximum 33% are considered. Secondly, 3 types of video editing operations are performed, namely insert logo (image size equals to 200x500 pixels, randomly placed in the frame), insert subtitle, and central zoom (by 10 values between 10% and 30%). Finally, two video encoding modifications are considered, namely CRF (Constant Rate Factor) and QP (Quantizing Parameter) changes. The former was applied 10 times, with parameters ranging between 20 and 40, while the latter by 10 values ranging from 8 to 35.

2. Experimental setup

The experiments are performed on in-premises servers, with Xeon E5 E5-1650 v3 @ 3.50GHz, 4 threads CPU, 32 GB of RAM and GeForce 1080Ti GPU.

The NN models are implemented in Python 3.9 using the TensorFlow v2.10.0 framework. For the backbone, we use ResNet50, ResNet101 and MobileNetV3small proposed by Keras [KER24] while ResNet18 implementation is available in [GIT24d].

The complete set of experiments is composed of 120 configurations: 2 databases (UVG, VID), 4 backbones (ResNet18, 50 and 101, MobileNetV3small), 3 fingerprinting sizes (32x32, 64x64, and 128x128) and 5 NN configurations.

ResNet models have been trained for 100 epochs, with a batch size of 64. The initial learning rate is set to 0.1 and kept unchanged during the first 10 epochs; then, an exponential relative decay of 0.15 each three epochs is considered. The training dataset is composed of 80% of the dataset. For each content in the training dataset, a Monte Carlo simulated version of the near duplicated modification is also considered. The validation is achieved by considering 20% of the database (without any Monte Carlo simulation). The same strategy is applied for MobileNet: in this case, if the results are not stable, 50 more epochs are added.

5 configurations are considered during evaluation: Baseline (Backbone and O3), End-to-End (I1, backbone, O1, O2, O3) and three intermediate models obtained by combining a subset of the elements presented in Figure 54, namely (1) B-O1 standing for the combination of the Backbone and the O1 dense layer, (2) B-O1-O2 standing for the combination of the Backbone and the O1 and O2 dense layers and (3) I-B-O1, standing for the combination of the I, Backbone and the O1 dense layer.

3. Result illustrations

The intermediate results are presented in Figure 55 – Figure 62 while a global view on the performances is provided by Table 6 and Table 7.

Experiment #1: Syntax element choice

Figure 55 displays the convergence of an End-to-End Resnet50 model trained with different combinations of input data on a 64x64 fingerprinting size for the VID dataset. It shows the training loss and the validation accuracy curves for three distinct data input variations.

The orange curve represents the model using only the luma residual elements (Y), achieving a maximum validation *Accuracy* of 0.55. The yellow curve, representing the model that utilizes both luma and chroma residual elements (Y, Cr, Cb), shows an improvement compared to the first model with the best *Accuracy* reaching 0.805.

The green curve indicates the model that includes all five extracted syntax elements (Y, Cr, Cb, IntraY, IntraC), demonstrating the quickest improvement in learning and the highest *Accuracy*, topping out at 0.897.

These results show that the combination of Luma and Chrome information, both as residuals and prediction modes is necessary when targeting convenient applicative performances.

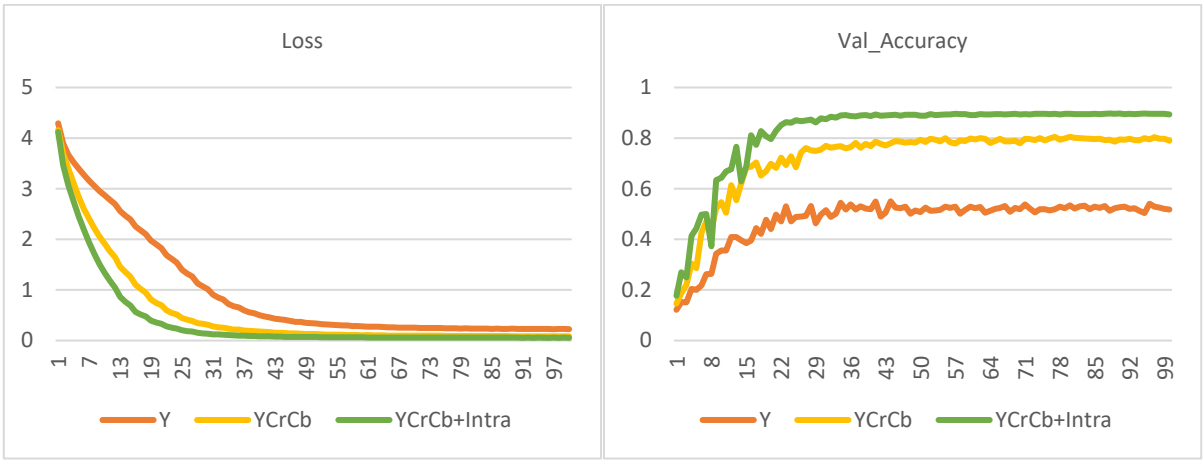


Figure 55: Syntax element selection Resnet50 (64x64)

The results presented in Figure 55 demonstrate that incorporating a larger set of syntax elements enhances the overall performance of the fingerprint matching. The model that integrates all five syntax elements outperforms the others, achieving the highest validation accuracy of 89.7%. So based on these results, the inclusion of all five syntax elements will be adopted for the following experiments. Note that results similar to those presented in Figure 55 are obtained for all the 4 investigated backbones.

Experiment #2: Model composition and convergency

The experimental results are presented in Figure 56 and Figure 57. These two figures are structured the same way. The abscissa corresponds to the number of training epochs, the left ordinate to the value of the training loss function while the right ordinate to the *Accuracy*, *Prec* and *Rec* values (between 0 and 1). Figure 56 and Figure 57 consider the largest fingerprinting size studied in our experiments (namely 128x128) and cover all the 5 variations for the DL model introduced in Section IV.3.B, namely Baseline, End-to-End, as well as the intermediate B-01, B-01-02 and I-B-01. Note that only the cases of Resnet18 and MobileNet as component of the Backbone are illustrated in these figures, yet similar results are obtained for the other backbone components.

Figure 56 and Figure 57 show the training loss and the validation accuracy, precision and recall for the VID and UVG databases, respectively. The visual analysis of the results thus presented brings forth that although the same convergence value tends to be reached, this process is faster and smoother for Resnet18.

When MobileNet is considered, an interesting behavior, contradicting a rule of thumb in DL applied to pixel domain, is identified: although the UVG dataset has less classes than VID, a backbone based on MobileNet provides worst classification performances for UVG. Two explanations can be provided. On the one hand, UVG classification case is easier, but

the training dataset is also smaller than in the VID case. On the other hand, this result may be linked to the difference in the very nature of the processed data, that are now stream syntax elements and no longer pixels.

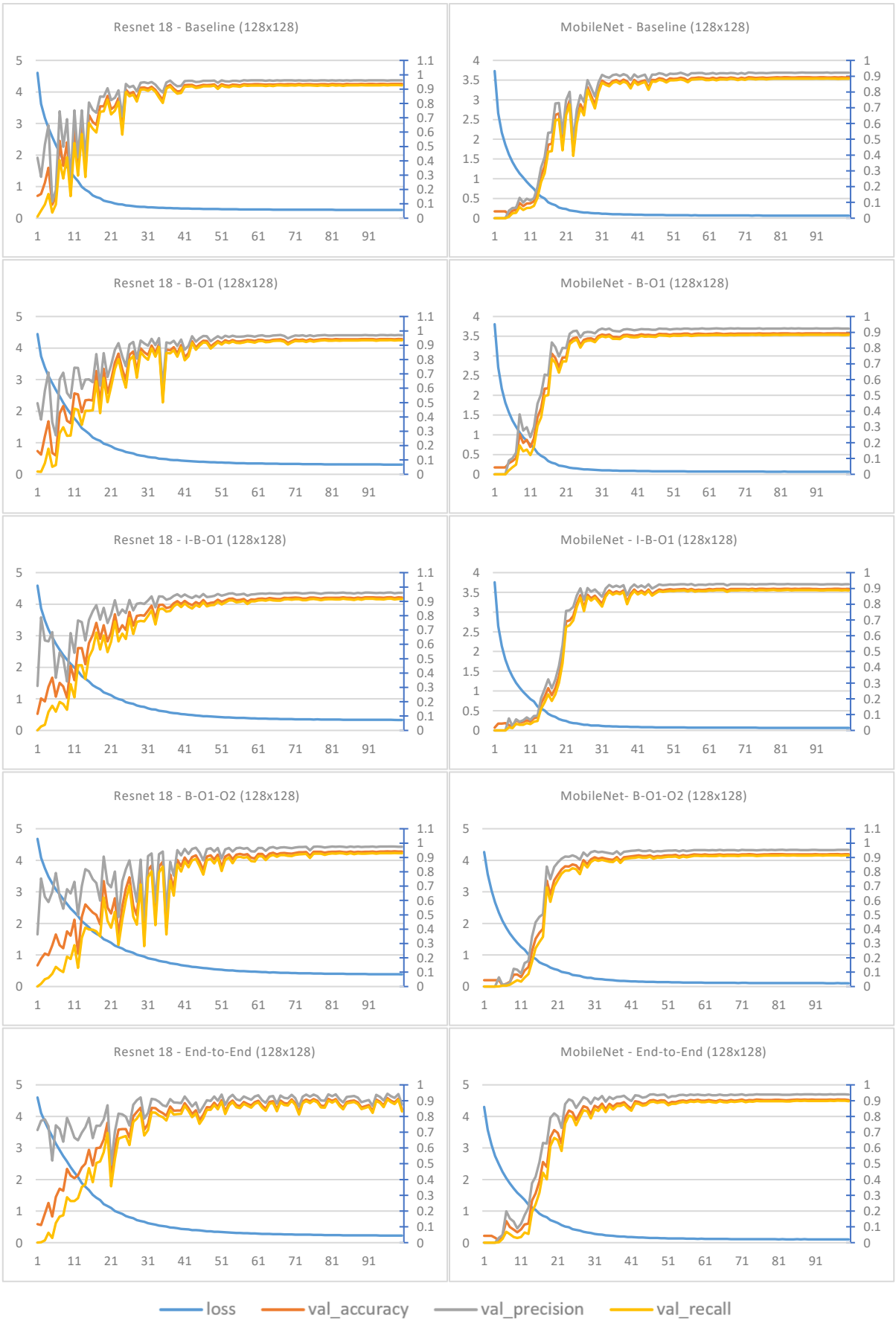


Figure 56: Experimental study on the DL solution components, VID dataset, 128x128 fingerprints

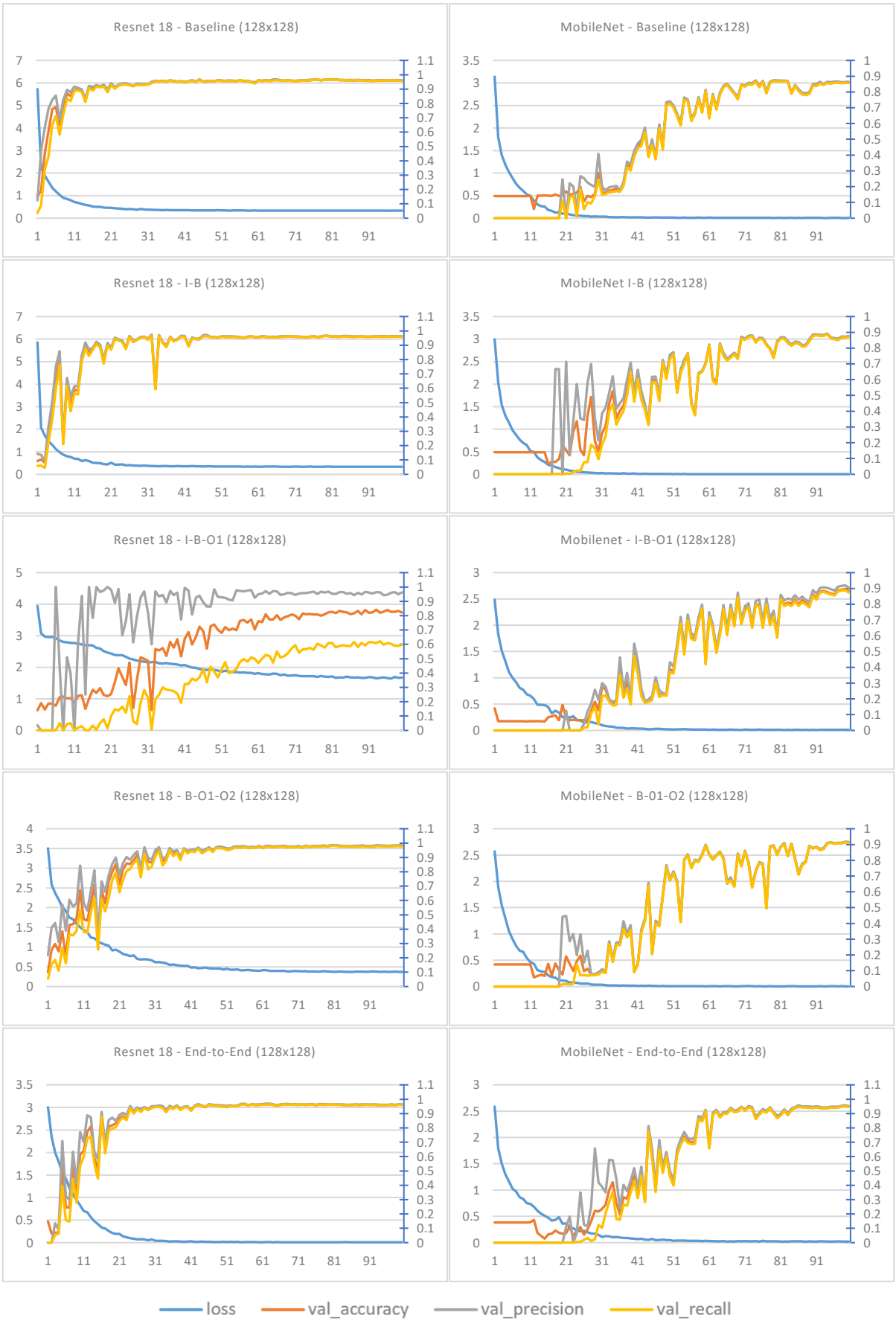


Figure 57: Experimental study on the DL solution components, UVG dataset, 128x128 fingerprints

Experiment #3: The impact of the fingerprint size

The impact of the fingerprint size in the quantitative results is illustrated in Figure 58 and Figure 59, that correspond to the VID and UVG databases, respectively.

These figures are organized the same way as Figure 56 and Figure 57 and consider Resnet18 as backbone, and two models, namely Baseline and End-to-End. Two fingerprint sizes are considered, namely 32x32, 64x64, as the cases of 128x128 fingerprints are already illustrated in Figure 56 and Figure 57.

The numerical values show that the less performant solution correspond to 32x32 fingerprints, thus reaching an *a priori* expectation: the more information about the video content we process, the better the performances.

However, the same expectation is not validated when comparing the results obtained in the cases of 64x64 and 128x128 fingerprints. Here again, two different explanations can be provided.

On the one hand, a limitation in database size might be invoked and considered that the DL classifier is not completely trained for 128x128 fingerprints; however, the shapes of the training curves do not support such an explanation.

On the other hand, a deeper analysis of the stream syntax elements show that they contain lot of uniform values (0, 1 or 2), and increasing the fingerprint size from 64x64 to 128x128 does not bring discriminant information for the classifier.

Experiment #4: Distribution of errors in fingerprint classification

Our analysis went deeper into detail and investigated the repartition of the errors over the video fingerprinting sequences. To this end, we illustrate the confusion matrices in Figure 60, Figure 61, and Figure 62.

Figure 60 and Figure 61 correspond to the VID database, End-to-End configuration, 32x32 fingerprints, and with two different backbones, namely Resnet18 and MobileNet, respectively. In these two figures, the video sequences to be fingerprinted are ordered according to their content creator (clients of VIDMIZER, that are anonymized in this study) and implicitly, to the type of content that is imposed the Creative Direction of that content creator. This ordering is represented by horizontal and vertical lines, creating some virtual super-classes, containing visually related content.

However, the visual inspection of Figure 60 and Figure 61 shows that although the distribution of the errors cannot be considered as uniform, it cannot be considered as related to the video content either. This demonstrates that the visual redundancy has been eliminated from the complex stream syntax elements (at least from conventional DL classifiers points of view) and that the fingerprinting matching is achieving by learning the very syntax elements values.

Figure 62 represents the same investigation, carried out on the UVG database. As this database contains by design visually unrelated contents, the idea of super-classes no

longer make sense. Yet, it can be again noticed that the error, although non uniformly distributed, do not show any clear clustering tendency.

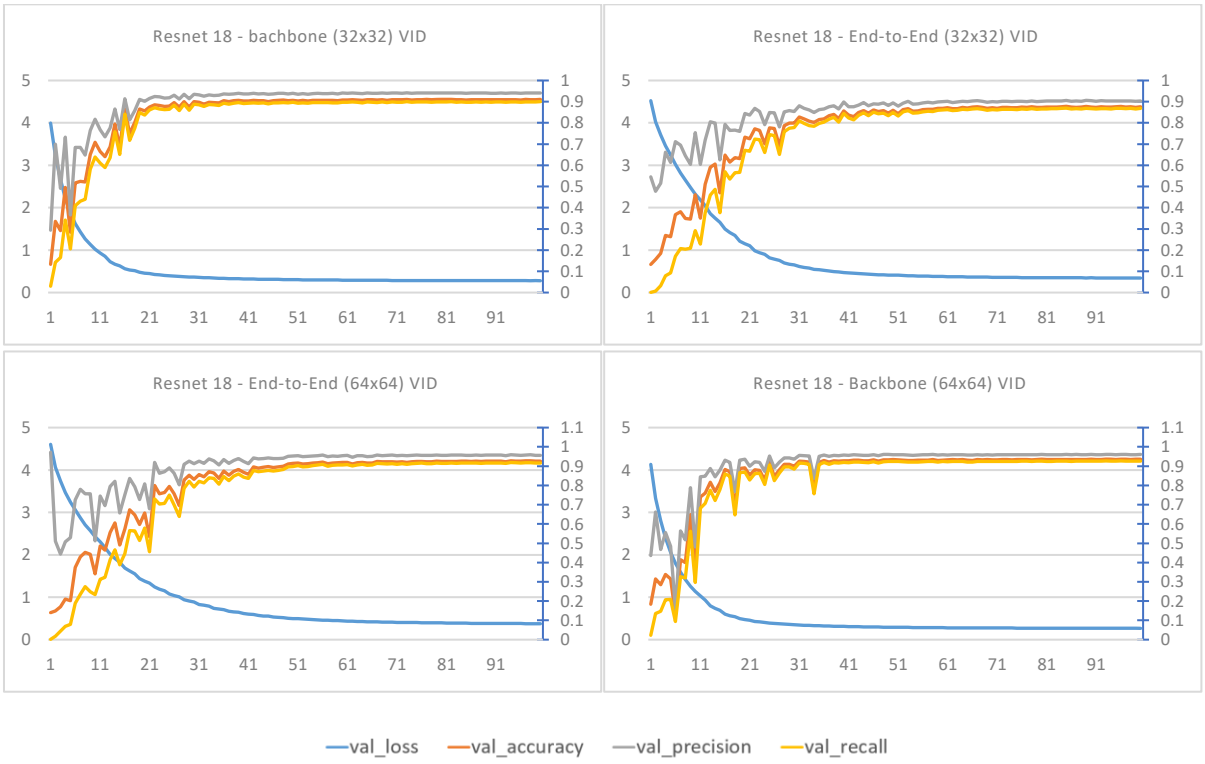


Figure 58: The impact of the fingerprint size in the performances, illustrated for the End-to-End (at the left) and Backbone (at the right), for the VID dataset

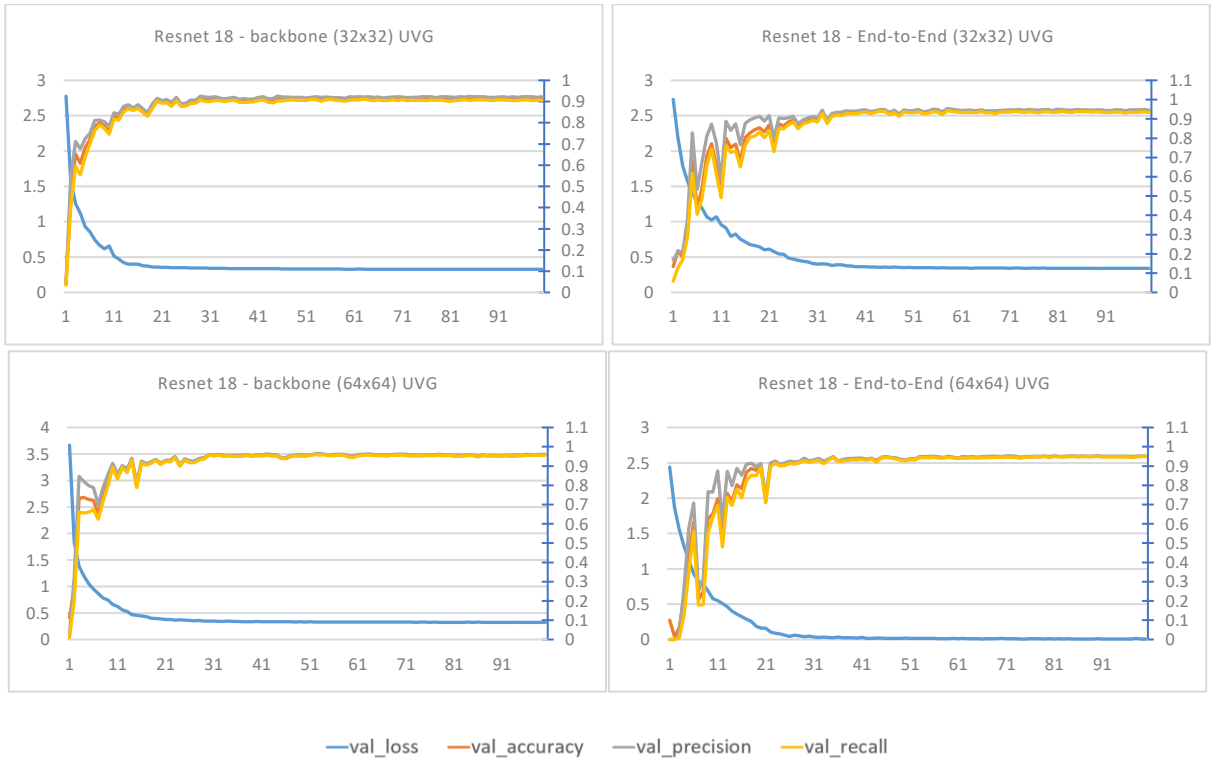


Figure 59: The impact of the fingerprint size in the performances, illustrated for the End-to-End (at the left) and Backbone (at the right), for the UVG dataset

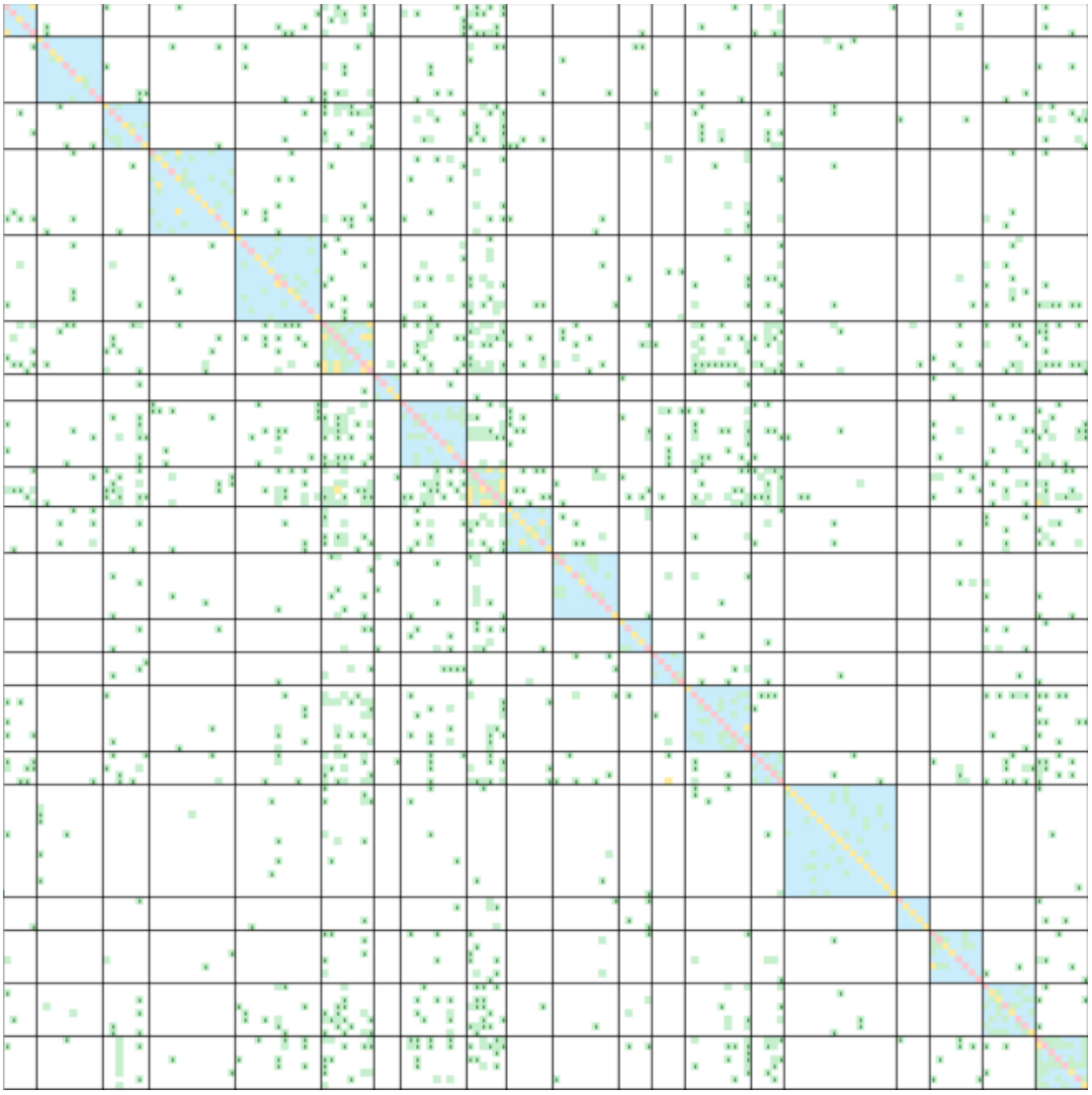


Figure 60: Confusion matrix, End-to-End model with Resnet18 as backbone, 32x32 fingerprint, VID dataset

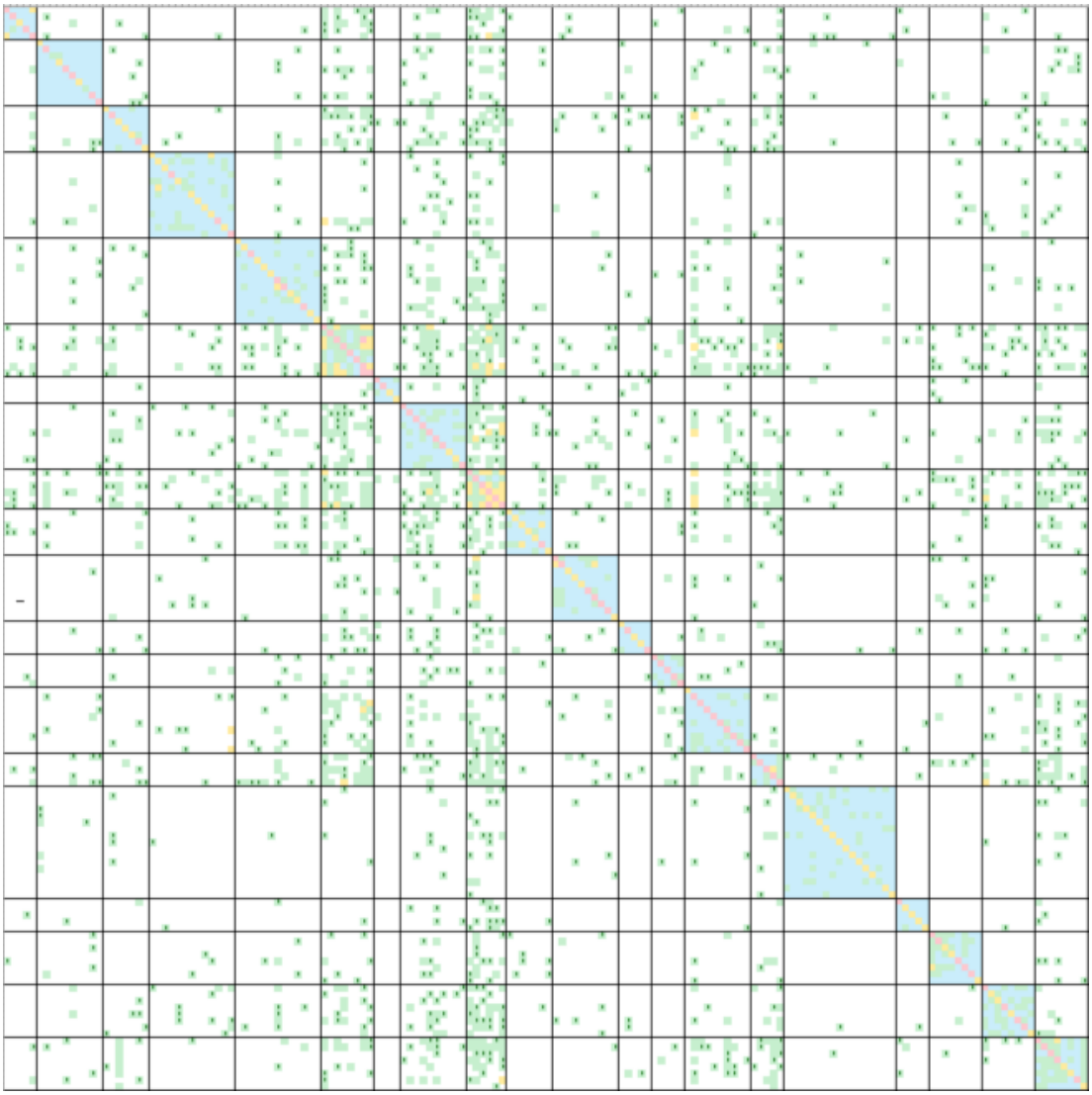


Figure 61: Confusion matrix, End-to-End model with MobileNet as backbone, 32x32 fingerprint, VID dataset

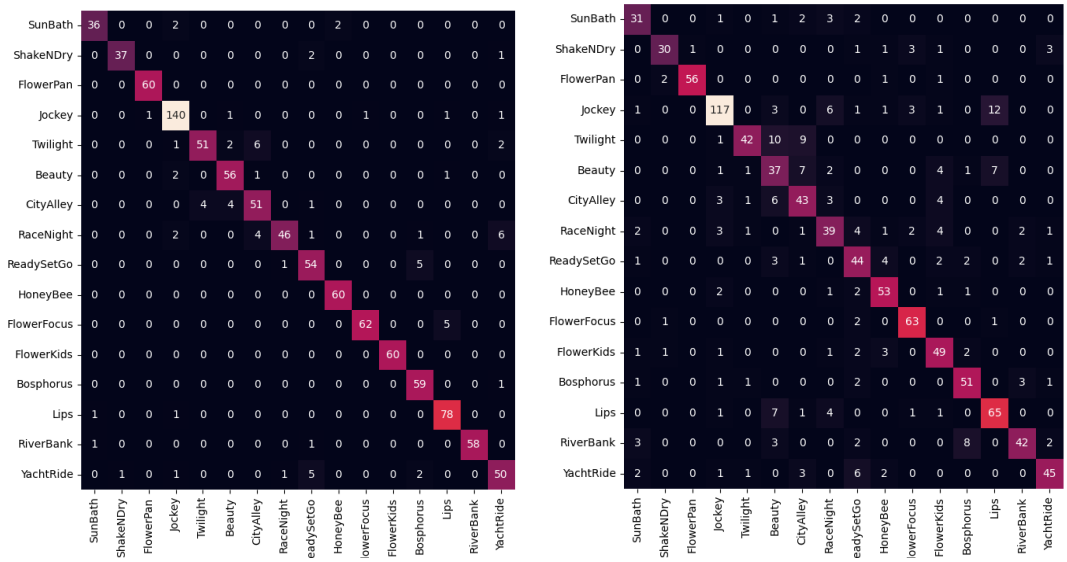


Figure 62: Confusion matrix, End-to-End model, 32x32 fingerprint, UVG dataset; Resnet18 as backbone (left) vs. MobileNet as backbone (right)

Experiment #5: Synoptic view on the quantitative results

After incrementally studying the DL compressed-domain fingerprinting throughout the previous 4 experiments, the complete set quantitative results will be synoptically presented now in Table 6 and Table 7.

Table 6 focusses on Resnet18 and MobileNet, and investigates the Accuracy (*Acc*), Precision (*Prec*) and Recall (*Rec*), by presenting the corresponding values multiplied by 100. The columns are grouped in three areas, according to the three investigated fingerprinting sizes. The rows are organized at three recursive levels: firstly, according to the two databases (UVG and VID), then to the backbone component (ResNet18 or MobileNet), and finally according to the 5 above-mentioned model configurations.

The values reported in Table 6 demonstrates the practical effectiveness of DL-based video fingerprinting by using only data extracted from the compressed domain.

When Resnet18 is included in the backbone, *Acc* values for the UVG database are higher than 0.9, irrespective to the model. Two exceptions are encountered, namely 64x64 fingerprints and B - I1 configuration (that is, when removing the O1 and O2 layers), and for 128x128 fingerprints and B - I1 - O2 configuration (that is, when removing the O1 layer). It can also be noticed that the *Prec* and *Rec* values are well balanced, with average relative differences lower than 3%. When considering the VID database, the same general trend is followed, yet the configurations resulting in *Acc* values lower than 0.9 are different. In its turn, when included as backbone, MobileNet results in *Acc* values larger than 0.8 for the UVG database while featuring some values as low as 0.567 in the case of VID database.

Table 6 also provides information about the usefulness of the I1, O1 and O2 layers added over the Baseline model. When considering the Resnet18 as backbone component and the UVG database, the baseline model is always outperformed by a configuration including at least one additional layer. This is not the case for the VID database where the baseline is the better solutions for 32x32 and 128x128 fingerprints, while being outperformed by the configuration including O1 and O2 in the case of 64x64 fingerprint. Also notice that the End-to-End configuration is never the better choice. When considering MobileNet as backbone, the conclusions change: the End-to-End is the best solution, with a singular exception: the 64x64 fingerprint and the UVG dataset, when it is outperformed by 0.5% by the B, I1, O2 configuration.

When comparing the results obtained on the VID dataset to the ones obtained on the UVG dataset, we would have expected better results on UVG, as it is *a priori* simpler, covering only 16 reference sequences. This result is confirmed for the End-to-End configuration but not for all the other four investigated configurations, as already pointed out in *Experiment #2*.

As a final remark related to the values in Table 6, we would have expected to notice a significant impact of the size of the fingerprint in the *Acc*: intuitively, the larger the size of the fingerprint the better the *Acc*. However, the results show that such a tendency is not always confirmed, as already pointed out in *Experiment #3*.

Table 6: *Accuracy* (Acc.), *Precision* (Pre.), and *Recall* (Rec.) according to the fingerprinting size, database, and model configuration; the numerical values are multiplied by 100.

			32x32			64x64			128x128		
			Acc.	***Prec.***	***Rec.***	***Acc.***	***Prec.***	***Rec.***	***Acc.***	***Prec.***	***Rec.***
UVG	ResNet18	Baseline (B)	91.2	92.2	90.6	95.9	96	95.7	96.1	96.2	95.9
		B, O1, O2	93	93.5	92.7	**98**	98.1	97.9	**98.2**	98.5	97.9
		B, I1	**95.7**	95.9	95.4	88.2	88.5	87.9	96.3	96.4	96.1
		B, I1, O2	92.8	93.1	92.4	96.6	96.8	96.4	83.9	94.5	60.8
		End-to-End	94.2	94.8	93.7	97	97	96.8	97.7	98.2	97.4
	MobileNet	Baseline (B)	81	81.5	80.2	87.1	87.1	86.8	86.4	86.7	86.2
		B, O1, O2	80	81.9	79.5	85.3	86.5	85.1	91.3	91.3	91.3
		B, I1	85.4	86.6	84.9	87	87.4	86.7	88.8	88.6	89.1
		B, I1, O2	85.4	86.6	84.9	**91.5**	92.2	91.2	89.9	92.5	89.1
		End-to-End	**87**	88.5	84.2	91	91.6	90.8	**95.2**	95.3	94.5
VID	ResNet18	Baseline (B)	**94**	96	93.1	94.6	96.3	89.6	**95.3**	97	94.6
		B, O1, O2	93.7	95.8	92.7	**95.8**	98.9	95	92.9	97.9	92.1
		B, I1	90.7	93.4	89.8	90.9	93.8	89.6	91.2	93.7	90.2
		B, I1, O2	88.3	91.5	87.4	94.3	96.5	93.4	92.7	95.9	91.8
		End-to-End	88.7	91.2	87.6	91.4	94.2	90.2	92.2	94.9	91.3
	MobileNet	Baseline (B)	59.4	66.8	55.2	88.1	90.8	86.9	90.3	92.2	89.3
		B, O1, O2	60.3	75.6	52.9	86.9	90.2	85.3	82.5	86.9	81.3
		B, I1	68.9	76.3	65.7	89.3	91.5	88.3	91.9	93.8	91.1
		B, I1, O2	56.3	71.1	48.9	90.9	92.9	90	91.9	94	91
		End-to-End	**86.6**	93.7	83	**91.8**	94.2	91	**94.2**	96.2	93.3

Table 7 complements the detailed information provided in Table 6 with a global information about the cases in which different components are considered in backbone, namely Resnet50 and Resnet101. Table 7 presents the *Acc*, *Prec* and *Rec* values multiplied by 100, corresponding to the End-to-End case. The results show that the global trend brought forth by the values in Table 6 is kept for the new backbone components. However, Table 7 also shows that the claim of Resnet architecture being robust against degradation [HE16] seems false in the compressed domain, as Resnet50 and Resnet101 are always outperform either by ResNet18 or by MobileNet. The impact of the fingerprinting size in the *Acc* value is now confirmed for the ResNet18 and MobileNet, while being contradicted by Resnet50 and Resnet101.

Table 7: *Accuracy* (Acc.), *Precision* (Pre.), and *Recall* (Rec.) according to the fingerprinting size, database, the End-to-End configurations and with different backbone components; the numerical values are multiplied by 100

		32x32			64x64			128x128		
		Acc.	***Prec.***	***Rec.***	***Acc.***	***Prec.***	***Rec.***	***Acc.***	***Prec.***	***Rec.***
UVG	Resnet18	**94.2**	94.8	93.7	**97**	97	96.8	**97.7**	98.2	97.4
	Resnet50	81.5	87.8	80.2	87.4	89	87.2	93.2	93.5	92.5
	Resnet101	87.1	88.8	86.6	91.8	92.6	91.7	85.4	88.1	84
	MobileNet	87	88.5	84.2	91	91.6	90.8	95.2	95.3	94.5
VID	Resnet18	**88.7**	91.2	87.6	91.4	94.2	90.2	92.2	94.9	91.3
	Resnet50	87.1	90.2	86	88.7	91.1	87.7	86.3	88.7	85.4
	Resnet101	74.5	80.1	72.4	88.3	90.7	87.5	84.6	87.8	83.3
	MobileNet	86.6	93.7	83	**91.8**	94.2	91	**94.2**	96.2	93.3

IV.4. COMMON at work

To the best of our knowledge, Section IV.3 COMMON introduced and discussed the first DL-based solution for video fingerprint based on stream syntax elements. The experimental results presented in Table 6 and Table 7 demonstrate its effectiveness for both industry relevant and scientific databases.

We shall go one step further toward the COMMON practical usage, by investigating three applicative issues, namely the robustness against video reencoding in Section IV.4.A, the possibility of extending on-the-fly the database in Section IV.4.B, and the possibility of applying day-by-day modifications on the trained model without affecting its performances in Section IV.4.C.

IV.4.A. Video encoder dependency

The main question raised by any compressed domain processing solution relates to the dependency of the results with respect to the codec choice. In other words, while the results presented in the previous section relate to the HEVC video codec, what happens if the video sequence is encoded in MPEG-4 AVC or in VVC?

Of course, all stream syntax elements intrinsically depend on the coding technology and trying the extract the fingerprinting directly from new type of stream would be both a conceptual and technological nonsense.

Hence, changing the video encoding format will be considered as an additional type of attack. The COMMON robustness against it is evaluated after a reencoding the attacked sequences back into the HEVC format and the results are presented in Table 8, for the VID database, and an End-to-End configuration backboned by Resnet18 and 64x64 fingerprints.

Table 7 shows that the initial (prior to reencoding in MPEG-4 AVC / VVC) performances were of 91.4, 94.2, and 90.2 in terms of *Accuracy*, *Precision*, and *Recall*, respectively.

Hence, Table 8 shows that changing the video codec to MPEG-4 AVC does not result in decrease in the performances, the *Accuracy* being even increase by 0.06. Yet, variations in performances are encountered for the VVC encoder, that results in values of 0.84, 0.87 and 0.82 in terms of *Accuracy*, *Precision*, and *Recall*, respectively.

After this experimental validation, our study went one step further and also considered two video codecs developed outside ISO and ITU communities, namely vp9 [MUK13] and av1 [CHE18]. For these two cases, the applicative performances are more severely impacted, with the *Accuracy* values going down to 0.67 and 0.77, respectively. Yet, note that in this situation also corresponds to severe modifications in the quality of the original content, subjected to two successive lossy encoding operations.

Table 8: *Accuracy*, *Precision*, and *Recall*, after encoding HEVC data test in AVC (first line) and VVC (second line); the detection is preceded by an HEVC reencoding.

	Accuracy	Precision	Recall
HEVC->**AVC**->HEVC	0.92	0.95	0.91
HEVC->**VVC**->HEVC	0.84	0.87	0.82
HEVC->**vp9**->HEVC	0.67	0.72	0.65
HEVC->**av1**->HEVC	0.77	0.80	0.76

IV.4.B. Database extensibility

The behavior of the COMMON method when the database is expected to be dynamically updated is now to be studied.

Experiment #1: Training from scratch with a larger classifier

The naïve approach would be to train from scratch, either by considering a unique model with a larger number of classes or by splitting the dataset into sub-sets of a fixed number of elements and creating different models for each subset.

For instance, consider the VID database and the case in which the database is expended by about 10% (*e.g.* 16 additional video sequences are supposed to be added to the database). We consider the best configuration identified by the experiments in the Section IV.3.C, namely Resnet18, B-O1-O2-O3 and 64x64 fingerprinting. As the number of classes is now enlarged, we must change the O3 accordingly (this is the layer of classification, and it should have the same size as the number of classes). The results obtained when retraining the model are illustrated in Figure 63 (that is organized the same way as Figure 56): *Accuracy* = 0.93, *Precision* = 0.96 and *Recall* = 0.92 are thus obtained. When comparing these values to the results presented in Table 6, we can state the COMMON methodology is stable with respect to small variations (about 10%) in class number.

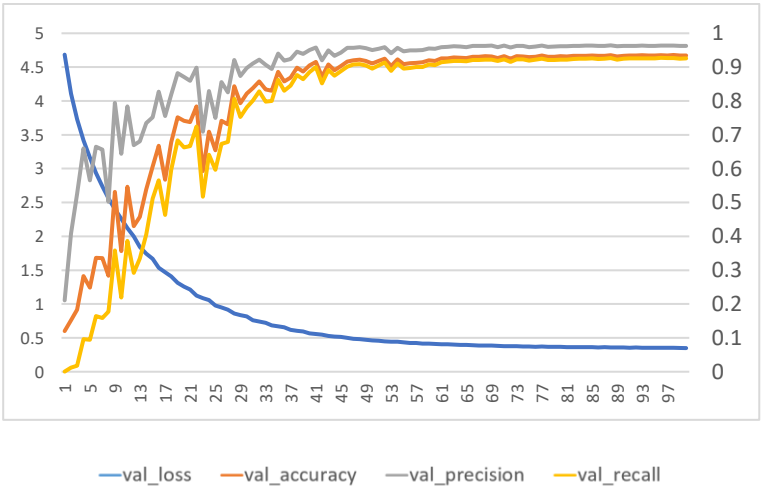


Figure 63: Increasing the database by 10%: illustrations for VID database, Resnet18, B-O1-O2-O3, 64x64 fingerprints

Of course, as such an approach requires a consequent training time, and is not suitable for adding on-the-fly new sequences to the database. Hence, we studied the possibility of extending the number of classes with a minimal extra training effort. Consequently, fine tuning solutions are looked for in the next experiments.

Experiment #2: Fine-tuning

Be there the VID dataset, and we consider as basis the best configuration brought forth Table 6, namely Resnet18, B-O1-O2-O3 and 64x64 fingerprints, resulting in *Accuracy* = 0.93, *Precision* = 0.96, and *Recall* = 0.9302.

We load the pre-trained COMMON model, and we create a new dense layer to replace O3. The new dense have the same weights and bias for the first *N* outputs (already existing in the initial dataset) and initializes the *M* new ones ($M = N/10$) with random values.

On this basis, three fine-tuning strategies are followed, as illustrated in Figure 64 and detailed in Experiments 2.a, 2.b and 2.c here after. The experimental results are presented in Table 9.

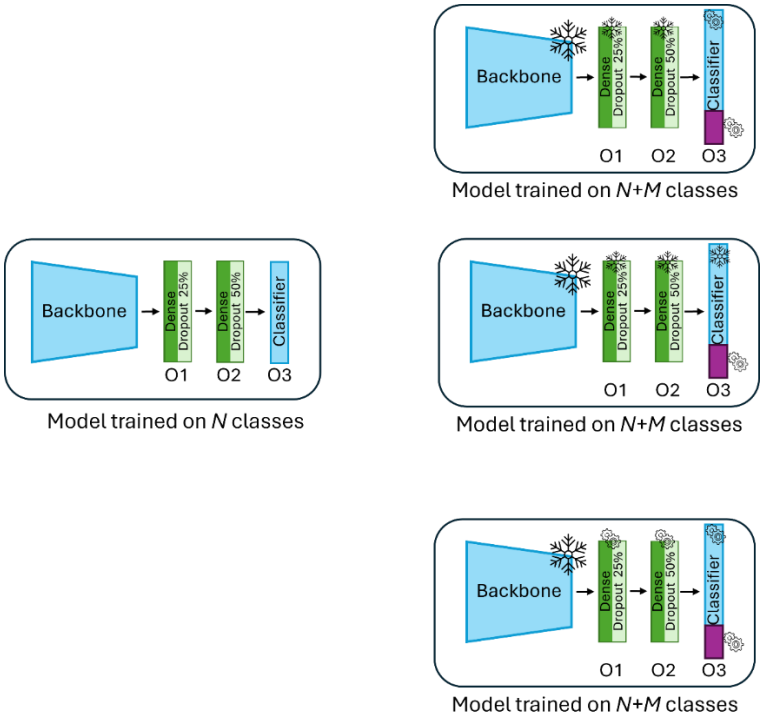


Figure 64: Increasing the database by 10%: fine-tuning strategies detailed in Experiments 2.a, 2.b, and 2.c. The elements frozen or trained are represented by the snowflake and processing symbols, respectively.

Experiment #2.a: Fine-tuning the complete O3 layer

In this experiment, the backbone and the O1 and O2 layers are frozen, while the O3 layer is completely fined tuned.

The model is trained for extra 20 epochs, alternatively on all (new and old) or on new data.

A learning rate equal to 1/10 of the learning rate used to train the initial model is considered when processing the global dataset; when processing only the new dataset, the learning rate equals 1/100 of the initial learning rate.

The validation is alternatively done, this time on the old or on the new data.

The experimental results corresponding to the 4 investigated configuration (2 possible training datasets and 2 possible testing datasets) are presented in the first three lines in Table 9.

Experiment #2.b: Fine-tuning the new elements in the O3 layer

In this experiment, the backbone, the O1 and O2 layers, as well as the elements (weights and biases) corresponding to the initial *N* classes in O3 are frozen, while *M* new elements (weights and biases) in the O3 layer are fined tuned.

The same learning and testing strategy as in *Experiment #2.a* are considered.

The experimental results are presented in the lines 4,5 and 6 in Table 9.

Experiment #2.c: Fine-tuning the three output layers O1, O2 and O3

In this experiment, all the three output layers O1, O2 and O3 are finetuned, while the backbone is frozen.

The same learning and testing strategy as in *Experiment #2.a* are considered.

The experimental results are presented in the lines 7, 8 and 9 in Table 9.

Table 9: Increasing the database by 10%: fine-tuning for VID datasets, Resnet18, B-O1-O2-O3 and 64x64 fingerprints. The fine tuning strategies are illustrated in Figure 64.

	Metric	***All the data training***		***New data only training***	
		Old Dataset	New Classes	Old Dataset	New Classes
Experiment #2.a *finetune*: O3	*Accuracy*	0.93	0.64	0.51	0.86
	Precision	0.96	0.71	0.55	0.91
	Recall	0.91	0.55	0.49	0.80
Experiment #2.b *finetune*: new weights in O3	*Accuracy*	0.92	0.60	0.91	0.42
	Precision	0.96	0.64	0.94	0.46
	Recall	0.91	0.50	0.90	0.38
Experiment #2.c *Finetune:* O1+O2+O3	*Accuracy*	0.93	0.88	0.56	0.57
	Precision	0.96	0.92	0.71	0.83
	Recall	0.92	0.85	0.48	0.31

Table 9 shows the performances and the limitations when trying to extend the video fingerprinting database without a complete learning process, as the one illustrated in Figure 64, for instance.

It is thus established that these three finetuning strategies are complementary.

The first finetuning strategy (*cf*. *Experiment 2.a*), in line with current-day finetuning approaches, cannot serve our purposes: either the model still recognizes the old classes but does not succeed in learning the new ones or, on the contrarily, learns the new ones while forgetting the old ones.

The second finetuning strategy (*cf*. *Experiment 2.b*) is designed in this study and preserves the performances on the old data classes (even when trained only on the new data), yet remains limited on the performances on the new classes.

The third finetuning strategy (*cf*. *Experiment 2.c*) is also designed in this study and seems the most performant one: when applied on all dataset, it preserves the performances on the old data classes while learning quite convenient the new ones: for the new *M* classes, *Accuracy*, *Precision* and *Recall* values of 0.88, 0.92, and 0.85 are obtained, respectively.

However, note that compressed-domain finetuning is more restrictive than state of the art finetuning: our application requires both the new and the old datasets to be processed during finetuning. Yet, we can still speak about finetuning, as the most complex training operation (the DL classifier) is frozen.

IV.4.C. Fingerprinting model stability

The last years testified not only a raise in the usage of DL solutions but, at the same, time a trend towards various post-training modifications made over DL models. For instance, model pruning and parameter quantization are today operations likely to occur during the usage of any trained model. Consequently, we shall study now how COMMON behaves when the trained fingerprinting models are subjected to such modifications.

Pruning involves identifying and removing redundant components (*e.g.*, weights, neurons, or layers) from a neural network. By redundant component it is understood component whose removal has little to no significant impact on the model's performance [TEN24].

Pruning has as objective to get to smaller, faster, and less memory-intensive models, efficient and deployable even in resource-constrained environments.

The same objective is also target by the model quantizing, defined as process of converting the model weights representation to reduced precision data types as 16 bit floats (suitable for GPU acceleration) or 8 bit integers (suitable for CPU execution) [GHO21, TEN24].

In our experiments, a weighted, magnitude-base pruning is applied, on a layer-based approach. Two thresholds are considered, namely 50% and 80%. The quantizing is applied in one of the extreme cases, namely post-pruning weights dynamic reduction to int8.

The experimental results are presented in Table 10 and Table 11.

Table 10 investigates the pruning and quantization effects on the *Accuracy* of DL-based compressed domain fingerprinting, on VID database and 64x64 fingerprints. The four backbone candidates are successively considered, namely MobileNet, Resnet18, Resnet50 and Resnet101. The values reported in Table 10 show that Resnet18 is the only backbone

featuring a quite stable behavior against pruning and quantizing, with minor decreases in *Accuracy*. On the contrary, when considering MobileNet, Resnet50 or Resnet101 as backbones, the *Accuracy* performances are destroyed.

Table 10: Pruning and quantization effects on the *Accuracy* of DL-based compressed domain fingerprinting, on VID database and 64x64 fingerprints.

	COMMON	Pruned (50%)	Pruned (50) & Quantized (int8)	Pruned (80%)	Pruned (80) & Quantized (int8)
MobileNet	0.85	0.79	0.78	0.57	0.57
Resnet18	0.93	0.93	0.90	0.90	0.89
Resnet50	0.87	0.83	0.82	0.22	0.21
Resnet101	0.83	0.82	0.82	0.33	0.31

Table 11 investigates the pruning and quantization effects on the pruning and quantization effects on the size (in MB) of the DL-based compressed domain fingerprinting model, when save on a memory support. The experimental configuration considers the VID database, 64x64 fingerprints, and TensorFlow. The values reported in Table 11 show an overall gain in storage footprint by a factor of 4, irrespective of the model considered in the backbone and of the pruning parameter. This factor 4, that can be explained by a quantizing from 32 bits floating point values 8 bit integers, rather shows a limitation of the DL frameworks in data management, than a result related to COMMON.

Table 11: Pruning and quantization effects on the size (in MB) of the DL-based compressed domain fingerprinting model when save on a memory support (VID database, 64x64 fingerprints, TensorFlow).

	COMMON	Pruned (50%) & Quantized (int8)	Pruned (80%) & Quantized (int8)
MobileNet	24.08	6.12	6.12
Resnet18	47.45	11.90	11.90
Resnet50	156.43	39.50	39.50
Resnet101	229.47	58.14	58.14

IV.5. Summary

The present Section presents a first of its kind, comprehensive study on the possibility of fingerprinting video content based on compressed stream syntax elements. The illustrations correspond to the HEVC standard while MPEG-4 AVC and VVC stand as validation criteria.

The study is two folded and encompasses an ML-based proof of concepts and a DL-based functional solution.

While the advantages and limitations of each of these two approaches are discussed in respective sections, we would like to emphasize here one additional feature of such an approach.

Conventional fingerprint methods usually take place in 2 times: offline phase or initialization phase that consist of create the reference fingerprinting database, and an online phase where the system is running, and a new query video retrieval process is initiated [GAR16]. For the model well-functioning, the reference database should be easily and effectively accessible which can present a challenge as the system get bigger.

For instance, the complete VID dataset (original and near-duplicated content) sums up to 135 GB, while the underlying fingerprint sizes are presented in Table 12.

Table 12: Fingerprint and storage management

Fingerprint size	Disk size (GB)	Compressed size/zip file (GB)
32x32	4.95	0.49
64x64	18.43	1.56
128x128	72.33	5.01

For NN based matching algorithms, the reference database is not required while deploying the system. The only required data is the COMMON model weights which are generally more compact than the entire database. For instance, the End-to-End COMMON model backboned by Resnet18 accounts, prior to pruning and quantizing, 47MB and this size can be reduced by a factor 4 by pruning and quantizing, *cf*. Table 11.

It can thus be ascertained that the DL-based solution we advance reduces the storage footprint requirements by factors between 10 (when considering 32x32 fingerprints and in absence of pruning and quantizing) and 400 (when considering 128x128 fingerprints and in presence of pruning and quantizing).

This experimental result demonstrates that the coupling of compressed-domain fingerprint to DL not only increases the processing speed but also reduces the storage requirements. Note that this benefic property was obtained as the structure we designed (*cf*. Figure 54) considers conventional DL classifiers and the need for complex structures (*e.g.* transformers) has not been identified.

Chapter V. Blockchain-fingerprinting applicative synergies

This section presents the mutual benefits obtained when coupling blockchain and fingerprinting solutions.

V.1. BIDDING presentation

While the previous *Chapter III On-chain / off-chain processing* and *Chapter IV Compressed domain video fingerprinting* delt with the possibility of virtually extending the computation and storage of blockchains as well as achieving compressed-domain video fingerprinting, the present Chapter establishes functional synergies between them.

The solution advanced in the thesis is represented by an end-to-end processing pipeline, further referred to as **BIDDING** – *BlockchaIn-baseD viDeo fINgerprintinG*.

BIDDING is conceptually illustrated in Figure 65, while its functioning is illustrated as a sequence diagram in Figure 66 and detailed here-after.

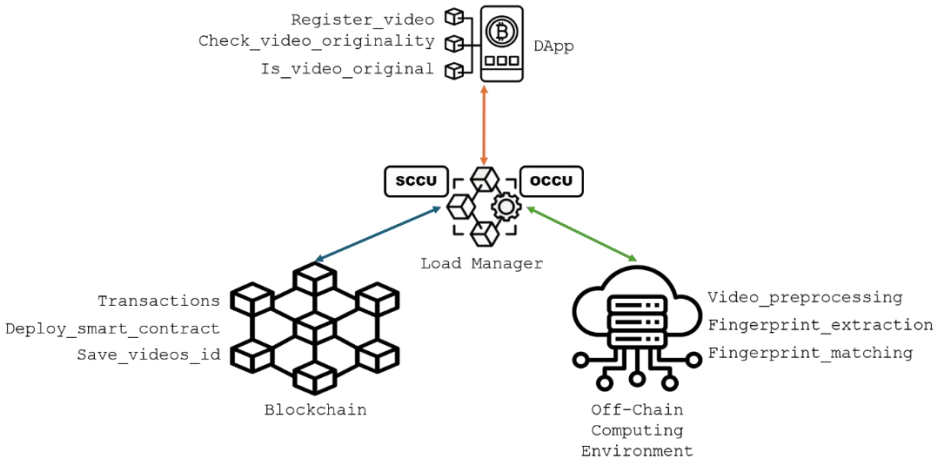


Figure 65: BIDDING workflow distribution

According to in Figure 65, when the user wants to check the uniqueness of a digital document, the fingerprint computation is launched through the graphic interface. Then, a transaction to the Smart Contract address precising the entry point and passing the fingerprint as an argument is made. As soon as the Smart Contract is triggered, it blocks the access of other users to avoid conflicts and avoid any potential security issues. When the block containing that transaction is processed and validated (*e.g.*, by the miner or baker), the user receives the transaction receipt containing its details and the connector receives a notification event that is subscribed to. The connector fetches the user's request and forwards it to the Python script where the document fingerprint is compared with all previous fingerprints processed by the system, and then returns the matching process results to the connector. At this stage, the connector uses a moderator account to make a transaction to the Smart Contract with the results. The Smart Contract authorizes the moderator transaction, allowing the user to access again and notify the user *via* an Event.

The sequence diagram presented by Figure 66 illustrates the interaction flow between a DApp, a blockchain, a `Load Manager` module as presented in Section III.1 and a COMMON model as presented in Section IV.3. It is one of the use cases for video authenticity application / decentralized video notary system where a user can upload a

video, check whether it presents an original work or not. Here's a breakdown of the steps depicted in the diagram:

1. `regiter_video`: the DApp initiates a request to the `Load Manager` to register a new video in the blockchain providing its name and URL.
2. `generate_fingerprint`: the off-chain environment extracts the fingerprint from the compressed domain generation (N, 64, 64, 5) matrix with N equal to the number of I frames in the video sequence and return the result to the `Load Manager`. Note that the blockchain implementation requires the number of I frames to be limited at 10.
3. `register_video`: before registering the video in the blockchain, the `Load Manager` transforms the fingerprint into Base64 words and apply a lossless compression to further optimize the size of the fingerprint yet being reversible. The `register_video` Smart Contract function takes as input the compressed video fingerprint, the video name and URL. Before validating the transaction, the blockchain runs a preliminary verification that the URL is not already present in this database.
4. `is_video_original`: the DApp requests a rigorous check about the video authenticity by providing the fingerprint.
5. `check_video_exist`: the `Load Manager` validates that the fingerprint provided correspond to the list of known videos and whether the DApp is the owner of that video or not as well as if it was already flagged as original content.
6. `match_video`: the off-chain COMMON module receives the fingerprint and runs the matching DL model to identify potential near-duplicates.
7. `Update_originality`: is case of no matches to the video among the known sequences for the model, the `Load Manager` transacts the information to the blockchain and inform the DApp of the results.

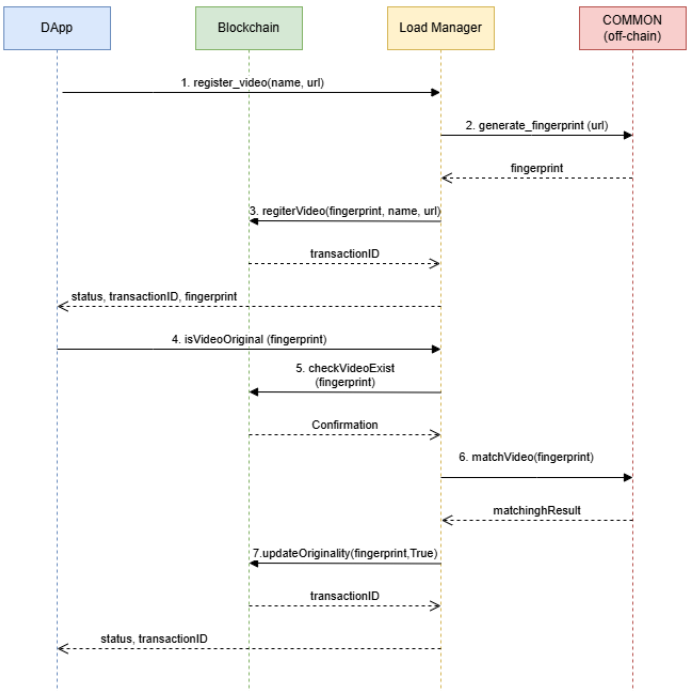


Figure 66: BIDDING execution sequence diagram

V.2. BIDDING deployment

The BIDDING workflow is alternatively deployed on types of setups, namely server oriented or Raspberry Pi oriented, as illustrated in Figure 67. In the two cases, TensorFlow frameworks are considered, namely TensorFlow v2.10.0 for server-oriented setups and tflite-runtime v2.14.0 for embedded setups.

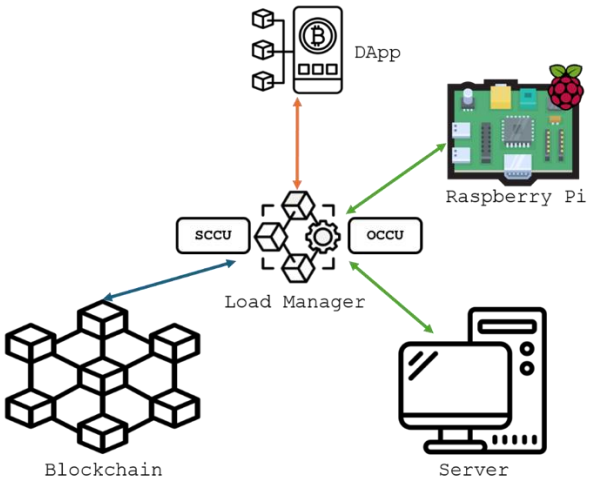


Figure 67: Two alternative BIDDING deployment set-ups: the off-line model might be based on a server or even a Raspberry Pi.

Initialization

If the node was turned down or run for the first time, it's initialization can take long time so the node gets synchronized with rest of the networks, as illustrated in Figure 68 and Figure 69 for Tezos and Ethereum blockchains, respectively.

We emphasize that the deployment is successful even when considering emended computing infrastructure (Raspberry Pi), as illustrated in Figure 70 for the case of Tezos, and in Figure 71 and Figure 72 for the case of Ethereum.

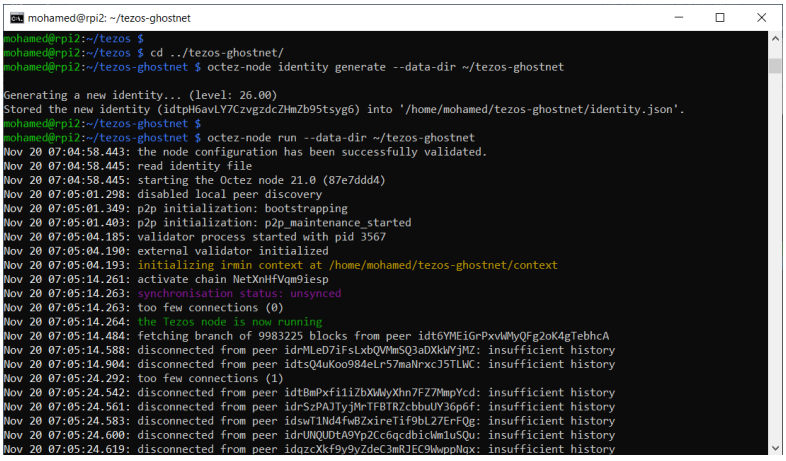


Figure 68: Tezos blockchain initialization and synchronization phase; the node is running in a testnet

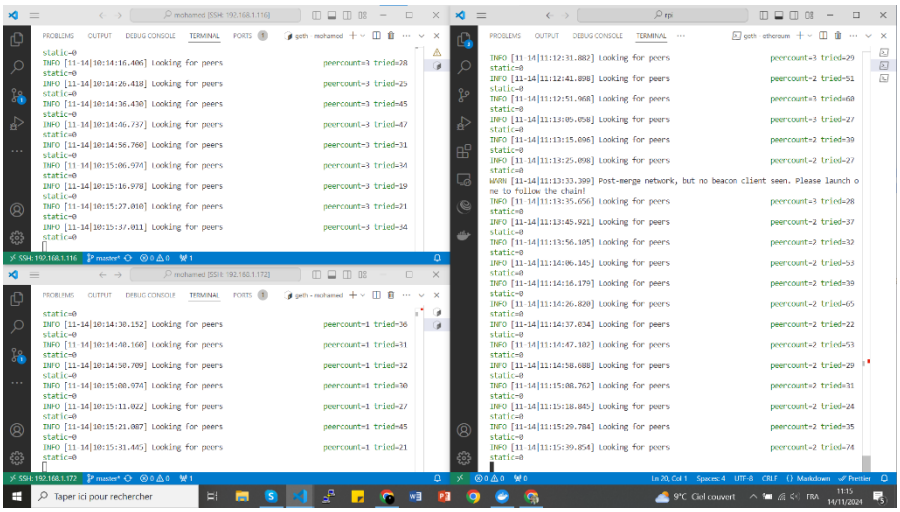


Figure 69: Ethereum blockchain initialization and synchronization phase; the node is running in a private network

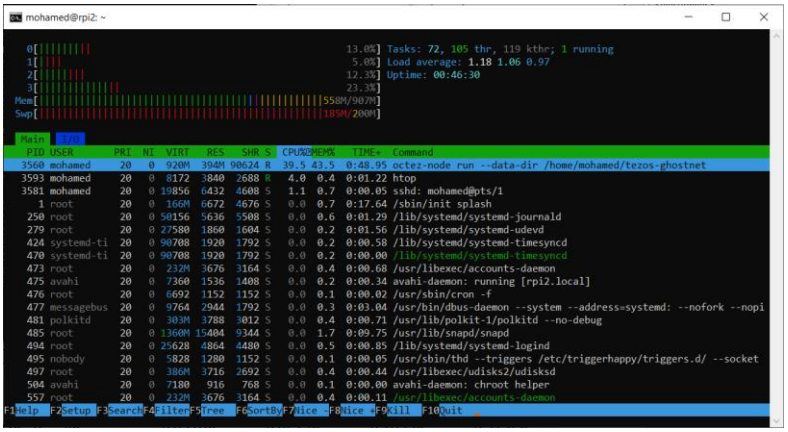


Figure 70: Raspberry Pi resources consumption during initialization of Tezos testnet

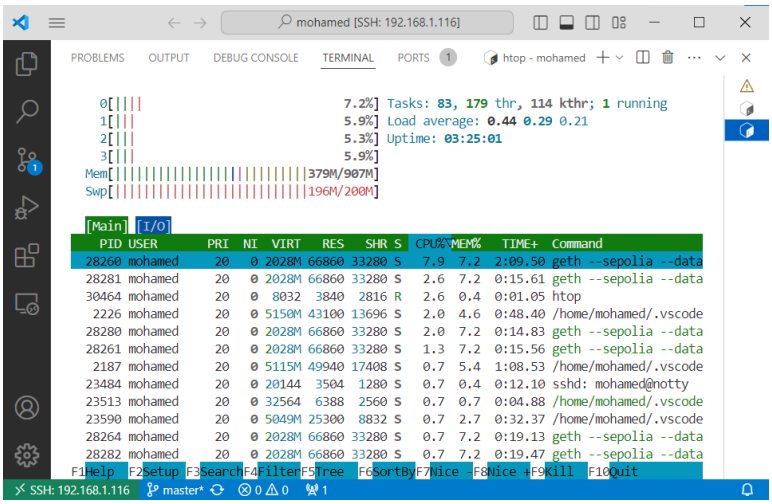


Figure 71: Raspberry Pi resource utilization running Ethereum node

```
// Register a video
function registerVideo(
    string memory fingerprint,
    string memory name,
    string memory url
) public {
    require(bytes(urlToFingerprint[url]).length == 0, "URL already registered");
    require(videos[fingerprint].timestamp == 0, "Video already registered");

    videos[fingerprint] = Video({
        fingerprint: fingerprint,
        owner: msg.sender,
        name: name,
        url: url,
        original: false, // Default to false
        timestamp: block.timestamp
    });

    // Map URL to fingerprint
    urlToFingerprint[url] = fingerprint;

    emit VideoRegistered(fingerprint, msg.sender);
}
```

Figure 72: Register Video Solidity implementation

Deployment

Once the solution initialized, the Smart Contracts are compiled and deployed, as illustrated in Figure 73 and Figure 74 for the Ethereum case.

Figure 74 also provides detail information about the fees (in ETH) corresponding to the deployment (represented into a red box), to registering a new video in the database (represented into a green box), and to checking the existence of a query video in the database (represented into a blue box). When comparing the values corresponding to registering a new video in the database (0.000954 ETH) to the value corresponding to checking the existence of a query video (0.000039 ETH), a factor of about 25 is encountered. Such a value validates our approach: although blockchain fees related to massive data related to video processing are significantly higher (by a factor of 25) than conventional transactions, thanks to BIDDING, they are not prohibitive in practice.

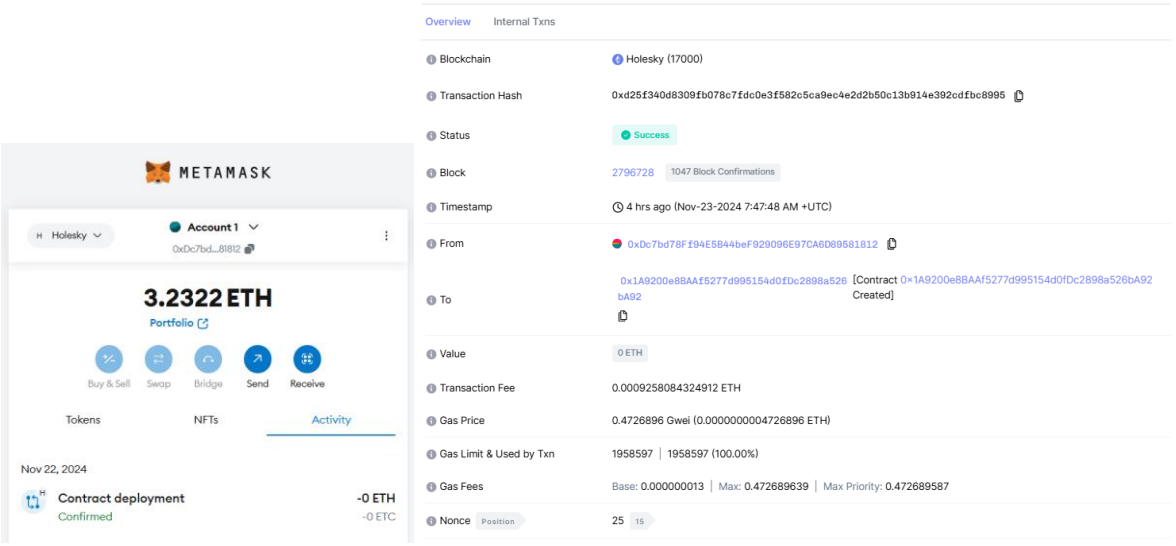


Figure 73: Ethereum Smart Contract deployment

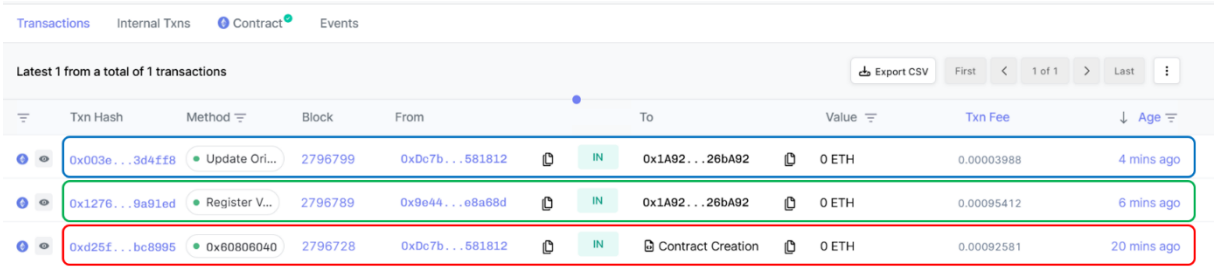


Figure 74: Snapshot of the Smart Contract history capturing transaction hash (Txn Hash), method invoked, block number, transaction initiator (From), recipient (To), Ether value transferred (Value), transaction fee (Txn fee)

BIDDING has also been deployed on Tezos. While the same general trend is followed, the ratio between the fees corresponding to conventional and to video processing operations is larger. Actually, when comparing the values corresponding to registering a new video in the database (0.467 ꜩ composed as follow baker fee 0.00096 ꜩ and storage fee 0.467 ꜩ) to the value corresponding to checking the existence of a query video (0.000918 ꜩ that present only the baker fee since no data has been added), a ratio of about 500 is encountered.

Execution

To forbid the access to edit the originality flag, only an authorized group of users is allowed to execute it which can be edited by the Smart Contract creator of the Smart Contract, as illustrated in Figure 75 (code) and Figure 76 (execution screenshot) for the Ethereum blockchain.

```
constructor() {
    admin = msg.sender; // Initialize contract deployer as admin
    superUsers[admin] = true; // Admin is also a super user by default
}

// Admin function to add a super user
function addSuperUser(address user) public onlyAdmin {
    require(!superUsers[user], "User is already a super user");
    superUsers[user] = true;
    emit SuperUserAdded(user);
}

// Admin function to remove a super user
function removeSuperUser(address user) public onlyAdmin {
    require(superUsers[user], "User is not a super user");
    superUsers[user] = false;
    emit SuperUserRemoved(user);
}
```

Figure 75: Smart Contract constructor and functions to edit the list of authorized addresses

Figure 76: Authorized addresses only can update the originality

For this application the communication between the DApp and the Load Manager is ensured thanks to a REST API developed using a python framework (FastAPI), as illustrated in Figure 77.

When evaluating the fingerprinting matching time on the off-chain side (common for both Ethereum and Tezos implementations), average values of 0.06 sec. and 0.8 sec. are recorded for the server-based and embedded-based setups, respectively, as illustrated in Figure 78 and Figure 79.

Note that while in server-based setup a small variability of the results is encountered (with maximal variations lower than 0.0008 sec.); yet, in the case of the embedded-based setup, the processing time is constant over all the video queries. Rather than being connected to our methodological setup, this invariance shows that current-day tflite-runtime v2.14.0 behavior is not really matched to the hardware resources, thus hiding the very computational specificities of the tasks.

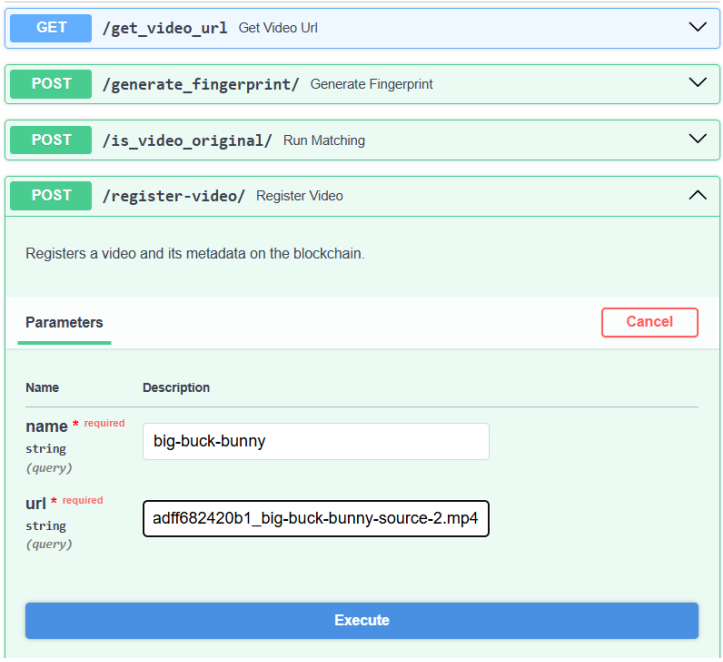


Figure 77: List of the available endpoints available to the DApp

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
0it [00:00, ?it/s]
Evaluated on 0 results so far.
1it [00:08,  8.20s/it]


2it [00:16,  8.15s/it]


3it [00:24,  8.13s/it]


4it [00:32,  8.18s/it]
```

Figure 78: Off-chain fingerprinting processing time for server setups; each iteration corresponds to a batch of 128 fingerprints

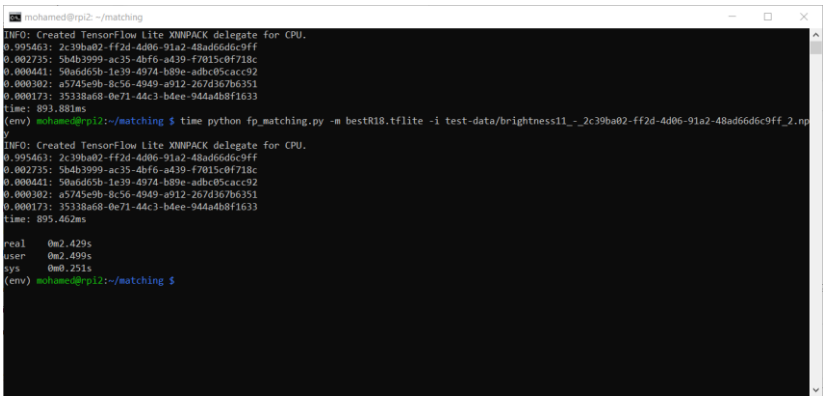


Figure 79: Off-chain fingerprinting processing time for embedded setups

V.3. Summary

The present Chapter V Blockchain-fingerprinting applicative synergies investigates the possibility of establishing synergies between the COLLATE blockchain load balancing solution and the COMMON compressed domain video fingerprinting. It lays between two incremental questioning realms.

First, at the very integration level, binding together COLLATE and COMMON rise doubts about the viability of such a complex end-to-end processing workflow. Secondly, at the application level, doubts about whether the to be deployed solution reaches the *a priori* consensual objectives expected from any blockchain application are rose. Such objective are set in conjunction with the execution speed and the blockchain inner fees. Specifically, current day practices expect for a new transactional block to be produced each 12 sec. on a Ethereum environment [ETH24e] and each 10 sec. for a Tezos environment [OPE24b]. In parallel, the maximum fees possible for producing a block are set at 30 000 000 gas units [ETH24e] and 1 040 000 gas units per transaction [OPE24c], for the same two blockchain environments, respectively.

Figure 80 wraps up the different BIDDING advantages and demonstrates the effectiveness of BIDDING by two key quantitative results:

- The fingerprinting matching time correspond to general expectancies in advertising video tracking, namely 0.06 sec. and 0.8 sec when considering server based and Raspberry Pi based off-chain environments, respectively,
- The underlying blockchain fees are not prohibitive (*cf.* Figure 74): although the fees related to video processing are significantly higher (by a factor of 25) than usual operations, they stay non-prohibitive (0.000954 ETH).

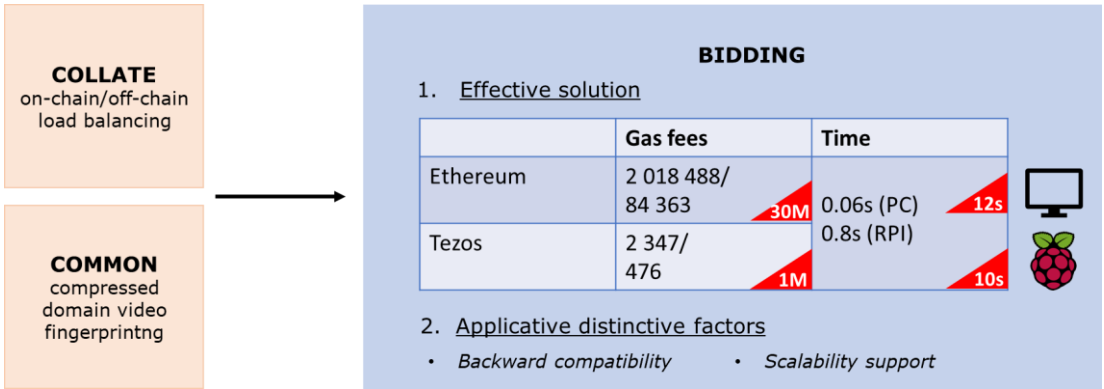


Figure 80: BIDDING performances in term of gas fee and fingerprint matching duration

As a final remark, note that any blockchain solution is ultimately and intrinsically unscalable, and that state-of-the-art solutions mainly consider the consensus protocol adjustment and/or layer 2 adjustment as palliatives when user/data volumetry is at stake. From this perspective, BIDDING has as advantage its backward compatibility while offering as alternative the possibility of dynamically adjusting the block size, composition

and pace. Moreover, fees related to massive data storage can be significantly reduced by using complementary solutions, like IPFS (Inter Planetary File System) or secured off-chain data storage as discussed in [MOR23].

Chapter VI. Conclusion

This chapter concludes our work, highlighting our contributions and putting them in perspective on the current-day mutations in the digital content landscape, covering its typology, distribution and computing aspects.

VI.1. A retrospective view on the results

The present thesis deals with compressed-domain video content tracking and considers two complementary scientific methodological and technical frameworks, *blockchain* and video *fingerprinting*.

When individually considered, these two frameworks come with complementary properties, encompassing from security, decentralized and transparent processing to robust and unique tracking of near duplicated visual content.

Hence, establishing synergies between these two approaches would potentially open the door towards a whole range of versatile and customizable applications for video content tracking.

Yet, when trying to pragmatically benefit from such synergies, several conceptual, methodological and technical dead-locks are identified:

- blockchain-based video tracking can be possible only if the current-day storage/computing/programming blockchain capabilities are virtually upgraded at the levels required by video processing applications,
- compressed-domain video stream syntax elements are not promising candidates for fingerprints composition, as they are *a priori* uncorrelated with the content visual features,
- applicative synergies between blockchain and fingerprinting frameworks are scarce in state-of-the-art studies.

While the main thesis contributions are detailed in Chapter I Overview, in Table 1 and illustrated in Figure 8,we shall briefly recap here some main issues:

- **COLLATE**, an on-Chain Off-chain Load baLancing ArchiTecturE, making it possible for the intimately constrained computing, storage and software resources of any blockchain to be abstractly extended by general-purpose computing machine resources; the distinctive factor of this architecture are (1) the platform generality (demonstrated for both Ethereum and Tezos platforms) and (2) its backward compatibility (demonstrated by the seamless integration of already existing Smart Contracts and AI algorithms, on the *on-chain* and *off-chain* fronts, respectively); the experiment validation relates to an use case of relevance for ISO/IEC 23093 – Internet of Media Things, namely the monetization of an AI algorithm deployed for celebrity recognition.
- **COMMON** – Compressed dOMain Marketing videO fiNgerprinting, demonstrating the possibility of modelling compressed video fingerprint under deep learning framework; the distinctive factors of this solution are (1) the PoC for compressed-domain stream syntax elements fingerprinting, (2) the robustness against codec modification, and (3) the stability with respect to mundane deep learning modifications, like pruning and/or quantizing; the experiments relate to two databases, of relevance for scientific community (UVG) or for the industrial partner (VIDMIZER) and show *Accuracy*, *Precision* and *Recall* values larger than 0.9.

- **BIDDING** – BlockchaIn-baseD viDeo fINgerprintinG, an end-to-end processing pipeline coupling compressed domain video fingerprinting to the blockchain load balancing solution; the distinctive factors of this processing pipeline are (1) the functional generality (demonstrated though de deployment of a realistic AI-based video fingerprinting scenario), and (2) the fine-grain control, allowing for *a priori* complex AI tasks to be deployed on embedded devices while interacting with blockchains; the experiments correspond to an use case of relevance for or industrial partner (VIDMIZER) and show that the blockchain security properties can be granted without any loss in *Accuracy*, *Precision* and *Recall* values.

VI.2. Perspectives

The digital content field is expected to face multiple revolutions that will change not only its very typology but also its distribution and processing frameworks. Consequently, the research perspectives of the thesis work will be drawn according to these three items, as illustrated in Figure 81 and detailed here-after.

Perspective 1: COMMON extensions: fingerprint for AI generated visual content and for AI

Over the last 20 years, the sources of content creation have already undergone essential changes: professional content (cinematography, TV, ...) has very quickly been dethroned by self-produced content: as an example, in 2012, 72 hours of video were posted every minute on social networks! Self-produced content, in its turn, was quickly dethroned by content generated by automated video production and processing solutions: as early as 2018, 140 hours of video surveillance content were generated every second, in London alone. This trend has been accentuated by autonomous vehicles (more than 8 cameras on board a single autonomous car) and, more recently, by generative AI solutions.

Note that from the content tracking point of view, AI generated content come with a two folded problem: the generated content and the AI in itself, that becomes now a new type of digital content.

Hence, in an incremental order, the first two directions of research open in the manuscript are the fingerprinting of AI generated content and the fingerprinting of the AI itself.

Note that research related to AI processing, where AI is considered as a new type of content, is already started (*e.g.* AI compression in ISO/IEC JTC1 SC29 a.k.a. MPEG) and the specific topic of AI fingerprinting is just launched inside MPAI standardization organization.

Moreover, on a longer time perspective, note that the findings related to AI fingerprinting will also represent a basis for approaching the fingerprinting of AI-based, end-to-end video coded content.

Perspective 2: COLLATE extensions: web3.0-based AI computing

The evolution of the digital world is intrinsically linked to the evolution of the web, which, despite its relatively short history, is already in its third generation. The last decade of the 20th century corresponded to web1.0, also known as the “read-only web”, characterized by static content, uploaded and downloaded by users, with sporadic updates reserved for professionals (or, at least, informed users). Its successor, web 2.0, is also known as the “social web”: driven by the emergence of social networks and the exponential growth of their users, content becomes dynamic, with continuous content updates requiring no technical knowledge. Nevertheless, in both cases, processing was (at least conceptually) centralized and dependent on trusted authorities (social networks, certificate providers,

etc.). Since 2014, web 3.0 or the "decentralized web" has opposed web 2.0 with a strategy based on distributed computing in a network where nodes are equal and security is assured by design, without the need for a trusted third party.

Yet, for time being, AI is not accounted as a web3.0 application and this is mainly because of their underlying computational complexity as well as of their lock-in with their processing environments.

Under this framework, COLLATE extensions will be able to accommodate complex, collaborative and distributed AI-applications to be deployed.

Perspective3: BIDDING extensions: Secure edge-computing video fingerprinting

6G networks offer the multimedia players (camera producers, infrastructure operators, service providers, broadcasters, journalists, etc.) unforeseen opportunities: proximity computing and storage resources enabling the execution of complex AI algorithms, transmission latencies reduced by factors ranging from 100 to 1000, bandwidth (for a given latency) increased by a factor of 1000, and carbon footprint reduction. This frees multimedia players from the constraints of distribution, processing and storage (read/write) latencies, enabling them to abstract the multimedia processing chain into a unitary operation in terms of time and resources.

This framework, which is conducive to economic and societal innovation, comes up against a lack of confidence in the content generated in this way. For example, video content captured by a 6G connected camera may be maliciously modified by a local AI before it even reaches a trusted server.

Under this framework, coupling at the very edge level video fingerprinting and blockchain will contribute to establish trust and to a smoother adoption of 6G.

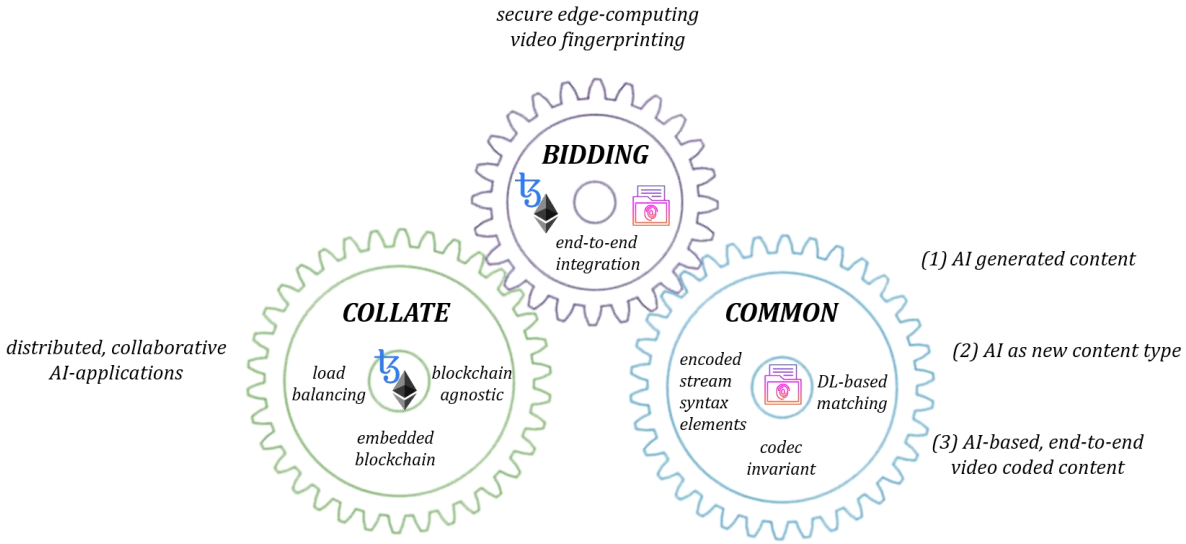


Figure 81: Short-term research perspectives for the work carried out in the thesis.

References

[ABI24] Contract ABI Specification — Solidity 0.8.29 documentation URL https://docs.soliditylang.org/en/latest/abi-spec.html (accessed 11.4.24).

[ABU16] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, A. (Paul), Toderici, G., Varadarajan, B., Vijayanarasimhan, S., 2016. YouTube-8M: A Large-Scale Video Classification Benchmark, in: arXiv:1609.08675.

[AFA22] Afandi, W., Bukhari, S.M.A.H., Khan, M.U.S., Maqsood, T., Khan, S.U., 2022. Fingerprinting Technique for YouTube Videos Identification in Network Traffic. IEEE Access 10, 76731–76741. https://doi.org/10.1109/ACCESS.2022.3192458

[ALL21a] Allouche, M., Ljubojevic, M., Mitrea, M., 2021. Visual document tracking and blockchain technologies in mobile world. Electronic Imaging 33, 1–7.

[ALL21b] Allouche, M., Mitrea, M., Moreaux, A., Kim, S.-K., 2021. Automatic Smart Contract generation for Internet of Media Things. ICT Express 7, 274–277. https://doi.org/10.1016/j.icte.2021.08.009

[ALL21c] Allouche, M., Frikha, T., Mitrea, M., Memmi, G., Chaabane, F., 2021. Lightweight Blockchain Processing. Case Study: Scanned Document Tracking on Tezos Blockchain. Appl. Sci. 11, 7169. https://doi.org/10.3390/app11157169

[ALL22] Allouche, M., Mitrea, M., 2022. Video fingerprinting: Past, present, and future. Front. Signal Process. 2, 984169. https://doi.org/10.3389/frsip.2022.984169

[ALL23] Allouche, M., Mitrea, M., De Sousa Trias, C., 2023. HEVC fingerprinting: conventional vs. ML methods. Case study: organic video content. TAIMA 2023

[ALL25a] Allouche, M., Mitrea, M., De Sousa Trias, C., "HEVC Compressed Video Fingerprinting", accepted for IS&T Electronic Imaging, 2025

[ALL25b] Allouche, M., Colle, E., Zoughebi, M., De Sousa Trias, C., Mitrea, M., "Green video encoder identification", accepted for IS&T Electronic Imaging, 2025.

[AL-N24] Al-Nbhany, W.A.N.A., Zahary, A.T., Al-Shargabi, A.A., 2024. Blockchain-IoT Healthcare Applications and Trends: A Review. IEEE Access 12, 4178–4212. https://doi.org/10.1109/ACCESS.2023.3349187

[AMM18] Ammar, M., Mitrea, M., Hasnaoui, M., Le Callet, P., 2018. MPEG-4 AVC stream-based saliency detection. Application to robust watermarking. Signal Process. Image Commun. 60, 116–130. https://doi.org/10.1016/j.image.2017.09.007

[ANU20] Anuranji, R., Srimathi, H., 2020. A supervised deep convolutional based bidirectional long short term memory video hashing for large scale video retrieval applications. Digital Signal Processing 102, 102729. https://doi.org/10.1016/j.dsp.2020.102729

[BAD18] Badia-Melis, R., Mc Carthy, U., Ruiz-Garcia, L., Garcia-Hierro, J., Robla Villalba, J.I., 2018. New trends in cold chain monitoring applications - A review. Food Control 86, 170–182. https://doi.org/10.1016/j.foodcont.2017.11.022

[BEN10] Benois-Pineau, J., 2010. Indexing of compressed video: Methods, challenges, applications, in: 2010 2nd International Conference on Image Processing Theory, Tools and Applications. Presented at the 2010 2nd International Conference on Image Processing Theory, Tools and Applications (IPTA), IEEE, Paris, France, pp. 3–4. https://doi.org/10.1109/IPTA.2010.5586830

[BER15] Bernd, J., Borth, D., Elizalde, B., Friedland, G., Gallagher, H., Gottlieb, L., Janin, A., Karabashlieva, S., Takahashi, J., Won, J., 2015. The YLI-MED Corpus: Characteristics, Procedures, and Plans. https://doi.org/10.48550/ARXIV.1503.04250

[BOS19] Bosu, A., Iqbal, A., Shahriyar, R., Chakraborty, P., 2019. Understanding the motivations, challenges and needs of Blockchain software developers: a survey. Empir. Softw. Eng. 24, 2636–2673. https://doi.org/10.1007/s10664-019-09708-7

[BUR10] Bursuc, A., Zaharia, T., Prêteux, F., 2010. OVIDIUS: an on-line video indexing universal system. Presented at the SPIE Optical Engineering + Applications, San Diego, California, United States, p. 77990C. https://doi.org/10.1117/12.863195

[CHA05] Changick Kim, Vasudev, B., 2005. Spatiotemporal sequence matching for efficient video copy detection. IEEE Trans. Circuits Syst. Video Technol. 15, 127–132. https://doi.org/10.1109/TCSVT.2004.836751

[CHE21] Chen, H., Pendleton, M., Njilla, L., Xu, S., 2021. A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses. ACM Comput. Surv. 53, 1–43. https://doi.org/10.1145/3391195

[CHE24] Chen, W., Gan, W., Yu, P.S., 2024. Digital Fingerprinting on Multimedia: A Survey. https://doi.org/10.48550/ARXIV.2408.14155

[CHO05] Chong-Wah Ngo, Yu-Fei Ma, Hong-Jiang Zhang, 2005. Video summarization and scene detection by graph modeling. IEEE Trans. Circuits Syst. Video Technol. 15, 296–305. https://doi.org/10.1109/TCSVT.2004.841694

[CIS18] Cisco, 2018. Cisco Visual Networking Index: Forecast and Trends, 2017–2022.

[CHO05] Chong-Wah Ngo, Yu-Fei Ma, Hong-Jiang Zhang, 2005. Video summarization and scene detection by graph modeling. IEEE Trans. Circuits Syst. Video Technol. 15, 296–305. https://doi.org/10.1109/TCSVT.2004.841694

[COR12] Correa, G., Assuncao, P., Agostini, L., Da Silva Cruz, L.A., 2012. Performance and Computational Complexity Assessment of High-Efficiency Video Encoders. IEEE Trans. Circuits Syst. Video Technol. 22, 1899–1909. https://doi.org/10.1109/TCSVT.2012.2223411

[COS06] Coskun, B., Sankur, B., Memon, N., 2006. Spatio–Temporal Transform Based Video Hashing. IEEE Trans. Multimed. 8, 1190–1208. https://doi.org/10.1109/TMM.2006.884614

[CHE18] Chen, Y., Murherjee, D., Han, J., Grange, A., Xu, Y., Liu, Z., Parker, S., Chen, C., Su, H., Joshi, U., Chiang, C.-H., Wang, Y., Wilkins, P., Bankoski, J., Trudeau, L., Egge, N., Valin, J.-M., Davies, T., Midtskogen, S., Norkin, A., De Rivaz, P., 2018. An Overview of Core Coding Tools in the AV1 Video Codec, in: 2018 Picture Coding Symposium (PCS). Presented at the 2018 Picture Coding Symposium (PCS), IEEE, San Francisco, CA, pp. 41–45. https://doi.org/10.1109/PCS.2018.8456249

[CHR16] Christidis, K., Devetsikiotis, M., 2016. Blockchains and Smart Contracts for the Internet of Things. IEEE Access 4, 2292–2303. https://doi.org/10.1109/ACCESS.2016.2566339

[DAS23] Das, S., 2023. What is MarTech? Medium. URL https://medium.com/@101writer/what-is-martech-40443a9e095d (accessed 10.15.24).

[DIG08] Digital Watermarking and Steganography, 2008. . Elsevier. https://doi.org/10.1016/B978-0-12-372585-1.X5001-3

[DIL24] Dilawari, A., Iqbal, S., Syed, F., Mudassar Ilyas, Q., 2024. Deep Metric Learning for Near-Duplicate Video Retrieval Leveraging Efficient Semantic Feature Extraction. IEEE Access 12, 88897–88903. https://doi.org/10.1109/ACCESS.2024.3411101

[DOC24] Oracles | Tezos Developer Documentation, URL https://docs.tezos.com/smart-contracts/oracles (accessed 12.13.24).

[DOS22] Dos Santos, S., Singh, J., Thulasiram, R.K., Kamali, S., Sirico, L., Loud, L., 2022. A New Era of Blockchain-Powered Decentralized Finance (DeFi) - A Review, in: 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC). Presented at the 2022 IEEE 46th Annual Computers,

Software, and Applications Conference (COMPSAC), IEEE, Los Alamitos, CA, USA, pp. 1286–1292. https://doi.org/10.1109/COMPSAC54236.2022.00203

[DOU08] Douze, M., Gaidon, A., Jégou, H., Marszalek, M., Schmid, C., 2008. INRIA-LEARs video copy detection system, in: TREC Video Retrieval Evaluation (TRECVID Workshop). Gaithersburg, United States.

[ETH24a] Token Standards. URL https://ethereum.org/en/developers/docs/standards/tokens/ (accessed 11.15.24).

[ETH24b] Smart Contract languages. URL https://ethereum.org/en/developers/docs/smart-contracts/languages/ (accessed 11.5.24).

[ETH24c] Non-fungible tokens (NFT). URL https://ethereum.org/en/nft/ (accessed 11.20.24).

[ETH24d] Oracles [WWW Document], URL https://ethereum.org/en/developers/docs/oracles/ (accessed 12.13.24).

[ETH24e] Blocks, ethereum.org. URL https://ethereum.org/en/developers/docs/blocks/ (accessed 12.13.24).

[GAR16] Garboan, A., Mitrea, M., 2016. Live camera recording robust video fingerprinting. Multimed. Syst. 22, 229–243. https://doi.org/10.1007/s00530-014-0447-0

[GHO21] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K., 2021. A Survey of Quantization Methods for Efficient Neural Network Inference. https://doi.org/10.48550/ARXIV.2103.13630

[GIT24a] GitHub - ethereum/solidity: Solidity, the Smart Contract Programming Language, URL https://github.com/ethereum/solidity (accessed 10.2.24).

[GIT24b] GitHub - OCamlPro/liquidity: A high-level language for Dune Network (and Tezos) with OCaml and ReasonML syntaxes, with a decompiler from Michelson [WWW Document]. URL https://github.com/OCamlPro/liquidity/tree/next (accessed 10.2.24).

[GIT24c] jvet / HM · GitLab URL https://vcgit.hhi.fraunhofer.de/jvet/HM (accessed 10.28.24).

[GIT24d] Keras-CIFAR10/classifiers/ResNet.py GitHub. URL https://github.com/jerett/Keras-CIFAR10/blob/master/classifiers/ResNet.py (accessed 11.12.24).

[GIT24e] Blockchain-fingerprinting / Common GitLab. URL https://gitlab.com/blockchain-fingerprinting/common (accessed 11.14.24).

[HAM01] Hampapur, A., Bolle, R.M., 2001. Comparison of distance measures for video copy detection, in: IEEE International Conference on Multimedia and Expo, 2001. ICME 2001. Presented at the IEEE International Conference on Multimedia and Expo, 2001. ICME 2001, IEEE, Tokyo, Japan, pp. 737–740. https://doi.org/10.1109/ICME.2001.1237827

[HAS14] Hasnaoui, M., Mitrea, M., 2014. Multi-symbol QIM video watermarking. Signal Process. Image Commun. 29, 107–127. https://doi.org/10.1016/j.image.2013.07.007

[HAS18] Hassanat, A.B.A., 2018. Norm-Based Binary Search Trees for Speeding Up KNN Big Data Classification. Computers 7, 54. https://doi.org/10.3390/computers7040054

[HEI15] Heilbron, F.C., Escorcia, V., Ghanem, B., Niebles, J.C., 2015. ActivityNet: A large-scale video benchmark for human activity understanding, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Boston, MA, USA, pp. 961–970. https://doi.org/10.1109/CVPR.2015.7298698

[HE16] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90

[HOC97] Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation 9, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[HOU15] Hou, Y., Wang, X., Liu, S., 2015. Multiple features video fingerprint algorithm based on optical flow feature, in: 2015 International Conference on Computers, Communications, and Systems (ICCCS). Presented at the 2015 International Conference on Computers, Communications, and Systems (ICCCS), IEEE, Kanyakumari, India, pp. 159–162. https://doi.org/10.1109/CCOMS.2015.7562893

[HU18] Hu, Y., Lu, X., 2018. Learning spatial-temporal features for video copy detection by the combination of CNN and RNN. Journal of Visual Communication and Image Representation 55, 21–29. https://doi.org/10.1016/j.jvcir.2018.05.013

[HOW19] Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., Le, Q., 2019. Searching for MobileNetV3, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Presented at the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, Seoul, Korea (South), pp. 1314–1324. https://doi.org/10.1109/ICCV.2019.00140

[HOW24] How Smart Contracts will Revolutionize the Real Estate Sector URL https://www.yourvirtualplace.com/blog/Real%20Estate/How-Smart-Contracts-will-Revolutionize-the-Real-Estate-Sector/1726233205999 (accessed 10.20.24).

[IDR97] Idris, F., Panchanathan, S., 1997. Review of Image and Video Indexing Techniques. J. Vis. Commun. Image Represent. 8, 146–166. https://doi.org/10.1006/jvci.1997.0355

[ISO20] ISO/IEC 23093-1:2020 Information technology - Internet of media things - Part 1: Architecture URL https://webstore.iec.ch/en/publication/66671 (accessed 11.4.24).

[ISO22] ISO/IEC 23093-2:2022 Information technology - Internet of media things - Part 2: Discovery and communication API URL https://www.iso.org/standard/81587.html (accessed 11.4.24).

[JIA11] Jiang, Y.-G., Ye, G., Chang, S.-F., Ellis, D., Loui, A.C., 2011. Consumer video understanding: a benchmark database and an evaluation of human and machine performance, in: Proceedings of the 1st ACM International Conference on Multimedia Retrieval. Presented at the ICMR'11: International Conference on Multimedia Retrieval, ACM, Trento Italy, pp. 1–8. https://doi.org/10.1145/1991996.1992025

[JIA12] Jiang, S., Su, L., Huang, Q., Cui, P., Wu, Z., 2013. A Rotation Invariant Descriptor for Robust Video Copy Detection, in: The Era of Interactive Media. Springer New York, New York, NY, pp. 557–567. https://doi.org/10.1007/978-1-4614-3501-3_46

[JIA16] Jiang, Y.-G., Wang, J., 2016. Partial Copy Detection in Videos: A Benchmark and an Evaluation of Popular Methods. IEEE Trans. Big Data 2, 32–42. https://doi.org/10.1109/TBDATA.2016.2530714

[KER24] Keras documentation: Keras Applications. URL https://keras.io/api/applications/ (accessed 11.15.24).

[KOR17a] Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., Kompatsiaris, Y., 2017. Near-Duplicate Video Retrieval by Aggregating Intermediate CNN Layers, in: Amsaleg, L., Guðmundsson, G.Þ., Gurrin, C., Jónsson, B.Þ., Satoh, S. (Eds.), MultiMedia Modeling, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 251–263. https://doi.org/10.1007/978-3-319-51811-4_21

[KOR17b] Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., Kompatsiaris, Y., 2017. Near-Duplicate Video Retrieval with Deep Metric Learning, in: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW). Presented at the 2017 IEEE International Conference on Computer Vision Workshop (ICCVW), IEEE, Venice, Italy, pp. 347–356. https://doi.org/10.1109/ICCVW.2017.49

[KRA13] Krawczyk, H., Paterson, K.G., Wee, H., 2013. On the Security of the TLS Protocol: A Systematic Analysis, in: Canetti, R., Garay, J.A. (Eds.), Advances in Cryptology – CRYPTO 2013, Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 429–448. https://doi.org/10.1007/978-3-642-40041-4_24

[KRI12] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks, in: Pereira, F., Burges, C.J., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.

[LAN24] Languages | Tezos Developer Documentation URL https://docs.tezos.com/smart-contracts/languages (accessed 10.29.24).

[LAW07] Law-To, J., Gouet-Brunet, V., Buisson, O., Boujemaa, N., 2007. Video Copy Detection on the Internet: The Challenges of Copyright and Multiplicity, in: Multimedia and Expo, 2007 IEEE International Conference On. Presented at the Multimedia and Expo, 2007 IEEE International Conference on, IEEE, Beijing, China, pp. 2082–2085. https://doi.org/10.1109/ICME.2007.4285092.

[LEE08] Lee, S., Yoo, C.D., 2008. Robust video fingerprinting for content-based video identification. IEEE Trans. Circuits Syst. Video Technol. 18, 983–988. https://doi.org/10.1109/TCSVT.2008.920739

[LI13] Li, M., Monga, V., 2013. Compact Video Fingerprinting via Structural Graphical Models. IEEE Trans.Inform.Forensic Secur. 8, 1709–1721. https://doi.org/10.1109/TIFS.2013.2278100

[LI21] Li, X., Guo, C., Yang, Y., Xu, L., 2021. Video fingerprinting based on quadruplet convolutional neural network. Systems Science & Control Engineering 9, 131–141. https://doi.org/10.1080/21642583.2020.1822946

[LI18] Li, Y., Huang, Y., Xu, R., Seneviratne, S., Thilakarathna, K., Cheng, A., Webb, D., Jourjon, G., 2018. Deep Content: Unveiling Video Streaming Content from Encrypted WiFi Traffic, in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). Presented at the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, Cambridge, MA, pp. 1–8. https://doi.org/10.1109/NCA.2018.8548317

[LIU13] Liu, J., Huang, Z., Cai, H., Shen, H.T., Ngo, C.W., Wang, W., 2013. Near-duplicate video retrieval: Current research and future trends. ACM Comput. Surv. 45, 1–23. https://doi.org/10.1145/2501654.2501658

[LIU18] Liu, M., Po, L.-M., Zhou, C., Yuen, W.Y.F., Cheung, H.-K., Wong, P.H.W., Luk, H.-T., Lau, K.-W., 2018. Content-based Video Copy Detection using Binary Object Fingerprints, in: 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). Presented at the 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), IEEE, Qingdao, pp. 1–6. https://doi.org/10.1109/ICSPCC.2018.8567827

[LIU19] Liu, M., Po, L.-M., Ur Rehman, Y.A., Xu, X., Li, Y., Feng, L., 2019. Video copy detection by conducting fast searching of inverted files. Multimed Tools Appl 78, 10601–10624. https://doi.org/10.1007/s11042-018-6639-4

[LIU23] Liu, J., 2023. Digital signature and hash algorithms used in Bitcoin and Ethereum, in Zhou, F., Ba, S. (Eds.), Third International Conference on Machine Learning and Computer Application (ICMLCA 2022). Presented at the Third International Conference on Machine Learning and Computer Application (ICMLCA 2022), SPIE, Shenyang, China, p. 235. https://doi.org/10.1117/12.2675431

[MAO16] Mao, J., Xiao, G., Sheng, W., Hu, Y., Qu, Z., 2016. A method for video authenticity based on the fingerprint of scene frame. Neurocomputing 173, 2022–2032. https://doi.org/10.1016/j.neucom.2015.09.001

[MAN07] Manerba, F., Benois-Pineau, J., Leonardi, R., Mansencal, B., 2007. Multiple Moving Object Detection for Fast Video Content Description in Compressed Domain. EURASIP J. Adv. Signal Process. 2008, 231930. https://doi.org/10.1155/2008/231930

[MET24] Top Three Libraries for Web3 Developers URL https://metamask.io/news/developers/top-three-libraries-for-web3-developers/ (accessed 11.5.24).

[MEZ04] Mezaris, V., Kompatsiaris, I., Boulgouris, N.V., Strintzis, M.G., 2004. Real-Time Compressed-Domain Spatiotemporal Segmentation and Ontologies for Video Indexing and Retrieval. IEEE Trans. Circuits Syst. Video Technol. 14, 606–621. https://doi.org/10.1109/TCSVT.2004.826768

[MIT19] Mitrea, M., 2019. IoMT White Paper. ISO/IEC JTC1/SC29/WG11 N18879, Geneva, CH. Available at: https://mpeg.chiariglione.org/sites/default/files/files/standards/docs/w18879.zip (accessed 10.3.24).

[MON19] Monrat, A.A., Schelen, O., Andersson, K., 2019. A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. IEEE Access 7, 117134–117151. https://doi.org/10.1109/ACCESS.2019.2936094

[MOR23a] Moreaux, A., 2023. Visual content tracking, IPR management, & blockchain : from process abstraction to functional interoperability (Theses). Institut Polytechnique de Paris.

[MOR23b] Moreaux, A.C., Mitrea, M.P., 2023. Royalty-Friendly Digital Asset Exchanges on Blockchains. IEEE Access 11, 56235–56247. https://doi.org/10.1109/ACCESS.2023.3283153

[MUK13] Mukherjee, D., Bankoski, J., Grange, A., Han, J., Koleszar, J., Wilkins, P., Xu, Y., Bultje, R., 2013. The latest open-source video codec VP9 - An overview and preliminary results, in: 2013 Picture Coding Symposium (PCS). Presented at the 2013 Picture Coding Symposium (PCS), IEEE, San Jose, CA, USA, pp. 390–393. https://doi.org/10.1109/PCS.2013.6737765

[NAK08] Nakamoto, S., & Bitcoin, A. (2008). A peer-to-peer electronic cash system. Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf, 4(2), 15.

[NIE15] Nie, X., Liu, J., Wang, Q., Zeng, W., 2015. Graph-based video fingerprinting using double optimal projection. Journal of Visual Communication and Image Representation 32, 120–129. https://doi.org/10.1016/j.jvcir.2015.08.001

[OPE24a] Token Standards | OpenTezos URL https://opentezos.com/defi/token-standards/ (accessed 11.15.24).

[OPE24b] Operations | OpenTezos, https://opentezos.com/tezos-basics/operations/ (accessed 12.13.24)

[OPE24c] Economics and rewards | OpenTezos, https://opentezos.com/tezos-basics/economics-and-rewards/ (accessed 12.13.24).

[OTH20] Othman, M., Chen, K., Mokraoui, A., 2020. A User-experience Driven SSIM-Aware Adaptation Approach for DASH Video Streaming. https://doi.org/10.48550/ARXIV.2012.05696

[FPE20] Pfeiffer, A., Kriglstein, S., Wernbacher, T., 2020. Blockchain Technologies and Games: A Proper Match?, in: International Conference on the Foundations of Digital Games. Presented at the FDG '20: International Conference on the Foundations of Digital Games, ACM, Bugibba Malta, pp. 1–4. https://doi.org/10.1145/3402942.3402996

[PRO24] Progonov, D., Mandrenko, A., Kuzmenko, D., Kuznetsov, V., Derkach, V., Morgunov, M., 2024. Robust near-duplicate video retrieval using compressed representation, in: 2024 IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). Presented at the 2024 IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), IEEE, Niagara Falls, ON, Canada, pp. 1–8. https://doi.org/10.1109/AVSS61716.2024.10672608

[PYT24] PyTezos by Baking Bad. URL https://pytezos.org/ (accessed 11.5.24).

[QUR20] Qureshi, A., Megías Jiménez, D., 2020. Blockchain-Based Multimedia Content Protection: Review and Open Challenges. Appl. Sci. 11, 1. https://doi.org/10.3390/app11010001

[REN16] Ren, D., Zhuo, L., Long, H., Qu, P., Zhang, J., 2016. MPEG-2 Video Copy Detection Method Based on Sparse Representation of Spatial and Temporal Features, in: 2016 IEEE Second International Conference on Multimedia Big Data (BigMM). Presented at the 2016 IEEE Second International Conference on Multimedia Big Data (BigMM), IEEE, Taipei, Taiwan, pp. 233–236. https://doi.org/10.1109/BigMM.2016.21

[SAB17] Sabour, S., Frosst, N., Hinton, G.E., 2017. Dynamic Routing Between Capsules. https://doi.org/10.48550/ARXIV.1710.09829

[SCH17] Schuster, R., Shmatikov, V., Tromer, E., 2017. Beauty and the Burst: Remote Identification of Encrypted Video Streams, in: 26th USENIX Security Symposium (USENIX Security 17). USENIX Association, Vancouver, BC, pp. 1357–1374.

[SCH21] Schär, F., 2021. Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets. r 103. https://doi.org/10.20955/r.103.153-74

[SCI24] Decision Trees. scikit-learn. URL https://scikit-learn/stable/modules/tree.html (accessed 11.14.24).

[SHA14] Shahid, Z., Puech, W., 2014. Visual Protection of HEVC Video by Selective Encryption of CABAC Binstrings. IEEE Trans. Multimedia 16, 24–36. https://doi.org/10.1109/TMM.2013.2281029

[STA24a] Advertising - Worldwide | Statista Market Forecast [WWW Document]. Statista. URL https://www-statista-com.mediaproxy.imtbs-tsp.eu/outlook/amo/advertising/worldwide (accessed 10.3.24).

[STA24b] Revenue in the TV & Video Advertising market for different segments Worldwide 2019-2029, Statista. URL https://www-statista-com.mediaproxy.imtbs-tsp.eu/forecasts/1442798/revenue-tv-video-advertising-market-for-different-segments-worldwide (accessed 10.3.24).

[STA24c] U.S. daily TV and digital viewing time 2025, Statista. URL https://www.statista.com/statistics/186833/average-television-use-per-person-in-the-us-since-2002/ (accessed 10.7.24).

[STA24d]Number of worldwide social network users 2028 [WWW Document], n.d. . Statista. URL https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/ (accessed 10.17.24).

[STE24] Telestream Resurrects encoding.com's Global Media Format Report, 2023. . Streaming Learning Center. URL https://streaminglearningcenter.com/codecs/telestream-resurrects-encoding-coms-global-media-format-report.html (accessed 11.15.24).

[SUN17] Sun, R., Yan, X., Gao, J., 2017. Robust video fingerprinting scheme based on contourlet hidden Markov tree model. Optik 128, 139–147. https://doi.org/10.1016/j.ijleo.2016.09.105

[SZA94] Szabo, N. (1994). Smart Contracts

[TAN09] Tan, H.-K., Ngo, C.-W., Hong, R., Chua, T.-S., 2009. Scalable detection of partial near-duplicate videos by visual-temporal consistency, in: Proceedings of the 17th ACM International Conference on Multimedia. Presented at the MM09: ACM Multimedia Conference, ACM, Beijing China, pp. 145–154. https://doi.org/10.1145/1631272.1631295

[TEN24] Post-training quantization | TensorFlow Model Optimization URL https://www.tensorflow.org/model_optimization/guide/quantization/post_training (accessed 11.20.24).

[THO15] Thomas, R.M., Sumesh, M.S., 2015. A Simple and Robust Colour Based Video Copy Detection on Summarized Videos. Procedia Computer Science 46, 1668–1675. https://doi.org/10.1016/j.procs.2015.02.106

[TSC16] Tschorsch, F., Scheuermann, B., 2016. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. IEEE Commun. Surv. Tutorials 18, 2084–2123. https://doi.org/10.1109/COMST.2016.2535718

[TRI24]Trias, C.D.S., Mitrea, M., Fiandrotti, A., Cagnazzo, M., Chaudhuri, S., Tartaglione, E., 2024. WaterMAS: Sharpness-Aware Maximization for Neural Network Watermarking. https://doi.org/10.48550/ARXIV.2409.03902

[VID24] x265, the free H.265/HEVC encoder VideoLAN URL https://www.videolan.org/developers/x265.html (accessed 11.12.24).

[WAN16] Wang, R.B., Chen, H., Yao, J.L., Guo, Y.T., 2016. Video Copy Detection Based On Temporal Contextual Hashing, in: 2016 IEEE Second International Conference on Multimedia Big Data (BigMM). Presented at the 2016 IEEE Second International Conference on Multimedia Big Data (BigMM), IEEE, Taipei, Taiwan, pp. 223–228. https://doi.org/10.1109/BigMM.2016.12

[WAN19] Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., Wang, F.-Y., 2019. Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends. IEEE Trans. Syst. Man Cybern. Syst. 49, 2266–2277. https://doi.org/10.1109/TSMC.2019.2895123

[WEI11]Wei, S., Zhao, Y., Zhu, C., Xu, C., Zhu, Z., 2011. Frame Fusion for Video Copy Detection. IEEE Trans. Circuits Syst. Video Technol. 21, 15–28. https://doi.org/10.1109/TCSVT.2011.2105554

[XIN09] Xing Su, Tiejun Huang, Gao, W., 2009. Robust video fingerprinting based on visual attention regions, in: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. Presented at the ICASSP 2009 - 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, Taipei, Taiwan, pp. 1525–1528. https://doi.org/10.1109/ICASSP.2009.4959886

[XIN21] Xinwei, L., Lianghao, X., Yi, Y., 2021. Compact video fingerprinting via an improved capsule net. Systems Science & Control Engineering 9, 122–130. https://doi.org/10.1080/21642583.2020.1833782

[YAN12] Yang, G., Chen, N., Jiang, Q., 2012. A robust hashing algorithm based on SURF for video copy detection. Computers & Security 31, 33–39. https://doi.org/10.1016/j.cose.2011.11.004

[YUA16] Yuan, F., Po, L.-M., Liu, M., Xu, X., Jian, W., Wong, K., Cheung, K.W., 2016. Shearlet Based Video Fingerprint for Content-Based Copy Detection. JSIP 07, 84–97. https://doi.org/10.4236/jsip.2016.72010

[ZHA19] Zhang, C., Lin, Y., Zhu, L., Liu, A., Zhang, Z., Huang, F., 2019. CNN-VWII: An efficient approach for large-scale video retrieval by image queries. Pattern Recognition Letters 123, 82–88. https://doi.org/10.1016/j.patrec.2019.03.015

[ZHA21] Zhao, W., Yang, S., Jin, M., 2021. Near-duplicate Video Retrieval Based on Deep Unsupervised Key Frame Hashing, in: 2021 IEEE 24th International Conference on Computational Science and Engineering (CSE). Presented at the 2021 IEEE 24th International Conference on Computational Science and Engineering (CSE), IEEE, Shenyang, China, pp. 80–86. https://doi.org/10.1109/CSE53436.2021.00021

[ZHA22] Zhao, Y., Kang, X., Li, T., Chu, C.-K., Wang, H., 2022. Toward Trustworthy DeFi Oracles: Past, Present, and Future. IEEE Access 10, 60914–60928. https://doi.org/10.1109/ACCESS.2022.3179374

[ZHO19] Zhou, Z., Chen, J., Yang, C.-N., Sun, X., 2019. Video Copy Detection Using Spatio-Temporal CNN Features. IEEE Access 7, 100658–100665. https://doi.org/10.1109/ACCESS.2019.2930173

[ZHO20] Zhou, Q., Huang, H., Zheng, Z., Bian, J., 2020. Solutions to Scalability of Blockchain: A Survey. IEEE Access 8, 16440–16455. https://doi.org/10.1109/ACCESS.2020.296721

ECOLE DOCTORALE

Titre : Traçage du contenu marketing vidéo

Mots clés : fingerprinting visuel, flux vidéo codés, blockchain,

Abstract : Au cours des dernières décennies, la production et la consommation du contenu vidéo a considérablement augmenté et il est communément admis que 80 % du trafic Internet est constitué par de vidéos. Dans ce cadre, le traçage du contenu vidéo compressé et subissant un enchainement de distributions sur des réseaux sociaux et/ou plateformes vidéo est un problème qui peut être géré à la fois par la blockchain et par l'empreinte sémantique (fingerprinting) vidéo. Les principaux résultats obtenus consistent en la conception, la spécification et la mise en œuvre : (1) d'une architecture de répartition du calcul et du stockage on-chain / off-chain, (2) d'un cadre méthodologique démontrant la possibilité de modéliser des empreintes vidéo compressées dans un cadre d'apprentissage profond et (3) d'un pipeline de traitement de bout en bout qui permet de coupler l'empreinte vidéo à la solution d'équilibrage de charge de la blockchain.

Title : Video tracking for marketing applications

Keywords : visual fingerprint, encoded video stream, blockchain

Abstract : Over the last few decades, the production and consumption of video content has significantly increased, and it is widely estimated that 80% of Internet traffic is video. In this context, the tracking of compressed video content undergoing subsequent distributions on social networks and/or video platforms is a problem that can be addressed by both blockchain and video fingerprinting. The main results consist in the design, specification and implementation of: (1) an *on-chain*/*off-chain* load balancing architecture, (2) a methodological framework demonstrating the possibility of achieving compressed video fingerprinting under the deep learning framework, and (3) an end-to-end processing pipeline that couples the video fingerprinting to the blockchain load balancing solution.