InVID

Deliverable D3.3: Final verification framework (This is a public redacted version of a confidential deliverable.)

Markos Zampoglou, Symeon Papadopoulos, Giorgos Kordopatis-Zilos,
Olga Papadopoulou, Vasileios Mezaris, Alexandros Metsai,
Konstantinos Apostolidis, Yiannis Kompatsiaris, Lazaros Apostolidis,
Christos Koutlis, Dimitrios Giomelakis / CERTH
Lyndon Nixon, Adrian M.P. Brasoveanu / MODUL
Roger Cozien, Gregoire Mercier / EXO MAKINA

24/12/2018

Work Package 3: Content Verification

InVID - In Video Veritas: Verification of Social Media Video Content for the News Industry

Innovation Action

Horizon 2020, Research and Innovation Programme

Grant Agreement Number 687786

Dissemination level	CO
Contractual date of delivery	31/12/2018
Actual date of delivery	24/12/2018
Deliverable number	D3.3
Deliverable name	Final verification framework (This is a public redacted version of a confidential deliverable.)
File	`output.tex`
Nature	Report
Status & version	Final & V1.0
Number of pages	58
WP contributing to the deliverable	3
Task responsible	CERTH
Other contributors	MODUL, EXO MAKINA
Author(s)	Markos Zampoglou, Symeon Papadopoulos, Giorgos Kordopatis-Zilos, Olga Papadopoulou, Vasileios Mezaris, Alexandros Metsai, Konstantinos Apostolidis, Yiannis Kompatsiaris, Lazaros Apostolidis, Christos Koutlis, Dimitrios Giomelakis / CERTH Lyndon Nixon, Adrian M.P. Brasoveanu / MODUL Roger Cozien, Gregoire Mercier / EXO MAKINA
Quality Assessors	Denis Teyssou / AFP, Gregoire Mercier / EXO
EC Project Officer	Alberto Rabbachin
Keywords	Content verification, video forensic analysis, near-duplicate detection, logo detection, context aggregation, location detection

Abstract

The InVID project has come to its conclusion. The tools and technologies that we have been developing, integrating, and testing during the previous years are now in their final form. Work Package 3 has provided an array of tools, integrated in a powerful platform, which aims to provide journalists and investigators with enhanced capabilities in verifying news-related user-generated videos. The previous two deliverables of the Work Package, D3.1 and D3.2, described the tools that were developed, the state-of-the-art algorithms and technologies that were implemented, adapted, and improved, the user feedback that was received and the way it shaped the component development, and the status of integration at the end of the first and second year respectively. Here, we present the final versions of the components and services, the achieved improvements in algorithmic and service performance, and the final integration status. Through their integration with the Verification Plugin and the Verification Application, these components of InVID have been seeing increasing real-world usage since the second project year, and an increased uptake in the third year. Compared to the previous year, all components have seen substantial improvements. Our work in Video Forensics remains confidential and the corresponding content has been redacted from the document. However, our progress in the other components is presented here openly, taken verbatim from the original, confidential version of D3.2.

- Our work in *Video Forensics* was geared towards automated or semi-automated video analysis. Besides our confidential work we also produced openly published research, in which our previous year's work into convolutional neural networks for tampering detection (i.e. taking the filter outputs and returning a single-value result on the probability that the video was tampered) was significantly extended with further models, datasets, and experiments.
- In *Near-Duplicate Detection*, the algorithm developed during the previous years was further improved, leading to an approach that further surpasses the state of the art in accuracy. This is achieved by combining the proposed Deep Metric Learning approach with a Chamfer Distance metric to exploit the distances between video frames during video similarity calculation. We also completed the development of a very large-scale dataset which allows for realistic evaluations of Near-Duplicate Video Retrieval algorithms, and also enables evaluations in Fine-grained Video Retrieval tasks. Furthermore, several improvements and extensions were made in the service functionalities according to the obtained feedback.
- The *Logo Detection* module was improved by replacing it with a more reliable deep learning framework, extending its coverage with user submitted contributions, and improving its performance by adapting and extending the synthetic training data generation process with further training data augmentation steps.
- The *Location Detection* module was further improved in terms of accuracy through the inclusion of several disambiguation steps that reduce the number of errors and lead to increased performance. Furthermore, our efforts to provide a more reliable evaluation dataset and methodology have led to the development of an entire ecosystem of tools for the integration of the Recognyze tool but also for performance evaluations.
- Finally, the *Context Aggregation and Analysis* module was extended with new functionalities, and underwent improvements with respect to speed, reliability, and the structure of the provided information. In parallel, the increasing user base that has been developed during the second and third year of InVID has allowed us to use this component to significantly expand the Fake Video Corpus dataset. Combined with the Near-Duplicate Detection algorithm, a large dataset including well-established cases of fake and real videos was created, including their reposts and near-duplicates. In the context of the CAA component, the characteristic patterns of this dataset were explored, with the aim of gathering insights for contextual video verification.

The integration of these components is now complete, having reached a level of seamless interaction with the InVID platform. Their constant use in operational conditions guarantees that, besides their achievements with respect to evaluations and benchmarks, these tools are also ready for large-scale real-world use, providing state-of-the-art performance for real-world video verification.

Content

1 Introduction

This deliverable presents the progress made during the third year of the InVID project for Work Package 3: Content Verification, and describes the final version of the delivered verification framework. The objective of WP3 is to develop a set of tools that can assist journalists with content verification tasks, by speeding up existing manual procedures, through innovative and intelligent software components.

The rest of the document is organized as follows: The remaining of this section provides a summary of the WP3 achievements during the third year of the project. Section 2 presents our analysis of the video verification problem, and the final version of the Fake Video Corpus (FVC-2018) dataset, which comprises several real-world examples of relevance for video verification. The section also analyzes the role of each WP3 component in tackling the problem, their interrelations, and the progress of WP3 as a whole. The subsequent sections are dedicated to individual components. Section 3 presents our progress with Video Forensics, Section 4 deals with Near Duplicate Detection, Section 5 presents the Logo Detection component, Section 6 presents our progress in Location Detection, and Section 7 presents the Context Aggregation and Analysis component. Finally, in Section 8 we provide an overview of the work done so far, the overall influence of InVID in the field of video verification, and our estimate of the future impact of our work.

1.1 History of the document

Table 1: History of the document

Date	Version	Name	Comment
2018/09/03	V0.1	M. Zampoglou / CERTH	Document structure
2018/09/07	V0.11	S. Papadopoulos, V. Mezaris / CERTH	Structure edits
2018/09/12	V0.2	O. Papadopoulou, L. Apostolidis, D. Giomelakis, C. Koutlis / CERTH	Context Aggregation and Analysis section
2018/09/18	V0.21	V. Mezaris, Y. Kompatsiaris, S. Papadopoulos / CERTH	Document structure revisions
2018/09/28	V0.3	L. Nixon, A. Brasoveanu / MODUL	Location Detection section
2018/10/12	V0.4	R. Cozien, G. Mercier / EXO MAKINA	Video Forensics section
2018/10/21	V0.5	M. Zampoglou, L. Apostolidis / CERTH	Logo detection section
2018/11/05	V0.51	V. Mezaris, A. Metsai, K. Apostolidis / CERTH	Video forensics section update
2018/11/12	V0.6	O. Papadopoulou, D. Giomelakis, C. Koutlis / CERTH	Content Verification - Overview section
2018/11/15	V0.7	G. Kordopatis-Zilos / CERTH	Near-Duplicate Detection section
2018/11/27	V0.8	M. Zampoglou, A. Metsai / CERTH	Video forensics section update
2018/12/05	V0.9	M. Zampoglou, O. Papadopoulou, S. Papadopoulos / CERTH	Proofreading and editing
2018/12/21	V1.0	M. Zampoglou, S. Papadopoulos, L. Apostolidis / CERTH	Final version

1.2 Purpose of the document

The document aims to present our work in WP3 during the third year of InVID and to describe the final outcomes of this work. The Work Package contains three tasks:

- **Multimedia forensics**, aiming to detect digital manipulations of the video content by examining the video bitstream (T3.1 - EXO MAKINA, CERTH).
- **Near-duplicate content detection**, aiming to identify whether a posted image or video has been reposted in the past (T3.2 - CERTH).
- **Contextual verification**, aiming to provide information regarding the location and social network context of a posted item to assist users with verification (T3.3 - CERTH, MODUL).

The purpose of this deliverable is to document the developments for all three of the aforementioned tasks during the third year of the project, and to provide an overall view of the progress achieved towards the

WP objectives. The aim of D3.3 is defined as "*...the final version of the content verification framework, following extensive evaluation and testing on top of the InVID platform. The final version will incorporate improvements and updates based on the results that will have been collected from the last cycles of testing and evaluation.*"

This deliverable presents these extensions and new implementations and their degree of integration with the platform. They are accompanied with qualitative and quantitative evaluations of the achieved performance of the new components, with a focus on progress since Year 2. The achievements of this year include:

1. The extension of the Fake Video Corpus, previously created and extended in InVID, into its final version, the Fake Video Corpus 2018 (FVC-2018). The dataset includes a large number of "fake" and "real" cases and its near-duplicates, allowing for large-scale analysis of disinformation dissemination.
2. The development of two deep learning algorithms, one aimed at semi-automatic tampering localization, and the other at fully-automated tampering detection. These models take as input the results of the video forensics filters developed in the previous years of the project and are trained on datasets of tampered and untampered videos. The tampering localization algorithm produces binary localization maps, while the tampering detection algorithm produces single value estimates on whether the video is tampered.
3. An improved near-duplicate retrieval algorithm, reaching superior performance to state-of-the-art methods, and a large-scale dataset of real-world videos for near-duplicate retrieval evaluations, also allowing evaluations of fine-grained video retrieval.
4. An improved, fast and accurate TV logo detection algorithm based on an artificial data augmentation approach combining high performance with scalability to a large number of known logos.
5. A superior location detection algorithm utilizing an array of disambiguation techniques, accompanied by an ecosystem of different components (data, tools) for evaluation.
6. Improvements in the context aggregation and analysis module, including the addition of further functionalities to provide more powerful contextual analysis. In addition, a second contribution is an analysis of the distinctive patterns within FVC-2018, and the potential of automatic verification systems for contextual analysis.

In this document, both the final status of individual components and of the WP as a whole are presented, and the overall current and future impact of our work during the InVID project is assessed.

1.3 Glossary and Abbreviations

Application Programming Interface (API): In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between software components.

Computer Generated Imagery (CGI): This refers to multimedia content (image, video) that is created exclusively or to a large extent with the assistance of software, i.e. does not depict a scene captured from the real world.

Convolutional Neural Networks (CNN): In machine learning, a CNN (or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. CNNs are typically applied on visual recognition tasks.

Deep Metric Learning (DML): A machine learning approach based on neural networks, where an embedding function is learned to map items to a new feature space based on the pair/triplet-wise relations of the training samples in a development corpus.

Deep Neural Network (DNN): A machine learning model consisting of multiple layers of "artificial neurons" or "units". A modern version of Artificial Neural Networks (ANNs).

Discrete Cosine Transform (DCT): The DCT is a technique for converting a signal into elementary frequency components.

Fake Video Corpus (FVC): The video dataset created within InVID for the purposes of identifying and classifying types of fakes, and evaluating various verification approaches.

Image/Video Tampering: This is the act of digitally altering an image or video file either to enhance it (e.g. improve contrast) or to mislead people by generating false evidence. Tampering is also referred to as forgery, manipulation or more colloquially as photoshopping.

JavaScript Object Notation (JSON): This is an open-standard format that uses human-readable text to transmit data objects consisting of attributevalue pairs. It is the most common data format used for asynchronous browser/server communication.

MPEG-4: This is a method of defining compression of audio and visual (AV) digital data. It was introduced in late 1998 and designated a standard for a group of audio and video coding formats and related technology agreed upon by the ISO/IEC Moving Picture Experts Group (MPEG).

Named Entity Recognition (NER): This is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

Named Entity Linking (NEL): This is an extension of the NEL task which seeks to also link the classified results to the corresponding entries from a Knowledge Base like Wikipedia, DBpedia or Wikidata.

Near-duplicate detection (NDD), Near-Duplicate Video Retrieval (NDVR): This refers to the task of retrieving multimedia items (images, videos) that are highly similar or identical to a given multimedia item, which is referred to as query.

Radial Basis Function Support Vector Machine (RBF-SVM): An Support Vector Machine is a supervised machine learning model able to achieve non-linear classification through so-called “kernel functions”. Radial Basis Functions are a type of such kernel functions.

Region proposal Convolutional Neural Network (RCNN): A type of Deep Neural Network which takes an image as input, and returns a number of region proposals and the classification results for each one of them, thus performing object detection.

Representational state transfer (REST): Also known as RESTful Web services, this refers to a paradigm of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.

SPARQL Protocol and RDF Query Language (SPARQL): This is an RDF query language, i.e. a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format.

Speeded Up Robust Features (SURF): In computer vision, SURF is a local feature detector and descriptor. It can be used for tasks such as object recognition, image registration and classification.

Slot Filling or Cold Start Slot Filling (SF or CSSF): Is an information extraction task in which a system needs to complete (or fill) all the available information on a particular entity. Typically this is done with respect to a schema that defines the type of information that can be extracted about particular entity types.

Term Frequency - Inverse Document Frequency (tf-idf): This is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, but also in the context of image retrieval in conjunction with *visual vocabularies*.

Uniform Resource Locator (URL): Commonly termed a web address, this is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

User Generated Content (UGC): This refers to multimedia content that is generated by any individual (i.e. often amateurs) and is publicly shared through some media sharing platform (e.g. YouTube, Facebook, Twitter, etc.).

Video Forensics: This refers to a class of video analysis methods that aim to detect traces of tampering in video content.

Work Package (WP): This refers to the structure of InVID work into units called Work Packages.

2 Content verification – Overview

2.1 Content verification in the Wild

The stated aims of InVID have been to provide tools for journalists, news professionals, and investigators, to gather and verify video with respect to their usefulness in news story production. In the course of the project, these tools were adopted by a significant part of the verification community, and this began generating a significant amount of traffic to the InVID services, including those of WP3. This had the added and unforeseen benefit of in turn allowing us to observe which video items these professionals were submitting for verification, and through this, which videos were considered newsworthy at the time that the project was running.

In the two previous WP3 deliverables, we presented two versions of the dataset we collected, named the Fake Video Corpus. The dataset contained cases of "fake" and "real" videos[1], that is videos which were used to spread misinformation, and videos that contained newsworthy UGC that might as well have been associated with misinformation, but were finally proven to be truthful. In both first versions of the dataset, we manually selected the videos to populate it, with the help of news professionals from our InVID partners. For the dataset gathered during the third project year, named the Fake Video Corpus 2018 (FVC-2018) and published in (Papadopoulou, Zampoglou, Papadopoulos, & Kompatsiaris, 2018), we followed a different strategy, made possible due to the high degree of adoption of the InVID services by the verification community.

By processing the anonymized logs of the Context Aggregation and Analysis service, which was serving the highly adopted InVID plugin, we collected a list of approximately 1600 videos that the users had been submitting during the lifetime of the tool. Since we are dealing with a free and open tool, this means that not all videos submitted are going to be relevant. Thus, we manually checked all submitted videos to remove those that were clearly irrelevant (e.g. clips from TV shows, games, etc.), and further removed all videos that were already present in the Fake Video Corpus. We noted a high degree of overlap between the videos collected in this manner, and the videos already in the FVC, which is a good indication that the videos we had been collecting during the first two years of the project are relevant cases for our user community. The remaining videos after the above culling process were manually verified using external sources (i.e. credible news sources and debunking websites such as *snopes.com*) and classified as *Real* or *Fake* similar to the annotation followed in the previous versions of the Fake Video Corpus.

Following this process, the additional videos were added to the previous version of the FVC, resulting in a dataset of 200 videos annotated as *fake* and 180 annotated as *real*. This collection can help researchers and the verification community to analyze the phenomenon of video-based disinformation, and is in itself a significant contribution to the state of the art. Figure 1 shows some cases out of this collection.

However, using the technologies developed in InVID, we took the opportunity to move one step further. Specifically, using the near-duplicate retrieval algorithm of WP3, presented in Section 4, we proceeded to study how information is disseminated through time. To achieve this, we followed a structured methodology in six steps:

1. For each video in the initial set, extract the title.
2. Simplify the title by retaining only the most crucial information. For example, the title Video Tornado IRMA en Florida EEUU Video impactante was simplified to Tornado IRMA at Florida.
3. Translate the event title into English, Russian, Arabic, French, and German using Google Translate.
4. Use the video title, event title, and the four translations as separate queries to three target platforms: YouTube, Facebook, Twitter.
5. Use the near-duplicate retrieval algorithm of (Kordopatis-Zilos et al, 2017) to search within the returned videos, for near-duplicates of the initial video.
6. Apply a manual confirmation step to remove any erroneous results of the method and only retain actual near-duplicates.

[1] We recognize that the labels "fake" and "real" oversimplify the problem and have been often misused in the public debate. However, for the sake of brevity and simplicity, we use them to refer to the annotations of the Fake Video Corpus.

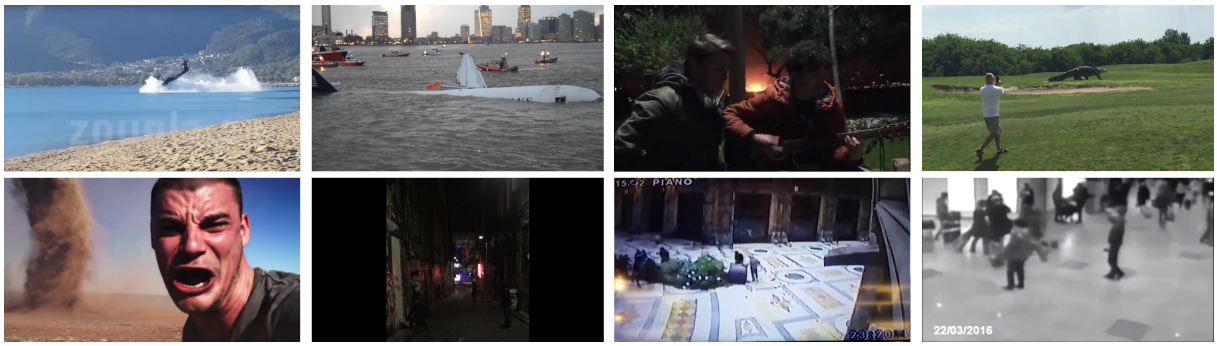


Figure 1: Indicative cases of real and fake User-Generated Videos from FVC-2018. Top: four *real* videos. a) A Greek army helicopter crashing into the sea in front of beach; b) US Airways Flight 1549 ditched in the Hudson River; c) A group of musicians playing in an Istanbul park while bombs explode outside the stadium behind them; d) A giant alligator crossing a Florida golf course. Bottom: four *fake* videos. a) A man taking a selfie with a tornado -CGI; b) The artist Banksy caught in action -staged; c) Muslims destroying a Christmas tree in Italy -out of context, there is no indication that they are Muslim; d) Bomb attack on Brussels airport -out of context, footage is from Moscow Domodedovo airport in 2011.

7. Temporally order all gathered videos, including the initial one, into a “video cascade”, starting with the oldest posted video containing all its near-duplicates ordered by publication time.

This led to the creation of cascades of near-duplicate videos, published at different times and with a varying degree of modifications (Figure 2). It should be noted that, the way that the title was restated in step 2) intentionally limits the scope of the search. That is, if the initial video was contextually fake, and the original content was not from hurricane Irma, then the algorithm will only retrieve those versions of the video that claim to depict hurricane Irma. Thus, the original video which was taken from a different tornado will not be included in the set, and the cascade will contain all versions of the specific falsehood. Thus each cascade corresponds to a particular misinformation and does not intend to contain all near-duplicates of the same video currently available online. This is a necessary limitation, as removing the search constraints (e.g. searching simply for ”Tornado” in order to collect all possible instances of the same content) would return too many results to handle.

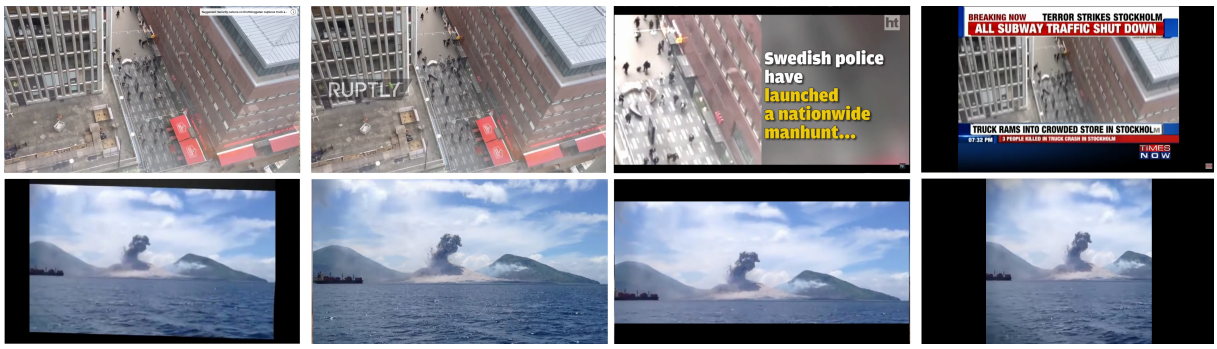


Figure 2: Indicative video near-duplicates for two different cascades.

Methodologically, two further steps were applied to extend and refine the dataset. The first was to submit the URL of the first video of each cascade to Twitter search, and collect all tweets sharing the video as a link. The second was to study all gathered videos and their context, to ensure they all correspond to the assigned title. This is important because, for example, when we start with a video that is contextually fake with respect to the claims made in its description, it is possible that we will also collect a real version of it containing a truthful description. Since each cascade is assigned a single ”fake” or ”real” title, clearly these two videos should not be analyzed as part of the same cascade.

Thus, all videos were re-annotated with respect to the intended class of the cascade they belong to. The entire process led to the collection of 3,929 fake videos and 2,463 real videos, organized in 200 and 180 cascades respectively. Out of those, we initially excluded 467 fake videos and 350 real ones, taken from Facebook, since they were listed as private and our access to their metadata was

restricted. The rest of the videos were annotated with respect to their relation to the initial video in the cascade. Thus, the categories for near-duplicates of fake videos are: a) Fake/Fake: those that reproduce the same false claims; b) Fake/Uncertain: those that express doubts on the veracity of the claim; c) Fake/Debunk: those that attempt to debunk the original claim; d) Fake/Parody: those that use the content for fun/entertainment; e) Fake/Real: those that contain the earlier, original source from which the fake was made. For near-duplicates of real videos, the corresponding categories are: a) Real/Real: those that reproduce the same factual claims b) Real/Uncertain, those that express doubts on the veracity of the claim; c) Real/Debunk: those that attempt to debunk their claims as false; d) Real/Parody: those that use the content for fun/entertainment. A special category concerns videos labeled Real/Private and Fake/Private, which describes Facebook videos that were relevant to the dataset but were published by individual users and thus could not be accessed through the API in order to extract their context. These were left out of the analysis entirely. Table 2 shows the number of videos that corresponded to each category and each platform. The column labeled Total corresponds to all videos, but does not include the twitter posts that share the video, which are counted separately, and the videos listed as "private" which are not counted at all. The resulting annotated dataset has been presented in (Papadopoulou et al., 2018) and is freely available for research purposes[2].

	Fake videos						**Real videos**				
	YT	FB	TW	Total	TW Shares		YT	FB	TW	Total	TW Shares
Initial	189	11	0	200	-	Initial	158	22	0	180	-
Fake	1,675	928	113	2,716	44,898	Fake	993	901	16	1,910	28,263
Private	-	*467*	-	*467*	-	Private	-	*350*	-	*350*	-
Uncertain	207	122	10	339	3,897	Uncertain	0	1	0	1	30
Debunk	68	19	0	87	170	Debunk	2	0	0	2	0
Parody	43	2	1	46	0	Parody	14	6	0	20	0
Real	22	51	1	74	0						
Total	**2,204**	**1,133**	**125**	**3,462**	**48,965**	**Total**	**1,167**	**930**	**16**	**2,113**	**28,293**

Table 2: Types of near-duplicate videos contained in FVC-18. Private videos are not included in the totals.

FVC-2018, i.e. the final version of the Fake Video Corpus produced within InVID, is, to our knowledge, the largest annotated research database of misleading and truthful video content currently available. As a first result of collecting and studying the FVC-2018, we have come to the conclusion that the original typology of misleading videos that we formed in D3.1, was not fully accurate. Thus, we devised a new typology as follows:

1. Decontextualized videos that are visually unchanged or almost unchanged, including low quality copies for clickbait purposes.
2. Decontextualized videos that have also been altered (e.g. cut in length to one or several fragments of the original video, or cropped to remove e.g. a timestamp in a CCTV camera footage).
3. Staged videos (e.g. produced on purpose by a video producer company).
4. Videos that have been tampered through editing software to remove, hide, duplicate or add some visual or audio content.
5. Computer-generated Imagery (CGI) including deep fakes (i.e. content generated by Artificial Intelligence) either generated from scratch or mixed with a blend of previous footage.

The distribution of these 5 categories among the 200 initial videos of the FVC-2018 is shown in Table 3. Although it can be seen that all categories are represented in the corpus, it is clear that CGI videos are a minority. This makes sense, as it is rather demanding to produce them. On the other hand, decontextualized videos are the largest category, especially if we add those with minor alterations. It is also interesting to note that the dataset also contains many staged videos, a number of which have been also post-processed.

These are the use-cases that we may encounter, and the role of each WP3 component in the overall verification process should be evaluated with respect to them.

[2] https://github.com/MKLab-ITI/fake-video-corpus

Category	# videos
1 – Decontextualized unchanged	77
2 – Decontextualized altered	13
3 – Staged	35
4 – Tampered	38
5 – CGI	9
Staged & Tampered	23
Staged & CGI	5
Total	200

Table 3: Types of near-duplicate videos contained in FVC-2018.

2.2 Content verification in InVID

In D3.1 we presented the first version of the WP3 modules, and in D3.2 we presented their improved versions. In this document, we present their final versions. As the InVID project is reaching its conclusion, and the various components are being evaluated for market application, it would be important to reiterate the role that each component is aimed to play in the overall verification process, how they are interrelated, and the overall framework coverage.

For videos that have been digitally tampered or contain CGI, detection can be achieved using the Video Forensics component. This component includes a set of filters, aimed to be applied on videos and allow investigators to spot inconsistencies after studying the results. Section 3 presents our progress in this area, including our efforts to design a system that can automatically classify videos as tampered or authentic by analyzing the filter outputs, and a system that can take the filter results as input and return a binary tampering localization map that can highlight where the tampering might have taken place.

With respect to videos that have been published in the past and are being reposted out of context, with or without alterations, the proposed solution is the Near Duplicate Detection module, which includes a growing index of videos from past events that may be used as future reposts, and an innovative near-duplicate retrieval algorithm used to search this index and check whether a video appearing as new is actually a near-duplicate of a video already present in the index. In Section 4 we present our improvements on the algorithm and the resulting evaluations, as well as our progress with increasing the size of the dataset.

In a similar manner of contextual analysis, Location Detection can allow us to detect inconsistencies in the video context, or to identify details concerning its source, which can be telltale of specific biases. Section 6 presents our progress in this module and the significant improvements we achieved with respect to accuracy and disambiguation.

Finally, for all cases of fake videos, analyzing their content as a whole can greatly assist verification, In Section 7 we present the improvements and modifications of the Context Aggregation and Analysis component, as well as our analysis of the FVC-2018 corpus with respect to identifying the tell-tale patterns that distinguish fake from real videos. Furthermore, we present our explorations into automatic contextual video verification.

2.2.1 Progress and evaluations during Year 3

During the final project year, all modules underwent significant improvements with respect to their features, their technologies, and their integrated implementations. Particular focus was placed in the evaluations of all components, both qualitatively and quantitatively. With respect to the latter, quantitative evaluations were run on all modules, using appropriate datasets. While we consider the FVC-2018 to be a central outcome of InVID, and a landmark in the field of video verification, offering a definitive collection of established fake and real news-related videos and their near-duplicates, it was not the only dataset used in our evaluations. FVC-2018 was appropriate for the quantitative evaluations of automatic contextual verification algorithms in the context of the CAA component, but three more datasets have also been produced as a result of our work in InVID[3].

– The Near-Duplicate Detection dataset presented in D3.2 has been extended and annotated, allowing not only evaluations on Near-Duplicate Retrieval tasks, but also evaluations on Fine-grained Video Retrieval. The dataset, named FIVR-200K, is presented in detail Section 4.

[3]More details regarding the data management aspects of these datasets are provided in the updated Data Management Plan.

- The Logo Detection dataset consisting of a large number of TV videos, presented in D3.1 and D3.2, and is used again in the evaluations of Section 5 following minor corrections in annotation and organization.
- The ("Lenses") dataset, initially created with geolocation in mind but subsequently expanded to cover events and other entity types as well. It was presented in D3.2.

Besides these datasets that we created ourselves, established benchmark datasets were also used for quantitative evaluations of the various modules, such as the NIST Media Forensics Challenge 2018 and the GRIP tampered video dataset used for Video forensics, the CC_WEB_VIDEO dataset used for near-duplicate detection, and the Reuters-128 dataset used in location detection evaluation. Furthermore, we tried to use the Fake Video Corpus for evaluations wherever it was relevant -in this document, parts of it are also used for certain qualitative and quantitative examples of the new Video Forensics algorithms, besides the automatic contextual analysis benchmark evaluations.

3 Video Forensics

A large part of the work conducted for video forensics during the third year of the project is confidential. For our published work in this area, we used certain forensic analysis filters for tampering detection, i.e. with the aim of producing single-value estimates indicating the probability that a video may have been tampered. The work presented here has been published in (Zampoglou et al., 2019).

3.1 State of the art

Multimedia forensics has been an active research field for more than a decade. A number of algorithms (known as *active* forensics) work by embedding invisible watermarks on images which are disturbed in case of tampering. Alternatively, *passive* forensics aim to detect tampering without any prior knowledge (Piva, 2013). Image forensics is an older field than video forensics, with a larger body of proposed algorithms and experimental datasets, and is slowly reaching maturity as certain algorithms or algorithm combinations are approaching sufficient accuracy for real-world application. Image tampering detection is often based on detecting local inconsistencies in JPEG compression information, or – especially in the cases of high-quality, low-compression images – detecting local inconsistencies in the high-frequency noise patterns left by the capturing device. A survey and evaluation of algorithms focused on image splicing can be found in (Zampoglou, Papadopoulos, & Kompatsiaris, 2016).

The progress in image forensics might lead to the conclusion that similar approaches could work for tampered video detection. If videos were simply sequences of frames, this might hold true. However, modern video compression is a much more complex process that often removes all traces such as camera error residues and single-frame compression traces (Sitara & Mehtre, 2016). Proposed video forensics approaches can be organized in three categories: double/multiple quantization detection, inter-frame forgery detection, and region tampering detection.

In the first case, systems attempt to detect if a video or parts of it have been quantized multiple times (Y. Su & Xu, 2010; J. Xu, Su, & liu, 2013). A video posing as a camera-original User-Generated Content (UGC) but exhibiting traces of multiple quantizations may be suspicious. However, with respect to newsworthy UGC, such approaches are not particularly relevant since in the vast majority of cases videos are acquired from social media sources. As a result, both tampered and untampered videos typically undergo multiple strong requantizations and, without access to a purported camera original, they have little to offer in our task.

In the second category, algorithms aim to detect cases where frames have been inserted in a sequence, which has been consecutively requantized (Y. Wu, Jiang, Sun, & Wang, 2014; Zhang, Hou, Ma, & Li, 2015). Since newsworthy UGC generally consists of a single shot, such frame insertions are unlikely to pass unnoticed. Frame insertion detection may be useful for videos with fixed background (e.g. CCTV footage) or for edited videos where new shots are added afterwards, but the task is outside the scope of this work.

Finally, the third category concerns cases where parts of a video sequence (e.g. an object) have been inserted in the frames of another. This the most relevant scenario for UGC, and the focus of our work. Video region tampering detection algorithms share many common principles with image splicing detection algorithms. In both cases, the assumption is that there exists some invisible pattern in the item, caused by the capturing or the compression process, which is distinctive, detectable, and can be disturbed when foreign content is inserted. Some approaches are based solely on the spatial information extracted independently from frames. Among them, the most prominent ones use oriented gradients (Subramanyam & Emmanuel, 2012), the Discrete Cosine Transform (DCT) coefficients' histogram (Labartino et al., 2013), or Zernike moments (D'Amiano, Cozzolino, Poggi, & Verdoliva, 2015). These work well as long as the video quality is high, but tend to fail at higher compression rates as the traces on which they are based are erased. Other region tampering detection strategies are based on the motion component of the video coding, modeling motion vector statistics (W. Wang & Farid, 2007; Li, Wang, Wang, & Hu, 2013) or motion compensation error statistics (Chen, Tan, Li, & Huang, 2016). These approaches work better with still background and slow moving objects, using motion to identify shapes/objects of interest in the video. However, these conditions are not often met by UGC.

Other strategies focus on temporal noise (Pandey, Singh, & Shukla, 2014) or correlation behavior (Lin & Tsay, 2014). The noise estimation induces a predictable feature shape or background, which imposes an implicit hypothesis such as a limited global motion. The *Cobalt* filter we use adopts a similar strategy. The Motion Compensated Edge Artifact is another alternative to deal with the temporal behavior of residuals between I, P and B frames without requiring strong hypotheses on the motion

or background contents. These periodic artifacts in the DCT coefficients may be extracted through a thresholding technique (L. Su, Huang, & Yang, 2015) or spectral analysis (Dong, Yang, & Zhu, 2012). This approach is also used for inter-frame forgery detection under the assumption that the statistical representativeness of the tampered area should be high.

Recently, the introduction of deep learning approaches has led to improved performance and promising results for video manipulation detection. In (Yao, Shi, Weng, & Guan, 2017), the inter-frame differences are calculated for the entire video, then a high-pass filter is applied to each difference output and the outputs are used to classify the entire video as tampered or untampered. High-pass filters have been used successfully in the past in conjunction with machine learning approaches with promising results in images (Fridrich & Kodovsky, 2012). In a similar manner, (Rössler et al., 2018) presents a set of deep learning approaches for detecting face-swap videos created by Generative Adversarial Networks. Besides presenting a very large-scale dataset for training and evaluations, they show that a modified Xception network architecture can be used to detect forged videos on a per-frame basis.

3.2 Method description

3.2.1 Tampering localization

[Content removed as confidential.]

3.2.2 Tampering detection

The work presented here aims at an approach which can take the filter outputs and return a single-value estimate that the video may have been forged, i.e. a video tampering detection system. The basis of the approach was this:

1. A video is split into frames, which are processed by a forensic filter.
2. The outputs of the forensic filter, annotated as “tampered” or “untampered” are used to fine-tune a pre-trained Convolutional Neural Network, in order to separate between the two classes.
3. When faced with a new, unknown video, the process is repeated for all its frames, and the video is classified as “tampered” or “untampered” by fusing the per-frame estimates.

During the second year, we had tested the Cobalt filter on a modified GoogLeNet model, on a small dataset of 23 tampered and 23 untampered videos. Building upon the promising results of the second year, we proceeded to conduct more in-depth experiments. While the essential methodology remained the same, we extended our effort to more models, filters and datasets.

Specifically, besides the *Cobalt* filter we also ran experiments with the *Q4* filter. Also, for comparison with the state of the art, we also implemented three other forensic analysis filters, specifically:

– *rawKeyframes* (Rössler et al., 2018). The video is decoded into its frames and the raw keyframes (without any filtering process) are given as input to the deep network.

– *highPass frames* (Fridrich & Kodovsky, 2012). The video is decoded into its frames, each frame is filtered by a high-pass filter and the filtered frame is given as input to the deep network.

– *frameDifference* (Yao et al., 2017). The video is decoded into its frames, the frame difference between two neighboring frames is calculated, the new filtered frame is also processed by a high-pass filter and the final filtered frame is given as input to the deep network.

Furthermore, in parallel to the GoogLeNet CNN model, we implemented the ResNet CNN model, also with the addition of an extra layer which has been shown to improve performance when fine-tuning (Pittaras, Markatopoulou, Mezaris, & Patras, 2017). Finally, the experimental datasets were extended, which allowed for more extensive training and evaluation experiments. For training and evaluation, we used three datasets. Two were provided by the NIST 2018 Media Forensics Challenge, called Dev1 and Dev2, consisting of 60 and 192 videos respectively, each equally split between tampered videos and their untampered sources. Correspondingly, the two datasets consist approximately of of 44,000 and 134,000 respectively, again equally split between tampered and untampered frames. The third experimental dataset was sourced from the Fake Video Corpus. The Corpus contains both videos that convey real information (“real”), and videos that are associated with disinformation (“fake”). However,

these categories do not strictly coincide with untampered and tampered videos. There are videos in the FVC annotated as “real”, that contain watermarks or overlaid text, or have otherwise been edited, and which would trigger a tampering detection algorithm. On the other hand, there are videos in the FVC annotated as “fake” which, while contextually misleading, have not been tampered. Out of all the videos contained in the FVC, we selected 35 “real”, unedited videos, and 33 “fake” videos that were tampered with the aim of deceiving viewers, but with no obvious edits such as logos, watermarks, or cuts/transitions. In total, the FVC-based dataset we created contains 163,000 frames, which are approximately evenly distributed between tampered and untampered videos.

3.3 Progress and evaluations during Year 3

3.3.1 Tampering localization qualitative results

[Content removed as confidential.]

3.3.2 Quantitative results

For our evaluation experiments of the tampering detection method, we first applied the two chosen filters, namely Q4 and Cobalt, on all videos, and extracted all frames of the resulting output sequences to use as training and test items. Then, each of the two chosen networks, GoogLeNet and ResNet, was trained on the task using these outputs. For comparison, we also applied the three features from the state of the art that we implemented, to be used for classification in a similar manner.

As explained, during training each frame is treated as an individual image. In order to test the classifier, however, we require a per-video result. To achieve this, we extract the classification scores for all frames, and calculate the average score separately for each class (tampered, untampered). If the average score for the “tampered” class is higher than the average score for the “untampered” class, then the video is classified as tampered.

We ran two types of experiments. In one case, we trained and evaluated the algorithm on the same dataset, using 5-fold cross validation, and ensuring that all frames from a video are placed either in the training or in the evaluation set to avoid information leak. In the other case, we used one of the datasets for training, and the other two for testing. These cross-dataset evaluations are important in order to evaluate an algorithm’s ability to generalize, and to assess whether any encouraging results we observe during within-dataset evaluations are actually the result of overfitting on the particular dataset’s characteristics, rather than a true solution to the task. In all cases, we used three performance measures: Accuracy, Mean Average Precision (MAP), and Mean Precision for the top-20 retrieved items (MP@20).

For the within-dataset evaluations, we used the two NIST datasets (Dev1, Dev2) and their union. This resulted in three separate runs, the results of which are presented in Table 4.

Table 4: Within-dataset evaluations

Dataset	Filter-DCNN	Accuracy	MAP	MP@20
Dev1	cobalt-gnet	**0.6833**	0.7614	-
	cobalt-resnet	0.5833	0.6073	-
	q4-gnet	0.6500	**0.7856**	-
	q4-resnet	0.6333	0.7335	-
Dev2	cobalt-gnet	0.8791	**0.9568**	**0.8200**
	cobalt-resnet	0.7972	0.8633	0.7600
	q4-gnet	**0.8843**	0.9472	0.7900
	q4-resnet	0.8382	0.9433	0.7600
Dev1 + Dev2	cobalt-gnet	**0.8509**	0.9257	0.9100
	cobalt-resnet	0.8217	0.9069	0.8700
	q4-gnet	0.8408	**0.9369**	**0.9200**
	q4-resnet	0.8021	0.9155	0.8700

As shown on the Table 4, Dev1 consistently leads to poorer performance in all cases, for all filters and both models. Accuracy is between 0.58 and 0.68 in all cases in Dev1, while it is significantly higher in Dev2, ranging from 0.79 to 0.88. MAP is similarly significantly higher in Dev2. The reason we did not apply the MP@20 measure on Dev1 is that the dataset is so small that the test set in all cases contains less than 20 items, and thus is inappropriate for the specific measure.

We also built an additional dataset by merging Dev1 and Dev2. The increased size of the Dev1+Dev2 dataset suggests that cross-validation results will be more reliable than for the individual sets. As shown on Table 4, Mean Average Precision for Dev1+Dev2 falls between that for Dev1 and Dev2, but is much closer to Dev2. On the other hand, MP@20 is higher than for Dev2, although that could possible be the result of Dev2 being relatively small. The cross-validation Mean Average Precision for Dev1+Dev2 reaches 0.937 which is a very high value and can be considered promising with respect to the task. It is important to note that, for this set of evaluations, the two filters yielded comparable results, with Q4 being superior in some cases and Cobalt in others. On the other hand, with respect to the two CNN models there seems to be a significant difference between GoogLeNet and ResNet, with the former yielding much better results.

Within-dataset evaluations using cross-validation is the typical way to evaluate automatic tampering detection algorithms. However, as we are dealing with machine learning, it does not account for the possibility of the algorithm actually learning specific features of a particular dataset, and thus remaining useless for general application. The most important set of algorithm evaluations for InVID automatic tampering detection concerned cross-dataset evaluation, with the models being trained on one dataset and tested on another.

The training-testing sets we ran were based on the three datasets we described above, namely Dev1, Dev2, and FVC. We combine Dev1 and Dev2 to creat an additional dataset, named Dev1+Dev2. Given that Dev1 and Dev2 are both taken from the NIST challenge, although different, we would expect that they would exhibit similar properties and thus should give relatively better results than when testing on FVC. In contrast, evaluations on the FVC correspond to the most realistic and challenging scenario, that is training on benchmark, lab-generated content, and testing on real-world content encountered on social media. All cross-dataset evaluations were ran five times, with the model retrained from scratch each time from a different initialization. The results presented below are the mean results from the five runs, to ensure that they are not derived by chance.

Table 5: Cross-dataset evaluations (Training set: Dev1)

Training	Testing	Filter-DCNN	Accuracy	MAP	MP@20
Dev1	**Dev2**	cobalt-gnet	0.5818	0.7793	0.8200
		cobalt-resnet	0.6512	0.8380	0.9000
		q4-gnet	0.5232	0.8282	0.9000
		q4-resnet	0.5240	0.8266	0.9300
		rawKeyframes-gnet (Rössler et al., 2018)	0.5868	0.8450	0.8500
		rawKeyframes-resnet (Rössler et al., 2018)	0.4512	0.7864	0.7500
		highPass-gnet (Fridrich & Kodovsky, 2012)	0.5636	0.8103	0.8800
		highPass-resnet (Fridrich & Kodovsky, 2012)	0.5901	0.8026	0.8400
		frameDifference-gnet (Yao et al., 2017)	**0.7074**	**0.8585**	**0.8700**
		frameDifference-resnet (Yao et al., 2017)	0.6777	0.8240	0.8100
	FVC	cobalt-gnet	0.5147	0.5143	0.4800
		cobalt-resnet	0.4824	0.5220	0.5000
		q4-gnet	0.5824	0.6650	0.6400
		q4-resnet	**0.6441**	**0.6790**	**0.6900**
		rawKeyframes-gnet (Rössler et al., 2018)	0.5265	0.5261	0.4900
		rawKeyframes-resnet (Rössler et al., 2018)	0.4882	0.4873	0.4400
		highPass-gnet (Fridrich & Kodovsky, 2012)	0.5441	0.5359	0.5100
		highPass-resnet (Fridrich & Kodovsky, 2012)	0.4882	0.5092	0.4900
		frameDifference-gnet (Yao et al., 2017)	0.5559	0.5276	0.4600
		frameDifference-resnet (Yao et al., 2017)	0.5382	0.4949	0.5100

The cross-dataset evaluation results can be seen in Tables 5, 6, and 7.

The results are shown in Tables 5, 6, and 7. Using Dev1 to train and Dev2 to test, and vice versa, yields comparable results to the within-dataset evaluations for the same dataset, confirming our expectation that, due to the common source of the two datasets, cross-dataset evaluation for these datasets would not be particularly challenging. Compared to other state-of-the-art approaches, it seems that our proposed approaches do not yield superior results in those cases. Actually, the *frameDifference* feature seems to outperform the others in those cases.

The situation changes in the realistic case where we are evaluating on the Fake Video Corpus. In that case, the performance drops significantly. In fact, most algorithms drop to an Accuracy of almost

Table 6: Cross-dataset evaluations (Training set: Dev2)

Training	Testing	Filter-DCNN	Accuracy	MAP	MP@20
Dev2	Dev1	cobalt-gnet	0.5433	0.5504	0.5500
		cobalt-resnet	0.5633	0.6563	0.6300
		q4-gnet	**0.6267**	0.6972	**0.7100**
		q4-resnet	0.5933	0.6383	0.6300
		rawKeyframes-gnet	**0.6467**	0.6853	0.6500
		rawKeyframes-resnet	0.6200	0.6870	0.6200
		highPass-gnet (Fridrich & Kodovsky, 2012)	0.5633	0.6479	0.6600
		highPass-resnet (Fridrich & Kodovsky, 2012)	**0.6433**	0.6665	**0.6500**
		frameDifference-gnet (Yao et al., 2017)	0.6133	**0.7346**	**0.7000**
		frameDifference-resnet (Yao et al., 2017)	**0.6133**	0.7115	**0.6700**
	FVC	cobalt-gnet	0.5676	0.5351	0.5800
		cobalt-resnet	0.5059	0.4880	0.4900
		q4-gnet	**0.6118**	**0.6645**	**0.7000**
		q4-resnet	0.5000	0.4405	0.3900
		rawKeyframes-gnet (Rössler et al., 2018)	0.5206	0.6170	0.6600
		rawKeyframes-resnet (Rössler et al., 2018)	0.5971	0.6559	0.6900
		highPass-gnet (Fridrich & Kodovsky, 2012)	0.4794	0.5223	0.4700
		highPass-resnet (Fridrich & Kodovsky, 2012)	0.5235	0.5541	0.5800
		frameDifference-gnet (Yao et al., 2017)	0.4882	0.5830	0.6400
		frameDifference-resnet (Yao et al., 2017)	0.5029	0.5653	0.5900

Table 7: Cross-dataset evaluations (Training set: Dev1+Dev2)

Training	Testing	Filter-DCNN	Accuracy	MAP	MP@20
Dev1 + Dev2	FVC	cobalt-gnet	0.5235	0.5178	0.5400
		cobalt-resnet	0.5029	0.4807	0.4700
		q4-gnet	**0.6294**	**0.7017**	**0.7200**
		q4-resnet	0.6000	0.6129	0.6400
		rawKeyframes-gnet	0.6029	0.5694	0.5300
		rawKeyframes-resnet	0.5441	0.5115	0.5200
		highPass-gnet	0.5147	0.5194	0.5300
		highPass-resnet	0.5294	0.6064	0.7000
		frameDifference-gnet	0.5176	0.5330	0.5500
		frameDifference-resnet	0.4824	0.5558	0.5400

0.5. One major exception, and the most notable finding in our investigation, is the performance of the Q4 filter when used to train a GoogLeNet model. In this case, the performance is significantly higher than in any other case, and remains promising with respect to the potential of real-world application. Being able to generalize into new data with unknown feature distributions is the most important feature in this respect, since it is very unlikely at this stage that we will be able to create a large-scale training dataset to model any real world case.

Trained on Dev1+Dev2, the Q4 filter combined with GoogLeNet yields a MAP of 0.702. This is a promising result and significantly higher than all competing alternatives. Still, however, it is not sufficient for direct real-world application, and further refinement would be required to improve this.

The aim of these experiments was to evaluate the extent in which we could automatize the process of analyzing the filter outputs using state-of-the-art algorithms. By observing the results, the conclusion was that, while alternative features performed better in within-dataset evaluations, the InVID filters were more successful in realistic cross-dataset evaluations, which are the most relevant in assessing the potential for real-world application.

Still, the performance is not yet sufficiently high for market application, and more effort is required to reach the desired accuracy. One major issue is the lack of accurate temporal annotations for the datasets. By assigning the “tampered” label on all frames of tampered videos, we are ignoring the fact that tampered videos may also contain frames without tampering, and as a result the labelling is inaccurate. This may be resulting in noisy training, which may be a cause of reduced performance. Furthermore, given the per-frame classification outputs, currently we calculate the per-video score by

comparing the average “tampered” score with the average “untampered” score. This approach may not be optimal, and different ways of transitioning from per-frame to per-video scores.

Currently, given the evaluation results, we cannot claim that we are ready for real-world application, nor that we have exhaustively evaluated the proposed automatic detection algorithm. In order to improve the performance of the algorithm and run more extensive evaluations, we intend to improve the temporal annotations of the provided datasets and continue collecting real-world cases to create a larger-scale evaluation benchmark. Finally, given that the current voting scheme may not be optimal, we will explore more alternatives in the hope of improving the algorithm performance. Furthermore, we should extend our investigations into more filters and CNN models, in order to improve performance, including the possibility of using feature fusion by combining the outputs of multiple filters in order to assess each video.

3.4 API layer and integration with InVID

[Content removed as confidential.]

4 Near-duplicate Detection

During the third year of the project, we dedicated effort on the composition of an evaluation dataset that provides a realistic representation of the Near-Duplicate Video Retrieval (NDVR) problem and extends it to the more general problem of Fine-grained Video Retrieval (FIVR). Additionally, we focused on the improvement of the retrieval performance of the developed approach, further surpassing the current state of the art in accuracy. We achieved this by casting the proposed Deep Metric Learning approach to a frame-level method by employing Chamfer Distance to exploit distances between video frames during video similarity calculation. Additionally, we also improved the NDD component through integration of all contributions into the InVID platform, providing new functionalities, and fixing problems or bugs that were identified during the project test cycles.

4.1 State of the art

Since D3.2, several works related to NDD have been proposed in the relevant literature. Wu et al. (Y. Xu, Monrose, Frahm, et al., 2017) developed a copy detection method based on derivative feature extraction computed as the temporal gradient of the video frames average intensity signal. For efficient indexing and retrieval, they employed a K-d tree structure (Bentley, 1975), and trained a SVM (Cortes & Vapnik, 1995) to recognize near-duplicate video sequence pair. Wang et al. (L. Wang, Bao, Li, Fan, & Luo, 2017) proposed a compact video representation based on CNN features combined with Sparse Coding (SC) (Coates & Ng, 2011) for video copy detection. They first extract CNN features from the video frames, encode them into a fixed length vector via the SC method, and generate video representations by applying max-pooling on each component of the frame vectors. In (Liu et al., 2018), the authors proposed a fast video searching strategy based on inverted file indexing. They extracted frame fingerprints from a hashing process which are stored in an inverted file structure, and devised a video retrieval process which involves table look-up and word counting operations for efficient fingerprint matching. The authors in (Guzman-Zavaleta & Feregrino-Uribe, 2018) proposed an adaptive decision strategy based on reinforcement learning. They first extracted two low-cost global descriptors based on the spatial information and the temporal variances of video sequences and then employed the Q-learning (Watkins & Dayan, 1992) algorithm to learn the optimal policy for the decision of the near-duplicate video segments. Finally, Baraldi et al. (Baraldi, Douze, Cucchiara, & Jégou, 2018) proposed a temporal layer in a deep network that calculates the temporal alignment between videos by maximizing a time-sensitive similarity metric in the Fourier domain. They trained the network minimizing a triplet loss that takes into account both the localization accuracy and the recognition rate.

The method developed in InVID and described in this deliverable was designed with different goals compared to the above methods, most of which focus on the problem of partial duplicate video detection and localization (i.e. identify a particular video segment that matches a segment of the query video). Although solving this problem could be valuable in the context of InVID, we found that such approaches suffer from two weaknesses: a) the definition of near-duplicity (either at the level of a video or at the level of a video segment) is very rigid, which typically results in only a small subset of almost identical videos being retrieved; b) methods for partial duplicate video retrieval typically suffer from high response times and big computational requirements. Of the above approaches, the one by Baraldi et al. bears considerable similarities with the one developed in InVID, i.e. video similarity calculation is based on a trained neural network model, which allows for a flexible definition of near-duplicity. This provides support for the validity of the InVID approach, which was also designed to overcome the computational challenges of the problem at hand, i.e. combine the speed of video-level matching methods with the accuracy of frame-level matching methods[4].

Additionally, we reviewed several relevant video datasets from the literature, since considerable effort during the final project year has been expended towards the construction of a video dataset that covers the needs of the FIVR problem setting. The most popular and publicly available dataset related to the NDVR problem is the CC_WEB_VIDEO (X. Wu, Hauptmann, & Ngo, 2007). It has been published by the research groups of City University of Hong Kong and Carnegie Mellon University. The dataset consists of 13,129 generated videos collected from the Internet. For the dataset collection, a total number of 24 popular text queries were submitted to popular video platforms, such as YouTube, Google Video, and Yahoo! Video. A set of videos were collected for each query and the video with the most views was

[4]Note that it was not possible to perform a systematic experimental comparison of the InVID approach with the methods discussed above, given that they were only recently published.

selected as the query video. Then, videos in the collected sets were manually annotated based on their relation to the query video.

Several variations of the CC_WEB_VIDEO dataset have been developed by researchers in the NDVR fields (Song, Yang, Huang, Shen, & Hong, 2011; Cai et al., 2011; Chou, Chen, & Lee, 2015). In order to make the NDVR problem more challenging and benchmark the scalability of their approaches, researchers usually extend the core CC_WEB_VIDEO dataset with many thousands of distractor videos. The most well-known public dataset that was created through this process is UQ_VIDEO (Song et al., 2011). The combined dataset contains 169,952 videos (including those of the CC_WEB_VIDEO) in total with 3,305,525 keyframes and the same 24 query videos as the ones accompanying the CC_WEB_VIDEO dataset.

Other popular public benchmark datasets are the Muscle-VCD dataset (Law-To, Joly, & Boujemaa, 2007), and the TRECVID dataset (Kraaij & Awad, 2011) developed for the video copy detection problem. The first one consists of 18 videos of 100 hours and the second one includes 11,503 reference videos of over 420 hours, respectively. For both datasets, a number of transformations were simulated by using video-editing software in order to generate synthetic video queries. The generated queries are used in order to detect the the original versions of the video in the dataset and determine the copied segment.

A more recent dataset that is relevant to our problem is the VCDB (Jia et al., 2014). This dataset is composed of videos derived from popular video platforms (i.e. YouTube and Metacafe) and has been compiled and annotated as a benchmark for the partial copy detection problem, which is highly related to the NDVR problem. VCDB contains two subsets, the core and the distractor subset. The core subset contains 28 discrete sets of videos composed of 528 query videos and over 9,000 pairs of partial copies. Each video set was manually annotated by seven annotators and the video chunks of the video copies were extracted. The distractor subset is a corpus of approximately 100,000 distractor videos that is used to make the video copy detection problem more challenging.

Although all the aforementioned video collections capture different aspects of the NDVR problem, all of them are limited in different ways, e.g. small size, no user-generated videos, high dissimilarity between distractor videos and queries, etc. To this end, we composed a large video dataset, namely FIVR-200K, that covers the evaluation needs of NDVR and extends its scope to the challenge of fine-grained video retrieval (such as detecting videos from the same event but from different viewpoints). The dataset consists of 225,960 videos derived from numerous real-world events, hence including a wide variety of videos and many distractor videos that render the near-duplicate video retrieval task very challenging.

4.2 Method description

In D3.2, we presented a video-level NDVR scheme based on Deep Metric Learning (DML) (Kordopatis-Zilos, Papadopoulos, Patras, & Kompatsiaris, 2017b). Its major drawback was that all frame features are combined into a single video descriptor by averaging all frame feature vectors to a single video-level vector. Consequently, the generated video representation lacked fine-grained video information coming from individual frames. To overcome this issue and improve performance, we extended the DML approach by incorporating frame-level matching between two compared videos. To this end, the video representation is now composed of all frame descriptors instead of their average. More specifically, the distance between two compared videos derives from the calculation of the distance between all frames of the two compared videos. This video representation helps to preserve the local information of the video content which leads to more accurate comparison between videos and facilitates the needs of Partial-Duplicate Video Retrieval (PDVR).

We build upon the scheme described in D3.2 in order to train the proposed DML model. The network architecture and the triplet generation method remain the same. We also use the feature extraction process described in D3.1. Since our goal is to incorporate frame-level matching in our approach, the similarity (or equivalently distance) between all pairs of frames of the two compared videos need to be calculated. To do so, we employ the Euclidean Distance Matrix (EDM), a table of all pairwise square-distances between the two sets of frames.

In particular, we consider two arbitrary videos q and p with sets of frame descriptors $\mathbf{q} = [\mathbf{q}_1, ..., \mathbf{q}_N] \in \mathbb{R}^{k \times N}$ and $\mathbf{p} = [\mathbf{p}_1, ..., \mathbf{p}_M] \in \mathbb{R}^{k \times M}$, where N, M are the total number of keyframes for video q, p respectively, and k the dimensionality of the feature vectors. All frame descriptors are provided to the DNN network to compute their feature embeddings; thus, the video representations are transformed to $f_\theta(\mathbf{q}) = [f_\theta(\mathbf{q}_1), ..., f_\theta(\mathbf{q}_N)] \in \mathbb{R}^{d \times N}$ and $f_\theta(\mathbf{p}) = [f_\theta(\mathbf{p}_1), ..., f_\theta(\mathbf{p}_M)] \in \mathbb{R}^{d \times M}$, where d is the dimensionality of the feature embedding. For convenience the transformed vectors are going to be indicated with the

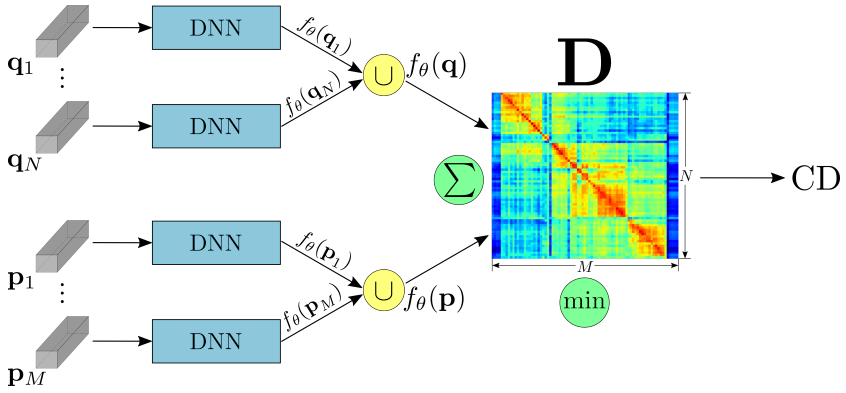


Figure 3: Illustration of the proposed process for the the calculation of the Chamfer Distance between two arbitrary videos.

subscript letter θ instead of the function $f_\theta(\cdot)$, for example $\mathbf{q}_\theta$. To construct the EDM, we have to calculate all possible pairwise distances between the feature embeddings $\mathbf{q}_{\theta,i}$ and $\mathbf{p}_{\theta,j}$, $i \in [1,N], j \in [1,M]$. Equation 1 illustrates the composition of the EDM.

$$\mathbf{D}(\mathbf{q}_\theta, \mathbf{p}_\theta) = \begin{bmatrix} d_{11} & d_{12} & d_{13} & \dots & d_{1M} \\ d_{21} & d_{22} & d_{23} & \dots & d_{2M} \\ d_{31} & d_{32} & d_{33} & \dots & d_{3M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & d_{N3} & \dots & d_{NM} \end{bmatrix} \tag{1}$$

where, $\mathbf{D}(x,y) \in \mathbb{R}^{N \times M}$ is the EDM between two sets x, y, and d_{ij} is the distance between the embeddings $\mathbf{q}_{\theta,i}$ and $\mathbf{p}_{\theta,j}$. The pairwise distances are calculated based on the Equation 2.

$$d_{ij} = \left\| \mathbf{q}_{\theta,i} - \mathbf{p}_{\theta,j} \right\|^2 \tag{2}$$

Expanding the norm yields

$$d_{ij} = \left\| \mathbf{q}_{\theta,i} - \mathbf{p}_{\theta,j} \right\|^2 = (\mathbf{q}_{\theta,i} - \mathbf{p}_{\theta,j})^\top (\mathbf{q}_{\theta,i} - \mathbf{p}_{\theta,j}) = \mathbf{q}_{\theta,i}^\top \mathbf{q}_{\theta,i} - 2\mathbf{q}_{\theta,i}^\top \mathbf{p}_{\theta,j} + \mathbf{p}_{\theta,j}^\top \mathbf{p}_{\theta,j} \tag{3}$$

Similarly, the EDM can be calculated using linear algebra instead of calculating the distance between all frame embeddings exhaustively. The computation of EDM is provided in Equation 4.

$$\mathbf{D}(\mathbf{q}_\theta, \mathbf{p}_\theta) = \text{diag}(\mathbf{q}_\theta^\top \mathbf{q}_\theta)\, \mathbf{1}_M^\top - 2\mathbf{q}_\theta^\top \mathbf{p}_\theta + \mathbf{1}_N\, \text{diag}(\mathbf{p}_\theta^\top \mathbf{p}_\theta)^\top \tag{4}$$

where, $\mathbf{1}_K$ denotes a column vector of all ones with size K and diag($\mathbf{A}$) is a column vector of the diagonal entries of matrix $\mathbf{A}$. To compute a single value as the distance between two videos, we employ Chamfer Distance (CD) (Barrow, Tenenbaum, Bolles, & Wolf, 1977). To this end, considering that $\mathbf{q}$ is the query and $\mathbf{p}$ is a candidate video set, we get the distance of the closest candidate frame in the embedding feature space for each one of the query frames. Finally, the average of the selected distances is the final video distance. The CD is formulated in Equation 5.

$$\text{CD}(q_\theta, p_\theta) = \frac{1}{N} \sum_{i=1}^{N} \min_{j \in [1,M]} \mathbf{D}(\mathbf{q}_{\theta,i}, \mathbf{p}_{\theta,j}) \tag{5}$$

Figure 3 summarizes the entire process for the CD calculation between two videos. The function applied on the EDM is depicted inside green circles and along the corresponding axis. The yellow circles represent the concatenation of all frame embedding into one video set. The triplet loss function can be rewritten as in Equation 6.

$$\mathscr{L}(v, v^+, v^-) = \max\{0, \text{CD}(\mathbf{v}_\theta, \mathbf{v}_\theta^+) - \text{CD}(\mathbf{v}_\theta, \mathbf{v}_\theta^-) + \gamma\} \tag{6}$$

where, v, v^+, v^- are the query, positive (NDV), and negative (dissimilar) videos of an arbitrary triplet, accordingly. To calculate the final video similarities, we employ the same scheme described in D3.2 for the conversion of video distance to video similarity.

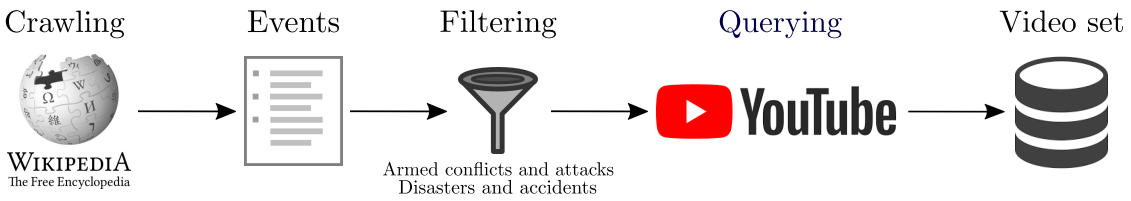


Figure 4: Overview of the video collection process.

During Year 3, we also made publicly available the code for the feature extraction step[5] and the Deep Metric Learning Near-Duplicate Video Retrieval algorithms[6].

4.3 FIVR-200K dataset

The FIVR-200K dataset (Kordopatis-Zilos, Papadopoulos, Patras, & Kompatsiaris, 2018) was designed with the following goals in mind: a) the videos should be associated with a large number of events, b) the categories of these events should be the same, and c) the dataset size needs to be sufficiently large to make retrieval of relevant results challenging.

To begin with, we define the following categories of related videos: a) Duplicate Scene Videos (DSV): these are videos that share at least one scene (captured by the same camera) regardless of any applied transformation. A special case of this category is Near-Duplicate Videos (NDVs), i.e. videos that have all scenes in common. b) Complementary Scene Videos (CSV): these are videos that contain part of the same spatio-temporal segment, but are captured from different viewpoints. c) Incident Scene Videos (ISV): these are videos that capture the same incident, i.e. they are spatially and temporally close, but have no temporal overlap.

For the dataset collection, we set up the process depicted in Figure 4 to retrieve videos about major events that took place during the recent years. First, we crawled Wikipedia's 'Current Event' page[7] to build a collection of the major events since the beginning of 2013. Each event is associated with a topic, headline, text, date, and hyperlinks. For the remaining steps of the process, we retain only events categorised as 'Armed conflicts and attacks' or 'Disasters and accidents'. We selected these two categories in order to find multiple videos on YouTube that report on similar events, so that they would bear relatively high visual similarity with each other (due to common depicted themes), with the ultimate goal of creating a challenging retrieval task. The time interval used for the crawling of the events was January 1st 2013 to December 31st 2017. A total of 9,431 events were collected, and 4,687 events were retained after filtering. In the next step, the public YouTube API[8] was used to collect videos by providing event headlines as queries. The results are filtered to contain only videos published at the corresponding event start date and up to one week after it. Furthermore, they are filtered to contain only videos with duration up to five minutes. This resulted in the collection of 225,960 videos ($\sim$48 videos/event).

Selecting "appropriate" queries is important for ensuring that the resulting annotations and evaluation protocol that accompany the dataset will be representative of and commensurate to the challenges arising in real-world problem settings. To this end, the query selection process was designed with two goals in mind: a) to generate challenging queries, i.e. queries that will lead to several distractor videos that will likely challenge content-based retrieval systems, and b) to find query videos that will lead to the retrieval of videos with various modifications that will not only be trivial NDV cases, but also contain interesting variations (e.g. different viewpoints of the same scene), i.e. CSV and ISV. To achieve those two goals, we implemented a largely automatic process that combines visual and textual video similarity.

First, the visual similarity between videos was computed based on the developed NDVR method described in D3.2. Second, the textual similarity between two videos was computed as the cosine similarity between the tf-idf representations of the words in their titles. To perform the similarity calculation, we first pre-processed video titles with the NLTK toolkit (Bird & Loper, 2004), applying part-of-speech (PoS) tagging, removing all verbs (which we found to introduce unnecessary noise) and providing the results to the NLTK WordNet-based lemmatizer to extract the lemmas, which constitute the word-based representation of the titles. The overall video similarity derives from the average of the visual and textual

[5] https://github.com/MKLab-ITI/intermediate-cnn-features
[6] https://github.com/MKLab-ITI/ndvr-dml
[7] https://en.wikipedia.org/wiki/Portal:Current_events
[8] https://developers.google.com/youtube/

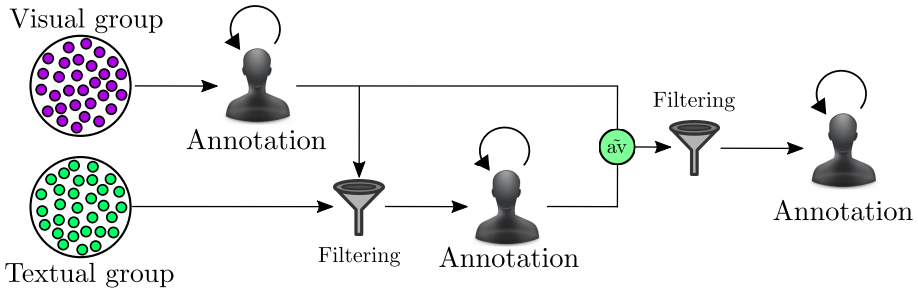


Figure 5: Overview of the annotation process. Two groups of videos are involved, derived based on their visual and textual similarity to the query. Three annotation phases take place and two filtering steps are applied. $\bar{av}$ stands for the average of visual and textual similarity scores between the query and each video in the visual or textual group.

similarity. Tf-idf was selected as a representation for both visual and text words because of its sparsity, which was practical for fast similarity calculation and efficient dataset annotation.

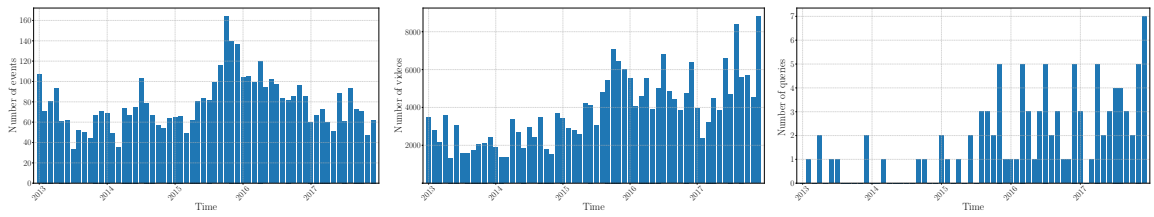
In the next step, we computed non-zero similarities between video pairs. Only video pairs that share at least one visual or text word were considered, which resulted in a complexity much lower than $O(n^2)$. Afterwards, we created a video graph G by connecting with an edge video pairs with similarity greater than a certain threshold t_s (empirically set to 0.7). To identify meaningful video groups, we extracted the connected components C of the video graph G with more than two videos. Then, we defined the uploader ratio r_c of each component $c \in C$ using Equation 7.

$$r_c = \frac{|\{u_v | v \in c, u_v \in U\}|}{N_c} \tag{7}$$

where the numerator is the number of unique uploaders in the component, v is a video in the component, u_v is the uploader of video v, U is the set of uploaders in the dataset, and N_c is the number of videos in the component. We have empirically found that components with low uploader ratio usually contain videos from a single specific channel (e.g. news channel) with titles that are very similar (e.g. exactly the same title with different date) or with content that is visually highly similar (e.g. the same presenter reporting news in the same background). However, based on our definition, such videos are neither considered DSV nor CSV or ISV. For that reason, we discard components with uploader ratio less than a threshold t_r (empirically set to 0.75).

Since we need components consisting of videos that refer to the same incident, we applied another criterion on the component set based on the publication date of their videos, and retained only components consisting of videos that were published within a time window of two weeks. Our goal is to find queries that will lead to result sets with many DSV, CSV and ISV. Intuitively, large components with many (visually and textually) similar videos have better chance of containing such videos. For that reason, we rank connected components based on their size and select one query video per component. We consider that short videos with few shots to be the most suitable candidates for having been modified and reposted several times (both as single videos or as part of mash-ups). Therefore, we select videos with duration less than a threshold t_d (empirically set to 90 seconds). Trying to find the original version of videos in each cluster, we choose as query the video that was published earliest. The total number of resulting queries using the above process was 635. Since it would be overly time consuming to annotate all of them, we selected the top 100 as the final query set (ranked based on the size of the corresponding graph component).

Figure 5 depicts the entire annotation process, which is carried out in three steps. Given a query, two groups of videos are retrieved, one based on visual similarity and one based on textual similarity. In the first step, we annotate the videos contained in the “visual” group. The end of the first step occurs when a total number of 100 irrelevant videos have been annotated after the last relevant result (i.e. annotated as NDV, DSV, CSV or ISV). In the second step, videos in the “textual” group that have been already annotated as part of the visual group are removed. The annotation process continues with the remaining videos in the textual group. Similarly, this step ends either when a total number of 100 irrelevant videos have been annotated after the last relevant one or after the first 1000 videos have been annotated (whatever of the two criteria applies first). To minimize the possibility of having missed relevant videos, in the third and final step, the remaining videos of the two groups are merged and filtered based on their publication date. We retain only videos that have been published within a time window



(a) events (b) videos (c) queries

Figure 6: Monthly distribution of a) events, b) videos and c) queries.

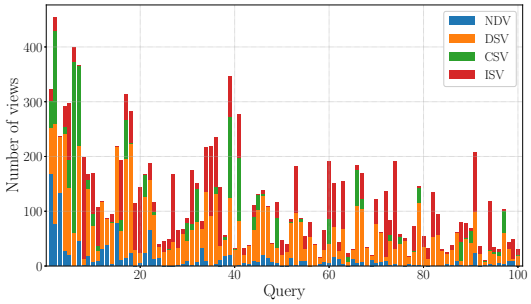


Figure 7: Distribution of annotation labels per query (best viewed in colour).

of a week before and after the publication date of the query[9]. The remaining videos are ranked based on the average visual-textual similarity and the annotation process proceeds until either 200 irrelevant videos have been found after the last relevant video, or there are no videos left in the merged group.

The annotation labels and corresponding definitions, which were used by the annotators, are the following: a) **Near-Duplicate (ND)**: These are a special case of DSVs, b) **Duplicate Scene (DS)**: DSVs are annotated with this label. c) **Complementary Scene (CS)**: CSVs are annotated with this label, d) **Incident Scene (IS)**: ISVs are annotated with this label, d) **Distractors (DI)**: videos that do not fall in any of the above cases are annotated as distractors. For the annotation of the dataset, the extracted queries were split in two parts, each assigned to a different annotator. After the end of the annotation process, all annotated videos (excluding the videos labeled with DI) were revisited and tested for their consistency to the definitions.

In total, the dataset comprises 225,960 videos associated with 4,687 Wikipedia events. Figure 6 illustrates the monthly distribution of the collected events, videos and queries. There is a noteworthy peak of events during the last quarter of 2015. During that period, major wars (e.g. the Syrian civil war, the war in Afghanistan, the Yemeni civil war) and a number of devastating natural disasters (e.g. hurricane Joaquin, Hindu Kush earthquake and an intense Pacific typhoon season) took place leading to daily newsworthy incidents. From the temporal video distribution, one may notice an increase in video sharing in the last two years which does not correspond to the trend in the timeline of major events. A possible explanation may be the increasing trend in video capturing and sharing on YouTube. Finally, it is noteworthy that the temporal distribution of queries approximately follows the one of videos over time with more query videos published during the last two years of the dataset. This confirms that the employed query selection process does not introduce any temporal bias.

Regarding the annotation labels, we found that the selected queries have on average 13 NDV, 57 DSV, 18 CSV and 35 ISV. Figure 7 illustrates the distribution of annotation labels per query. The queries are ranked based on the size of the cluster they are associated with. As expected, there is considerable correlation (Pearson correlation=0.62) between the cluster size and number of videos that have been annotated with one of the four relevant labels. For all 100 queries, the total number of unique videos that were annotated (including DIs) is about 140 thousands. Some videos have been annotated multiple times, because they have different labels for different queries.

[9]In this annotation step, we consider videos published up until one week before the query video, because of some rare cases that one or more related videos were not included in the component of the query video.

4.4 Progress and evaluations during Year 3

To evaluate the proposed approach we use the same experimental setup described in D3.2. The VCDB dataset (Jiang, Jiang, & Wang, 2014) is employed as the development set exclusively, which is exploited to generate video triplets and to train the network. The results of the proposed approach are obtained on various setups of CC_WEB_VIDEO dataset (X. Wu et al., 2007) and compared with the previous methods as well as five approaches from the state-of-the-art. In addition, we use the FIVR-200K dataset (Kordopatis-Zilos et al., 2018) for validating the results on a second independent dataset. We have devised three different tasks on FIVR-200K: a) the Duplicate Scene Video Retrieval (DSVR) task where only videos annotated with ND and DS are considered relevant, b) the Complementary Scene Video Retrieval (CSVR) task which accepts only the videos annotated with ND, DS or CS as relevant, and c) Incident Scene Video Retrieval (ISVR) task where all labels (with the exception of DI) are considered relevant. The proposed approach is benchmarked against the developed methods and the five competing approaches from state-of-the-art that were described in the past deliverables.

We study the performance of the proposed DML approach with frame-level matching in two variants of CC_WEB_VIDEO dataset, and in relation to the underlying CNN architecture. We experiment with AlexNet and GoogleNet. For each of them, four configurations are tested: i) **CNN** (baseline): average all frame descriptors to a single vector and use it for retrieval without any transformation, ii) **DML**: is the vanilla DML approach with late fusion as presented in D3.2, iii) **CNN-CD**: combine the frame descriptors without any transformation with the proposed CD scheme, iv) **DML-CD**: apply the learned embedding function to every frame descriptor and then apply the proposed CD scheme.

	CC_WEB_VIDEO		**CC_WEB_VIDEO***	
Method	**AlexNet**	**GoogleNet**	**AlexNet**	**GoogleNet**
CNN	0.948	0.952	0.887	0.898
DML	0.964	0.969	0.922	0.934
CNN-CD	0.976	0.977	0.957	0.962
DML-CD	0.979	**0.981**	0.960	**0.964**

Table 8: mAP of every CNN architecture with the four system setups.

Table 8 illustrates the mean Average Precision (mAP) of the two CNN architectures with the four system setups. It is evident that the application of the proposed CD scheme has considerable impact on the performance of the model. In both cases, DML and CNN, the improvement from the incorporation of the CD scheme ranges from 0.012 to 0.07 in terms of mAP. GoogleNet achieves better results for all four settings with considerable margin, with mAP scores of 0.981 and 964 on CC_WEB_VIDEO and CC_WEB_VIDEO* respectively. Furthermore, the DML approach consistently outperforms the pre-trained CNN features in every system setup, which indicates that the similarity learning process benefits the overall component performance.

To test the full potential of the proposed approach, we trained both DML variants with end-to-end training. Instead of keeping the CNN network fixed during training, we trained the CNN network as well by backpropagating the DML error to the convolutional layers. However, due to memory insufficiency, we have tested only the AlexNet architecture. For the utilization of bigger and more accurate networks (e.g. ResNet, Inception, VGG), we have to redesigned the training process. Hence, for the sake of comparison, four training configurations are benchmarked: i) **DML**: the vanilla DML approach with late fusion and fixed network during training, ii) **DML**$_{e2e}$: the end-to-end version of the vanilla DML, iii) **DML-CD**: the DML approach with the proposed CD extension and fixed network during training, iv) **DML-CD**: the end-to-end version of the DML approach with the proposed CD extension.

Run	CC_WEB_VIDEO	CC_WEB_VIDEO*
DML	0.964	0.922
DML$_{e2e}$	0.971	0.948
DML-CD	0.979	0.960
DML$_{e2e}$**-CD**	**0.980**	**0.965**

Table 9: mAP comparison of four different training variants of the DML methods (with AlexNet).

Table 9 presents the mAP of the four different training configurations of the DML method. Regarding the proposed CD scheme, the improvement from the end-to-end training is marginal. For

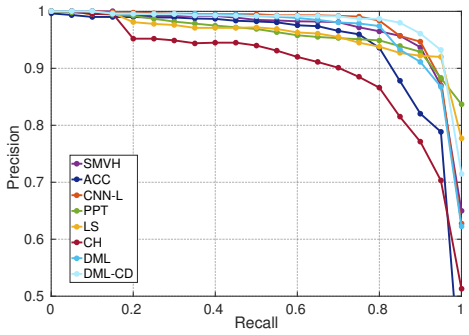


Figure 8: Precision-Recall curve comparison of the proposed DML approach and existing approaches.

CC_WEB_VIDEO* the improvement is 0.005, whereas for CC_WEB_VIDEO it is only 0.001. The only case where end-to-end training has a clear impact is for the vanilla DML on CC_WEB_VIDEO*, where the mAP increased from 0.922 to 0.948.

Furthermore, for comparing the performance of our approach with the developed NDVR approaches and five from the literature, we select the setup using GoogleNet pre-trained features denoted as DML-CD, since it achieved the best results. The compared methods are: **CNN-L** (Kordopatis-Zilos, Papadopoulos, Patras, & Kompatsiaris, 2017a), **DML** (Kordopatis-Zilos et al., 2017b), **CH** & **LS** (X. Wu et al., 2007), **ACC** (Cai et al., 2011), **SMVH** (Hao et al., 2017) and **PPT** (Chou et al., 2015) .Table 10 presents the mAP scores of the competing methods. Our approach outperforms all methods with a clear margin. The same result derived from the comparison of the PR curves illustrated in Figure 8, with the cyan line (proposed approach) clearly lying upon all others up to 95% recall. It is noteworthy that our approach is trained on the VCDB dataset and does not have any knowledge from the evaluation dataset, yet it achieves the best results among all other state-of-the-art approaches with a significant margin.

Method	**CH**	**ACC**	**LS**	**SMVH**	**PPT**	**CNN-L**	**DML**	**DML-CD**
mAP	0.892	0.944	0.952	0.971	0.958	0.974	0.969	**0.981**

Table 10: mAP comparison between the proposed DML-CD approach and existing approaches.

In addition, the developed approaches are also benchmarked on the FIVR-200K dataset. As it has already been described, it includes three tasks that accept different type of video results as relevant. The performance of the compared approaches is quantified based on the mean Average Precision, and their scalability based on the execution time per query. We compare the previously deployed in the NDD service approaches. In particularly, we compiled the following runs: **BoW-L** (Kordopatis-Zilos et al., 2017a) that is a layer-based Bag-of-Word (BoW) scheme (D3.1), **DML+BoW** that uses DML (Kordopatis-Zilos et al., 2017b) for feature extraction and BoW for aggregation (D3.2), and **DML-CD** which is the currently proposed approach. Due to the high execution time of the DML-CD method, we implemented a combination of DML+BoW and DML-CD, since both of them incorporate DML features. The former method is used in order to initially calculate video similarity, and then rerank the videos that surpass a given threshold based on the latter method. This hybrid method is denoted as **H-DML** and two similarity thresholds are tested, i.e. 0.4 and 0.1. All runs were implemented with frame features derived from the VGGNet and the underlying visual vocabularies built with videos sampled from FIVR-200K dataset.

task	**DSVR**	**CSVR**	**ISVR**	**Time**
BoW-L	0.681	0.641	0.573	2.92s
DML+BoW	0.650	0.623	0.533	1.26s
DML-CD	**0.744**	**0.715**	**0.613**	292.5s
H-DML$_{0.4}$	0.711	0.672	0.549	1.33s
H-DML$_{0.1}$	0.736	0.695	0.580	1.69s

Table 11: mAP and execution time of the benchmarked runs on the FIVR-200K dataset.

Table 11 summarizes the performance of the benchmarked runs on the FIVR-200K dataset. It is evident that the DML-CD approach achieves noticeably better performance in comparison to the other

runs for all experimental tasks. For the DSVR task which is equivalent to the NDVR problem, it achieves 0.744 mAP, and outperforms all other runs by a large margin. The performance of all runs marginally drops for the CSVR task in comparison to DSVR with a reduction of approximately 0.04 in terms of mAP. Moving to the ISVR task, all runs exhibit a considerable drop in their performance, with the DML-CD achieving the best performance with 0.601 mAP. However, the superior accuracy of DML-CD comes at a huge cost in terms of execution time. With 292.5s per query, the DML-CD is impractical. The fastest run is DML+BoW with 1.26s per query, but its retrieval performance is limited. The runs that strike good trade-off between accuracy and speed are the hybrid ones (H-DML). In order to keep response time as low as possible (following feedback from the evaluation test cycles that gave importance to response time), we chose to use the H-DML$_{0.4}$ variant in the NDD service, which achieves a 0.711 mAP in the DSVR task at 1.33s average response time.

The above experiments made clear that the FIVR-200K dataset offers a much more challenging and realistic benchmark compared to the CC_WEB_VIDEO dataset, which has been used so far in the literature. This means that this dataset can serve as a valuable benchmark for future content-based video retrieval methods.

4.5 API layer and integration with InVID

During the final year of the project, the API layer was upgraded and its functionalities were extended. The service was updated with the implementation of the deployed approach and the core functionalities of the service (i.e. index and search) was implemented in a distributed fashion. Also, the video index was significantly extended counting approximately 1.2M videos from various video platforms (compared to 400K reported in D3.2). To fully comply with the ToS of the source video platforms, we implemented a video deletion module, which periodically (every week) checks whether the videos of the index are still available in the source platforms. If a video is found to have been removed, then it is also removed from the InVID video index. On average, this module removes 0.7% of the indexed videos on each periodic check.

In terms of the NDD service calls, several functionalities were added. The calls and their parameters are displayed in Table 4.5. Compared to the version reported in D3.2, two API calls have been added to the NDD service, i.e. the `/partial` and `/youtube` call.

The `/partial` call performs retrieval of near-duplicate shots in the indexed videos. More precisely, all videos in the database have been segmented in non-overlapping video shots based on the sequence of their visual words. Given a query video, the NDD service performs search for each extracted video shot individually, in order to retrieve near-duplicate shots from the videos in the video index. For each query video shot, the candidate video shots are ranked in descending order and precise information about them are returned, i.e. the start and end of the candidate video shot, information about the container video, the shot similarity between the query shot and candidate shot, and the rank of the candidate shot in the returned list.

The `/youtube` call collects videos from YouTube based on the provided arguments (text query or query video) and adds them to the video index. In particular, the input arguments to the NDD service are either a text query or the ID of a YouTube video. Then, the service queries the YouTube API based on the given input and collects a number of videos. The returned videos are processed by the service and are finally added to the video index. The response of the NDD service includes the list of YouTube videos that have been added to the video index.

Additionally, the two search calls (`/search` and `/partial`) support video retrieval based on a specific video segment, specified by the start and the end of the video segment (in seconds). Particularly, the `start` and `end` arguments have to be provided, with the value of the latter argument been greater than the former one and both have to be greater than zero. Otherwise, the NDD service returns an error message that indicates the cause of error.

The NDD module consists of three major components: the main service, the feature extractor and the video searcher. The architecture of the system is illustrated in Figure 9. For improved stability, the main service is responsible for handling the API requests, scheduling the necessary processes, communicating with all service components, and storing all necessary video metadata are stored to the NDD MongoDB. The feature extractor and the video searcher components have been designed and implemented to facilitate distributed computing, and both of them provide REST APIs. More than one feature extractors or video searchers can be connected to the main service, possibly deployed on different machines. The feature extractor is responsible for downloading the requested video, and extracting visual feature descriptors from each video frame. This component needs to be deployed

Service	Request	URL	Parameter
index video	GET	`/index`	`url=`<video url> `async=`<true or false> `force=`<true or false>
search video	GET	`/search`	`url=`<video url> `t_sim=`<similarity threshold> `t_rank=`<rank threshold> `start=`<segment start> `end=`<segment end>
search video shots	GET	`/partial`	`url=`<video url> `v_sim=`<video sim. threshold> `s_sim=`<shot sim. threshold> `start=`<segment start> `end=`<segment end>
collect & index Youtube videos	GET	`/youtube`	`video_id=`<Youtube video id > `text=`<text query> `max=`<max added videos>
delete video	DELETE	`/delete`	`url=`<video url>

Table 12: Calls exposed by the Near-Duplicate Detection module API.

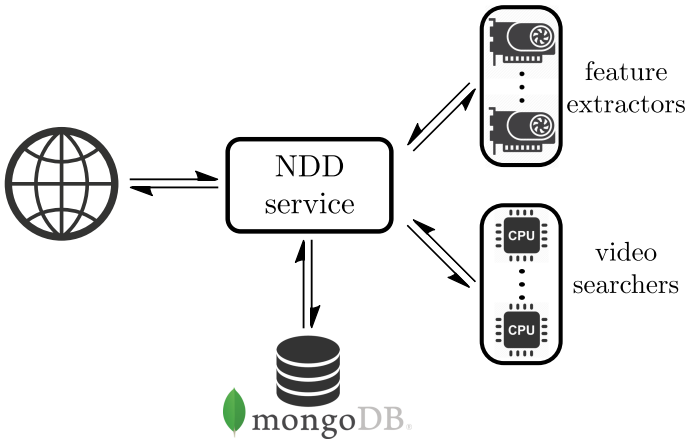


Figure 9: Architecture of the NDD service.

on machines that are equipped with high-end GPUs for optimal performance. The video searcher is responsible for calculation of video or shot similarities between the query and the candidate videos, and is heavily parallelized. All individual components have been dockerized to be OS agnostic and easy to deploy in any machine. However, the feature extractor requires the nvidia-Docker for accessing GPU resources within the Docker environment and is currently only supported in the Linux OS.

Finally, we have set up a separate instance of the NDD service in a different endpoint with the goal of supporting the needs of the CAA service for querying against the FVC-2018 collection. More information about the composition of the video index and the use of the NDD instance is presented in Section 7.

5 Logo Detection

The logo detection component started as a keypoint-based approach in D3.1, and was replaced by a deep learning-based approach in D3.2, which provided greater scalability and extensibility, significant speed improvements at comparable detection accuracy, and potential for better performance even against an increasing number of known logos. This year, we further experimented with the network and training algorithm, ported to a different framework to improve maintenance, and experimented with various aspects of the training process to improve the component performance.

5.1 State of the art

The task that we defined in D3.2 as "TV logo detection" has not attracted further significant attention from the research community in the recent years, and thus the state of the art remains essentially the same as it was described in D3.2. It is interesting to note that, the approach we proposed in D3.2 for training a deep model using synthetic examples (which we create by overlaying the logo templates over generic images) instead of using manually annotated videos, has been independently proposed by different authors in other instances during the same time period (H. Su, Zhu, & Gong, 2017; Montserrat, Lin, Allebach, & Delp, 2018). This is an encouraging observation, as it demonstrates that the InVID approach remained on par with the state of the art with respect to this task.

5.2 Method description

During the third year of InVID, the core of the method remained the same as the year before, and our work focused on improving the performance of the model while extending the coverage of the dataset with new logos submitted by the users.

Similar to what was described in D3.2, the method design was based on an object detection method, namely Faster-RCNN (Ren, He, Girshick, & Sun, 2015a), which is a type of Region proposal Convolutional Neural Network (Figure 10). Faster-RCNN takes an image as input, and returns a number of candidate bounding boxes (region proposals) and a classification result for each box.

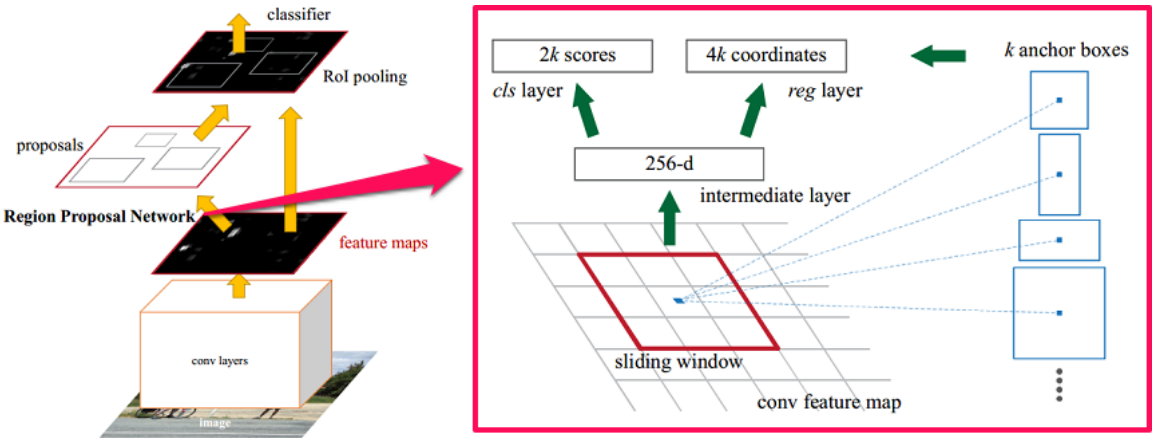


Figure 10: Depiction of the Faster-RCNN architecture. (image taken from (Ren et al., 2015b))

The algorithm of Faster-RCNN builds upon a typical convolutional neural network designed for classification, typically pre-trained on a large-scale classification task such as ImageNet. In the approach presented in D3.2, the model was VGG-16. During our experiments in Year 3, this was replaced by the much more powerful Inception-ResNet-v2 model (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017). This may incur a certain increase in training times, but is clearly the most powerful model currently available in terms of accuracy, and thus was deemed a preferable choice.

In addition to the model replacement, a number of experiments took place with respect to logo data augmentation for the aims of training data generation. One major modification to the process was the addition of perspective transform to the logo augmentation process. In the initial implementation of the CNN-based approach, the logos only underwent scaling, blurring, and brightness/color modification. To increase the robustness of the logo detection process to variants of the logo templates, we also implemented a new augmentation step including a degree of emulated perspective transform. This serves a

two-fold purpose: First, there exist a number of logos, both in the evaluation dataset and in real-world cases submitted by users, which feature rotation around the vertical axis. Including transformed versions of the logo in a similar rotation will strengthen the retrieval performance. Second, such transformations can assist the broader aims of data augmentation, that is to train the system to recognize variants of the logo templates by learning the broader patterns that characterize the logo. Theoretically, this could also allow the system to detect logos that are physically present in the scene (e.g. painted on a wall) instead of being overlaid in the frame. Figure 11 shows examples of three types of transformation. The first row displays the original approach used during Year 2, which included no perspective transform. The middle row shows examples of rotation along the horizontal axis. The bottom row shows examples along all axes, which was experimentally implemented but, since no such examples have been encountered either in our benchmark dataset or in submitted cases, we decided against using it, as it could needlessly increase the complexity of the system.

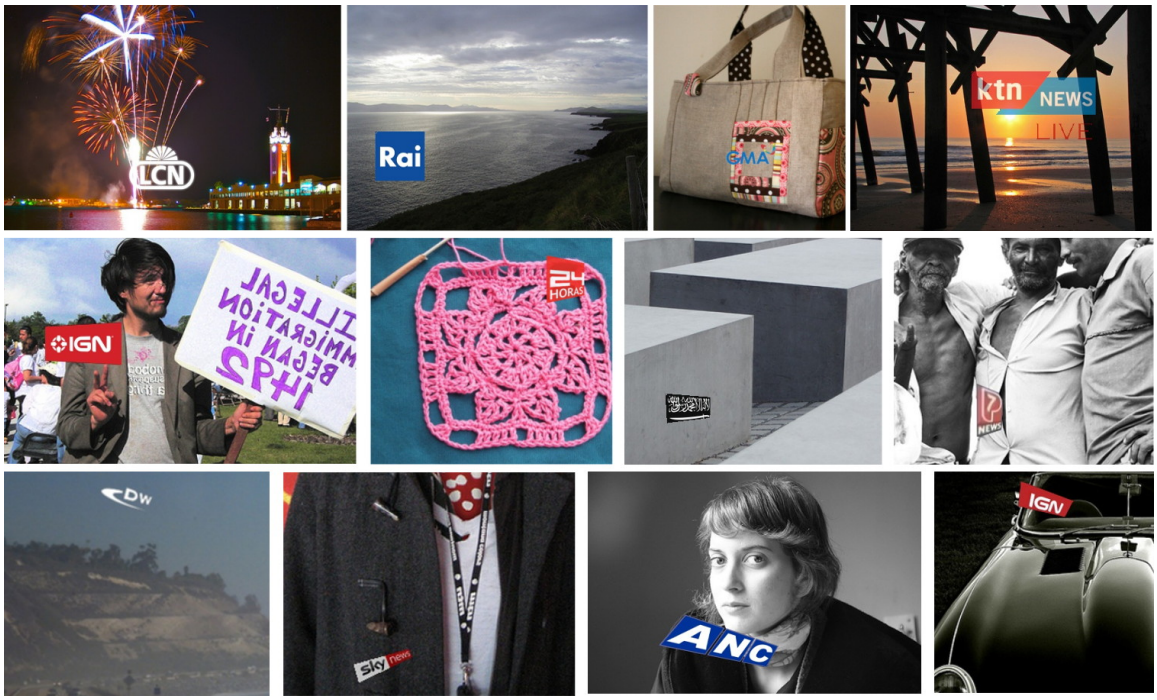


Figure 11: Examples of logo data augmentation in the artificially generated examples. Top row: no perspective transform. Middle row: perspective transform emulating rotation along the vertical axis only. Bottom row: perspective transform emulating rotation along all axes.

Further methodological modifications and considerations included lowering the minimum size of the logos in the training images, as the system tended to miss certain small logos, and experimentation with color augmentation. The former modification returned improved performance and was integrated in the final system, while the latter did not yield any significant improvement and was thus not included.

5.3 Progress and evaluations during Year 3

One disadvantage of the system presented in D3.2 concerned its implementation. At the time, the only available implementation of Faster-RCNN was py-faster-rcnn[10], which was based on a custom branch of Caffe. However, the branch stopped being maintained soon after, which became increasingly problematic in terms of integration and maintenance. To this end, during Year 3, we transitioned to a TensorFlow implementation[11], which, due to the popularity and traction of the framework, is expected to be steadily maintained for a long period of time.

The transition entailed the adaptation of the data augmentation code and the retraining of the model for the new framework. Furthermore, the evaluation dataset was revised, to correct various erroneous annotations, remove clips that included no logos at all, fuse logos that covered the same channel but were listed separately, etc. Additionally, a number of user-submitted logos were added to the model,

[10] https://github.com/rbgirshick/py-faster-rcnn
[11] https://github.com/tensorflow/models/tree/master/research/object_detection

which overall increased the complexity of the task. These models were submitted by actual end users of the service during Year 3 and were included in the final version of the model prior to the service update and the final evaluation (Figure 12). Overall, 12 new logos were added to the system in response to user requests.

Figure 12: Some of the additional logos that were added to the model during Year 3.

We re-ran evaluations on our benchmark dataset[12] for both the Caffe and TensorFlow version, using the extended logo template collection and the revised dataset. Table 13 shows the True Positive and False Positive rates per video and per shot, where the results from the two models appear comparable. However, the more standardized metric of mean Average Precision (mAP) shown in Table 14 shows a small but clear advantage for the new, TensorFlow-based implementation under the new augmentation scheme.

Table 13: Logo detection evaluation results

	Per video		per shot	
	Caffe	TensorFlow	Caffe	TensorFlow
True Detections	0.78	0.75	0.61	0.67
False Detections	0.10	0.06	0.01	0.04

Table 14: Mean Average Precision for the logo detection evaluation

	Caffe	TensorFlow
mAP	58.35	63.7

5.4 API layer and integration with InVID

In addition to the above updates, the architecture of the integrated version of the component was re-designed around two parts. One part is the manager, containing the service API, accepting and responding to calls from the platform, and handling communication with the outside world: querying and downloading content from Web and social media sources, sending requests to the video fragmentation service, and downloading video keyframes. It also manages preprocessing of the content prior to the detection. This part of the service has relatively low computational requirements and can be deployed on any server. The second component contains the neural network and is tasked with the detection of logos in the input images/keyframes. It exposes a simple API, essentially accepting a single call, and is only accessible to the manager (i.e. no other third-parties may access it). This separation was made

[12]The benchmark dataset remained almost the same as the one used in D3.2. Minor differences include a few data quality issues that were fixed.

Table 15: Calls exposed by the Logo Detection module API.

Service	URL	Parameter	Notes
image from URL	`/fromimageurl`	`url=`<image url> `timeline=`<1 or 0>	
video from URL	`/fromvideourl`	`url=`<video url> `timeline=`<1 or 0>	YouTube, DailyMotion, Facebook, Twitter, Dropbox, or video file.
identify content	`/fromurl`	`url=`<content url>	
analysis status	`/analysisstatus`	`id=`<item id>, `timeline=`<1 or 0>	Useful for tracking "image from file"
get logo list	`/logolist`		
submit new logo	`/submitlogo`	`imageurl=`<logo image URL> `logoname=`<logo class name> `wiki=`<Wikipedia link>	

because this part of the service needs to be deployed on a more powerful system with a CUDA-enabled NVIDIA GPU to take advantage of the increased computational speed, and we did not want to expose this system to an external API, nor burden it with the cost of management, which can be performed by any other PC.

This two-part architecture enabled us to easily replace the Caffe-based detector with a TensorFlow-based one, since no modifications to the manager part were necessary for that. Thus, this transition took place seamlessly, leading to a Logo Detection component for InVID, which will ease its maintenance. The manager part of the component also underwent various modifications, mostly with respect to input handling, but also to better align with the final requirements of the InVID platform. As a result, since InVID operates using external item URLs and not some internal database index system, the two calls `/fromimageid` and `/fromvideoid` were removed from the API. The same applies to the `/fromimagefile` call.

The remaining calls are integrated as part of the Verification Application. The timeline-based format is the one currently used by the App, while the non-timeline based one is maintained for reasons of backwards compatibility. The current implementation of the service now offers, stability, portability, speed, and maintainability, while its list of known logos is still open for extension based on user submissions.

6 Location detection

During the third year of InVID, our work in Location detection focused on both improving the system performance, mostly with respect to disambiguation, where various techniques are combined to achieve state-of-the-art performance, and with respect to creating an ecosystem around the Recognyze tool (Weichselbraun, Kuntschik, & Braşoveanu, 2018) and aiming to provide data and methodologies for algorithm evaluation that goes beyond the current limitations of the field. Thus, besides providing a powerful location detection component for the InVID platform, the impact of our work in this task also led to the creation of a set of evaluation tools aimed to lead to further improvements in the field of location detection.

6.1 State of the art

The most successful Named Entity Linking (NEL) systems currently proposed in the literature can be organized into three broad classes:

- **Knowledge Graph (KG) disambiguation** is currently considered among the most effective approaches towards NEL. Several graph disambiguation NEL tools have been listed among the top performers in NLP competitions (e.g., TAC-KBP, OKE, SemEval): Hachey's system (Hachey, Radford, & Curran, 2011), AIDA (Hoffart et al., 2011), HITS (Guo, Che, Liu, & Li, 2011), Babelfy (Moro, Raganato, & Navigli, 2014) and AGDISTIS (Usbeck et al., 2014).
- **Statistical models** including mixtures of Conditional Random Fields models - (e.g., ADEL (Plu, Rizzo, & Troncy, 2016) or DBpedia Spotlight (Daiber, Jakob, Hokamp, & Mendes, 2013)) exploit classic Machine Learning approaches.
- **Neural models** are relatively new, but promising (e.g., the Convolutional Semantic Similarity model for NEL proposed by Francis-Landau (Francis-Landau, Durrett, & Klein, 2016)) and are used in order to jointly resolve the detection and resolution of links.

Regardless of the model that is globally used for disambiguation, all NEL tools need to link the entities to a target Knowledge Base, therefore they need to exploit the relations between the entities or the graph structure of Linked Open Data. Our own tool, Recognyze, builds upon the graph-based disambiguation method.

6.2 Method description

6.2.1 Graph Disambiguation

The approach used in Recognyze for NEL exploits the links between entities found in a text. Similar to Usbek et al. (Usbeck et al., 2014) we define our approach as follows: Given a knowledge base K as a directed graph $G = (V, E)$ with vertices V and edges E, Recognyze uses SPARQL queries to obtain a sub-graph $G' = (V', E')$ with the following properties:

1. resources $s \in V'$ and $o \in V'$ where o might either refer to a resource or a literal (i.e. in this case a name used to identify a named entity)
2. for every pair $(s, o) \in E \Rightarrow \exists p : (s, p, o)$ which is denoted to as an RDF triple in G'.

The named entity disambiguation process comprises multiple sub-tasks: (i) Directed Acyclic Word Graphs (DAWGs) (Scharl, Weichselbraun, Göbel, Rafelsberger, & Kamolov, 2016) provide fast text search within the input documents to identify candidate entities by locating mentions of their name variances. (ii) A controlled vocabulary is applied to search for potential affixes that hint on relevant entity types. (iii) These affixes are then used to remove candidate mentions that do not match the type implied by the affix. (iv) The remaining candidate entities are then linked using multiple disambiguation algorithms in sequence. In this sub-task, the relations between the candidate mentions, as well as the significance of a single mention are used to determine the best fitting network of entities. (v) Finally, Recognyze transforms the accepted entities into the desired output format.

Name variance is the problem of finding all names that refer to a single entity within a collection of text. In theory, enriching G' with name variances improves recall, whereas adding name variance related features to the NEL extraction pipelines improves precision. Several cases of variance have been

described in the literature (e.g., (Ehrmann, Jacquet, & Steinberger, 2017) or (Ji et al., 2016)). Locations have more problems with name variances than the other classes due to overlap and assimilation (e.g., people and organization names often contain location references), but can still include place qualifiers (e.g., N/E/S/W, “So” for “Southern”); regional abbreviations (e.g., “OH” for “Ohio”); embeddings or nested entities (e.g., “New York Stadium”); possessive names (e.g., “Hawaii’s Waikiki”); and addresses (e.g., “221B Baker Street”).

6.2.2 Recognyze Architecture

Recognyze uses lexicons and profiles for defining the sources and algorithms that will be used when performing a named entity search. A lexicon contains all the details related to the extraction of data from a particular source (e.g., details about repository, entity types, if it includes abbreviations, etc.). A profile can use multiple lexicons in order to deliver the extracted entities, therefore allowing for the possibility of also using multiple Linked Data sources for providing additional details about an entity (e.g., a location’s name can be taken from DBpedia, whereas alternative names can be taken from Wikidata). Several low-level components can be used when defining a set of lexicons and profiles as it can easily be seen in the Background Knowledge Acquisition section from Figure 13):

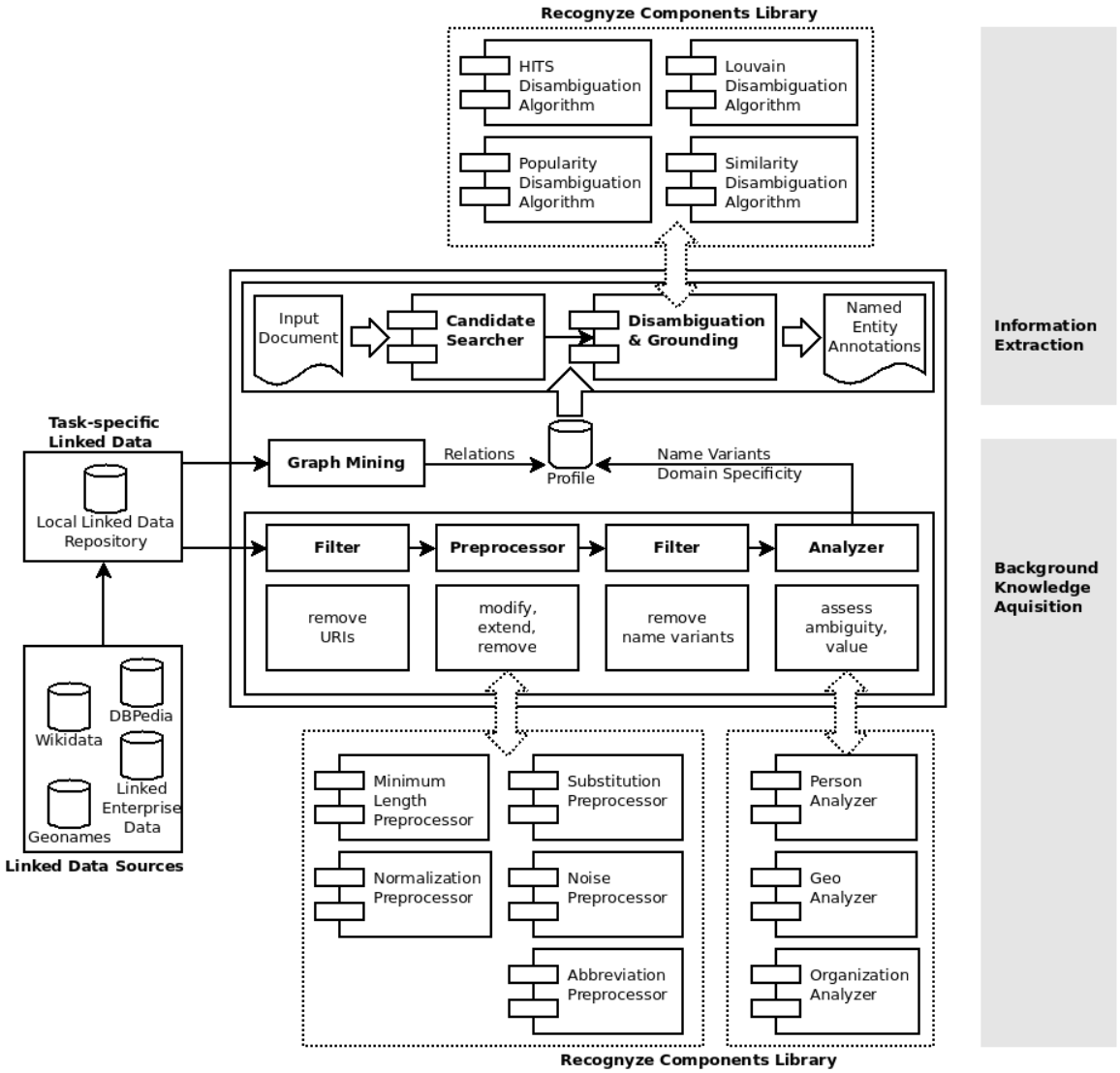


Figure 13: Recognyze architecture

– **Linked Data Sources.** A repository that contains a set of custom builds for well-known KBs like Wikidata or DBpedia. Typically only the entity types defined through the lexicon (e.g., Location types like Places, Natural Locations or Facilities) will be extracted.

- **Filters.** A set of filters used for removing bad URIs (e.g., pornography) or name variants.
- **Preprocessors.** A set of components designed to perform specific tasks upon the entities contained in the Linked Data sources like extracting abbreviations, removing noise or limiting the minimum character count.
- **Analyzers.** A set of components that receive a set of name variants and return only those that match certain criteria (e.g., entities with certain types).

A set of higher-level modules use the profiles in order to perform a set of Information Extraction tasks as shown in Figure 13:

- **Graph Mining Configuration Component.** This is a component that includes the lexicons and profiles that will later be used during the NEL process. Both lexicons and profiles are defined as JSON files and include the low-level configuration needed for different types of search.
- **Candidate Searcher.** A component that searches for the best candidates for a certain query.
- **Disambiguation and Grounding Component.** A component that implements a set of algorithms used for performing disambiguations and delivering the entity annotations.

There is no need for a separate Linker component as the URI is always used as a key for a certain entity. This also assures that every entity will have a unique identifier regardless of its provenance.

6.2.3 Recognyze Ecosystem

Recognyze was developed jointly by MODUL Technology and HTW Chur and it is integrated in the webLyzard Platform (Scharl, Weichselbraun, Göbel, Rafelsberger, & Kamolov, 2016) as the default annotation engine for top entities (e.g., Location, Organization, Person). The ecosystem includes a set of tools built for using Recognyze in production (e.g., enrichment tool), as well as for continuous evaluation (e.g., visualizing annotations or evaluation results):

- **Recognyze wrappers for NER tools** like Stanford[13] or Spacy[14] that are available as individual components.
- **Jairo** is an annotation enrichment component developed jointly by MODUL Technoloy and HTW Chur.
- **Recognyze Annotation Visualizer** - a component that is used for debugging in order to create and test new corpora for Recognyze.
- **Orbis** is an evaluation and debugging engine jointly developed by MODUL Technology and HTW Chur. The component is described in the paper (Odoni, Kuntschik, Brasoveanu, & Weichselbraun, 2018) and is an extension of the error analysis methodology developed jointly by MODUL Technology, ISMB Turin and HTW Chur and published in (Braşoveanu, Rizzo, Kuntschick, Weichselbraun, & Nixon, 2018).

While not immediately apparent, NEL evaluations are still plagued by errors, even though the number of good scorers is on the rise. Issues can appear due to different guidelines or taxonomies used during the initial annotation of the gold standards, changes between KB versions, redirects, links in multiple languages or even due to the scoring components. A taxonomy of error classes collected from multiple annotators and gold standards based on the most likely location where the error was triggered is presented in (Braşoveanu et al., 2018), together with examples of the five discussed error classes: Knowledge Base (KB), Dataset (DS), Annotator (AN), NIL Clustering (NIL), and Scorer (SE). The proposed taxonomy can also help KB or evaluation systems maintainers to spot errors in their tools, which makes it ideal as a basis for rapid debugging. A tool from the GERBIL ecosystem, EAGLET (Jha, Röder, & Ngomo, 2017), presents similar ideas, but focuses mostly on classifying several error types (e.g., redirects or missing annotations) found in gold standards.

As already highlighted in the previous paragraph, evaluation is a difficult topic. While evaluation tools like TAC-KBP (Ji et al., 2016) or GERBIL (Usbeck et al., 2015) have existed for years, the evaluated

[13] https://stanfordnlp.github.io/CoreNLP/
[14] https://spacy.io/

systems themselves had various issues and have traditionally been hard to debug. In order to fix this, we have introduced the concept of **transparent benchmarking** by building on top of the primary analyses from the TAC-KBP tools (Hachey, Nothman, & Radford, 2014).

In our opinion, **transparent benchmarking** systems need to fulfill the following six requirements:

– *widely recognized metrics* - precision, recall, F1, accuracy or clustering measures;

– *explained evaluation runs* - we should not only be able to see the evaluation results, but also the classification into test results like false positives or false negatives or even into more fine-grained error classes if possible;

– *integrated visual analysis methods* - drill-down analysis should be used for inspecting and debugging the results;

– *support for resource versioning* - is needed in order to allow an evaluation to run with a previous version of a KB (e.g., run with DBpedia 3.9 or DBpedia 2015-10);

– *reproducible settings for the annotator tools and the annotation tasks* - the settings that correspond to results published in a paper should be publicly available;

– *machine-readable annotation guidelines* - while annotation guidelines like those from TAC-KBP (Ji & Nothman, 2016) are publicly available, it is hard to do reasoning with them or to combine them according to the task due to the fact that they are not available in a machine readable-format like RDF or its derivatives.

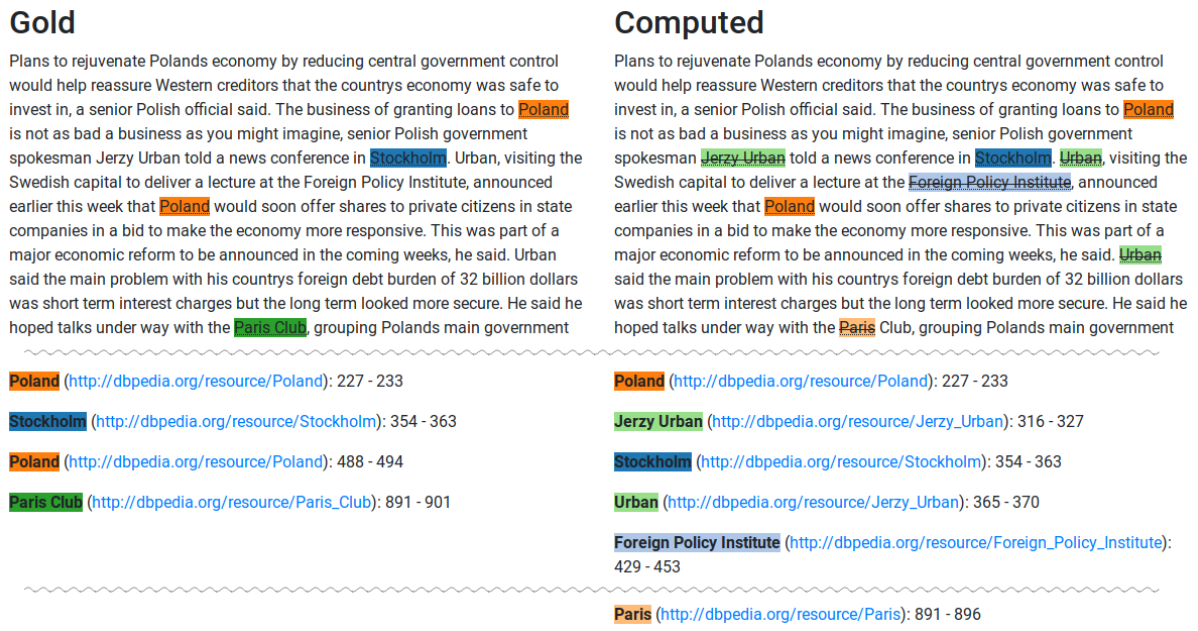


Figure 14: A cropped screenshot of the results generated by the Orbis evaluation. Left demonstrates the gold standard, right demonstrates the results returned by the annotator system. The upper half highlights the annotations in the used test document, while the lower half lists the annotations in textual order. Matching colors indicate identical resources.

Since the current generation of annotation tools rarely publish their best settings and annotation guidelines are not really available in machine-readable formats to the best of our knowledge, the last two steps can be themselves considered open research problems at the moment.

Orbis (Odoni et al., 2018), designed to be our first iteration of a transparent benchmarking system, is an extensible evaluation framework written in Python 3.6 which offers multiple evaluation modes, resource versioning, parallel evaluation runs, dataset normalization, and drill-down analyses. These features were built with a flexible pipeline system designed to help configure, modify and extend evaluation processes. Orbis addresses the need for transparent benchmarking and visual inspection of the evaluation runs.

Jairo is a small component that is used for enriching the extracted entities. Due to constraints related to size, building large lexicons for full DBpedia or Wikipedia builds is prohibitive. One method of reducing the size is to simply save the basic data (e.g., latitude, longitude, type, short abstract) about existing entities in lexicons and later query the KB in order to get additional data (e.g., long abstract, links to other KBs, etc). The preferred KB is our Semantic Knowledge Base (SKB) which was originally set up to improve keyword extraction for the story detection and social media retrieval. It aggregates data from multiple Knowledge Bases into a triple store and is being expanded from lexical entities to other entity types such as Events and Works. This helps lower the storage footprint for the Recognyze entity lexicons.

Jairo's configuration allows to (i) define entity extractors from an input stream, (ii) specify which fields to expand, as well as which sources and predicates to use in order to fill these fields, and (iii) define formatters for the returned entity. The output of Jairo will contain both the original information returned by Recognyze, as well as the mined additional information returned by Jairo.

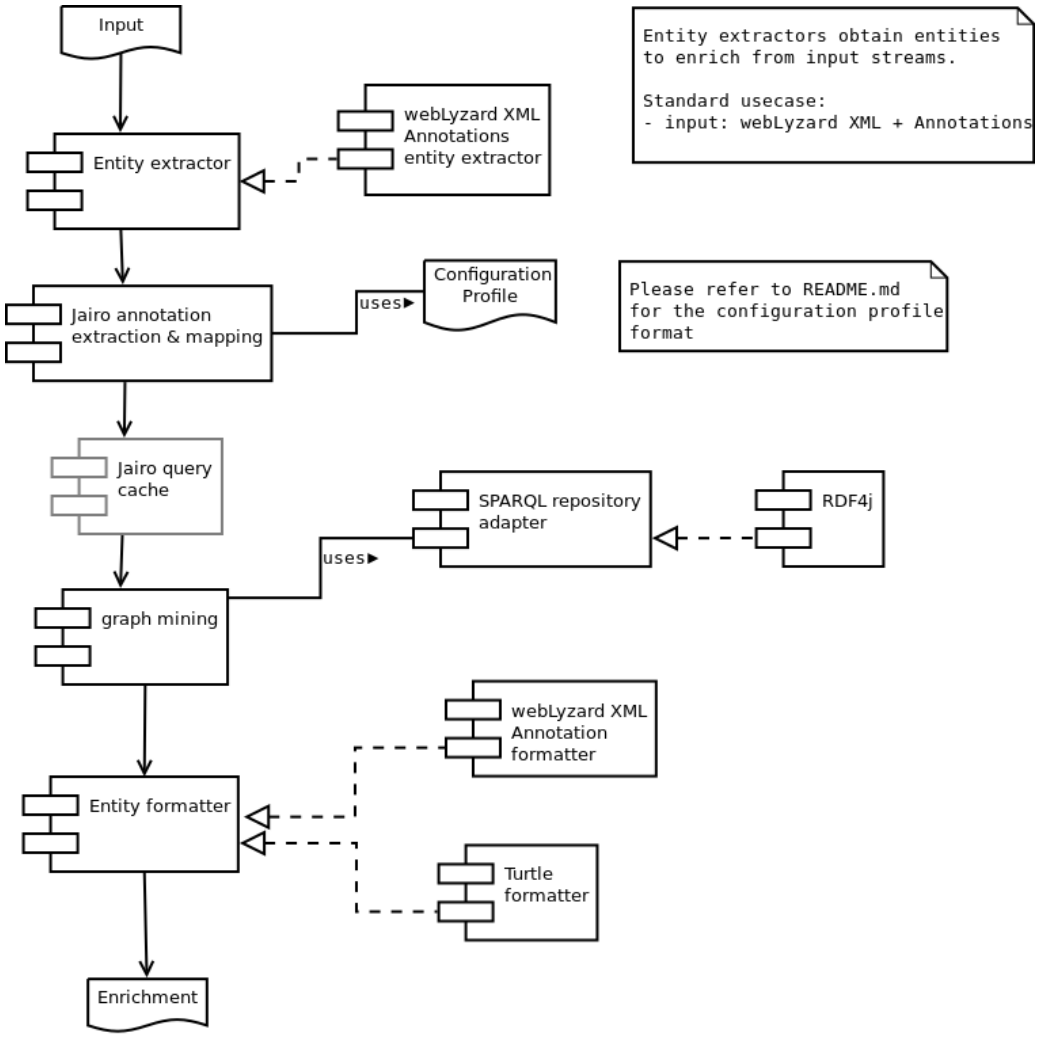


Figure 15: Jairo architecture

6.3 Progress and evaluations during Year 3

Named Entity Linking (NEL) and associated Knowledge Base Population Tasks (Named Entity Recognition - NER, Cold Start Slot Filling - CSSF, Automatic Knowledge Base Completion - AKBC, etc.) are among the most difficult tasks in the fields of NLP and Semantic Web. While NER tasks typically have good results (an F1 of 0.8-0.9 being expected), it is rare for the rest of the Knowledge Base Population

tasks to yield such good results. In fact, an F1 value of over 0.60 will rank in top 10 systems for Named Entity Recognition Linking and Classification or NERLC tasks (see Tables 5, 6 and 7 from (Ji et al., 2017)) in any competition, whereas an F1 value of over 0.5 will rank in top 5 systems for Cold Start Slot Filling (see (Huang, Sil, Ji, & Florian, 2017)).

Many of the settings included in Table 16 shed light on pitfalls relevant to name variance for NEL. When we designed Recognyze, we proceeded incrementally, therefore expecting better results for each setting. This has not always been the case. For instance, the setting (b1) *baseline+wikidata* yields worse results than the profiles that surround it. Initially we suspected that this effect might have been caused by data quality issues within Wikidata which is considered a relatively novel data source (Erxleben, Günther, Krötzsch, Mendez, & Vrandecic, 2014). An analysis of the issue uncovered that the quality of Wikidata is actually high and that it yields a lot of name variants per entity. This in itself is a problem as it can lead to many different types of clashes.

By far the most common problem was related to ambiguous name variances introduced by string splitting. Longer strings were often split into multiple entities (e.g., *Canadian Bashaw Leduc Oil and Gas Ltd* was split into *Canadian*, *Bashaw* and *Leduc*). This might not be an issue if the entity is a Person and some of the resulting splits are actually roles, but if each token actually references a different entity (e.g. *West German Finance Minister Gerhard Stoltenberg* included links to such ambiguous entities like `dbr:West_Texas`, `dbr:German_New_York` and `dbr:Minister_(Catholic_Church)`) or if there are any containment issues (e.g. *Texas Gulf Coast* is a part of *Texas*), the resulting effect on the overall results will be rather similar to negative compounding. This observation triggered our research in Name Analyzer heuristics and machine learning algorithms which addresses this problem. When used in combination, both the algorithmic name generation and name analyzer components perform considerably better than the baseline+wikidata precisely because they delivered less ambiguous name variants.

The comparison presented in Table 17 aims at getting a clear understanding of the competitiveness of the discussed name variance methods and assessing whether other NEL systems could benefit from it as well. Each tool has committed a different set of errors, although the issue of ambiguous name variances due to the splitting of longer names was noticed in all tools to some degree. Most of the systems (e.g., AIDA, Babelnet) also failed to correctly identify all the name variants that belong to an entity. In addition, they either do not take into account abbreviations or they rarely get them correctly. In some cases, prefixes (e.g., country abbreviations - *U.S.*, *U.K.*) and suffixes (e.g., terminations like *Land*) have also created problems. Based on our analysis at least name analyzers and techniques for obtaining abbreviations would be beneficial for improving the performance of all analyzed systems.

Nevertheless, it needs to be noted that name variance techniques will probably not be always sufficient to address these kinds of errors, since often assigning all name variants to the correct entities is also a coreference or clustering issue.

Table 16: Impact of name variance on the Recognyze Named Entity Linking performance for the Reuter128 dataset. Bold figures indicate statistically significant improvements over the baseline.

	Setting	LOC			All		
		P	R	F_1	P	R	F_1
	baseline	0.63	0.54	0.58	0.66	0.39	0.49
(a)	additional properties	0.63	0.54	0.58	0.66	0.38	0.49
(b1)	Wikidata	0.14	0.41	0.20	0.21	0.41	0.28
(b2)	Wikipedia	0.61	0.54	0.57	0.64	0.39	0.48
(b3)	GeoNames	0.60	0.54	0.57	0.64	0.38	0.48
(b4)	baseline + (b1 + b2 + b3)	0.14	0.41	0.21	0.21	0.41	0.28
(c)	algorithmic name generation	0.54	**0.72**	**0.62**	0.43	**0.58**	0.50
(d1)	name generation on Wikidata	0.52	0.54	0.53	0.61	0.42	0.50
(d2)	name generation on Wikipedia	0.58	0.52	0.55	0.63	0.39	0.48
(d3)	name generation on GeoNames	0.48	0.53	0.51	0.58	0.38	0.46
(d4)	baseline + (d1 + d2 + d3)	0.46	0.53	0.50	0.58	0.42	0.49
(e1)	name analyzer (heuristic)	0.64	0.52	0.57	0.54	**0.48**	0.51
(e2)	name analyzer (machine learning)	**0.65**	0.51	0.57	0.42	**0.48**	0.45
(f)	baseline + (a, c, d1, e1)	0.53	**0.70**	0.61	0.58	**0.58**	**0.58**

Table 17: Comparison of the system performance on the Reuters 128 and OKE2015 corpora.

Corpus	System	LOC			All		
		P	R	F_1	P	R	F_1
Reuters 128	AIDA	0.44	0.64	0.52	0.53	0.43	0.47
	BabelNet	0.29	0.31	0.30	0.32	0.22	0.26
	Recognyze	**0.53**	**0.70**	**0.61**	**0.58**	**0.58**	**0.58**
	Spotlight	0.41	0.70	0.52	0.50	0.49	0.49
OKE 2015	AIDA	0.25	0.37	0.30	0.50	0.41	0.45
	BabelNet	0.21	0.35	0.26	0.40	0.26	0.32
	Recognyze	**0.62**	**0.73**	**0.67**	**0.73**	**0.59**	**0.65**
	Spotlight	0.50	0.72	0.59	0.61	0.36	0.45

6.4 API layer and integration with InVID

Recognyze has an entire ecosystem built around it, therefore its core product needs to be run in multiple ways. Typically it can either be built with Maven or built and consecutively run as a standalone *fat* jar. Most often, it is deployed as a Docker container and accessed via the provided API. Java and Python clients for it are available through the webLyzard library[15].

When the Recognyze API is deployed, it displays a Swagger interface. Once deployed, a complete list of the available services can be found at the following address: `http://<base_host>:63007/index.html?url=rest/swagger.json`

Table 18: Recognyze API for Named Entity Linking

Operation	**URL**	**Expected Output**
Status	*status*	Check service availability.
List Profiles	*list_profiles*	List available profiles.
Load Profile	*load_profile/profile-name*	Load serialized profile.
Remove Profile	*remove_profile/profile-name*	Remove serialized profile.
Annotate Document	*search_document?profileName=profile-name*	Returns an annotated document.

Recognyze typically returns the surface names and links for the detected entities. Jairo is the API that enriches the Recognyze results delivering the rest of the details about a certain entity (e.g., for locations we might get short abstracts, labels, region, country, population size, and so on).

Table 19: Jairo API for data enrichment

Operation	**URL**	**Expected Output**
Status	*status*	Check service availability.
List Profiles	*profiles/list*	List available profiles.
Load Profile	*profiles/add/profile-name*	Load new profile.
Enrich Annotation	*annotations/enrich/profile-name*	Returns enriched annotations.

Since Recogynze is dockerized, it can be deployed in its InVID configuration (optimized location detection) and made available to the InVID Verification Application or any other service which needs accurate extraction of locations from content.

[15] https://github.com/weblyzard/weblyzard_api

7 Context Aggregation and Analysis

During the last project year the Context Aggregation and Analysis (CAA) component was improved and optimized based both on experimental evaluations and user feedback received via the project test cycles. This led to the component reaching its final version as described in this section. It currently supports YouTube, Facebook and Twitter video sources. Significant effort was dedicated to improving the analysis and presentation of data drawn directly from the video platform APIs, which are aimed to be analyzed by a human investigator. In addition, an automatic approach of assigning credibility scores to suspicious videos was implemented. The Fake Video Corpus 2018 (FVC-2018), that is, the large-scale corpus of debunked and verified videos presented in Section 2.1 was built in part with the help of the CAA service, and in return it was the subject of an analysis as part of our work in contextual verification. FVC-2018 played another important part in this year's work in this task, as it was used to train automatic credibility scoring models and to support a new CAA feature.

In this section, we present our work in improving the CAA service, as well as the development of an automatic tool for contextual video verification. We also present our analysis of the dominant patterns within the Fake Video Corpus 2018 and how they could be useful from a verification perspective. We then show the results of our evaluations, which highlight both the potential of the current method and the complexity of the FVC-2018 as a benchmark dataset. We conclude by presenting the final structure of the API and the component's integration with the InVID platform.

7.1 State of the art

The journalist behavior and practices that are used to collect and verify user generated content from social platforms are described in (Rauchfleisch, Artho, Metag, Post, & Schäfer, 2017) and (Heravi & Harrower, 2016) focusing on the dissemination of information through Twitter. Related work from the literature that deals with means of helping journalists or other news professionals to decide on the truthfulness of UGC can be broadly classified in two types: i) verification services, including fact-checking organizations and websites, and ii) verification tools. Verification services take the responsibility of verifying content themselves, and publish reports explaining their findings and justifying their conclusions. They are very useful for dealing with past cases, or with old cases that are being refurbished as new, but due to the effort required to authoritatively verify or debunk a new case, they generally cannot help with breaking news, when journalists expect to be able to verify a piece of information within a short time span (e.g. minutes, or hours). Fact-checking initiatives and operations are gaining popularity and their number is increasing: there were 161 active services in 2018 based on Duke Reporters Lab[16] (Stencel, n.d.). Some of the most well-known ones include FactCheck.org[17], Snopes[18], and StopFake[19].

With regards to verification tools, there are works, such as (Elkasrawi, Dengel, Abdelsamad, & Bukhari, 2016) and (Pasquini, Brunetta, Vinci, Conotter, & Boato, 2015), that focus on image authenticity and consequently can be used to debunk the online news story where a forged image is attached. Fakebox[20] is another tool of verifying news articles by providing information about the title, the content and the domain of the article. Similarly to Google reverse image search, TinEye[21] supports searches for similar images on the web, which may be useful for journalists when conducting provenance analyses of online video and images.

Another relevant field concerns tweet verification. Tweets are a common way for user contributions to breaking news, and they can contain any type of content: text, image, or video. TruthNest[22] and the Tweet Verification Assistant[23] provide an integrated solution for Tweet verification using contextual information.

Finally, a verification task that is distinct from our work but related to it is rumour detection. Rumour detection concerns the accumulation and analysis of a collection of social media items posted around a claim. The field of rumour detection is quite relevant to the task of video verification since the suspicious disinformation videos are potentially disseminated through multiple social networks and by multiple users, thus following similar patterns to rumours. Since the FVC-2018 is organized into 'video

[16] https://reporterslab.org/fact-checking/
[17] http://www.factcheck.org
[18] http://snopes.com
[19] http://stopfake.org
[20] https://machinebox.io/docs/fakebox#uses-for-fakebox
[21] http://tineye.com
[22] http://www.truthnest.com/
[23] http://reveal-mklab.iti.gr/reveal/fake/

cascades', which consist of the first chronologically video of the event and its near duplicate instances, each 'video cascade' could be considered to correspond to a rumour. A survey of studies in presented in (Cao et al., 2018) regarding the task of rumour detection. On the other hand, however, there are distinct differences between rumours and video cascades, the most prominent being that, when dealing with video near-duplicates, we only know the time sequence in which they appeared, and not the exact dissemination pattern (i.e. which exact video version was used as input for a new post). Thus, it is very difficult to apply some of the most successful rumour analysis methods to video cascades. Applying techniques from rumour analysis to video cascade verification is thus, an interesting opportunity for future research in this area.

7.2 Method description

The CAA component accepts as input the URL of a video published on YouTube, Facebook or Twitter. Regarding Twitter, a tweet contains either a Twitter Native video or an embedded YouTube video, i.e. a link to a YouTube video. CAA is able to deal with both cases. The component collects all the metadata that the corresponding API can provide, and applies a number of filtering, organization, and analysis steps to make them more easy for a user to analyze and decide on the veracity of the video. Furthermore, in order to enrich the verification report, the data and location where the event supposedly happened can be provided as input, in which case the weather data corresponding to the claimed time and place are fetched and included in the report. These two steps were also present in the version of the CAA component presented in D3.2, and are still there, with enhancements and improvements. Furthermore, two more procedures were implemented to supplement the CAA features. First, credibility features are extracted from the video, and a machine learning model is used to automatically evaluate its veracity. The approach draws from the automatic verification algorithm we described in D3.2, with extensions and improvements presented here. On the other hand, since the FVC-2018 dataset includes a large number of well-established fakes, which often reappear as new fake cases, we also leveraged a separate instance of the near-duplicate detection service described in Section 4 to search for near-duplicates of the submitted video within FVC-2018, and if a duplicate is found, to notify the end user, with an alert including the oldest found instance of the particular video.

The information that is collected based on the input URL is:

- Data from the video source; the component communicates with the corresponding API and collects information about the account and the video.
- Data from Twitter search; the URL of the submitted video is submitted as query to the Twitter search API and the tweets that include it are returned.
- Video and channel features; these are calculated using the video and the channel data extracted from the video source.

The collected information is then used as input to the CAA analysis processes, resulting in five verification reports that together constitute the final CAA verification report shown in Fig. 16.

7.2.1 Filtering, organization and analysis

Each video platform API provides a large amount of data about the hosted video itself as well as the publisher of the video. We filter and organize this information and create a subset of verification cues. As explained in D3.1, the term verification cues refers to information that can assist investigators in identifying any type of fake video. Although we tried to create a common reference for videos of different platforms, we finally decided that separate reports per video platform is a more helpful and effective approach. In D3.2 the indicators per video platform were listed, for Facebook and Twitter videos. However, due to changes in the Facebook Graph API[24], the metadata of videos posted by Facebook Groups and the information about the Facebook Page or Group that posted a video are no longer available and thus these fields are not included in the metadata report. The video indicators (i.e. the information about the video itself) and channel/user indicators for YouTube and Twitter (since for Facebook this information is not available) are listed in Fig. 17 and 18 respectively. Both figures contain the indicators that are common between the two video platforms at the top, and the individual indicators below.

In addition to the features that are directly drawn from the video platform APIs, there are several features that are calculated using this data. These features include the verification-related comments

[24] https://developers.facebook.com/blog/post/2018/04/24/new-facebook-platform-product-changes-policy-updates/

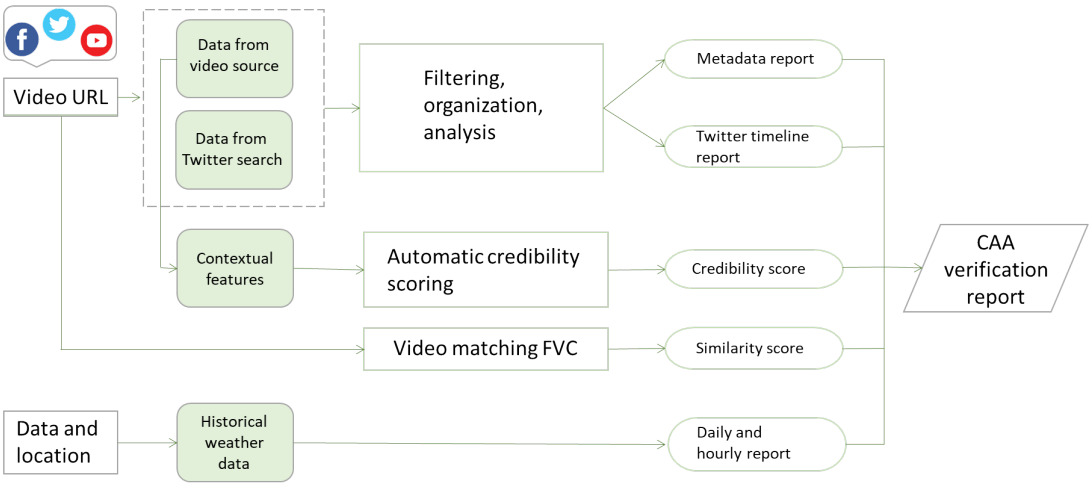


Figure 16: The architecture of the Context Aggregation and Analysis component.

and their number, the locations mentioned in the video title and description which are extracted using the updated and more accurate release of Recognyze module (Section 6), the average number of videos per month uploaded by the channel which is calculated by dividing the total number videos posted by this channel with the number of months since this channel was created, reverse image search to Google and Yandex image search machines and the Twitter URL for searching the tweets that share the link of the submitted video.

Furthermore, taking into account feedback by media experts and other users, we enriched the 'calculate' part of the verification report by implementing a new feature which builds on the verification-related comments approach. Specifically, we implemented a mechanism that can extract subsets of the overall video comments based on user-defined keywords. Although the predefined list of verification-related keywords has proven very useful for the verification process during the first two years of InVID, from user feedback we realized that there exist cases where different keywords would be more helpful to the investigation. Moreover, the investigator will have the ability to use keywords based on his/her experience on the subject. The user can provide the keywords and define either that every one of them should be present in the text of a comment, for it to be selected (logical operator AND) or that at least one of those words should be contained in the text of the comment (logical operator OR). In Fig. 19 an example is presented, where the user has given the keywords 'filmed' and 'Syria'. In 19(a) the words are combined with the OR logical operator for search, while in figure 19(b) the same two keywords are used with the AND operator.

7.2.2 Automatic credibility scoring

In D3.2 we experimented with an automatic video verification approach which built classification models over two sets of features (Table 20), comment- and video-based. The first one is based on the comments under the video. The second set of features is based on the video metadata, and specifically linguistic features extracted from the video description text and statistics extracted from the video channel.

Having extracted these features, the automatic credibility scoring algorithm then proceeds to train Support Vector Machine (SVM) models. The features are combined using the agreement-based approach of (Boididou et al., 2018) and feature concatenation, as described in D3.2. Feature evaluation takes place using 10-fold cross validation. The method was implemented as shown in the flowchart of Figure 20. The resulting credibility score that is assigned to each video by the automatic verification approach aims to give another cue of the video truthfulness, which, in combination with the first level verification report, can help the user make the final decision.

7.2.3 Video matching against the Fake Video Corpus

Finally, another feature of the CAA service is based on the FVC-2018 (described in detail in Section 2.1). This feature aims to detect videos that have already been associated with (mis-) disinformation in the past. Besides the debunked and verified videos, all cases contained in the FVC-2018 are accompanied with infomation including a description of the case, a label indicating the type of the misinformation, and

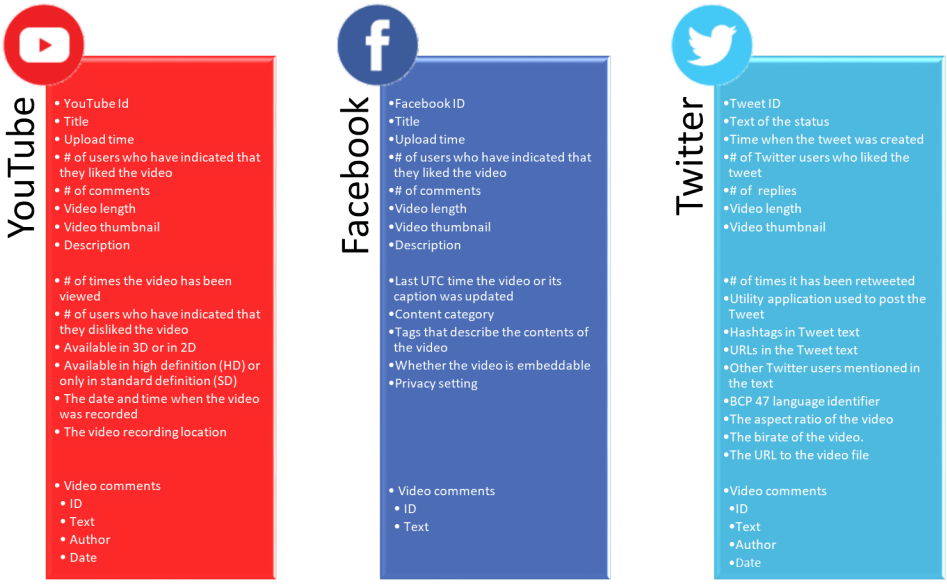


Figure 17: Video verification indicators derived directly by the video platform API. Video indicators refer to the information about video itself.

a link to a reliable source debunking it (if available). The near duplicate detection algorithm of Section 4.2 is used to search within the pool of already debunked videos of the FVC-2018. If there is a match with a video in the dataset, then the system checks whether that video belongs to a cascade containing more videos. If that is the case, then the URL of the chronologically oldest video among all near duplicate instances is returned. Otherwise, the URL of the matched video is provided. Additionally, as part of the FVC-2018 all accompanying pieces of information (i.e. the type of manipulation and the explanation of the false claim) are also included in the report. Furthermore, there are two special cases that require special treatment.

– The matched video has been removed from the video source and is currently not available online, existing only in the FVC-2018 index. In this case the report contains the metadata of the video, specifically the date that it was published and the publisher (channel in case of YouTube, page in case of Facebook and user in case of Twitter video). Moreover, URLs of other near duplicate instances are provided, if they exist, to help the user verify that it was indeed a near-duplicate of the submitted video.

– The matched video is actually later chronologically than the submitted one. This may occur when a user submits an old video that should be part of the dataset, but was not caught by our semi-automatic video collection process. In that case, a more recent version of the video is displayed. However, this is generally an unlikely event, since the CAA service is primarily designed for news professionals and non-experts who want to verify current breaking events. Submitting a video older than all its FVC-2018 near-duplicates is possible, but does not generally fall into the intended mode of use of the service.

The idea of this feature is to add another layer of user assistance, by automatically detecting well-established, already debunked fake videos. The module does not provide a final decision and does not label the video as fake or real.

7.3 Empirical Analysis of FVC-2018

Besides using FVC-2018 as a reference database for capturing well-known cases of past fakes, as in Subsection 7.2.3, the dataset also provided an opportunity to gain a better perspective on the characteristics and dissemination patterns of real and fake videos, and what distinguishes one from the other. We analysed the characteristics of the fake and real videos in terms of the videos themselves, their

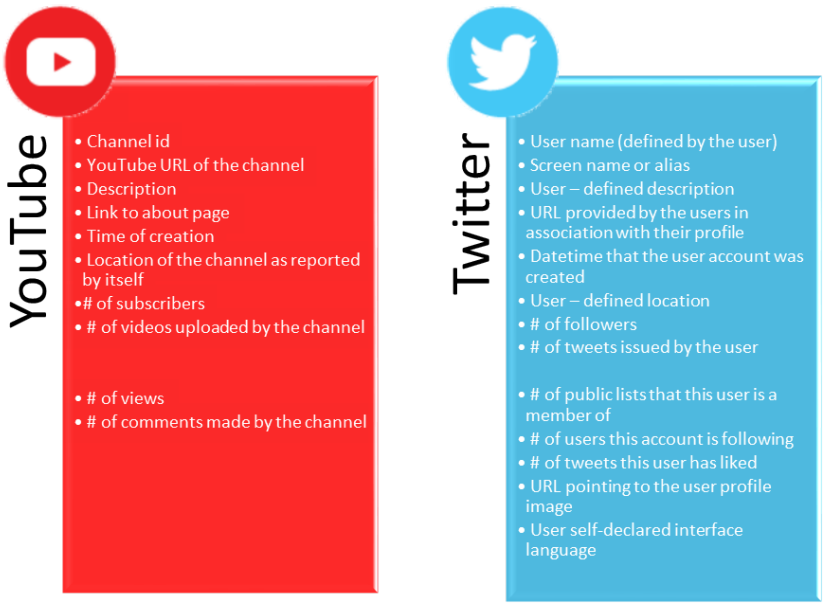
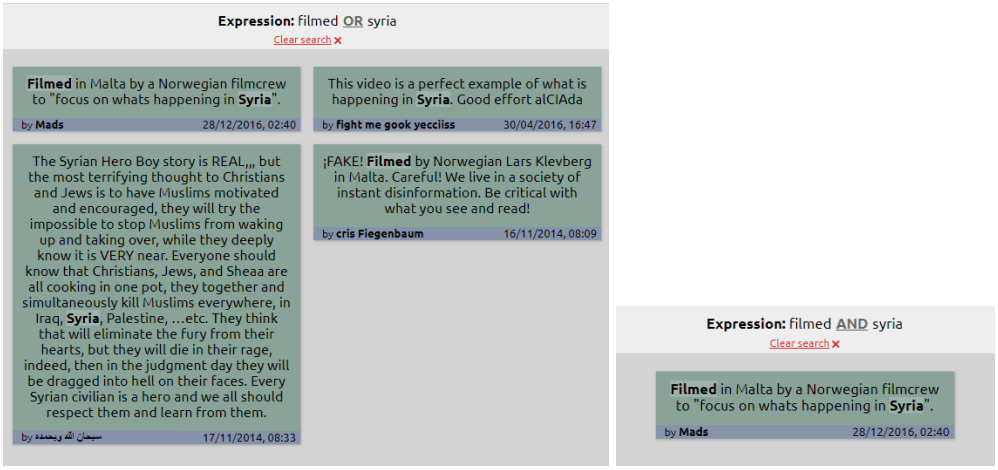


Figure 18: Channel/User verification indicators derived directly by the video platform API. Channel/User indicators refer to the information about channel (for YouTube) or user (for Twitter) that posted the video. Due to Facebook API limitation, the information about the page posted the video is not available.

accompanying text and the account that posted each of them. We compare feature distributions among fake and real data and present the mean when normal distribution is followed, or median if not. To further evaluate the statistical significance of the differences between fake and real cases, we compare the mean values using the Welch t-test or the MannWhitneyWilcoxon test and report the associated p-values. The results of our analysis were published in (Papadopoulou et al., 2018). As a first indicator, video duration was considered of interest. We analysed separately the first video of each cascade and the overall videos of a cascade. Thus, for real videos the average duration of the first video of each cascade is 149, seconds and including its near-duplicates the average duration decreases to 124 seconds. On the other side, for the initial fake videos the average duration is 92 seconds (p $<10^{-3}$) and including the cascades 77 seconds (p $<10^{-3}$). The fake videos tend to be remarkably shorter than the real ones and this is also empirically confirmed taking into account that there are several cases of videos, which are manipulated for disinformation, where one or several fragments of the original one are cut to create an allegedly new video.

Concerning the video uploader, the analysis concerns only the YouTube channel and Twitter accounts (both Twitter native videos and tweets sharing a video) while Facebook pages are excluded since there is no available information due to Facebook API limitations. First, we examined the age of the channel/account posting the video. For YouTube and for real videos, the channel median age is 811 days prior to the day that the video was published while the corresponding value for fake videos is 425 (p $<10^{-3}$). The values for Twitter videos are 2,325 and 473 days (p = 10^{-3}), respectively. For Twitter shares (tweets containing the link on the initial videos), the difference is minor (1,297 days for real and 1,127 days for fake links) but, given the size of the samples, it is still statistically significant (p $<10^{-3}$). Overall, newly created channels and users are more likely to post fake videos than older accounts. Other interesting features to observe are the YouTube channel subscriber count which is 349 users for real videos and 92 (p $<10^{-3}$), a much lower value, for fake ones. The corresponding median follower count for Twitter accounts is 163,325 users, which is a particularly high value. The reason for this is that the dataset contains a relatively small number of native Twitter videos, originating from only 16 well-established accounts. Each of these accounts has a large number of followers, leading to the high median follower count we observe. In contrast, the median number of followers of the Twitter accounts which shared the video as a link is just 333. For fake videos, the median follower count is 2,855 (p $<10^{-3}$) for Twitter videos and 297 (p $<10^{-3}$) for Twitter shares, a significantly lower value.

Following the analysis of the user features, we ran a different analysis from a linguistic point of view on the text that accompanies the video. The video description for YouTube and Facebook videos and



(a) Logical operator OR (b) Logical operator AND

Figure 19: Search comments using user-defined keywords.

Table 20: Comment and Video - level features

Comment-level features	**Video-level features**	
	From video description	**From channel description**
Text length	Text length	Channel view count
#words	#words	Channel comment count
Contains question mark	Contains question mark	Channel subscriber count
Contains exclamation mark	Contains exclamation mark	Channel video count
Contains happy emoticon	Contains 1st person pronoun	
Contains sad emoticon	Contains 3rd person pronoun	
Contains 1st person pronoun	Number of uppercase characters	
Contains 2nd person pronoun	Number of positive sentiment words	
Contains 3rd person pronoun	Number of negative sentiment words	
Number of uppercase characters	Number of slang words	
Number of positive sentiment words	Has : symbol	
Number of negative sentiment words	#question marks	
Number of slang words	#exclamation marks	
Has : symbol		
Has ”please”		
#question marks		
#exclamation marks		
Readability score		

the post text for Twitter videos was submitted for language detection using the Python langdetect[25] library. For both real and fake videos, the relatively most frequent language is English. However, it is interesting to note that for fake videos the percentages are lower (Table 21). As presented in Table 21, a significant number of posts/descriptions, generally smaller for real videos than fake ones, did not contain enough text for automatic language recognition. Other extracted languages which appear at a lower frequency of less than 6% are Russian, Spanish, Arabic, German, Catalan, Japanese and Portuguese with the exception of Russian fake Twitter videos which are strikingly high (28 per cent). We selected a set of features to calculate for the post/description texts: i) Polarity, ii) Subjectivity, iii) Flesh reading ease ((Kincaid, Fishburne Jr, Rogers, & Chissom, 1975)) and iv) ColemanLiau index ((Coleman & Liau, 1975)). Python libraries were used to calculate the features, TextBlob library[26] for Polarity and Subjectivity and textstat[27] library for Flesh reading ease and ColemanLiau index. Despite the common assumption that fake posts have distinctive linguistic qualities, e.g. stronger sentiment and poorer language ((Castillo, Mendoza, & Poblete, 2011)), no noticeable differences were found between fake and real videos in our study.

An important piece of information that can be extracted from FVC-2018 is the temporal distribution

[25] https://pypi.org/project/langdetect/
[26] http://textblob.readthedocs.io/en/dev/
[27] https://pypi.org/project/textstat/

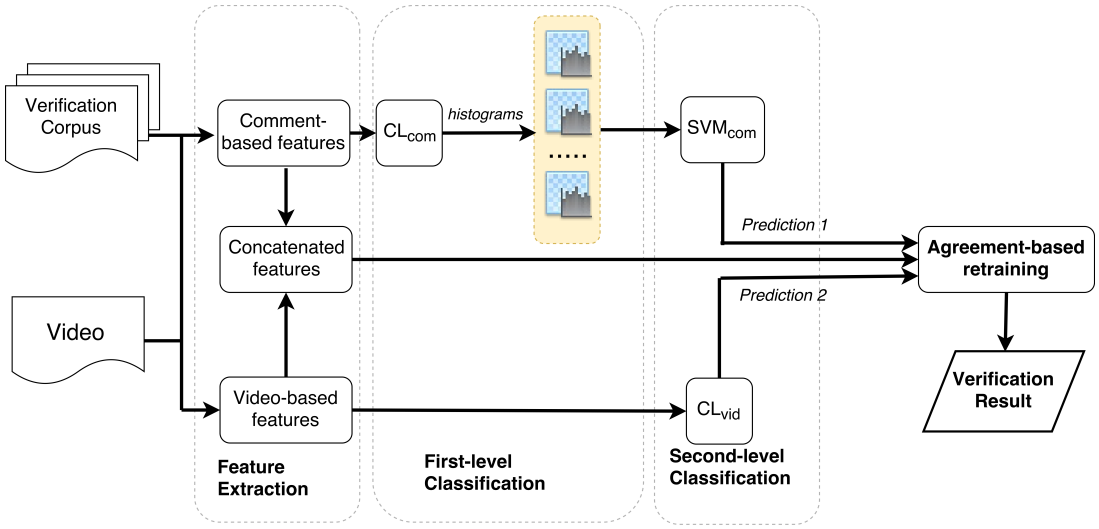


Figure 20: Overview of the automatic video verification approach.

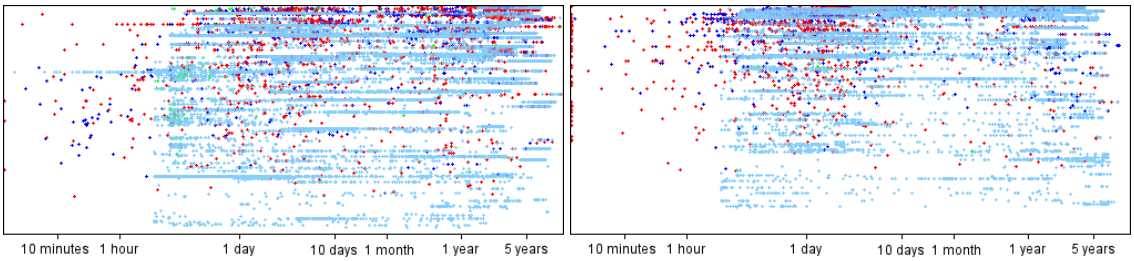
Table 21: Percentage of the videos with video title and description in English and with not enough text for detecting the language in relation to all videos of the FVC-2018.

		YouTube	Facebook	Twitter	Twitter shares
English	Fake videos	38%	28%	43%	52%
	Real videos	63%	41%	75%	62%
Not enough text	Fake videos	28%	51%	0	4%
	Real videos	13%	48%	0	5%

of video cascades. A timeline was created in Figure 21 to show how the near-duplicates of real and fake videos are distributed. At the vertical axis each line corresponds to a video cascade (i.e. the original video and its near-duplicates) while the horizontal axis is the time (in log scale) between the posting of the initial video and its near-duplicates. Each dot in Figure 21 represents a near-duplicate posted at that time. YouTube videos are marked with red, Facebook videos with blue, Twitter videos with green, and tweets sharing videos as YouTube links with light blue. For clarity, the videos are sorted from top to bottom from the most disseminated (i.e. having more near-duplicate instances) to the least disseminated ones. Observing Figure 21, the time range of near duplicate spread ranges from a few minutes after the initial video was posted, up to 10 years later. From an analysis perspective, the most important part is that of the difference between fake and real near-duplicate distributions. We can observe that, for real videos, the vast majority of near-duplicates are posted within 10 days of the original posting, whereas for fake videos, near-duplicates keep being posted for years.

There are relatively few near-duplicates of real videos posted on YouTube after 10 days from the original post, in contrast to fake videos where near-duplicates are posted at a much higher rate for a much longer interval. This observation also stands for Twitter shares. By calculating the median time difference between the initial video and its near-duplicate, we can numerically confirm this discrepancy. Specifically, for YouTube the media temporal distance is one day for real and 62 ($p < 10^{-3}$) for fake videos, which is a significant difference, and correspondingly the values for Facebook videos are 3 and 148 ($p < 10^{-3}$). Regarding Twitter videos, although the values are comparable, one and zero days for real and fake videos, respectively, the difference is still significant ($p = 3x10^{-2}$). Finally, for tweets sharing the initial video link the median distance is 6 days and 27 ($p < 10^{-3}$) days for real and fake videos, respectively.

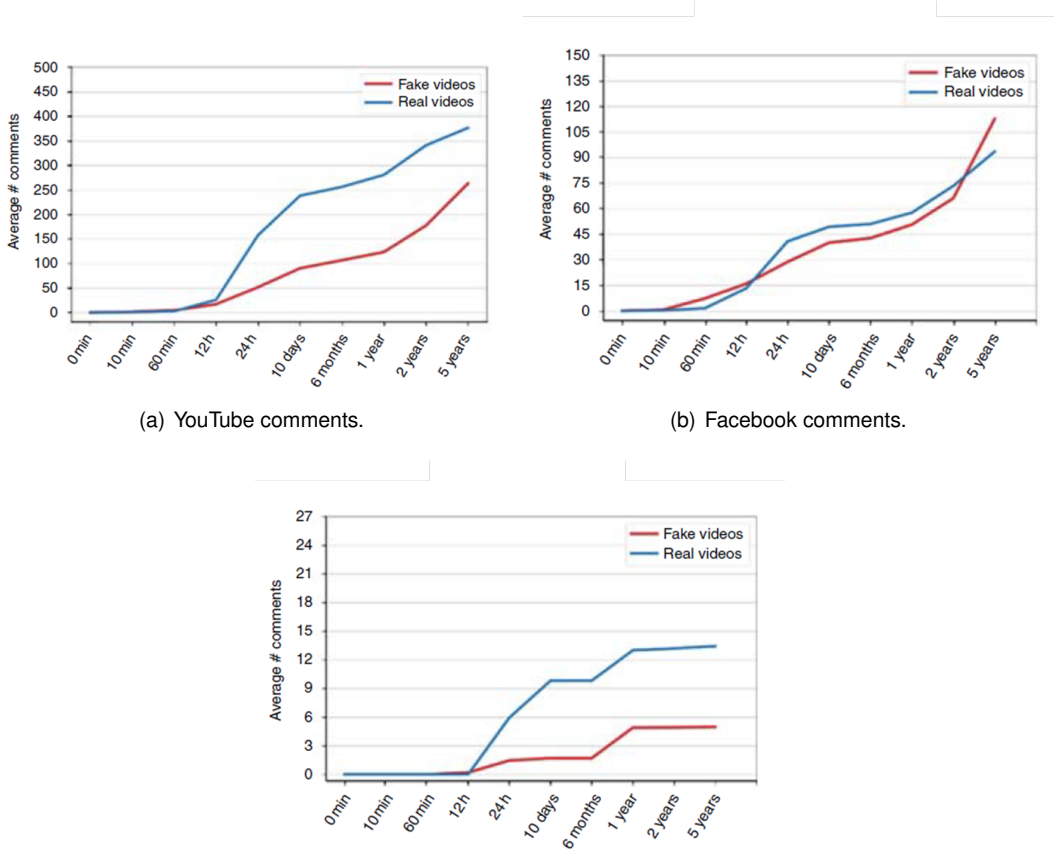
A potentially very helpful piece of information surrounding the videos is the comments or replies (for Twitter) which different users post below the video. As indicated by the usefulness of the "verification-related comments" feature of the CAA service, as well as the new comment search feature, comments can provide clues that can support or debunk the video content or claim. A total of 598,016 comments where found on the entire dataset for fake videos from which 491,639 comes from YouTube videos, 105,816 from Facebook videos and 561 from Twitter videos. Embedded Twitter videos seem to attract significantly fewer comments than other media, although, given the small number of Twitter videos in the dataset, this finding may not necessarily hold in general.



(a) Real videos cascades. (b) Fake video cascades.

Figure 21: Temporal distribution of video cascades. The near duplicates are from YouTube (red dots), Facebook (blue dots), Twitter (green dots) and Twitter shares (light blue dots).

Regarding the real videos, the comments on YouTube videos are 433,139, 86,326 for Facebook and 215 for Twitter videos. These comments are summed in a pool of 519,680 comments. In Figure 22, the cumulative average number of comments over time per video for the three video platforms is presented. In analyzing the comment distribution, we come to four major observations on the video comments; A major percentage of the comments, especially for the YouTube videos, appear in the first video of the cascade with 81 per cent for fake videos against 69 for real ones, and 22 against 9 per cent for Facebook, respectively; the number of comments between fake and real videos is significantly different with the fake ones clearly prevailing; there is a steep increase in the number of YouTube comments in real videos for a certain period (between 12 hours and 10 days after the video is posted) which consecutively tapers off. On the other side, fake videos maintain a steadier rate of accumulation, which ends up to be relatively steeper than for real videos, especially after one year from the posting.



(a) YouTube comments. (b) Facebook comments.

(c) Twitter replies.

Figure 22: Cumulative average number of comments/replies over time per video for the YouTube, Facebook and Twitter.

These observations show certain patterns that may potentially prove useful in separating real from fake videos, however they are essentially preliminary. The potential for analysis within the Fake Video Corpus 2018 goes far beyond these first-level observations, and is a significant aspect of the dataset contribution to research on disinformation and our efforts to limit its consequences.

7.4 Progress and evaluations during Year 3

During the final project year, the CAA component was evaluated both as a standalone tool and as integrated component of the InVID platform. The tests and evaluations focus on UGC of breaking news disseminated through YouTube, Facebook and Twitter. The component was evaluated in three out of four test cycles that ran during this year. The obtained feedback was carefully studied and used to guide our adaptations and improvements of the component during the third year of the project. Examples of the testers' suggestions and comments are listed below:

- We received several comments noting the absence of tweet replies when analyzing Twitter videos. However, this was due to time delays in displaying the replies. When using the Twitter API, the speed of collecting tweet replies depends on the account popularity and how old the tweet is. For old and popular tweets the replies collection might take a very long time. For that reason, we decided to inform the user that the replies will be returned with delay and let them decide whether it is worth waiting or not.
- With respect to requests for additional keywords and multi-language support, besides the updated verification-related keyword list and the translation into more languages, we implemented an additional feature where the user can define his/her own keywords and search the comments with them.
- Error handling and informative messages on the status of the execution are included upon request through the service API.
- Several objections on the performance of the new feature of video similarity to the debunked videos of FVC-2018 were reported at the first release of the feature. We investigated all reported cases and updated the feature in order to overcome the pointed limitations (i.e. when a video of the FCV-2018 is not available online anymore, we provide the video title and published date of the video along with links to its near duplicate instances if they exist.)

With respect to quantitative evaluations of automatic video contextual verification, in D3.2 we presented some preliminary evaluations on a first, small version of the FVC, consisting exclusively of a small number of YouTube videos without any near-duplicates. The FVC-2018 dataset, besides the larger number of real and fake cases and their near-duplicates, also contains videos from three different platforms (YouTube, Facebook and Twitter). The multi-platform nature of the dataset leads to some restrictions if we try to model all video items with similar descriptors. Specifically, the main restriction concerns the channel description feature of YouTube videos, which can not be applicable to Facebook videos due to the unavailability of this information. Moreover, regarding Twitter videos, although metadata about Twitter users are provided, YouTube channel features and Twitter user features do not completely match. Hence, the experiments that contain videos from all three video platforms exclude the channel description features of Table 20. The absence of these features might cause degradation of performance. For this reason, experiments using only the YouTube videos of the dataset (which is the most represented platform) are also included for comparison and for the evaluation of the potential of platform-specific models.

The dataset is split into training and test sets using 10-fold cross-validation to evaluate the performance of the approaches. The splits are carefully conducted so that all videos from the same cascade fall exclusively either into the training or into the evaluation set to avoid bias. A Radial Basis Function Support Vector Machine (RBF SVM) classifier is then trained and evaluated on the dataset. In Table 22, we present the results for the two individual features ("Comment credibility" and "Video metadata"), and the performance of the system when concatenating the two features ("Concatenated") which leads to an increase in performance. With respect to the different datasets, the evaluation metrics show a degradation of performance on the new dataset compared to the earlier experiments of (Papadopoulou, Zampoglou, Papadopoulos, & Kompatsiaris, 2017), both in the case of only using the first posted video in each cascade and when using all the videos. Observing the F1 scores, results are significantly lower than the first column in all cases. Furthermore, removing the channel-based features in order to merge

Table 22: Automatic verification results for the (Papadopoulou et al., 2017), the first video per cascade and the entire FVC-2018 in terms of 'Prec': precision, 'Rec': recall and 'F1': F1-measure

Papadopoulou	First video per cascade (YT only)	First video per cascade (YT + FB)	All videos in the cascade (YT only)	All videos in the cascade (YT + FB + TW)
Comment credibility				
Prec.: 0.88 Rec.: 0.74 F1: 0.79	Prec.: 0.91 Rec.: 0.53 F1: 0.67	Prec.: 0.97 Rec.: 0.52 F1: 0.68	Prec.: 0.96 Rec.: 0.64 F1: 0.77	Prec.: 0.94 Rec.: 0.60 F1: 0.73
Video metadata				
Prec.: 0.88 Rec.: 0.79 F1: 0.82	Prec.: 0.87 Rec.: 0.59 F1: 0.70	Prec.: 0.87 Rec.: 0.58 F1: 0.70	Prec.: 0.95 Rec.: 0.69 F1: 0.80	Prec.: 0.95 Rec.: 0.60 F1: 0.74
Concat.				
Prec.: 0.88 Rec.: 0.82 F1: 0.85	Prec.: 0.79 Rec.: 0.61 F1: 0.69	Prec.: 0.77 Rec.: 0.60 F1: 0.67	Prec.: 0.92 Rec.: 0.70 F1: 0.79	Prec.: 0.87 Rec.: 0.64 F1: 0.74
Agreement-based				
Prec.: 0.84 Rec: 0.88 F1: 0.86	Prec.: 0.58 Rec.: 0.93 F1: 0.71	Prec.: 0.53 Rec.: 0.98 F1: 0.70	Prec.: 0.70 Rec.: 0.96 F1: 0.80	Prec.: 0.61 Rec.: 0.96 F1: 0.74
Agreement-based with retraining				
Prec.: 0.77 Rec.: 0.86 F1: 0.81	Prec.: 0.57 Rec.: 0.92 F1: 0.70	Prec.: 0.54 Rec.: 0.98 F1: 0.69	Prec.: 0.69 Rec.: 0.96 F1: 0.80	Prec.: 0.60 Rec.: 0.97 F1: 0.74
Ideal fusion				
Prec.: 1.00 Rec.: 0.83 F1: 0.90	Prec.: 0.64 Rec.: 0.99 F1: 0.79	Prec.: 0.56 Rec.: 0.99 F1: 0.71	Prec.: 0.73 Rec.: 0.99 F1: 0.84	Prec.: 0.64 Rec.: 0.99 F1: 0.78

videos from the three compatible video platforms leads to significantly reduced performance. Finally, the theoretical ideal fusion between the two features does increase performance in all cases, but still the results are much lower than when using the smaller dataset of (Papadopoulou et al., 2017). This is an indication that the new version of the Fake Video Corpus is much more representative of the real-world problem, and as a result much more challenging for automatic verification algorithms.

7.5 API layer and integration with InVID

The CAA component is implemented as a REST service using the Java Spring Framework. Some of its internal processes are implemented separately as microservices using Python for better performance and scalability. Moreover in order to improve the portability and maintenance, the CAA component has been dockerized. With regards to the microservices, i) the ID extraction of the submitted URL is deployed as a standalone microservice, accepts the video URL and returns the video id and the name of the video platform that the submitted video belongs to, ii) a second microservice manages the task of collecting the tweets that share the submitted video URL, and iii) the new functionality of assigning credibility scores to the submitted videos is deployed as a third microservice. There are two GET requests for triggering the CAA component and two GET request for obtaining the responses, listed in Table 24. An additional optional parameter is included at the 'Video verify' service, named `twtimeline`. The default value of the parameter is 0 and indicates that the twitter timeline will be created with only the tweet IDs and no additional information. Otherwise, additional information is included, such as the "fake" or "real" annotation for each tweet in the timeline, assigned using the Tweet Verification Assistant tool. For a complete report the value should be set to 1. Moreover, a POST request is implemented as presented in Table 24. It accepts a JSON Object containing the video metadata, extracts a feature using the provided metadata and returns a calculated credibility score based on a pre-trained model.

A limited edition of the CAA API is integrated in the InVID plugin, while the entire range of the CAA API capabilities is being exploited by the Verification App. In both cases client communicates with the CAA by the REST API calls of Table 23. By the end of the project, the component will have also been integrated in the InVID Dashboard, which will use the 'Credibility' scores in order to rank the large number of videos that it collects.

Table 23: GET Calls exposed by the Context Aggregation and Analysis module API.

Service	URL	Parameter	Notes
Video verify	`/verify_videoV3`	`url=`<image url> `reprocess=`<1 or 0> `twtimeline=`<1 or 0>	YouTube, Facebook and Twitter URL
Weather conditions	`/weatherV3`	`location=`<Place> `time=`<Unix timestamp>	
Get metadata report	`/get_verificationV3`	`url=`<image url>	
Get twitter timeline	`/get_twverificationV3`	`url=`<image url> `reduced=`<1 or 0> `twtimeline=`<1 or 0>	

Table 24: POST Calls exposed by the Context Aggregation and Analysis module API.

Service	URL	Parameter	Notes
Credibility score	`/credibility`	`json object`	YouTube, Facebook and Twitter metadata

8 Impact and outlook

In this document, we presented our progress in all WP3 Tasks during the final year of the project. We presented our extension of the Fake Video Corpus into a rich collection of fake and real UGC videos, accompanied by their near-duplicates as collected from a number of social media outlets. FVC-2018 is an important outcome of InVID, which played an important role in the evaluations of project components, and will hopefully enable further research in the area of online disinformation beyond the end of the project.

With respect to datasets, the FIVR-200K dataset presented in Section 4 is another important contribution to the field, not only by providing a large scale dataset of real-world cases, but also by providing annotations for Fine-grained Video Retrieval, besides Near-duplicate video retrieval. Finally, the Lenses dataset presented in D3.2 is also a large-scale dataset further providing the advantage of multiple viewpoints. The contributions of all these datasets in their individual fields reflects the spirit of InVID, that is, maintaining focus on real-world application, openness and systematic large-scale benchmarking.

The WP3 components offered a number of individual contributions to the state of the art in different verification/retrieval problems. In each problem, achieving the desired level of accuracy required targeted advances and improvements over existing technologies, and the adaptation and combination of established methods, to move beyond the limitations of the field of multimedia verification at the time when InVID was beginning. This has led to a number of innovative components, each implementing novel solutions advancing the state of the art, but also integrated in the InVID platform and smoothly working together with other InVID components.

Video forensics. Most of the work conducted in this task remains confidential. With respect to the part of the work that we published, we designed and evaluated a method for producing single-value tampering probability estimates (i.e. tampering detection) by analyzing certain forensics filter outputs using deep convolutional neural networks. During the third year, this method was extended and improved. Beyond the end of the project, we expect the tools to slowly begin penetrating the market, and expertise in interpreting the filter outputs to be developed, establishing new standards in video verification. To the extent that InVID moves into financial viability, the algorithm improvements will continue, especially with respect to the automatic verification algorithms, which are a much needed tool in the field.

Near-duplicate detection. With respect to Near-duplicate Video Retrieval, the InVID NDVR component contributed a significant leap in algorithm accuracy and retrieval capabilities with respect to the state of the art, and furthermore, allowed us to move beyond simple video duplicates, to automatically detecting shots that depict the same moment from different viewpoints, or even shots that depict different moments from the same event. Within InVID, we were able to collect a very large dataset of related videos, and provide large-scale annotations with respect to video associations. The FIVR-200K dataset makes possible the training and evaluation of near-duplicate retrieval algorithms both in typical near-duplicate retrieval tasks, and with respect to more fine-grained retrieval challenges (such as detecting videos from the same event, or partial duplicates). The superiority of the proposed algorithm to the state of the art, and its efficiency have made it an important component of InVID. In parallel, we have been constantly indexing new videos sourced from the InVID Dashboard, which has led to a very large index of relevant videos against which to compare new videos. Thus, the integrated tool can provide significant assistance to investigators in detecting video reposts. Beyond the end of the project, the index will continue to expand, which will increase the capacity of the component in detecting reposts, which are currently the most common type of video fakes.

Logo detection. The initial attempts to develop a logo detection component followed the state-of-the-art at the beginning of the project, which suggested that, in the absence of a very large annotated training set, a traditional keypoint-based method should be followed. During the course of the project, a novel method for training deep learning models was devised, by generating large amounts of synthetic training data. This led to a more scalable and robust system during the second year of the project, which was further refined during the third year of the project. Furthermore, an evaluation dataset was created and used for evaluations. The addition of new logos, submitted by users, is leading to an extended service coverage, which will continue past the end of the project. Despite not being a verification tool per se, logo detection is very helpful in identifying valuable cues in the content and thus guiding investigators to look for disinformation.

Location detection. The InVID Location Detection component has been constantly improving since the beginning of the project, and it has led to a very robust algorithm, outperforming the state of the art in benchmark evaluations. It has also led to the creation of an array of additional tools, including evaluation frameworks, datasets, and wrappers. Due to the integration of the Location Detection component in the

CAA module, which was part of the InVID plugin since the first steps of the project, the component has seen heavy use already. The ecosystem of tools developed around the component ensure that it will remain relevant and will continue to improve and provide state-of-the-art performance in the problem of location detection.

Context Aggregation and Analysis. The Context Aggregation and Analysis module began as a tool intended to collect available information from social media APIs, restructure it, analyze it, and present it to human analysts for verification. The early integration of the module to the InVID Plugin and the Verification Application allowed us to collect extensive user feedback, which led to the development of new features and adjustments to existing ones. Each version of the module included changes guided by user feedback, as well as an extension into new platforms, languages, and functionalities. In parallel, the aggregated data from the module usage allowed us to collect newsworthy fake and real videos to populate the Fake Video Corpus, which in its final version has become a large-scale dataset which we expect to have a significant impact on the field. Furthermore, besides the manual verification component, effort was dedicated to advancing the state of the art in automatic verification. The published experiments using contextual features and various versions of the Fake Video Corpus have contributed to advancements in the field which could, in the future, lead to the development of automatic verification systems able to assist investigators by providing quantitative estimates of an item's credibility.

Finally, the integrated platform where all these components are linked, provides a state-of-the-art array of tools, more complete than any other tool currently available in the market. The contributions made in the various WP3 tasks have started making an impact on academic research, but have also been firmly grounded on real-world use, and have already been used by investigators in operational settings. The resulting services and tools are expected to continue to make an impact, both as potential products (integrated or separate), and as a stepping stone for future research in the area.

In the three years that InVID ran, the problem of disinformation -and in particular, disinformation using video presented as newsworthy UGC- has only grown. Thanks to our efforts during the project, we were able to follow several real-world cases unfold, and thanks to the high uptake of the services, we had the opportunity to watch the evolution of the problem and the response of the verification community. We provided novel tools, honed them to the needs of the users, and hopefully leaving a positive mark in the ongoing struggle between disinformation and verification.

However, we recognize that the problem is far from solved, and it seems that, with every advance in the arsenal of investigators, new challenges arise. The recent challenge of "Deep Fakes", which arose during the course of InVID, and which was not anticipated when conceiving the project, is indicative of the fact that no single technology can permanently solve the problem of disinformation in its entirety. The landscape of the task is constantly shifting, and new challenges will continue to arise. Despite not being able to dedicate extensive efforts to developing novel forensics techniques targeted specifically on Deep Fakes, we incorporated these new challenges in our analysis. Within InVID and WP3 in particular, we dedicated our efforts to adapt the scope of InVID to cover such cases, and indeed, the CAA component is intended to work regardless of content, and thus should also be helpful to spot Deep Fakes. The tampering localization algorithm developed during Year 3 as part of the Video Forensics component could also be part of the solution to the problem, given enough relevant training data. Since, in fact, no known real-world cases of Deep Fake have entered the news landscape so far, the challenge is still limited to a theoretical level. While the scope of InVID has not reached this far into the future, and further work should be (and surely will be) devoted to the task, InVID has provided solid ground for such future work.

References

Baraldi, L., Douze, M., Cucchiara, R., & Jégou, H. (2018). Lamv: Learning to align and match videos with kernelized temporal layers. In *Ieee/cvf conference on computer vision and pattern recognition.*

Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., & Wolf, H. C. (1977). *Parametric correspondence and chamfer matching: Two new techniques for image matching* (Tech. Rep.). SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Bird, S., & Loper, E. (2004). Nltk: the natural language toolkit. In *Proceedings of the acl 2004 on interactive poster and demonstration sessions* (p. 31).

Boididou, C., Papadopoulos, S., Zampoglou, M., Apostolidis, L., Papadopoulou, O., & Kompatsiaris, Y. (2018). Detection and visualization of misleading content on twitter. *International Journal of Multimedia Information Retrieval*, *7*(1), 71–86.

Braşoveanu, A. M. P., Rizzo, G., Kuntschick, P., Weichselbraun, A., & Nixon, L. J. (2018, may). Framing named entity linking error types. In N. Calzolari et al. (Eds.), *Proceedings of the eleventh international conference on language resources and evaluation (lrec 2018)* (p. 266-271). Paris, France: European Language Resources Association (ELRA). Retrieved from `http://www.lrec-conf.org/proceedings/lrec2018/summaries/612.html`

Cai, Y., Yang, L., Ping, W., Wang, F., Mei, T., Hua, X.-S., & Li, S. (2011). Million-scale near-duplicate video retrieval system. In K. S. Candan, S. Panchanathan, B. Prabhakaran, H. Sundaram, W. chi Feng, & N. Sebe (Eds.), *Proceedings of the 19th international conference on multimedia 2011, scottsdale, AZ, USA, november 28 - december 1, 2011* (pp. 837–838). ACM. Retrieved from `http://doi.acm.org/10.1145/2072298.2072484`

Cao, J., Guo, J., Li, X., Jin, Z., Guo, H., & Li, J. (2018). Automatic rumor detection on microblogs: A survey. *arXiv preprint arXiv:1807.03505*.

Castillo, C., Mendoza, M., & Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th international conference on world wide web* (pp. 675–684).

Chen, S., Tan, S., Li, B., & Huang, J. (2016, November). Automatic detection of object-based forgery in advanced video. *IEEE Trans. on Circuits Systems and Video Technologies*, *26*(11), 2138-2151.

Chou, C.-L., Chen, H.-T., & Lee, S.-Y. (2015). Pattern-based near-duplicate video retrieval and localization on web-scale videos. *IEEE Trans. Multimedia*, *17*(3), 382–395. Retrieved from `http://dx.doi.org/10.1109/TMM.2015.2391674`

Coates, A., & Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 921–928).

Coleman, M., & Liau, T. L. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, *60*(2), 283.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Daiber, J., Jakob, M., Hokamp, C., & Mendes, P. N. (2013). Improving Efficiency and Accuracy in Multilingual Entity extraction. In M. Sabou, E. Blomqvist, T. D. Noia, H. Sack, & T. Pellegrini (Eds.), *I-SEMANTICS 2013 - 9th international conference on semantic systems, ISEM '13, graz, austria, september 4-6, 2013* (pp. 121–124). ACM. Retrieved from `http://dl.acm.org/citation.cfm?id=2506182` doi: 10.1145/2506182.2506198

D'Amiano, L., Cozzolino, D., Poggi, G., & Verdoliva, L. (2015). Video forgery detection and localization based on 3d patchmatch. In *Ieee int. conf. on multimedia expo workshop (icmew).*

Dong, Q., Yang, G., & Zhu, N. (2012). A MCEA based passive forensics scheme for detecting frame based video tampering. *Digital Investigation*, 151-159.

Ehrmann, M., Jacquet, G., & Steinberger, R. (2017). Jrc-names: Multilingual entity name variants and titles as linked data. *Semantic Web*, *8*(2), 283–295. Retrieved from `http://dx.doi.org/10.3233/SW-160228` doi: 10.3233/SW-160228

Elkasrawi, S., Dengel, A., Abdelsamad, A., & Bukhari, S. S. (2016). What you see is what you get? automatic image verification for online news content. In *Document analysis systems (das), 2016 12th iapr workshop on* (pp. 114–119).

Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., & Vrandecic, D. (2014). Introducing wikidata to the linked data web. In P. Mika et al. (Eds.), *The semantic web - ISWC 2014 - 13th international semantic web conference, riva del garda, italy, october 19-23, 2014. proceedings, part I* (Vol. 8796,

pp. 50–65). Springer. Retrieved from http://dx.doi.org/10.1007/978-3-319-11964-9_4 doi: 10.1007/978-3-319-11964-9_4

Francis-Landau, M., Durrett, G., & Klein, D. (2016). Capturing semantic similarity for entity linking with convolutional neural networks. In K. Knight, A. Nenkova, & O. Rambow (Eds.), *NAACL HLT 2016, the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies, san diego california, usa, june 12-17, 2016* (pp. 1256–1261). The Association for Computational Linguistics. Retrieved from http://aclweb.org/anthology/N/N16/N16-1150.pdf

Fridrich, J., & Kodovsky, J. (2012). Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, *7*(3), 868–882.

Guo, Y., Che, W., Liu, T., & Li, S. (2011). A graph-based method for entity linking. In *Fifth international joint conference on natural language processing, IJCNLP 2011, chiang mai, thailand, november 8-13, 2011* (pp. 1010–1018). Chiang Mai, Thailand: ACLCLP. Retrieved from http://aclweb.org/anthology/I/I11/I11-1113.pdf

Guzman-Zavaleta, Z. J., & Feregrino-Uribe, C. (2018). Partial-copy detection of non-simulated videos using learning at decision level. *Multimedia Tools and Applications*, 1–20.

Hachey, B., Nothman, J., & Radford, W. (2014). Cheap and easy entity evaluation. In *Proceedings of the 52nd annual meeting of the association for computational linguistics, ACL 2014, june 22-27, 2014, baltimore, md, usa, volume 2: Short papers* (pp. 464–469). The Association for Computer Linguistics. Retrieved from http://aclweb.org/anthology/P/P14/P14-2076.pdf

Hachey, B., Radford, W., & Curran, J. R. (2011). Graph-based named entity linking with wikipedia. In *Web information system engineering - WISE 2011 - 12th international conference, sydney, australia, october 13-14, 2011. proceedings* (Vol. 6997, pp. 213–226). Berlin, Germany: Springer. Retrieved from http://dx.doi.org/10.1007/978-3-642-24434-6\char‘_16 doi: 10.1007/978-3-642-24434-6\char‘_16

Hao, Y., Mu, T., Hong, R., Wang, M., An, N., & Goulermas, J. Y. (2017). Stochastic multiview hashing for large-scale near-duplicate video retrieval. *IEEE Transactions on Multimedia*, *19*(1), 1–14.

Heravi, B. R., & Harrower, N. (2016). Twitter journalism in ireland: Sourcing and trust in the age of social media. *Information, Communication & Society*, *19*(9), 1194–1213.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., ... Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the 2011 conference on empirical methods in natural language processing, EMNLP 2011, 27-31 july 2011, john mcintyre conference centre, edinburgh, uk, A meeting of sigdat, a special interest group of the ACL* (pp. 782–792). ACL. Retrieved from http://www.aclweb.org/anthology/D11-1072

Huang, L., Sil, A., Ji, H., & Florian, R. (2017). Improving slot filling performance with attentive neural networks on dependency structures. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 conference on empirical methods in natural language processing, EMNLP 2017, copenhagen, denmark, september 9-11, 2017* (pp. 2588–2597). Association for Computational Linguistics. Retrieved from https://aclanthology.info/papers/D17-1274/d17-1274

Jha, K., Röder, M., & Ngomo, A. N. (2017). All that glitters is not gold - rule-based curation of reference datasets for named entity recognition and entity linking. In E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler, & O. Hartig (Eds.), *The semantic web - 14th international conference, ESWC 2017, portorož, slovenia, may 28 - june 1, 2017, proceedings, part I* (Vol. 10249, pp. 305–320). Retrieved from https://doi.org/10.1007/978-3-319-58068-5_19 doi: 10.1007/978-3-319-58068-5_19

Ji, H., & Nothman, J. (2016). Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end kbp. In *Eighth text analysis conference (tac).* NIST. Retrieved from https://tac.nist.gov/publications/2016/additional.papers/TAC2016.KBP_Entity_Discovery_and_Linking_overview.proceedings.pdf

Ji, H., Pan, X., Zhang, B., Nothman, J., Mayfield, J., McNamee, P., & Costello, C. (2016). Overview of TAC-KBP2015 Tri-lingual Entity Discovery and Linking. In *Eighth text analysis conference (tac).* NIST.

Ji, H., Pan, X., Zhang, B., Nothman, J., Mayfield, J., McNamee, P., & Costello, C. (2017). Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the 2017 text analysis conference, TAC 2017, gaithersburg, maryland, usa, november 13-14, 2017.* NIST. Retrieved from https://tac.nist.gov/publications/2017/additional.papers/TAC2017.KBP_Entity_Discovery_and_Linking_overview.proceedings.pdf

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., ... Darrell, T. (2014). Caffe:

Convolutional architecture for fast feature embedding. In K. A. Hua, Y. Rui, R. Steinmetz, A. Hanjalic, A. Natsev, & W. Z. 0001 (Eds.), *Proceedings of the ACM international conference on multimedia, MM '14, orlando, FL, USA, november 03 - 07, 2014* (pp. 675–678). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=2647868

Jiang, Y.-G., Jiang, Y., & Wang, J. (2014). Vcdb: a large-scale database for partial copy detection in videos. In *European conference on computer vision* (pp. 357–371).

Kincaid, J. P., Fishburne Jr, R. P., Rogers, R. L., & Chissom, B. S. (1975). Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.

Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., & Kompatsiaris, I. (2018). FIVR: Fine-grained Incident Video Retrieval. *arXiv preprint arXiv:1809.04094*.

Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., & Kompatsiaris, Y. (2017a). Near-duplicate video retrieval by aggregating intermediate cnn layers. In *International conference on multimedia modeling* (pp. 251–263).

Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., & Kompatsiaris, Y. (2017b, Oct). Near-duplicate video retrieval with deep metric learning. In *Web-scale vision and social media (iccv-w).*

Kraaij, W., & Awad, G. (2011). TRECVID 2011 content-based copy detection: Task overview. *Online Proceedings of TRECVid 2010*.

Labartino, D., Bianchi, T., Rosa, A. D., Fontani, M., Vazquez-Padin, D., & Piva, A. (2013). Localization of forgeries in mpeg-2 video through gop size and dq analysis. In *Ieee 15th int. workshop on multimedia and signal processing (mmsp)* (p. 494-499).

Law-To, J., Joly, A., & Boujemaa, N. (2007). *Muscle-VCD-2007: a live benchmark for video copy detection.*

Li, L., Wang, X., Wang, G., & Hu, G. (2013). Detecting removed object from video with stationary background. In *Proc. of the 11th int. conf. on digital forensics and watermarking (wdw)* (p. 242-252).

Lin, C.-S., & Tsay, J.-J. (2014). A passive approach for effective detection and localization of region-level video forgery with spatio-temporal coherence analysis. *Digit. Investig.*, *11*(2), 120-140.

Liu, M., Po, L.-M., Rehman, Y. A. U., Xu, X., Li, Y., & Feng, L. (2018). Video copy detection by conducting fast searching of inverted files. *Multimedia Tools and Applications*, 1–24.

Montserrat, D. M., Lin, Q., Allebach, J., & Delp, E. J. (2018). Logo detection and recognition with synthetic images. *Electronic Imaging*, *2018*(10), 337–1.

Moro, A., Raganato, A., & Navigli, R. (2014). Entity linking meets word sense disambiguation: a unified approach. *TACL*, *2*, 231–244. Retrieved from https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/291

Odoni, F., Kuntschik, P., Brasoveanu, A. M. P., & Weichselbraun, A. (2018). On the importance of drill-down analysis for assessing gold standards and named entity linking performance. In A. Fensel et al. (Eds.), *Proceedings of the 14th international conference on semantic systems, SEMANTICS 2018, vienna, austria, september 10-13, 2018* (Vol. 137, pp. 33–42). Elsevier. Retrieved from https://doi.org/10.1016/j.procs.2018.09.004 doi: 10.1016/j.procs.2018.09.004

Pandey, R., Singh, S., & Shukla, K. (2014). Passive copy-move forgery detection in videos. In *Ieee int. conf. on computer and communication technology (iccct)* (p. 301-306).

Papadopoulou, O., Zampoglou, M., Papadopoulos, S., & Kompatsiaris, I. (2018). A corpus of debunked and verified user-generated videos. *Online Information Review*.

Papadopoulou, O., Zampoglou, M., Papadopoulos, S., & Kompatsiaris, Y. (2017). Web video verification using contextual cues. In *Proceedings of the 2nd international workshop on multimedia forensics and security* (pp. 6–10).

Pasquini, C., Brunetta, C., Vinci, A. F., Conotter, V., & Boato, G. (2015). Towards the verification of image integrity in online news. In *Multimedia & expo workshops (icmew), 2015 ieee international conference on* (pp. 1–6).

Pittaras, N., Markatopoulou, F., Mezaris, V., & Patras, I. (2017). Comparison of fine-tuning and extension strategies for deep convolutional neural networks. In *International conference on multimedia modeling* (pp. 102–114).

Piva, A. (2013). An overview on image forensics. *ISRN Signal Processing*, 1-22.

Plu, J., Rizzo, G., & Troncy, R. (2016). Enhancing entity linking by combining NER models. In H. Sack, S. Dietze, A. Tordai, & C. Lange (Eds.), *Semantic web challenges - third semwebeval challenge at ESWC 2016, heraklion, crete, greece, may 29 - june 2, 2016, revised selected papers* (Vol. 641,

pp. 17–32). Springer. Retrieved from https://doi.org/10.1007/978-3-319-46565-4_2 doi: 10.1007/978-3-319-46565-4_2

Rauchfleisch, A., Artho, X., Metag, J., Post, S., & Schäfer, M. S. (2017). How journalists verify user-generated content during terrorist crises. analyzing twitter communication during the brussels attacks. *Social Media+ Society*, *3*(3), 2056305117717888.

Ren, S., He, K., Girshick, R., & Sun, J. (2015a). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

Ren, S., He, K., Girshick, R. B., & Sun, J. (2015b). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, *abs/1506.01497*. Retrieved from http://arxiv.org/abs/1506.01497

Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2018). Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*.

Scharl, A., Weichselbraun, A., Göbel, M., Rafelsberger, W., & Kamolov, R. (2016). Scalable Knowledge Extraction and Visualization for Web Intelligence. In *49th hawaii international conference on system sciences (hicss-2016)* (pp. 3749–3757). IEEE.

Scharl, A., Weichselbraun, A., Göbel, M. C., Rafelsberger, W., & Kamolov, R. (2016). Scalable knowledge extraction and visualization for web intelligence. In *49th hawaii international conference on system sciences, HICSS 2016, koloa, hi, usa, january 5-8, 2016* (pp. 3749–3757). New York, NY, USA: ACM. Retrieved from https://doi.org/10.1109/HICSS.2016.467 doi: 10.1109/HICSS.2016.467

Sitara, K., & Mehtre, B. M. (2016). Digital video tampering detection: An overview of passive techniques. *Digital Investigation*, *18*, 8-22.

Song, J., Yang, Y., Huang, Z., Shen, H. T., & Hong, R. (2011). Multiple feature hashing for real-time large scale near-duplicate video retrieval. In K. S. Candan, S. Panchanathan, B. Prabhakaran, H. Sundaram, W. chi Feng, & N. Sebe (Eds.), *Proceedings of the 19th international conference on multimedia 2011, scottsdale, AZ, USA, november 28 - december 1, 2011* (pp. 423–432). ACM. Retrieved from http://doi.acm.org/10.1145/2072298.2072354

Stencel, M. (n.d.). *International fact checking gains ground, duke census finds. duke reporters' lab, duke university, durham, nc, feb. 28, 2017.*

Su, H., Zhu, X., & Gong, S. (2017). Deep learning logo detection with data expansion by synthesising context. In *Applications of computer vision (wacv), 2017 ieee winter conference on* (pp. 530–539).

Su, L., Huang, T., & Yang, J. (2015). A video forgery detection algorthm based on compressive sensing. *Multimedia Tools and Applications*, *74*, 6641-6656.

Su, Y., & Xu, J. (2010). Detection of double compression in mpeg-2 videos. In *Ieee 2nd international workshop on intelligent syetems and application (isa).*

Subramanyam, A., & Emmanuel, S. (2012). Video forgery detection using hog features and compression properties. In *Ieee 14th int. workshop on multimedia and signal processing (mmsp)* (p. 89-94).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Aaai* (Vol. 4, p. 12).

Usbeck, R., Ngomo, A. N., Röder, M., Gerber, D., Coelho, S. A., Auer, S., & Both, A. (2014). AGDISTIS - agnostic disambiguation of named entities using linked open data. In T. Schaub, G. Friedrich, & B. O'Sullivan (Eds.), *ECAI 2014 - 21st european conference on artificial intelligence, 18-22 august 2014, prague, czech republic - including prestigious applications of intelligent systems (PAIS 2014)* (Vol. 263, pp. 1113–1114). IOS Press. Retrieved from https://doi.org/10.3233/978-1-61499-419-0-1113 doi: 10.3233/978-1-61499-419-0-1113

Usbeck, R., Röder, M., Ngomo, A. N., Baron, C., Both, A., Brümmer, M., ... Wesemann, L. (2015). GERBIL: General entity annotator benchmarking framework. In *Proceedings of the 24th international conference on world wide web, WWW 2015* (pp. 1133–1143). Retrieved from http://doi.acm.org/10.1145/2736277.2741626 doi: 10.1145/2736277.2741626

Wang, L., Bao, Y., Li, H., Fan, X., & Luo, Z. (2017). Compact cnn based video representation for efficient video copy detection. In *International conference on multimedia modeling* (pp. 576–587).

Wang, W., & Farid, H. (2007). Exposing digital forgeries in interlaced and deinterlaced video. *IEEE trans. on Information Forensics and Security*, *2*(3), 438-449.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.

Weichselbraun, A., Kuntschik, P., & Braşoveanu, A. M. P. (2018). Mining and leveraging background knowledge for improving named entity linking. In *Proceedins of the 8th international conference on web intelligence, mining and semantics (wims 2018).* Novi Sad, Serbia: ACM. Retrieved from http://doi.acm.org/10.1145/3227609.3227670 doi: 10.1145/3227609.3227670

Wu, X., Hauptmann, A. G., & Ngo, C.-W. (2007). Practical elimination of near-duplicates from web video search. In R. Lienhart, A. R. Prasad, A. Hanjalic, S. Choi, B. P. Bailey, & N. Sebe (Eds.), *Proceedings of the 15th international conference on multimedia 2007, augsburg, germany, september 24-29, 2007* (pp. 218–227). ACM. Retrieved from `http://dl.acm.org/citation.cfm?id=1291233`

Wu, Y., Jiang, X., Sun, T., & Wang, W. (2014). Exposing video inter-frame forgery based on velocity field consistency. In *Icassp.*

Xu, J., Su, Y., & liu, Q. (2013). Detection of double mpeg-2 compression based on distribution of dct coefficients. *Int. J. Pattern Recognition and Artificial Intelligence*, *27*(1).

Xu, Y., Monrose, F., Frahm, J.-M., et al. (2017). Caught red-handed: Toward practical video-based subsequences matching in the presence of real-world transformations. In *Computer vision and pattern recognition workshops (cvprw), 2017 ieee conference on* (pp. 1397–1406).

Yao, Y., Shi, Y., Weng, S., & Guan, B. (2017). Deep learning for detection of object-based forgery in advanced video. *Symmetry*, *10*(1), 3.

Zampoglou, M., Markatopoulou, F., Mercier, G., Touska, D., Apostolidis, E., Papadopoulos, S., ... Kompatsiaris, I. (2019). Detecting tampered videos with multimedia forensics and deep learning. In *International conference on multimedia modeling* (pp. 374–386).

Zampoglou, M., Papadopoulos, S., & Kompatsiaris, Y. (2016). Large-scale evaluation of splicing localization algorithms for web images. *Multimedia Tools and Applications*, online first. doi: 10.1007/s11042-016-3795-2

Zhang, Z., Hou, J., Ma, Q., & Li, Z. (2015, 25 January). Efficient video frame insertion and deletion detection based on inconsistency of correlations between local binary pattern coded frames. *Security and Communicatin networks*, *8*(2).

