

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Video Similarity Measurement: Using Convolutional Neural Networks To Create Video Signatures

João Gabriel Marques Costa



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Luís Filipe Pinto de Almeida Teixeira

Company Supervisor: Alexandre Ulisses Silva

June 27, 2018



# **Video Similarity Measurement: Using Convolutional Neural Networks To Create Video Signatures**

**João Gabriel Marques Costa**

Mestrado Integrado em Engenharia Informática e Computação

June 27, 2018



# Abstract

The widespread of the Internet along with the appearance of a multitude of video sharing and streaming services has increased the amount of video data available dramatically. On the other hand, a big portion of the videos available online show a large degree of similarity between each other. In fact, YouTube alone is estimated to have over 859 million videos with 31.7% of them considered to be near-duplicates.

This ease of sharing and redistribution has many implications, such as decreasing the amount of useful storage space and increasing the risk of illegal distribution of copyrighted contents. As such, there has been an increasing interest in the research of near-duplicate video detection and retrieval techniques.

In this dissertation we propose a solution for measuring similarity between videos by using Convolutional Neural Networks, which have been successfully applied in a variety of image and video recognition tasks. We demonstrate that using semantic information extracted from videos by these networks can be useful in the context of video similarity by testing our solution in the task of near-duplicate video detection, using the CC\_WEB\_VIDEO dataset, and achieving an Average Precision of 0.974 competitive with the current state-of-the-art.



# Resumo

A constante disseminação da Internet pelo mundo, assim como o surgimento de vários serviços de partilha e streaming de vídeo, fez aumentar drasticamente a quantidade de dados de vídeo disponíveis. Por outro lado, uma grande parte destes vídeos mostram um grau de semelhança elevado entre eles. Só no YouTube é estimado que 31,7% dos mais de 859 milhões de vídeos disponíveis são cópias.

Esta facilidade de partilha e redistribuição gera vários problemas, tal como a diminuição de espaço de armazenamento livre e o aumento do risco de distribuição ilegal de conteúdos protegidos por direitos de autor. Como tal, tem se notado um crescente interesse em desenvolvimento de técnicas para deteção e procura de vídeos duplicados.

Nesta dissertação, propomos uma solução para medir a similaridade entre vídeos usando Convolutional Neural Networks, que têm sido aplicadas, com elevada taxa de sucesso, em várias tarefas de reconhecimento de vídeo e imagem. Demonstramos que o uso de informação semântica, extraída por estas redes, pode ser útil no contexto de similaridade entre vídeos testando a nossa solução em deteção de vídeos duplicados, usando o dataset CC\_WEB\_VIDEO, e obtendo uma Average Precision de 0.974 comparável com o estado da arte atual.





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	1
1.2	Motivation . . . . .	3
1.3	Objectives . . . . .	4
1.4	Dissertation Outline . . . . .	5
<b>2</b>	<b>Image and Video Recognition</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Object Detection . . . . .	8
2.3	Scene Recognition . . . . .	9
2.4	Technologies . . . . .	9
<b>3</b>	<b>Video Similarity</b>	<b>13</b>
3.1	Introduction . . . . .	13
3.2	Applications . . . . .	14
3.2.1	Copyright Protection . . . . .	15
3.2.2	Video Monitoring . . . . .	15
3.2.3	Video Re-ranking . . . . .	16
3.2.4	Video Recommendation . . . . .	16
3.3	Common Approaches . . . . .	16
3.3.1	Feature Extraction . . . . .	17
3.3.2	Signature Generation . . . . .	18
3.3.3	Signature Comparison . . . . .	19
<b>4</b>	<b>Proposed Solution</b>	<b>21</b>
4.1	Overview . . . . .	21
4.2	Object Detection and Scene Recognition . . . . .	22
4.3	Object Tracking . . . . .	23
4.4	Shot Boundary Detection . . . . .	28
4.5	Feature Extraction . . . . .	29
4.6	Signature Generation and Comparison . . . . .	30
<b>5</b>	<b>Experimental Results</b>	<b>33</b>
5.1	Initial Observations . . . . .	33
5.1.1	Semantic Similarity . . . . .	33
5.1.2	Visual Transformations . . . . .	35
5.1.3	Problems . . . . .	37
5.2	Benchmarks . . . . .	38

## CONTENTS

5.2.1	Feature Extraction . . . . .	39
5.2.2	Signature Generation and Comparison . . . . .	41
5.3	Near-Duplicate Video Detection . . . . .	42
<b>6</b>	<b>Conclusions and Future Work</b>	<b>47</b>
	<b>References</b>	<b>49</b>

# List of Figures

1.1	Computer Vision driven by advancements in multiple fields . . . . .	2
1.2	ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Results . . . . .	2
1.3	Example images from the classification task in ILSVRC . . . . .	3
2.1	Example output from an object detector . . . . .	8
2.2	Some examples available on the official Places2 Demo . . . . .	10
3.1	The two types of similarity: (a) and (b) are visually similar while (c) and (d) are semantically similar . . . . .	14
3.2	Overview of NDVR and NDVD processes . . . . .	18
4.1	Overview of implemented system . . . . .	23
4.2	Example object detection outputs from the implemented YOLOv3 network . . . . .	24
4.3	Visual definition of Intersection over Union (IoU) . . . . .	25
4.4	Overview of the whole object tracking process . . . . .	26
4.5	Example tracking with occlusion . . . . .	26
4.6	Example tracking person across the screen with various occlusions and overlaps . . . . .	27
4.7	Various types of possible video annotations. (a) and (b) show scene information and tracked objects. The red square in the lower right corner of (c) indicates the current frame is a shot boundary . . . . .	30
4.8	Visual representation of a shot signature (feature vector) . . . . .	31
5.1	All 4 shots have a similarity value above 93% between each other . . . . .	34
5.2	All three shots have a similarity value above 90% between each other . . . . .	34
5.3	Semantically similar shots with a measured value of 78 % . . . . .	34
5.4	Similar locations and context but different camera angles and distance: 60 % similar . . . . .	35
5.5	Examples of manually applied visual transformations and their measured similarities . . . . .	36
5.6	Black bars are taken into account during scene recognition, which wrongly classifies all of these shots as being indoor . . . . .	37
5.7	Ambiguity in semantic information leads to all these shots being considered highly similar (around 70 %). They have a small number of common objects (one or two people) and almost no relevant background information . . . . .	37
5.8	Fast consecutive changes in color information, specially brightness, leads to the incorrect detection of multiple shot boundaries . . . . .	38
5.9	Examples of queries (first column on the left) and near-duplicate videos from the CC_WEB_VIDEO dataset . . . . .	44
5.10	Precision-Recall and ROC curves of the whole subset (2537 videos) . . . . .	45
5.11	Resulting confusion matrix from K-Fold Cross Validation . . . . .	45

## LIST OF FIGURES

# List of Tables

2.1	Scene recognition results from the official Places2 Demo for the images in Figure 2.2	9
4.1	Different types of features used to detect shot transitions . . . . .	28
4.2	Feature sets part of the shot signature feature vector . . . . .	31
5.1	Scene recognition summary for the shots in Figure 5.4 . . . . .	35
5.2	Benchmarking machine specifications . . . . .	39
5.3	Benchmark results for Object Detection and Scene Recognition . . . . .	39
5.4	Benchmark results for Object Tracking . . . . .	40
5.5	Benchmark results for Shot Boundary Detection . . . . .	40
5.6	Benchmark results for the whole feature extraction process. Running time refers to the average time a task takes per iteration/frame processed . . . . .	41
5.7	Benchmarks for the signature generation and comparison processes . . . . .	42
5.8	Similarity annotations available in the CC_WEB_VIDEO dataset . . . . .	42
5.9	Summary of the subset of videos used from the CC_WEB_VIDEO dataset . . . . .	43
5.10	Average Precision and ROC AUC measures for every query done . . . . .	43
5.11	Results from K-Fold Cross Validation . . . . .	44

## LIST OF TABLES

# Abbreviations

AI	Artificial Intelligence
CVIS	Computer Vision
ML	Machine Learning
ANN	Artificial Neural Network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
NDV	Near-Duplicate Video
NDVD	Near-Duplicate Video Detection
NDVR	Near-Duplicate Video Retrieval
ROC	Receiver Operating Characteristic
AP	Average Precision
PR	Precision-Recall
AUC	Area Under the Curve





# Chapter 1

## Introduction

In recent years, there has been an increasing growth in research in both the areas of Computer Vision and Machine Learning. As per Lee et al. [Lee17], this is mostly due to the emergence of Deep Learning algorithms, which use Artificial Neural Networks that are inspired by the biological neural networks in animal brains.

“From the biological science point of view, computer vision aims to come up with computational models of the human visual system. From the engineering point of view, computer vision aims to build autonomous systems which could perform some of the tasks which the human visual system can perform.” [Hua96, chap. What is Computer Vision?]

Computer Vision is an inherently difficult research field due to the human visual system being very accurate in many complex tasks, such as face recognition, edge detection, and scene recognition. By using massive computing power and large volumes of data to train different architectures of Artificial Neural Networks, researchers are able to stop trying to solve these visual tasks that the human vision is so good at and, instead, focus on how to formulate them so that machines can learn how to do it themselves.

### 1.1 Context

Computer Vision (CVIS) is a very broad research field and is driven by various advancements in other fields [Lee17], as seen in Figure 1.1. These other fields also benefit from CVIS’s own advancements in technology and research. Machine Learning, in particular, is recently seeing a rise in popularity due to the emergence of Deep Learning algorithms [Lee17].

The continuous increase in performance of Graphics Processing Units (GPU’s), starting in the early 2010s, has allowed researchers to use them for performing the necessary Artificial Neural Network computations with a noticeable increase of performance over common Central Processing Units (CPU’s). This lowered the barrier of entry for research in the field which, in turn, led to

## Introduction

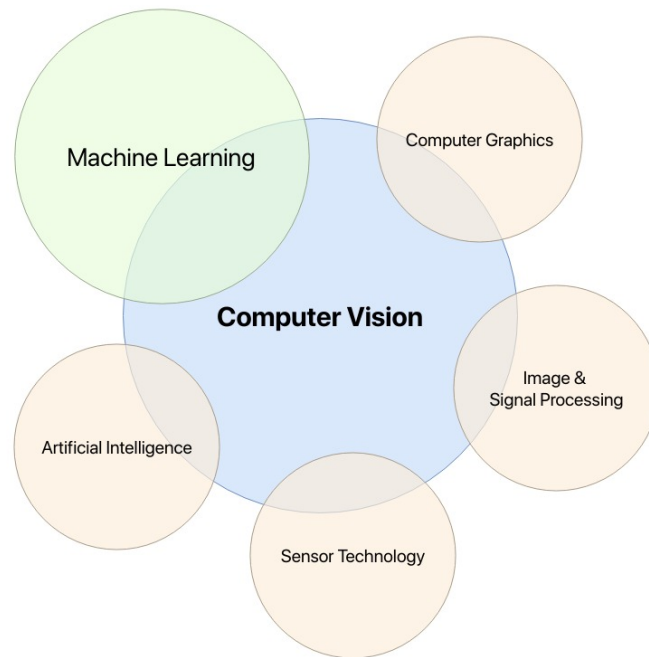


Figure 1.1: Computer Vision driven by advancements in multiple fields

exploration into even deeper and more computationally expensive networks. The year 2010 also marked the start of the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC): “[ILSVRC] is a benchmark in object category classification and detection on hundreds of object categories and millions of images” [RDS<sup>+</sup>15]. The winner models throughout the years can be seen in Figure 1.2.

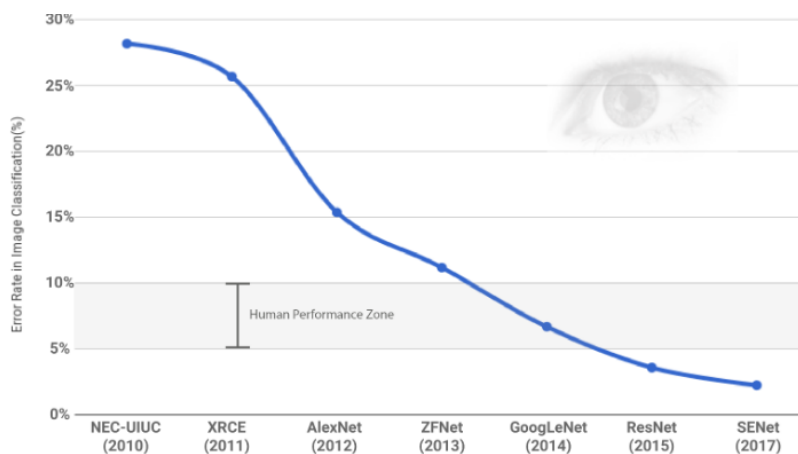


Figure 1.2: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Results

Two years later, in 2012, a version of a Deep Neural Network (DNN), called Convolutional Neural Network (CNN), showed a big leap in accuracy and performance in image classification and object detection tasks over previous non-ANN solutions. The CNN was called AlexNet [KSH12]

## Introduction

and it won the ILSVRC that year. As of the time of writing, all the following years' winners were also architectures based on CNNs, each showing small iterative improvements in accuracy and error rate over the previous.

The winner of 2015 was the first to achieve an error rate below that of humans which, according to the experiment ran by Andrej Karpathy et al. [Kar14][RDS<sup>+</sup>15, Section 6.4], is estimated to be between 5 % and 10 %. This marked the point where image classification could be considered a solved problem.

In the context of Computer Vision, the process of changing the representation of an image into something that is more meaningful and easier to analyze for both machines and humans is called *image recognition*. This includes various tasks such as identifying objects, places, people, and text. *Video recognition* is a superset of image recognition, in which the temporal dimension is also accounted for. Common tasks in video recognition are object tracking and action recognition, where actions could be running, playing, driving, etc.

Recent advancements in both image and video recognition have enabled the creation of various technologies:

- Automated image and video tagging [SSBNMSBJSVSB15]
- Content-based search for images and videos [SSBNMSBJSVSB15]
- Self-driving vehicles [RB15]
- Face recognition [Ko18]
- Detect or censor inappropriate content [ZF13, PAM<sup>+</sup>17]
- Text extraction [WG15]
- Accessibility utilities for the visually impaired [KM15, MTZ17]

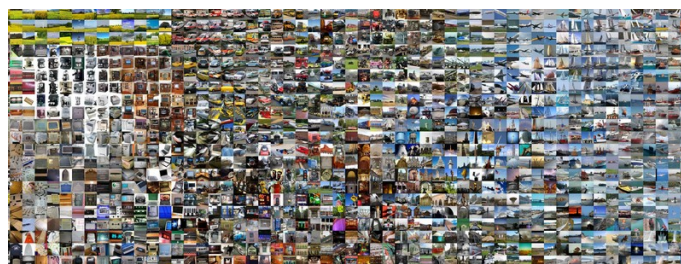


Figure 1.3: Example images from the classification task in ILSVRC

## 1.2 Motivation

A major setback in Machine Learning is the amount of data—and its subsequent labelling—that is needed in order to train an accurate model. This is especially the case for problems that require working with media contents, where manual annotations are very labor intensive.

Due to the widespread of the Internet and its increasing broadband speeds, along with the appearance of a multitude of video sharing and streaming services, lack of data isn't considered a problem anymore. In fact, according to Cisco [Cis17], IP video traffic accounted for 73 % of all IP traffic globally in 2016. It doesn't show signs of stopping either, with an estimate of 82 % by the year 2021. In North America alone, Netflix, which is one of the most popular on-demand video providers, is the top site by percentage of downstream Internet traffic with a value of 37 %. YouTube follows in second place with a value of 18 % [San15].

The existence of large amounts of media data also brings new challenges. On one hand, in order to continue providing a good experience for users as well as facilitate the management of the underlying servers, there is the need to reconsider the way that retrieval, archiving, sharing, and mining of these types of contents is done. On the other hand, we can observe that a big portion of the videos available online show a large degree of similarity between each other. This is explained by the increasing popularity of video sharing services, which provide an accessible platform to share media contents on. In a study ran by Liu et al. in 2015 [LBX<sup>+</sup>15, Chapter 5], they estimated that there are over 859 million videos on YouTube, 31.7 % of which are considered duplicates or near-duplicates. Another major implication of this ease of share is the increased risk of illegal redistribution and use of copyrighted content, where the emergence of video on demand services also plays a big role by providing quick and easy access to such contents.

This work was developed in the care of the company named MOG Technologies<sup>1</sup>. MOG is involved in the markets of media contents and broadcasting. It has developed world-class media libraries, state of the art broadcast systems, and a cloud-based interactive and multi-content platform. The main theme of the dissertation was proposed the company which, in turn, came from the need to provide a solution to the problem of video monitoring, with the potential of transforming it into an additional service to clients.

### 1.3 Objectives

Our main purpose in this dissertation is to try to approach the challenges described in the previous section using the recent advancements in the area of Computer Vision and Machine Learning. More specifically, to develop a system that extracts relevant information from video assets using the latest CNN architectures in research and then uses that information to measure similarity between videos.

The common approach for extracting information from media assets is through the use of meta-data—data that describes data—which, besides some given properties like file name, size, duration and encoding, needs to be manually annotated. Useful textual information can include title, category, description, comments and tags.

Manually analyzing a video can be a lengthy and labour intensive process, as it requires re-watching said video multiple times, depending on the level of detail wanted. In the context of video

---

<sup>1</sup><http://www.mog-technologies.com/>

sharing services with hours of videos uploaded every minute<sup>2</sup>, this can be an outright impossible task to accomplish. The usual compromise is to make it the responsibility of the users to manually annotate their own uploads.

Having a good annotation and organization of media contents is crucial for the development of useful functionality such as content-based image and video retrieval, automatic tagging, and content-based recommendation engines. By using the latest results in image and video recognition research involving CNNs, we can automate this video annotation process, which is commonly known as *video signature generation*. Furthermore, these same techniques can also be used in applications dealing with duplicate videos, like copyright protection.

In summary, we test the hypothesis of whether the results from Convolutional Neural Networks trained in image and video recognition tasks can be used in the creation of video signatures.

### 1.4 Dissertation Outline

#### ***Chapter 2 Image and Video Recognition***

This chapter begins with an introduction to the areas of image and video recognition followed by a brief history of Convolutional Neural Networks and their use in various visual related tasks. Next, we review the state-of-the-art CNN architectures used in object detection and scene recognition tasks. The chapter ends with a rundown of the most popular technologies used in the field.

#### ***Chapter 3 Video Similarity***

In this chapter we explain the concepts of video similarity and near-duplicate videos. This is followed by a description of some practical applications made possible by systems able to measure these concepts in an automated fashion. We end the chapter with an in-depth look at the current approaches used in literature.

#### ***Chapter 4 Proposed Solution***

This chapter is dedicated to the presentation of our proposed solution of using the results from CNNs to generate video signatures and its subsequent implementation details.

#### ***Chapter 5 Experimental Results***

In this chapter we discuss a variety of results obtained from our implemented video similarity system. It starts with initial observations and performance benchmarks and ends with the experimental results in the task of near-duplicate video detection using the CC\_WEB\_VIDEO dataset.

#### ***Chapter 6 Conclusions and Future Work***

This final chapter summarizes the results of the whole work with respect to our initial hypothesis as well as contributions to the area. It concludes with possible improvements and future development.

---

<sup>2</sup><https://www.youtube.com/yt/about/press/> (Accessed 3 March 2018)

## Introduction

## Chapter 2

# Image and Video Recognition

### 2.1 Introduction

Image and video recognition is the process of changing the representation of an image or video into something that is more meaningful and easier to analyze for both machines and humans. It involves tasks such as identifying objects, places, people, writing, and actions. Ever since they were first introduced, Convolutional Neural Networks have been successfully used to solve these various tasks.

Convolutional Neural Networks are a class of deep, feed-forward artificial neural networks. Their creation was inspired by biological processes [MMMK03], with its architecture decisions being influenced by the organization of the animal visual cortex. As previously said in Section 1.1, 2012 was the year that CNNs came to prominence as Alex Krizhevsky's AlexNet [KSH12] won the ILSVRC that year. It achieved a classification error of 15 % compared to 26 % from the previous record, a big improvement at the time. This sparked interest in both the industry and scientific community. Researchers started focusing on finding new uses and architectures for CNNs while companies put the already existing ones to the test, with their access to high amounts of data and computational resources.

Various versions of CNNs have been developed throughout the years in an attempt to improve on the concepts introduced by AlexNet:

- **VGG Net (2014)** [SZ14] reinforced the idea that, in order to effectively represent visual data in an hierarchical way, CNNs needed to have a deep network of layers
- **GoogLeNet (2014)** [SLJ<sup>+</sup>14] showed that CNN layers didn't always have to be stacked up sequentially. With carefully crafted design, the depth and width of the network can be increased while keeping the computational budget constant
- **ResNet (2015)** [HZRS15] presented a solution to ease the training of networks that are substantially deeper than those used previously, achieving a 152 layer network that set new

records in classification, detection, and localization. It won ILSVRC 2015 with an error rate of 3.6 %, which is below the human’s 5–10 % [Kar14] [Kar14, Section 6.4].

This chapter describes some particular tasks that are necessary to accomplish the main objective of the dissertation, showing the current most successful architectures and approaches for each. The last section goes over the more popular technologies for implementing these architectures and also some already existing solutions to the challenges at hand.

## 2.2 Object Detection

Object detection is the process of localizing and identifying multiple instances of semantic objects of a certain class (e.g. humans, cars, flowers) in a single image or video.



Figure 2.1: Example output from an object detector

According to Huang et al. [HRS<sup>+</sup>16], a lot of the progress that has been made in recent years on object detection is because of the use and advancement of CNNs, with R-CNN [GDDM13] being one of the first object detectors based on them. The way R-CNN works is by first proposing regions with possible objects, extract features from those regions using a CNN and, finally, classify them. Although it manages to achieve great results, there are serious problems related to the training. Most notably, the region proposals are statically generated, which results in huge training datasets and a slow training time. This led to two subsequent iterations called Fast R-CNN [Gir15] and Faster R-CNN [RHGS15], as well as a new approach named YOLO [RDGF15]. What all these share in common is the fact that both detection and classification are done by a single network. Not only are they faster during execution, but also allow end-to-end training, reducing the training time significantly. YOLO, in particular, was the first object detector proposal that was fast enough for real time usage, with the drawback of making more localization errors than the alternatives.



The current state-of-the-art object detectors are improvements over the Faster R-CNN and YOLO architectures. R-FCN [DLHS16] and Mask R-CNN [HGDG17] take on slightly different approaches to the former while YOLOv3 [RF18], YOLOv2 [RF16] and SSD [LAE<sup>+</sup>15] are inspired by the latter.

## 2.3 Scene Recognition

Scene recognition is a special case of an image classification task where the goal is to successfully identify the name or category of a specific scene in an image or video. As such, CNNs have been successfully used for solving the problem. “A key aspect of scene recognition is to identify the place in which objects seat” [ZLK<sup>+</sup>17]. This provides the appropriate level of abstraction for putting the objects detected into context without needing to extensively describe all the present objects and their spatial relationships.

The main contributions in the area come from MIT’s Places2 Database and Places Challenge [ZLX<sup>+</sup>14, ZLK<sup>+</sup>17]. The database is made of 10 million scene photographs, each labeled with one of the 434 scene semantic categories and, in total, represents about 98% of the types of places that exist in the world. In Table 2.1 we can see the scene recognition results obtained from the official Places2 Demo<sup>1</sup> for the images in Figure 2.2.

Figure	Type of Environment	Scene Categories
(a)	Indoor	Entrance hall (53%) Lobby (17%)
(b)	Outdoor	Aqueduct (94%)
(c)	Outdoor	Hayfield (80%) Wheat field (12%)
(d)	Indoor	Bathroom (80%) Shower (17%)

Table 2.1: Scene recognition results from the official Places2 Demo for the images in Figure 2.2

## 2.4 Technologies

In the area of Machine Learning, the main programming language in use by both the research community as well as the industry is Python. As such, a large number of mature Deep Learning libraries and frameworks are written in it.

The most popular frameworks for building computational graphs (neural networks) are PyTorch<sup>2</sup> and Caffe2<sup>3</sup> by Facebook, and TensorFlow<sup>4</sup> by Google. All three have a big, growing

<sup>1</sup><http://places2.csail.mit.edu/demo.html>

<sup>2</sup><http://pytorch.org>

<sup>3</sup><https://caffe2.ai>

<sup>4</sup><https://www.tensorflow.org>

## Image and Video Recognition



Figure 2.2: Some examples available on the official Places2 Demo

community behind them and a multitude of pre-trained models available for use. This allows faster prototyping and lowers the requirement for powerful computing resources during the early research stages, because there is rarely a need to train every network completely from scratch.

Generally, PyTorch is considered to be aimed for use during the research stages while both TensorFlow and Caffe2 are more optimized for production environments. In PyTorch, computational graphs are built and run in a dynamic way during code execution. In the other frameworks, computational graphs are first built and compiled once—including any possible optimizations—and continuously executed later. This provides better performance and easier deployment at the cost of research flexibility, as the compilation process usually obfuscates the meaning of the operations executed. Although, there are some wrappers around TensorFlow, like Keras and TensorFlow Fold, which enable the construction of graphs dynamically similarly to PyTorch.

Since these frameworks are pretty low-level, considering they only allow to operate up to the network level, some other frameworks have been built on top of them that provide a higher level usage. For object detection there is Detectron [GRG<sup>+</sup>18] and TensorFlow Object Detection API [HRS<sup>+</sup>16]. Both of these open-source projects offer a collection of pre-trained object detection models and tools that help training these models with new object classes. For scene recognition, there are various CNN models pre-trained on the Places2 Database [ZLK<sup>+</sup>17].

## Image and Video Recognition

In terms of a more complete solution, there are various commercial cloud services available. The most notable ones being Amazon Rekognition, Google Cloud Vision API and Clarifai. All of them provide ready-made solutions for a variety of image and video recognition tasks such as detecting objects, scenes and actions; facial analysis and recognition; and text extraction.

## Image and Video Recognition

## Chapter 3

# Video Similarity

### 3.1 Introduction

Video similarity is often defined as how visually similar two given videos are. However, human visual perception operates on high-level concepts, such as scenery, objects and depth. A video also has a temporal dimension, which humans perceive as how these concepts change over time and are recognized as gestures, actions, events, etc.

Formally describing how humans perceive visual similarity is a hard problem. Traditionally, this is done by comparing different sets of *features* from two given videos. These features can either be low-level, such as color, texture and shape; or high-level, such as the concepts described above. It is also usual to refer to these as non-semantic and semantic features, respectively.

A common term in the context of video similarity is what's referred to as near-duplicate or near-identical. Fundamentally, a video is considered a near-duplicate if it is highly similar to another. However, being a relatively new topic in research, there isn't yet a universally accepted definition of what actually contributes to the measurement of similarity between videos. Liu et al. [LHCS13] review a variety of representative definitions [WHN07, TSZH<sup>+</sup>07, BZS08, CdOO09]. In summary, there are usually two types of similarity considered when classifying a video as a near-duplicate: visual similarity, where the visual content largely remains the same allowing for differences in file formats, encoding, frame rate, color and even some editing operations such as insertion of logos, captions and subtitles; and semantic similarity, in which completely different videos can be considered similar if they share the same semantic concept like, for example, two distinct videos showing a dog running on a beach.

In Figure 3.1 we show visual examples for the two types of similarity. Video (a) is a clip from the original movie GoldenEye and video (b) is the same clip except that it suffered the following modifications: insertion of two logos and a caption; changes in lighting and color; and different encoding parameters and file format. Despite all this, the visual content largely remained the same. As such, these two clips can be considered visually similar. In contrast, videos (c) and (d) are completely unrelated. They were filmed in different places and have different actors on scene. Despite this, they both represent the same concept: an interview process. Both scenes have the

## Video Similarity

same setting, as in the type of place, and contain the same semantic objects: two formally dressed people facing each other with a table in between. Hence, they can be considered semantically similar.



(a)



(b)



(c)



(d)

Figure 3.1: The two types of similarity: (a) and (b) are visually similar while (c) and (d) are semantically similar

In order to test how close the human perception of video similarity is to the definitions described previously, as well as measure what particular features are important for humans when classifying a video as a near-duplicate, Cherubini et al. [dOCO09, CdOO09] have ran a user study through the use of an online survey. In the end, they concluded that the technical definitions for visual similarity are closely related to that of humans: changes in file format, lighting, color and small editing operations aren't significant for considering a video as different. The impact of semantic differences wasn't as conclusive. However, it was clear that semantic similarity does play a role in the human perception of near-identical videos.

## 3.2 Applications

As seen previously in Section 1.2, there has recently been a rapid growth in video traffic and data, mostly due to the continuous increase in popularity of video sharing websites, such as YouTube, Yahoo! Video and Vimeo, and video on demand providers like Netflix and Hulu. On the other hand, this growth has also surfaced some particularly hard to solve problems. First, video on demand providers provide easy access to copyrighted material. Second, video sharing websites constantly attract millions of users that both produce and consume videos. This ease of sharing caused a mass increase in the presence of near-duplicate videos on the web [LW15]. Consequently,

some of those near-duplicates refer to an original that is protected by copyright laws, including user generated videos, which may have exclusive rights as well.

The existence of a large number of near-duplicate videos has other implications besides that of copyright infringement:

- Storage space is filled with highly redundant data, which leads to an unnecessary increase in maintenance and operation costs;
- Search queries are polluted with a large number of seemingly identical results;
- Video recommendation systems are harder to implement correctly;
- User activity is spread throughout the various versions of a video;

As the amount of data continues to increase, solving these challenges through manual intervention quickly becomes impossible. Thus, a variety of near-duplicate video applications have emerged [LHCS13, ZN15] in the hopes of solving or, at least, mitigating these problems in an automated fashion. In the next subsections we review some of these applications.

### 3.2.1 Copyright Protection

The ease of access to copyrighted contents provided by on demand video services coupled with the ease of sharing enabled by online video platforms has increased the risk of exclusive rights being violated by unauthorized copying, editing, and redistribution. This decreases the number of potential customers to these contents, which leads to a financial loss by the part of the creators. Not only that, some video sharing platforms like YouTube also provide a notion of partnership with its users, which enables them access to some of the revenue generated by their uploaded contents. While not as problematic financially, as big studio companies have a substantially higher return of investment value, this still increases the amount of copyrighted content dramatically.

Redistributed content may differ from its original in a variety of ways, such as format, resolution, contrast, and brightness. It can also have additional visual information like banners, logos and even entirely new scenes. There is a seemingly infinite number of transformations a video may suffer while still remaining very similar to its original. This unpredictability makes it all the more difficult and complicated to detect possible copyright violations.

Eliminating these violations is a real need for both content providers and video sharing services. The development of techniques for near-duplicate detection and retrieval can potentially eliminate the majority of cases. For example, video sharing services can check newly updated videos against a list of protected videos provided by content providers.

### 3.2.2 Video Monitoring

The ability to find videos contained inside other videos, that is, to be able to match partial parts of videos, is very important for some applications, such as nudity, inappropriate content and commercial detection. For example, a company wants to ensure the terms of a TV commercial contract

are being met by checking if it's being broadcasted for its expected duration and during the agreed time period.

Manually monitoring videos can be a very demanding task, specially in the context of real-time broadcasts, which might even require multiple workers depending on the number of different frequencies it's being broadcasted on. Furthermore, in the context of a very popular video sharing website like YouTube, tasks like nudity detection are downright impossible to accomplish manually, due to the amount of videos being uploaded constantly. As such, both detection and retrieval techniques can be very useful to solve or, at least, mitigate these problems by filtering the number of videos which need manual intervention.

### 3.2.3 Video Re-ranking

NDVs take a huge part in video search experience. The majority of users who use video sharing websites do so with the main intent of actively searching for specific videos [dOCO09, CdOO09]. In addition, users also show preference to a single video in a list of near-duplicates, often wishing to be able to discard the rest from the search results.

As the number of near-duplicate videos increases, video search service providers might want to improve their results ranking algorithm by adding ability to detect novelty in videos. In the context of user-generated content, this means finding the original video in the midst of a large number of duplicates or near-duplicates. However, some instances of near-duplicates are often considered useful by users, such as when they provide additional information like subtitles or commentary [CdOO09]. NDVD techniques can prove very effective in these cases, either by clustering NDVs using different criteria [HHC<sup>+</sup>10], or by moving the original to the top of the list [WHN07].

### 3.2.4 Video Recommendation

A very common application related to video re-ranking is video recommendation. That is, to be able to find videos which are not completely identical but do share some similarity or, at least, add useful information. Cherubini et al. [CdOO09] observed that there's a high probability that if users like a video, then they are also interested in other very similar videos. As such, video recommendation systems which only work with text based data, such as title, description and comments, might want to enhance their user's experience by including the visual information obtained from NDV techniques in the video's meta-data. This way, they can not only improve their recommendation system, but also complement an additional text-based search system as well, which can make use of the already implemented database to associate words to visual information.

## 3.3 Common Approaches

Near-duplicate video processes can be classified into two categories [LHCS13]: Near-Duplicate Video Detection (NDVD) and Near-Duplicate Video Retrieval (NDVR). While very similar, it is



important to note the differences in application between the two. In retrieval processes a video database is previously established and a signature that represents each video is generated from the results of feature extraction [JTL<sup>+</sup>17]. This procedure is usually done offline. Later, when a query video is presented, its signature is generated and compared against the signatures in the database. In the end, a set of near-duplicate videos is returned, commonly ranked by descending similarity value (or ascending distance value). Copyright protection and video recommendation are the most common applications of these systems.

Conversely, in detection processes the aim is to find pairs or groups of near-duplicate videos from a given set of videos as input [WNHT09, TWNZ08]. The resulting groups are often obtained either by using clustering techniques or a manually defined similarity threshold. Given that the input set of videos is not known in advance, it is uncommon to have a previously established database of videos and signatures. As such, detection usually takes longer than its counterpart and is also subject to combinatorial explosion depending on the size of the input. Examples of applications for NDVD include database cleaning, video re-ranking and video thread tracking.

Both detection and retrieval share a lot techniques. More specifically, feature extraction, signature generation and comparison largely remain the same. Figure 3.2 shows an overview of how both systems work. In the rest of this section, we go over the most common approaches for NDVR and NDVD from the aspects of feature extraction, and signature generation and comparison.

### 3.3.1 Feature Extraction

Feature extraction is the process of building derived values, known simply as *features*, from a set of initial data. Its main purpose is to reduce redundancy while still retaining the relevant information needed to describe that set of data. As videos are essentially a series of static images (frames) in which the visual content doesn't change significantly from frame to frame, they have high levels of redundancy. In fact, most video compression methods work on the basis of mitigating that redundancy. As such, feature extraction is usually the first and most important step in any kind of computer analysis procedure which involves video data.

Most of the feature extraction methods used for videos are derived from the ones used in image processing and can be classified into three groups [JTL<sup>+</sup>17]: *global visual features*, *local visual features*, and *deep learning features*.

Global visual features refer to the statistical visual information of an image such as the color histogram [HHC<sup>+</sup>10, TXZ<sup>+</sup>07, WHN07]. These types of features are very compact and can usually be computed efficiently. However, they can't represent shape, geometric or texture information, which proves to be problematic in cases where two completely distinct videos have similar color distributions. Conversely, all it takes is a slight color change on the original video in order to not be detected.

On the flip side, local visual features are used to describe both geometric and shape information. These are often referred to as keypoint descriptors and there's a variety of available algorithms to generate them, such as Scale-Invariant Feature Transform (SIFT), Principal Components Analysis (PCA) and Speeded Up Robust Features (SURF). These types of features provide much

## Video Similarity

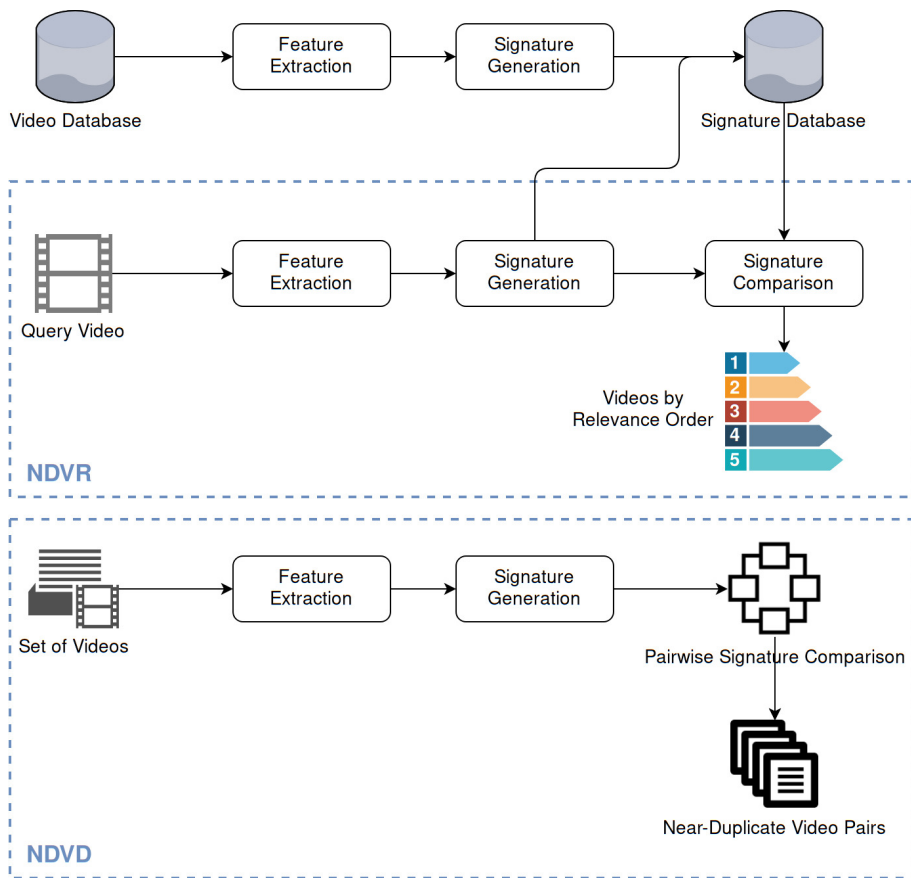


Figure 3.2: Overview of NDVR and NDVD processes

better accuracy compared to global features [ZN15]. They are also invariant to changes in scale, rotation, viewpoint, brightness as well as the presence of noise. A major disadvantage is that they are much more expensive to compute and also require more storage space.

The latest advancements in Computer Vision and Machine Learning (Section 1.1) have enabled a new kind of features to be used for NDV tasks: deep learning features. Or, more specifically, using the values of intermediate layers in a Convolution Neural Network pre-trained in image recognition tasks as features [KZPPK17, Per15]. These have proven to be able to compete with the state-of-the-art keypoint descriptors techniques in terms of accuracy and, if ran on a GPU, in terms of performance as well.

### 3.3.2 Signature Generation

A video signature is defined as a high-level video summarization derived from low-level frame features [LHCS13]. Its main purpose is to reduce the data needed to describe a video which, in turn, reduces the computational complexity associated with the later signature comparison step. As such, the process usually involves picking independent and informative features out of all the ones previously extracted.

Generally, different applications require the development of their own custom video signatures as a consequence of distinct scenarios and purposes. According to Liu et al. [LHCS13], the existing approaches in video signature generation can be classified into five categories:

- **Textual Signature** [AWB03]: represent a whole video using features from its associated text meta-data, such as title, description, authors and comments;
- **Video-level Global Signature** [HSS<sup>+</sup>09, WHN07]: represent a whole video with a single signature. It is often very efficient for storage and retrieval due to its small data size. May not be suitable if the application demands high accuracy or granularity like, for example, when dealing with long videos;
- **Frame-level Local Signature** [WNHT09, WHN07]: represent a frame using local features such as keypoint descriptors obtained from algorithms like SIFT and SURF. Its main use is in NDVD applications which demand a high level of accuracy. The main downside is that it requires more space and is much more computationally expensive than a global signature, due to the nature of the algorithms used for local keypoint extraction;
- **Frame-level Global Signature** [JN09]: represent a frame with a single signature. It provides a middle ground between local frame signatures and global signatures in both disk space and computation efficiency. Depending on the features used, it can reach accuracy levels close to those of local frame signatures;
- **Spatio-temporal Signature** [SYW<sup>+</sup>10]: represent a video using spatial and temporal information. Most of the times, it involves tracking changes in video content over time. Its main use is in detecting identical scenes shot from different camera angles. So, just like some keypoint descriptors, it is invariant to viewpoint changes and is usually more computationally efficient;

### 3.3.3 Signature Comparison

The final step for both NDVD and NDVR tasks is the comparison between signatures in order to get a similarity measurement between two videos. A very common strategy is to represent a video signature as a feature vector [WHN07, WNHT09]. A feature vector is essentially just an  $n$ -dimensional vector in which the features previously extracted are represented numerically. This enables the use of a vast family of distance metrics which operate on vectors and matrices, such as the Euclidean and Cosine distance metrics. Feature vectors can also be combined with weights using a dot product in order to give more importance to specific sets of features when comparing. The other common representation for signatures is using string data [HLCD13], where methods like Jaccard similarity and edit distance can be used as similarity measures.

## Video Similarity

## Chapter 4

# Proposed Solution

In the following sections we explain our proposed video similarity system, which uses Convolutional Neural Networks to capture semantic information. The technology stack is based on the Anaconda Distribution<sup>1</sup> with Python 3.6 as the main programming language. The core libraries used are NumPy<sup>2</sup> and PyTorch<sup>3</sup> 0.4. NumPy and PyTorch's Tensors are scientific computing packages which provide a high-level abstraction for working with multidimensional data efficiently. The main distinguishing feature between them is that Tensors support running some of the operations on the GPU, which speeds up computations significantly when working with large portions of data. PyTorch is also used as the main Deep Learning framework.

### 4.1 Overview

In order to be able to test the system in existing challenges as well as provide for exploration during the research stages, a few requirements had to be met:

- The system should accept a variety of inputs including videos in multiple formats, streams, and plain sequences of images;
- Similarity measure needs to be a percentage where 100% represents an exact duplicate and 0% means completely dissimilar;
- Extracted features should be interpretable by humans;

Our system's architecture is similar to that of common NDV applications, where two main modules can be identified: one that is responsible for analyzing videos and creating a concise and structured representation of them, and another that uses this representation to generate and

---

<sup>1</sup><https://www.anaconda.com/distribution/>

<sup>2</sup><http://www.numpy.org/>

<sup>3</sup><https://pytorch.org/>

compare video signatures. An overview of the system and its main processes can be seen in Figure 4.1. Our approach is novel in the sense that it uses the results from CNNs pre-trained in image and video recognition tasks, such as object detection, as the sole features to represent a video. Essentially, we are testing the impact of using only high-level or semantic features to measure similarity between videos. In addition, using these high level features means we can visually annotate a processed video with relevant information, which can later be used to empirically analyze what data our system is working with. An example of such annotations can be seen later in Section 4.5.

We found two other proposals which use CNNs as feature extractors in literature [KZPPK17, Per15]. While they managed to achieve very good results, the works mostly focus on using the results from intermediate layers, which aren't as easily interpretable. Because CNNs are usually trained using a variety of image augmentation techniques, a major advantage of using them as feature extractors is that they are inherently invariant to changes in scale, rotation, viewpoint and, to some extent, color. The use of CNNs also usually implies a trade-off between precision and processing time. Careful decisions must be made when choosing which neural network architectures to use depending on the application and available hardware.

For the signature generation module, we decided to go with a shot based approach in the style of Zobel and Hoad [ZH06]. In short, a video is split into shots based on major changes in camera position or detected scene cuts and a signature is generated for each shot. Video signatures are then composed by a sequence of shots and any relevant meta-data. This provides for a middle-ground in accuracy and performance between video-level global signatures and frame-level signatures. As an upside, it is invariant to changes in timescale. Some applications made possible by this approach are: detecting parts of a video contained in another; retrieving similar videos based on only a portion of the query video; and creating custom video queries by building a video signature based on a collection of relevant shots.

## 4.2 Object Detection and Scene Recognition

We obtain high-level features from individual frames using CNNs pre-trained on object detection and scene recognition tasks. For object detection, three of the state-of-the-art networks in inference speed [RF16, RF18, LAE<sup>+</sup>15] were tested. We chose YoloV3 [RF18] as it was the one which demonstrated the best accuracy while still allowing some headroom for real-time applications. The implementation's input size is 416x416 and the region proposals are filtered using Non-Maximum Suppression with a threshold of 40%. Its output is stored as list of bounding boxes, each with an associated object class and confidence score. Some example detections can be seen in Figure 4.2.

As for scene recognition, we use Places365-Standard [ZLK<sup>+</sup>17], a collection of Convolutional Neural Networks pre-trained on a subset of 365 categories from the Places2 dataset. Specifically, we are using the WideResNet-18 version as it not only provides the labels for the predicted categories, but also labels for what are called scene attributes, which can be seen as additional information of what features influenced the category prediction decision. The scene attributes are

## Proposed Solution

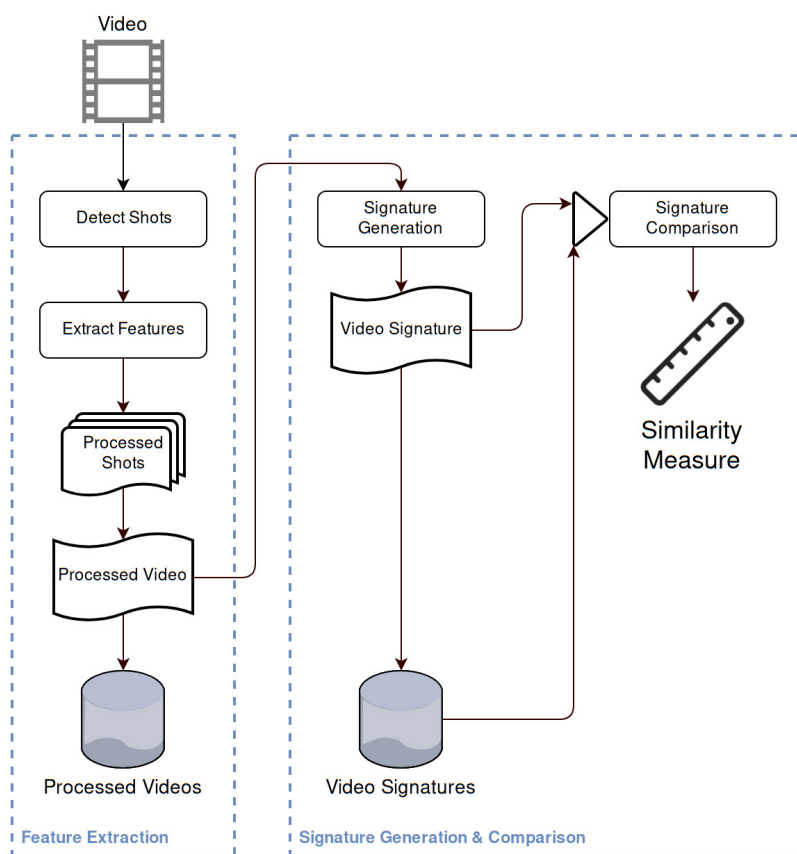


Figure 4.1: Overview of implemented system

obtained by multiplying the results from the last average pooling layer of the network and an officially provided weight matrix<sup>4</sup>.

### 4.3 Object Tracking

In the context of video recognition, object detection is rarely used on its own, as the information provided by detectors is only relevant on a frame-by-frame basis. To associate detected objects between consecutive frames, a tracking method must be used. In this case, since we are dealing with multiple detections of various different classes, the tracking needs to be able to handle multiple objects simultaneously, including partial or full occlusions.

A common tracking algorithm which has been extensively used in multiple object tracking applications is Kalman Filter [KWM94, MFMR02, LWL10, JTJ14]. Kalman Filter is a very popular signal processing algorithm which uses a series of measurements observed over time, containing noise and other inaccuracies, and produces estimates of unknown variables. It only

<sup>4</sup>[https://raw.githubusercontent.com/csailvision/places365/master/labels\\_sunattribute.txt](https://raw.githubusercontent.com/csailvision/places365/master/labels_sunattribute.txt) (Accessed 20 June 2018)

## Proposed Solution

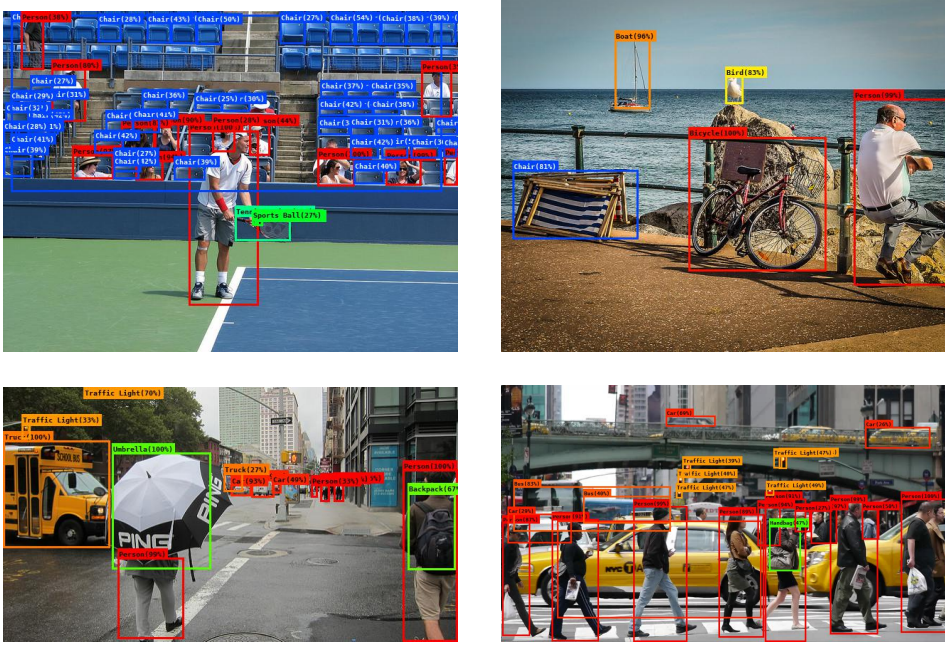


Figure 4.2: Example object detection outputs from the implemented YOLOv3 network

needs the previous estimated state and the current measurement to compute the estimate for the current state. As such, it is very computationally efficient and, if designed well, also very accurate.

Kalman Filter is often conceptualized as two distinct steps [JTJ14]: predict and update. During the prediction step, the current state estimate is derived from the state estimate of the previous iteration. In the update step that estimate is then refined using the measurements made since the last iteration. In our case, the state is a bounding box and we're trying to predict its position, velocity and size, using the detections obtained from our object detector as our measures. Our filter is modeled as seen in Equation 4.1, where  $x_k$  represents the filter's state,  $F_k$  the state-transition model and  $z_k$  an example measurement.

$$x_k = \begin{bmatrix} pos_x \\ pos_y \\ scale \\ ratio \\ vel_x \\ vel_y \\ vel_{scale} \end{bmatrix} \quad F_k = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad z_k = \begin{bmatrix} pos_x \\ pos_y \\ scale \\ ratio \end{bmatrix} \quad (4.1)$$

Bounding boxes are usually defined by the position of one of their corners and their width and height. However, in the filter they are represented by the position of their center, their scale, and width to height ratio. This is because we expect the size of the box to change much more rapidly than it's center, due to the inherent jittery in detections from YoloV3. What we're trying to predict is the bounding box's velocity, both for it's center position as well as scale. We don't consider predicting the rate of change for its ratio because it is uncommon to change significantly.



## Proposed Solution

An important task in the implementation is the association between detections and their corresponding trackers. This is needed in the update step of the Kalman Filter, where each detection serves as the measurement for its associated filter. To do this, we calculate the *Intersection over Union* (IoU) for every  $(Detection, FilterState)$  pair of the same object class. IoU is an evaluation metric used to measure how much two areas overlap each other where a value of 100 % means they completely overlap while 0 % would mean no overlap. A visual definition can be seen in Figure 4.3. The resulting IoUs are then used to do a linear assignment between trackers and detections using the Hungarian algorithm. For any tracker, if its assigned detection has an IoU value below 35 %, it is discarded and the update step of its Kalman Filter is not run. Any unassigned detections are considered new objects and start getting tracked. Every tracker is assigned a unique ID.


$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 4.3: Visual definition of Intersection over Union (IoU)

There are a few parameters in our implementation. More specifically, every tracker requires a minimum number of measurements before it is considered "alive", which avoids false detections that only appear for a small number of frames. Furthermore, a tracker also has a maximum time to live. This means that if no measurements are made within a specified window, the tracker is considered "dead" and is discarded. While having a high time to live is beneficial for dealing with temporary occlusions, it also means that objects which have effectively left the scene take a long time to be discarded. After some experimentation, we ended up with a value of half a second for minimum measurements and one second for time to live, which proved to be a good trade-off. These values are derived from the frame rate of the video being analyzed. A full overview of the tracking process can be seen in Figure 4.4

We made use of the videos from the MOT dataset [LTMR<sup>+</sup>15, MLTR<sup>+</sup>16] for testing. The detections were discarded in favour of our own implementation (YoloV3) and validation was done visually. Some examples can be seen in Figures 4.5 and 4.6.

## Proposed Solution

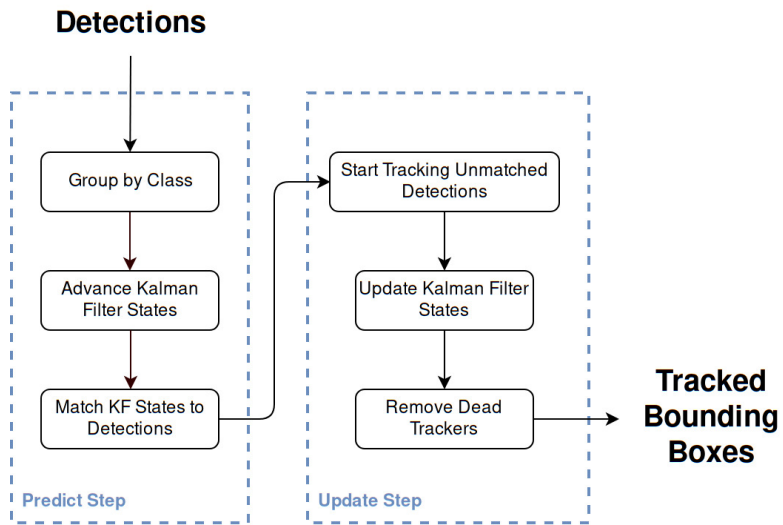


Figure 4.4: Overview of the whole object tracking process

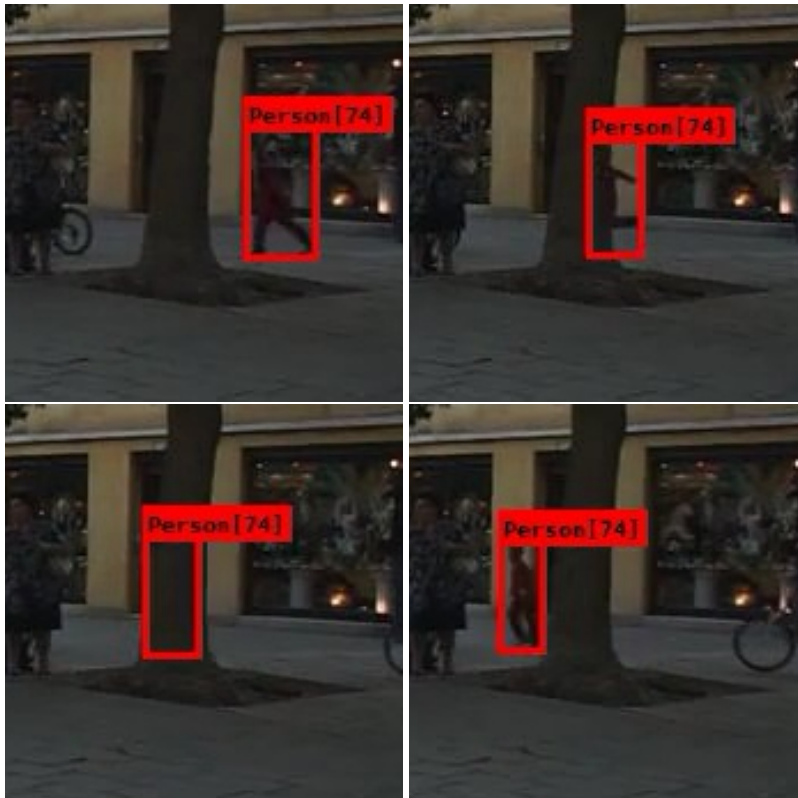


Figure 4.5: Example tracking with occlusion

## Proposed Solution

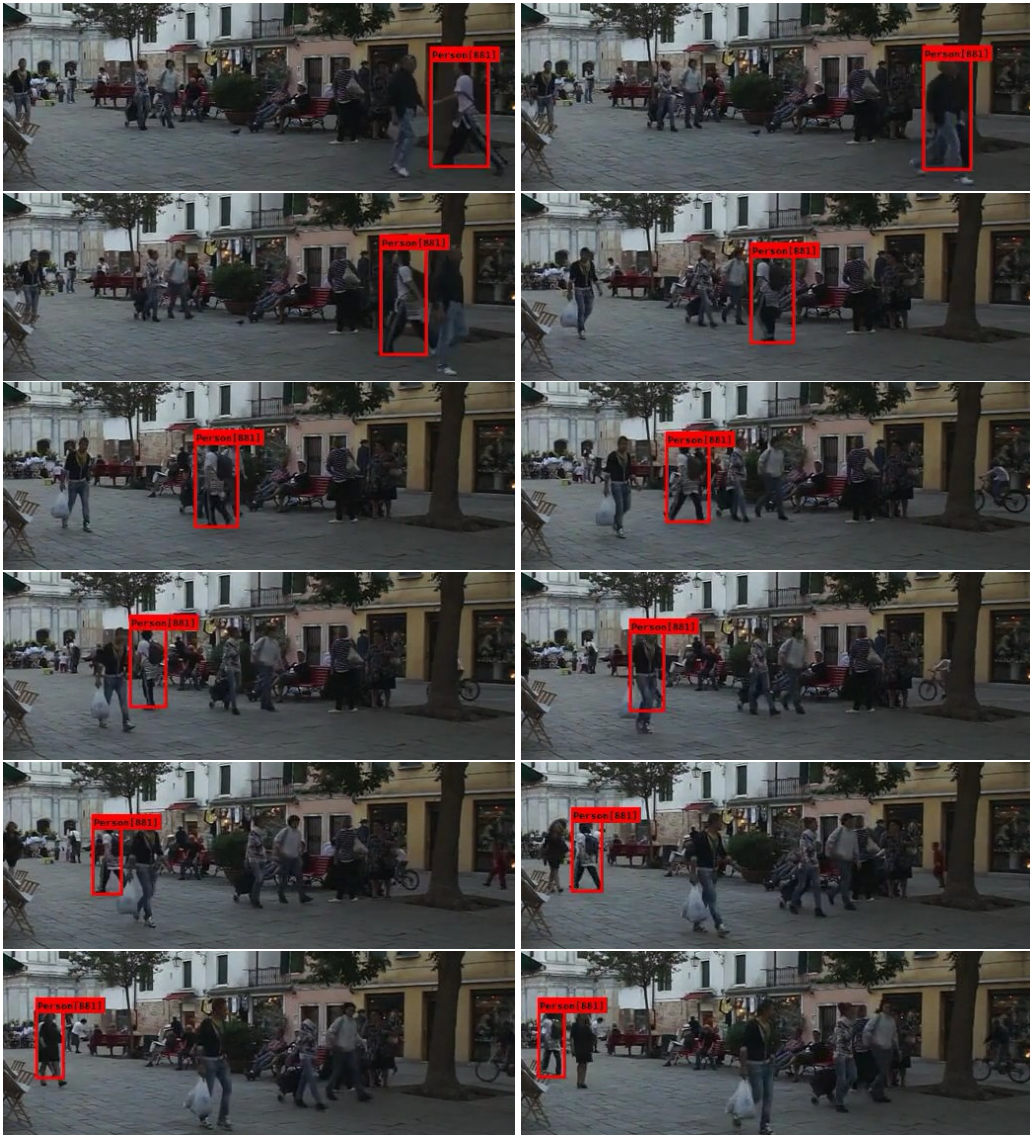


Figure 4.6: Example tracking person across the screen with various occlusions and overlaps  
(Other bounding boxes hidden for clarity)

## 4.4 Shot Boundary Detection

A very important step in the whole implementation is the detection of shot boundaries, which may be separated by various transitions like: simple cuts; fades to and from black; black frame cuts; and dissolves. Essentially, a shot represents a segment of a video where the semantic context and visual content largely remain the same for its duration. Since we are extracting information related to the scene and objects per frame, well defined shot boundaries are imperative in order to obtain a signature that is representative of its contents. For example, if a cut to a completely different scene isn't detected, like from an indoor to outdoor location, the summary of the scene information gathered for that shot will be a mix of features which aren't representative of neither of the distinct scenes. Furthermore, detecting boundaries is also important in order to reset the state of the object tracker.

According to Gajanan [Gaj10], there are four types of features commonly used to detect various transitions, which are summarized in Table 4.1. They also describe the distinction between soft and hard cuts. The former represents transitions that happen gradually during a time frame, such as fades and dissolves, whereas the latter represents abrupt changes, usually no more than a few frames long. As such, methods that need to detect soft cuts often involve calculations using multiple frames and perform slower, contrary to hard cut detection which is generally just a comparison between a pair of frames.

Features	Transitions		
	Cuts	Fades	Dissolves
Color Histogram	✓	✓	-
Edge Change Ratio	✓	✓	✓
Edge Based Contrast	-	✓	-
Standard Deviation of Pixel Intensity	-	✓	✓

Table 4.1: Different types of features used to detect shot transitions

Since our implementation must be able to run in real-time, and shot boundary detection is a procedure that is run for every frame, we had to make a compromise between accuracy and speed in order to not bottleneck the other more expensive operations such as object detection and tracking. Our approach is similar to that of Liu and Wan [LW15], and Sun and Zhang [SZ17], where a cut is detected by comparing the HSV color histogram of two adjacent frames. This comparison is done by using a distance measure which represents the average absolute difference between the H, S and V components of two frames. The frames must have the same dimensions, as the difference between components is calculated using pixel pairs.

Equations 4.2 to 4.4 show the calculations for each HSV component where  $a$  and  $b$  are consecutive frames with width  $w$  and height  $h$ ; and  $x$  and  $y$  are the position in 2D space of the pixels

## Proposed Solution

being compared.

$$\delta_{a,b}^H(w,h) = \frac{\sum_{x=1}^w \sum_{y=1}^h |H_a(x,y) - H_b(x,y)|}{w * h} \quad (4.2)$$

$$\delta_{a,b}^S(w,h) = \frac{\sum_{x=1}^w \sum_{y=1}^h |S_a(x,y) - S_b(x,y)|}{w * h} \quad (4.3)$$

$$\delta_{a,b}^V(w,h) = \frac{\sum_{x=1}^w \sum_{y=1}^h |V_a(x,y) - V_b(x,y)|}{w * h} \quad (4.4)$$

The final distance measurement  $D$  is then calculated by averaging the component's results, as seen in Equation 4.5.

$$D_{a,b}(w,h) = \frac{\delta_{a,b}^H(w,h) + \delta_{a,b}^S(w,h) + \delta_{a,b}^V(w,h)}{3} \quad (4.5)$$

A shot boundary is detected if the distance  $D$  between two consecutive frames is above a given threshold  $t$ . Other available parameters are downscale factor, to reduce computation costs, and minimum shot length. The latter is useful in the cases where multiple, major changes in the histogram occur rapidly, like quick flashes of light. In our implementation we use the threshold  $t = 30$ , a downscale factor of 4 and minimum shot length of 1 second.

## 4.5 Feature Extraction

The main feature extraction process is responsible for converting an input video into a structured representation for later use in signature generation. It is also designed to work with streaming applications and, as such, works on a frame-by-frame basis. The pseudo code of this process can be seen in Algorithm 1.

In short, each frame received is compared to the previous for shot detection. When a new shot is detected, all the information gathered previously is condensed into a single structure, representing an analyzed shot. This is then appended to a list of shots, which will later represent an analyzed video. Each shot is composed of scene information and tracked objects per frame. The former is generated by running our Places365-CNN implementation on a defined interval of half a second. The scene category confidences are averaged out and the scene attributes are counters for number of occurrences in all measurements made.

**Algorithm 1** Feature Extraction Process

---

```

1:  $i \leftarrow 0$ 
2:  $shots \leftarrow$  empty list
3:  $currentShot \leftarrow$  NEWSHOT( $i$ )
4: while  $frame \leftarrow$  GRABFRAME() do
5:   if ISSHOTBOUNDARY( $frame$ ) then
6:      $i \leftarrow i + 1$ 
7:      $shots \leftarrow$  APPEND( $shots, currentShot$ )
8:      $currentShot \leftarrow$  NEWSHOT( $i$ )
9:   end if
10:   $currentShot \leftarrow$  SCENERECOGNITION( $currentShot, frame$ )
11:   $detections \leftarrow$  DETECTOBJECTS( $frame$ )
12:   $currentShot \leftarrow$  TRACKOBJECTS( $currentShot, detections$ )
13: end while
14: return  $shots$ 

```

---

Since the information obtained during the feature extraction process can be considered high-level and semantically relevant, we can use it to visually annotate the videos being processed, as seen in Figure 4.7. This allows for an intuitive understanding of what data the signature comparison process will be working with, which proved to be very useful during the research and development stages.

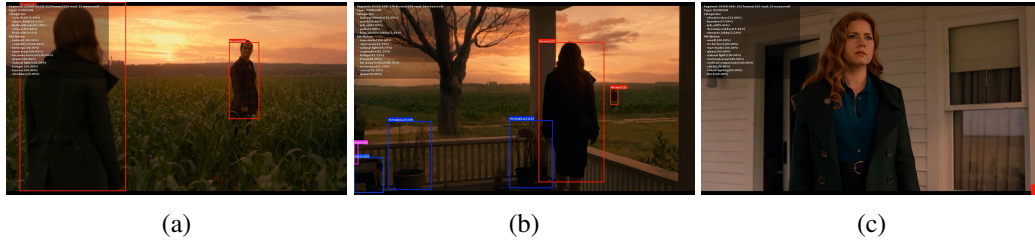


Figure 4.7: Various types of possible video annotations. (a) and (b) show scene information and tracked objects. The red square in the lower right corner of (c) indicates the current frame is a shot boundary

## 4.6 Signature Generation and Comparison

The final step in the video similarity process is to generate a signature using the features previously extracted. In our case, a video signature is simply a collection of shot signatures and additional meta-data extracted from the input file or stream, such as name, duration, file size, frame dimensions and encoding format. Shot signatures are generated using the scene and tracked objects information available. They are represented as a feature vector with four feature sets: scene type, scene category predictions, scene attributes frequency and number of distinct objects per class.

## Proposed Solution

Table 4.2 gives an in-depth description for each feature set and Figure 4.8 serves as a visual representation of the whole feature vector.

Feature Set	Size	Values	Description
Scene Type	1	$\{-1, 1\}$	Represents an indoor (1) or outdoor (-1) scene
Scene Category Predictions	365	$[0, 1]$	Confidence value per scene category as returned by the scene recognition network
Scene Attributes Frequency	102	$[0, 1]$	Relative frequency given by $f_a = \frac{n_a}{N}$ , where $a$ is an attribute, $n_a$ its number of occurrences and $N$ the total number of scene measurements made
Distinct Class Objects	80	$[0, +\infty[$	Number of distinct objects per class that appear in the shot, calculated by counting the distinct IDs of tracked objects per class

Table 4.2: Feature sets part of the shot signature feature vector

Comparison between signatures is done using the Cosine Similarity metric (Equation 4.6), where the resulting value  $s \in [-1, 1]$ ,  $-1$  meaning completely dissimilar and  $1$  meaning exactly the same. A few transformations are done on the feature vectors before comparison: feature sets are normalized as unit vectors and then the whole feature vector is also normalized as a unit vector. This effectively makes cosine similarity the same as the dot product (Equation 4.7), often called the Cosine Normalization. Additionally, any columns between the feature vectors being compared which only contain the value 0, meaning that none of the shots being compared have a specific feature, are discarded in order to improve performance. Our micro-benchmarks showed an improvement of 13% in comparison speed.

$$\text{similarity}(S_1, S_2) = \cos(\theta) = \frac{S_1 \cdot S_2}{\|S_1\| \|S_2\|} \quad (4.6)$$

$$\text{similarity}(S_1, S_2) = S_1 \cdot S_2, \text{ if } \|S_1\| = \|S_2\| = 1 \quad (4.7)$$

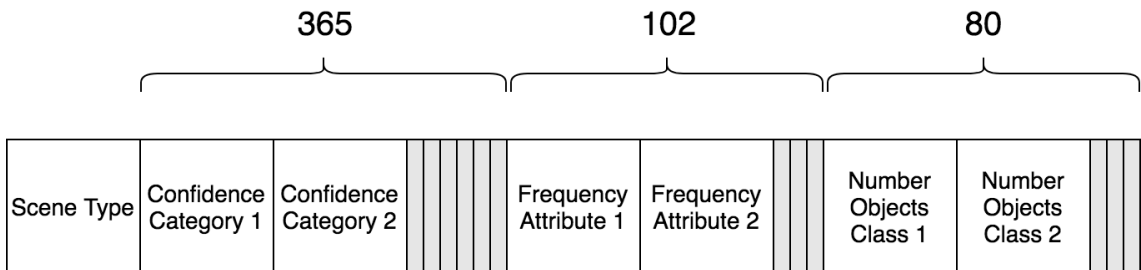


Figure 4.8: Visual representation of a shot signature (feature vector)

## Proposed Solution



## Chapter 5

# Experimental Results

In this chapter we show the results of several experiments conducted on the proposed system to evaluate its performance, both in terms of computational efficiency as well as various video similarity tasks. We begin with some observations on visual and semantic similarity tests, followed by the various problems and edge cases identified. We then show the results obtained and conclusions drawn from benchmarking the feature extraction, signature generation and signature comparison processes. Finally, we measure the system’s performance in the task of near-duplicate video detection using the CC\_WEB\_VIDEO dataset.

### 5.1 Initial Observations

#### 5.1.1 Semantic Similarity

Given that the features used in the shot signatures aren’t based on any pixel information but rather only its contents, we should expect the system to be able to match semantically similar shots together, even if their visual representation is substantially different. In our case, since we are using scenery information along with the number of different objects present, semantic similarity would mean any two shots that have close to the same number of objects on a similar scene. As the scenery information is dependant on the visible portion of the background, we should also expect shots to show different degrees of similarity depending on their field size or camera distance—can be close-up, medium or long shots—and camera angle.

We first tested this hypothesis by comparing videos with themselves. Since it is common for a video to show a specific scene from a variety of camera angles and distances, the system should successfully match a shot not only with itself but also with the other shots of that particular scene. As an example, figures 5.1 and 5.2 show a series of shots from GoldenEye that happen throughout a one minute time-frame. The shots from 5.2 are intertwined with the the shots from 5.1: 5.2a is between 5.1a and 5.1b, 5.2b is between 5.1b and 5.1c, and so on. We can see our system

## Experimental Results

successfully grouped the similar shots together, measuring over 93 % similarity for all shot pairs in Figure 5.1 and over 90 % for the pairs in Figure 5.2.

We also ran the same experiment but with one of the videos being sped up (1.5x and 2x) or slowed down (0.75x and 0.5x). The results obtained were largely the same, except for some shot boundaries that were not marked in the sped up versions because the resulting shot's duration would be lower than the configured minimum of 1 second. This confirms the system's invariance to time-scale changes previously mentioned in Section 4.1.

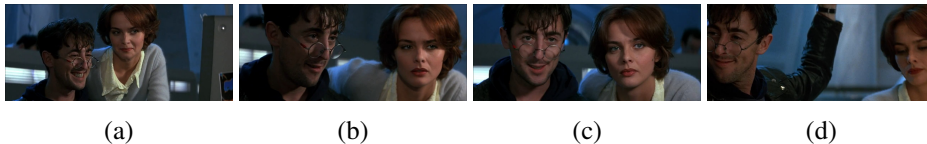


Figure 5.1: All 4 shots have a similarity value above 93% between each other

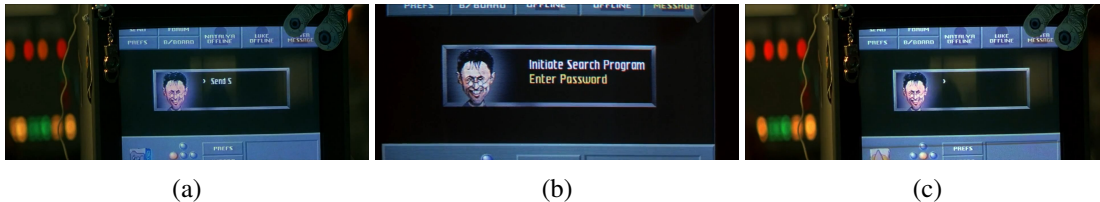


Figure 5.2: All three shots have a similarity value above 90% between each other

To further test the limits of semantic similarity, we fed the system completely unrelated videos but with similar context. Figure 5.3 shows two different videos, each having only a single shot. The similarity value of 78 % is justified by the fact that the scene recognition task identifies crosswalk (48 % and 80 %) and sidewalk (29 % and 18 %) as the two most likely categories for both videos. The remaining 22 % are the result of the differences between objects, as the shot (b) has more people (28 vs 17) and also includes vehicles (18 in total).



Figure 5.3: Semantically similar shots with a measured value of 78 %

Another example can be seen in Figure 5.4 which shows the impact of different camera angles and distance. None of the shots contain any objects, so only the scene recognition information is

## Experimental Results

being used. Since the shot (b) is a close-up, there isn't enough contextual information to correctly categorize the scene, which results in the top predicted categories to be evenly split in confidence scores. On the other hand, despite the shot (a) having a similar angle, it extends farther into the distance and, consequently, shows more contextual information, which directly translates into a more confident prediction. The resulting similarity value of 60 % can then be explained by the close match between scene attributes. A summary of the scene recognition results for both shots can be seen in Table 5.1.

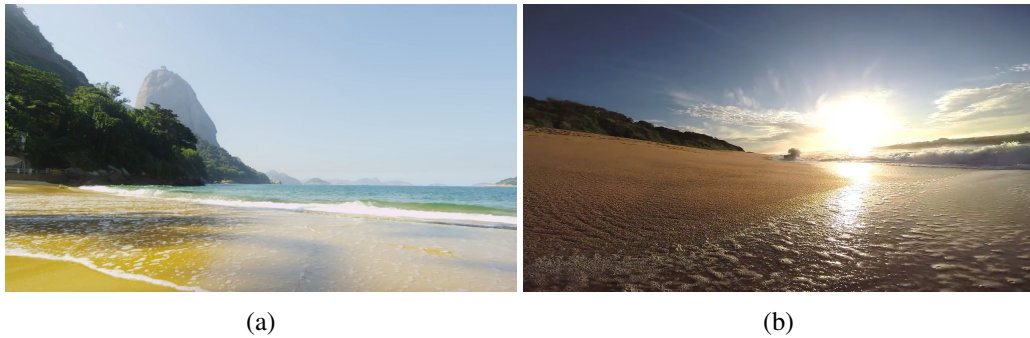


Figure 5.4: Similar locations and context but different camera angles and distance: 60 % similar

		<b>Shot (a)</b>	<b>Shot (b)</b>	
<b>Type</b>		Outdoor	Outdoor	
<b>Categories</b>	Lagoon	41 %	17 %	Coast
	Coast	14 %	13 %	Beach
	Ocean	12 %	12 %	Cliff
	Islet	9 %	8 %	Ocean
	Beach	9 %	8 %	Sky
<b>Attributes</b>		Boating	Boating	
		Far-away horizon	Far-away horizon	
		Ocean	Ocean	
		Clouds	Clouds	
		Sunny	Sunny	
		Swimming	Swimming	
		Natural	Natural	
		Open Area	Open Area	
		Natural Light	Natural Light	

Table 5.1: Scene recognition summary for the shots in Figure 5.4

### 5.1.2 Visual Transformations

Most commonly, NDV systems are designed to detect similarity by comparing visual information only. A useful property of such systems is the ability to successfully match near-duplicate videos

## Experimental Results

even in the presence of various visual transformations, such as cropping, scaling, rotating, flipping and color grading.

Our system works on top of features extracted from CNNs. An often employed technique when training such networks is image augmentation, which is essentially the process of generating multiple different versions of an original source image by applying different kinds of transformations on it. As such, it is expected that our system should also be invariant to most kinds of visual transformations.

We tested this assumption on videos modified by ourselves with various transformations. A subset of those videos can be seen in Figure 5.5 along with their similarity score based on the original video (a). Overall, the system proved to be reliable to most types of transformations, obtaining over 85 % similarity for most combinations of common changes such as horizontal flip, rotation, scaling, text overlay and blurring, as seen in Figures (b), (e) and (f). Although, it is important to note two particular types of cases: major color changes as seen in (c) and (d), and major rotations/vertical flip as seen in (d). Color differences mostly affect scene recognition, leading to unpredictable classifications and, in the case of (d), coupled with the vertical flip, it even led to the network classifying the scene as being indoor, opposed to outdoor like the original video. This is what led to such a low score of 13 %, taking into consideration the negative impact from different scene types.

While major rotations alone aren't able to automatically make every scene be misclassified, we did observe a big decrease in the performance of the object detection task, resulting in a lot less detections. To be specific, rotations between  $90^\circ$  and  $180^\circ$  in any direction lead to a major drop in performance for object detection and a substantial drop for scene recognition.

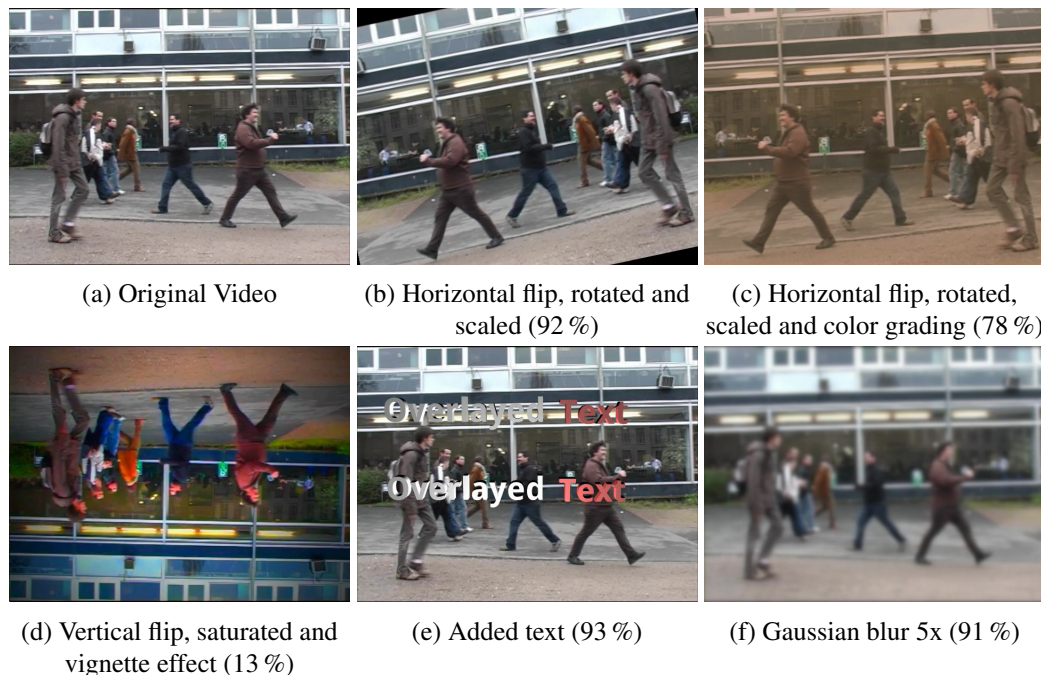


Figure 5.5: Examples of manually applied visual transformations and their measured similarities

### 5.1.3 Problems

Through the various experiments done it was possible to identify three major problems or edge cases. The first one is related to the scene type misclassification previously mentioned. This can be observed in a variety of cases where changes in color can tip the balance between the two scene types. While these particular cases could probably be solved by changing the scene type representation from a simple binary value to a quantitative one, the problem is most often observed in cases where the actual video doesn't fill the whole screen, such as the existence of black bars in either direction. Given that there is no pre-processing done on the video itself before sending it to the CNNs, all the pixel data is taken into consideration for classification. Although this doesn't pose any issue in object detection, it leads to what we call the "movie theater problem" in scene recognition, where every video with any kind of black bars results in a very high confidence score in the movie theater scene category and, consequently, the indoor scene type. An example of this can be seen in Figure 5.6, which shows some outdoor shots from the Justice League trailer classified as indoor and movie theater with over 44 % confidence. Almost all of the outdoor scenes of the trailer were wrongly classified as indoor.

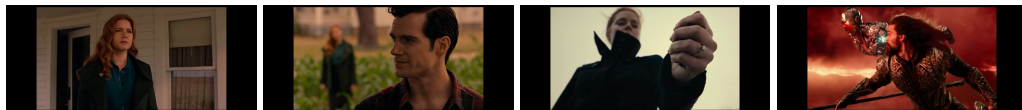


Figure 5.6: Black bars are taken into account during scene recognition, which wrongly classifies all of these shots as being indoor

The second major problem are semantically ambiguous shots, which commonly get clustered together into a single big group. These shots are usually close-ups, such as the examples in Figure 5.7: low light conditions and/or no background information, and low number of objects. Since the background information isn't enough to get a confident scene classification score, and the difference between the number of objects isn't significant, none of these shots have any identifying features. As such, they show a high degree of similarity between each other, even if they can be observed to be completely different. The only possible solution to this problem is to include additional information in the signature, whether from meta-data, other feature extraction techniques, or even entirely new image and video recognition CNNs.

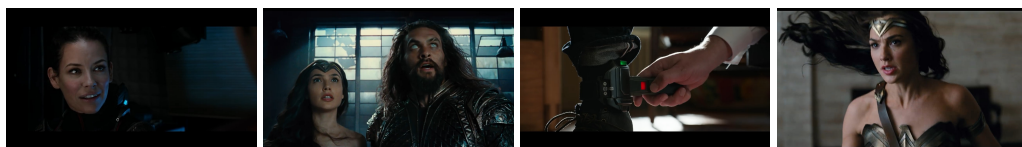


Figure 5.7: Ambiguity in semantic information leads to all these shots being considered highly similar (around 70 %). They have a small number of common objects (one or two people) and almost no relevant background information

The third and last problem identified is related to the method used for shot boundary detection. Since we use the color difference between frames, anything that alters the color information for

## Experimental Results

a brief moment leads to the detection of an incorrect shot boundary. This is most often observed with flashes of light, which cause a fast change in brightness levels. When occurred consecutively, it results in the detection of shot boundaries very close to each other in time, as shown in the two examples of Figure 5.8.

We mitigate this problem by enforcing a minimum shot length of one second, as previously seen in Section 4.4. However, while this prevents identifying successive boundaries caused by flashes of light, it still counts the first of those boundaries as correct, which might not always be the case. Furthermore, by enforcing a minimum shot duration, we might prevent the system from identifying actual shot boundaries if they happen too quick.

Incorrect shot boundary detection has many implications, as it can drastically change the resulting signature depending on where in the shot those boundaries are detected. An easy exploit could be to artificially insert a small number of frames in random places on a video in order to trick the system. Sun and Zhang [SZ17] propose an approach to this problem by splitting the boundary detection into two steps. The first step is the same as our current method, which is to detect based on the HSV color histogram. The second step then uses image perceptual hashing to re-detect and eliminate any fake boundaries.



Figure 5.8: Fast consecutive changes in color information, specially brightness, leads to the incorrect detection of multiple shot boundaries

## 5.2 Benchmarks

Different kinds of NDV applications have different requirements in respect to both performance and accuracy. While NDV systems are usually separated into two distinct processes, the type and size of information obtained during feature extraction directly influences the amount of work necessary in the later signature generation and comparison phases.

Generally, it is acceptable to have less than desirable performance during the feature extraction process, which is usually done offline, in order to obtain an accurate and space efficient representation of a video. This, in turn, eases the computational load during the generation and comparison of video signatures, which is typically an online process. However, for certain applications, such as real-time video monitoring, some of the input videos aren't available beforehand and, as such, the feature extraction process has to be able to run concurrently with the signature generation and comparison processes.

## Experimental Results

In order to get more insight into the possible applications and performance improvements of our system, we have ran extensive benchmarks for all of the underlying computations. In this section we show the results obtained as well as identify possible bottlenecks and improvements. The machine specifications where the benchmarks were ran on can be seen in Table 5.2.

---

<b>Operating System</b>	Ubuntu 18.04 LTS
<b>CPU</b>	Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz
<b>RAM</b>	16GB DDR3 @ 1600 MHz
<b>GPU</b>	GeForce GTX 1080 8GB GDDR5X
<b>DISK</b>	HDD 500GB 7200 RPM 32MB SATA 6Gb/s

---

Table 5.2: Benchmarking machine specifications

### 5.2.1 Feature Extraction

Feature extraction is the main module in our system and is responsible for converting an input video into a structured representation of high level features. The performance of its various sub-tasks varies widely depending on the input video provided. To provide a baseline performance report, all the measurements were made by running the extraction process single-threaded on 200 different videos with various resolutions, duration and formats. Tables 5.3 to 5.5 show the results for the different sub-tasks and Table 5.6 the results for the whole process, where running time is referring to the average time taken in each iteration or frame processed.

---

<b>Component</b>	<b>Execution Time</b>	<b>Component</b>	<b>Execution Time</b>
Pre-Processing*	3.43 ms	Pre-Processing*	4.39 ms
Forward Pass	5.53 ms	Region Proposal	21.27 ms
Results Processing	0.47 ms	Non-Maximum Suppression	3.27 ms
<b>Total</b>	<b>9.43 ms</b>	<b>Total</b>	<b>28.93 ms</b>

---

(a) Scene Recognition using Places365-WideResNet18

(b) Object Detection using YoloV3

Table 5.3: Benchmark results for Object Detection and Scene Recognition

\*Includes resizing input frame and transferring it to the GPU

## Experimental Results

Component	Execution Time	Component	Execution Time
Predict Step	0.15 ms	Pre-Processing**	0.01 ms
IoU Matching	0.32 ms	Predict & Update	3.92 ms
Update Step	0.37 ms	<b>Total</b>	<b>3.93 ms</b>
Post-Processing*	0.10 ms	(b) Multiple class tracking	
<b>Total</b>	<b>0.94 ms</b>		

(a) Single class multiple object tracking using Kalman Filter

Table 5.4: Benchmark results for Object Tracking

\*Includes checking for "dead" trackers and converting the Kalman Filter states back into bounding boxes

\*\*Grouping input detections by object class

As expected, the main bottlenecks are related to the use of CNNs in the object detection and scene recognition tasks. However, it is worth noting the impact of the pre-processing steps in both (15% for object detection and 36% for scene recognition). This is mostly due to the constant moving of memory from the CPU to the GPU, which accounts for around 45 % of the pre-processing time. While this isn't as significant for scene recognition, which only runs every half a second, as can be seen by the resulting running time of 1.57 ms in Table 5.6, it is extremely detrimental to the object detection task, which runs every frame. A possible solution would be to send the input frames as a batch, at the expense of a slight delay in real-time applications because of the need to buffer incoming frames.

Another interesting observation is related to object tracking. In Table 5.4a we can see it takes an average of 1 ms to track all the objects of a single class. Also, Table 5.4b shows the average tracking time for all objects is around 4 ms. A possible conclusion is that, on average, there are four different classes of objects being tracked at any given time. Since the tracking of different classes is completely independent from each other, taking advantage of parallel computing by having a process per class tracker would likely yield a performance improvement.

Component	Execution Time
Pre-Processing*	0.85 ms
HSV Delta	0.61 ms
<b>Total</b>	<b>1.46 ms</b>

Table 5.5: Benchmark results for Shot Boundary Detection

\*Includes downscaling input frame and RGB to HSV conversion



## Experimental Results

Task	Metrics	
	Execution Time	Running Time
Shot Boundary Detection	1.46 ms	1.41 ms (709 fps)
Scene Recognition	9.43 ms	1.57 ms (637 fps)
Object Detection	28.93 ms	30.09 ms (33 fps)
Object Tracking	3.93 ms	4.09 ms (245 fps)
<b>Total</b>	-	37.16 ms (27 fps)

Table 5.6: Benchmark results for the whole feature extraction process. Running time refers to the average time a task takes per iteration/frame processed

One of the conclusions we take from the results in Table 5.6 is that, in our machine, the current implementation with a throughput of 27 fps can't be reliably used for real-time applications, at least not without some kind of external pre-processing on the incoming stream, like downscaling or throttling the frame-rate. This is taking into consideration that the majority of the videos tested were 480p@24 fps and that the throughput can vary wildly depending on the number of objects being tracked.

The other conclusion is that the major bottleneck is the object detection process. Initial testing revealed that, on object detection alone, the previously mentioned batching solution could likely improve the throughput by 10 %, reaching a total of 30 frames per second. Further improvements should be focused on taking advantage of multiple GPUs for the forward passes in the CNNs, using an alternative Non-Maximum Suppression implementation that takes advantage of graphics cards and, finally, the before mentioned parallel object tracking.

### 5.2.2 Signature Generation and Comparison

The other module in the system is signature generation and comparison, which takes the results from the feature extraction module and processes them into a list of shot signatures (feature vectors) ready to be compared. The results of the benchmarks ran on this module can be seen in Table 5.7. On average, the videos used have a duration between 3 to 7 minutes and contain 15 to 30 shots.

Just like in feature extraction, the performance here also varies depending on the input video, specially during the comparison phase, which measures the similarity between two videos by comparing every possible shot pair. As such, comparison time increases exponentially with every additional shot that needs to be compared. We conclude that the observed average of 1.975 ms per comparison, which translates to about 500 comparisons per second, poses no significant overhead for real-time applications. Although, it is worth noting that the timings shown in 5.7a refer to the generation of a single shot's signature, and that the execution time of the object counting step can quickly become a bottleneck if signature generation is done in real-time.

## Experimental Results

Feature Set	Execution Time	Component	Execution Time
Scene Type	0.006 ms	Pre-Processing*	1.309 ms
Scene Attributes	0.015 ms	Cosine Similarity	0.535 ms
Scene Categories	0.033 ms	Filter by Threshold	0.033 ms
Objects Count	0.549 ms	Segment Matching	0.098 ms
<b>Total</b>	<b>0.603 ms</b>	<b>Total</b>	<b>1.975 ms</b>

(a) Signature Generation

(b) Signature comparison

Table 5.7: Benchmarks for the signature generation and comparison processes

\*Includes removal of columns with all zeros and creating a matrix of every possible shot-pair combination

### 5.3 Near-Duplicate Video Detection

We conducted experiments on the CC\_WEB\_VIDEO dataset [WHN07, WNHT09]. It consists of a sample of videos retrieved by submitting 24 text queries to popular video sharing websites like YouTube and Yahoo! Video. For every query, a set of video clips ordered by descending popularity was collected. The most popular video of the set is considered to be the original video. Subsequently, the rest of the videos were manually annotated based on their near-duplicate relation to the original video. Some examples of the entries in the dataset can be seen in Figure 5.9, where the leftmost column are the original videos. The dataset is widely used in literature [KZPPK17, JTL<sup>+</sup>17, WHN07, TWNZ08] as a good performance baseline for near-duplicate video detection, since it reflects the real user behavior on transformations used in the generation of near-duplicates.

Table 5.8 depicts the different available near-duplicate annotations. In the present work, we consider videos annotated with anything but **X** or **M** to be near-duplicates. Although the dataset contains a total of 13,129 videos collected from 24 queries, we only use a small subset, given the time it would take to run the feature extraction on all of them, as seen previously in the benchmarks section (5.2). Our results are based on a total of 2537 videos gathered from the queries 1, 2, 7, 13, 14, 16 and 20 (Table 5.9).

Annotation	Meaning
<b>E</b>	Exactly duplicate
<b>S</b>	Similar video
<b>V</b>	Different version
<b>M</b>	Major change
<b>L</b>	Long version
<b>X</b>	Dissimilar video

Table 5.8: Similarity annotations available in the CC\_WEB\_VIDEO dataset

The commonly used performance metrics in NDV applications are precision, recall and Average Precision (AP) [LHCS13]. The usual visualization is the Precision-Recall (PR) curve, which

## Experimental Results

Query	Videos	Near-Duplicates
<b>1</b>	812	334 (41 %)
<b>2</b>	427	87 (20 %)
<b>7</b>	365	154 (42 %)
<b>13</b>	419	387 (92 %)
<b>14</b>	108	72 (67 %)
<b>16</b>	208	20 (10 %)
<b>20</b>	198	72 (36 %)
<b>Total</b>	2537	1126 (44 %)

Table 5.9: Summary of the subset of videos used from the CC\_WEB\_VIDEO dataset

is a useful graph that provides a good intuition of the system’s predictive performance in the case of a very imbalanced class distribution. However, since our subset’s class distribution is well balanced, with 44 % near-duplicates and 56 % dissimilar, we also include the Receiver Operating Characteristic (ROC) curve and its Area Under Curve (AUC) measure as additional metrics.

Table 5.10 shows the Average Precision and ROC AUC for every query, and Figure 5.10 shows the PR and ROC curves of the total subset. The resulting AP value of 0.974 matches that of the state-of-the-art as seen in [KZPPK17, Section 5.4], which varies between 0.892 for Color Histogram approaches and 0.976 for CNN based approaches. Future work includes the analysis of the whole dataset, to further increase the confidence in this value.

Query	AP	ROC AUC
<b>1</b>	0.99383	0.99486
<b>2</b>	0.99602	0.99912
<b>7</b>	0.99361	0.99221
<b>13</b>	0.99960	0.99532
<b>14</b>	0.95838	0.91358
<b>16</b>	1.00000	1.00000
<b>20</b>	0.97283	0.97906

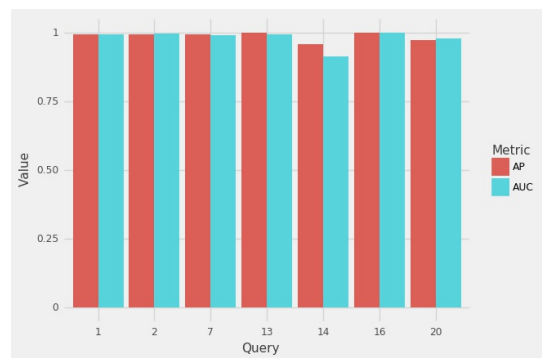


Table 5.10: Average Precision and ROC AUC measures for every query done

In order to better assess the applicability of the system in real applications, we also ran Cross Validation using the K-Fold method. Essentially, we randomly split our subset of videos into  $k$  equally sized subsamples. Then, one of the subsamples is used for validation while the remaining  $k - 1$  are used for training. In our case, we use the training subsamples to derive the optimal similarity threshold for the system, which is then used on the validation subsample to measure the performance. This process is repeated  $k$  times, with each of the  $k$  subsamples used exactly once as the validation data. In the end, the validation results as well as the optimal thresholds are averaged in order to obtain the final estimate.

Taking into consideration the size of our subset, a value  $k = 5$  was used, which means each of

## Experimental Results

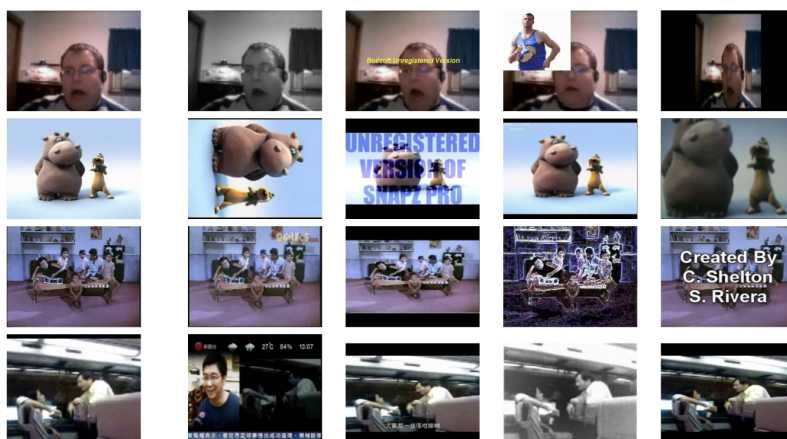


Figure 5.9: Examples of queries (first column on the left) and near-duplicate videos from the CC\_WEB\_VIDEO dataset

the folds have on average 2030 videos for training and 507 videos for validation. The individual and final averaged results can be seen in Table 5.11, accompanied by a confusion matrix in Figure 5.11. With this, we conclude that our implementation is slightly optimistic, as explained by the high recall value (0.9655) and lower precision (0.8655), as well as the number of false positives being 4 times higher than false negatives (168 vs 41). Further investigation into the false cases concluded our initial observations of the system problems. A majority of the false positives resulted from shot ambiguity, while 38 of the 41 false negatives resulted from the presence of black bars or invalid shot boundaries.

Fold	Threshold	Metrics		
		Precision	Recall	F1 Score
1	0.7254	0.8320	0.9674	0.8946
2	0.7254	0.9027	0.9547	0.9280
3	0.7310	0.8823	0.9677	0.9230
4	0.7310	0.8450	0.9732	0.9046
5	0.7254	0.8656	0.9647	0.9125
<b>Average</b>	0.7276	0.8655	0.9655	0.9125

Table 5.11: Results from K-Fold Cross Validation

## Experimental Results

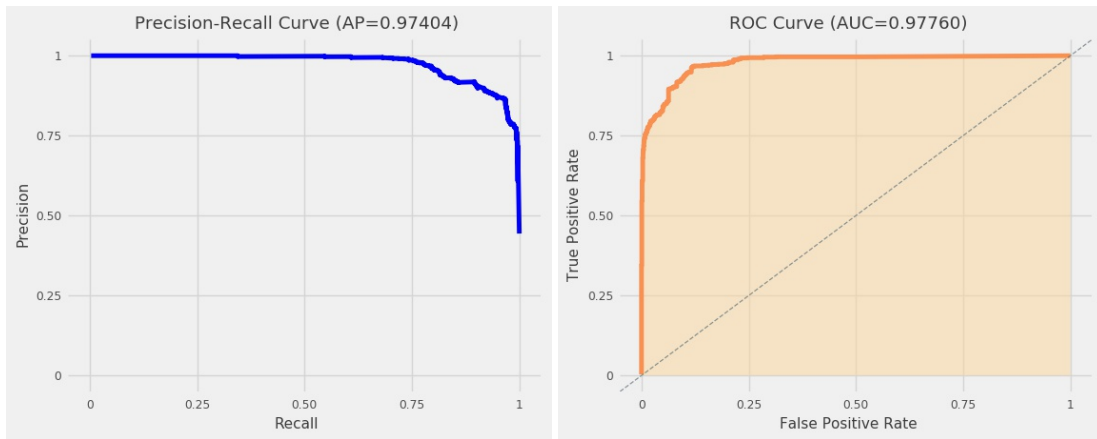


Figure 5.10: Precision-Recall and ROC curves of the whole subset (2537 videos)

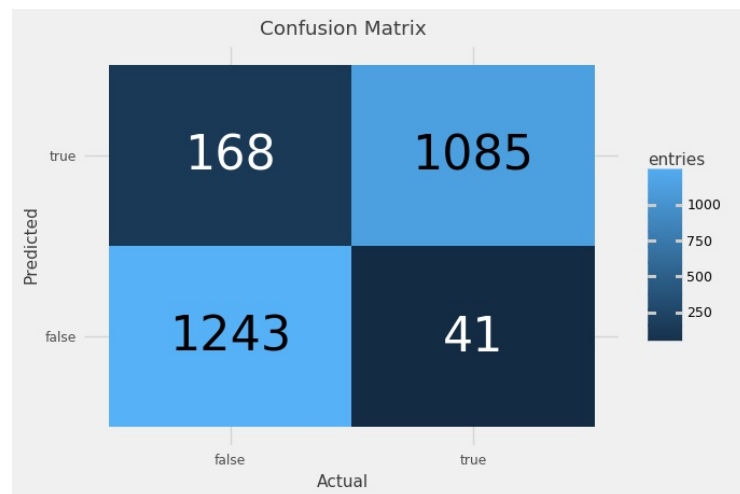


Figure 5.11: Resulting confusion matrix from K-Fold Cross Validation

## Experimental Results

## Chapter 6

# Conclusions and Future Work

In this work, we have presented a shot-based video signature generation solution that uses Convolutional Neural Networks to extract high-level features from video data. With it, we showed that utilizing only semantic information, such as objects and scenery information, to represent a video not only results in compact signatures and fast comparison time, but also good performance in the task of near-duplicate video detection, reaching an Average Precision value of 0.974 that is competitive with the state-of-the-art. This supports the assumption by Basharat et al. [BZS08] and Cherubini et al. [CdOO09] that semantic similarity might play a bigger role in the definition of near-duplicate videos than previously thought.

Using semantic information also proved to be very helpful during the research and development stages. It enabled annotating the analyzed videos with visual representations of the underlying data that are easily interpretable by humans. This, in turn, was useful for understanding the results returned by the system as well as the further optimization of the signature comparison process.

Representing videos as a collection of shots enables expanding the current solution into other applications. For example, the representation of videos and their shots as a graph database, in which similarity is given as a relation between shots, could prove very useful for video tracking cases, such as a news outlet who wants to do a report on an event by following a connection of online articles which contain recordings of that same event.

As the system is designed to be easily extended, further improvements can be done in a number of areas depending on the desired application. First, in order to increase the precision and mitigate the problem of shots without relevant information, results from other image and video recognition networks can be used by just appending them as a new feature sets in the video signature. Second, as the feature extraction process is highly modular, a distributed solution like the one proposed by Jiang et al. [JTL<sup>+</sup>17] could be applied in order to decrease the bottleneck that CNNs introduce in the whole feature extraction pipeline. Finally, to allow for applications that require fast retrieval

## Conclusions and Future Work

in databases with a lot of entries, signature indexing methods, such as tree-like structures and hashing, can be added.



# References

- [AWB03] James Allan, Courtney Wade, and Alvaro Bolivar. Retrieval and novelty detection at the sentence level. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '03*, page 314, New York, New York, USA, 2003. ACM Press.
- [BZS08] Arslan Basharat, Yun Zhai, and Mubarak Shah. Content based video matching using spatiotemporal volumes. *Computer Vision and Image Understanding*, 110(3):360–377, 6 2008.
- [CdOO09] Mauro Cherubini, Rodrigo de Oliveira, and Nuria Oliver. Understanding Near-duplicate Videos: A User-centric Approach. In *Proceedings of the 17th ACM International Conference on Multimedia*, number April in MM '09, pages 35–44, New York, New York, USA, 2009. ACM Press.
- [Cis17] Cisco. The Zettabyte Era: Trends and Analysis. Technical Report June 2017, Cisco, 2017.
- [DLHS16] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. *CoRR*, abs/1605.0, 5 2016.
- [dOCO09] Rodrigo de Oliveira, Mauro Cherubini, and Nuria Oliver. Human Perception of Near-Duplicate Videos. In Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates, and Marco Winckler, editors, *Human-Computer Interaction – INTERACT 2009*, pages 21–24, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [Gaj10] Padalkar Milind Gajanan. *Histogram Based Efficient Video Shot Detection Algorithms*. PhD thesis, Sardar Vallabhbhai National Institute of Technology, 2010.
- [GDDM13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2, 11 2013.
- [Gir15] Ross Girshick. Fast R-CNN. *CoRR*, abs/1504.0, 4 2015.
- [GRG<sup>+</sup>18] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. Retrieved from <https://github.com/facebookresearch/detectron>, 2018. Accessed 2 June 2018.

## REFERENCES

- [HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *CoRR*, abs/1703.0, 3 2017.
- [HHC<sup>+</sup>10] Zi Huang, Bo Hu, Hong Cheng, Heng Tao Shen, Hongyan Liu, and Xiaofang Zhou. Mining Near-duplicate Graph for Cluster-based Reranking of Web Video Search Results. *ACM Trans. Inf. Syst.*, 28(4):22:1–22:27, 11 2010.
- [HLCD13] Zi Huang, Jiajun Liu, Bin Cui, and Xiaoyong Du. A Gram-Based String Paradigm for Efficient Video Subsequence Search. *IEEE Transactions on Multimedia*, 15(3):608–620, 4 2013.
- [HRS<sup>+</sup>16] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.1, 11 2016.
- [HSS<sup>+</sup>09] Zi Huang, Heng Tao Shen, Jie Shao, Xiaofang Zhou, and Bin Cui. Bounded coordinate system indexing for real-time video clip search. *ACM Transactions on Information Systems*, 27(3):1–33, 5 2009.
- [Hua96] T. Huang. Computer Vision: Evolution And Promise. *19th CERN School of Computing*, pages 21–25, 1996.
- [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Multimedia Tools and Applications*, 77(9):10437–10453, 12 2015.
- [JN09] Yu-Gang Jiang and Chong-Wah Ngo. Visual word proximity and linguistics for semantic video indexing and near-duplicate retrieval. *Computer Vision and Image Understanding*, 113(3):405–414, 3 2009.
- [JTJ14] Jong-Min Jeong, Tae-Sung Yoon, and Jin-Bae Park. Kalman filter based multiple objects detection-tracking algorithm robust to occlusion. In *2014 Proceedings of the SICE Annual Conference (SICE)*, pages 941–946. IEEE, 9 2014.
- [JTL<sup>+</sup>17] Jiawei Jiang, Yunhai Tong, Hua Lu, Bin Cui, Kai Lei, and Lele Yu. GVoS: A General System for Near-Duplicate Video-Related Applications on Storm. *ACM Transactions on Information Systems*, 36(1):1–36, 6 2017.
- [Kar14] Andrej Karpathy. What I learned from competing against a ConvNet on ImageNet. Retrieved from <https://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet>, September 2014. Accessed 7 February 2018.
- [KM15] Rahul Kumar and Sukadev Meher. A Novel method for visually impaired using object recognition. In *2015 International Conference on Communications and Signal Processing (ICCSP)*, pages 0772–0776. IEEE, 4 2015.

## REFERENCES

- [Ko18] Byoung Ko. A Brief Review of Facial Emotion Recognition Based on Visual Information. *Sensors*, 18(2):401, 2018.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [KWM94] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In Jan-Olof Eklundh, editor, *Computer Vision — ECCV '94*, pages 189–196, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
- [KZPPK17] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Yiannis Kompatsiaris. Near-Duplicate Video Retrieval with Deep Metric Learning. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, volume 2018-Janua, pages 347–356. IEEE, 10 2017.
- [LAE<sup>+</sup>15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot Multi-Box Detector. *CoRR*, abs/1512.0, 12 2015.
- [LBX<sup>+</sup>15] Yao Liu, Sam Blasiak, Weijun Xiao, Zhenhua Li, and Songqing Chen. A Quantitative Study of Video Duplicate Levels in Youtube. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8995:235–248, 2015.
- [Lee17] Frank Lee. The Rise of Ubiquitous Computer Vision in IOT. Retrieved from <https://www.iotforall.com/computer-vision-iot>, 2017. Accessed 6 February 2018.
- [LHCS13] Jiajun Liu, Z Huang, Hongyun Cai, and HT Shen. Near-duplicate video retrieval: Current research and future trends. *ACM Computing Surveys* . . . , 45(4):1–23, 2013.
- [LTMR<sup>+</sup>15] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *CoRR*, abs/1504.0, 4 2015.
- [LW15] Fang Liu and Yi Wan. Improving the video shot boundary detection using the HSV color space and image subsampling. In *2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*, pages 351–354. IEEE, 3 2015.
- [LWWL10] Xin Li, Kejun Wang, Wei Wang, and Yang Li. A multiple object tracking method using Kalman filter. *Information and Automation (ICIA)* . . . , 1(1):1862–1866, 2010.
- [MFMR02] L. Marcenaro, M. Ferrari, L. Marchesotti, and C.S. Regazzoni. Multiple object tracking under heavy occlusions by using Kalman filters based

## REFERENCES

- on shape matching. In *Proceedings. International Conference on Image Processing*, volume 1, pages III–341–III–344. IEEE, 2002.
- [MLTR<sup>+</sup>16] Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A Benchmark for Multi-Object Tracking. *CoRR*, abs/1603.0, 3 2016.
- [MMM03] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559, 2003.
- [MTZ17] Bogdan Mocanu, Ruxandra Tapu, and Titus Zaharia. Seeing Without Sight — An Automatic Cognition System Dedicated to Blind and Visually Impaired People. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1452–1459. IEEE, 10 2017.
- [PAM<sup>+</sup>17] Mauricio Perez, Sandra Avila, Daniel Moreira, Daniel Moraes, Vanessa Testoni, Eduardo Valle, Siome Goldenstein, and Anderson Rocha. Video pornography detection through deep learning techniques and motion information. *Neurocomputing*, 230:279–293, 3 2017.
- [Per15] L Nathan Perkins. CONVOLUTIONAL NEURAL NETWORKS AS FEATURE GENERATORS FOR NEAR-DUPLICATE VIDEO DETECTION. Technical report, Boston University, 2015.
- [RB15] Juan Rosenzweig and Michael Bartl. A Review and Analysis of Literature on Autonomous Driving. *The Making-of Innovation*, com(October):1–57, 2015.
- [RDGF15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, abs/1506.0, 6 2015.
- [RDS<sup>+</sup>15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [RF16] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. *CoRR*, abs/1612.0, 12 2016.
- [RF18] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.0, 4 2018.
- [RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, abs/1506.0, 6 2015.
- [San15] Sandvine. Global Internet Phenomena Report: Africa, Middle East & North America. Technical report, Sandvine, 12 2015.

## REFERENCES

- [SLJ<sup>+</sup>14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv:1409.4842*, 2014.
- [SSBNMSBJSVSB15] Adnan Siddiqui SOIT, RGPV Bhopal Nischcol Mishra SOIT, RGPV Bhopal Jitendra Singh Verma SOIT, and RGPV Bhopal. A Survey on Automatic Image Annotation and Retrieval. *International Journal of Computer Applications*, 118(20):975–8887, 2015.
- [SYW<sup>+</sup>10] Lifeng Shang, Linjun Yang, Fei Wang, Kwok-ping Chan, and Xian-sheng Hua. Real-time large scale near-duplicate web video retrieval. In *Proceedings of the international conference on Multimedia - MM '10*, page 531, New York, New York, USA, 2010. ACM Press.
- [SZ14] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, 51(4):769–784, 9 2014.
- [SZ17] Bin Sun and Dengyin Zhang. A method for video shot boundary detection based on HSV color histogram and DPHA feature. In *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC '17*, pages 1–4, New York, New York, USA, 2017. ACM Press.
- [TSZH<sup>+</sup>07] Heng Tao Shen, Xiaofang Zhou, Zi Huang, Jie Shao, and Xiangmin Zhou. UQLIPS: A real-time near-duplicate video clip detection system. *VLDB*, 2007.
- [TWNZ08] Hung-Khoon Tan, Xiao Wu, Chong-Wah Ngo, and Wan-Lei Zhao. Accelerating near-duplicate video matching by combining visual similarity and alignment distortion. In *Proceeding of the 16th ACM international conference on Multimedia - MM '08*, page 861, New York, New York, USA, 2008. ACM Press.
- [TXZ<sup>+</sup>07] Heng Tao, Shen Xiaofang, Zhou Zi, Huang Jie, and Shao Xiangmin. UQLIPS : A Real-time Near-duplicate Video Clip Detection System. *Vldb*, pages 1374–1377, 2007.
- [WG15] Kanika Wadhawan and E Gajendran. Automated Recognition of Text in Images : A Survey. *International Journal of Computer Applications*, 127(15):15–19, 2015.
- [WHN07] Xiao Wu, Alexander G. Hauptmann, and Chong-Wah Ngo. Practical elimination of near-duplicates from web video search. In *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07*, page 218, New York, New York, USA, 2007. ACM Press.
- [WNHT09] Xiao Wu, Chong-Wah Ngo, Alexander G. Hauptmann, and Hung-Khoon Tan. Real-Time Near-Duplicate Elimination for Web Video Search With Content and Context. *IEEE Transactions on Multimedia*, 11(2):196–207, 2 2009.
- [ZF13] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *CoRR*, 11 2013.

## REFERENCES

- [ZH06] J. Zobel and Timothy C. Hoad. Detection of video sequences using compact signatures. *ACM Transactions on Information Systems*, 24(1):1–50, 1 2006.
- [ZLK<sup>+</sup>17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [ZLX<sup>+</sup>14] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning Deep Features for Scene Recognition using Places Database. *Advances in Neural Information Processing Systems 27*, pages 487–495, 2014.
- [ZN15] Wan-Lei Zhao and Chong-Wah Ngo. Near-Duplicate Image and Video Detection. In *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 1–13. American Cancer Society, Hoboken, NJ, USA, 2015.