Aston University

Spatial and Temporal Representations for Multi-Modal Visual Retrieval

---

Noa Garcia Docampo

Doctor of Philosophy

September 28, 2018

Aston University

Spatial and Temporal Representations for Multi-Modal Visual Retrieval

Noa Garcia Docampo

Doctor of Philosophy

September 2018

This dissertation studies the problem of finding relevant content within a visual collection according to a specific query by addressing three key modalities: symmetric visual retrieval, asymmetric visual retrieval and cross-modal retrieval, depending on the kind of data to be processed.

In symmetric visual retrieval, the query object and the elements in the collection are from the same kind of visual data, i.e. images or videos. Inspired by the human visual perception system, we propose new techniques to estimate visual similarity in image-to-image retrieval datasets based on non-metric functions, improving image retrieval performance on top of state-of-the-art methods.

On the other hand, asymmetric visual retrieval is the problem in which queries and elements in the dataset are from different types of visual data. We propose methods to aggregate the temporal information of video segments so that image-video comparisons can be computed using similarity functions. When compared in image-to-video retrieval datasets, our algorithms drastically reduce memory storage while maintaining high accuracy rates.

Finally, we introduce new solutions for cross-modal retrieval, which is the task in which either the queries or the elements in the collection are non-visual objects. In particular, we study text-image retrieval in the domain of art by introducing new models for semantic art understanding, obtaining results close to human performance.

Overall, this thesis advances the state-of-the-art in visual retrieval by presenting novel solutions for some of the key tasks in the field. The contributions derived from this work have potential direct applications in the era of big data, as visual datasets are growing exponentially every day and new techniques for storing, accessing and managing large-scale visual collections are required.

Keywords: image retrieval, video retrieval, cross-modal retrieval

Acknowledgement

This work would not have been possible without the support of many people that helped me to go through this journey in many different ways, and to whom I am sincerely grateful.

First, I would like to express my gratitude to my supervisors Dr. George Vogiatzis and Dr. Maria Chli for the time they invested in me and their support during these years.

I would like to thank my lab mates, which are also my friends, for all the lunches together, all the fruitful discussions and all the laughs. Especially, thanks to Arezoo, Deepeka, Thomas, Luis, Aamir, Gabriele, Vishwash, Rasmus, Vania, and Reham.

Also, I would like to thank my qualifying report examiners, Prof. Yulan He and Prof. Ian Nabney, and my final PhD examiners, Dr Neill D.F. Campbell and Dr Yordan Raykov, for their helpful advice and discussion.

It would be unfair to not acknowledge this thesis to David Vilares, who supported me unconditionally during the last two years of my PhD in many different ways: from reviewing papers and discussing new ideas to just being there listening and understanding. This work would not have been the same without him.

Finally, I would like to thank my family: Carmela, Felix, Brais, for always supporting me. And especially my mum, for not allowing me to go back to Spain when things were being difficult in the UK.

Contents

List of Figures

List of Tables

Abbreviations

BOW Bag-of-Words

BRIEF Binary Robust Independent Elementary Features

CBIR Content-Based Image Retrieval

CNN Convolutional Neural Network

CV Computer Vision

FPS Frame per Second

FV Fisher Vectors

LSTM Long Short-Term Memory

mAP Mean Average Precision

R@K Recall at K

ReLU Rectified Linear Unit

RMAC Regional Maximum Activation of Convolutions

RNN Recurrent Neural Network

SIFT Scale Invariant Feature Transform

SURF Speeded-Up Robust Features

List of Publications

[1] Noa Garcia, George Vogiatzis (2019). Learning Non-Metric Visual Similarity for Image Retrieval. In: *Image and Vision Computing*. DOI: 10.1016/j.imavis.2019.01.001.

[2] Noa Garcia, George Vogiatzis (2018). How to Read Paintings: Semantic Art Understanding with Multi-Modal Retrieval. In: *European Conference on Computer Vision Workshops*, pp. 676-691.

[3] Noa Garcia, George Vogiatzis (2018). Asymmetric Spatio-Temporal Embeddings for Large-Scale Image-to-Video Retrieval. In: *British Machine Vision Conference*, pp. 1-13.

[4] Noa Garcia (2018). Temporal Aggregation of Visual Features for Large-Scale Image-to-Video Retrieval. In: *ACM International Conference on Multimedia Retrieval*, pp. 489-492.

[5] Noa Garcia, George Vogiatzis (2017). Dress like a Star: Retrieving Fashion Products from Videos. In: *IEEE International Conference on Computer Vision Workshops*, pp. 2293 - 2299.

[6] Noa Garcia, George Vogiatzis (2016). Exploiting Redundancy in Binary Features for Image Retrieval in Large-Scale Video Collections. In: *European Conference on Visual Media Production.*

Part I

Introduction and Background

1 Introduction

This thesis explores multiple modalities of visual retrieval, i.e. finding relevant samples in large collections of visual content by using different types of data queries, and presents a number of techniques for multi-modal visual search. Multi-modal visual search is classified into different modalities according to the type of data involved in the task. Specifically, we study three main multi-modal visual retrieval problems:

1. *symmetric visual retrieval*, in which dataset content and queries are from the same type of visual data (e.g. searching images with images);

2. *asymmetric visual retrieval*, in which dataset content and queries are from different types of visual data (e.g. searching videos with images);

3. *cross-modal retrieval*, in which dataset content and queries are from different type of data (e.g. searching images with text).

Throughout this thesis,, we introduce new methods and datasets for each of these modalities, contributing to advance the state-of-the-art in multi-modal retrieval tasks. The experiments conducted here show that our proposed methods outperform previous work in terms of both accuracy and efficiency, obtaining results close to human performance in high-level recognition tasks.

1.1 Motivation

In the era of information explosion, the amount of visual content stored in online platforms is growing exponentially every day. The incorporation of high-quality digital cameras to smartphones and the popularity of social media platforms, boosts the use of visual data in people's day-to-day communications. Nowadays, it is faster to share a picture of a meal than to describe it to a friend and it is easier to learn the latest make-up techniques by watching videos in Youtube than by reading beauty tips in magazines.

The explosion of visual data leads to the accumulation of images and videos in *very* large collections. As an illustration, according to the latest statistics[1], a million photos and videos are being created every day in Snapchat, 95 million photos are being uploaded daily to Instagram and 300 hours of video are being shared on Youtube *every minute*. In Facebook along, around 136,000 photos are being uploaded every 60 seconds.

Considering that these examples are just from some of the most popular social media platforms, with personal photo collections (i.e. photos that are not posted on social media) and private databases (e.g. medical records, surveillance cameras, etc.) excluded, the actual volume of images and videos that has been created in the digital era is beyond count. In this context, a specific image or video amongst a large collection of visual data is the proverbial needle in a haystack.

Computer vision (CV) techniques can assist in accessing and managing large-scale datasets of images and videos efficiently. Specifically, visual retrieval is the field in CV that finds relevant data within a visual collection according to a specific input. In visual retrieval, the visual content of images and videos is used to automatically decide whether an image or a video is relevant to the query input

[1] Statistics from: https://www.omnicoreagency.com/ [Accessed: 08 Jul. 2018]

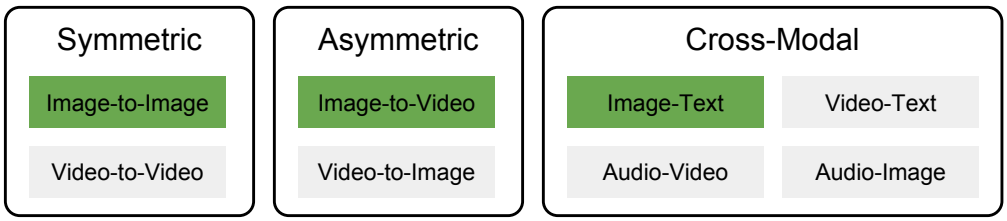


Fig. 1.1: Different types of multi-modal visual retrieval problems, categorized in three modalities: symmetric visual retrieval, asymmetric visual retrival and cross-modal retrieval. In green, the specific tasks studied in this dissertation.

or not. Depending on the application, the input used as query can be from a large range of data types.

Visual retrieval can be classified into three main modalities depending on the kind of query input and the kind of data to be retrieved. These three categories are: symmetric visual retrieval, asymmetric visual retrieval and cross-modal retrieval, as shown in Figure 1.1. The way of approaching a specific visual retrieval problem depends on the modality of the task.

In symmetric visual retrieval, both the query input and the dataset content belong to the same kind of visual data. This is the case of image-to-image retrieval (Smeulders et al., 2000; Zheng et al., 2018), in which a query image is used to rank the pictures within a collection according to their similarities. Another example is video-to-video retrieval (Geetha and Narayanan, 2008), which is used in automatic copy detection, and consists on finding videos that are similar to an original input video. In symmetric visual retrieval, the query and the dataset are commonly processed using the same techniques.

In asymmetric visual retrieval, however, query inputs and visual collections contain different types of visual data. For example, in image-to-video retrieval a query image is used to find videos (Araujo and Girod, 2017). Similarly, in video-to-image retrieval, a video is used to find pictures (Takacs et al., 2008). Whereas images are static, videos contain richer information, such as time, optical flow or motion. In asymmetric visual retrieval, the peculiarities of each data

type are considered independently, and thus, queries and dataset content are processed using asymmetric techniques.

In cross-modal retrieval, the data used for querying and the data in the visual collection are of completely different nature. This is the case of image-text retrieval (Karpathy et al., 2014), in which textual descriptions are used to find images, or audio-video retrieval (Ngiam et al., 2011), in which audio content is used to retrieve videos. Although there are many types of cross-modal retrieval tasks, such as video-text retrieval (Pan et al., 2016), audio-image retrieval (Aytar et al., 2017), etc., they all have in common that query data and collections are processed with independent approaches.

In summary, visual retrieval is an essential field within computer vision, both because its utility to manage visual collections in the era of the information explosion and because its multiple modalities and applications. Considering the three main modalities within visual retrieval detailed above, this dissertation focuses on how to find visual content from collections by using different kinds of data queries. First, in symmetric visual retrieval, we study the specific case of image-to-image retrieval, which is also known as content-based image retrieval (CBIR). Second, in asymmetric visual retrieval, we consider applications in image-to-video retrieval. Finally, in cross-modal retrieval, we focus on text-image retrieval.

1.2 Context

Visual retrieval has been a field in expansion since the early 1990s. At first, visual retrieval systems were based on textual tags (Tamura and Yokoya, 1984; Chang and Hsu, 1992), where images were manually annotated with semantic concepts. At search time, a textual query was used to find all the images associated with the tag, without considering the actual visual content of the pictures. In these

kinds of systems, images were considered simply as entities stored in a database and they were not involved in the retrieval process.

With the explosion of visual information, visual retrieval based on manual labels became impractical. The manual annotation of images with textual tags was expensive and imprecise, as different annotators could use different tags for the same image. Therefore, a more scalable and robust alternative was necessary and content-based image retrieval (CBIR) appeared as a field within computer vision.

The term CBIR was firstly introduced in Kato, 1992. Its aim is to use the visual content of images in the retrieval process. Relevant elements in a visual collection are retrieved according to their similarity to a query input. CBIR systems rely on two main processes: feature extraction and similarity estimation. Whereas feature extraction algorithms represent the visual content of images in a compact and robust descriptor, similarity estimation functions measure the visual similarity between a query and each element in the collection. These two processes are common in all the visual retrieval tasks, although some variations are applied depending on the type of visual search problem, see Figure 1.2.

There have been many attempts to produce robust feature extraction algorithms for visual retrieval, from human-engineered local features such as SIFT (Lowe, 2004) or SURF (Bay et al., 2006) and global aggregation methods (Sivic and Zisserman, 2003; Perronnin et al., 2010), to deep feature representations (Babenko et al., 2014; Tolias et al., 2016). As shown in Figure 1.2, symmetric visual retrieval techniques apply the same feature extraction algorithms to both queries and database content. On the other hand, asymmetric visual retrieval methods extract different visual features from videos and images. Whereas videos are spatio-temporal representations, images contain only spatial information. Cross-modal retrieval approaches also use different pipelines to process visual, textual or audio data.

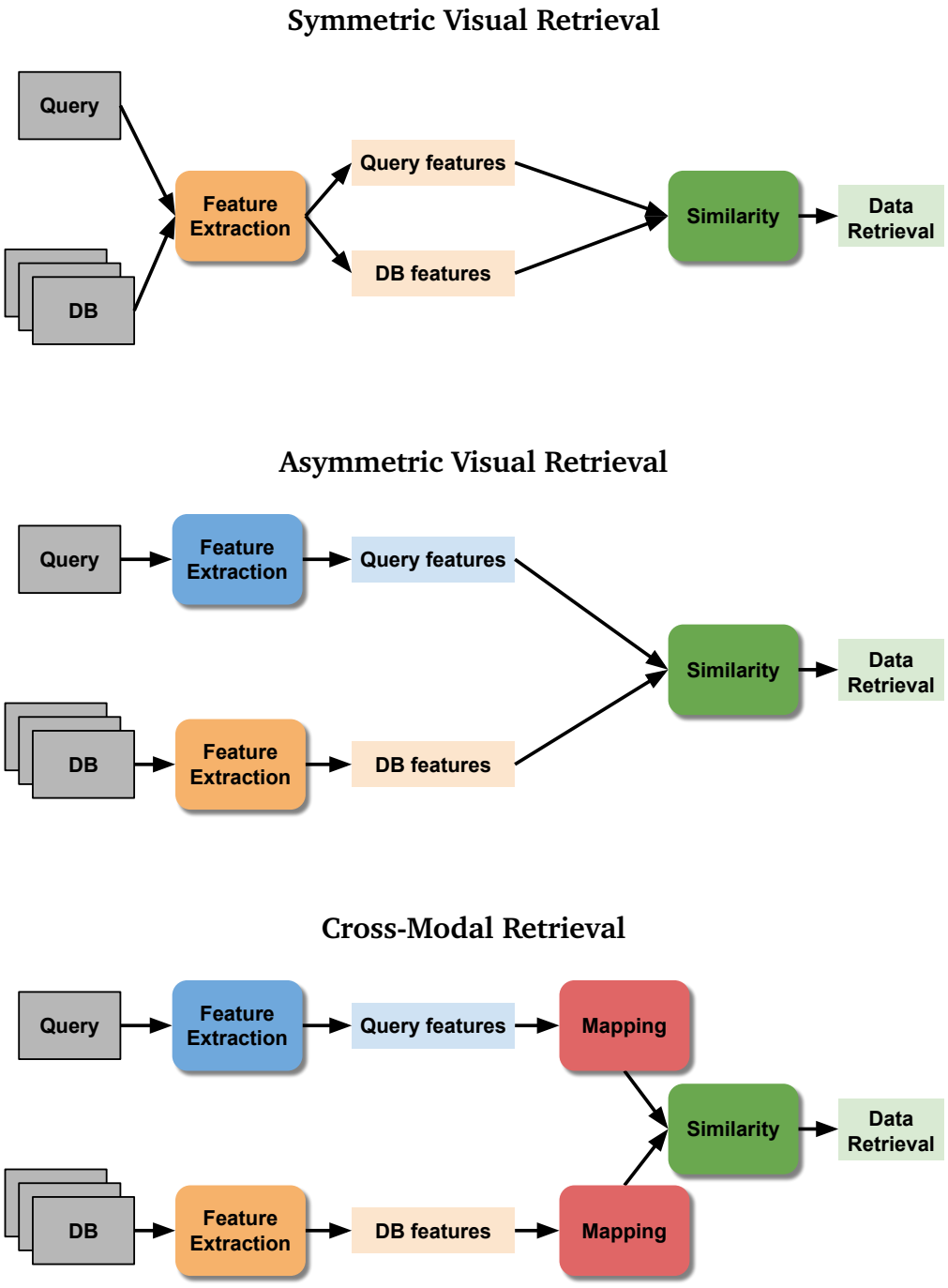


Fig. 1.2: Basic algorithmic components of visual retrieval systems.

Similarity in visual retrieval is usually estimated with a metric distances (e.g. Euclidean distance) between a pair of feature representations. In symmetric and asymmetric retrieval, this distance can be applied directly to the feature representations, as query and dataset content both belong to the same visual space. In cross-modal retrieval, however, features from queries and collections are described in two different spaces (e.g. visual and textual). A prior feature mapping process to transform features from their original spaces to a common latent space is needed. Once features are in a common space, query elements and collection are compared using standard similarity functions.

1.3 Contributions

The aim of this dissertation is to explore visual retrieval in all of its three modalities by considering the specific challenges of each modality and proposing solutions accordingly. In particular, this thesis addresses the following research questions:

Q1. In symmetric visual retrieval, is it possible to learn a non-metric function to estimate visual similarity between two or more visual samples that does not suffer from the rigid restrictions and limitations of metrics distances?

Q2. In asymmetric visual retrieval, how can we aggregate temporal information to compare spatio-temporal against spatial content? Is it possible to project images and videos in a common visual space and compare them in terms of visual similarity?

Q3. In cross-modal retrieval, is it possible to estimate the semantic similarity between texts and images in very specific context-heavy domains such as art? Is it possible to learn semantic art understanding through text-image retrieval?

The work reported in this dissertation investigates each of these research questions and contributes to the advancement of the state of the art in multi-modal visual retrieval as follows:

C1. To address Q1, we introduce the use of convolutional neural networks to estimate a non-metric visual similarity between a pair of images in symmetric visual retrieval. We show that by using similarity networks, image retrieval performance is improved considerably on top of high quality image representations;

C2. In order to study Q2, we create an asymmetric visual retrieval dataset with up to 80 hours of video and more than 25,000 query images to provide a common and public benchmark for large-scale image-to-video retrieval algorithms, being the image-to-video retrieval dataset with the largest number of query images introduced so far;

C3. To address Q2, we first propose a method to aggregate local binary features over time for efficient asymmetric image-to-video retrieval. This method compresses temporal information in videos by a factor of 42.5 while maintaining accuracy at similar levels as linear search;

C4. Alternatively, we propose the use of deep learning spatio-temporal features for asymmetric visual retrieval to compress video segments into compact 512-dimensional vectors. Our compact spatio-temporal features outperform previous methods in standard image-to-video retrieval datasets;

C5. As a direct application of Q2, we introduce a video item retrieval framework for finding clothes in videos based on asymmetric image-to-video retrieval;

C6. To address Q3, we create SemArt, a cross-modal retrieval dataset for the specific domain of semantic art understanding. SemArt is the first dataset of fine-art painting images associated with attributes and text descriptions;

C7. Additionally, we propose an evaluation protocol for cross-modal retrieval in semantic art understanding with which future research can be benchmarked under a common and public framework;

C8. We address Q3 by implementing a number of cross-modal retrieval models to estimate the semantic similarity between images and texts in a joint semantic space in the specific domain of art. Our best model obtains results close to human performance in this high-level recognition task.

1.4 Thesis Outline

This thesis is structured into five parts. The first part is the introduction, in which the motivation, contributions and background techniques are detailed. The second part describes the proposed techniques for symmetric visual retrieval. The third part contains our contribution in asymmetric visual retrieval. The fourth part addresses the cross-modal retrieval problem. Finally, in the last part, conclusions and final remarks are presented.

Part I: Introduction and Background

CHAPTER 2 introduces the technical background and the fundamental visual retrieval techniques that are applied in the following parts of the thesis, such as feature extraction, similarity estimation and evaluation methods.

Part II: Symmetric Visual Retrieval

CHAPTER 3 addresses symmetric visual retrieval and CBIR by exploring deep learning techniques for similarity estimation. It first presents the related work in the field and then proposes a model to push image retrieval performance by using convolutional neural networks to estimate visual similarity and replace standard metric functions (Garcia and Vogiatzis, 2019).

Part III: Asymmetric Visual Retrieval

CHAPTER 4 introduces asymmetric visual retrieval in general, and image-to-video retrieval in particular. It reviews current literature and datasets, presents a collection of videos and query images for public and common benchmark of image-to-video retrieval systems and introduces a framework for applying image-to-video techniques to recognise and retrieve clothes shown in videos (Garcia and Vogiatzis, 2017; Garcia, 2018).

CHAPTER 5 presents an approach for asymmetric image-to-video retrieval based on the temporal local aggregation of binary features. In this chapter, the temporal redundancy in videos is exploited to reduce the amount of data to be processed (Garcia and Vogiatzis, 2016; Garcia and Vogiatzis, 2017).

CHAPTER 6 explores asymmetric temporal global aggregation by using deep learning techniques. In this chapter, global video representations are obtained via a spatio-temporal encoder based on a combination of convolutional neural networks and recurrent neural networks (Garcia and Vogiatzis, 2018a).

Part IV: Cross-Modal Retrieval

CHAPTER 7 studies cross-modal retrieval by applying text-image retrieval techniques for semantic art understanding. It introduces the current literature, presents a dataset for semantic art understanding as well as an evaluation protocol based on text-image retrieval, and proposes a number of models for semantic art understanding (Garcia and Vogiatzis, 2018b).

Part V: Conclusions and Final Remarks

CHAPTER 8 summarizes the work presented in this dissertation, introduces the conclusions and highlights the future lines of research within multi-modal visual retrieval.

2 Background

In multi-modal visual retrieval, an input query is used to find relevant elements within a visual collection. Elements in the dataset and query inputs are represented by feature vectors, which describe the content of each element in a compact way. Feature vectors are then compared using similarity functions, so that elements within the collection can be ranked and retrieved according to a similarity score (see Figure 1.2).

In this chapter, we introduce the fundamental techniques involved in visual retrieval systems. First, we describe some of the most common feature extraction approaches to map the visual information from image pixels to vector representations (Section 2.1). Then, we present visual similarity techniques to estimate the similarity between a pair of samples (Section 2.2). Finally, we detail evaluation metrics for visual retrieval (Section 2.3). Additionally, specific methods to address the requirements of each visual retrieval modality addressed in this dissertation (i.e. symmetric visual retrieval, asymmetric visual retrieval and cross-modal retrieval) are reviewed at the beginning of each relevant part.

2.1 Feature Extraction

This section is an introduction to general feature extraction techniques for visual retrieval. Feature extraction methods for specific visual retrieval tasks are reviewed in each relevant part of this dissertation. For example, CBIR techniques are reviewed in Chapter 3, spatio-temporal features for asymmetric retrieval are summarized in Chapter 4 and visual and textual features for text-image cross-modal retrieval are described in Chapter 7.

Feature extraction is the process during which the salient visual information in images is represented into compact and stable vectors, known as features, descriptors or image representations. A desirable feature extraction method should describe salient visual information in a unique and invariant representation, so that visual patterns can be easily identified under different conditions, i.e. visual features should be reasonably invariant to scaling, rotation and illumination changes.

Feature extraction methods can be separated into three different types. Methods based on *local features* (Section 2.1.1) detect the interesting regions of the image and describe the local information of each of these regions into visual vectors using human-engineered techniques. For a more compact representation, *global features* (Section 2.1.2) aggregate local visual information into a single image representation. Recently, *deep features* (Section 2.1.3) based on convolutional neural networks were introduced as robust feature extraction methods for visual retrieval.

2.1.1 Local Features

Methods based on local features identify relevant patches of the image and represent the local visual content of these patches into descriptor vectors. The relevance of each region depends on its visual content. For example, let us consider the image from Figure 2.1 and its three extracted patches. The patches with strong edges, as the ones located in the mountain, contain distinctive information, whereas the textureless patch from the sky is the least representative region.

In visual retrieval, local features are used to identify similarities between images. Relevant patches are compared against other relevant patches to find common patterns within different images. A simple approach to compare patches' content is to measure the average Euclidean distance between their pixels. However,

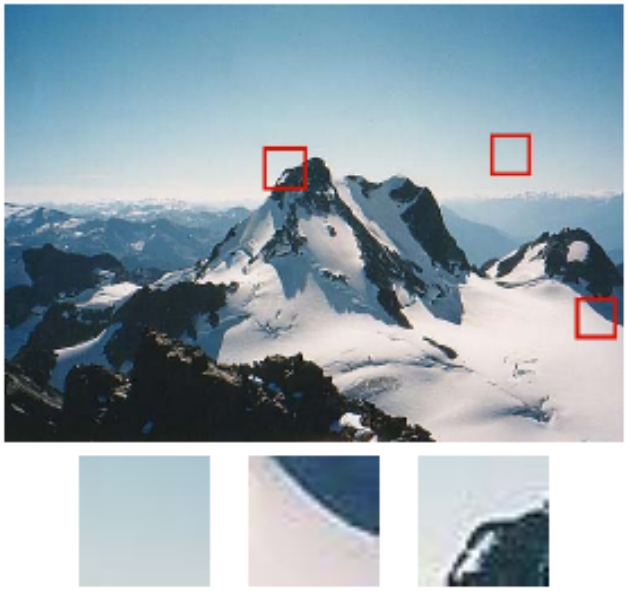


Fig. 2.1: Image with extracted feature patches, some of them more distinctive than others (image from Szeliski, 2010).

pixel intensity is very sensitive to noise and illumination changes. For a more robust comparison, local features methods rely on human-engineered algorithms to describe the visual content of relevant patches in stable and invariant vector descriptors.

There are many different local features extraction methods, some of the most well-known being SIFT (Lowe, 2004) and SURF (Bay et al., 2006). SIFT features are obtained using histograms of oriented gradients, which makes them robust to illumination and scaling variations. Despite of being widely used during many years in many computer vision applications, they are considerably slow to compute (Wu et al., 2013a). SURF features, which are also based on gradient orientations, speed up the computation time by using integral images (Crow, 1984). A different approach to obtain local features is the ones based on binary strings. BRIEF (Calonder et al., 2010) describes the visual content of each region by using a binary vector that encodes intensity comparisons between pairs of pixels. Binary features are faster to compute and compare than SIFT and SURF features, and as shown in Chapter 5, more stable over time. A complete review

on local features can be found in Mikolajczyk and Schmid, 2005 and (Miksik and Mikolajczyk, 2012).

2.1.2 Image Global Features

As a single image may contain hundreds of interesting regions, local features extraction methods do not scale well with large datasets of images. To reduce memory requirements and simplify the search process, global features that aggregate multiple local features into a single image representation, such as bag-of-words (BOW) or Fisher Vectors (FV), were introduced.

Inspired by text retrieval techniques, BOW (Sivic and Zisserman, 2003) learns a visual vocabulary by quantizing local features into a set of visual words using k-means (MacQueen, 1967). Images are described by the frequencies of appearance of each visual word, i.e. for each image, local features are extracted, assigned to their closest visual word and used to build a histogram of word frequencies. The dimensionality of the BOW descriptor is the number of visual words, which is usually relatively large.

To reduce vector dimensionality, FV (Perronnin et al., 2010) characterizes local features by their deviation from a Gaussian Mixture Model (GMM) distribution. In FV, a smaller visual vocabulary is built using a GMM. Then, the partial derivatives of the quantized features with respect to the parameters of the model are computed and concatenated into the visual descriptor. Alternatively, the Vector of Locally Aggregated Descriptors (VLAD, Jégou et al., 2010) accumulates the differences between the local features associated to a visual word and its centroid to build the visual descriptor.

In general, global aggregation methods simplify the search process in visual retrieval tasks by aggregating a set of human-engineered local features into a single global vector. However, the most advanced techniques rely on deep

learning methods to automatically find the important regions of an image and aggregate their visual information into an image representation.

2.1.3 Image Deep Features

Deep features are image representations obtained from Convolutional Neural Networks (CNNs, Fukushima and Miyake, 1982). Although CNNs were presented by the first time in the early 1980s, it was not until 2012, with the introduction of the AlexNet network (Krizhevsky et al., 2012) in the ImageNet challenge (ILSVRC, Deng et al., 2009), when their popularity expanded within the CV community.

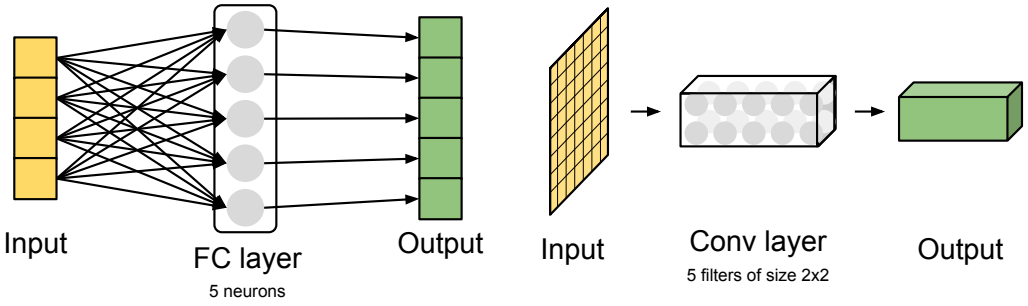
CNNs consist on a set of layers stacked on top of each other that learn non-linear functions by using relevant training data. The learning is performed by computing the error between the output of the CNN and the expected value, and backpropagating it through the parameters of the architecture. The representations obtained from these methods are very powerful, as they are trained to solve a specific task by providing thousands of examples.

In CNN architectures there are usually an input layer, which processes the input image, an output layer, which returns the final results, and several hidden layers. Each hidden layer takes the output from the previous layer, applies some transformation and forwards the new data to the following layer.

The basic computation unit in a CNN is the neuron, which takes a d-dimensional input vector $\mathbf{x} \in \mathbb{R}^d$, applies a linear transformation with weights $\mathbf{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$, and returns the output $s \in \mathbb{R}$ as:

$$s = \sum_{1 \leq k \leq d} w_k x_k + b \tag{2.1}$$

where w_k and x_k are the k-th element in $\mathbf{w}$ and $\mathbf{x}$, respectively.



(a) (b)

Fig. 2.2: Difference between fully connected (a) and convolutional (b) layers.

Neurons are commonly arranged in layers, so that a layer with multiple neurons returns a multi-dimensional output. Let us consider $\mathbf{x}^i$ and $\mathbf{s}^i$ as the input and output of the i-th layer, respectively. Typical layers that are found in modern CNNs architectures are:

- **Fully Connected layers**. A fully connected layer is a group of neurons grouped together in which each neuron is connected to all the dimensions of the input vector (Figure 2.2a). The output of a fully connected layer is computed as a matrix multiplication between the input and the trainable weights of the neurons, $\mathbf{W}^i$, with a trainable bias offset, $\mathbf{b}^i$:

$$\mathbf{s}^i = \mathbf{W}^i\mathbf{x}^i + \mathbf{b}^i \tag{2.2}$$

  The dimensionality of the output is equal to the number of neurons in ech layer.

- **Convolutional layers**. In convolutional layers, neurons are arranged in a three dimensional volume (Figure 2.2b). The depth of the volume is associated with the number of filters in the layer. Each 2-dimensional filter is slid across the width and height of the layer input with a pre-defined stride, to compute a local response at each location. By gathering all the local responses of all the filers, a three dimensional volume is obtained as

output. The width and height of the output depend on the input and the filer size, whereas the depth is equal to the number of filters.

- **Activation layers**. Activation layers are non-linear functions that are usually stacked after each fully connected or convolutional layer. Activation layers provide the network the ability to learn non-linear functions. Any non-linear function can be used as activation layer, for example:

$$\text{ReLU}: \qquad s_k^i = max(0, x_k^i) \tag{2.3}$$

$$\tanh: \qquad s_k^i = \frac{e^{x_k^i} - e^{-x_k^i}}{e^{x_k^i} + e^{-x_k^i}} \tag{2.4}$$

- **Pooling layers**. Pooling layers reduce the dimensionality of the input by pooling the values of a local region into a single number. The most common pooling layers are max-pooling, which takes the maximum value within each region, sum-pooling, which computes the sum, and mean-pooling which uses the average. An example of the different pooling strategies is shown in 2.3. The dimensionality of the region to be pooled and the stride are hyperparameters of the network.

- **Recurrent layers**. Recurrent layers are used with temporal sequences (e.g. word sentences or video frames) to create Recurrent Neural Networks. Recurrent layers consist on several states in which each state receives as input an element from the temporal input sequence (e.g. a word from a text sentence or a frame from a video sequence). The output at each state is computed by considering the input's state as well as the output from the previous state of the sequence.

These basic layer, along with regularization layers such as Dropout (Srivastava et al., 2014) or Batch Normalization (Ioffe and Szegedy, 2015), in a specific layout form the CNN model.

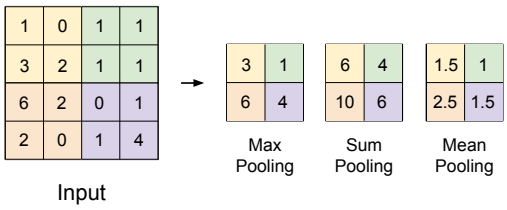


Fig. 2.3: Different pooling strategies on a 2D input map. In this case, the pooling region size is 2×2 and the stride is 2.

With respect to visual retrieval, deep features are commonly extracted from an intermediate layer of a CNN architecture pretrained for image classification. Here, we introduce some of the most popular approaches. A more specific review about deep features in CBIR problems is presented in Chapter 3.

Neural Codes

The extraction of deep features from pre-trained CNNs for visual retrieval was firstly introduced in Babenko et al., 2014 and Razavian et al., 2014. In particular, neural codes (Babenko et al., 2014) are deep image representations extracted from a pre-trained AlexNet network (Krizhevsky et al., 2012) for image classification.

The AlexNet architecture consists on five convolutional layers, with ReLU activations and max-pooling layers, and three fully connected layers, as shown in Figure 2.4. Input images are resized to 224×224 pixels and neural codes are extracted from layers 5, 6 or 7. Resizing the input image to such a low resolution may be perjudical for visual retrieval, as it is more probable than important information such as texture may get lost.

Although the success of using networks trained for image classification in a retrieval task, neural codes were not able to outperform classic image retrieval techniques based on SIFT features. However, neural codes can be further improved by fine-tunning the network architecture on images that are related to the retrieval task of interest.

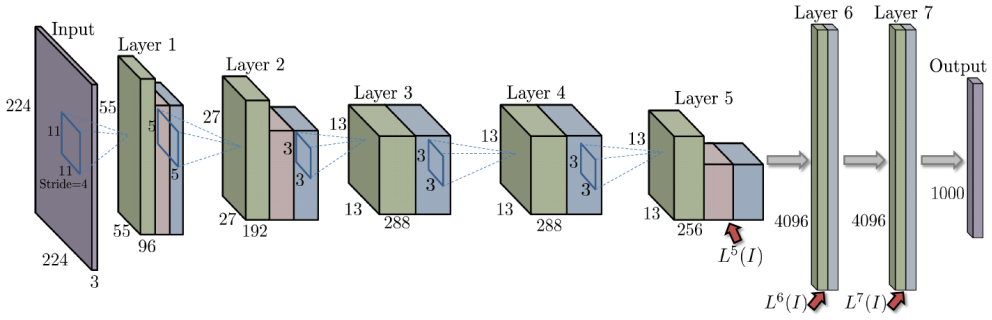


Fig. 2.4: Neural codes are extracted from layers 5, 6 or 7 of a pre-trained AlexNet network and used as image descriptors for image retrieval (image from Babenko et al., 2014).

RMAC

Regional Maximum Activation of Convolutions (RMAC, Tolias et al., 2016) is a deep feature extraction method obtained from the last convolutional layer of a CNN pre-trained for image classification. Originally, RMAC was computed from AlexNet (Krizhevsky et al., 2012) or VGG16 (Simonyan and Zisserman, 2015) networks, although recent work (Gordo et al., 2017) uses deeper ResNet (He et al., 2016) architectures.

When an input image is fed into the CNN model, the last convolutional layer outputs an activation volume with dimensions $w \times h \times d$, where d is the number of filters and w and h are the spatial width and height of the output volume, respectively. In RMAC, the response of the k-th filter is represented by $\mathbf{\Omega}^k$, a 2D tensor of size $w \times h$. If $\mathbf{\Omega}^k(\mathbf{p})$ is the response at a particular position $\mathbf{p}$, and $\mathbf{R}$ is a spatial region within the feature map, the regional feature vector $\mathbf{f}^{\mathbf{R}}$ is defined as:

$$\mathbf{f}^{\mathbf{R}} = [f_1^{\mathbf{R}} \dots f_k^{\mathbf{R}} \dots f_d^{\mathbf{R}}]^\top \tag{2.5}$$

where

$$f_k^{\mathbf{R}} = \max_{\mathbf{p} \in \mathbf{R}} \mathbf{\Omega}^k(\mathbf{p}) \tag{2.6}$$

Thus, $\mathbf{f_R}$ consists of the maximum activation of each filter (i.e. max-pooling) inside the spatial region $\mathbf{R}$.

To obtain the RMAC representation, several regional features are extracted at different multi-scale overlapping regions. Each of these regional vectors is independently post-processed with $\ell 2$-normalization, PCA-whitening and $\ell 2$-normalization. Regional vectors are summed up and $\ell 2$-normalized once again to obtain the final compact vector, whose dimensionality is d (i.e. the number of filters in the last convolutional layer) and it is independent of the size of the input image, its aspect ratio or the number of regions used. Commonly images are resized to 1024 pixels, which is a tradeoff between image resolution and memory requirements.

In this thesis, we use RMAC features as feature extraction in symmetric visual retrieval along with our proposed similarity computation method (Chapter 3), in asymmetric visual retrieval with our proposed spatio-temporal encoder for image-to-video retrieval (Chapter 6) and also in cross-modal retrieval for semantic art understanding (Chapter 7).

2.2 Visual Similarity

Visual similarity techniques estimate how alike a pair of samples are by evaluating a similarity function between their descriptors. Similarity estimation is a problem-dependent task, i.e. the similarity between a pair of elements depends not only on the elements but on the data within the collection, as shown in Figure 2.5. Similarity functions commonly used for visual retrieval are either data independent, such as standard metric distances (Section 2.2.1), or data dependent, such as metric learning algorithms (Section 2.2.2).

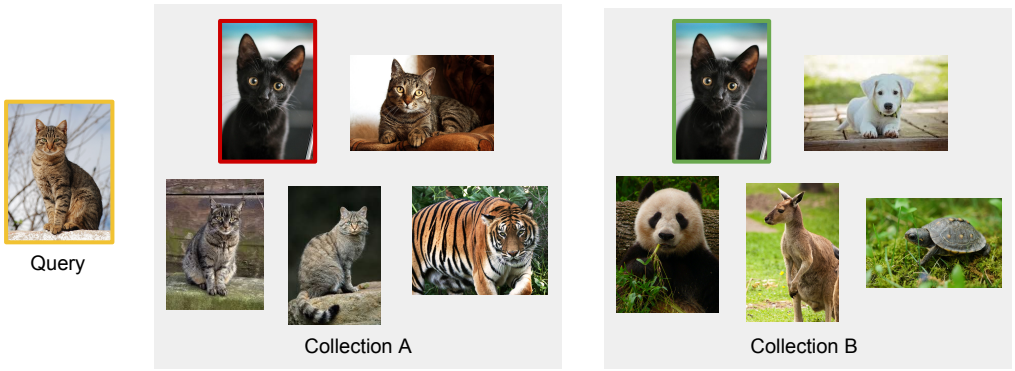


Fig. 2.5: Visual similarity is a context-dependent task: whereas the black cat may be considered dissimilar to the query image in collection A, the same image may be considered similar in collection B.

2.2.1 Standard Metric Distances

Consider $\mathbf{x}$ and $\mathbf{y}$ as two feature vectors, their similarity can be computed using a standard distance. The smaller the distance is, the more similar the two vectors are. Some of the most common distances used for visual similarity are:

- **L_1-distance**: $d_{L_1}(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$
- **Euclidean distance**: $d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$
- **Cosine distance**: $d_C(\mathbf{x}, \mathbf{y}) = 1 - \cos(\mathbf{x}, \mathbf{y})$, where $\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$ is the cosine similarity between $\mathbf{x}$ and $\mathbf{y}$.
- **Hamming distance** $d_H(\mathbf{x}, \mathbf{y}) = \sum_i [x_i \neq y_i]$, which counts the number of mismatches between $\mathbf{x}$ and $\mathbf{y}$. It is frequently used with binary vectors.

Similarity functions based on metric distances are easy and fast to implement. However, these methods do not consider the inner data distribution within the visual collection or the context of the task (Figure 2.5).

2.2.2 Metric Learning

Metric learning algorithms infer the similarity function directly from data. Metric learning uses pairs or triplets of data to learn the weights of a parametric distance, such as the Mahalanobis or the bilinear distance. There are many metric learning algorithms, with OASIS (Chechik et al., 2010) being one of the most popular ones. A more complete review on metric learning approaches can be found in Chapter 3.

OASIS learns the bilinear similarity between a pair of vectors, $\mathbf{x}$ and $\mathbf{y}$, as:

$$d_B(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{M} \mathbf{y} \tag{2.7}$$

where $\mathbf{M}$ are the parameters of the similarity function, which are initialized as $\mathbf{M} = \mathbf{I}$ ($\mathbf{I}$ being the identity matrix) and optimized online. At each time step t, the triplet $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, with $\mathbf{x}$ and $\mathbf{y}$ from the same class and $\mathbf{z}$ from a different class, is used to solve the optimization problem:

$$\begin{aligned} \mathbf{M}^t = \underset{\mathbf{M}, \xi \geq 0}{\arg\min} \quad & \frac{1}{2} \|\mathbf{M} - \mathbf{M}^{t-1}\|_{\mathcal{F}}^2 + C\xi \\ \text{s.t.} \quad & 1 - d_B(\mathbf{x}, \mathbf{y}) + d_B(\mathbf{x}, \mathbf{z}) \leq \xi \end{aligned} \tag{2.8}$$

where C is a trade-off parameter and $\| \cdot \|_{\mathcal{F}}$ is the Frobenius norm.

Although OASIS considers the distribution of the data within the task of interest to compute distances and thus, may be able to fit similarities better than standard metrics, its computation is based on linear metric learning. Linear functions are simpler and less prone to overfitting than non-linear functions, however, better results are expected with non-linear methods.

As an alternative to both standard metrics and linear metric learning, in Chapter 3, we propose to learn a non-metric similarity function using neural networks,

which outperform state-of-the-art techniques when compared in standard CBIR datasets.

2.3 Visual Retrieval Evaluation

Given an input query, i, visual retrieval systems return a list of relevant elements, ranked by their similarity to i. The quality of a retrieval system is commonly measured using information retrieval evaluation metrics, such as precision (p_i) and recall (r_i):

$$\mathrm{p}_i = \frac{|\{\text{relevant elements}\}_i \cap \{\text{retrieved elements}\}_i|}{|\{\text{retrieved elements}\}_i|} \tag{2.9}$$

$$\mathrm{r}_i = \frac{|\{\text{relevant elements}\}_i \cap \{\text{retrieved elements}\}_i|}{|\{\text{relevant elements}\}_i|} \tag{2.10}$$

where

- $\{\text{retrieved elements}\}_i$ is the list of returned elements
- $\{\text{relevant elements}\}_i$ is the list of elements that are relevant to i

These metrics measure the quality of the system based on the whole list of returned elements. In large-scale collections, however, users are commonly interested only in the top ranked positions of the list. Recall at K ($\mathrm{R@K}_i$) measures the number of relevant results on these top K positions:

$$\mathrm{R@K}_i = \frac{|\{\text{relevant elements in } K\}_i|}{K} \tag{2.11}$$

Recall an precision do not consider the order of the list. To measure the quality of a ranking list in terms of sorting, average precision (AP_i) is used:

$$\mathrm{AP}_i = \sum_{k=1}^{n} \mathrm{p}_i(k)\Delta\mathrm{r}_i(k) \tag{2.12}$$

where n is the size of the list, $\mathrm{p}_i(k)$ is the precision at position k, and $\Delta\mathrm{r}_i(k)$ is the change in recall from elements $k-1$ to k. When multiple queries are available, results are provided as the mean R@K and AP over all the queries.

2.3.1 Summary

This chapter provided a general overview of the common techniques used in visual retrieval problems. Visual retrieval models consist on two fundamental parts: feature extraction and similarity estimation. We presented some of the most popular algorithms to perform each of these parts.

On the one hand, feature extraction is the process in which image pixels are transformed into vectors that represent the visual information of the image. We introduced three types of visual extraction algorithms: local features, global features and deep features. Local features detect relevant patches in the image and describe each of the patches with a feature vector. As describing every single relevant patch in an image is not scalable to large datasets, global features aggregate local features into a single global representation. Both local and global features are based on human-engineered techniques. In contrast, deep features extraction methods use deep learning to both detect relevant regions in the image and describe them, alleviating the need for any hand-crafted algorithm. To compute deep features, images are input into a CNN architecture and the output of some the mid-layer representations is used to compute the feature vector, as in RMAC. Deep features can be obtained from pre-trained CNN, although better results are achieved when the CNN models are fine-tunned in a relevant dataset, as shown in Neural Codes.

On the other hand, similarity estimation measures how alike two images are by comparing their feature vectors. Commonly, visual similarity is estimated using standard metrics, such as Euclidean distance or cosine similarity. However, visual similarity is a problem-dependent task, and standard metrics do not consider the

inner data distribution of the problem. Metric learning algorithms can estimate visual similarity functions by fitting data more accurately, although a training process is required.

More specific reviews for each of the parts considered in this thesis can be found in the following chapters: Chapter 3 for CBIR, Chapter 4 for image-to-video retrieval and Chapter 7 for text-image retrieval.

Part II

Symmetric Visual Retrieval

3

Learning Non-Metric Visual Similarity for Image Retrieval

In this part, we study symmetric visual retrieval, the visual retrieval modality in which both the query image and the elements in the collection are from the same kind of visual data. As a symmetric task, queries and dataset elements are commonly processed using the same methods and techniques. These techniques are usually task-specific, depending on whether the visual data is based on images (i.e. image retrieval, Smeulders et al., 2000; Zheng et al., 2018) or videos (i.e. video retrieval, Geetha and Narayanan, 2008). In this dissertation, we address symmetric visual retrieval by studying image similarities in content-based image retrieval (CBIR).

Given a query image, CBIR systems rank pictures in a dataset according to how similar they are with respect to the query input. This is commonly performed in a two-step process, by first computing meaningful image representations that capture the most salient visual information from pixels (as introduced in Section 2.1), and then measuring accurate visual similarity between these image representations to rank images according to a similarity score (as introduced in Section 2.2).

Recently, several methods to represent visual information from raw pixels in images have been proposed, first by designing handcrafted features (Lowe, 2004; Bay et al., 2006), then by compacting these local features into a single global image descriptor (Sivic and Zisserman, 2003; Perronnin et al., 2010; Jégou et al., 2010) and more recently by extracting deep image representations from neural networks (Babenko et al., 2014; Gordo et al., 2017).

Once two images are described by feature vectors, visual similarity is commonly estimated by using a standard metric between their image descriptors. Although regular distance metrics are fast and easy to implement, non-linear functions are supposed to fit to the data distribution more accurately (Kulis, 2013), and thus, better results are expected. In this chapter, we hypothesize that learning a non-metric similarity function directly from data could push image retrieval performance on top of high quality image representations. Consequently, we propose a model to learn a non-metric visual similarity function based on deep learning techniques which improves image retrieval performance in up to 40% with respect to cosine similarity in standard datasets.

The chapter is structured as follows: related work in CBIR is reviewed in Section 3.1, the proposed model is detailed in Section 3.2 and the evaluation process is described in Section 3.3. Finally, conclusions are summarized in Section 3.4.

3.1 Related Work

In this section, we provide a deeper review in deep learning for image retrieval and expand the background content from Chapter 2 with specific related work in CBIR.

3.1.1 Feature Representation

In Chapter 2, we presented the three main approaches to obtain image representations for visual retrieval, i.e. local features, global features and deep features. With the latest advancements on deep learning, we showed that deep image retrieval, which uses activations from CNNs as image representations, rapidly become the state-of-the-art in CBIR.

Early work in deep image retrieval, such as Neural Codes (Babenko et al., 2014) among others (Razavian et al., 2014; Wan et al., 2014; Liu et al., 2015), proposed

to use representations from the last fully connected layers of pre-trained networks (AlexNet (Krizhevsky et al., 2012)) on ImageNet classification task (Deng et al., 2009) as deep image representations. However, with the introduction of deeper networks (e.g. GoogLeNet (Szegedy et al., 2015), VGG (Simonyan and Zisserman, 2015) or ResNets (He et al., 2016)), mid-layer representations from convolutional layers were shown to obtain better accuracy in retrieval problems (Babenko and Lempitsky, 2015; Yue-Hei Ng et al., 2015; Razavian et al., 2014; Xie et al., 2015).

As the output of a convolutional layer consists on a 3-dimensional activation volume, several methods have been proposed to aggregate the activation output into a compact vector, with RMAC (Tolias et al., 2016) being one of the most succesful ones. For example, Gong et al., 2014b and Yue-Hei Ng et al., 2015 proposed to aggregate activations from convolutional layers with VLAD (Jégou et al., 2010); Mohedano et al., 2016 encoded multiple deep representations into a BOW (Sivic and Zisserman, 2003); Babenko and Lempitsky, 2015 and Kalantidis et al., 2016 sum-pooled the activation maps; and Razavian et al., 2016 and Tolias et al., 2016 (i.e. RMAC) aggregated deep features by max-pooling them into a new vector.

Although deep image representations from networks pre-trained for image classification perform well in CBIR problems, techniques to push deep image retrieval results even further have been proposed, such as fine-tunning or visual attention. Fine-tunning pre-trained networks with similar data to the target retrieval task improves the performance considerably (Babenko et al., 2014; Gordo et al., 2016; Radenović et al., 2016; Salvador et al., 2016; Gordo et al., 2017), however, the fine-tunning process is expensive and time-consuming as it requires retraining all the layers of the model with thousands of training examples. Similarly, using attention models to automatically select the more meaningful features in every image has been shown to be beneficial (Jiménez et al., 2017; Noh et al., 2017), but it also requires a large number of training samples to learn the weights of

the attention layers. Alternatively, we propose an approach to improve image retrieval performance by learning the last layers of a similarity function on top of high-quality image representations, simplifying the fine-tunning process and improving results with respect to standard methods.

3.1.2 Visual Similarity

As shown in Chapter 2, visual similarity can be estimated using either standard metrics or similarity learning models. Some of the most popular similarity learning work, such as OASIS (Chechik et al., 2010) and MLR (McFee and Lanckriet, 2010), is based on linear metric learning by optimizing the weights of a linear transformation matrix. Although linear methods are easier to optimize and less prone to overfitting, nonlinear algorithms are expected to achieve higher accuracy by modeling the possible nonlinearities of data (Kulis, 2013).

Nonlinear similarity learning based on deep learning has been applied to many different visual contexts. In low-level image matching, CNNs have been trained to match pairs of patches for stereo matching (Zagoruyko and Komodakis, 2015; Luo et al., 2016) and optical flow (Dosovitskiy et al., 2015; Thewlis et al., 2016). In high-level image matching, deep learning techniques have been proposed to learn low-dimensional embedding spaces in face verification (Chopra et al., 2005), retrieval (Wu et al., 2013b; Wang et al., 2014), classification (Hoffer and Ailon, 2015; Qian et al., 2015; Oh Song et al., 2016) and product search (Bell and Bala, 2015), either by using siamese (Chopra et al., 2005) or triplet (Wang et al., 2014) architectures. In general, these methods rely on learning a mapping from image pixels to a low dimensional target space to compute the final similarity decision by using a standard metric.

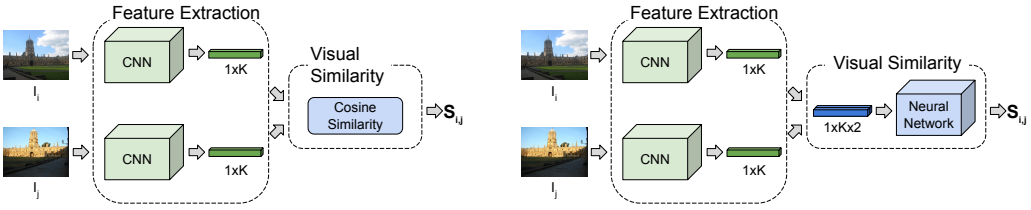
Instead of projecting the visual data into some linear space, that may or may not exist, we propose to learn the non-metric visual similarity score itself. In a similar way, Li et al., 2014 and Han et al., 2015 trained a CNN as a binary

classification problem to decide whether or not two input images are a match applied to pedestrian reindentification and patch matching, respectively. In CBIR, however, a regression score is required for ranking dataset images according to a specific similarity value. Inspired by the results of Wan et al., 2014, which showed that combining deep features with similarity learning techniques can be very beneficial in image retrieval systems, we propose to train a deep learning algorithm to learn non-metric similarities for image retrieval. We show that this provides up to a 40% improvement in performance in standard CBIR datasets with respect to standard methods.

3.2 Methodology

We propose a method to learn a non-metric visual similarity function from the visual data distribution. The main idea is shown in Figure 3.1. In standard CBIR, the similarity score between a pair of images is usually computed with a metric distance (Zheng et al., 2018). In contrast, we use a *visual similarity network* to estimate this score. As in deep image retrieval systems, we extract K-dimensional visual vectors from images by using a CNN and then, the similarity neural network outputs the similarity score between the pair of visual vectors.

We directly apply the output of the model as a similarity estimation to rank images accordingly. In this way, the similarity network can be seen as a replacement of the standard metric distance computation, overcoming the limitations of the rigid metric constrains and improving results on top of them. To precisely capture the different similarity degrees between images, we design a supervised regression learning framework. The proposed similarity network is end-to-end differentiable, which allows us to build an architecture for *real* end-to-end training: from the input image pixels to the final similarity score.



(a) (b)

Fig. 3.1: Standard deep image retrieval versus our model: (a) Standard deep image retrieval, in which a function based on metric distances is used to estimate the similarity score; (b) Our proposed system, in which the visual similarity network estimates the score by using a non-metric function.

3.2.1 Problem Formulation

Visual similarity measures how alike two images are. Formally, given a pair of images $\mathbf{I}^i$ and $\mathbf{I}^j$ in a collection of images ξ, we define $s_{i,j}$ as their similarity score. The higher $s_{i,j}$ is, the more similar $\mathbf{I}^i$ and $\mathbf{I}^j$ are. To compute $s_{i,j}$, images are represented by K-dimensional image representations, which are obtained by mapping image pixels into the feature space $\mathbb{R}^K$, as $\mathbf{x}^l = f(\mathbf{I}^l, w_f)$ with $\mathbf{I}^l \in \xi$, where $f(\cdot)$ is a non-linear image representation function and w_f its parameters. We propose to learn a visual similarity function, $g(\cdot)$, that maps a pair of image representations $\mathbf{x}^i$ and $\mathbf{x}^j$ into a visual score as:

$$s_{i,j} = g(\mathbf{x}^i, \mathbf{x}^j, w_g) = g(f(\mathbf{I}^i, w_f), f(\mathbf{I}^j, w_f), w_g)$$
$$s.t. \quad s_{i,j} > s_{i,k} \rightarrow \mathbf{I}^i, \mathbf{I}^j \text{ more similar than } \mathbf{I}^i, \mathbf{I}^k \quad (3.1)$$

with $\mathbf{I}^i, \mathbf{I}^j, \mathbf{I}^k \in \xi$ and w_g being the trainable parameters of the similarity function.

Visual similarity functions are commonly based on metric distance functions such as $g(\mathbf{x}^i, \mathbf{x}^j) = \frac{\mathbf{x}^i \cdot \mathbf{x}^j}{\|\mathbf{x}^i\| \|\mathbf{x}^j\|}$ or $g(\mathbf{x}^i, \mathbf{x}^j) = \|\mathbf{x}^i - \mathbf{x}^j\|$, i.e. cosine similarity and Euclidean distance, respectively. Metric distance functions, $d(\cdot)$, perform mathematical comparisons between pairs of objects in a collection Π, by satisfying the following axioms:

1. $d(\mathbf{a}, \mathbf{b}) \geq 0$ (non-negativity)

2. $d(\mathbf{a}, \mathbf{b}) = 0 \leftrightarrow \mathbf{a} = \mathbf{b}$ (identity)

3. $d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$ (symmetry)

4. $d(\mathbf{a}, \mathbf{b}) \leq d(\mathbf{a}, \mathbf{c}) + d(\mathbf{c}, \mathbf{b})$ (triangle inequality)

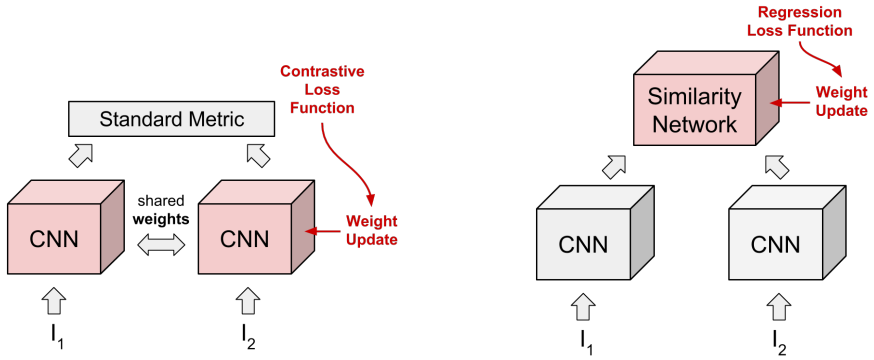
with $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \Pi$.

However, metric axioms are not always the best method to represent visual human perception (Gavet et al., 2014; Tan et al., 2006; Tversky and Gati, 1982). For example, non-negative and identity axioms are not required in visual perception as long as relative similarity distances are maintained. Symmetry axiom is not always true, as the human perception of similarity may be influenced by the order of appearance of the objects being compared. Finally, triangle inequality does not always correspond to visual human perception either. This can be easily understood when considering the images of a person, a horse and a centaur: although a centaur might be visually similar to both a person and a horse, the person and the horse are not similar to each other.

3.2.2 Similarity Network

To remove some of the difficulties with metric distance functions, we propose to learn the similarity function using neural networks. The network used to estimate the visual similarity, which we named similarity network, is composed of a set of fully connected layers, each one of them, except by the last one, followed by a rectified linear unit[1] (ReLU) non-linearity. The input of the network is a concatenated pair of image representations vectors, $\mathbf{x}^i$ and $\mathbf{x}^j$, which can be obtained using any standard technique, such as Babenko et al., 2014 or Tolias et al., 2016,

[1] $z = max(o, x)$



Siamese Network Similarity Network

Fig. 3.2: Similarity versus siamese networks. Siamese networks learn to map pixels into vector representations, whereas similarity networks learn a similarity function on top of the vector representations.

and the output is the similarity score, $s_{i,j}$. In that way, the similarity network learns the similarity function, $g(\cdot)$, from the image representation vectors.

At this point, we would like to emphasize that, as shown in Figure 3.2, the proposed similarity network is conceptually different to the siamese architecture in Chopra et al., 2005. Siamese networks use pairs of images to learn the feature extraction function, $f(\cdot)$, which maps image pixels images into vector representations. Then, similarity is computed with a metric distance based function, such as cosine similarity or Euclidean distance. In contrast, our approach learns the function $g(\cdot)$ on top of the image representations, replacing the standard metric distance computation.

3.2.3 Training Framework

We design a training framework to learn the weights of the similarity network as a supervised regression task. However, as providing similarity labels for every possible pair of training images is infeasible, we propose a training procedure in which the visual similarity is learned progressively using standard image classification annotations.

The model is trained to discriminate whether two images, $\mathbf{I}^i, \mathbf{I}^j$, are similar or dissimilar. Then, a similarity score, $s_{i,j}$, is assigned accordingly by improving a standard similarity function, sim$(\cdot)$. To optimize the weights, w_g, of the similarity function $g(\cdot)$ from Equation 3.1, the following regression loss function is computed between each training pair of image representations, $\mathbf{x}^i, \mathbf{x}^j$:

$$\text{Loss}(\mathbf{I}^i, \mathbf{I}^j) = |s_{i,j} - \ell_{i,j}(\text{sim}(\mathbf{x}^i, \mathbf{x}^j) + \Delta) - (1 - \ell_{i,j})(\text{sim}(\mathbf{x}^i, \mathbf{x}^j) - \Delta)| \quad (3.2)$$

where Δ is a margin parameter and $\ell_{i,j}$ is defined as:

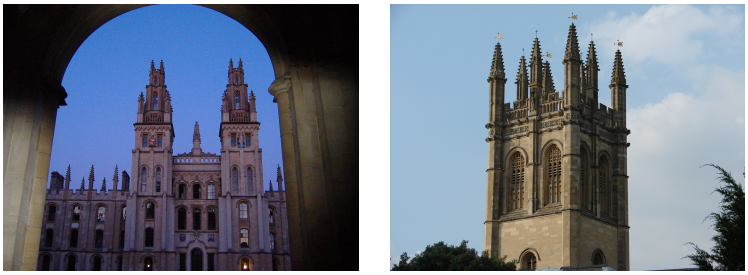
$$\ell_{i,j} = \begin{cases} 1 & \text{if } \mathbf{I}^i \text{ and } \mathbf{I}^j \text{ are similar} \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

In other words, the similarity network learns to increase the similarity score when two matching images are given and to decrease it when a pair of images is not a match. Similarity between pairs might be decided using different techniques, such as image classes, score based on local features or manual labelling, among others. Without loss of generality, we consider two images as similar when they belong to the same annotated class and as dissimilar when they belong to different classes.

Choosing appropriate examples when using pairs or triplets of samples in the training process is crucial for a successful training (Gordo et al., 2016; Radenović et al., 2016; Movshovitz-Attias et al., 2017). This is because if the network is only trained by using *easy pairs* (e.g. a car and a dog), it will not be able to discriminate between *difficult pairs* (e.g. a car and a van).

We design our training framework by emphasizing the training of difficult examples. First, we randomly select an even number of similar and dissimilar pairs of training samples and train the similarity network until convergence. We then choose a new random set of images and compute the similarity score between

Similar images



Dissimilar images

Fig. 3.3: Examples of difficult pairs, i.e. dissimilar images in which the network score is lower than the metric distance (top) and similar images in which the network score is higher than the metric distance (bottom).

all possible pairs by using the converged network. Pairs in which the network output is worse than the metric distance function are selected as difficult pairs for retraining, where a worse score means a score that is lower in the case of a match and higher in the case of a non-match. Finally, the difficult pairs are added to the training process and the network is trained until convergence one more time. Examples of difficult image pairs are shown in Figure 3.3.

3.3 Evaluation

We evaluate and compare the use of similarity networks against other similarity functions using standard CBIR datasets.

3.3.1 Datasets

Evaluation Datasets

Our approach is evaluated on three standard image retrieval datasets: OXFORD5K (Philbin et al., 2007), PARIS6K (Philbin et al., 2008) and LAND5K, a validation subset of LANDMARKS dataset (Babenko et al., 2014). OXFORD5K consists on 5,062 images of 11 different Oxford landmarks and 55 query images. PARIS6K contains 6,412 images of 11 different Paris landmarks and 55 queries. LAND5K consists of 4,915 images from 529 classes with a random selection of 45 images to be used as queries.

For experiments on larger datasets, we also use the standard large-scale versions OXFORD105K and PARIS106K, by including 100,000 distractor images (Philbin et al., 2007). In both OXFORD5K and PARIS6K collections, query images are cropped according to the region of interest. Evaluation is performed by computing the mean Average Precision (mAP). For LAND5K results are also reported as mAP, by considering an image to be relevant to a query when they both belong to the same class.

Training Dataset

For training, we use the cleaned version of the LANDMARKS dataset (Babenko et al., 2014) from (Gordo et al., 2016). Due to broken URLs, we could only download 33,119 images for training and 4,915 for validation. To ensure visual similarity is

learnt from relevant data, we create two more training sets, named LANDMARKS-EXTRA500 and LANDMARKS-EXTRA, by randomly adding about 500 and 2000 images from OXFORD5K and PARIS6K classes to LANDMARKS, respectively. Query images are not added in any case and they remain unseen by the system.

3.3.2 Implementation Details

Image Representation

Unless otherwise stated, we use RMAC (Tolias et al., 2016) as image representation method. VGG16 network (Simonyan and Zisserman, 2015) is used off-the-shelf without any retraining or fine-tunning. Images are re-scaled up to 1024 pixels, keeping their original aspect ratio. RMAC features are sensitive to the PCA matrices used for normalization. For consistency, we use the PCA whitening matrices trained on PARIS6K on all the datasets, instead of using different matrices in each evaluation collection. This leads to slightly different results than the ones provided in the original paper.

Similarity Training

We use cosine similarity as the similarity function, $\text{sim}(\mathbf{x}^i, \mathbf{x}^j) = \frac{\mathbf{x}^i \cdot \mathbf{x}^j}{\|\mathbf{x}^i\| \|\mathbf{x}^j\|}$ in Equation 3.2. For a faster convergence, we warm-up the weights of the similarity network by training it with random generated pairs of vectors and $\Delta = 0$. In this way, the network first learns to imitate the cosine similarity. Visual similarity is then trained using almost a million of image pairs. We experiment with several values of the margin parameter Δ, ranging from 0.2 to 0.8. The network is optimized using backpropagation and stochastic gradient descent with a learning rate of 0.001, a batch size of 100, a weight decay of 0.0005 and a momentum of 0.9.

Computational cost

Standard metric functions are relatively fast and computationally cheap. Our visual similarity network involves the use of millions of parameters that inevitable increase the computational cost. However, it is still feasible to compute the similarity score in a reasonable amount of time. In our experiments, training time is about 5 hours in a GeForce GTX 1080 GPU without weight warm-up and testing time for a pair of images is 1.25 ms on average. For reference, cosine similarity takes 0.35 ms to compute in a single CPU.

3.3.3 Results Analysis

Architecture Discussion

As shown in Table 3.1, we first experiment with four different architectures. We compare the performance of each configuration during the network warm-up (i.e. $\Delta = 0$), by using 22.5 million and 7.5 million pairs of randomly generated vectors for training and validation, respectively.

During the training warm-up, the network is intended to imitate the cosine similarity. We evaluate each architecture by computing the mean squared error, MSE, and the correlation coefficient, ρ, between the network output and the cosine similarity. Configuration C, which is the network with the largest number of parameters, achieves the best MSE and ρ results. However, considering a trade-off between performance and number of parameters of each architecture, we keep configuration B as our default architecture for the rest of the experiments.

Config	Architecture	Params	MSE	ρ
A	FC-1024, FC-1024, FC-1	2.1	$3.5 \cdot 10^{-4}$	0.909
B	FC-4096, FC-4096, FC-1	21	$1.9 \cdot 10^{-4}$	0.965
C	FC-8192, FC-8192, FC-1	76	$1.2 \cdot 10^{-4}$	0.974
D	FC-4096, FC-4096, FC-4096, FC-1	38	$1.9 \cdot 10^{-4}$	0.964

Tab. 3.1: Four similarity network architectures. Fully connected layers are denoted as (FC-{filters}). Number of parameters (Params) is given in millions.

Similarity Evaluation

We then study the benefits of using a non-metric similarity network for image retrieval by comparing it against several similarity methods. The similarity functions under evaluation are:

- **Cosine**: the similarity between a pair of vectors is computed with the cosine similarity: $\cos(\mathbf{x}^i, \mathbf{x}^j) = \frac{\mathbf{x}^i \cdot \mathbf{x}^j}{\|\mathbf{x}^i\| \|\mathbf{x}^j\|}$. No training is required.

- **OASIS**: OASIS algorithm (Chechik et al., 2010) is used to learn a linear function to map a pair of vectors into a similarity score. The training of the matrix transformation is performed in a supervised way by providing the class of each image.

- **Linear**: we learn an affine transformation matrix to map a pair of vectors into a similarity score by optimizing Equation 3.2 in a supervised way. Classes of images are provided during training. The margin Δ is set to 0.2.

- **SimNet**: the similarity function is learnt with our proposed similarity network by optimizing Equation 3.2 without difficult pairs refinement. Classes of images are provided during training and different margin Δ are tested, ranging from 0.2 to 0.8.

- **SimNet***: same as SimNet but with difficult pairs refinement.

	LANDMARKS			LANDMARKS-EXTRA500			LANDMARKS-EXTRA		
	OX5K	PA6K	LA5K	OX5K	PA6K	LA5K	OX5K	PA6K	LA5K
Cosine	**0.665**	0.638	0.564	0.665	0.638	0.564	0.665	0.638	0.564
OASIS	0.514	0.385	0.578	0.570	0.651	0.589	0.619	0.853	0.579
Linear (0.2)	0.598	**0.660**	0.508	0.611	0.632	0.514	0.602	0.581	0.502
SimNet (0.2)	0.658	0.460	0.669	0.717	0.654	0.671	0.718	0.757	0.668
SimNet* (0.2)	0.655	0.503	0.697	**0.719**	0.677	0.693	0.786	0.860	0.662
SimNet* (0.4)	0.637	0.504	0.737	0.703	0.701	0.745	0.794	0.878	0.706
SimNet* (0.6)	0.613	0.514	0.776	0.703	**0.716**	0.776	0.789	0.885	0.735
SimNet* (0.8)	0.600	0.511	**0.783**	0.685	0.710	**0.803**	**0.808**	**0.891**	**0.758**

Tab. 3.2: Comparison between different similarity functions. Δ value is set in brackets.

Results are summarized in Table 3.2. Trained similarity networks (SimNet, SimNet*) outperform trained linear methods (OASIS, Linear) in all but one evaluation datasets. As all Linear, SimNet and SimNet* are trained using the same supervised learning protocol and images, the results suggest that the improvement obtained with our method is not because of the supervision but because of the non-metric nature of the model.

When using LANDMARKS-EXTRA as training dataset, results are boosted with respect to the standard metric, achieving improvements ranging from 20% (OXFORD5K) to 40% (PAIRS6K). When using LANDMARKS-EXTRA500 dataset, our similarity networks also improve the mAP with respect to the cosine similarity in the three testing datasets. This indicates that visual similarity can be learnt even when using a reduced subset of the target image domain. However, visual similarity does not transfer well across domains when no images of the target domain are used during training, which is a well-known problem in metric learning systems (Kulis, 2013). In that case, cosine similarity is the best option over all the methods.

3.3.4 Image Representation Discussion

Next, we study the generalisation of our similarity networks when used on top of different feature extraction methods: the output of a VGG16 network Simonyan and Zisserman, 2015, the output of a ResNet50 network He et al., 2016, MAC Tolias et al., 2016, RMAC Tolias et al., 2016 and the model from Radenović et al., 2016. We compare the results of our networks, SimNet and SimNet*, against cosine similarity. Results are provided in Figure 3.4. Our similarity networks outperform cosine similarity in all the experiments, improving retrieval results when used on top of any standard feature extraction method. Moreover, performance is boosted when SimNet* is applied, specially in features with poor retrieval performance, such as ResNets.

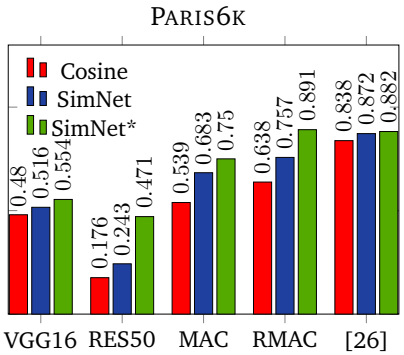


Fig. 3.4: Image Representation Discussion. mAP for different visual similarity techniques on top of different feature extraction methods.

Domain Adaptation

We further investigate the influence of the training dataset on the similarity score when the similarity network is transfered between different collections of images. As already noted in Table 3.2, visual similarity does not transfer well across domains and a subset of samples from the target dataset is required during training to learn a meaningful similarity function. This is mainly because similarity estimation is a problem-dependent task (Figure 2.5), as the similarity between a pair of elements depends on the data collection.

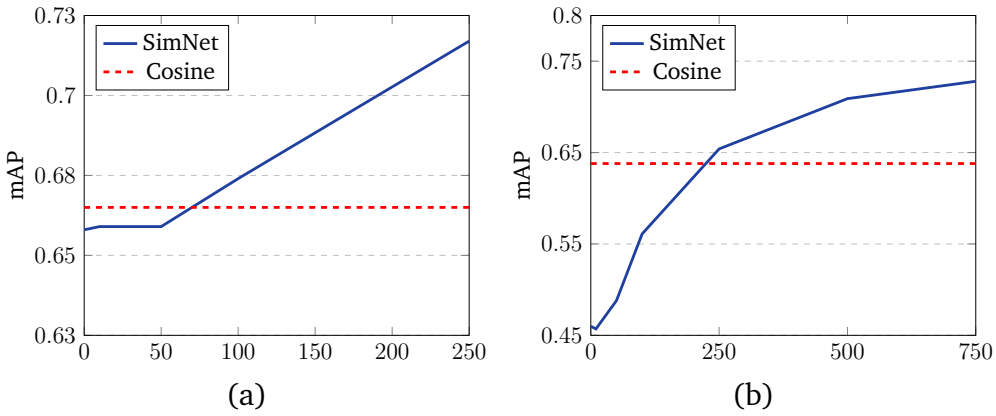


Fig. 3.5: Domain adaptation evaluation when using different number of target samples in the training set: (a) OXFORD5K; (b) PARIS6K.

To explore this effect, we evaluate the results when using different subsets of samples from the target collection in addition to the LANDMARKS dataset. Results are shown in Figure 3.5. There is a direct correlation between accuracy and the number of samples from the target dataset used during training. Indeed, in agreement with previous work in metric learning (Kulis, 2013), we observe that not considering samples from the target dataset at all might be harmful.

The similarity network, however, outperforms standard metric results even when a small number of samples from the target collection is used during training, i.e. only 100 images from OXFORD5K and 250 images from PARIS6K are required in OXFORD5K and PARIS6K datasets, respectively, which shows that the similarity network generalizes the similarity estimation from a small subset of samples.

3.3.5 End-to-End Training

So far, we have isolated the similarity computation part to verify that the improvement in the evaluation datasets compared to when using other similarity methods is, in fact, due to the similarity network. In this section, however, we explore a *real* end-to-end training architecture for image retrieval, which is depicted in Figure 3.6.

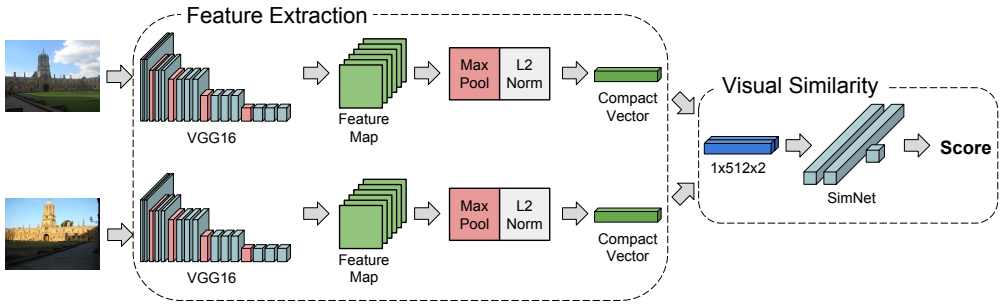


Fig. 3.6: End-to-End image retrieval model. MAC is used as feature extraction method and the similarity network (SimNet) as visual similarity function.

For the feature extraction part, we adopt MAC (Tolias et al., 2016) as a feature extraction technique, although any differentiable image representation method may be used. To obtain MAC vectors, images are fed into a VGG16 network (Simonyan and Zisserman, 2015). The output of the last convolutional layer is max-pooled and l2-normalized. For the visual similarity part, we use the similarity network with $\Delta = 0.2$. As the whole architecture is end-to-end differentiable, the weights are fine-tunned through backpropagation.

To train the end-to-end architecture, we first freeze the MAC computation weights and learn the similarity network parameters. Then, we unfreeze all the layers and fine-tune the model one last time. As all the layers have been already pre-trained, the final end-to-end fine-tunning is performed in only about 200,000 pairs of images from LANDARMARKS-EXTRA dataset for just 5,000 iterations.

Results are presented in Table 3.3. There is an improvement of up to 25% when using the similarity network with respect to the cosine similarity, as already seen in the previous section. When the architecture is trained end-to-end results are improved up to a 40%, since fine-tuning the entire architecture allows a better fit to the particular dataset.

Features	Similarity	OXFORD5K	PARIS6K	LAND5K
MAC	Cosine	0.481	0.539	0.494
MAC	*SimNet*	0.509	0.683	0.589
MAC	*SimNet*	0.555	0.710	0.685

Tab. 3.3: End-to-end architecture results. mAP when different parts of the image retrieval pipeline are trained. In *italic*, the modules that are fine-tunned.

3.3.6 Comparison with State of the Art

We compare our method against several state-of-the-art techniques. As standard practice, works are split into two groups: off-the-shelf and fine-tunned. Off-the-shelf are techniques that extract image representations by using pre-trained CNNs, whereas fine-tunned methods retrain the network parameters with a relevant dataset. For a fair comparison, we only consider methods that represent each image with a global vector, without query expansion or image re-ranking.

Off-the-shelf results are shown in Table 3.4 and fine-tunned results are presented in Table 3.5. When using off-the-shelf RMAC features, our SimNet* approach outperforms previous methods in every dataset. To compare against fine-tunned methods, we compute RMAC vectors using the fine-tunned version of VGG16 proposed in Radenović et al., 2016. Accuracy is boosted when our similarity network is used instead of the analogous cosine similarity method (Radenović et al., 2016). SimNet* achieves the best mAP precision in OXFORD5K dataset and comes second in OXFORD105K and PARIS106K after Gordo et al., 2017, which uses the more complex and higher-dimensional ResNet network (He et al., 2016) for image representation.

Method	Dim	Similarity	Ox5k	Ox105k	Pa6k	Pa106k
Babenko et al., 2014	512	L2	0.435	0.392	-	-
Razavian et al., 2014	4096	Averaged L2	0.322	-	0.495	-
Wan et al., 2014	4096	OASIS	0.466	-	0.867	-
Babenko and Lempitsky, 2015	256	Cosine	0.657	0.642	-	-
Yue-Hei Ng et al., 2015	128	L2	0.593	-	0.59	-
Kalantidis et al., 2016	512	L2	0.708	0.653	0.797	0.722
Mohedano et al., 2016	25k	Cosine	0.739	0.593	0.82	0.648
Salvador et al., 2016	512	Cosine	0.588	-	0.656	-
Tolias et al., 2016	512	Cosine	0.669	0.616	0.83	0.757
Jiménez et al., 2017	512	Cosine	0.712	0.672	0.805	0.733
Ours ($\Delta = 0.8$)	512	**SimNet***	**0.808**	**0.772**	**0.891**	**0.818**

Tab. 3.4: Comparison with state-of-the-art off-the-shelf methods. Dim corresponds to the dimensionality of the feature representation and Similarity is the similarity function.

3.4 Conclusions

In this chapter, we studied asymmetric visual retrieval through CBIR. In CBIR, a query image is used to rank images in a collection according to their visual similarity. Standard CBIR methods, extract feature representations from activations of CNN and compute image similarity between a pair of images by applying standard metrics, such as Euclidean distance or cosine similarity.

To overcome the limitations of metric distances, we presented a method for learning visual similarity directly from visual data. Instead of using a metric distance function, we proposed to train a neural network model to learn a similarity score between a pair of visual representations. Our method was able to capture visual similarity better than other techniques, with improvements of up to 40% in standard image retrieval datasets.

We also proposed a real end-to-end trainable architecture for image retrieval, as all the layers in the similarity network are differentiable. We showed that results

Method	Dim	Similarity	Ox5k	Ox105k	Pa6k	Pa106k
Babenko et al., 2014	512	L2	0.557	0.522	-	-
Gordo et al., 2016	512	Cosine	0.831	0.786	0.871	0.797
Wan et al., 2014	4096	OASIS	0.783	-	**0.947**	-
Radenović et al., 2016	512	Cosine	0.77	0.692	0.838	0.764
Salvador et al., 2016	512	Cosine	0.71	-	0.798	-
Gordo et al., 2017	2048	Cosine	0.861	**0.828**	0.945	**0.906**
Ours ($\Delta = 0.8$)	512	**SimNet***	**0.882**	0.821	0.882	0.829

Tab. 3.5: Comparison with state-of-the-art fine-tunned methods. Dim corresponds to the dimensionality of the feature representation and Similarity is the similarity function.

are considerably improved when a similarity network is used, as this allows us to get a better fit of the input data distribution.

In summary, the use of a similarity network can push performance in image retrieval systems on top of high-quality image representations, even without the need of fine-tunning the whole architecture. This, combined with other CBIR techniques such as visual attention, query expansions or image re-ranking, would create more accurate algorithms and improve image retrieval systems considerably.

Part III

Asymmetric Visual Retrieval

4 Techniques for image-to-video retrieval

In this part, we introduce asymmetric visual retrieval by studying image-to-video retrieval. In image-to-video retrieval, the aim is to find a specific frame or scene in a video collection according to a given query image. Whereas videos in the collection contain spatio-temporal information, queries are images with only spatial visual content. Thus, asymmetric techniques for extracting visual features are required.

This chapter reviews the current literature in image-to-video retrieval (Section 4.1), formulizes the problem (Section 4.2) and introduces a large-scale dataset for benchmarking systems (Section 4.3). Additionally, we propose a framework for video content augmentation based on image-to-video retrieval (Section 4.4). Then, Chapter 5 and Chapter 6 describe our proposed models.

4.1 Related Work

Image-to-video retrieval started to attract attention since the early 2000s. With relatively small datasets, early work in the field (Sivic and Zisserman, 2003; Nister and Stewenius, 2006) processed frames as independent images by applying symmetric image retrieval techniques. For example, Sivic and Zisserman, 2003 indexed frames from two different movies by using BOW and the temporal structure in videos was used only to reject noisy keypoints. Similarly, Nister and Stewenius, 2006 used vocabulary trees to index BOW features from each video frame independently, without considering any temporal redundancy.

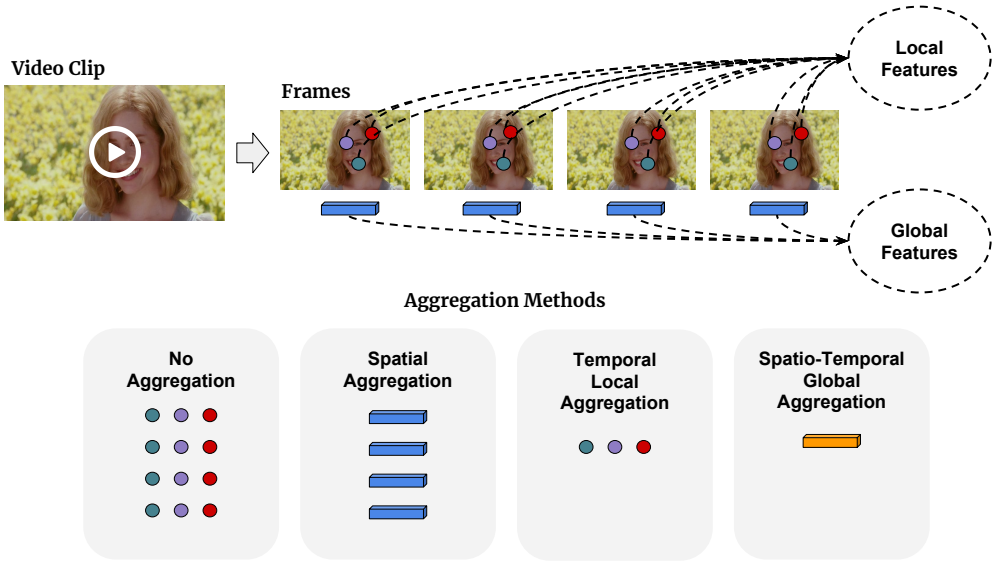


Fig. 4.1: Types of aggregation techniques for asymmetric visual retrieval. In *No Aggregation*, all the local features in all the frames are indexed. In *Spatial Aggregation*, a global visual feature per frame is indexed. These two methods do not take advantage of the temporal structure of videos. In *Temporal Local Aggregation*, the number of local features is reduced by identifying recurrent features along time. In *Spatio-Temporal Global Aggregation*, all the visual information in a video clip is compacted into a single vector representation.

As the number of frames in a video collection scales very fast with the size of the dataset, processing frames as independent images in large-scale datasets is impractical. To reduce the amount of data to be processed, Chen et al., 2010 indexed SURF features (Bay et al., 2006) extracted from specific keyframes in a vocabulary tree. Keyframes were sampled uniformly at a specific frame per second (FPS) rate from the video collection. However, sampling frames uniformly from videos may lead to suboptimal performance, as using a low FPS rate might discard important visual information from the collection, whereas using a high FPS rate may end up indexing unnecessary data.

To index frames more efficiently in large-scale image-to-video retrieval, the temporal structure of videos needs to be exploited. As consecutive video frames are usually highly correlated and share strong similarities, the visual information in similar looking frames can be compressed by aggregating their visual features into more compact representations. Methods for aggregating visual features along

time are classified in two categories: temporal local aggregation methods, in which temporal information is compacted by identifying recurrent local features (Section 4.1.1), and spatio-temporal global aggregation methods, in which the spatio-temporal information in multiple frames is aggregated into a single global representation (Section 4.1.2). The different types of aggregation techniques are shown in Figure 4.1.

Note that image-to-video retrieval uses a query image to find viusally similar scenes. A related but different problem is video instance search, in which the aim is to find all the frames where a query instance, usually an object, a person or a location, appears under different viewpoints (Sivic et al., 2006; Over et al., 2011; Meng et al., 2016). An example of this is the Instance Search task on the TRECVID challenge (Over et al., 2011), which uses up to four images from the same instance to find all the frames where the query appears. The main difference with the problem considered here is that video instance search performs retrieval at the object level, whereas image-to-video retrieval finds frames according to their scene similarity.

4.1.1 Temporal Local Aggregation Methods

Temporal local aggregation methods (Anjulan and Canagarajah, 2007; Araujo et al., 2014) identify recurrent local features (Section 2.1.1) in a video segment and compacts them into a reduced set of features. For example, Anjulan and Canagarajah, 2007 proposed to extract SIFT (Lowe, 2004) features from each frame, track them along time and average visual features within the same track to get a single vector per track. Similarly, Araujo et al., 2014 explored tracking SIFT features along time and proposed different methods to aggregate features within the same track, including averaging, keeping just one or computing the minimum distance.

At query time, temporal local aggregation methods match local features from the query image against the aggregated features from the collection of videos. This process involves performing as many searches as the number of features extracted from the query image. As each image may contain a few hundreds of local features, conducting multiple searches might be time-consuming in large-scale datasets. To ease the search process and reduce memory requirements, in Chapter 5 we propose a temporal local aggregation method based on binary features and fast indexing, which is able to reduce the memory storage by more than 40 times with respect to non-aggregation methods.

4.1.2 Spatio-Temporal Global Aggregation Methods

For a more compact search, spatio-temporal global aggregation methods (Zhu and Satoh, 2012; Araujo and Girod, 2017) compact the visual information of video segments into a single vector representation. For example, Zhu and Satoh, 2012 aggregated all the SIFT local features in a video clip into a single high-dimensional BOW vector; and Araujo and Girod, 2017 computed compact Fisher Vectors (Perronnin et al., 2010) per frame and aggregated several frames using super high-dimensional Bloom Filters (Bloom, 1970). In these approaches, query to video matching is performed by computing the distance between the query global image representation (see Section 2.1.2) and the video aggregated representation.

Although most of the image-to-video techniques proposed in the literature are based on the aggregation of SIFT features, some authors (Araujo and Girod, 2017; Wang et al., 2017) explored the aggregation of deep learning features (see Section 2.1.3). Araujo and Girod, 2017 compared pre-trained CNN architectures as feature extraction models against systems based on SIFT and Fisher Vectors, obtaining worse performance when using the deep features. Similarly, Wang et al., 2017 averaged MAC features (Tolias et al., 2016) from pre-trained CNN for image-to-video retrieval, obtaining poor accuracy rates in a standard dataset.

However, considering the outstanding results of deep learning in many retrieval tasks (Gordo et al., 2017; Wang et al., 2016), we hypothesise that the lack of success of deep learning models in image-to-video retrieval might be due to (1) the model architecture and, (2) the use of pre-trained features. To address these issues, in Chapter 6 we propose a deep learning model for learning spatio-temporal visual representations that outperforms previous global aggregation methods in image-to-video retrieval datasets.

4.2 Problem Formulation

Videos consist on a set of consecutive images or frames, which are grouped into a set of shots and scenes. Shots are defined as a set of consecutive frames that have been captured with the same camera without interruptions, whereas scenes are defined as a set of consecutive shots that share a common theme or topic, regardless of how similar these shots are between them.

Let us consider a set of videos $\mathbf{V} = \{\mathbf{V}_i\}_{i\in(0,..,N)}$ of size N, where each video $\mathbf{V}_i$ is at the same time a set of shots $\mathbf{V}_i = \{\mathbf{S}_{i,j}\}_{j\in(0,..,N_{V_i})}$ of size N_{V_i}, and where each shot $\mathbf{S}_{i,j}$ is at the same time a set of frames $\mathbf{S}_{i,j} = \{f_{i,j,k}\}_{k\in(0,..,N_{S_{i,j}})}$ of size $N_{S_{i,j}}$. Note that $f_{i,j,k}$ corresponds to the k-th frame of the j-th shot of the i-th video in the collection. Given a query image q, the goal is to find the most similar frame $\hat{f}$, belonging to the shot $\hat{\mathbf{S}}$, according to a specific metric distance d, such as:

$$\hat{f} = \underset{f_{i,j,k}\in\mathbf{V}}{\arg\min}\, d(\phi(q), \phi(f_{i,j,k})) \tag{4.1}$$

where $\phi(q)$ and $\phi(f_{i,j,k})$ are the visual representations of q and $f_{i,j,k}$, respectively.

For large-scale datasets, performing a search over all frames within the collection $\mathbf{V}$ is prohibitive. To alleviate the search, two techniques are used. Firstly, the

amount of visual features $\phi(f_{i,j,k})$ within a shot $\mathbf{S}_{i,j}$ is reduced by using a temporal aggregation method $\Theta(\cdot)$ over each shot:

$$\Theta(\mathbf{S}_{i,j}) = \Theta(\{\phi(f_{i,j,k})\}) \tag{4.2}$$

Secondly, taking advantage of the inner visual structure of videos, the search is performed in two stages. In the first stage, a shot-level search is conducted to find the shot of interest $\hat{\mathbf{S}}$:

$$\hat{\mathbf{S}} = \underset{\mathbf{S}_{i,j} \in \mathbf{V}}{\arg\min}\, d(\phi(q), \Theta(\mathbf{S}_{i,j})) \tag{4.3}$$

Finally, in the second stage, a frame-level search retrieves $\hat{f}$ from the frames contained in $\hat{S}$:

$$\hat{f} = \underset{f_{i,j,k} \in \hat{\mathbf{S}}}{\arg\min}\, d(\phi(q), \phi(f_{i,j,k})) \tag{4.4}$$

4.3 Datasets

To evaluate image-to-video retrieval approaches under a common public framework, there exist a number of public datasets.

4.3.1 Public Datasets

Early work in image-to-video retrieval conducted experiments and reported results on private collections of videos (Sivic and Zisserman, 2003; Nister and Stewenius, 2006). However, for a standard comparison between different models and to push performance in the field, publicly available datasets were introduced. Table 4.1 summarizes the existing public image-to-video retrieval collections:

Name	Lenght	Domain	#Queries	Queries
CNN2h (Araujo et al., 2014)	2h	Newscast	139	Websites, Camera
Stanford I2V (Araujo et al., 2015a)	1,079h	Newscast	229	Websites
VB (Araujo et al., 2016)	1,079h	Newscast	282	Camera
ClassX (Araujo et al., 2016)	408h	Lectures	258	Slides
MoviesDB (Ours)	80h	Movies	25,000	Camera

Tab. 4.1: Image-to-video retrieval datasets. For each dataset, we provide the total duration, the domain, the number of queries and the type of query images.

- **CNN2h** (Araujo et al., 2014): a collection with 2 hours of newscast videos and 139 query images. Query images are photos taken with an external camera as well as related pictures collected from websites.

- **Stanford I2V** (Araujo et al., 2015a): a collection of newcast videos and 229 query images collected from news websites. There exist a light version (SI2V-600k) and a large verion (SI2V-4M) with 160 and 1,079 hours of video, respectively.

- **VB** (Araujo et al., 2016): same video collection as in Stanford I2V with 282 queries captured with an external camera while videos are played in a screen. Some queries contain strong perspective distortion. As in Stanford I2V, there are a light version (VB-600k) and a large version (VB-4M).

- **ClassX** (Araujo et al., 2016): a collection of lecture videos with 258 query images. Query images are slides from the lectures. There are also a light version (ClassX-600k) with 169 hours of video and a large version (ClassX-1.5M) with 408 hours of video.

Although the collections listed above are in general large in terms of video duration, the amount of query images in each dataset is relatively small. Moreover, these collections contain videos from very specific domains (i.e. newscast and lectures). In the next section, we introduce the MoviesDB, a dataset of movies with a larger number of query images.

4.3.2 MoviesDB

The MoviesDB is a collection of 40 movies with more than 25,000 query images specifically designed to evaluate image-to-video retrieval systems.

Video collection

The MoviesDB contains 40 movies, with more than 7 million frames, and a total duration of more than 80 hours. To ensure diversity in the dataset, we collect a wide range of movie genres, from animation and fantasy to comedy or drama. All the videos have at least 720 pixels width resolution. The shortest video in the dataset is 1 hour 21 minutes and 46 seconds long, whereas the longest movie is 3 hours, 6 minutes and 32 seconds long. Each movie has at least 400 query images. More details, as the duration of each movie, the resolution, the FPS rate, the number of frames or the number of query images, are provided in Table 4.2.

Query images

Query images are captured by a Logitech HD Pro Webcam C920 while movies are being played on a computer screen. We use a Matlab script to control both the movie player and the webcam acquisition time. With this script, when the webcam captures a new query image at a random timestamp, we save the number of the frame that is being shown as a ground truth. To avoid delays produced by the webcam, the video is paused a few seconds before acquiring a new image. Query images have either 960×720 or 2304×1536 pixels resolution and are captured randomly in a time lapse between 0 to 20 seconds. Some examples of query images can be seen in Figure 4.2.

Movie	Length	Resolution	fps	Frames	Queries
12 Years a Slave	2:14:10	832x352	30	196572	723
2 Francs, 40 Pesetas	1:39:27	720x304	25	149195	548
300: Rise of an Empire	1:42:35	720x304	24	147590	541
A Single Man	173542	720x304	24	193008	418
Absolutely Anything	1:21:46	720x400	25	129816	437
American Hustle	2:18:04	720x304	24	198624	753
Ant-Man	1:57:15	720x384	30	211073	603
Big Fish	2:05:08	720x384	24	180035	648
Captain Phillips	2:12:25	720x304	24	190496	618
Casablanca	1:38:20	720x572	25	147483	565
Despicable Me	1:34:48	1280x696	24	136388	499
El Niño	2:16:02	720x320	24	195908	726
Family United	1:37:10	720x306	25	145768	525
Grave of the Fireflies	1:28:30	960x544	25	132750	463
Groundhog Day	1:36:58	720x432	24	145473	542
Harry Potter and the Deathly Hallows: Part I	2:20:06	720x300	25	210165	548
Her	2:05:50	720x384	24	181027	507
Intolerable Cruelty	1:39:40	752x418	30	179234	544
Lee Daniels' The Butler	2:12:04	720x384	24	171262	547
Magnolia	3:00:52	720x304	25	271313	948
Maleficent	1:37:28	720x304	24	140213	467
Marshland	1:39:59	720x316	25	149994	511
Match Point	1:58:54	720x384	25	178351	667
Neon Genesis Evangelion: The End of Evangelion	1:26:49	848x480	25	130225	401
Out of Africa	2:41:02	720x384	24	231673	926
Pirates of the Caribbean: At World's End	2:41:35	720x576	25	241127	881
Puss in Boots	1:30:14	720x304	24	129816	436
Rise of the Planet of the Apes	1:44:19	720x304	24	150072	556
Seven Pounds	2:03:05	720x300	24	177073	631
Spanish Affair 2	1:48:28	720x304	24	156042	586
The Body	1:45:19	720x304	25	157997	568
The Devil Wears Prada	1:49:20	832x352	30	196572	611
The Great Gatsby	2:21:42	720x304	24	203853	764
The Help	2:26:14	720x384	24	210387	813
The Hobbit: The Desolation of Smaug	3:06:32	720x304	24	268357	1040
The Last Circus	1:40:58	720x404	30	181559	543
The Physician	2:34:46	720x304	24	222876	797
The Social Network	2:00:27	720x296	24	173277	626
The Wolf of Wall Street	2:59:52	720x304	24	258759	1027
Witching and Bitching	1:48:42	720x304	25	163069	588

Tab. 4.2: Details of the MoviesDB.

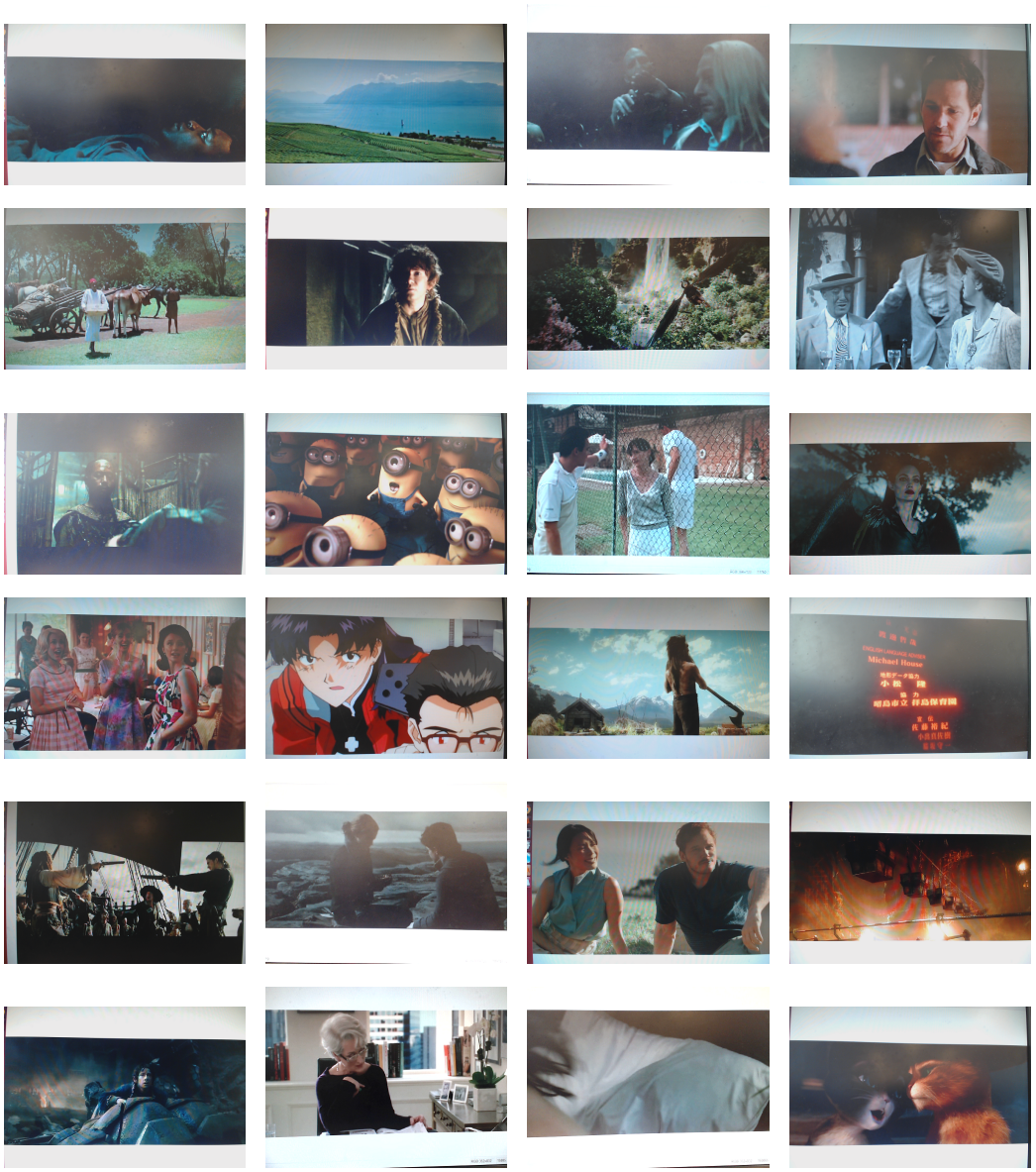
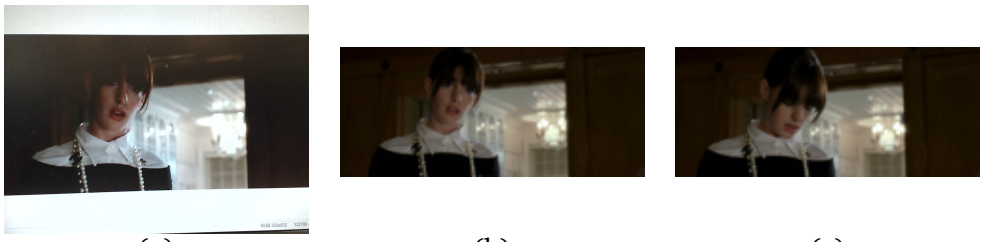


Fig. 4.2: Examples of query images in the MoviesDB.

Evaluation

In the MoviesDB, the performance is evaluated in terms of recall at 1 (R@1) by considering all the frames that are similar to the annotated ground truth as relevant elements, i.e. if the retrieved frame shares strong similarities with the ground truth frame, it is considered a *visual match*. An example of a visual match is shown in Figure 4.3.



(a) (b) (c)

Fig. 4.3: Visual similarities between frames: (a) Query image; (b) Ground truth frame; (c) Retrieved frame, which is similar to ground truth frame and, thus, a Visual Match.

We use SURF features (Bay et al., 2006) to find visual matches. Given a query image, the frame retrieved by the system is compared against the ground truth frame by matching their SURF features. The matching between SURF features is performed as an all-vs-all search, where features in the retrieved frame, i, are compared in terms of distance against features in the ground truth frame, j. Then, the visual similarity score between a pair of frames is computed as:

$$s_{ij} = \frac{|\{\text{Matches}\}_{ij}|}{|\{\text{Features}\}_i|} \tag{4.5}$$

where

- $\{\text{Matches}\}_{ij}$ is the list of matching features between frame i and frame j, where two features are a match when their distance is below a threshold.
- $\{\text{Features}\}_i$ is the list of features in frame i.

If s_{ij} is greater than a threshold, τ, then i is a visual match, VM_i, of j and its associated query:

$$\text{VM}_i = \begin{cases} 1 & \text{if } s_{ij} > \tau \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

The overall performance for a set with Q query images is computed as:

$$\text{R@1} = \frac{1}{Q} \sum_{i=1}^{Q} \text{VM}_i \tag{4.7}$$

The linear comparison between frames that do not present any noise or perspective distortion is a task that SURF features can perform with high accuracy. To measure the precision of this evaluation protocol, we manually annotate either if a pair of frames (i.e. ground truth frame and retrieved frame) is a visual match or not, along with its score. For different values of τ the True Positive Rate (TPR) as well as the False Positive Rate (FPR) are computed as:

$$\text{TPR} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{4.8}$$

$$\text{FPR} = \frac{\text{fp}}{\text{fp} + \text{tn}} \tag{4.9}$$

where

- tp is the number of true positives (i.e. visual match with score $> \tau$)
- fp is the number of false positives (i.e. no visual match with score $> \tau$)
- tn is the number of true negatives (i.e. no visual match with score $\leq \tau$)
- fn is the number of false negatives (i.e. visual match with score $\leq \tau$)

Figure 4.4 shows both the TPR and the FPR computed with the annotations of 615 pairs of frames and 406 different values of τ.

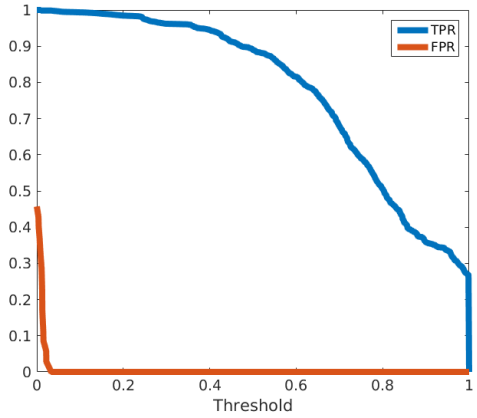


Fig. 4.4: Precision of the evaluation method measured in terms of TPR and FPR for 406 different threshold values and 615 pairs of frames.

We choose $\tau = 0.15$, with TPR $= 0.98$ and FPR $= 0$, for computing visual matches in the MoviesDB. Figure 4.5 shows examples of retrieved frames and their scores with this method.

Fig. 4.5: Examples of scores between ground truth frames (left column) and retrieved frames (right column) in the MoviesDB evaluation.

4.4 Applications

Image-to-video retrieval has been successfully applied in fields such as video search (Araujo et al., 2015a) and video bookmark (Chen et al., 2010). As an alternative application, we propose a framework based on image-to-video retrieval for video content augmentation. The main idea is shown in Figure 4.6.

Fig. 4.6: Video content augmentation with fashion items in a TV show.

Videos such as films and TV shows are a powerful marketing tool, especially for the fashion industry, since they can reach thousands of millions of people all over the world and impact on fashion trends. Spectators may find clothing appearing in movies and television appealing and people's personal style is often influenced by the multimedia industry. Also, online video-sharing websites, such as YouTube[1], have millions of users generating billions of views every day and famous *youtubers* are often promoting the latest threads in their videos.

Fashion brands are interested in selling the products that are advertised in movies, television or YouTube. However, buying clothes from videos is not straightforward. Even when a user is willing to buy a fancy dress or a trendy pair of shoes that

[1] https://www.youtube.com/

appear in the latest blockbuster movie, there is often not enough information to complete the purchase. Finding the item and where to buy it is, most of the times, difficult and it involves time-consuming searches.

To help in the task of finding fashion products that appear in multimedia content, some websites, such as *Film Grab*[2] or *Worn on TV*[3], provide catalogs of items that can be seen on films and TV shows, respectively. These websites, although helpful, still require some effort before actually buying the fashion product: users need to actively remember items from videos they have previously seen and navigate through the platform until they find them.

To retrieve fashion products from videos in an effortless and non-intrusive way, we propose a framework based on image-to-video retrieval. By taking a picture of the screen during video playback, the framework identifies the corresponding frame of the video sequence and returns that frame augmented with the fashion items in the scene. In this way, users can find a product as soon as they see it by simple taking a photo.

Instead of retrieving products directly as in clothing retrieval (Liu et al., 2012; Hadi Kiapour et al., 2015), we propose to first retrieve frames from the video collection. The reasons are three-fold. Firstly, in standard clothing retrieval users usually provide representative images of the object of interest (e.g. dresses in front view, high-heeled shoes in side view, etc.), whereas in a movie, the view of the object of interest cannot be chosen, and items might be partially or almost completely occluded, such as the red-boxed dress in Figure 4.6. Secondly, clothing retrieval usually requires to select a bounding box around the object of interest. This is undesirable in video as it may distract user's attention from the original content. Finally, performing frame retrieval instead of product retrieval in videos allows users to get the complete look of a character, including small accessories such as earrings, watches or belts. Some of these items are normally very small,

[2]http://filmgarb.com/
[3]https://wornontv.net

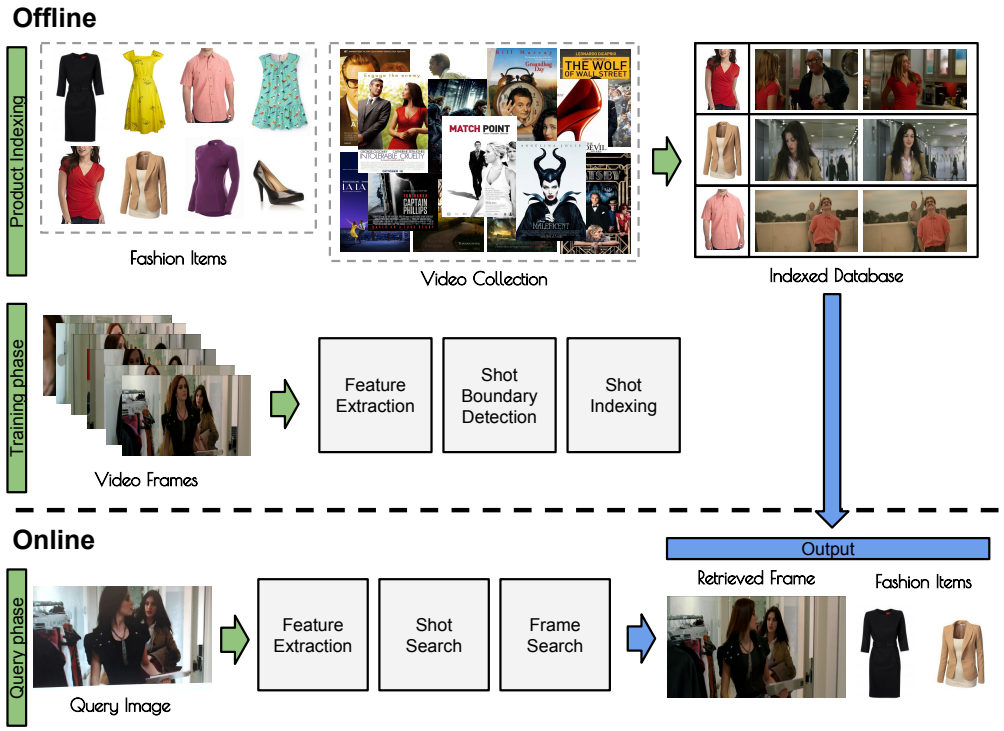


Fig. 4.7: Framework for video content augmentation based on image-to-video retrieval. During the product indexing, fashion items and frames are associated in an indexed database. Then, frames are indexed by using image-to-video retrieval techniques. Finally, a query image is used to retrieve frames and their associated fashion products.

sometimes almost invisible, and are very difficult to detect and recognize using standard object retrieval techniques.

As shown in Figure 4.7, the framework consists on three phases. In the product indexing, fashion items and frames from the video collection are related in an indexed database. This process can be done manually (e.g. with Amazon Mechanical Truck[4]) or semi-automatically with the support of a standard clothing retrieval algorithm. Then, the training and the query phase are performed using image-to-video retrieval techniques, as the ones presented in Chapter 5 and Chapter 6, to extract and index spatio-temporal features from frames and detect and identify shots in video segments.

[4]https://www.mturk.com

5 Image-to-video retrieval based on binary features

In this chapter, we address image-to-video retrieval by exploiting temporal redundancy in local binary features, achieving a 42:1 compression ratio with respect to non-aggregation methods while maintaining accuracy at similar levels.

5.1 Methodology

We propose an aggregation method for image-to-video retrieval based on the temporal local aggregation of video features (see Figure 4.1), in which recurrent local features in a video segment are aggregated to compress the temporal local information. The main advantage of temporal local aggregation methods with respect to spatio-temporal global aggregation methods, where a single compact vector per video segment is computed, is a better retrieval accuracy at the expense of a lower compression ratio and a slower search.

To improve compression and search time in temporal local aggregation methods, we propose the system shown in Figure 5.1. In the training phase, we firstly reduce the memory requirements by extracting and aggregating local binary features (Section 5.1.1), as binary features require less memory storage than floating-point local features such as SIFT and SURF. Moreover, when the Hamming distance is applied, the matching between binary features is faster than computing the Euclidean distance between standard local features (Miksik and Mikolajczyk, 2012). Secondly, to speed-up the computation time, we conduct a nearest neighbours search by indexing the aggregated binary features in a kd-tree (Section

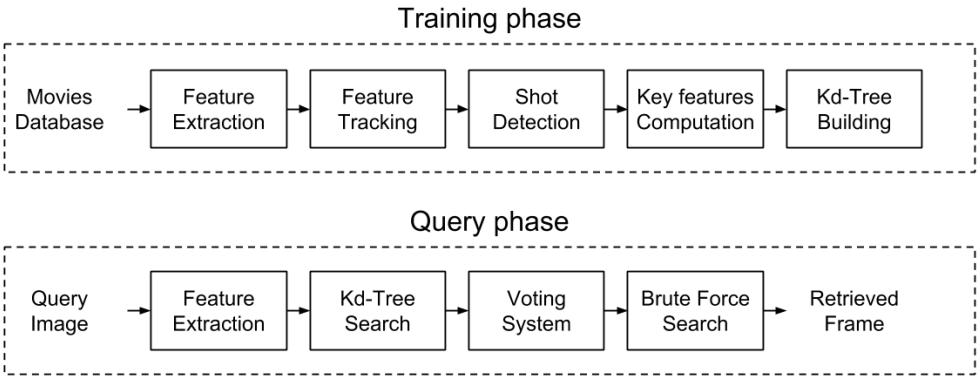


Fig. 5.1: Block diagram of the local temporal aggregation system.

5.1.2). In the query phase (Section 5.1.3), binary features extracted from a query image are used to find their nearest aggregated features in the kd-tree. To retrieve the video shot the query belongs to, we implement a voting system using the nearest aggregated features. For a more precise retrieval, a frame search within all the frames in the retrieved video shot is conducted.

5.1.1 Local Temporal Aggregation of Binary Features

Our local temporal aggregation process is summarized in Figure 5.2.

Feature Extraction and Tracking

To detect recurrent local features in a video segment, hand-crafted binary features (e.g. BRIEF, Calonder et al., 2010) are extracted from every frame in the video collection and tracked along time by applying descriptor and spatial filters. The tracking is performed in a bidirectional way so features within a track are unique, i.e. each feature is only matched with up to two features: one in the previous frame and one in the following frame.

The reasons for using local binary features instead of more popular local features, such as SIFT, are two-fold. Firstly, Hamming distance for binary features is faster to compute than Euclidean distance for floating-points vectors (Miksik and

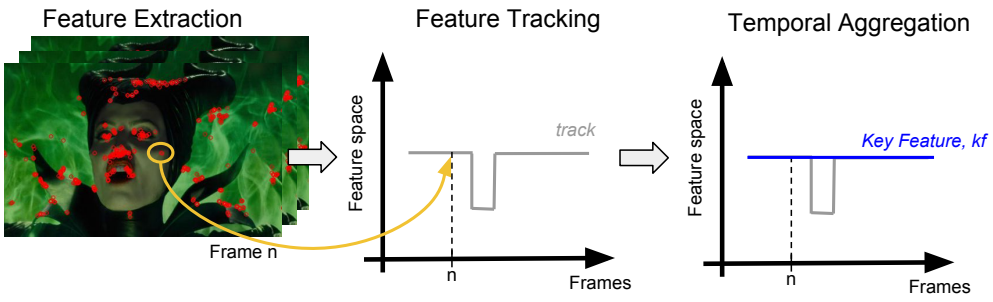


Fig. 5.2: Temporal aggregation of local binary features.

Mikolajczyk, 2012). Secondly, binary features are more stable over time than SIFT, as shown in Figure 5.3, and hence, less information may be lost.

Shot Detection

Consecutive frames that share visual similarities are grouped into shots. The boundaries of different shots are detected when two consecutive frames have no common tracks. Each shot contains a set of tracks, each track representing the trajectory of a particular feature along time.

Key Feature Aggregation

We define a *key feature* as the aggregation of all the features in the same track into a single vector. Subsequently, each shot is then represented by a set of key features, similarly to how frames are represented by a set of features. For each track, a key feature is computed by using majorities (Grana et al., 2013). If the majority value at a certain dimension of the features of the track is 1, then associated key feature's value at that dimension will be 1. Otherwise, the value will be 0. To avoid adding noisy features to the system, only stable tracks longer than a fixed number of frames are considered.

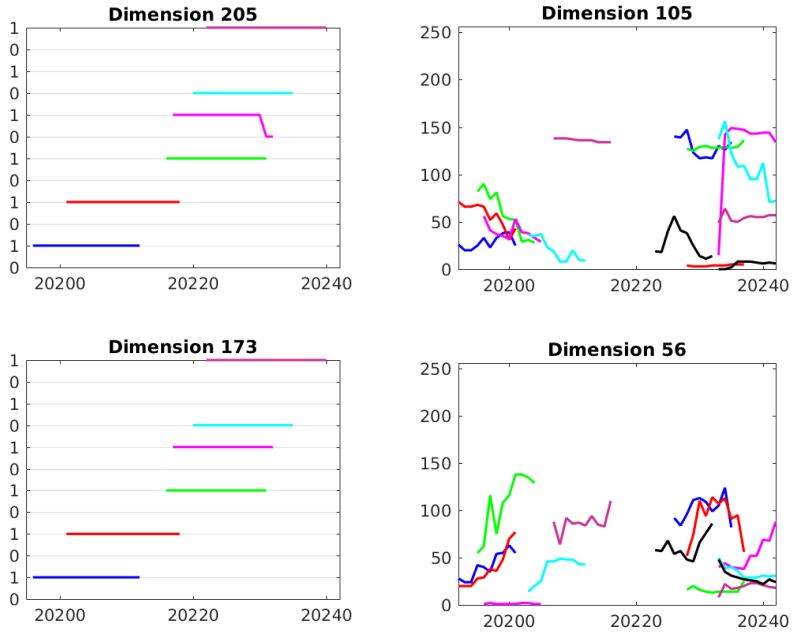


Fig. 5.3: Trajectories of tracks along a sequence of frames. Binary features are more constant over time than SIFT features: (a) BRIEF; (b) SIFT.

5.1.2 Feature Indexing

To speed-up the search computation time, we use a kd-tree (Bentley, 1975) to index the aggregated key features. A popular method for searching in binary space is FLANN (Muja and Lowe, 2012), which uses multiple, randomly generated hierarchical structures. However, kd-trees have been shown to be highly parallelizable (Aly et al., 2011), which is very suitable for large-scale solutions. Thus, we propose to modify the basic kd-tree structure to handle binary features and perform a fast search of aggregated key features.

In a kd-tree (Bentley, 1975), each decision node has an associated dimension, a splitting value and two child nodes. For a query vector, if the value of in the associated dimension is greater than the splitting value, the vector is assigned to the left child; otherwise, the vector is associated to the right child. This process is repeated at each node until a leaf node is reached.

To deal with binary features, we modify the standard kd-tree so that each decision node has an associated dimension, dim, such that descriptor vectors, $\mathbf{d}$, with $\mathbf{d}[dim] = 1$ are assigned to the left child, and vectors with $\mathbf{d}[dim] = 0$ are assigned to the right child. The value dim is chosen such that the training data is split more evenly in that node, i.e. its entropy is maximum. Note that this criterion is similar to the one used in the ID3 algorithm (Quinlan, 1986) for the creation of decision trees, but where the splitting attribute is the one with smallest entropy. Leaf nodes have as many as S_L indices pointing to the features that ended up in that node. This kd-tree building algorithm is described in Algorithm 1.

Algorithm 1 Kd-tree building with binary features building

1: $\mathbf{D}$: set of binary descriptors
2: **procedure** BUILD-TREE($\mathbf{D}$)
3: **if** $|\mathbf{D}| < S_L$ **then**
4: **return** Leaf($\mathbf{D}$)
5: **else**
6: $dim = \arg\min_x |0.5 - \#\{\mathbf{d} \text{ in } \mathbf{D} : \mathbf{d}[x] = 1\}/|\mathbf{D}||$
7: $\mathbf{D}_{\text{left}} = \mathbf{d}$ in $\mathbf{D}$: where $\mathbf{d}[dim] = 1$
8: $\mathbf{D}_{\text{right}} = \mathbf{d}$ in $\mathbf{D}$: where $\mathbf{d}[dim] = 0$
9: **return** Tree(dim, build-tree($\mathbf{D}_{\text{left}}$), build-tree($\mathbf{D}_{\text{right}}$))

5.1.3 Search and Retrieval

In the query phase, binary features are extracted from an input image and assigned to its nearest set of key features by searching down the kd-tree. A first-in first-out (FIFO) queue keeps record of the already visited nodes in the kd-tree to backtrack B times and explore them later. We use a queue system to ensure that even if some of the bits in a query vector are wrong, the vector can reach its closest neighbours by exploring unvisited nodes latter.

Key features in the leaf nodes found in the search are added to a candidates list, so the candidate vectors with the minimum Hamming distance to the query vector are their nearest neighbours key features. The kd-tree search algorithm using binary features is described in Algorithm 2.

Each nearest neighbour key feature found in the kd-tree votes for the shot it belongs to, and the most voted shot is retrieved. For a fine-grained retrieval, the frames in the retrieved shot are compared against the input image using linear search, i.e. the candidate frame with the minimum distance to the query image is retrieved. Shots are commonly groups of a few hundreds of frames, thus the computation can be performed very rapidly when applying the Hamming distance.

Algorithm 2 Kd-tree search with binary features

```
1: T: kd-tree
2: q: binary query vector
3: B: maximum number of backtracking steps
4: procedure SEARCH-TREE(T, q, B)
5:     Q ← T.Root                                  ▷ add root node to FIFO queue
6:     Candidates ← []                             ▷ empty list of candidate vectors
7:     while B > 0 do
8:         Node ← pop(Q)                           ▷ get node from Q
9:         while Node not leaf node do
10:            Q ← T.Node
11:            if d[Node.dim] = 1 then
12:                Node ← Node.leftChild
13:            else
14:                Node ← Node.rightChild
15:        Candidates ← Node.D                     ▷ features to candidates
16:        B ← B − 1
17:    D_nn ← d ∈ Candidates| arg min_d Hamming(d, q)
18:    return D_nn
```

5.2 Evaluation

To evaluate the proposed system, we perform two different experiments. In the first one, our image-to-video retrieval system is compared against other retrieval systems. In the second one, we evaluate the performance of the system when scaling up the video collection.

5.2.1 Implementation Details

We evaluate our method using the MoviesDB. Frames and query images are resized to 720 pixels in width. Binary features are computed by using ORB detector (Rublee et al., 2011) and BRIEF extractor (Calonder et al., 2010). In the tracking module, only matches with a Hamming distance less than 20 and a spatial distance less than 100 pixels are considered, whereas in the key feature computation algorithm, only tracks longer than 7 frames are used. The default values for the kd-tree are set at $S_L = 100$ and $B = 50$.

5.2.2 Retrieval Results

First, we compare our system against other retrieval baselines using the movie The Devil Wears Prada in the MoviesDB, which consists of 196,572 frames with a total duration of 1 hour 49 minutes and 20 seconds and 615 query images. The other systems under evaluation are:

- **Bi-BruteForce**: Brute force search using binary features. Query images are matched against all frames and all features in the database using Hamming distance. Brute Force system is only used as an accuracy benchmark, since each query take, in average, 46 minutes to be processed. Temporal information is not exploited.

- **Bi-KdTree**: Kd-Tree search using binary features, in which all BRIEF features from all frames are indexed using a binary kd-tree structure. Temporal information is not exploited.

- **Bi-KeyFrame-KdTree**: Key frame extraction method (Sun et al., 2008), in which temporal information is used to reduce the amount of frames of each shot into a smaller set of key frames. Key frames are chosen as the ones at the peaks of the distance curve between frames and a reference image

Method	Feat	Mem	R@1			
			B=10	B=50	B=100	B=250
Bi-BruteForce	85	2591	——— 0.98 ———			
Bi-KdTree	85	2591	0.90	0.94	0.96	0.97
Bi-KeyFrame-KdTree	25	762	0.91	0.92	0.93	0.93
SIFT-Aggregation-KdTree	0.9	446	0.61	0.67	0.70	0.73
Bi-Aggregation-KdTree	2	61	0.92	0.93	0.94	0.94

Tab. 5.1: Comparison between different systems on a single movie. Number of features (Feat) is given in millions and memory (Mem) is given in megabytes.

computed for each shot. For each key frame, binary features are extracted and indexed in a binary kd-tree structure.

- **Bi-Aggregation-KdTree**: Our proposed method using binary features, temporal aggregation based on majorities and a binary kd-tree for indexing aggregated binary features.

- **SIFT-Aggregation-KdTree**: SIFT variant of our method, in which SIFT features are extracted and tracked along time. To aggregate tracks we compute the average value. Aggregated features are indexed in a standard kd-tree with a priority queue as in Aly et al., 2011.

Results are detailed in Table 5.1. R@1 is similar for all the methods based on binary features. However, the number of features and the memory requirements are drastically reduced when the temporal information is used. In our proposed model (Bi-Aggregation-KdTree), by exploiting temporal redundancy between frames, the memory is reduced by 42.5 times with respect to Bi-BruteForce and Bi-KdTree and by 12.5 times with respect to Bi-KeyFrame. Theoretically, that means that when implemented in a distributed kd-tree system as the one in Aly et al., 2011, where the authors were able to process up to 100 million images, our system might be able to deal with 4,250 million frames, i.e. more than 20,000 movies and 40,000 hours of video. Accuracy with the aggregated SIFT features

(SIFT-Aggregation-KdTree) is considerably worse than with the aggregated binary features (Bi-Aggregation-KdTree), probably because of the variability on the temporal tracks, as shown in Figure 5.3.

5.2.3 Large-Scale Results

We explore the scalability of our framework by increasing the size of the video collection and using the whole MoviesDB, with 40 movies and more than 25,000 query images. Results are shown in Table 5.2. By using our temporal aggregation method, the amount of data is reduced from 7 million frames and 3,040 million features to only 116,307 shots and 58 million key features. Even so, the total number of key features in the 40 movie collection is still smaller than the 80 million features that, in average, a single movie contains. The total accuracy over the 40 movies is 0.87, reaching values of 0.98 and 0.97 in The Help and Intolerable Cruelty movies, respectively. Movies with very dark scenes such as Captain Phillips and Pirates of the Caribbean 3 perform the worst, as fewer descriptors can be found in those kinds of dimly lit images.

Figure 5.4 shows the evolution of accuracy when the size of the database increases for five different movies. Most of the movies are not drastically affected when the number of frames in the database is increased from 200,000 to 7 million. For example, both Intolerable Cruelty and 12 Years a Slave maintain almost a constant accuracy for different sizes of the collection. Even in the worst case scenario, The Devil Wears Prada movie, the loss in accuracy is less than a 8.5%. This suggests that our image-to-video retrieval system is enough robust to handle large-scale video collections without an appreciable loss in performance.

Title	Frames	Feat	Shots	KFeat	R@1
12 Years a Slave	193008	86M	1409	1.6M	0.94
2 Francs, 40 Pesetas	149195	69M	1126	1.4M	0.95
300: Rise of an Empire	147590	68M	2110	1.2M	0.83
A Single Man	173542	53M	3584	1.1M	0.90
Absolutely Anything	129816	56M	1684	1.1M	0.95
American Hustle	198624	83M	2047	1.5M	0.88
Ant-Man	211073	91M	5057	1.8M	0.81
Big Fish	180035	74M	3682	1.5M	0.90
Captain Phillips	190496	59M	7578	0.6M	0.67
Casablanca	147483	71M	881	1.5M	0.96
Despicable Me	136388	65M	1886	1.2M	0.92
El Niño	195908	86M	3424	1.3M	0.82
Family United	145768	61M	2152	1.3M	0.90
Grave of the Fireflies	132750	60M	1399	1.2M	0.94
Groundhog Day	145473	62M	1174	1.2M	0.93
Harry Potter	210165	60M	7187	1.3M	0.78
Her	181027	53M	5131	1.1M	0.93
Intolerable Cruelty	179234	86M	1306	2M	0.97
Lee Daniels' The Butler	171262	65M	2413	1.3M	0.91
Magnolia	271313	100M	3806	2M	0.89
Maleficent	140213	57M	3355	1M	0.81
Marshland	149994	60M	2310	1.1M	0.90
Match Point	178351	81M	918	1.7M	0.91
Neon Genesis Evangelion	130225	53M	4914	1.1M	0.90
Out of Africa	231673	108M	2595	2.3M	0.93
Pirates of the Caribbean	241127	108M	3695	1.7M	0.74
Puss in Boots	129816	54M	2835	0.8M	0.80
Planet of the Apes	150072	69M	2482	1.1M	0.82
Seven Pounds	177073	70M	2878	1.3M	0.88
Spanish Affair 2	156042	75M	1270	1.5M	0.96
The Body	157997	71M	2048	1.5M	0.89
The Devil Wears Prada	196572	85M	1822	2M	0.85
The Great Gatsby	203853	98M	3427	1.7M	0.88
The Help	210387	101M	1726	2.2M	0.98
The Hobbit	268357	120M	4762	1.8M	0.83
The Last Circus	181559	85M	3126	1.4M	0.81
The Physician	222876	86M	3051	1.7M	0.85
The Social Network	173277	63M	2804	1.3M	0.89
The Wolf of Wall Street	258759	123M	3060	2.3M	0.87
Witching and Bitching	163069	66M	4193	0.8M	0.74
Total	**7M**	**3040M**	**116307**	**58M**	**0.87**

Tab. 5.2: Results on the MoviesDB using a local temporal aggregation method based on binary features and kd-trees.

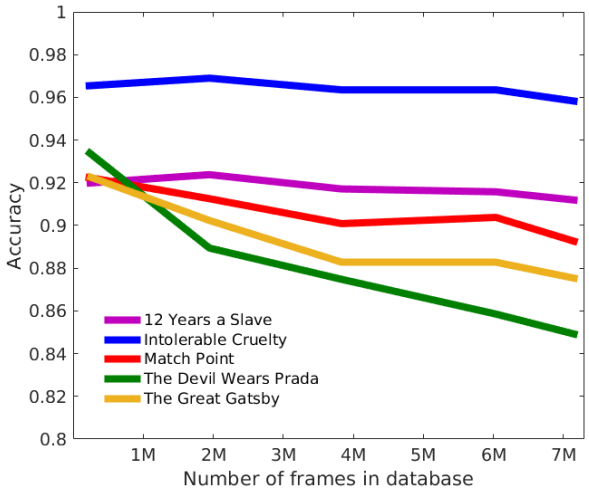


Fig. 5.4: Accuracy vs Database size for 5 different movies.

5.3 Conclusions

In this chapter, we proposed a system to aggregate local features in videos using binary descriptors. Local temporal aggregation methods usually obtain better performance than global temporal aggregation methods at the expense of an increased search time and more memory storage. We efficiently reduced memory requirements by using binary features and fast indexing techniques.

We aggregated recurrent binary descriptors in a video segment using majorities, and we indexed aggregated features in a kd-tree. At query time, the kd-tree was used to find the nearest aggregated features, which voted for the shot they belonged to. In the experiments, the amount of data to be processed could be reduced by a factor of 42.5 with respect to linear search, whereas accuracy was maintained at similar levels. We also showed that our system scaled well when the number of frames increased from 200,000 to 7 million.

However, despite the improvements in memory storage, the proposed approach needs to perform multiple searches per query image to retrieve a single video frame. In the next chapter, we present a global temporal aggregation system, in which only one search per query image is performed.

6 Image-to-video retrieval based on deep learning

This chapter approaches image-to-video retrieval by using deep learning techniques. Instead of aggregating multiple local features per frame as in the previous chapter, here we propose to encode multiple frames into a single compact representation, which allows us to perform a single search per query image. Our method is specifically trained for image-to-video retrieval and outperforms previous work on global temporal aggregation methods.

6.1 Methodology

Temporal global aggregation methods for image-to-video retrieval are usually more efficient than methods based on the temporal aggregation of local features, as only one search per query image needs to be performed (see Figure 4.1). In this chapter, we propose a deep learning architecture for learning asymmetric spatio-temporal visual embeddings specifically for image-to-video retrieval. We use a spatio-temporal encoder (Section 6.1.1) to project images and videos into a common embedding space, as depicted in Figure 6.1, where a standard similarity function can be applied to rank videos according to their similarity with respect a query image (Section 6.1.2). To learn the network parameters, a contrastive or margin loss function is computed between pairs of images and video clips during training (Section 6.1.3).

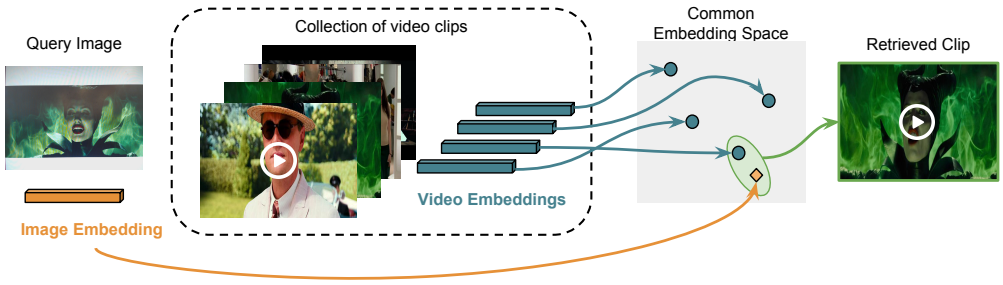


Fig. 6.1: In temporal global aggregation methods for image-to-video retrieval, image and video global embeddings obtained from query images and video clips are projected into a common embedding space to compute similarities and find a specific video clip.

6.1.1 Spatio-Temporal Encoder

The spatio-temporal encoder architecture is shown in Figure 6.2. With a spatial encoder based on convolutional neural networks (CNN), query images and video frames are independently mapped into image embeddings. Image embeddings from frames within the same video clip are then input into a temporal encoder, which is based on recurrent neural networks (RNN), to obtain a set of video or shot embeddings describing the visual content of the whole scene.

Spatial Encoder

To capture the spatial visual content of query images and frames and compute meaningful image embeddings, we use RMAC image descriptor (Tolias et al., 2016), which is based on max-pooling the output of the last convolutional layer of a pre-trained CNN over several regions. A detailed description of the RMAC algorithm can be found in Section 2.1.3.

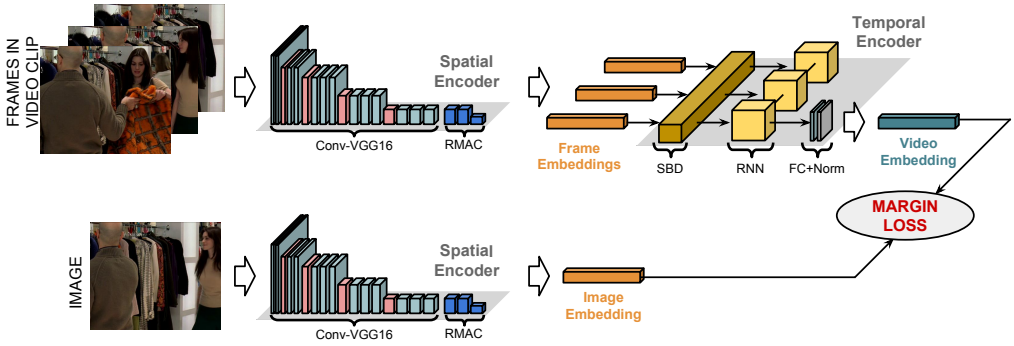


Fig. 6.2: Spatio-temporal encoder to learn compact embeddings for image-to-video retrieval. Images and frames are mapped into image embeddings with the Spatial Encoder. Videos are mapped into video embeddings with the Temporal Encoder. A margin loss function between image and video embeddings is computed to learn the weights of the architecture.

Temporal Encoder

We capture the temporal visual information using a temporal encoder model. As detailed in Section 4.2, videos are composed of a set of frames, shots and scenes. Commonly, frames belonging to the same shot are highly correlated. With our temporal encoder, we take advantage of this inner temporal structure of videos to first, identify shots within a video and then, encode the visual information within a shot with a recurrent neural network.

Shot Boundary Detection: We split up each video into a collection of shots by using a shot boundary detection (SBD) algorithm. The aim of the SBD algorithm is to detect groups of similar looking frames to encode them into a single video embedding. Although there exist many SBD algorithms (Hassanien et al., 2017; Gygli, 2017) to detect different kinds of transitions between shots, such as fade in, fade out, wipes or dissolves, to aggregate features for image-to-video retrieval we are only interested in *hard cuts*, i.e. when two shots are put one after the other without any transition effect.

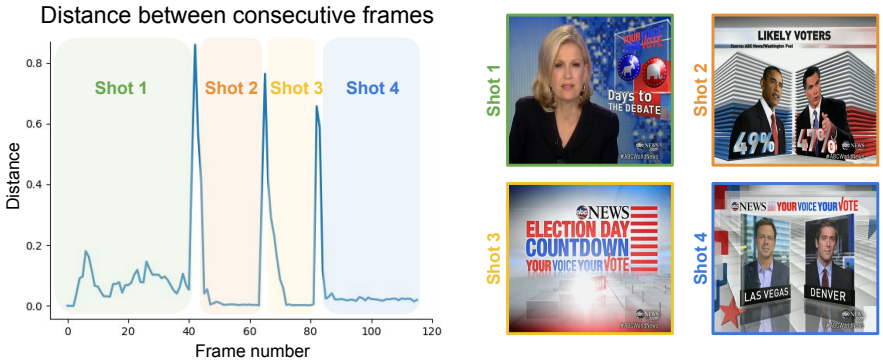


Fig. 6.3: Shot boundary detection algorithm; Left: Shot boundaries detected when the distance between consecutive frames is high; Right: Frames from each detected shot.

To detect hard cuts, we use Algorithm 3. We compute the distance between each pair of consecutive frame embeddings along the duration of a video and assign a shot boundary when the distance is higher than a predefined threshold, Th. An example of the frame distances computed with the SBD algorithm along with sample frames from each detected shot can be seen in Figure 6.3.

Algorithm 3 Shot Boundary Detection with RMAC

procedure SBD($video$)
  $List_{SB} \leftarrow []$
  $i \leftarrow 0$
  $\mathbf{F} \leftarrow$ getFrame$(video, i)$ ▷ Get first frame
  $\mathbf{v}_0 \leftarrow$ RMAC$(\mathbf{F})$ ▷ Compute image embedding
  while hasFrame$(video, i+1)$ **do**
    $\mathbf{F} \leftarrow$ getFrame$(video, i+1)$ ▷ Get new frame
    $\mathbf{v} \leftarrow$ RMAC$(\mathbf{F})$ ▷ Compute image embedding
    $dist \leftarrow 1 - \frac{\mathbf{v}_0 \cdot \mathbf{v}}{\|\mathbf{v}_0\|_2 \|\mathbf{v}\|_2}$ ▷ Distance between consecutive frames
    if $dist >$ Th **then**
      push$(List_{SB}, i)$ ▷ Boundary if $dist >$ Th
    $\mathbf{v}_0 \leftarrow \mathbf{v}$
    $i \leftarrow i + 1$
  return $List_{SB}$

Recurrent Neural Networks: To aggregate the temporal visual information of frames within a shot, we use a RNN. We explore different RNN models, such as long short-term memory networks (LSTM, Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU, Cho et al., 2014). At each state of the RNN, we

input each of the frame embeddings belonging to the shot. The RNN captures the salient temporal information in the sequence by using at each state both the current frame embedding and the output of the previous frame embeddings. The output of the last state is further processed with a fully connected layer, a *tanh* non-linearity and a $\ell 2$-normalization to obtain the video or shot embedding.

Formally, let $\mathbf{h}_{|\mathbf{X}|} = \text{RNN}(\mathbf{X})$ be the output of the last state of a recurrent neural network that processes the sequence of frame embeddings $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{|\mathbf{X}|}]$. The shot embedding is computed as $\mathbf{v}_\mathbf{X} = \text{norm}(\tanh(\mathbf{W} \cdot \mathbf{h}_{|\mathbf{X}|} + \mathbf{b}))$, where $\text{norm}(\mathbf{z}) = \frac{\mathbf{z}}{\|\mathbf{z}\|_2}$, $\tanh(\mathbf{z}) = \frac{e^{\mathbf{z}} - e^{-\mathbf{z}}}{e^{\mathbf{z}} + e^{-\mathbf{z}}}$ and $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and the bias vector of the last fully connected layer, respectively. The shot embedding is set to have the same dimensionality, K, as the image embedding.

6.1.2 Search and Retrieval

To perform image-to-video search and retrieval, we rank videos according their similarity to the query image. We compute image-shot similarity between an image embedding, $\mathbf{u}$, and a shot embeddings, $\mathbf{v}$, as the cosine similarity:

$$\text{sim}_{\text{shot}}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2} \tag{6.1}$$

The visual similarity between a query embedding, $\mathbf{q}$ and a specific video, $\mathbf{V}$, with a set of shot embeddings, $\mathbf{V} = \{\mathbf{v}_k\}$ with $k \leq |\mathbf{V}|$, is the max-pooled image-shot similarity between the query and all the shot embeddings in the video:

$$\text{sim}_{\text{video}}(\mathbf{q}, \mathbf{V}) = \max(\text{sim}_{\text{shot}}(\mathbf{q}, \mathbf{v}_k)) \tag{6.2}$$

6.1.3 Margin Loss Function

We train the model using pairs of images and video shots, where images are frames from the same video collection as shots. For the i-th training pair, we

denote as $\mathbf{u}^i$ to the image embedding representing the frame and as $\mathbf{v}^i$ to the shot embedding representing the shot. For each pair, we automatically assign a positive or a negative label, y^i, as:

$$y^i = \begin{cases} 1 & \text{if } \mathbf{u}^i \text{ and } \mathbf{v}^i \text{ belong to the same shot} \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

We compute the loss of a pair as the cosine similarity with a margin, Δ, between the image and shot embeddings:

$$\text{Loss}(\mathbf{u}^i, \mathbf{v}^i) = y^i(1 - \cos(\mathbf{u}^i, \mathbf{v}^i)) + (1 - y^i)(\max(0, \cos(\mathbf{u}^i, \mathbf{v}^i) - \Delta)) \tag{6.4}$$

6.2 Evaluation

We evaluate the spatio-temporal encoder and compare the model against state-of-the-art temporal global aggregation methods for image-to-video retrieval.

6.2.1 Datasets

To learn the parameters of our model, we use a related video collection. Then, we evaluate the model using some of the image-to-video retrieval datasets introduced in Section 4.3.

Training Dataset

To train our model we use the data from the LSMDC dataset (Rohrbach et al., 2017). The LSMDC dataset contains 202 movies split into 128,118 short video clips of about 5 seconds. We remove the movies that overlap with our evaluation datasets and select a subset of 40 movies with 26,495 clips for training and 10

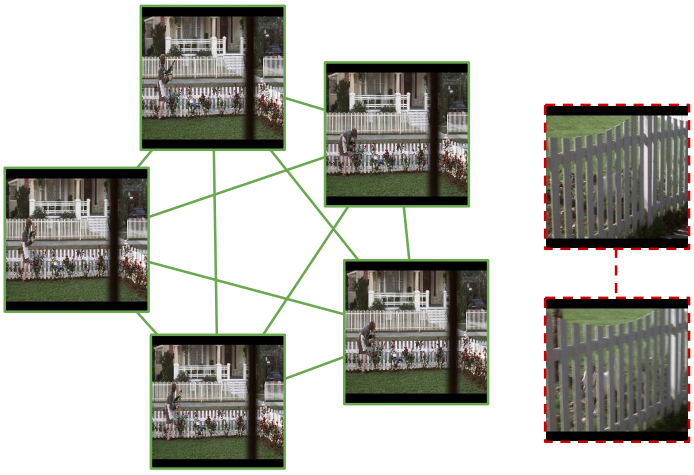


Fig. 6.4: Data graph of frames in which nodes are frames and connections are matches. In the cleaning process, we keep frames in the strongest component (solid green lines) and remove the rest (dashed red lines).

movies with 7,440 clips for validation, to speed up the training. We use clips provided in LSMDC as training shots.

As clips in LSMDC do not exactly correspond to video shots (i.e. each clip is a short sequence which may contain frames from one or more shots), we conduct a cleaning process (Gordo et al., 2017) to keep only frames from the longest shot for each clip. We extract SIFT features (Lowe, 2004) from all the frames in the clip and perform an all-versus-all feature matching between all possible pairs of frames in the video clip. For each pair of frames, we assign a score as the number of shared descriptors over the total number of descriptors, keeping only scores greater than 0.25. Next, we build a graph where each node corresponds to a frame and each connection corresponds to their assigned score. We extract the strongest component of the graph and remove the nodes (i.e. frames) that do not belong to it. See Figure 6.4 for an example of graph.

Evaluation Datasets

We evaluate the proposed model in the following datasets, which are detailed in Section 4.3.2:

Fig. 6.5: Query images and video clip examples for each of the evaluation datasets.

- SI2V-600k (Araujo et al., 2015a): 164 hours of newscast videos with 3,401 clips and 229 images from news websites.

- VB-600k (Araujo et al., 2016): same videos as in SI2V-600k with 282 queries captured with an external camera.

- MoviesDB: the lighter version, which consists on a single movie (The Devil Wear Prada), with about 2 hours duration and 615 query images.

Examples of each of the evaluation datasets are shown in Figure 6.5.

6.2.2 Implementation Details

Frames are extracted at three frames per second rate and resized to 1024 pixels width. In the spatial encoder, RMAC representations are obtained with a VGG16 network (Simonyan and Zisserman, 2015) pre-trained for image classification, without the last fully connected layers. The dimensionality of the image embeddings, K, is 512. PCA-whitening is implemented as a fully connected layer and its weights are computed using American Beauty movie from the training collection. In the temporal encoder, the dimensionality of the hidden state of the RNN is 512, as well as the number of filters in the last fully connected layer. The maximum number of frames per shot used in the RNN is 50. At training time, Δ is set

		SI2V-600k		VB-600k	
Method	**dim**	**mAP**	**R@1**	**mAP**	**R@1**
Edge Histogram [1]	-	0.154	0.372	-	-
JDC [1]	-	0.174	0.384	-	-
PHOG [1]	-	0.223	0.450	-	-
AlexNet FC6 [2]	4,096	0.484	-	0.182	-
AlexNet FC7 [2]	4,096	0.363	-	0.157	-
VGG16 FC6 [2]	4,096	0.344	-	0.070	-
VGG16 FC7 [2]	4,096	0.316	-	0.048	-
FV [2]	4,096	0.715	-	0.704	-
RMAC	512	0.718	0.834	0.643	0.592

Tab. 6.1: Comparison between different image representations methods when frames are processed independently without temporal aggregation (i.e. image-to-image retrieval). [1] are implementations from Oliveira Barra et al., 2016 and [2] from Araujo and Girod, 2017.

to 0.1. We assign 20% of training pairs as positive and 80% as negative. The spatial encoder is frozen. We optimize the parameters of the temporal encoder using Adam (Kingma and Ba, 2015) with backpropagation, batch size of 512 and learning rate of 0.0001. In the SBD algorithm, Th is experimentally chosen as 0.5. Query images are resized to 960 pixels width.

6.2.3 Spatial Encoder Results

We first evaluate the performance of the RMAC representation as the spatial encoder of the model. We compare results obtained with different image representation methods when frames are processed as independent images (i.e. image-to-image retrieval). For a fair comparison, none of the networks are retrained or fine-tunned.

Results are summarized in Table 6.1. RMAC obtains comparable performance to the best reported results (FV in Araujo and Girod, 2017) by using 8 times less

	SI2V-600k		VB-600k		MoviesDB
Method	**mAP**	**R@1**	**mAP**	**R@1**	**R@1**
Max-Pooling	0.038	0.066	0.033	0.011	-
Sum-Pooling	0.152	0.275	0.316	0.262	-
Temporal Encoder (LSTM)	0.602	0.773	0.580	0.525	0.833
Temporal Encoder (GRU)	0.606	0.777	0.572	0.514	0.833

Tab. 6.2: Comparison between different techniques to aggregate image embeddings (RMAC) from multiple frames (i.e. image-to-video retrieval).

memory. When compared against other deep learning features (AlexNet FC6, AlexNet FC7, VGG16 FC6 and VGG FC7) RMAC is, by far, superior.

6.2.4 Temporal Encoder Results

Next, we evaluate our proposed temporal encoder. RMAC is used to obtain frame embeddings. We evaluate two versions of the temporal encoder, one based on LSTM (Hochreiter and Schmidhuber, 1997) and another one based on GRU (Cho et al., 2014) and we compare them against simple aggregation baselines, such as Max-Pooling and Sum-Pooling. Max-Pooling consists on computing the maximum value of the frame embeddings for each dimension and Sum-Pooling consist on summing up all the frame embeddings into a single vector.

Results are shown in Table 6.2. Max-Pooling and Sum-Pooling are not evaluated on MoviesDB as there is only one video in the collection. Our temporal encoder is considerably superior to both baselines. In SI2V-600k, the temporal encoder based on GRU obtains the best performance. In VB-600k, LSTM is superior. With respect to MoviesDB, both LSTM and GRU perform equally well.

Method	mAP	R@1
Sum-Pool, No SBD	0.152	0.275
Sum-Pool, Rand. SBD	0.322	0.589
Sum-Pool, Our SBD	0.596	0.764
LSTM, No SBD	0.121	0.319
LSTM, Rand. SBD	0.390	0.616
LSTM, Our SBD	0.602	0.773

Tab. 6.3: Analysis of the SBD algorithm in the SI2V-600k dataset.

Shot Boundary Detection

We evaluate the contribution of the shot boundary detection algorithm by considering three different scenarios: no shot boundary detection (No SBD), shot boundaries are selected randomly (Rand. SBD), and our proposed shot boundary detection algorithm based on RMAC distances (Our SBD). For a more extended comparison, we use both Sum-Pooling and LSTM as temporal aggregation methods.

Results are reported in Table 6.3. Detecting shots is always beneficial, even if the shots are detected randomly, as the number of visual embeddings per video is increased. Moreover, when our SBD is used, the performance of the overall system is improved considerably with respect the Rand. SBD. It also can be seen that the LSTM encoder is superior to sum-pooling aggregation method when shots are considered.

6.2.5 Comparison with State-of-the-Art

Finally, we compare our asymmetric spatio-temporal embeddings with state-of-the-art compact methods in image-to-video retrieval. For a fair comparison, we only report results of compact aggregation methods, that is, methods that encode multiple frames into a single compact vector rather than using multiple local

Method	dim	SI2V-600k	VB-600k
Scene FV* (DoG) [1]	65,536	0.473	-
Scene FV* [2]	65,536	0.500	0.622
Sum-Pool AlexNet FC6 [2]	4,096	0.071	0.012
Sum-Pool AlexNet FC7 [2]	4,096	0.065	0.013
Sum-Pool VGG16 FC6 [2]	4,096	0.067	0.013
Sum-Pool VGG16 FC7 [2]	4,096	0.069	0.011
Spatio-Temporal-LSTM (Ours)	512	0.602	0.580
Spatio-Temporal-GRU (Ours)	512	0.606	0.572

Tab. 6.4: Comparison with state-of-the-art. Results provided as mAP. [2] are implementations from Araujo et al., 2015b and [2] from Araujo and Girod, 2017.

vectors per query or super high-dimensional vectors. Also, as previous work is mostly based on non-trainable architectures, for a fair comparison we keep our spatial encoder with the original pre-trained weights (i.e. no additional training).

Accuracy

Results are detailed in Table 6.4. Our techniques based on LSTM and GRU outperform reported methods in terms of both memory and accuracy. When compared with previous state-of-the-art (Scene FV*), our embeddings are superior on the SI2V-600k dataset and obtain comparable results in VB-600k. This performance is remarkable considering that our methods are using embeddings 128 times smaller than Scene FV* (512 versus 65,536 dimensions).

Computational Cost

In Araujo and Girod, 2017, memory requirements are reported in SI2V-4M and VB-4M datasets, which are larger versions of SI2V-600k and VB-600k with 1080 hours of video. Their simplest baseline with FV* and no temporal aggregation

needed 20.59 gigabytes (GB) of memory to encode the whole dataset, whereas their Scene FV* required 3.01 GB. Their best performing method based on super high-dimensional Bloom Filters (33.5 million binary dimensional vectors) needed 10.76 GB of memory. With our approach, we are able to encode 160 hours of video in SI2V-600k and VB-600k datasets in only 0.15 GB. With a simple conversion, for the larger versions (1080 hours of video), we would need just about 1 GB of memory, which is a compression of 20 times with respect to the baseline, 10 times with respect to the Bloom Filters approach and 3 times with respect to Scene FV*.

6.3 Conclusions

In this chapter, we proposed a model to obtain asymmetric spatio-temporal embeddings for large-scale image-to-video retrieval. We introduced an asymmetric architecture to project images and videos into a common embedding space, so that they can be easily matched with a cosine similarity. We computed image embeddings with a spatial encoder based on convolutional neural networks. Video embeddings were obtained by using a temporal encoder based on recurrent neural networks. We trained our model with relevant pairs of images and videos using a margin loss function.

Experiments in different image-to-video retrieval datasets showed that our spatio-temporal encoder is superior to the state-of-the-art methods both in terms of accuracy and computational cost. In contrast to the method introduced in Chapter 5, this model aggregates the spatio-temporal information in a video shot into a single vector. Query images are also represented by a single vector, which makes the retrieval task easier and faster than when using local features.

Part IV

Cross-Modal Retrieval

7

Semantic Art Understanding with Text-Image Retrieval

In this part, we study cross-modal retrieval, which is the retrieval problem in which non-visual queries are used to search for visual elements in a dataset (see Figure 1.2). In particular, this chapter is focused on text-image retrieval, where images in a visual collection are found according to a textual query, and vice versa. Recently, with the latest advancements in the fields of computer vision (CV) and natural language processing (NLP), text-image retrieval has attracted a lot of attention, especially with natural images and text captions (Wang et al., 2017). In this chapter, however, we address image-text retrieval from a different domain perspective by studying high-level image recognition in art.

We refer to this high-level recognition problem as *semantic art understanding*. An example is shown in Figure 7.1. In contrast to natural image understanding, there is not a wide volume of previous work in understanding images in the domain of art, which turns into a lack of datasets for common and public benchmark. We address this problem by firstly, introducing SemArt, a cross-modal dataset in the art domain; secondly, presenting the Text2Art challenge, a text-image retrieval task for the SemArt dataset; and thirdly, proposing a number of cross-modal retrieval models, which achieves recognition levels close to human evaluation.

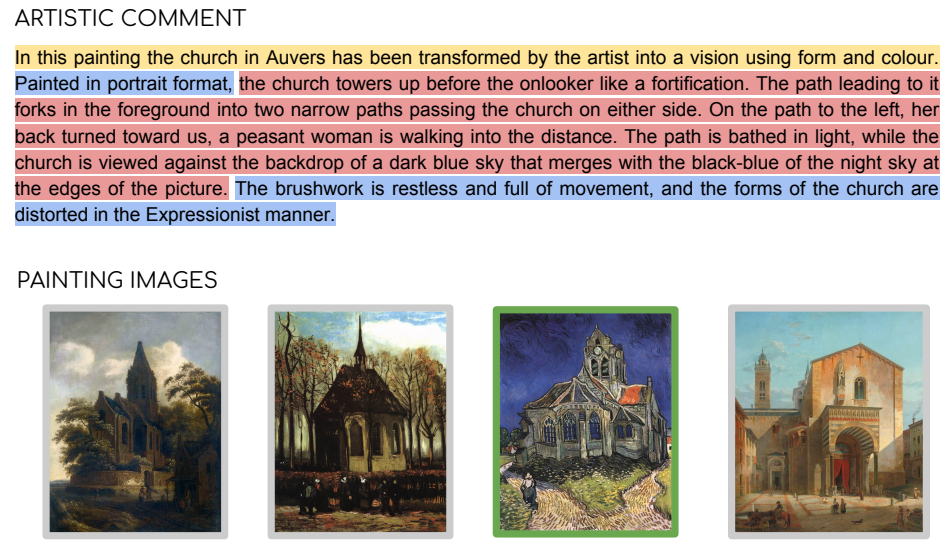


Fig. 7.1: Example of semantic art understanding with text-image retrieval. In the top, artistic comment that describes context (yellow), technique (blue) and content (red) of a painting. In the bottom: painting images relevant to the artistic comment; in green, the painting the comment belongs to.

7.1 Related Work

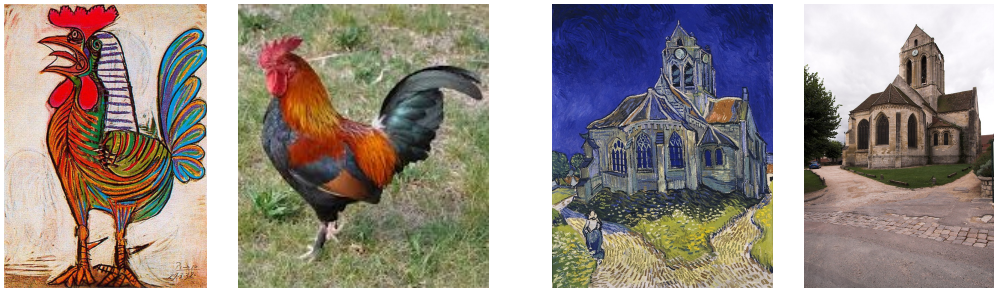
7.1.1 Text-Image Retrieval

Retrieving images from text and vice versa is an active field of research within the computer vision community (Nam et al., 2017; Salvador et al., 2017; Wang et al., 2018). Before the introduction of NLP techniques, early work approached text-image retrieval by applying CBIR methods (Ordonez et al., 2011), that is, computing similarities between a query image and a set of images from a collection, which have been previously associated with textual descriptions. In this way, given a query image, the sentence associated to the most similar image was retrieved. However, performing image-to-image retrieval to find text is very restrictive, as text sentences need to be previously associated to images in order to be searchable.

Nowadays, most text-image retrieval approaches use both CV and NLP tools to project images and sentences into a common latent space where visual and textual representations can be compared in terms of similarity (Wang et al., 2017). One of the first approaches along these lines (Farhadi et al., 2010) represented both images and sentences with a triplet of (object, action, scene). To find image-sentence correspondences, the authors computed similarities between triplets. A drawback of this approach is that it is limited to use a fixed number of objects, actions and scenes to project the data to.

A popular method for projecting data from two different distributions into a common latent space is Canonical Correlation Analysis (CCA, Hotelling, 1936). CCA learns the weights of the projection matrix by maximizing the correlation between the projected vectors of the two data sources. Despite the simplicity of the method, Gong et al., 2014a showed that normalized CCA (Gong et al., 2014b) along with state-of-the-art visual and textual features outperforms more complex methods, such as Wsabie (Weston et al., 2011) in text-image retrieval. Although CCA is a linear method, non-linear projections can be achieved by using kernel CCA (Bach and Jordan, 2002), as in Hodosh et al., 2013. However, CCA methods do not scale well to large-scale collections of images, as the covariance matrix computation rapidly incurs a high memory cost. To overcome this issue, deep CCA (Andrew et al., 2013; Yan and Mikolajczyk, 2015) optimizes the covariance matrix by using deep learning techniques.

As an alternative to CCA, many deep learning models have been proposed (Karpathy et al., 2014; Nam et al., 2017; Salvador et al., 2017; Wang et al., 2018). These methods were based on optimizing a ranking loss function by using matching and non-matching image-sentence pairs or triplets. For example, Karpathy et al., 2014 proposed to find alignments between objects from images and fragments from sentences and compute pairwise distances to estimate a similarity score; Nam et al., 2017 introduced an attention mechanism to focus on specific fragments of images and sentences to gather the essential information from both modalities;



(a) (b)

Fig. 7.2: Paintings versus natural images. Paintings are figurative representations, which commonly exhibit some layers of abstraction and symbolism, whereas natural images represent reality as we see it. (a) The Roaster by Pablo Picasso (1983) versus a photograph of rooster; (b) The Church at Auvers by Vincent van Gogh (1890) versus a photograph of the church.

Salvador et al., 2017 proposed a trijoint model using images, recipes and ingredients to find food images from textual recipes, and vice versa; and Wang et al., 2018 implemented a bi-directional ranking loss by considering the distances from image to text as well as from text to image.

Most of the aforementioned methods implemented systems to find natural images by given a description of the scene. On the contrary, we propose to apply text-image retrieval for semantic art understanding in the domain of fine-art paintings. As already noted in previous work (Crowley and Zisserman, 2014a; Crowley and Zisserman, 2014b; Crowley et al., 2015), paintings are substantially different from natural images in several aspects. As an illustration, Figure 7.2 shows two examples in which the same object or scene is being represented in a painting and in a natural images.

The main differences between studying natural images and art images are threefold. Firstly, paintings, unlike natural images, are figurative representations of people, objects, places or situations which may or may not correspond to the real world. Secondly, the study of fine-art paintings usually requires previous knowledge about history of art, artistic styles as well as contextual information about the subjects represented. Thirdly, paintings commonly exhibit one or more layers of abstraction and symbolism which creates ambiguity in interpretation.

Thus, to perform semantic art understanding, a deep analysis not only about the elements of the image but also about its style, author, influences and context is required.

7.1.2 Semantic Art Understanding

With the digitalization of large collections of fine-art paintings and the emergence of publicly available online art catalogs such as WikiArt[1] or the Web Gallery of Art[2], computer vision researchers became interested in applying computer vision techniques for automatic art understanding. So far, work in automatic art understanding has been mostly focused on painting classification (Carneiro et al., 2012; Mensink and Van Gemert, 2014; Mao et al., 2017) and image retrieval (Carneiro et al., 2012; Seguin et al., 2016) to either detect the style, year or author of a specific fine-art painting or to find relevant paintings according to a query.

Early work (Johnson et al., 2008; Shamir et al., 2010; Carneiro et al., 2012; Khan et al., 2014) proposed methods based on handcrafted visual features to identify an author and/or a specific style in a piece of art. Datasets used in these kinds of approaches, such as PRINTART (Carneiro et al., 2012) and Painting-91 (Khan et al., 2014), were rather small, with 988 and 4,266 painting images, respectively. The larger Rijksmuseum dataset (Mensink and Van Gemert, 2014), containing 112,039 images from artistic objects, was also introduced for multi-class prediction, although only 3,593 of the pictures were from fine-art paintings.

With the success of CNN in large-scale image classification (Krizhevsky et al., 2012), deep features obtained from CNNs replace handcrafted features in many computer vision applications, including painting image classification (Bar et al., 2014; Karayev et al., 2014; Saleh and Elgammal, 2015; Tan et al., 2016; Ma et al., 2017; Mao et al., 2017). In these kinds of classification methods, images from

[1] http://www.wikiart.org
[2] https://www.wga.hu/

Dataset	#Paintings	Meta	Text	Task
PRINTART (Carneiro et al., 2012)	988	✓	✗	Classification, Retrieval
Painting-91 (Khan et al., 2014)	4,266	✓	✗	Classification
Rijksmuseum (Mensink and Van Gemert, 2014)	3,593	✓	✗	Classification
Wikipaintings (Karayev et al., 2014)	85,000	✓	✗	Classification
Paintings (Crowley and Zisserman, 2014a)	8,629	✗	✗	Object Recognition
Face Paintings (Crowley et al., 2015)	14,000	✗	✗	Face Retrieval
VisualLink (Seguin et al., 2016)	38,500	✓	✗	Instance Retrieval
Art500k (Mao et al., 2017)	554,198	✓	✗	Classification
SemArt	21,383	✓	✓	Semantic Retrieval

Tab. 7.1: Datasets for art understanding. Meta and Text columns state whether image metadata and textual information are provided, respectively.

paintings were fed into a CNN to predict its artistic style or author by studying its visual aesthetics. Also, to learn the weights of deep CNN models, large-scale datasets were made publicly available (Karayev et al., 2014; Mao et al., 2017).

Besides painting classification, other work has been focused on exploring search of artistic paintings. In Carneiro et al., 2012, monochromatic painting images were retrieved by using artistic-related keywords, whereas in Seguin et al., 2016 a pre-trained CNN was fine-tuned to find paintings with similar artistic motifs. Crowley et al., 2015 explored domain transfer to retrieve image of portraits from real faces, in the same way as Crowley and Zisserman, 2014a and Crowley and Zisserman, 2016 explored domain transfer to perform object recognition in paintings.

A summary of the existing datasets for fine-art understanding is shown in Table 7.1. In essence, previous work studied art from an aesthetics point of view to classify paintings according to their author or style, to find relevant images according to a query input or to identify objects in artistic representations. However, as shown in Figure 7.1, semantic art understanding involves also other processes, such as identifying relations between elements in the scene, the artistic influences of the author or the historical context of the work. To understand such complex processes we propose to interpret fine-art paintings in a semantic way. To that

end, we collect a cross-modal dataset for semantic art understanding that, unlike previous datasets, not only contains fine-art images and their attributes, but also semantic descriptions.

7.2 SemArt Dataset

In this section, we introduce the SemArt dataset, a cross-modal corpus that provides artistic comments along with fine-art paintings and their attributes for studying semantic art understanding. In order to push the performance of CV algorithms in image recognition and semantic are understanding, we make SemArt dataset publicly available at:

http://noagarciad.com/SemArt/

7.2.1 Data Collection

To create the SemArt dataset, we collect artistic data from the Web Gallery of Art (WGA), a website with more than 44,809 images of European fine-art reproductions between the 8th and the 19th century. WGA provides links to all their images in a downloadable comma separated values file (CSV). In the CSV file, each image is associated with some attributes or metadata: author, author's birth and death, title, date, technique, current location, form, type, school and time-line. Following the links provided in the CSV file, we only collect images from artworks whose field *form* is set as painting, as opposite to images of other forms of art such as sculpture or architecture.

We create a script to collect artistic comments for each painting image, as they are not provided in the aforementioned CSV file. We omit images that are not associated to any comment and we remove irrelevant metadata fields, such as author's birth and death and current location. The final size of the cleaned collection is downsampled to 21,384 triplets, where each triplet is formed by an

Field	Values	Samples	Most Common
Author	3,281	21,383	GOGH, Vincent van
Title	14,902	21,383	Still-Life
Technique	13,403	21,358	Fresco
Date	3,124	18,989	c. 1500
Type	10	21,383	Religious
School	26	21,383	Italian
Timeframe	22	21,383	1601-1650

Tab. 7.2: SemArt metadata details. For each metadata field in the dataset: number of different values, total number of samples and the most common value.

image, a text and a number of attributes. Examples from the SemArt dataset are depicted in Figure 7.3.

7.2.2 Data Analysis

For each sample, metadata is provided as a set of seven fields, which describe the basic attributes of the painting: *Author*, *Title*, *Date*, *Technique*, *Type*, *School* and *Timeframe*. Details about the different fields can be found in Table 7.2.

In total, there are 3,281 different authors, the most frequent one being Vincent van Gogh with 327 paintings. There are 14,902 different titles in the dataset, with 38.8% of the paintings presenting a non-unique title. Among all the titles, Still-Life and Self-Portrait are the most common ones. *Technique* and *Date* fields are not available for all samples, but provided for completeness. *Type* field classifies paintings according to ten different genres, such as religious, landscape or portrait. There are 26 artistic schools in the collection, Italian being the most common, with 8,860 paintings and Finnish the least frequent with just 5 samples. Also, there are 22 different timeframes, which are periods of 50 years evenly distributed between 801 and 1900. The distribution of values over the fields *Type*, *School* and *Timeframe* is shown in Figure 7.4.

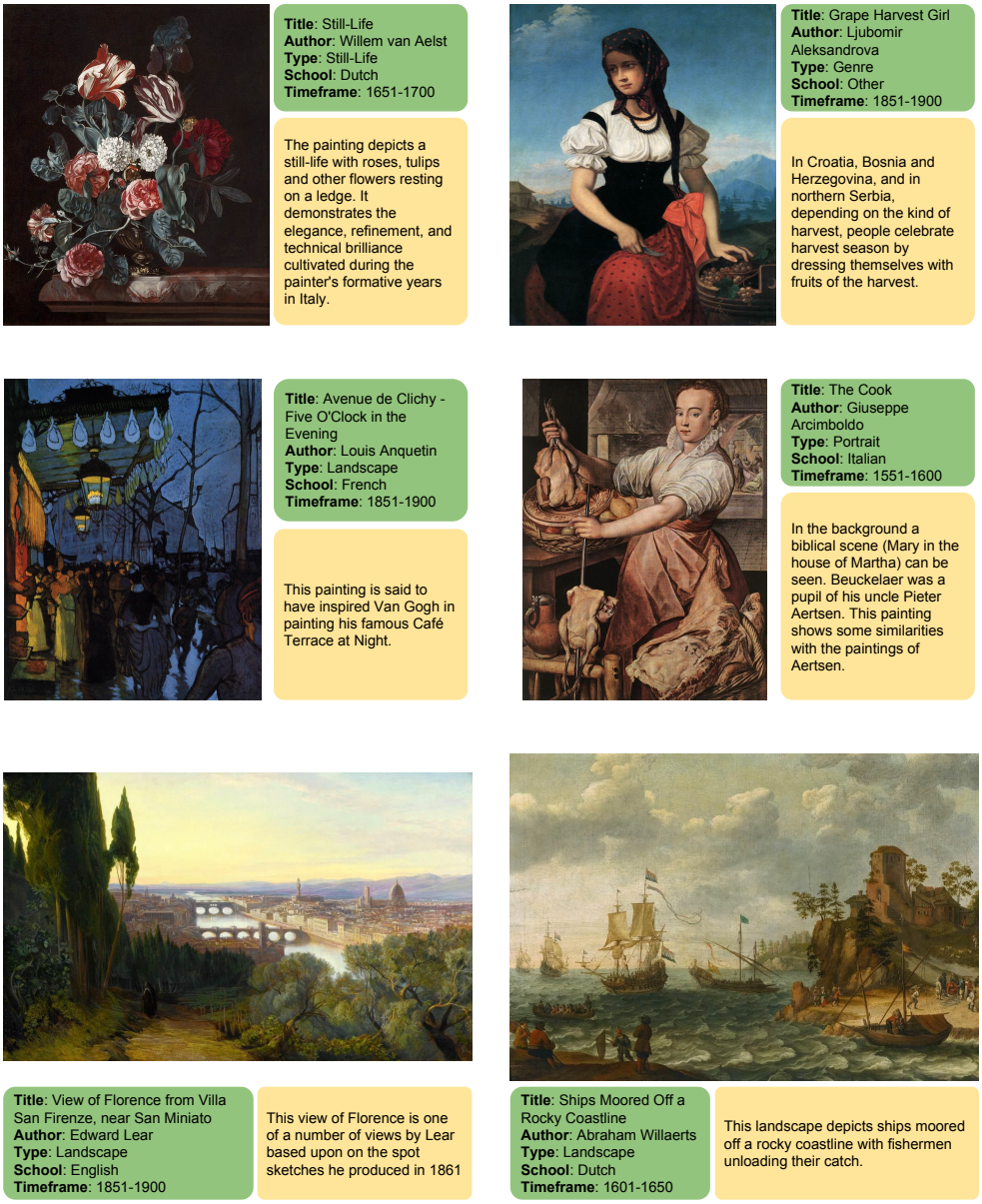
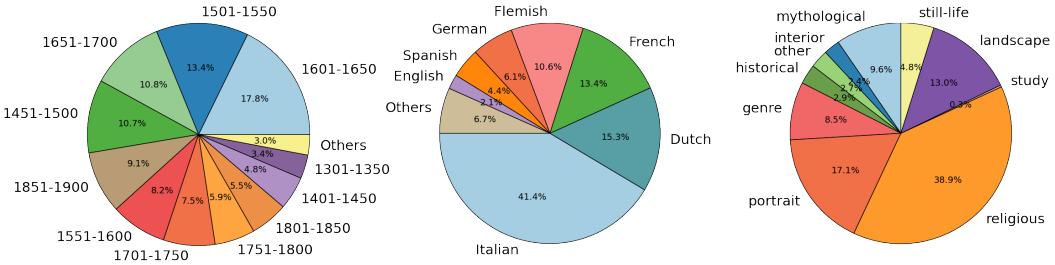


Fig. 7.3: Samples from the SemArt dataset. Each sample is a triplet of (image, comment, attributes).



Timeframe **School** **Type**

Fig. 7.4: Distribution of samples in Timeframe, School and Type attributes.

With respect to the artistic comments, the vocabulary set follows the Zipf's law (Manning et al., 1999). Most of the comments are relatively short, with almost 70% of the them containing 100 words or less. Images are provided in different aspect ratios and sizes. The dataset is randomly split into training, validation and test sets with 19,244, 1,069 and 1,069 triplets, respectively.

7.3 Method

To address semantic art understanding, we first introduce the Text2Art challenge for the SemArt dataset (Section 7.3.1). We then propose a number of models to map paintings and artistic comments into a common semantic space (Section 7.3.2), thus enabling artistic semantic comparisons between images and texts.

7.3.1 Text2Art Challenge

Given a collection of artistic samples K, the k-th sample in K is given by the triplet of (img^k, com^k, att^k), being img^k the artistic image, com^k the artistic comment and att^k the artistic attributes. Images, comments and attributes are input into specific encoding functions, f_{img}, f_{com}, f_{att}, to map raw data from the corpus into vector representations, $\mathbf{i}^k$, $\mathbf{c}^k$, $\mathbf{a}^k$, as:

$$\mathbf{i}^k = f_{img}(img^k; \phi_{img}) \tag{7.1}$$

$$\mathbf{c}^k = f_{com}(com^k; \phi_{com}) \tag{7.2}$$

$$\mathbf{a}^k = f_{att}(att^k; \phi_{att}) \tag{7.3}$$

where ϕ_{img}, ϕ_{com} and ϕ_{att} are the parameters of each encoding function.

As comment encodings, $\mathbf{c}^k$, and attribute encodings, $\mathbf{a}^k$, are both from textual data, a joint textual vector, $\mathbf{t}^k$ can be obtained as:

$$\mathbf{t}^k = \mathbf{c}^k \oplus \mathbf{a}^k \tag{7.4}$$

where $\oplus$ is vector concatenation.

The transformation functions, g_{vis} and g_{text}, can be defined as the functions that project the visual and the textual encodings into a common cross-modal space. The projected vectors $\mathbf{p}^k_{vis}$ and $\mathbf{p}^k_{text}$ are then obtained as:

$$\mathbf{p}^k_{vis} = g_{vis}(\mathbf{i}^k; \theta_{vis}) \tag{7.5}$$

$$\mathbf{p}^k_{text} = g_{text}(\mathbf{t}^k; \theta_{text}) \tag{7.6}$$

being θ_{vis} and θ_{text} the parameters of each transformation function.

For a given similarity function d, the similarity between any text (i.e. pair of comments and attributes) and any image in K is measured as the distance between their projections:

$$d(\mathbf{p}^k_{text}, \mathbf{p}^j_{vis}) = d(g_{text}(\mathbf{t}^k; \theta_{text}), g_{vis}(\mathbf{i}^j; \theta_{vis})) \tag{7.7}$$

In semantic art understanding, the aim is to learn f_{img}, f_{com}, f_{att}, g_{vis} and g_{text} such that images, comments and attributes from the same sample are mapped closer in terms of d than images, texts and attributes from different samples:

$$d(\mathbf{p}^k_{text}, \mathbf{p}^k_{vis}) < d(\mathbf{p}^k_{text}, \mathbf{p}^j_{vis}) \text{ for all } k, j \leq |K| \tag{7.8}$$

and

$$d(\mathbf{p}^k_{text}, \mathbf{p}^k_{vis}) < d(\mathbf{p}^j_{text}, \mathbf{p}^k_{vis}) \text{ for all } k, j \leq |K| \tag{7.9}$$

To evaluate semantic art understanding, we propose the Text2Art challenge as a cross-modal retrieval problem. Within Text2Art, we define two tasks: text-to-image retrieval and image-to-text retrieval. In text-to-image, the aim is to find the most relevant painting image in the collection, $img^* \in K$, given a query comment and its attributes:

$$img^* = \underset{img^j \in K}{\arg\min}\, d(\mathbf{p}^k_{text}, \mathbf{p}^j_{vis}) \tag{7.10}$$

In the image-to-text task, when a painting image is given, the aim is to find the comment and the attributes, $com^* \in K$ and $att^* \in K$, that are most relevant to the visual query:

$$com^*, att^* = \underset{com^j, att^j \in K}{\arg\min}\, d(\mathbf{p}^j_{text}, \mathbf{p}^k_{vis}) \tag{7.11}$$

7.3.2 Models

We propose several models to learn meaningful textual and visual encodings and transformations for semantic art understanding. First, images, comments and attributes are encoded into visual and textual vectors. Then, a multi-modal transformation model is used to map these visual and textual vectors into a multi-modal common space where a similarity function is applied.

Visual Encoding

We represent each painting image as a visual vector, $\mathbf{i}^k$, using CNNs. We use different CNN architectures, such as VGG16 (Simonyan and Zisserman, 2015), different versions of ResNet (He et al., 2016) and RMAC (Tolias et al., 2016).

- **VGG16**: convolutional neural network introduced in Simonyan and Zisserman, 2015 designed for image classification. It contains 13 3x3 convolutional layers and three fully-connected layers stacked on top of each other. We use the output of one of the fully connected layers as the visual encoding.

- **ResNet**: architecture introduced in He et al., 2016. It uses shortcut connections to connect the input of a layer to the output of a deeper layer. In this way, deeper architectures are easier to train. There exist several models depending on the number of layers, such as ResNet50 and ResNet152 with 50 and 152 layers, respectively. We use the output of the last layer as the visual encoding.

- **RMAC**: visual descriptor introduced in Tolias et al., 2016 for image retrieval. The activation map from the last convolutional layer from a CNN model is max-pooled over several regions to obtain a set of regional features. The regional features are normalized, PCA-whitened and normalized, sum-up together and normalized once again to obtain the final visual representation.

Textual Encoding

With respect to the textual information, comments are encoded into a comment vector, $\mathbf{c}^k$, and attributes are encoded into an attribute vector, $\mathbf{a}^k$. To get the joint textual encoding, $\mathbf{t}_k$, both vectors are concatenated.

Comment Encoding: To encode comments into a comment vector, $\mathbf{c}^k$, we first build a comment vocabulary, V_C. V_C contains all the alphabetic words that appear at least ten times in the training set. The comment vector is obtained using three different techniques: a comment bag-of-words (BOW$_c$), a comment multi-layer perceptron model (MLP$_c$) and a comment recurrent model (LSTM$_c$).

- **BOW$_c$**: each comment is encoded as a one-hot frequency-inverse document frequency (tf-idf) weighted vector. The tf-idf algorithm weights each word in the comment by its relevance within the corpus.

- **MLP$_c$**: each comment is encoded as a one-hot tf-idf-weighted vector. The one-hot vector is then fed into a fully connected layer with a tanh activation function[3] and a normalization layer. The output of the normalization layer is used as the comment vector.

- **LSTM$_c$**: each sentence in a comment is encoded into a sentence vector using a $2,400$ dimensional pre-trained skip-thought model (Kiros et al., 2015). Sentence vectors are input into a long short-term memory network (LSTM, Hochreiter and Schmidhuber, 1997) and the last state of the LSTM is normalized and used as the comment vector.

Attribute Encoding: We use the attribute field *Title* to augment the textual content in our model. We propose three different techniques to encode a title into an attribute vector representation, $\mathbf{a}^k$: an attribute bag-of-words (BOW$_a$) an attribute multi-layer perceptron (MLP$_a$) and an attribute recurrent model (LSTM$_a$).

- **BOW$_a$**: as in comments, titles is encoded as a one-hot tf-idf-weighted vector using a title vocabulary, V_T. V_T is built with all the alphabetic words in the titles of the training set.

[3] $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

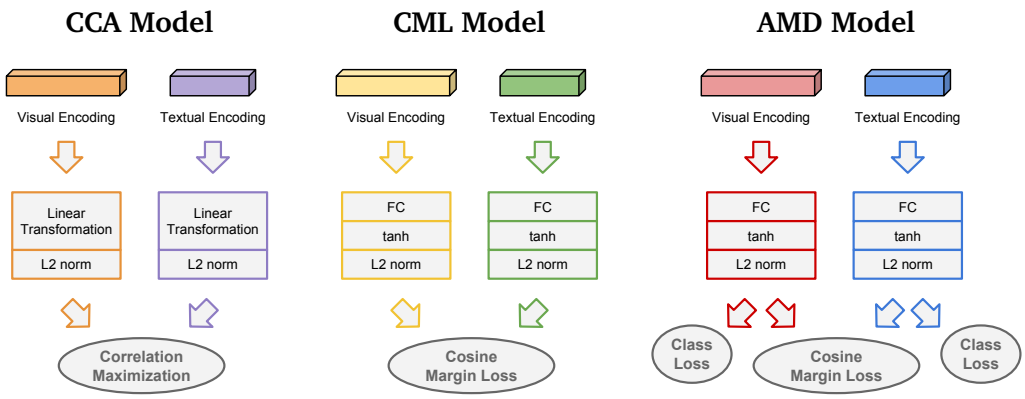


Fig. 7.5: Cross-modal transformation models for mapping textual and visual representations into a common semantic space.

- **MLPa**: also as in comments, one-hot encoded titles are fed into a fully connected layer with a tanh activation and a normalization. The output of the normalization layer is used as the attribute vector.

- **LSTMa**: in this case, each word in a title is fed into an embedding layer followed by a LSTM network. The output of the last state of the LSTM is normalized and used as the attribute vector.

Cross-Modal Transformation

The visual and textual encodings, $\mathbf{i}^k$ and $\mathbf{t}^k$ respectively, encode visual and textual data into two different spaces. We use a cross-modal transformation model to map the visual and textual representations into a common semantic space. In this common space, textual and visual information can be compared in terms of the semantic similarity with the function d. We propose three different models, which are illustrated in Figure 7.5.

- **CCA**: Normalized Canonical Correlation Analysis (CCA, Gong et al., 2014b) is a linear approach for projecting data from two different sources into a common space by maximizing the normalized correlation between the projected data. The CCA projection matrices are learnt by using training

pairs of samples from our corpus. At test time, the textual and visual encodings from a test sample are projected using these CCA matrices.

- **CML**: in the Cosine Margin Loss model (CML) a deep learning architecture is trained end-to-end to learn the visual and textual encodings and their projections all at once. In this model, each image encoding is fed into a fully connected layer followed by a tanh activation function and a normalization layer to project the visual feature, $\mathbf{i}^j$, into a D-dimensional space, obtaining the projected visual vector $\mathbf{p}_{vis}^j$. Similarly, each textual vector $\mathbf{t}^k$, is input into another network with identical layer structure (fully connected layer with tanh activation and normalization) to map the textual feature into the same D-dimensional space, obtaining the projected textual vector $\mathbf{p}_{text}^k$. We train the CML model with both positive ($k = j$) and negative ($k \neq j$) pairs of textual and visual data. We use the cosine similarity with maring as the loss function:

$$L_{CML}(\mathbf{p}_k^{vis}, \mathbf{p}_j^{text}) = \begin{cases} 1 - \cos(\mathbf{p}_k^{vis}, \mathbf{p}_j^{text}), & \text{if } k = j \\ \max(0, \cos(\mathbf{p}_k^{vis}, \mathbf{p}_j^{text}) - m), & \text{if } k \neq j \end{cases} \tag{7.12}$$

  where cos is the cosine similarity between two normalized vectors and m is the margin hyperparameter.

- **AMD**: in the Augmented Metadata model (AMD) the network is informed with attribute data for an extra alignment between the visual and the textual encodings in the artistic domain. The AMD model consists on a deep learning architecture that projects both visual and textual vectors into the common semantic space. As in the CML model, image and textual encodings are transformed into D-dimensional vectors using fully connected layers and the loss between the cross-modal transformations is computed using a cosine margin loss. The extra metadata information is used to train a pair of classifiers, as shown in Figure 7.5: AMD Model. Each classifier consists

of a fully connected layer without activation and is trained using a standard cross entropy classification loss function:

$$L_{META}(\mathbf{x}, class) = -\log\left(\frac{\exp(\mathbf{x}[class])}{\sum_j \exp(\mathbf{x}[j])}\right) \tag{7.13}$$

The AMD classifiers contribute to the total loss of the model in addition to the cosine margin loss. The total loss of the model is then computed as:

$$\begin{aligned} L_{AMD}(\mathbf{p}^k_{text}, \mathbf{p}^j_{vis}, l_{p^k_{text}}, l_{p^j_{vis}}) = (1-2\alpha)L_{CML}(\mathbf{p}^k_{text}, \mathbf{p}^j_{vis}) \\ +\alpha L_{META}(\mathbf{p}^k_{text}, l_{p^k_{text}}) \\ +\alpha L_{META}(\mathbf{p}^j_{vis}, l_{p^j_{vis}}) \end{aligned} \tag{7.14}$$

where $l_{p^k_{text}}$ and $l_{p^j_{vis}}$ are the class labels of the k-th text and the j-th image, respectively, and α is the weight of the classifier loss.

7.4 Evaluation

We now evaluate the proposed models in the Text2Art challenge. We perform independent experiments for each of the three main modules, i.e. visual encoding, textual encoding and cross-modal transformation. We also conduct a human evaluation in the SemArt dataset as a benchmark for future research and to compare the proposed algorithms against human performance.

7.4.1 Implementation Details

In the visual encoding part, each network is initialized with its standard pre-trained weights for image classification. Images are scaled down to 256 pixels per side and randomly cropped into 224 $\times$ 224 patches. Visual data is augmented by randomly flipping images horizontally.

In the textual encoding part, the dimensionality of the LSTM hidden state for comments is 1,024, whereas in the LSTM for titles is 300. The title vocabulary size is 9,092. Skip thoughts dimensionality is set to 2,400.

For the deep learning architectures, we use Adam optimizer and the learning rate is set to 0.0001, m to 0.1 and α to 0.01. Training is conducted in mini batches of 32 samples. Cosine similarity is used as the distance function d in all of our models.

In the Text2Art challenge, painting images are ranked according to their similarity to a given text, and vice versa. The ranking is computed on the whole set of test samples and results are reported as median rank (MR) and recall rate at K (R@K), with K being 1, 5 and 10. MR is the value separating the higher half of the relevant ranking position amount all samples, so the lower the better. Recall at rate K is the rate of samples for which its relevant image is in the top K positions of the ranking, so the higher the better.

7.4.2 Results Analysis

Visual Encoding

We first evaluate the transferability of visual features from the natural image domain to the artistic domain. In this experiment, texts are encoded with the BOW$_c$ approach with $V_C = 3,000$. As cross-modal transformation model, a 128-dimensional CCA is used. We extract visual encodings from networks pre-trained for classification of natural images without further fine-tunning or refinement.

For the VGG16 model, we extract features from the first, second and third fully connected layer (VGG16$_{FC1}$, VGG16$_{FC2}$ and VGG16$_{FC3}$). For the ResNet models, we consider the visual features from the output of the networks (ResNet50

and ResNet152). Finally, RMAC representation is computed using a VGG16, a ResNet50 and a ResNet152 ($RMAC_{VGG16}$, $RMAC_{Res50}$ and $RMAC_{Res152}$).

Results are detailed in Table 7.3. As semantic art understanding is a high-level task, it is expected that representations acquired from deeper layers perform better (Girshick et al., 2014), as in the VGG16 models, where the deepest layer of the network obtains the best performance. RMAC features respond well when transferring from natural images to art, although ResNet models obtain the best performance. Considering these results, we use ResNets as visual encoders in the following experiments.

Encoding		Text-to-Image				Image-to-Text			
Img	**Dim**	**R@1**	**R@5**	**R@10**	**MR**	**R@1**	**R@5**	**R@10**	**MR**
VGG16 FC1	4,096	0.069	0.129	0.174	115	0.061	0.129	0.180	121
VGG16 FC2	4,096	0.051	0.097	0.109	278	0.051	0.085	0.103	275
VGG16 FC3	1,000	0.101	0.211	0.285	44	0.094	0.217	0.283	51
ResNet50	1,000	0.114	0.231	0.304	42	0.114	0.242	0.318	44
ResNet152	1,000	0.108	**0.254**	**0.343**	**36**	**0.118**	**0.250**	**0.321**	**36**
RMAC VGG16	512	0.092	0.206	0.286	41	0.084	0.202	0.293	44
RMAC Res50	2,048	0.084	0.202	0.293	48	0.097	0.215	0.288	49
RMAC Res152	2,048	**0.115**	0.233	0.306	44	0.103	0.238	0.305	44

Tab. 7.3: Visual encoding results. Transferability of visual features from natural image classification to the Text2Art Challenge.

Textual Encoding

We then compare the performance between the different text encoding models. In this experiment, images are encoded with a ResNet50 network and the CML model is used to learn the mapping of the visual and the textual encodings into a common 128-dimensional space.

The different encoding methods are comapred in Table 7.4. The best performance is obtained when using the simple bag-of-words approach both for comments and attributes (BOW_c and BOW_a), although the multi-layer perceptron model (MLP_c

Encoding		Text-to-Image				Image-to-Text			
Com	**Att**	**R@1**	**R@5**	**R@10**	**MR**	**R@1**	**R@5**	**R@10**	**MR**
LSTMc	LSTMa	0.053	0.162	0.256	33	0.053	0.180	0.268	33
MLPc	LSTMa	0.089	0.260	0.376	21	0.093	0.249	0.363	21
MLPc	MLPa	0.137	0.306	0.432	16	**0.140**	0.317	0.436	15
BOWc	BOWa	**0.144**	**0.332**	**0.454**	**14**	0.138	**0.327**	**0.457**	**14**

Tab. 7.4: Comparison between different text encodings in the Text2Art Challenge.

and MLP$_a$) obtain similar results. Models based on recurrent networks (LSTM$_c$ and LSTM$_a$) are not able to capture the insights of semantic art understanding. These results are consistent with Wang et al., 2018 work, which shows that text recurrent models perform worse than non-recurrent methods for cross-modal tasks that do not require text generation.

Cross-Modal Transformation

Finally, we compare the three proposed cross-modal transformation models: CCA, CML and AMD. For the AMD approach, we use four different attributes to inform the model: Type (AMD$_T$), TimeFrame (AMD$_{TF}$), School (AMD$_S$) and Author (AMD$_A$). ResNet50 is used to encode visual features.

Results are shown in Table 7.5. Random ranking results are provided for reference. Overall, the best performance is achieved with the CML model with the textual bag-of-words encodings. CCA achieves the worst results among all models, which suggests that linear transformations are not able to adjust properly to the task. Adding extra information to the CML model does not lead to further improvement, and in some cases (AMD$_{TF}$) it may even be harmful. We suspect that this might be due to a lack of enough samples in some of the attributes of the dataset.

Some qualitative results, both positive and negative, are shown in Figures 7.6 and 7.7, respectively. The rankings are computed with the CML model with ResNet50

Technique			Text-to-Image				Image-to-Text			
Model	**Com**	**Att**	**R@1**	**R@5**	**R@10**	**MR**	**R@1**	**R@5**	**R@10**	**MR**
Random	-	-	0.0008	0.004	0.009	539	0.0008	0.004	0.009	539
CCA	MLP_c	MLP_a	0.117	0.283	0.377	25	0.131	0.279	0.355	26
CML	BOW_c	BOW_a	**0.144**	**0.332**	**0.454**	**14**	0.138	**0.327**	**0.457**	**14**
CML	MLP_c	MLP_a	0.137	0.306	0.432	16	**0.140**	0.317	0.436	15
AMD_T	MLP_c	MLP_a	0.114	0.304	0.398	17	0.125	0.280	0.398	16
AMD_{TF}	MLP_c	MLP_a	0.117	0.297	0.389	20	0.123	0.298	0.413	17
AMD_S	MLP_c	MLP_a	0.103	0.283	0.401	19	0.118	0.298	0.423	16
AMD_A	MLP_c	MLP_a	0.131	0.303	0.418	17	0.120	0.302	0.428	16

Tab. 7.5: Comparison between cross-modal transformation models in the Text2Art Challenge.

as visual encoding and bag-of-words as text encoding. In the positive examples, not only the ground truth painting is ranked within the top five paintings in the evaluation collection, but also all the images within the top five are semantically similar to the query text. In the unsuccessful examples, although the ground truth image is not ranked in the top positions of the list, the algorithm returns images that are semantically meaningful to fragments of the content contained in the text, which indicates how challenging the task is.

7.4.3 Human Evaluation

We design a task in Amazon Mechanical Turk[4] for evaluating human performance in the Text2Art challenge. For a given artistic text, which includes comment, title, author, type, school and timeframe, standard human evaluators (i.e. not art experts) are asked to choose the most appropriate painting from a pool of ten images.

The task has two different levels: *easy*, in which the pool of images is chosen randomly from all the paintings in test set, and *difficult*, in which the ten images in the pool share the same type (i.e. portraits, religious, landscapes, etc.). For

[4]https://www.mturk.com/

each level, evaluators are asked to perform the task in 100 artistic texts. Accuracy is measured as the ratio of correct answers over the total number of answers.

Results are shown in Tables 7.6 and 7.7. Human performance is considerably high, reaching a total accuracy of 88.9% in the easiest set of images. Even for humans, there is a drop in performance between the easy and the difficult level, mostly because images from the same type contain more similar artistic comments than images from different types.

We evaluate two models in the same data split as humans, a CCA model and a CML model. The CML model based on bag-of-words and ResNet50 encodings is able to correctly find the relevant image in the 75% of the samples in the easy set and in the 62% of the cases in the difficult task. There is only ten points of difference between CML model and human evaluation. This suggests that, although there is still room for improvement, meaningful representations for semantic art understanding are being obtained from the proposed models.

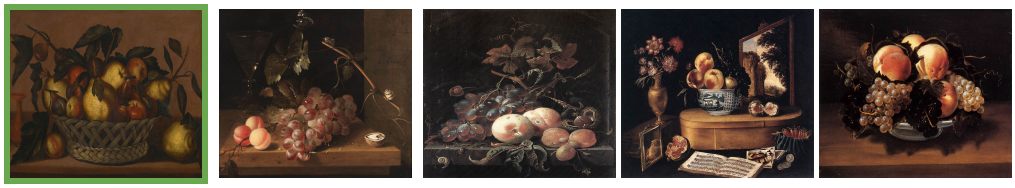
Technique				**Text-to-Image**					
Model	**Img**	**Com**	**Att**	**Land**	**Relig**	**Myth**	**Genre**	**Port**	**Total**
CCA	ResNet152	MLP_c	MLP_a	0.708	0.609	0.571	0.714	0.615	0.650
CML	ResNet50	BOW_c	BOW_a	0.917	0.683	0.714	1	0.538	0.750
Human	-	-	-	0.918	0.795	0.864	1	1	0.889

Tab. 7.6: Models and human evaluation in the easy set.

Technique				**Text-to-Image**					
Model	**Img**	**Com**	**Att**	**Land**	**Relig**	**Myth**	**Genre**	**Port**	**Total**
CCA	ResNet152	MLP_c	MLP_a	0.600	0.525	0.400	0.300	0.400	0.470
CML	ResNet50	BOW_c	BOW_a	0.500	0.875	0.600	0.200	0.500	0.620
Human	-	-	-	0.579	0.744	0.714	0.720	0.674	0.714

Tab. 7.7: Models and human evaluation in the difficult set.

Title: Still-Life of Apples, Pears and Figs in a Wicker Basket on a Stone Ledge
Comment: The large dark vine leaves and fruit are back-lit and are sharply silhouetted against the luminous background, to quite dramatic effect. Ponce's use of this effect strongly indicates the indirect influence of Caravaggio's Basket of Fruit in the Pinacoteca Ambrosiana, Milan, almost 50 years after it was created.



0.778 0.772 0.767 0.754 0.754

Title: A Saddled Race Horse Tied to a Fence
Comment: Horace Vernet enjoyed royal patronage, one of his earliest commissions was a group of ten paintings depicting Napoleon's horses. These works reveal his indebtedness to the English tradition of horse painting. The present painting was commissioned in Paris in 1828 by Jean Georges Schickler, a member of a German based banking family, who had a passion for horse racing.

0.755 0.732 0.718 0.662 0.660

Title: Portrait of a Girl
Comment: This painting shows a girl in a yellow dress holding a bouquet of flowers. It is a typical portrait of the artist showing the influence of his teacher, Agnolo Bronzino.

0.870 0.848 0.847 0.827 0.825

Title: The Kreuzkirche in Dresden
Comment: A few years later, during his second stay in Saxony, Bellotto depicted the demolition of this Gothic church. There exists an almost identical version in the Gemldegalerie, Dresden.

0.841 0.834 0.803 0.800 0.799

Fig. 7.6: Qualitative positive results. For each text (i.e. title and comment), the top five ranked images, along with their score, are shown. The ground truth image is in green.

Title: Brunette with Bare Breasts
Comment: The 1870s were rich in female models for Manet: the Brunette with Bare Breasts, the Blonde with Bare Breasts and the Sultana testify to it.

rank 28, 0.445

0.640 0.622 0.605 0.572 0.569

Title: Battle of the Gabbard
Comment: The naval Battle of the Gabbard, also known as the Battle of Gabbard Bank, the Battle of the North Foreland or the second Battle of Nieuwpoort took place on 12-13 June 1653 during the First Anglo-Dutch War near the Gabbard shoal off the coast of Suffolk, England between fleets of the Commonwealth of England and the United Provinces. In Dutch the battle is known as the Zeeslag bij Nieuwpoort.The picture shows the Dutch flagship Brederode, right, in action ...

rank 38, 0.486

0.756 0.720 0.680 0.660 0.646

Title: Virgin and Child with the Young St John the Baptist
Comment: The stylistic characteristics of this painting, such as rounded faces and narrow, elongated eyes seem to be a general reflection of the foreign presence in Genoese painting at this time.

rank 17, 0.690

0.754 0.751 0.730 0.727 0.721

Fig. 7.7: Qualitative negative results. For each text, the ground truth image is shown next to it, along with its ranking position and its score. The top five ranked images are shown with their ranking scores.

7.5 Conclusions

In this chapter, we addressed cross-modal retrieval by studying semantic art understanding with text-image retrieval.

We first introduced the SemArt dataset, the first corpus of fine-art paintings for semantic art understanding. In SemArt, each image is provided with metadata information and an artistic comment. Artistic comments can describe any artistic information in the painting, such as its content, techniques or context. We also designed the Text2Art challenge based on text-image retrieval to evaluate the performance of semantic art understanding, whereby given an artistic text, a relevant image is found, and vice versa.

Additionally, we proposed several models for semantic art understanding and we compared their performance in the Text2Art challenge. We showed that for visual encoding, ResNet is the model that performs the best. For textual encoding, recurrent models perform worse than a multi-layer preceptron or a simple bag-of-words. We also studied models for projecting the visual and textual encodings into a common semantic space. We obtained the best results with a neural network trained with a cosine margin loss.

Finally, we conducted experiments to compare machine performance against standard human evaluators in two different levels of difficulty. Current approaches are not able to reach human levels of art understanding yet, although we showed that these algorithms are learning meaningful representations for semantic art understanding. We made the SemArt dataset publicly available to encourage future research in the field.

Part V

Conclusion and Final Remarks

8 Conclusion

The aim of this dissertation was to study visual retrieval in its different modalities. In particular, we study three main categories of visual retrieval: (1) symmetric visual retrieval, in which queries and elements in the collection are from the same type of visual data (Chapter 3); (2) asymmetric visual retrieval, in which queries and elements in the collection are from different type of visual data (Chapters 4, 5 and 6); and (3) cross-modal retrieval, in which queries and elements in the collection are from different data types, including visual and non-visual objects (Chapter 7). Here, we conclude the dissertation by outlining our main contributions and identifying future lines of research.

8.1 Contributions

Symmetric Visual Retrieval

In Part II, we addressed asymmetric visual retrieval by studying content-based image retrieval. In standard image retrieval methods, visual features are used to represent images with either hand-crafted or deep learning approaches. In these methods, visual similarity between a pair of images is estimated with a standard metric function between their visual features. However, standard metrics are rigid and limited. To overcome some of their limitations and to fit the data distribution in a better way, we trained a neural network to estimate similarity scores on top of visual representations.

Experiments conducted in Chapter 3 showed that our method was able to capture visual similarity better than other techniques, mostly because of its non-metric nature. We also introduced a real end-to-end trainable architecture for image retrieval, which takes image pixels as input and, after processing the information through all the layers of the model, returns a similarity score as output.

In summary, with respect to symmetric visual retrieval, we showed that the use of a similarity network can push performance in image retrieval systems on top of high-quality image features, by only training the last few layers of the model and without fine-tunning the whole deeper architecture.

Asymmetric Visual Retrieval

In Part III, asymmetric visual retrieval was studied with a focus on image-to-video retrieval. In image-to-video retrieval, static visual features from query images are used to find a relevant scene or timestamp in a video collection. As videos contain spatio-temporal visual features, asymmetric techniques for processing videos and images are required.

In Chapter 4, we introduced a dataset for image-to-video retrieval, the MoviesDB, with the largest number of query images in an image-to-video retrieval collection until now. We also introduced a framework to find fashion products in videos based on image-to-video retrieval. We addressed image-to-video retrieval from two different perspectives. In Chapter 5, we proposed to aggregate recurrent local features in videos to reduce the amount of data to be processed whereas in Chapter 6, we encoded global spatio-temporal information using convolutional and recurrent neural networks.

Local temporal aggregation methods for image-to-video retrieval commonly obtain better performance than global temporal aggregation methods at the expense of an increased search time and more memory storage. In Chapter 5, we

overcame these challenges by proposing a solution to efficiently reduce memory requirements by aggregating binary features and indexing them using kd-trees.

Despite these improvements, the approach proposed in Chapter 5 needed to perform as many searches as visual features found in the query image in order to retrieve a scene from the video collection. In Chapter 6, in contrast, we presented a global temporal aggregation system, in which only one search per query image was performed. This system was based on an asymmetric architecture to project images and videos into a common embedding space. Experiments conducted in Chapter 6 showed that our spatio-temporal encoder was superior to other global temporal aggregation methods based on hand-crafted features.

In summary, we introduced two different methods for aggregating temporal information in videos to perform image-to-video retrieval. The first one was based on the aggregation of local binary features, which obtained high accuracy rates but performed multiple searches per query image. The second method was based on deep learning techniques to aggregate global spatio-temporal features into a single vector representation per video segment, which simplified the search process and outperformed previous global aggregation methods.

Cross-Modal Retrieval

In Part IV, we addressed cross-modal retrieval in the specific domain of semantic art understanding. Cross-modal retrieval, especially text-image retrieval, has been widely studied in the domain of natural images and text captions. However, in Chapter 7, we proposed cross-modal and text-image retrieval techniques for art understanding.

We introduced the SemArt dataset, the first cross-modal corpus for semantic art understanding with artistic images, attributes and artistic comments. We also

introduced the Text2Art challenge based on text-image retrieval, where a query comment was used to find its relevant image and vice versa.

To address cross-modal retrieval in art understanding, we proposed and evaluated several models for encoding paintings, texts and for projecting cross-modal data into a common semantic space. We also conducted experiments with human evaluators to compare machine versus human performance. We showed that current approaches are not far from human accuracy and they extract meaningful representations from the artistic cross-modal dataset.

8.2 Future Work

This work was focused on exploring multi-modal visual retrieval problems, especially symmetric, asymmetric and cross-modal retrieval. The outcome of this research can be further explored in future work both to study novel applications and to improve current performance.

8.2.1 Similarity Networks

In this work, we were able to show consistent improvements in content-based image retrieval datasets through the use of our proposed similarity networks. These results could encourage future research on similarity networks to create more accurate and generalizable models. Some topics that could be explored in future research are:

1. *Domain adaptation*. As similarity estimation is a problem-dependent task, similarity networks do not transfer well across different domains. Future research could study novel approaches to minimize the impact of this phenomenon via transfer learning, domain adaptation and zero-shot learning mechanisms.

2. *Task generalization*. Similarity estimation between a pair of objects is a common task within different visual retrieval modalities. Future work could study the adaptation of these kinds of networks to other retrieval tasks such as video-to-video retrieval, image-to-video retrieval or text-image retrieval.

8.2.2 Asymmetric Visual Retrieval

In asymmetric visual retrieval, we studied temporal feature aggregation through image-to-video retrieval. In future research, temporal aggregation methods could address the following topics:

1. *Processing Time*. The temporal aggregation methods proposed in this work are able to successfully reduce memory requirements in large-scale datasets. However, the architectures used to aggregate redundant visual features are not cheap in terms of processing time. Future research could study how to reduce computation time to create fast and efficient compact visual features for asymmetric visual retrieval.

2. *Task transfer*. Our proposed temporal aggregation methods were specially designed for image-to-video retrieval datasets. It would be interesting to explore further adaptation of these methods to other asymmetric visual retrieval tasks, such as video-to-image retrieval in video content augmentation applications.

8.2.3 Semantic Art Understanding

Research conducted in this work showed encouraging results in the field of semantic art understanding. Although the performance obtained so far is still below human accuracy, future research could explore how to bridge the gap between machine and human evaluation. Some interesting directions towards that end are:

1. *Style versus content analysis*. Identifying the visual features that represent the style of a painting versus the ones that correspond to the content of the image has been shown to be helpful in art classification tasks (Mao et al., 2017). However, it remains as an open question if the introduction such kinds of mechanisms is also beneficial for semantic art understanding.

2. *Knowledge integration*. Paintings are strongly related to their historical, social and artistic context. Providing such information to a model could potentially improve the understanding of a specific artwork through its related masterpieces. However, the acquisition and use of this kind of knowledge is non-trivial. Future work could study the use novel mechanisms to find, extract and introduce human knowledge into the study of art.

3. *Additional tasks*. For a better understanding of how algorithms study art, it would be interesting to evaluate semantic art understanding through additional cross-modal tasks, such as automatic title generation or visual question answering. Future research could adapt and expand the proposed SemArt dataset for a wider and common framework in semantic art understanding evaluation.

Bibliography

Aly, Mohamed, Mario Munich, and Pietro Perona (2011). "Distributed kd-trees for retrieval from very large image collections". In: *Proceedings of the British Machine Vision Conference*. Vol. 17 (cit. on pp. 81, 85).

Andrew, Galen, Raman Arora, Jeff Bilmes, and Karen Livescu (2013). "Deep canonical correlation analysis". In: *International Conference on Machine Learning*, pp. 1247–1255 (cit. on p. 105).

Anjulan, Arasanathan and Nishan Canagarajah (2007). "Object based video retrieval with local region tracking". In: *Signal Processing: Image Communication* 22.7-8, pp. 607–621 (cit. on p. 64).

Araujo, A, Mina Makar, Vijay Chandrasekhar, et al. (2014). "Efficient video search using image queries". In: *International Conference on Image Processing*. IEEE, pp. 3082–3086 (cit. on pp. 64, 68).

Araujo, Andre and Bernd Girod (2017). "Large-scale video retrieval using image queries". In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.6, pp. 1406–1420 (cit. on pp. 15, 65, 97, 100).

Araujo, André, Jason Chaves, David Chen, Roland Angst, and Bernd Girod (2015a). "Stanford I2V: a news video dataset for query-by-image experiments". In: *ACM Multimedia Systems Conference*. ACM, pp. 237–242 (cit. on pp. 68, 75, 96).

Araujo, André, Jason Chaves, Roland Angst, and Bernd Girod (2015b). "Temporal aggregation for large-scale query-by-image video retrieval". In: *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, pp. 1519–1522 (cit. on p. 100).

Araujo, André, Jason Chaves, Haricharan Lakshman, Roland Angst, and Bernd Girod (2016). "Large-scale query-by-image video retrieval using bloom filters". In: *arXiv preprint arXiv:1604.07939* (cit. on pp. 68, 96).

Aytar, Yusuf, Carl Vondrick, and Antonio Torralba (2017). "See, hear, and read: Deep aligned representations". In: *arXiv preprint arXiv:1706.00932* (cit. on p. 16).

Babenko, Artem and Victor Lempitsky (2015). "Aggregating local deep features for image retrieval". In: *IEEE International Conference on Computer Vision*, pp. 1269–1277 (cit. on pp. 42, 59).

Babenko, Artem, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky (2014). "Neural codes for image retrieval". In: *European Conference on Computer Vision*. Springer, pp. 584–599 (cit. on pp. 17, 31, 32, 40–42, 46, 50, 59, 60).

Bach, Francis R and Michael I Jordan (2002). “Kernel independent component analysis”. In: *Journal of machine learning research* 3.Jul, pp. 1–48 (cit. on p. 105).

Bar, Yaniv, Noga Levy, and Lior Wolf (2014). “Classification of artistic styles using binarized features derived from a deep neural network”. In: *Workshop at the European Conference on Computer Vision*. Springer, pp. 71–84 (cit. on p. 107).

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006). “Surf: Speeded up robust features”. In: *European Conference on Computer Vision*. Springer, pp. 404–417 (cit. on pp. 17, 26, 40, 63, 72).

Bell, Sean and Kavita Bala (2015). “Learning visual similarity for product design with convolutional neural networks”. In: *ACM Transactions on Graphics* 34.4, p. 98 (cit. on p. 43).

Bentley, Jon Louis (1975). “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9, pp. 509–517 (cit. on p. 81).

Bloom, Burton H (1970). “Space/time trade-offs in hash coding with allowable errors”. In: *Communications of the ACM* 13.7, pp. 422–426 (cit. on p. 65).

Calonder, Michael, Vincent Lepetit, Christoph Strecha, and Pascal Fua (2010). “Brief: Binary robust independent elementary features”. In: *European Conference on Computer Vision*. Springer, pp. 778–792 (cit. on pp. 26, 79, 84).

Carneiro, Gustavo, Nuno Pinho da Silva, Alessio Del Bue, and João Paulo Costeira (2012). “Artistic image classification: An analysis on the printart database”. In: *European Conference on Computer Vision*. Springer, pp. 143–157 (cit. on pp. 107, 108).

Chang, S-K and Arding Hsu (1992). “Image information systems: where do we go from here?” In: *IEEE Transactions on Knowledge and Data Engineering* 4.5, pp. 431–442 (cit. on p. 16).

Chechik, Gal, Varun Sharma, Uri Shalit, and Samy Bengio (2010). “Large scale online learning of image similarity through ranking”. In: *Journal of Machine Learning Research* 11.Mar, pp. 1109–1135 (cit. on pp. 35, 43, 53).

Chen, David, Ngai-Man Cheung, Sam Tsai, et al. (2010). “Dynamic selection of a feature-rich query frame for mobile video retrieval”. In: *IEEE International Conference on Image Processing*. IEEE, pp. 1017–1020 (cit. on pp. 63, 75).

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, et al. (2014). “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734 (cit. on pp. 92, 98).

Chopra, Sumit, Raia Hadsell, and Yann LeCun (2005). “Learning a similarity metric discriminatively, with application to face verification”. In: *Computer Vision and Pattern Recognition*. Vol. 1. IEEE, pp. 539–546 (cit. on pp. 43, 47).

Crow, Franklin C (1984). “Summed-area tables for texture mapping”. In: *ACM SIGGRAPH Computer Graphics*. Vol. 18. 3. ACM, pp. 207–212 (cit. on p. 26).

Crowley, Elliot and Andrew Zisserman (2014a). “The State of the Art: Object Retrieval in Paintings using Discriminative Regions”. In: *British Machine Vision Conference* (cit. on pp. 106, 108).

Crowley, Elliot J and Andrew Zisserman (2014b). “In search of art”. In: *Workshop at the European Conference on Computer Vision*. Springer, pp. 54–70 (cit. on p. 106).

– (2016). “The art of detection”. In: *European Conference on Computer Vision*. Springer, pp. 721–737 (cit. on p. 108).

Crowley, Elliot J, Omkar M Parkhi, and Andrew Zisserman (2015). “Face Painting: querying art with photos”. In: *British Machine Vision Conference*, pp. 65–1 (cit. on pp. 106, 108).

Deng, Jia, Wei Dong, Richard Socher, et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition*. Ieee, pp. 248–255 (cit. on pp. 28, 42).

Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, et al. (2015). “Flownet: Learning optical flow with convolutional networks”. In: *IEEE International Conference on Computer Vision*, pp. 2758–2766 (cit. on p. 43).

Farhadi, Ali, Mohsen Hejrati, Mohammad Amin Sadeghi, et al. (2010). “Every picture tells a story: Generating sentences from images”. In: *European Conference on Computer Vision*. Springer, pp. 15–29 (cit. on p. 105).

Fukushima, Kunihiko and Sei Miyake (1982). “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition”. In: *Competition and cooperation in neural nets*. Springer, pp. 267–285 (cit. on p. 28).

Garcia, Noa (2018). “Temporal Aggregation of Visual Features for Large-Scale Image-to-Video Retrieval”. In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. ACM, pp. 489–492 (cit. on p. 22).

Garcia, Noa and George Vogiatzis (2016). “Exploiting Redundancy in Binary Features for Image Retrieval in Large-Scale Video Collections”. In: *Proceedings of the European Conference on Visual Media Production* (cit. on p. 22).

– (2017). “Dress like a star: Retrieving fashion products from videos”. In: *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*. IEEE, pp. 2293–2299 (cit. on p. 22).

– (2018a). “Asymmetric Spatio-Temporal Embeddings for Large-Scale Image-to-Video Retrieval”. In: *Proceedings of the British Machine Vision Conference* (cit. on p. 22).

– (2018b). “How to Read Paintings: Semantic Art Understanding with Multi-Modal Retrieval”. In: *Proceedings of the European Conference in Computer Vision Workshops* (cit. on p. 22).

– (2019). “Learning Non-Metric Visual Similarity for Image Retrieval”. In: *Image and Vision Computing* (cit. on p. 21).

Gavet, Yann, Mathieu Fernandes, Johan Debayle, and Jean-Charles Pinoli (2014). “Dissimilarity criteria and their comparison for quantitative evaluation of image segmentation: application to human retina vessels”. In: *Machine Vision and Applications* 25.8, pp. 1953–1966 (cit. on p. 46).

Geetha, P and Vasumathi Narayanan (2008). “A survey of content-based video retrieval”. In: (cit. on pp. 15, 40).

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587 (cit. on p. 121).

Gong, Yunchao, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik (2014a). “Improving image-sentence embeddings using large weakly annotated photo collections”. In: *European Conference on Computer Vision*. Springer, pp. 529–545 (cit. on p. 105).

Gong, Yunchao, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik (2014b). “Multi-scale orderless pooling of deep convolutional activation features”. In: *European Conference on Computer Vision*. Springer, pp. 392–407 (cit. on pp. 42, 105, 117).

Gordo, Albert, Jon Almazán, Jerome Revaud, and Diane Larlus (2016). “Deep image retrieval: Learning global representations for image search”. In: *European Conference on Computer Vision*. Springer, pp. 241–257 (cit. on pp. 42, 48, 50, 60).

Gordo, Albert, Jon Almazan, Jerome Revaud, and Diane Larlus (2017). “End-to-end learning of deep visual representations for image retrieval”. In: *International Journal of Computer Vision* 124.2, pp. 237–254 (cit. on pp. 32, 40, 42, 58, 60, 66, 95).

Grana, Costantino, Daniele Borghesani, Marco Manfredi, and Rita Cucchiara (2013). “A fast approach for integrating ORB descriptors in the bag of words model”. In: *Multimedia Content and Mobile Devices*. Vol. 8667. International Society for Optics and Photonics, pp. 866–709 (cit. on p. 80).

Gygli, Michael (2017). “Ridiculously Fast Shot Boundary Detection with Fully Convolutional Neural Networks”. In: *arXiv preprint arXiv:1705.08214* (cit. on p. 91).

Hadi Kiapour, M, Xufeng Han, Svetlana Lazebnik, Alexander C Berg, and Tamara L Berg (2015). “Where to buy it: Matching street clothing photos in online shops”. In: *IEEE International Conference on Computer Vision*, pp. 3343–3351 (cit. on p. 76).

Han, Xufeng, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg (2015). “Matchnet: Unifying feature and metric learning for patch-based matching”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3279–3286 (cit. on p. 43).

Hassanien, Ahmed, Mohamed Elgharib, Ahmed Selim, Mohamed Hefeeda, and Wojciech Matusik (2017). “Large-scale, fast and accurate shot boundary detection through spatio-temporal convolutional neural networks”. In: *arXiv preprint arXiv:1705.03281* (cit. on p. 91).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (cit. on pp. 32, 42, 55, 58, 115).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780 (cit. on pp. 92, 98, 116).

Hodosh, Micah, Peter Young, and Julia Hockenmaier (2013). “Framing image description as a ranking task: Data, models and evaluation metrics”. In: *Journal of Artificial Intelligence Research* 47, pp. 853–899 (cit. on p. 105).

Hoffer, Elad and Nir Ailon (2015). “Deep metric learning using triplet network”. In: *International Workshop on Similarity-Based Pattern Recognition*. Springer, pp. 84–92 (cit. on p. 43).

Hotelling, Harold (1936). “Relations between two sets of variates”. In: *Biometrika* 28.3/4, pp. 321–377 (cit. on p. 105).

Ioffe, Sergey and Christian Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *International Conference on Machine Learning*, pp. 448–456 (cit. on p. 30).

Jégou, Hervé, Matthijs Douze, Cordelia Schmid, and Patrick Pérez (2010). “Aggregating local descriptors into a compact image representation”. In: *Computer Vision and Pattern Recognition*. IEEE, pp. 3304–3311 (cit. on pp. 27, 40, 42).

Jiménez, Albert, Jose M Alvarez, and Xavier Giró Nieto (2017). “Class-weighted convolutional features for visual instance search”. In: *British Machine Vision Conference* (cit. on pp. 42, 59).

Johnson, C Richard, Ella Hendriks, Igor J Berezhnoy, et al. (2008). “Image processing for artist identification”. In: *IEEE Signal Processing Magazine* 25.4 (cit. on p. 107).

Kalantidis, Yannis, Clayton Mellina, and Simon Osindero (2016). “Cross-dimensional weighting for aggregated deep convolutional features”. In: *European Conference on Computer Vision*. Springer, pp. 685–701 (cit. on pp. 42, 59).

Karayev, Sergey, Matthew Trentacoste, Helen Han, et al. (2014). “Recognizing Image Style”. In: *British Machine Vision Conference* (cit. on pp. 107, 108).

Karpathy, Andrej, Armand Joulin, and Li F Fei-Fei (2014). “Deep fragment embeddings for bidirectional image sentence mapping”. In: *Advances in Neural Information Processing Systems*, pp. 1889–1897 (cit. on pp. 16, 105).

Kato, Toshikazu (1992). “Database architecture for content-based image retrieval”. In: *Image Storage and Retrieval Systems*. Vol. 1662. International Society for Optics and Photonics, pp. 112–124 (cit. on p. 17).

Khan, Fahad Shahbaz, Shida Beigpour, Joost Van de Weijer, and Michael Felsberg (2014). “Painting-91: a large scale database for computational painting categorization”. In: *Machine Vision and Applications* 25.6, pp. 1385–1397 (cit. on pp. 107, 108).

Kingma, Diederik P and Jimmy Ba (2015). “Adam: A method for stochastic optimization”. In: *International Conference on Learning Representations* (cit. on p. 97).

Kiros, Ryan, Yukun Zhu, Ruslan R Salakhutdinov, et al. (2015). “Skip-thought vectors”. In: *Advances in Neural Information Processing Systems*, pp. 3294–3302 (cit. on p. 116).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105 (cit. on pp. 28, 31, 32, 42, 107).

Kulis, Brian et al. (2013). “Metric learning: A survey”. In: *Foundations and Trends® in Machine Learning* 5.4, pp. 287–364 (cit. on pp. 41, 43, 54, 56).

Li, Wei, Rui Zhao, Tong Xiao, and Xiaogang Wang (2014). “Deepreid: Deep filter pairing neural network for person re-identification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 152–159 (cit. on p. 43).

Liu, Si, Zheng Song, Guangcan Liu, et al. (2012). “Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3330–3337 (cit. on p. 76).

Liu, Yu, Yanming Guo, Song Wu, and Michael S Lew (2015). “Deepindex for accurate and efficient image retrieval”. In: *ACM on International Conference on Multimedia Retrieval*. ACM, pp. 43–50 (cit. on p. 41).

Lowe, David G (2004). “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2, pp. 91–110 (cit. on pp. 17, 26, 40, 64, 95).

Luo, Wenjie, Alexander G Schwing, and Raquel Urtasun (2016). “Efficient deep learning for stereo matching”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5695–5703 (cit. on p. 43).

Ma, Daiqian, Feng Gao, Yan Bai, et al. (2017). “From Part to Whole: Who is Behind the Painting?” In: *ACM on Multimedia Conference*. ACM, pp. 1174–1182 (cit. on p. 107).

MacQueen, James (1967). “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA, pp. 281–297 (cit. on p. 27).

Manning, Christopher D, Christopher D Manning, and Hinrich Schütze (1999). *Foundations of statistical natural language processing*. MIT press (cit. on p. 112).

Mao, Hui, Ming Cheung, and James She (2017). “DeepArt: Learning Joint Representations of Visual Arts”. In: *ACM on Multimedia Conference*. ACM, pp. 1183–1191 (cit. on pp. 107, 108, 134).

McFee, Brian and Gert R Lanckriet (2010). “Metric learning to rank”. In: *International Conference on Machine Learning*, pp. 775–782 (cit. on p. 43).

Meng, Jingjing, Junsong Yuan, Jiong Yang, Gang Wang, and Yap-Peng Tan (2016). “Object instance search in videos via spatio-temporal trajectory discovery”. In: *IEEE Transactions on Multimedia* 18.1, pp. 116–127 (cit. on p. 64).

Mensink, Thomas and Jan Van Gemert (2014). “The rijksmuseum challenge: Museum-centered visual recognition”. In: *Proceedings of International Conference on Multimedia Retrieval*. ACM, p. 451 (cit. on pp. 107, 108).

Mikolajczyk, Krystian and Cordelia Schmid (2005). “A performance evaluation of local descriptors”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10, pp. 1615–1630 (cit. on p. 27).

Miksik, Ondrej and Krystian Mikolajczyk (2012). “Evaluation of local detectors and descriptors for fast feature matching”. In: *International Conference on Pattern Recognition*. IEEE, pp. 2681–2684 (cit. on pp. 27, 78, 79).

Mohedano, Eva, Kevin McGuinness, Noel E O’Connor, et al. (2016). “Bags of local convolutional features for scalable instance search”. In: *ACM on International Conference on Multimedia Retrieval*. ACM, pp. 327–331 (cit. on pp. 42, 59).

Movshovitz-Attias, Yair, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh (2017). “No Fuss Distance Metric Learning Using Proxies”. In: *IEEE International Conference on Computer Vision*. IEEE, pp. 360–368 (cit. on p. 48).

Muja, Marius and David G Lowe (2012). “Fast matching of binary features”. In: *Computer and Robot Vision*. IEEE, pp. 404–410 (cit. on p. 81).

Nam, Hyeonseob, Jung-Woo Ha, and Jeonghee Kim (2017). “Dual Attention Networks for Multimodal Reasoning and Matching”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 299–307 (cit. on pp. 104, 105).

Ngiam, Jiquan, Aditya Khosla, Mingyu Kim, et al. (2011). “Multimodal deep learning”. In: *International Conference on Machine Learning*, pp. 689–696 (cit. on p. 16).

Nister, David and Henrik Stewenius (2006). “Scalable recognition with a vocabulary tree”. In: *Computer Vision and Pattern Recognition*. Vol. 2. IEEE, pp. 2161–2168 (cit. on pp. 62, 67).

Noh, Hyeonwoo, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han (2017). “Large-Scale image retrieval with attentive deep local features”. In: *IEEE International Conference on Computer Vision*, pp. 3456–3465 (cit. on p. 42).

Oh Song, Hyun, Yu Xiang, Stefanie Jegelka, and Silvio Savarese (2016). “Deep metric learning via lifted structured feature embedding”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4004–4012 (cit. on p. 43).

Oliveira Barra, Gabriel de, Mathias Lux, and Xavier Giro-i Nieto (2016). “Large scale content-based video retrieval with LIvRE”. In: *Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on*. IEEE, pp. 1–4 (cit. on p. 97).

Ordonez, Vicente, Girish Kulkarni, and Tamara L Berg (2011). “Im2text: Describing images using 1 million captioned photographs”. In: *Advances in Neural Information Processing Systems*, pp. 1143–1151 (cit. on p. 104).

Over, Paul, George Awad, Jon Fiscus, et al. (2011). “TRECVID 2011-An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics”. In: *TRECVID 2011-TREC Video Retrieval Evaluation Online* (cit. on p. 64).

Pan, Yingwei, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui (2016). “Jointly modeling embedding and translation to bridge video and language”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4594–4602 (cit. on p. 16).

Perronnin, Florent, Yan Liu, Jorge Sánchez, and Hervé Poirier (2010). “Large-scale image retrieval with compressed fisher vectors”. In: *Computer Vision and Pattern Recognition*. IEEE, pp. 3384–3391 (cit. on pp. 17, 27, 40, 65).

Philbin, James, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman (2007). “Object retrieval with large vocabularies and fast spatial matching”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–8 (cit. on p. 50).

– (2008). “Lost in quantization: Improving particular object retrieval in large scale image databases”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–8 (cit. on p. 50).

Qian, Qi, Rong Jin, Shenghuo Zhu, and Yuanqing Lin (2015). “Fine-grained visual categorization via multi-stage metric learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3716–3724 (cit. on p. 43).

Quinlan, J. Ross (1986). “Induction of decision trees”. In: *Machine learning* 1.1, pp. 81–106 (cit. on p. 82).

Radenović, Filip, Giorgos Tolias, and Ondřej Chum (2016). “CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples”. In: *European Conference on Computer Vision*. Springer, pp. 3–20 (cit. on pp. 42, 48, 55, 58, 60).

Razavian, Ali S, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki (2016). "Visual instance retrieval with deep convolutional networks". In: *ITE Transactions on Media Technology and Applications* 4.3, pp. 251–258 (cit. on p. 42).

Razavian, Ali Sharif, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson (2014). "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 806–813 (cit. on pp. 31, 41, 42, 59).

Rohrbach, Anna, Atousa Torabi, Marcus Rohrbach, et al. (2017). "Movie Description". In: *IJCV* (cit. on p. 94).

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski (2011). "ORB: An efficient alternative to SIFT or SURF". In: *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, pp. 2564–2571 (cit. on p. 84).

Saleh, Babak and Ahmed Elgammal (2015). "Large-scale Classification of Fine-Art Paintings: Learning The Right Metric on The Right Feature". In: *International Journal for Digital Art History* 2, pp. 72–93 (cit. on p. 107).

Salvador, Amaia, Xavier Giró-i Nieto, Ferran Marqués, and Shin'ichi Satoh (2016). "Faster r-cnn features for instance search". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 9–16 (cit. on pp. 42, 59, 60).

Salvador, Amaia, Nicholas Hynes, Yusuf Aytar, et al. (2017). "Learning Cross-Modal Embeddings for Cooking Recipes and Food Images". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3068–3076 (cit. on pp. 104–106).

Seguin, Benoit, Carlotta Striolo, Frederic Kaplan, et al. (2016). "Visual link retrieval in a database of paintings". In: *European Conference on Computer Vision Workshops*. Springer, pp. 753–767 (cit. on pp. 107, 108).

Shamir, Lior, Tomasz Macura, Nikita Orlov, D Mark Eckley, and Ilya G Goldberg (2010). "Impressionism, expressionism, surrealism: Automated recognition of painters and schools of art". In: *ACM Transactions on Applied Perception* 7.2, p. 8 (cit. on p. 107).

Simonyan, K. and A. Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations* (cit. on pp. 32, 42, 51, 55, 57, 96, 115).

Sivic, Josef and Andrew Zisserman (2003). "Video Google: A text retrieval approach to object matching in videos". In: *International Conference on Computer Vision*. IEEE, pp. 1470–1477 (cit. on pp. 17, 27, 40, 42, 62, 67).

Sivic, Josef, Frederik Schaffalitzky, and Andrew Zisserman (2006). "Object level grouping for video shots". In: *International Journal of Computer Vision* 67.2, pp. 189–210 (cit. on p. 64).

Smeulders, Arnold WM, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain (2000). "Content-based image retrieval at the end of the early years". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 12, pp. 1349–1380 (cit. on pp. 15, 40).

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958 (cit. on p. 30).

Sun, Zhonghua, Kebin Jia, and Hexin Chen (2008). “Video key frame extraction based on spatial-temporal color distribution”. In: *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, pp. 196–199 (cit. on p. 84).

Szegedy, Christian, Wei Liu, Yangqing Jia, et al. (2015). “Going deeper with convolutions”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9 (cit. on p. 42).

Szeliski, Richard (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media (cit. on p. 26).

Takacs, Gabriel, Vijay Chandrasekhar, Natasha Gelfand, et al. (2008). “Outdoors augmented reality on mobile phone using loxel-based visual feature organization”. In: *ACM International Conference on Multimedia Information Retrieval*. ACM, pp. 427–434 (cit. on p. 15).

Tamura, Hideyuki and Naokazu Yokoya (1984). “Image database systems: A survey”. In: *Pattern Recognition* 17.1, pp. 29–43 (cit. on p. 16).

Tan, Wei Ren, Chee Seng Chan, Hernán E Aguirre, and Kiyoshi Tanaka (2016). “Ceci n’est pas une pipe: A deep convolutional network for fine-art paintings classification”. In: *IEEE International Conference on Image Processing*. IEEE, pp. 3703–3707 (cit. on p. 107).

Tan, Xiaoyang, Songcan Chen, Jun Li, and Zhi-Hua Zhou (2006). “Learning Non-Metric Partial Similarity Based on Maximal Margin Criterion”. In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*. IEEE Computer Society, pp. 168–145 (cit. on p. 46).

Thewlis, J., S. Zheng, P. Torr, and A. Vedaldi (2016). “Fully-Trainable Deep Matching”. In: *British Machine Vision Conference* (cit. on p. 43).

Tolias, Giorgos, Ronan Sicre, and Hervé Jégou (2016). “Particular object retrieval with integral max-pooling of CNN activations”. In: *Proceedings of the International Conference on Learning Representations* (cit. on pp. 17, 32, 42, 46, 51, 55, 57, 59, 65, 90, 115).

Tversky, Amos and Itamar Gati (1982). “Similarity, separability, and the triangle inequality.” In: *Psychological review* 89.2, p. 123 (cit. on p. 46).

Wan, Ji, Dayong Wang, Steven Chu Hong Hoi, et al. (2014). “Deep learning for content-based image retrieval: A comprehensive study”. In: *ACM International Conference on Multimedia*. ACM, pp. 157–166 (cit. on pp. 41, 44, 59, 60).

Wang, Jiang, Yang Song, Thomas Leung, et al. (2014). “Learning fine-grained image similarity with deep ranking”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1386–1393 (cit. on p. 43).

Wang, Liwei, Yin Li, and Svetlana Lazebnik (2016). “Learning deep structure-preserving image-text embeddings”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5005–5013 (cit. on p. 66).

Wang, Liwei, Yin Li, Jing Huang, and Svetlana Lazebnik (2018). “Learning two-branch neural networks for image-text matching tasks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (cit. on pp. 104–106, 122).

Wang, Mao, Yuewei Ming, Qiang Liu, and Jianping Yin (2017). “Image-based video retrieval using deep feature”. In: *IEEE International Conference on Smart Computing*. IEEE, pp. 1–6 (cit. on pp. 65, 103, 105).

Weston, Jason, Samy Bengio, and Nicolas Usunier (2011). “Wsabie: Scaling up to large vocabulary image annotation”. In: *IJCAI*. Vol. 11, pp. 2764–2770 (cit. on p. 105).

Wu, Jian, Zhiming Cui, Victor S Sheng, et al. (2013a). “A Comparative Study of SIFT and its Variants”. In: *Measurement science review* 13.3, pp. 122–131 (cit. on p. 26).

Wu, Pengcheng, Steven CH Hoi, Hao Xia, et al. (2013b). “Online multimodal deep similarity learning with application to image retrieval”. In: *ACM international conference on Multimedia*. ACM, pp. 153–162 (cit. on p. 43).

Xie, Lingxi, Richang Hong, Bo Zhang, and Qi Tian (2015). “Image classification and retrieval are one”. In: *ACM on International Conference on Multimedia Retrieval*. ACM, pp. 3–10 (cit. on p. 42).

Yan, Fei and Krystian Mikolajczyk (2015). “Deep correlation for matching images and text”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3441–3450 (cit. on p. 105).

Yue-Hei Ng, Joe, Fan Yang, and Larry S Davis (2015). “Exploiting local features from deep networks for image retrieval”. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 53–61 (cit. on pp. 42, 59).

Zagoruyko, Sergey and Nikos Komodakis (2015). “Learning to compare image patches via convolutional neural networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4353–4361 (cit. on p. 43).

Zheng, Liang, Yi Yang, and Qi Tian (2018). “SIFT meets CNN: A decade survey of instance retrieval”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.5, pp. 1224–1244 (cit. on pp. 15, 40, 44).

Zhu, Cai-Zhi and Shin’ichi Satoh (2012). “Large vocabulary quantization for searching instances from videos”. In: *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*. ACM, p. 52 (cit. on p. 65).