UNIVERSITÁ DI PISA
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE

RELATIONAL LEARNING IN COMPUTER VISION

DOCTORAL THESIS

Author
Nicola Messina

Tutor (s)
Dr. Fabrizio Falchi, Dr. Giuseppe Amato, Prof. Marco Avvenuti

Reviewer (s)
Prof. Jakub Lokoč, Prof. Elisa Ricci

The Coordinator of the PhD Program
Prof. Fulvio Gini

Pisa, May 2022

Cycle XXXIV

To my family

«The *thing-in-itself* nonsensical. If I remove all the relationships, all the *properties*, all the *activities* of a thing, the thing does not remain over.»

Friedrich Nietzsche

Acknowledgements

I don't think there is enough space here to thank from the bottom of my heart all the people that contributed, directly or indirectly, to this thesis. A huge first thank goes to Dr. Fabrizio Falchi, Dr. Giuseppe Amato, and Dr. Claudio Gennaro for allowing me to start this extraordinary journey. Thank you for supporting me at every moment, from the uncertainty of the first steps in the complex research world to the writing of this thesis. Each chapter tells a three-years-long story where you are the undisputed protagonists.

I would like to thank Prof. Marco Avvenuti for supervising the work and Prof. Jakub Lokoč and Prof. Elisa Ricci for revising the manuscript and returning me valuable feedback.

I thank all the extraordinary people of the Artificial Intelligence for Media and Humanities lab that I met and with whom I had the honor of collaborating. First of all, I would like to thank my adventure companion, Luca. With him, I shared frustrations and fears, but also joys and many great ideas. Infinite thanks to Fabio Carrara, exceptional mentor and my fellow both in the world of Deep Learning and in that of music and drums. Thanks to Marco Di Benedetto, for his infinite availability of memes and memorable videos from the web, and Paolo Bolettieri, for the very elegant jokes, exchanged on Skype during the pandemic (which have often further cracked relations between Massa and Carrara). A great thank goes to Claudio Vairo, that wisely and courageously guarded room I-12 during our absence caused by the pandemic. I would like to mention the extraordinary VISIONE group, led by the tireless Lucia Vadicamo. You gave a practical meaning to my research. The beating heart of the thesis would not exist without you.

I infinitely thank all my friends, Carla, Irene, Giovanni, Agnese, Martina, Elena, Federico, Alessandro, Lorenzo. You enlightened me with many evenings of lightheartedness and fun, despite these last two difficult years of forced confinement. Thanks to Claudio, Manuel and Alessio for the unforgettable evenings spent in the rehearsal room to produce authentic music, and to Gabriele and Niccolò for the endless hours in the night spent cursing Blender and Unreal Engine.

But I wouldn't be here writing if it weren't for my parents, Mario and Milena. You

supported me at any time, from the beginning of this adventure, and you brought me, with patience and extreme wisdom, day by day, on the right path. I will never thank you enough.

To Michele, my brother, I owe more than he can expect. The passion we share for science and music created that magical whirlwind of ideas that are now indelibly embedded in the spine of this manuscript. I admire you, and you will always be my point of reference in science, music and life.

And finally, an infinite thank to the person who was by my side every single moment since the beginning of this incredible experience. Marianna, you made this adventure possible. You followed the most difficult but also the most beautiful moments of this journey, often supporting my desire to walk kilometers to release tension and constantly reminding me to proceed one step at a time. I tried to follow your advice, and if I'm writing these lines, it's because it perfectly worked. I'll never stop loving you. You will always be my safe haven.

Summary

THE increasing interest in social networks, smart cities, and Industry 4.0 is encouraging the development of techniques for processing, understanding, and organizing vast amounts of data. Recent important advances in Artificial Intelligence brought to life a subfield of Machine Learning called Deep Learning, which can automatically learn common patterns from raw data directly, without relying on manual feature selection. This framework overturned many computer science fields, like Computer Vision and Natural Language Processing, obtaining astonishing results. Nevertheless, many challenges are still open. Although deep neural networks obtained impressive results on many tasks, they cannot perform non-local processing by explicitly relating potentially interconnected visual or textual entities. This relational aspect is fundamental for capturing high-level semantic interconnections in multimedia data or understanding the relationships between spatially distant objects in an image.

This thesis tackles the relational understanding problem in Deep Neural Networks, considering three different yet related tasks. First, we introduce a challenging variant of the Content-Based Image Retrieval (CBIR) task, called Relational CBIR. In R-CBIR, we aim to retrieve images also having similar relationships among the multiple objects present in the images. We define some architectures able to extract relationship-aware visual descriptors, and we extend the CLEVR synthetic dataset for obtaining a suitable ground-truth for evaluating R-CBIR. Then, we move a step further, considering real-world images and focusing on cross-modal visual-textual retrieval. We use the Transformer Encoder, a recently introduced module that relies on the power of self-attention, to relate different sentence words and image regions, with large-scale retrieval as the main goal. We show that the obtained features contain very high-level semantics and defeat current image descriptors on the challenging Semantic CBIR task. We then propose some solutions for scaling the search to possibly millions of images or texts. In the end, we deploy the developed networks in a large-scale interactive video retrieval software, called VISIONE, developed in our laboratory. Sticking to the multi-modal Transformer framework, we tackle another critical task in the modern Internet: detecting persuasion techniques in memes spread on social networks during disinformation campaigns. Finally, we probe current state-of-the-art CNNs on challenging visual rea-

soning benchmarks requiring non-local spatial comparisons. After understanding the drawbacks of CNNs on these tasks, we propose a hybrid CNN-Transformer architecture, constraining the model complexity and reaching higher data efficiency.

In the end, the research presented in this thesis aims to explore novel and exciting directions for an effective and efficient semantic and relational understanding of multimedia data.

Sommario

Il crescente interesse verso i social network, le smart cities e l'Industria 4.0 sta incentivando lo sviluppo di tecniche per processare, comprendere e organizzare enormi quantità di dati. I recenti sviluppi nell'ambito dell'Intelligenza Artificiale hanno dato vita al Deep Learning, una branca del Machine Learning che riconosce autonomamente i pattern più rilevanti nei dati in input, senza dover dipendere da una selezione guidata da un esperto umano. Il Deep Learning ha rivoluzionato importanti campi applicativi, come la Computer Vision e il Natural Language Processing; nonstante ciò, soffre ancora di importanti limitazioni. Sebbene siano stati raggiunti risultati straordinari in molti campi applicativi, le reti neurali hanno ancora difficoltà nel comprendere la relazione tra elementi semanticamente collegati tra loro ma *distanti*, in riferimento sia alla dimensione spazio-temporale ma anche più genericamente alla loro forma (un testo è in sua essenza diverso da un'immagine, anche se può perfettamente descriverla). Questa mancanza ha ripercussioni negative sulla ricerca di interconnessioni tra oggetti multimediali aventi natura differente o sulla ricerca di relazioni tra oggetti spazialmente distanti in un'immagine.

In questa tesi abbiamo affrontato il problema della comprensione relazionale nelle reti neurali profonde, prendendo come riferimento tre task differenti ma strettamente correlati tra loro. In primo luogo, abbiamo introdotto il Relational Content-Based Image Retrieval (R-CBIR) – un'estensione al task di CBIR classico – il cui scopo è quello di cercare tutte le immagini che condividano una similarità tra le relazioni che insistono tra gli oggetti in esse contenuti. Abbiamo affrontato il Relational CBIR definendo alcune architetture capaci di estrarre dei descrittori relazionali ed estendendo il dataset sintetico CLEVR per ottenere un ground-truth adatto alla valutazione di questo nuovo task. Il passo successivo ha riguardato l'ampliamento di questi risultati preliminari verso l'utilizzo di immagini reali nel contesto di ricerce cross-modali, dove descrizioni in linguaggio naturale vengono usate come query per cercare in grossi database di immagini (e viceversa). Abbiamo utilizzato l'architettura Transformer per correlare elementi visuali e testuali, ponendoci come obiettivo finale la ricerca su larga scala. Dopo aver effettuato l'integrazione di queste reti in uno strumento per la ricerca interattiva di video su larga scala (VISIONE), abbiamo osservato come i descrittori ottenuti

siano capaci di codificare elementi altamente semantici, raggiungendo risultati eccellenti sul task di Semantic CBIR. Abbiamo infine utilizzato queste stesse tecnologie per risolvere un problema estremamente importante nei social network: la rilevazione di tecniche di persuasione nelle campagne di disinformazione. L'ultima parte della ricerca si è focalizzata sullo studio delle architetture convoluzionali su semplici problemi di ragionamento visivo, che richiedono confronti tra forme distanti nello spazio. In questo contesto abbiamo proposto un'architettura ibrida CNN-Transformer che ha ottenuto ottimi risultati, rimanendo comunque meno complessa e più efficiente rispetto alle reti concorrenti.

Lo scopo primario di questa tesi è stato quello di esplorare nuovi modelli neurali per la comprensione semantica e relazionale di immagini e testi, con applicazioni su larga scala e con immediate estensioni a ulteriori modalità quali audio e/o video.

List of publications

International Journals

1. Messina, N., Amato, G., Carrara, F., Falchi, F., and Gennaro, C. (2020). Learning Visual Features for Relational CBIR. *International Journal of Multimedia Information Retrieval*. (Vol. 9(2), pp. 113-124). Springer.

2. Ciampi, L., Messina, N., Falchi, F., Gennaro, C., and Amato, G. (2020). Virtual to Real Adaptation of Pedestrian Detectors. *Sensors*. (Vol. 20(18), pp. 5250). MDPI.

3. Messina, N., Amato, G., Carrara, F., Gennaro, C., and Falchi, F. (2021). Solving the Same-different Task with Convolutional Neural Networks. *Pattern Recognition Letters*. (Vol. 143, pp. 75-80). Elsevier.

4. Messina, N., Amato, G., Esuli, A., Falchi, F., Gennaro, C., and Marchand-Maillet, S. (2021). Fine-grained Visual Textual Alignment for Cross-modal Retrieval using Transformer Encoders. *Trans on Multimedia Computing Communications and Applications (TOMM)*. ACM.

International Conferences/Workshops with Peer Review

1. Messina, N., Amato, G., Carrara, F., Falchi, F., and Gennaro, C. (2018). Learning Relationship-aware Visual Features. *European Conference on Computer Vision (ECCV) Workshops*. (pp. 486-501). Springer, Cham.

2. Messina, N., Amato, G., Carrara, F., Falchi, F., and Gennaro, C. (2019, September). Testing Deep Neural Networks on the Same-different Task. *International Conference on Content-Based Multimedia Indexing (CBMI)*. (pp. 1-6). IEEE.

3. Amato, G., Ciampi, L., Falchi, F., Gennaro, C., and Messina, N. (2019, September). Learning Pedestrian Detection from Virtual Worlds. *International Conference on Image Analysis and Processing (ICIAP)*. (pp. 302-312). Springer, Cham.

4. Messina, N., Falchi, F., Esuli, A., and Amato, G. (2021, January). Transformer Reasoning Network for Image-text Matching and Retrieval. *International Conference on Pattern Recognition (ICPR)*. (pp. 5222-5229). IEEE.

5. Messina, N. (2020, September). Relational Visual-Textual Information Retrieval. *International Conference on Similarity Search and Applications*. (pp. 405-411). Springer, Cham.

6. Messina, N., Amato, G., Falchi, F., Gennaro, C., and Marchand-Maillet, S. (2021). Towards Efficient Cross-Modal Visual Textual Retrieval using Transformer-Encoder Deep Features. *International Conference on Content-Based Multimedia Indexing (CBMI)*. (pp. 1-6). IEEE.

7. Amato, G., Bolettieri, P., Falchi, F., Gennaro, C., Messina, N., Vadicamo, L., and Vairo, C. (2021, June). VISIONE at Video Browser Showdown 2021. *International Conference on Multimedia Modeling (ICMM)*. (pp. 473-478). Springer, Cham.

8. Messina, N., Falchi, F., Gennaro, C., and Amato, G. (2021, August). AIMH at SemEval-2021 Task 6: Multimodal Classification using an Ensemble of Transformer Models. *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval)*. (pp. 1020-1026). ACL.

9. Messina, N., Amato, G., Carrara, F., Gennaro, C., and Falchi, F. (2021). Recurrent Vision Transformer for Solving Visual Reasoning Problems. To appear in the proceedings of *International Conference on Image Analysis and Processing (ICIAP)*. Springer, Cham.

List of Abbreviations

Symbols

2S-RN	Two-stage Relation Network. 46–50, 53–57

A

Adam	Adaptive Moment Estimation. 11, 12
ANN	Artificial Neural Network. 6
AP	Average Precision. 36
AVF-RN	Aggregated Visual Features Relation Network. 46, 48–50, 53, 55–58, 60

B

BCE	Binary Cross-Entropy. 113, 119
BERT	Bidirectional Encoder Representations from Transformers. 27–31
BoC	Bag-of-Concepts. 84, 86
BoW	Bag-of-Words. 41, 92, 130

C

CBIR	Content-based Image Retrieval. 3, 32–35, 42, 43, 45–47, 52, 59, 91, 126, 127
CLEVR	Compositional Language and Elementary Visual Reasoning. 38, 39, 41, 43, 49–55
CNN	Convolutional Neural Network. 3–5, 14, 17–24, 29, 30, 34, 63, 109, 111, 116, 124, 126, 127

D

DCG	Discounted Cumulative Gain. 38
DL	Deep Learning. 2, 5–8, 10, 11, 14, 17, 30, 34, 42, 126, 128

DNN	Deep Neural Network. 4–11, 13, 14, 20, 23, 34, 38, 41–43, 67, 85, 108, 110, 124, 126, 128, 130

F

FCNN	Fully-Connected Neural Network. 8
FFNN	Feed-Forward Neural Network. 10, 17, 20

G

GED	Graph Edit Distance. 51, 52
GeM	Generalized Mean. 35, 92, 99
GNN	Graph Neural Network. 14, 23, 24, 64, 65, 119
GRU	Gated Recurrent Unit. 21, 22, 25, 63

I

ILSVRC	ImageNet Large Scale Visual Recognition Challenge. 19

K

KNN	K-nearest neighbors. 16, 32, 35, 65, 83

L

LSTM	Long Short-term Memory. 21, 22, 25, 63

M

mAP	mean Average Precision. 36, 92
ML	Machine Learning. 2, 5, 8, 12
MLP	Multilayer Perceptron. 6–8, 18, 19, 22, 24, 47
MSE	Mean Squared Error. 15

N

NDCG	Normalized Discounted Cumulative Gain. 38, 45, 63, 67, 73, 92
NLP	Natural Language Processing. 25, 29, 118

R

R-CBIR	Relational Content-based Image Retrieval. 3, 34, 42–46, 50, 53, 56, 57, 59, 91, 127–129
R-MAC	Regional Maximum Activations of Convolutions. 34, 35, 45, 52, 53, 56, 92, 94, 95, 99, 127, 128
R-VQA	Relational Visual Question Answering. 22, 23, 38, 42–46, 49, 59, 60, 109, 127

R@K Recall@K. 37, 66, 73, 88, 94, 95
ReLU Rectified Linear Unit. 7, 10, 19
ResNet Residual Network. 19, 20, 30, 63, 70, 111, 122
RN Relation Network. 2–4, 22–24, 45–50, 53–57, 59, 64, 121, 126–129
RNN Recurrent Neural Network. 5, 20, 21, 25, 26

S

S-CBIR Semantic Content-based Image Retrieval. 41, 91, 93, 98, 99, 107, 127, 128
SGD Stochastic Gradient Descent. 11, 12
SIFT Scale Invariant Feature Transform. 34
STR Surrogate Text Representation. 83, 84, 94, 95
SURF Speeded Up Robust Features. 34
SVRT Synthetic Visual Reasoning Test. 40, 109, 110, 128

T

TE Transformer Encoder. 4, 27–31, 61, 64, 68, 70, 71, 102, 105, 107, 119
TERAN Transformer Encoder Reasoning and Alignment Network. 62, 71, 128
TERN Transformer Encoder Reasoning Network. 62, 71, 73, 92, 97, 99, 128

V

VCR Visual Commonsense Reasoning. 30
VGG Visual Geometry Group. 19, 63, 111
ViT Vision Transformer. 22, 29, 30, 118, 119, 121
VLAD Vector of Locally Aggregated Descriptors. 34
VQA Visual Question Answering. 15, 24, 30, 41, 44
VRD Visual Relationship Detection. 41, 42, 44

Contents

CHAPTER 1

Introduction

The world is inherently relational, as it is built of complex interconnections between multiple and possibly abstract entities. Humans perform short-term predictions and make logical deductions by understanding how entities are interacting — for example, by observing a frisbie in the air we deduce that a nearby person should have launched it. At a more fundamental level, physics always describes the world as mathematical dependencies between measurable variables, constructing working *models* of the world. Therefore, understanding relationships is the key to comprehending the complexity of the reality in which we live. Humans learn to distill high-level concepts and abstract relationships by processing the sensory data they perceive through their senses. For this reason, distilling relationships is a way to create knowledge from raw unstructured data.

Today we live in a world where machines have been given almost full access to human senses: information is spread over the internet in the form of multi-modal data like images, texts, and videos carrying high-level concepts, customs of a people, and ideas. In 2020, according to Statista[1], in a single minute more than 500 hours of video were uploaded to YouTube, and more than 340,000 stories were posted on Instagram. Digital computers, in the range of a few decades, evolved from evaluating simple formulas to handling an increasing volume of complex multi-modal data.

Today there is the necessity of *understanding* all these data to obtain high-level usable knowledge, pretty much as humans do. In fact, the vast volumes of information need to be automatically organized and indexed to be efficiently retrieved; the good old manual annotation procedure today does not scale with the amount of data produced. Also, with the recent development of smart cities and Industry 4.0, there is an increasing interest in processing videos from surveillance cameras to automatically understand

[1] www.statista.com

abnormalities and monitor human activities to raise alerts when the situation is potentially dangerous or unsafe. All this requires careful filtering of the salient data and a thought interconnection of the related abstract entities.

For all these reasons, research focused on methods for autonomously extracting high-level and relational knowledge from raw information. Images, videos, audio, and texts are *unstructured data*, in which knowledge is hidden or not directly accessible, mainly because it is present at different levels of abstractions. We can concentrate on the color tonalities of an image, and at the same time, we can appreciate its content, or we can understand the various interactions between the elements contained in the image to guess how the scene is going to evolve in the successive instants of time.

In the last years, a branch of Machine Learning (ML), called Deep Learning (DL), really transformed the way unstructured inputs like images, videos, and natural language text are processed to derive high level representations. Deep Learning uses neural networks built of multiple cascading layers, emulating the inner working of the mammal brain to derive abstract representations of the given unstructured input. This novel paradigm defines an effective way to automatically craft informative features from unstructured data, relieving the human from the feature creation loop. Despite this new technology demonstrated astonishing results in many tasks — image classification, image captioning, and object detection, to name a few — it often fails to capture long-term dependencies and understand abstract relationships among entities that are distant both in space and/or time. For example, in image processing, current models excel in aggregating information from nearby pixels; however, they struggle to explicitly relate two distant objects, such as two persons in the opposite edges of the frame. In other words, although most architectures can *perceive* the world we live in quite well, they often struggle in *reasoning* about the high-level interconnections between the extracted concepts.

Deep Learning take inspiration from the brain-inspired paradigm, under which external symbols are processed and converted to internal vectors of neural activity. Under this paradigm, many recent architectures, like Transformers [220] or Relation Networks [205] capture interdependencies among visual or textual tokens, producing vectors that carry not only entities but also abstract relationships between them. According to the Turing Award winners Yoshua Bengio, Yann Lecun, and Geoffrey Hinton, «the main advantage of using vectors of neural activity to represent concepts and weight matrices to capture relationships between concepts is that this leads to automatic generalization. [...] This facilitates analogical reasoning and suggests that immediate, intuitive analogical reasoning is our primary mode of reasoning» [31].

Understanding relationships is the key for constructing a very high-level knowledge of the data, very different from the one that can be collected from analyzing image patches, distributions of colors, or even frequencies of appearance of words in a document. This knowledge can be derived by understanding what are the salient *elements* or *actors* in a multimedia file — a person or a tree in an image or analogously two words in a sentence — and then understanding their spatial, temporal, or abstract interconnections. At this level of abstraction, the potential is huge: we can enable high-level understanding of complex situations comprising multiple elements, such as an image or a video with many interacting actors and objects, or we can even study and enforce direct correspondence between elements laying in different modalities — for example,

we could connect regions from an image with words from a sentence describing it. All the possibilities offered by the study of relationships move toward creating more *semantic* representations, to the point that the terms *semantic* and *relational* almost line up.

1.1 Objectives and Contributions

This thesis presents the research carried out during the three-year period of the PhD program 2018-2021. Our focus has been directed towards using the well-known and widely studied deep learning framework to understand high-level entity relationships in visual and textual data, mainly — but not exclusively — for information retrieval purposes.

More in detail, we tried to stick to the following objectives:

1. Understand the current limits of deep learning architectures when facing visual *relational* tasks involving the comprehension of spatial and abstract relationships between entities.

2. Propose relationship-aware descriptors for enabling highly semantic information retrieval.

3. Introduce appropriate metrics and benchmarks for evaluating the relational abilities of the proposed architectures.

In the first two chapters, we introduce the reader to our work and provide sufficient background for understanding the rest of this thesis. In the light of the above-listed objectives, the main dissertation chapters propose solutions for addressing the following problems:

Understanding Relationships in Images for Content-Based Image Retrieval Content-based Image Retrieval (CBIR) consists of searching digital images in large databases, without relying on manual annotations or user-provided metadata. It is an inherently difficult task due to the well-known *semantic gap* [54]. Many works in the past used local features (SIFT, ORB) and their aggregates (VLAD, Fisher Vectors) as image descriptors. In contrast, more recently, features like R-MAC used the representation power of Convolutional Neural Network (CNN) trained on image classification to acquire higher-level single-entity details. Nevertheless, some key architectures were recently proposed to capture higher-level interactions between multiple visual entities; without correctly understanding abstract or spatial relationships between objects in the image, it is fairly challenging to forge highly semantic descriptors. In Chapter 3 we introduce a novel task that directly derives from CBIR, called Relational Content-based Image Retrieval (R-CBIR), in which we aim to retrieve images containing similar pairwise relationships between visual entities. In the light of this, we modified a promising relationship-aware architecture, the Relation Network (RN), to extract relationship-aware features. To foster this research direction, we extended a promising synthetic benchmark for evaluating architectures on this novel and challenging task.

Processing Images and Texts for Multi-Modal Matching and Retrieval It is often difficult to fill the semantic gap by considering only a single modality; humans understand the world also by communicating concepts and complex relationships using *natural language*. For this reason, the computer vision community is showing a growing interest in the joint processing of visuals and texts as a promising tool for extracting richer semantics and abstract entity-entity relationships from the raw image pixels. In Chapter 4 we go beyond the single-modal relational image analysis presented in Chapter 3 to explore visual-textual matching, mainly for applications in large-scale text-to-image retrieval. Notably, we use the very recently introduced Transformer architecture, and in particular its Encoder module, to derive relationships between different text or image entities. Differently from the Relation Network, the Transformer Encoder discovers relationships through the powerful self-attention mechanism, that automatically weights important vectors by measuring their affinities using dot products. We show the effectiveness of the produced image and textual features when performing cross-modal retrieval, and we study and discuss their application in large-scale scenarios. In the end, we also show how Transformer-based visual-textual architectures can be used to solve other real-world critical tasks, such as detecting persuasion techniques in social networks by analyzing *memes*.

Solving Visual Reasoning Tasks Understanding relationships between visual entities is interesting on its own, as it inspires in-depth studies on both the *perception* and *reasoning* abilities of Deep Neural Network (DNN) required to solve many real-world problems. In particular, there are some challenging tasks — involving reasoning on spatial configurations of some simple shapes in an image — which are easy for humans and still extremely challenging for a state-of-the-art neural network. In Chapter 5 we tackle a very specific subset of visual reasoning problems, called *same-different* tasks, that broadly consist in understanding if two shapes in an image satisfy a specific rule that should be automatically discovered from data. Initially, we probe various state-of-the-art Convolutional Neural Networks on this task. Then, we propose a Transformer-based recurrent architecture that can solve the proposed tasks while being simpler and more data-efficient. Although these tasks involve elementary visual reasoning problems, they enable several long-term returns: visual puzzles are one of the ways to evaluate human intelligence; the resulting conclusions can be transferred to critical real-world scenarios that require more logical and decision-making skills by looking at distant and apparently unrelated inputs, from self-driving or anomaly detection for surveillance applications to the detection of anomalous patches in medical imaging.

Finally, in Chapter 6, we conclude this dissertation: we sum up our contributions, and we propose novel research directions in the field of highly semantic multi-modal information retrieval and abstract visual reasoning. Furthermore, we discuss engaging real-world scenarios where the results of this research can be applied.

CHAPTER 2

Background

In this chapter, we provide the reader with the background needed to better appreciate the core chapters of this dissertation. The main topic deserving a detailed discussion is the Deep Learning paradigm, the beating heart of the whole thesis. It is discussed in details in Section 2.1. Namely, we present the overall optimization framework and some of the regularization techniques, and the loss functions relevant for the whole discussion. In Section 2.2, we build on these pillars, introducing the main concepts behind Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), the foundations of modern visual and textual processing architectures. We also provide details on non-local processing models that capture relationships among distant entities, such as Relation Networks and Graph Networks. Building on these architectures, in Section 2.3, we present the recently introduced Transformer network. This particular attentive architecture recently unified visual and textual processing under the same framework and is very relevant for Chapter 4 and Chapter 5. In Section 2.4, we present the content-based information retrieval framework, which explains the basics for large-scale multimedia retrieval using the vector space approach. Lastly, in Section 2.5, we leave some details about the datasets used — directly or indirectly — in this dissertation.

2.1 Deep Learning

Deep Learning is a subfield of Machine Learning, more specifically of Representation Learning, in which knowledge is represented at different levels of abstraction and encoded in successive layers of a so-called Deep Neural Network (DNN). The core characteristic of Deep Learning, opposed to classical Machine Learning, is the automatic construction of the high-level representations from the low-level input data [27, 29, 72], to solve a particular task.

Deep Learning lays its foundations on the idea of Artificial Neural Networks (ANNs), a model inspired by information processing and distributed communication nodes in biological systems. Although Artificial Neural Networks have notoriously significant differences from biological brains [170, 30], many studies evidenced the many analogies [37, 118], up to the point that the inner working of Artificial Neural Networks and biological brains can be analyzed using similar methods [20].

The Deep Neural Networks used in Deep Learning are composed of a sequence — or more generally, a *graph* — of computational units called *layers*. The layers of a Deep Neural Networks are non-linear mapping functions which process the information coming from the previous layers and propagate the elaborated representation to the next ones. The term *deep* comes precisely from the fact that in such architectures there are multiple — in theory, an unbounded amount — of layers. Deep Neural Networks are the modern generalization of the perceptron [201] model, which was shown to be a universal classifier when composed of a hidden layer of unbounded width and a non-polynomial activation function [124]. These results derived from an important milestone in deep learning, the *universal approximation theorem* [83, 50], which states that a feedforward network with a linear output layer and at least one hidden layer with any squashing activation function, such as the logistic sigmoid, can basically approximate any desired function.

The idea behind Deep Learning is that data distribution in the specific domain of interest is explained by some unknown function $f^\star$. The goal is to learn the optimal function f which better approximates $f^\star$. More formally, given an input $\mathbf{x}$, the output $\mathbf{y}$ of a Deep Neural Network (DNN) can be simply formalized as

$$\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta}), \tag{2.1}$$

where $f(\cdot)$ is an arbitrary connection of non-linear parametric layers, and $\boldsymbol{\theta}$ is the vector of parameters of the network to be learned during the training phase. The function f with its parameters $\boldsymbol{\theta}$ is learned from a dictionary of data samples, called *dataset*. In the more general case, a dataset is composed of a list of N input-output pairs $\mathbf{X} = \{(\mathbf{x}_i, \mathbf{y}_i^\star), i = 1, \ldots, N\}$. The values $\mathbf{x}_i$ are samples drawn from the real data distribution, and $\mathbf{y}_i^\star$ are the expected output values — ideally obtained by evaluating the unknown function $f^\star$ on the provided data samples. The values $\mathbf{y}_i^\star$ are called *targets*, and they are usually obtained by manual annotation, although recent advances in self-supervised methods try to derive this information directly from the inputs $\mathbf{x}_i$ [101].

The process of *training* a DNN consists of updating the parameters $\boldsymbol{\theta}$ of f so that the error between the ground-truth values $\mathbf{y}_i^\star$ and the predicted outputs $\mathbf{y}_i$ is as small as possible. Formally, the optimal parameter's configuration $\boldsymbol{\theta}^\star$, considering the whole dataset $\mathbf{X}$, can be found as follows:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathbf{X}} \mathcal{L}(f(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}^\star), \tag{2.2}$$

where $\mathcal{L}(\cdot)$ is the so-called *loss* (or *cost*) function, measuring the discrepancy between the network's outputs and the target values for each of the given examples. The optimal parameters $\boldsymbol{\theta}^\star$ are the ones that minimize the error expectation over the training dataset.

In the following two paragraphs, we dive deeper into the core elements of Deep Learning. In particular, we first present the Multilayer Perceptron (MLP), the main

ingredient for most — if not all — of the modern DNNs. Then, we present some details and insights about the optimization procedure to solve Equation (2.2) practically.

2.1.1 The Multilayer Perceptron Model

The Multilayer Perceptron (MLP) model is the beating heart of basically all of the modern DNN. To understand what a Multilayer Perceptron (MLP) is, we are required to first introduce its core building block, the *perceptron*, a model developed back in 1958 [201].

The perceptron constitutes the simplest computation unit in DNNs, and it is inspired by the biological model of the *neuron*. The neuron is a very specialized cell that receives electrical stimuli as inputs through its *dendrites* and produces an all-or-nothing electrochemical pulse that can excite other neurons. This pulse, called *action potential*, is fired with the right combination of input stimuli only when the *threshold potential* is reached. Although the accurate inner-working model of the neuron is very complex, the perceptron model tries to capture the essence of the computation performed by the neuron. The electrical stimuli in input and output are modeled by means of real values, and the firing of the electrical pulse is modelled by the so-called *activation function*. Mathematically, given an input $\mathbf{x} \in \mathbb{R}^n$, the output $y \in \mathbb{R}$ is computed in the following way:

$$y = g(\mathbf{x}^\top \mathbf{w} + b) = g\left(\sum_i w_i x_i + b\right), \tag{2.3}$$

where $g(\cdot)$ is the activation function, $\mathbf{w} \in \mathbb{R}^n$ is the vector of weights associated with each input, and b is the so-called *bias*. Seen from a distance, Equation (2.3) defines the perceptron as a mathematical operator which computes a specific function to an affine transformation of the inputs.

The activation function should be a non-linear operator which models the frequency of action potentials of biological neurons. The two historically common activation functions are S-shaped functions, namely the *hyperbolic tangent* and the *sigmoid*. In recent developments of deep learning, the Rectified Linear Unit (ReLU) is more frequently used as one of the possible ways to overcome the numerical problems related to the S-shaped functions, such as the gradient-vanishing problem [71]. In recent years, many ReLU variants — such as the Leaky-ReLU or the Parametric-ReLU — demonstrated their effectiveness in image classification [237].

The Multilayer Perceptron model trivially derives from the perceptron. In particular, in Deep Learning, the perceptron is used to create *wider* and *deeper* neural networks. Concerning the network width, m different perceptrons can be used to produce m outputs by feeding them with n inputs. This model is obtained by trivially extending the mathematical formulation of the perceptron as follows:

$$\mathbf{y} = f(\mathbf{x}, \mathbf{W}, \mathbf{b}) = g(\mathbf{x}^\top \mathbf{W} + \mathbf{b}). \tag{2.4}$$

Differently from Equation (2.3), the term $\mathbf{W} \in \mathbb{R}^{n \times m}$ is a matrix of parameters encoding, in each column, the weights of the m different perceptrons. Consequently, also the bias term is extended into a vector $\mathbf{b} \in \mathbb{R}^m$. The resulting module represented by $f(\cdot)$ is called *layer* with a width equal to m. Given d different layers $f^{(1)}, f^{(2)}, \ldots, f^{(d)}$, a Multilayer Perceptron (MLP) is obtained by creating a chain $f(\mathbf{x}) = f^{(d)}(f^{(d-1)}(\ldots f^{(1)}(\mathbf{x}))$,

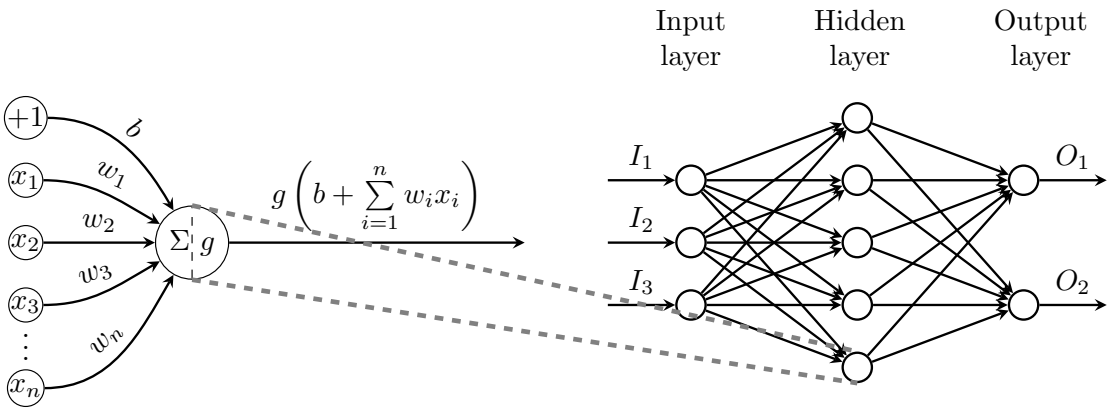


Figure 2.1: *The perceptron model (left), which is used to construct MLP with one hidden-layer (right).*

which corresponds to stacking together, in sequence, the many layers. A detailed graphical representation of the MLP is reported in Figure 2.1. In the case of MLP, the parameters $\boldsymbol{\theta}$ to be learned correspond to the set of weights and biases of the different layers. Usually, the MLP is also known as Fully-Connected Neural Network (FCNN). This term explicitly underlines the interconnections between different layers of a MLP, where every output unit is connected to all the inputs of the next one. The MLP model is widely used in many recent visual and textual processing architectures presented in the following sections.

2.1.2 Optimization

As already anticipated, training a DNN corresponds to finding the parameters $\boldsymbol{\theta}$ that significantly reduce a given cost function $J(\boldsymbol{\theta})$ defined over the training set. Given an empirical data distribution defined by the training set $\mathbf{X}$, the cost function is defined as

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},\mathbf{y}^\star)\sim\mathbf{X}}\mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}^\star), \tag{2.5}$$

where $\mathcal{L}(\cdot)$ is the loss function defined over each data sample. We emphasize that if we were given the true underlying distribution instead of the empirical one depicted by the training dataset $\mathbf{X}$, the task of obtaining $\boldsymbol{\theta}$ would be solvable by a standard optimization algorithm. This is where optimization in Deep Learning, and more generally, in Machine Learning, differs from the standard optimization approaches: we have access only to a *sample* drawn from the true underlying distribution. In the light of this observation, the objective in Deep Learning is to minimize the so-called *empirical risk*

$$\mathbb{E}_{(\mathbf{x},\mathbf{y}^\star)\sim\mathbf{X}} = \frac{1}{N}\sum_{i}^{N}\mathcal{L}(f(\mathbf{x}_i;\boldsymbol{\theta}),\mathbf{y}_i^\star) \tag{2.6}$$

which gives us a way to compute an estimate of $J(\boldsymbol{\theta})$ over the training examples.

Given the difficulty of directly minimizing this quantity due to the non-convex and non-linear nature of DNNs, it is possible to search for optimal minima in the parameters space using the *gradient descent* algorithm.

Gradient Descent Gradient descent is an iterative algorithm that searches for the local minima of a differentiable function. In deep learning, we are interested in finding the

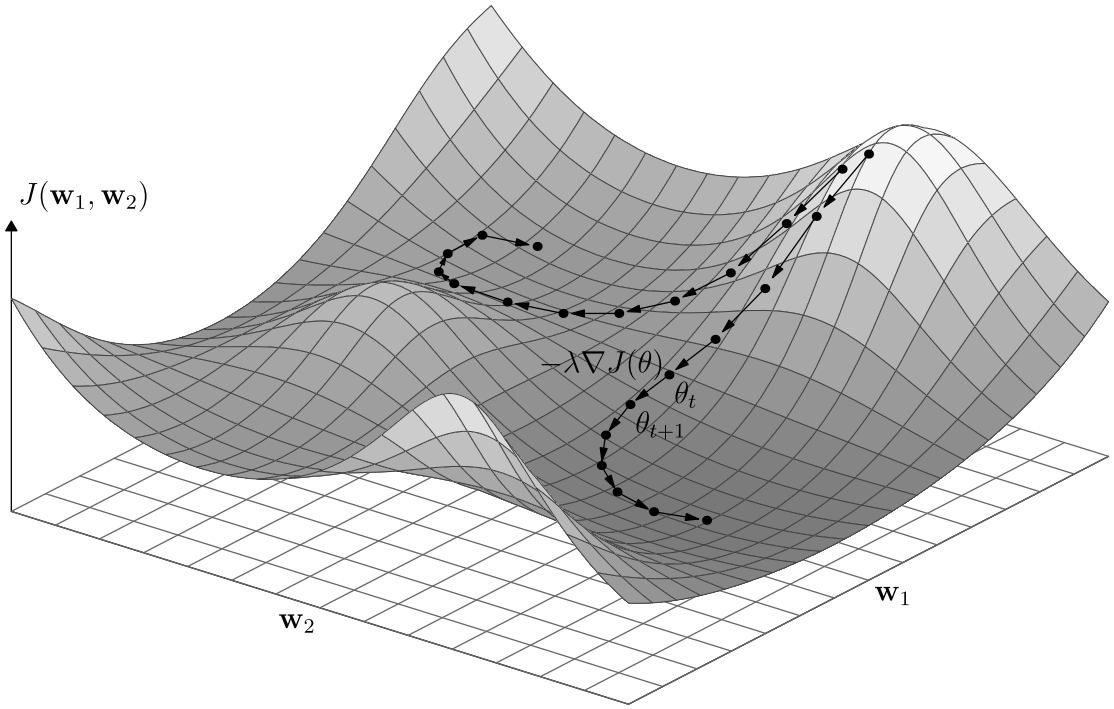


Figure 2.2: *Some iterations of the gradient descent algorithm using two different starting points in a simple parameter space $\boldsymbol{\theta} = (w_1, w_2)$. Figure re-adapted from the work in [40].*

minima of the $J(\boldsymbol{\theta})$ cost function with respect to the parameters $\boldsymbol{\theta}$. At each iteration, the gradient descent algorithm computes a step in the direction opposite to the gradient of the function evaluated in the current point:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{2.7}$$

where λ is a hyper-parameter known as *learning rate*, which directly affects the magnitude of the update at each iteration. In principle, this parameter can be estimated empirically by performing a manual search using the validation set. A too low learning rate causes a very slow convergence and can increase the probability of getting caught in local minima; on the other hand, a too high learning rate avoids a curated search of the parameter space, skipping potentially good solutions.

Although there are no global convergence guarantees in most scenarios, gradient descent is a straightforward yet effective procedure to search for optimal minima of very complex non-linear functions. An example of a few iterations of gradient descent is reported in Figure 2.2. In the case of Deep Neural Network, the gradient of the cost function $J(\boldsymbol{\theta})$ can be estimated starting from Equation (2.8) as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i}^{N} \nabla_{\boldsymbol{\theta}} \mathcal{L}(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i^\star), \tag{2.8}$$

which, in turn, shifts the problem to computing the gradients of the f function, which is a DNN, with respect to its parameters. The gradients estimation of the parameters laying in different layers of a DNN is performed by a crucial algorithm called *back-propagation*.

Back-propagation Back-propagation [202] is a widely used algorithm for computing the gradients in a Feed-Forward Neural Network. In modern Deep Learning (DL) frameworks, this algorithm has been extended to work with any architecture that can be unrolled under the form of a directed acyclic graph so that it can deal with more complex network topologies like recurrent networks (see Section 2.2.2). Suppose we are given a Feed-Forward Neural Network $f(\mathbf{x};\boldsymbol{\theta})$ composed of L layers. The objective of the back-propagation algorithm is to compute the gradients $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ with respect to the parameters through all the layers. According to Equation (2.8), the head of the computation starts from the scalar error value, or *loss*, provided by the network: $e(\boldsymbol{\theta}) = \mathcal{L}(f(\mathbf{x};\boldsymbol{\theta}),\mathbf{y})$. Therefore, if $\mathbf{o} = f(\mathbf{x};\boldsymbol{\theta})$ is the output of the network, the gradient $\frac{\partial e(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ can be re-written using the chain rule as

$$\frac{\partial e(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial e(\boldsymbol{\theta})}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial \boldsymbol{\theta}}, \tag{2.9}$$

where $\frac{\partial e(\boldsymbol{\theta})}{\partial \mathbf{o}}$ depends on the specific implementation of the loss function, while $\frac{\partial \mathbf{o}}{\partial \boldsymbol{\theta}}$ can be found by recursively applying the chain rule of calculus backward through the layers of f. Generalizing Equation (2.9), we can recursively apply this differentiation technique, rising all the network layers from its head up to the inputs. More formally, let f be composed of the L layers $\{f^{(1)}, f^{(2)}, \ldots, f^{(L)}\}$, and let $\mathbf{o}_i$ be the output from the i-th layer. Then the Feed-Forward Neural Network can be described as follows:

$$\mathbf{o}_0 = \mathbf{x} \tag{2.10}$$

$$\mathbf{o}_i = f^{(i)}(\mathbf{o}_{i-1};\boldsymbol{\theta}_i), \tag{2.11}$$

where $\boldsymbol{\theta}_i$ is the set of parameters associated with the i-th layer. We can apply the chain rule to find the derivative of the loss value with respect to the parameters laying in the i-th layer by computing

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_i} = \frac{\partial \mathcal{L}}{\partial \mathbf{o}_i} \cdot \frac{\partial \mathbf{o}_i}{\partial \boldsymbol{\theta}_i} \tag{2.12}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{o}_L} \cdot \frac{\partial \mathbf{o}_L}{\partial \mathbf{o}_{L-1}} \cdots \frac{\partial \mathbf{o}_{i+1}}{\partial \mathbf{o}_i} \cdot \frac{\partial \mathbf{o}_i}{\partial \boldsymbol{\theta}_i}. \tag{2.13}$$

The derivatives $\frac{\partial \mathbf{o}_{i+1}}{\partial \mathbf{o}_i}$ and $\frac{\partial \mathbf{o}_i}{\partial \boldsymbol{\theta}_i}$ are known since they depend on the specific implementation of the i-th layer of the network, and they can be computed using the outputs collected during the forward pass with the actual parameters $\boldsymbol{\theta}$. This procedure works only if all the network layers are implemented with differentiable functions — or as a combination of differentiable functions. Usually, this is the case, as the core element of DNNs, the perceptron, is fully differentiable except in zero if the Rectified Linear Unit (ReLU) activation function is used. However, this single point discontinuity is not a problem since we can arbitrarily assign one of the two possible values for the derivative — either 0 or 1. Once the gradients are computed for all the parameters, the back-propagation algorithm stops. At this point, the actual training process is carried out by the *optimizer*, which modifies the parameters on the basis of the computed gradients.

Epochs and Minibatches In Equation (2.8) we described how the gradient is estimated over the empirical distribution defined by all the training examples. However, in most

machine learning scenarios, only a subset of training samples is used to estimate the gradients. This subset is usually known as a *minibatch*, and it is composed of randomly extracted elements from the training set without replacement. A minibatch also defines the number of training samples to be processed before computing one gradient descent step. An *epoch* is a training cycle that ends when all the training samples have been processed — which is when every sample has been included in exactly one minibatch. The minibatch strategy for gradient estimation has some highly desirable effects. First of all, recall that the standard error of the gradient mean estimated from N samples in Equation (2.8) is given by $\frac{\sigma}{\sqrt{N}}$, where σ is the true standard deviation of the value of the samples. The denominator $\sqrt{N}$ shows that there are less than linear returns to using more examples to estimate the gradient. For this reason, minibatches make the computation more efficient while producing an adequate estimation for the gradient. On the other hand, the noise introduced in the gradient estimation has some nice effects during the actual optimization phase, as it increases the probability of escaping local minima [212].

Optimizers Optimizers are an important tool in the optimization ecosystem of Deep Learning frameworks. They directly implement the gradient descent iterative algorithm, which updates the parameters in the negative gradient direction as described in Equation (2.7). However, the criteria used for updating the weights given the gradients is not fixed beforehand. Instead, many variants of Equation (2.7) have been introduced. It has been observed that optimizers have an important role in the generalization abilities of Deep Neural Network. Over the years, many weight update techniques have been proposed, giving rise to a large corpus of different optimization algorithms. Stochastic Gradient Descent (SGD), and in particular minibatch SGD [202], is one of the largely used optimizers. Unlike the basic formulation of gradient descent in Equation (2.7), SGD computes the gradients over a minibatch, updating the weights once every minibatch instead of once every epoch. The most important variation of the SGD is the addition of the momentum term [186]. In fact, vanilla SGD has problems navigating ravines — areas in the surface curve much more steeply in one dimension than in another. Therefore, the momentum adds some inertia to the gradient to dampen out undesired oscillations:

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{2.14}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{v}_t, \tag{2.15}$$

where γ is a hyper-parameter controlling the amount of momentum to use during gradient descent. More recently, the Adaptive Moment Estimation (Adam) optimizer has been introduced. Differently from SGD, Adam computes adaptive learning rates for each parameter, similarly to other optimizers like Adagrad [62], Adadelta [250], and RMSprop. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients, similar to momentum:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{2.16}$$

$$\mathbf{v}_t = \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1) (\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))^2, \tag{2.17}$$

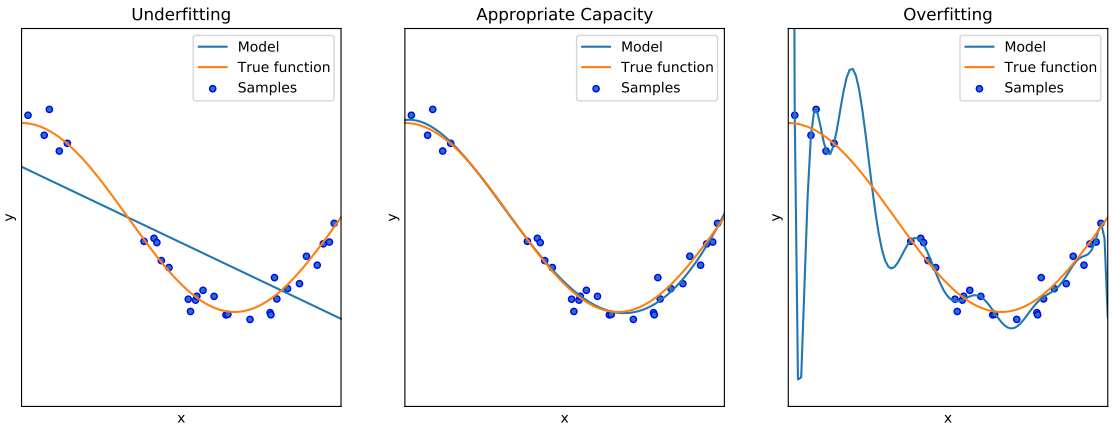


Figure 2.3: *Underfitting and overfitting on toy data.*

where $\mathbf{m}_t$ and $\mathbf{v}_t$ are the first- and second-order moments of the gradient, respectively. These moments are biased towards zero, especially in the first iterations. For this reason, the authors decided to introduce the bias-corrected moments' estimates:

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \tag{2.18}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}. \tag{2.19}$$

$$\tag{2.20}$$

At this point, the parameters update rule for the Adam optimization algorithm is as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}\hat{\mathbf{m}}_t, \tag{2.21}$$

where ϵ is a small number that prevents the division by zero, and η is the general learning rate. Adam works well in practice and compares favorably to other adaptive learning-method algorithms. At the time of writing, Adam is the primarily used optimizer; it demonstrated very fast convergence while still maintaining very good generalization abilities. Very recently, however, recent research papers have noted that Adam can fail to converge to an optimal solution under specific settings [111]. AdaBound [149] tried to solve this issue by employing dynamic bounds on learning rate, filling the gap between the speed of Adam and the generalization abilities of SGD.

2.1.3 Regularization

Until now, we defined how to update the network parameters using the samples from the empirical distribution, which is defined by the data points in the training set. However, we need to reinforce that the central problem in Machine Learning is not to achieve a good performance on the training set but to construct a model able to *generalize* to unseen data. All the strategies to achieve this goal are known as *regularization* techniques. A good regularizer is the one that trades increased bias for reduced variance. In general, given a family of models or architectures, three possible scenarios are possible: (1) the model family being trained excluded the true data-generating distribution,

incurring in a high induced bias; (2) the model family fits the true data-generating distribution just right; (3) the model family fits the generating process but also many other possible generating processes, bringing to high variance and low bias.

The limit cases (1) and (3) are also known as *underfitting* and *overfitting* regimes graphically explained in Figure 2.3. Regularization, in particular, refers to all the techniques which try to move the architectures from scenario (3) back to scenario (2). Searching directly for the model with the right size is far from being easy. Real-world generating processes — the ones that underlie images, texts, or audio — are way too complex to make a reasonable deduction about an appropriately fitting model family. Instead, in practical scenarios, the best fitting model (in the sense of minimizing/generalization error) is a large model that has been regularized appropriately.

Weight Decay One of the widespread regularization techniques, called *weight decay*, works by introducing some constraints to the norm of the parameter values. This kind of regularization is a simple yet effective way to limit the capacity of the model. Intuitively, the main objective is to discourage the model from learning excessively large parameters, which could cause steep slopes in the learned function (as shown in the rightmost scenario in Figure 2.3) that directly brings to an overfitting regime. To this aim, weight decay works by adding a penalty $\Omega(\boldsymbol{\theta})$ to the task-specific loss function:

$$\hat{J}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = J(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) + \alpha\Omega(\boldsymbol{\theta}), \tag{2.22}$$

where $\alpha \in [0, \inf)$ is a hyperparameter that weights the relative contribution of the norm penalty term. The most commonly used type of weight decay is the $\mathbf{L_2}$ weight decay [169], where $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta}||_2^2$. As studied in [169], $\mathbf{L_2}$ weight decay produces a more uniform utilization of all the available parameters, penalizing under-utilized and over-utilized weights.

Dropout Dropout [213] is a regularization technique that acts on the neural activations directly inside the model. In particular, a certain number of activations from a DNN layer are randomly ignored or *dropped out* during training. This per-neuron random deactivation is guided by sampling from a Bernoulli distribution, with a probability p chosen a priori. The dropout reduces the co-adaptation of neurons, which in turn brings the network to learn different yet redundant representations, increasing the network generalization. During inference, the random deactivation is disabled so that all the neurons can contribute to the final augmented representation. Dropout can be thought of as a method of making model ensemble practical for large DNNs. In fact, model ensembling has been proved to achieve very good generalization results [35], at the expense of higher computational complexity and memory consumption. From this perspective, dropout solves this problem by simulating, with a single model, a large number of different network instantiations.

Dataset Augmentation Given that generalization heavily depends on the number of available training samples, dataset augmentation is a simple yet effective way to produce new data from existing ones. This method is especially easy to apply on images: dataset augmentation randomly applies geometric transformations — rotations, flips, crops, shrinking — to the input images. Pushed to the limit, augmentation also consists

Figure 2.4: *The original image (top-left) with a set of strong augmentations, altering both image geometry and pixel values. From* `https://github.com/aleju/imgaug`.

of the superimposition of photographic filters, the addition of noise, or the deliberate deletion of small chunks from the image (Figure 2.4).

Augmentation does not change the underlying semantics on classification tasks, and the provided labels do not need to be modified. In tasks like object detection, instead, the labels — made up of the bounding boxes enclosing the different objects — should be changed accordingly to the geometric transformation applied to the images.

Parameter Sharing The easiest way to constrict our search for the right family of model architectures is to exploit some prior knowledge about the specific task we want to solve. In particular, from knowledge of the domain and model architecture, we might know that there should be some dependencies between the model parameters. A popular way to fix a constraint between parameters is to force a given set of parameters to be *equal*. The most obvious and striking consequence of this constraint is that the free model parameters decrease, thus making the model lighter and less prone to overfitting. Secondly, parameter sharing imposes knowledge priors to the model so that the space of possible solutions is drastically pruned. This kind of regularization is essential for understanding and generalizing knowledge in very specific domains. For example, Convolutional Neural Networks (CNNs) [121] process image patches using kernels with shared weights along the *spatial* dimensions to enforce the translation invariance prior — e.g., a cat remains a cat even after being moved in the image. Similarly, recently introduced Transformers [220], as well as Graph Neural Network (GNN) [257] share weights among the input vectors, to enforce the permutation invariance prior. Parameter sharing, in the end, is one of the keys for achieving combinatorial generalization, and the resulting *relational inductive biases* [23] can facilitate learning about abstract entities, relations, and rules.

2.1.4 Common Loss Functions

In Section 2.1.2 we discussed how a DNN could be trained given a model, some data, and a loss function $\mathcal{L}$ defined over every data sample. In this section, we dive into some common loss functions used in real-world DL applications and useful in this dissertation. The loss function mathematically represents the objective that we want to reach and, in turn, defines the *task* that we want to solve.

Cross-Entropy Loss Cross-entropy is the standard loss function employed in *classification* problems. The cross-entropy measures the performance of a classification model

whose output is a discrete probability distribution over C classes. More formally, cross-entropy is defined as the divergence between two probability distributions — the empirical one $q(x)$ produced by the network and the target one $p(x)$. In particular, it computes the average number of bits required to encode data having distribution $p(x)$ when we use instead a distribution $q(x)$:

$$H(p, q) = -\sum_{x} p(x) \log(q(x)) \tag{2.23}$$

Applied to neural network classifiers, for a single sample, it assumes the following form:

$$\mathcal{L}_{XE}(\mathbf{y}, \mathbf{y}^{\star}) = -\mathbf{y}^{\star} \cdot \log(\mathbf{y}), \tag{2.24}$$

where $\mathbf{y}$ is the estimated distribution over C classes and $\mathbf{y}^{\star}$ is the ground-truth vector. The estimated distribution $\mathbf{y}$ can be easily obtained from the activations of the last layer of a network, by using the softmax$(\cdot)$ function:

$$\text{softmax}(\mathbf{y})_i = \frac{\exp(\mathbf{y}_i)}{\sum_j \exp(\mathbf{y}_j)}, \tag{2.25}$$

which rescales the elements of the C-dimensional output so that they lie in the range $[0, 1]$ and sum to 1. This loss is widely used for solving tasks like image classification [117], object detection [198, 195], and Visual Question Answering (VQA) [205, 174, 106].

Mean Squared Error Loss Mean Squared Error (MSE) is used for *regression* problems. In many cases, it is useful to force the model to output vectors that should be as near as possible to a given target real-valued vector. In this case, the Mean Squared Error (MSE) is defined as

$$\mathcal{L}_{MSE}(\mathbf{y}, \mathbf{y}^{\star}) = \frac{1}{2}||\mathbf{y} - \mathbf{y}^{\star}||^2, \tag{2.26}$$

where $\mathbf{y}$ is the network's output, and $\mathbf{y}^{\star}$ is the ground-truth real-valued vector. The scale factor $\frac{1}{2}$ is used for simplifying the gradient computation. This loss is largely used for predicting the bounding box coordinates in object detectors [198, 195] or reconstructing images by regressing on the pixel values [243].

Triplet Loss Unlike MSE and cross-entropy losses, whose objective is to learn to reproduce the labels or the vectors provided as targets, triplet loss is used to predict relative distances between inputs. For this reason, the triplet loss is usually employed to perform *metric learning*. In metric learning, the objective is to construct a space equipped with a given *metric* (e.g., the Euclidean metric), where similar elements lay near together while dissimilar elements are placed far apart. This task requires that every element of the dataset is annotated with a distance or a similarity score. Often, this score is binary, meaning that a pair of elements from the dataset are either completely similar or absolutely dissimilar. This formulation has very intuitive use cases. For example, suppose every point represents an image of a person. In that case, this loss can easily solve the *re-identification* problem: the vectors associated with the same person are placed near

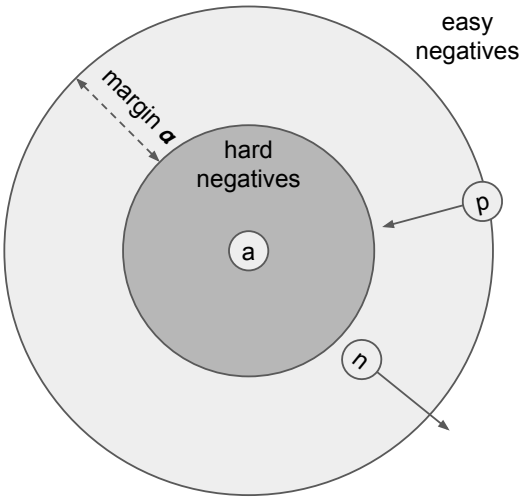


Figure 2.5: *A visual representation of the dynamics operated by the triplet loss. Positive examples are pulled towards the anchor, while the negatives are pushed away, so that there is a margin of α between the farthest positive and the nearest negative.*

in the space, while vectors from different persons are distant. Given a query person, we can perform a K-nearest neighbors (KNN) search in the resulting space to find the most similar persons — hopefully finding images of the same person in the first positions. The triplet loss uses *triplets* of points from the training set; these triplets are composed of an *anchor* point $\mathbf{x}_a$, a *positive* example $\mathbf{x}_p$, and a *negative* example $\mathbf{x}_n$. The triplet loss aims to increase the similarity between the anchor point and the positive example while decreasing the similarity between the anchor point and the negative example. Formally,

$$\mathcal{L}_{\text{tri}}(\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n) = \max(0, \alpha - S(\mathbf{x}_a, \mathbf{x}_p) + S(\mathbf{x}_a, \mathbf{x}_n)), \tag{2.27}$$

where α is a margin that controls the minimum gap between the farthest positive and the nearest negative. The dynamic operated by this constraint is depicted in Figure 2.5. Instead, $S : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a similarity function taking two points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, and returning their similarity score. Usually, the output score is a value laying in $[0, 1]$. The most widely adopted similarity function is the cosine similarity, defined as

$$S_{\cos}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{||\mathbf{x}_1||||\mathbf{x}_2||}. \tag{2.28}$$

The choice of the negative selection policy is critical for the success of this kind of learning. The trivial solution consists in choosing negatives at random. Although being a simple policy, it usually yields suboptimal solutions: many negatives are too easy, and they cannot push the discriminative power of the model to the limit. Therefore, the *hard negative mining* policy is usually used: the one closest to the anchor is chosen among all the negatives. The search for the hard negative is often performed inside the training minibatch for performance reasons and not on the whole training set. These concepts are widely employed in Chapter 4, where a variation of the triplet loss is employed in a multi-modal scenario to learn a common embedding space.

2.2 Visual and Textual Processing Architectures

In Section 2.1, we provided the reader with a general understanding of the sophisticated Deep Learning framework. In this section, we move a step further, presenting the most common and widely adopted models able to process the most widespread media types — *images* and *texts*. Being both these modalities of primary importance for the whole dissertation, it is worth spending some time on the most important and recent architectures.

2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a particular kind of Feed-Forward Neural Network particularly suitable for processing data having a known grid-like topology. Many raw digital data coming from the sensory world have this format. For example, digital images are nothing more than 2-D arrays of pixels, and time-series — like audio signals — can be considered 1-D arrays of samples. The term *convolutional* refers to the mathematical concept of *convolution*, which is, in information theory, a widely adopted operation used to process this kind of sensory data. A Convolutional Neural Network (CNN) is a network that uses the convolution operation in at least one of its layers.

Mathematically speaking, the convolution operation defined over a discrete-time input sequence $X[t]$ is defined as

$$S[t] = (X * K)[t] \sum_i X[i]K[t-i], \tag{2.29}$$

where $K[t]$ is the so-called *kernel* of the convolution. The kernel can be thought of as a fixed discrete-time function applied to a window sliding over the input sequence. If the input sequence is 2-D, as in the case of images, the formula in Equation (2.29) can be easily extended by using two summations, together with a 2-D kernel function. Therefore, given an image $I[i,j]$ indexed using row indexes i and column indexes j, the convolution operation is defined as

$$S[t] = (I * K)[i,j] \sum_m \sum_n I[m,n]K[i-m, j-n]. \tag{2.30}$$

Although the term *convolution* has been widely adopted in the DL community, the actual operation implemented in CNNs is called *cross-correlation*. Indeed, the definition of the cross-correlation operation is very similar to the convolution; however there is a foundamental difference: cross-correlation does not flip the kernel with respect to the input sequence:

$$S[t] = (I * K)[i,j] \sum_m \sum_n I[m,n]K[i+m, j+n]. \tag{2.31}$$

With this new definition, the cross-correlation loses the commutativity property, which is not a mandatory requirement for the theoretical foundations of CNNs. However, following the trend, for the rest of the dissrtation we refer to the cross-correlation operation simply as *convolution*. A graphical example of 2-D convolution is shown in Figure 2.6.

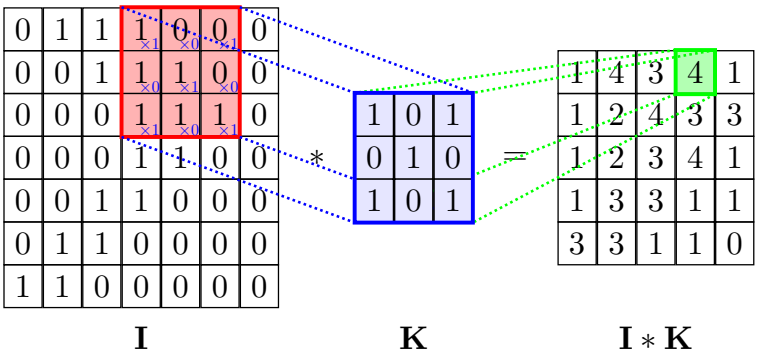


Figure 2.6: *A trivial example of 2-D convolution. The input grid of size* 7×7 *is processed by a* 3×3 *kernel to produce an output feature map of size* 5×5.

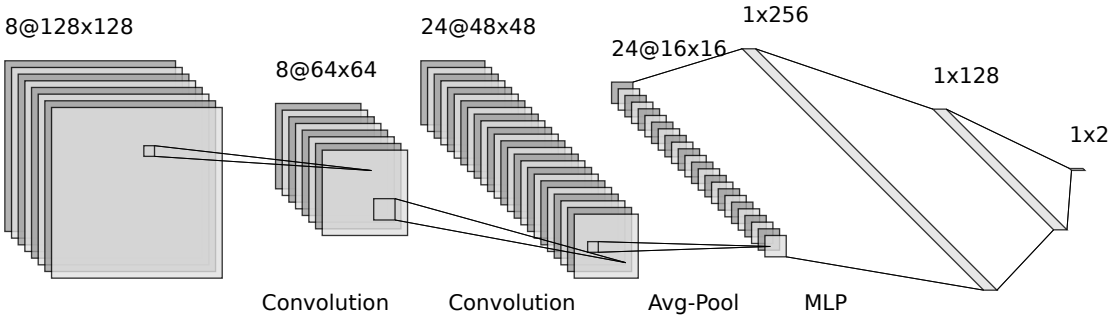


Figure 2.7: *An example of a CNN with two convolutional layers, an average pooling layer, and a final three-layer MLP to produce the probabilities over two output classes.*

In practical implementations, convolutional layers in a CNN are composed of several kernels, each processing the same input image. Different kernels, in fact, can capture different features of interest from the image and propagate the acquired information to deeper layers for further processing. Each output image resulting from applying a single kernel to the input image is usually called *feature map*; thus, each layer outputs a certain number of feature maps, more precisely one for each applied kernel. Also, different from the strict mathematical definition, in CNNs the convolution can be further customized by acting on the so-called *padding* and *stride* — parameters that control the amount of input padding and the amount of movement of the kernels respectively.

Usually, going deeper, the features map spatial resolution is decreased while the number of kernels is increased so that features maps lose their spatial resolution on behalf of higher-level and more abstract representations. For the classification task, the final spatial information in output from the last convolutional layer is collapsed to a single vector by either flattening or average pooling, and a MLP produces the logits over the output classes [117]. A graphical representation of a classical CNN network is shown in Figure 2.7.

Sparse Connections and Parameter Sharing One of the winning characteristics of CNNs is their use of sparse connections, which simplify the computational graph and provide an essential architectural prior: the *spatial locality* prior. Distant pixels are not usually directly correlated, especially when searching for low-level features like edges,

contours, or corners; for this reason, it makes perfect sense to process together only nearby pixels. Thus, the convolution operation can be implemented using *perceptrons* that are stimulated by only neighboring pixels — the ones in the grid defined by the kernel. Another important peculiarity of CNNs is the massive use of parameter sharing. Practically speaking, the sliding kernel is implemented by simultaneously processing all the possible windows using many perceptrons with tied weights along the spatial dimensions. This constraint forces the CNN layers to be *invariant* to translation: intuitively, the network is discouraged from embedding positional information into the kernel weights.

Pooling Convolutional layers can be optionally interleaved by *pooling layers*. The pooling operation has the role of diminishing the spatial resolution of feature maps by aggregating pixels using a symmetric function — sum, average, or maximum. Pooling is often a very aggressive way to aggregate spatial information. For this reason, in many modern implementations, pooling is progressively less used in favor of convolutions with higher stride or learnable pooling layers [249]. Pooling layers are widely used at the end of the CNN for obtaining a flattened vector suitable for the final MLP classifier [138, 80].

Widely-used CNNs Many different CNNs have been proposed in the literature for solving the image classification task — one of the core tasks in modern computer vision. The first Deep Convolutional Neural Network to defeat classical approaches on the classification task was the AlexNet [117]. Alexnet is comprised of five convolutional layers and a final three-layer MLP outputting logits over the output classes. It uses ReLU activations as non-linearities, and it is composed of around 80 million parameters. It achieves a Top-5 error rate of 15.3% on ImageNet. Immediately after the success of this straightforward network at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, many variants were proposed. Inception [218], also known as GoogLeNet, won the ILSVRC in 2014. Different from Alexnet, the Inception network analyses an input feature map using different convolutional layers with different kernel sizes and strides. In the end, it concatenates the obtained representations to produce the output feature map. With the use of parallel convolutions — together with the large amount of max-pooling and 1-D convolutions to reduce the features map sizes — the Inception (Version 1) network is able to achieve better results at lower costs — it is comprised of only 5 million parameters, 12 times less than AlexNet. Together with Inception, also the VGG network [138] participated in the challenge in 2014, securing the 1st runners-up position. Despite not winning the competition, VGG architecture was appreciated and became one of the most popular image classification models. VGG takes its name from the group that developed it, the Visual Geometry Group from the University of Oxford. The main characteristic is that, instead of using large-sized filters like AlexNet and ZFnet, it uses several 3×3 kernel-sized filters consecutively. The downside of this network is that it has a lot of parameters — about 138 million — so it is quite expensive to train. There exist two major versions of the Visual Geometry Group (VGG), the VGG-16 and VGG-19, composed of 16 and 19 layers, respectively. Residual Networks (ResNets) [80] won the ImageNet competition in 2015, and they defined a major milestone in image processing. ResNets solve an important problem present in

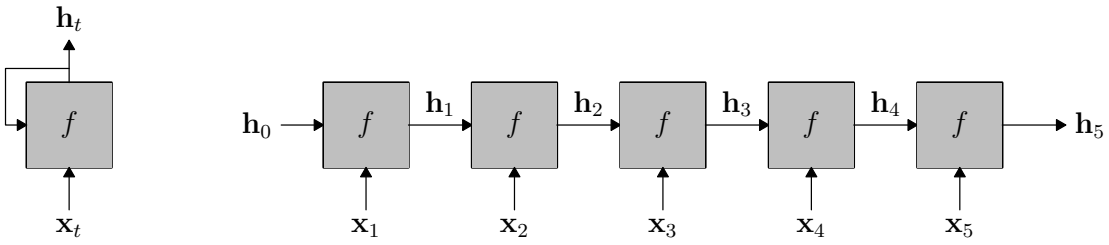


Figure 2.8: *A blueprint of a Recurrent Neural Network (RNN). The schema on the right is the unrolled version of the recurrent block on the left.*

almost all the previously presented architectures. Increasing the network depth has the undesired effect of augmenting the chances of overfitting. Although there usually exists an optimal network depth that perfectly fits the requirements of the downstream task, it is not usually known beforehand. Residual Networks solve this problem by enabling each layer to learn the increments with respect to the inputs $\mathbf{x}$ to obtain the next feature map $\mathbf{y}$: $\mathbf{y} = \mathbf{x} + f(\mathbf{x}, \boldsymbol{\theta})$. This is implemented by adding *residual* skip connections to the computational graph. With this trick, a layer $f^{(l)}$ of the network can easily learn the identity transformation — putting its parameters $\boldsymbol{\theta}_l = \mathbf{0}$ to bypass the computation. There exist different variants of the ResNet architecture, obtained by varying the number of residual modules. The most famous ones are the ResNet-18, the ResNet-50, ResNet-101 and ResNet-152. A major variation of ResNets, called ResNeXt [236] secured the 1st place in the competition in 2016, settling down the Top-5 error rate to 4.1%.

2.2.2 Recurrent Neural Networks

While CNNs are suitable for processing grid-like data, Recurrent Neural Networks (RNNs) are architectures meant to process *sequences*. RNNs are of great importance as they are widely used to process natural language text as a sequence of words.

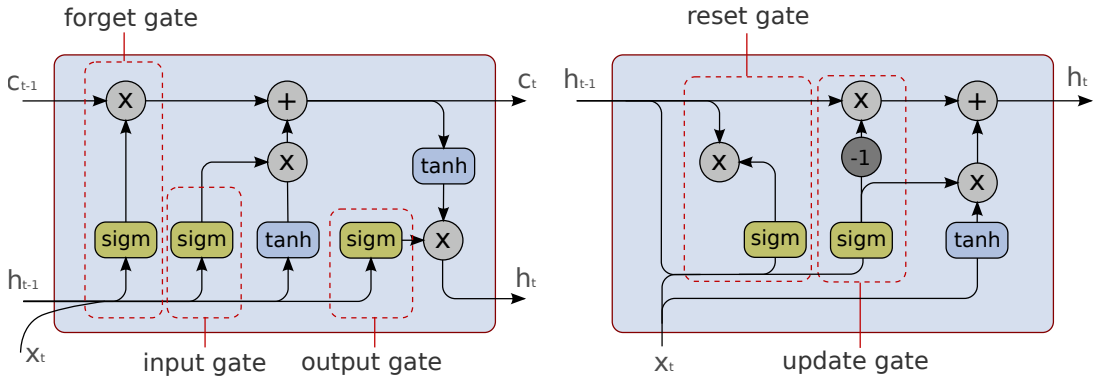
Unlike Feed-Forward Neural Networks, RNNs are stateful architectures. The output depends not only on the given input, but also on the actual internal state — called also *hidden* state — of the network [72]. The mathematics driving RNNs is very similar to the one describing discrete-time non-linear dynamical systems: given a sequence $S[t]$ of vectors $\mathbf{x}_t \in \mathbb{R}^n$, the output $\mathbf{y}_t \in \mathbb{R}^m$ at each timestep t is given by

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{2.32}$$

$$\mathbf{y}_t = g(\mathbf{h}_t), \tag{2.33}$$

where f, g are non-linear functions implemented by DNNs, and $\mathbf{h}_t \in \mathbb{R}^s$ is the hidden state. Notice how the dimensionalities of the different vectors involved $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{h}$ can in principle be different and depend on the actual implementation of the f and g functions. Usually, many core implementations assume the output to be directly the hidden state (i.e., they assume g to be the identity function). The g function is then actively employed in many downstream applications, like text generation, to produce meaningful word embeddings from the network's hidden state.

This mathematical description can be graphically visualized by adding a *recurrent* connection to the f function block. Instead, the underlying computational graph is



(a) *LSTM* **(b)** *GRU*

Figure 2.9: *The cell architectures of GRUs and LSTMs. Note how the GRU eliminates the need of the cell state c_t, in favour of the hidden state only.*

always *unrolled*, and it is often used to understand better how the gradients can be computed when recurrent connections are present. The two graphical representations are reported in Figure 2.8. Given that the functions f and g are fixed for every element of the sequence, the weights of the functions f and g are shared over the different timesteps. This is very similar to the parameter sharing adopted in CNNs, except that here it happens along the *time* dimension. Parameter sharing, in this case, makes the network weights independent of the actual timestep, which is a highly desirable property to handle variable length — and virtually endless — sequences of vectors.

Widely-used RNNs In literature, many RNNs variants have been proposed, mainly leveraging different implementations for the f function. Vanilla RNNs usually refer to the Elmann implementation [64], which uses the following hidden state update function:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{b}_{ih} + \mathbf{W}_{hh}\mathbf{x}_t + \mathbf{b}_{hh}), \tag{2.34}$$

where the outputs of two perceptrons processing the input and the previous hidden state respectively — $\mathbf{W}_{ih}\mathbf{x}_t + \mathbf{b}_{ih}$ and $\mathbf{W}_{hh}\mathbf{x}_t + \mathbf{b}_{hh}$ — are summed together and then passed through a $\tanh(\cdot)$ activation function to produce the next hidden state. Despite their simplicity, vanilla RNNs suffer from two significant issues: catastrophic forgetting [114] and gradient vanishing [28]. To mitigate these problems, two architectures were proposed: Long Short-term Memorys (LSTMs) and Gated Recurrent Units (GRUs), both introducing the concept of *gates*. The LSTMs [82] main recurrent block is called *cell*, and the cell state can be considered the memory of the network. The cell of an LSTM is built of three gates: the *forget gate*, which decides what information should be thrown away or kept; the *input* gate, which decides how the input influences the actual cell state; and the *output gate*, forming the next hidden state from the cell state. Gates are implemented as neural networks with their own parameters — always shared among the different timesteps. The internal arrangement of the LSTM cell is shown in Figure 2.9a. The LSTMs demonstrated very good abilities on memorization of longer sequences with respect to vanilla RNNs, while improving the optimization due to the drastic attenuation of the gradient vanishing problem. GRUs [46] are the next-generation RNNs. GRUs are in principle very similar to LSTMs, except that they

use only the hidden state — and not also the cell state — to define the memory of the network, as shown in Figure 2.9b. The *reset* and the *update* gates serve similar purposes as the ones in the LSTM cell; however, the simplicity of the GRU cell drastically increases the network efficiency, while maintaining a competitive performance with respect to the LSTMs. In real use cases, these networks are often arranged in a *bi-directional* setup, where two different instantiations of the architecture are used to analyze the sequence; the former processes the sequence in the forward direction, while the latter in reverse. The final outputs are obtained by merging the individual output vectors from the two models.

2.2.3 The Relation Network (RN)

The Relation Network (RN) architecture [205] is the first architecture proposed to handle spatial relationships explicitly and perform visual-language relational reasoning. The RN module obtained impressive results on the Relational Visual Question Answering (R-VQA) task, which consists in answering complex relational questions on the objects' attributes and positions from the input image. It combines input objects forming all possible pairs, and it applies a common transformation to them, producing activations aimed to store information about possible relationships among input objects.

For the specific task of R-VQA, authors used a small CNN to learn $N \times N$ visual object representations of size C, that are then fed to the Relation Network (RN) module and combined with the textual embedding of the question produced by an LSTM, conditioning the relationship information on the textual modality. Notice that, in this formulation, an *object* is each of the features from the last convolutional layer. Therefore, it could represent a relevant item in the image or simply an element from the background. Mathematically, the RN module is described by the following equations:

$$\mathbf{r} = \sum_{i,j} g_\theta(\mathbf{o}_i, \mathbf{o}_j, \mathbf{q}) \tag{2.35}$$

$$\mathbf{y} = f_\phi(\mathbf{r}), \tag{2.36}$$

where g_θ and f_ϕ are parametric functions whose parameters θ and ϕ can be learned during the training phase. Specifically, in the original implementation, they are MLPs. The features $\mathbf{o}_i$ and $\mathbf{o}_j$ correspond to the pair of C-dimensional object features in output from the CNN, and $\mathbf{q}$ is the question embedding vector obtained from the LSTM module. There are N^2 possible objects in output from the CNN; therefore the number of binary relationships is N^4. The architecture is shown in Figure 2.10. We notice from Equation (2.35) that the g_θ function shares the weights with all the pairs being processed, while f_ϕ in Equation (2.36) operates on the single aggregated vector $\mathbf{r}$.

The overall architecture composed of CNN, LSTM, and the RN core can be trained fully end-to-end, and it can reach superhuman results on the challenging CLEVR dataset, described in more detail in Section 2.5. This innovative architecture, in some ways, anticipates the self-attention mechanism of the more recent Vision Transformers, explained in detail in the following section. In fact, the mechanism put in place by the Relation Network is a simple yet effective way to produce visual contextualized vectors, where the early concatenation of the question vector to all the resulting visual couples before the application of the g_θ function works as a simple yet effective visual-textual attention mechanism.

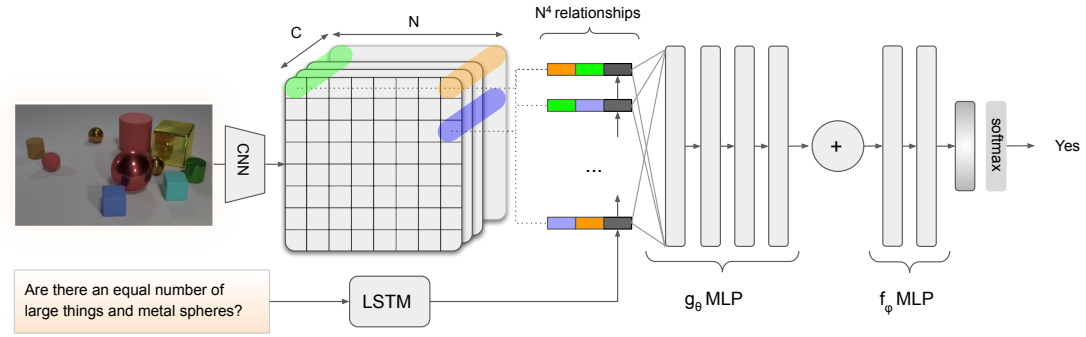


Figure 2.10: *Relation Network (RN) architecture.*

In some sense, even CNNs can construct high-level representations by merging all the activations in a predefined spatial neighborhood. Nevertheless, CNNs only allow the discovery of local relationships (i.e, the ones between a pixel or a small image patch and the neighboring ones); this is certainly useful for giving a group of pixels a high-level meaning. However, complex real-world images are usually composed of many interacting elements, where abstract interactions can happen among spatially distant actors (e.g., a football player and the ball he is going to receive). For this reason, the coupling among distant representations performed by the Relation Network helps in capturing possible binary relationships among apparently distant and unrelated elements in the image. Thanks to these insights, the RN obtained remarkable results even in other contexts outside R-VQA. The authors in [216] used the relational abilities of this module to pre-train a network for few-shot learning; in [19], the RN was used to solve Raven Matrices [253], and in [112] the authors studied the effect of this relational module in the same-different task, which is discussed in detail in Chapter 5.

2.2.4 Graph Networks

In [23] the authors demonstrated how the Relation Network architecture could be generalized to Graph Neural Networks (GNNs) [208, 229, 221], which recently acquired an important role in the processing of structured data. A GNN is a particular kind of DNN, whose underlying architecture is a *graph*. A graph is a collection $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ of vertexes $\mathbf{V} = \{v_0, v_1, \ldots, v_N\}$ and edges $\mathbf{E} \subseteq \{(u, v) | u, v \in \mathbf{V}\}$. A GNN associates vertexes and edges the embeddings $\mathbf{h}_v \in \mathbb{R}^n$ and $\mathbf{e}_{u \to v} \in \mathbb{R}^m$, respectively, and it is driven by the *message-passing paradigm* formalized as follows:

$$\mathbf{m}_{u \to v}^{(\ell)} = g_\theta^{(\ell)} \left(\mathbf{h}_v^{(\ell-1)}, \mathbf{h}_u^{(\ell-1)}, \mathbf{e}_{u \to v}^{(\ell-1)} \right) \tag{2.37}$$

$$\mathbf{m}_v^{(\ell)} = \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{u \to v}^{(\ell)} \tag{2.38}$$

$$\mathbf{h}_v^{(\ell)} = f_\phi^{(\ell)} \left(\mathbf{h}_v^{(\ell-1)}, \mathbf{m}_v^{(\ell)} \right). \tag{2.39}$$

These equations describe how a message is processed and propagated through the nodes in the graph at a certain timestep ℓ. In particular, the features $\mathbf{h}_u$ of the nodes in a neighborhood $\mathcal{N}(v)$ of the node v, are first processed by the g_θ parametric function, also called *message function*. Then, the resulting messages $\mathbf{m}_{u \to v}$ are collected at node

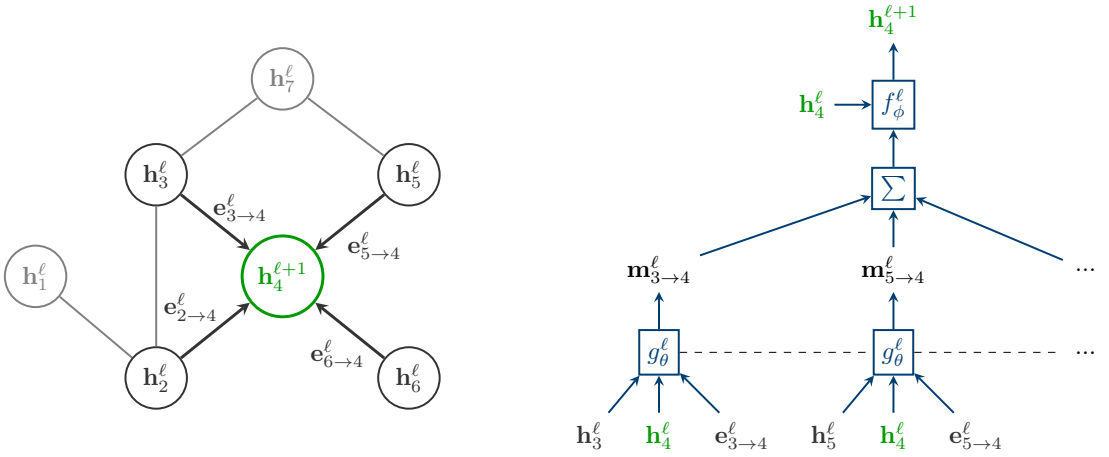


Figure 2.11: *A single step of message passing for the sample graph shown on the left. For ease of understanding, we only show the message passing on the target node $v = 4$, where the neighborhood is $\mathcal{N}(4) = \{2, 3, 5, 6\}$. The same procedure should be applied to all the other nodes.*

v and aggregated using the function $\sum$ — which is usually a summation, although it could be any symmetric function. Finally, the resulting aggregated message $\mathbf{m}_v$ is processed by the parametric function f_ϕ, also called *update function*. The resulting vector $\mathbf{h}_v$ is stored in v and propagates to its neighbors in the next message passing step. A graphical representation of this procedure is shown in Figure 2.11. The number of steps of the message passing algorithm is usually fixed, and it can be considered the *depth* of the GNN.

We can notice how the g_θ and f_ϕ functions play the same role as the ones in the Relation Network (RN) architecture. In particular, the RN can be considered as a particular kind of GNN having (a) no edge features $\mathbf{e}_{u\to v}$, (b) g_θ and f_ϕ implemented as MLPs, and (c) the underlying graph *complete* — i.e., the underlying graph has all the nodes connected with all the others.

The Graph Neural Network framework is very general, and it can be used to model other well-known architectures like CNNs. In fact, convolutional feature maps can be modeled as a grid-like graph, where (1) the sliding kernel plays the role of the message function g_θ, (2) the aggregation function is a sum (implementing the scalar product between the features and the kernel weights) and (3) the update function f_ϕ is the identity function. Nevertheless, Graph Neural Networks became the reference architecture for understanding abstract relationships between high-level concepts extracted from images rather than processing low-level pixel inputs. The RN is the first step in this direction. More recently, Graph Neural Networks have been used to *reason* on the interconnections between words in the sentences or between visual concepts in images for image-text matching [255, 137], image captioning [129], or VQA [126].

Graph Neural Network are also at the core of two important architectures proposed in the last few years. The first one, Capsule Networks [203, 81], tries to solve the problem of spatially relating parts of an image with the whole (for example, understanding that a face is a composition between some possibly distant objects — eyes, years, nose — properly positioned and oriented). The second, which grasped more attention in the Computer Vision community, is the Transformer architecture [220]. Given the role

Transformers play in this thesis, they are discussed in great detail in the next section.

2.3 Transformer Networks

The Transformer architecture [220] is a model used to process data that is natively encoded as a *set* of vectors. The Transformer sees his birth in Natural Language Processing (NLP). In fact, it has been initially proposed to address the catastrophic forgetting problem of RNNs in text traslation tasks, and more generally, in the deep comprehension of long texts. Previous works attempted to solve the problem augmenting RNNs with *attention mechanisms* [17, 113]. The attention tool enables direct weighted shortcut connections to the input tokens, in turn avoiding the necessity to store all the meaning of a long sentence in the single hidden state of a LSTM or a GRU. In RNNs the attention mechanism can be considered a sort of *plugin* which dynamically learns to weigh the contribution of each of the input tokens during the decoding phase. The attention module requires a minimal engineering effort: it is learned end-to-end, as it is fully differentiable, and it automatically adapts the per-token weights to maximize the performance on the downstream task.

Given the promising role of the attention mechanism, the authors in [220] moved a step further, demonstrating that the attention mechanism alone — without the underlying RNN — is sufficient to outperform previous architectures on translation tasks. The Transformer architecture they proposed is structurally similar to a recurrent encoder-decoder architecture [217, 17]; however, they completely redefined the implementation of the encoder and decoder modules using attention solely as the core mechanism.

More in detail, the source sequence $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_S\}$ is processed using the *transformer encoder* model, which creates a set of contextualized vectors $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_S\}$, called *memory vectors*, encoding the input sequence. Using the memory vectors, the *transformer decoder* module predicts the output representations $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$, which in turn are used to generate the probabilities over each output word. A complete scheme of the Transformer architecture for text translation is reported in Figure 2.12. At each decoding step t, the model is auto-regressive, consuming the previously generated symbols $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}\}$ as additional input when generating the next $\mathbf{y}_t$. This is the reason why, at training time, the input to the decoder module is shifted right, and a *start* token is prepended to the input target sequence. The iterative inference stops when the *end* token is obtained in output from the decoder, exactly as in RNNs-powered text generation schemes.

Notice how, during the decoding process, the decoder is conditioned, at each time step, by all the memory vectors generated by the encoder. Also, since the Transformer architecture is natively permutation invariant, the sequentiality of the input tokens is recovered by performing the so-called *positional encoding*: every token is augmented with a vector that univocally encodes its position inside the sequence. The beating heart of the Transformer architecture — where the actual attention computation is performed — is called *multi-head attention*.

2.3.1 Multi-Head Attention

The multi-head attention mechanism tries to weigh the vectors $\{\mathbf{x}\}_1^S$ of the *input* sequence by computing their affinity with each of the vectors from a *target* sequence

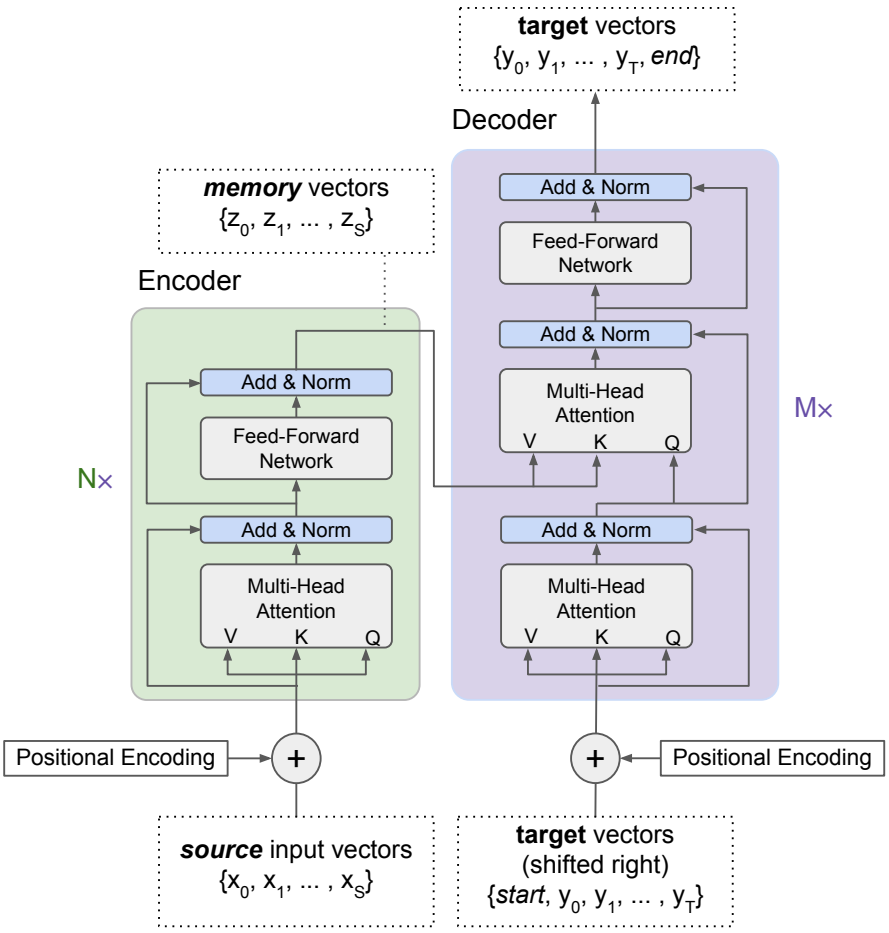


Figure 2.12: *The Transformer architecture. Note how, for the translation task, the target sequence in input to the decoder module is shifted right, and the model is initially fed with a start token. This enables the model to be autoregressive (the first $t-1$ target tokens are used during inference to predict the t one).*

$\{\mathbf{y}\}_1^T$. In sequence to sequence translation, the target sequence is the translated sequence, while the input sequence is the sequence to translate. However, this mechanism is general enough to be applied to any two arbitrary sequences, and, pushing the architecture to the limit, to any two arbitrary *sets* of vectors, given that the architecture has no hard-coded sequential priors as RNNs. In particular, three different vectors, called *queries*, *keys*, and *values*, indicated as $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^d$, are used to compute the attention. The queries are extracted from the input sequence vectors, while values and keys are extracted from the target one using linear projections, as follows:

$$\mathbf{q}_i = \mathbf{W}^q \mathbf{x}_i \quad i = 1, 2, \dots S \tag{2.40}$$

$$\mathbf{k}_j = \mathbf{W}^k \mathbf{y}_j \quad j = 1, 2, \dots T \tag{2.41}$$

$$\mathbf{v}_j = \mathbf{W}^v \mathbf{y}_j \quad j = 1, 2, \dots T. \tag{2.42}$$

Notice that the weights of the projection matrices $\mathbf{W}^q$, $\mathbf{W}^k$, $\mathbf{W}^v$ do not depend on the set indexes i or j, as these weights are shared among the elements of the sequences. This is the core insight driving the permutation invariance prior of Transformers: thanks

to this weight sharing constraint, the Transformer output sequence permutes as the input sequence. The intuition behind the introduction of queries, keys, and values quantities is that the attention works similarly to a dictionary-like memory [75]: a query from the input sequence is used to match the more relevant keys from a pre-existing dictionary — the target sequence — and the values present at the chosen memory locations are aggregated and returned.

More formally, the attention-aware vector in output from the attention module is computed for every input element as a weighted sum of the values, where the weight assigned to each value is computed as a similarity score — the *scaled dot-product* — between the query with the corresponding key:

$$\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{softmax}\left(\frac{\mathbf{q} \cdot \mathbf{k}^\top}{\sqrt{d_k}}\right)\mathbf{v}, \tag{2.43}$$

where the factor $\sqrt{d_k}$ is used to mitigate the vanishing gradient problem of the softmax function in case the inner product assumes too large values.

The term *multi-head* refers to the fact that, in real implementations, the input vectors are sliced in h equally-sized chunks. Every chunk is processed independently using h different instantiations of the above-described mechanism, where h is the number of attention *heads*. In other words, multiple instantiations of the attention mechanism act on different slices of the input and target vectors. In the end, every single output is concatenated to form the final output vector. The multi-head variation helps capture the relationships between the different portions of every input vector, and empiric studies in [220] confirm its effectiveness.

Many of the great achievements of the Transformer architecture — for example Bidirectional Encoder Representations from Transformers (BERT) [58] — were obtained using the *encoder* part of the Transformer solely. Following, we give the reader a comprehensive overview of the important Transformer Encoder (TE) architecture.

2.3.2 The Transformer Encoder Architecture

The core mechanism behind the Transformer Encoder (TE) is called *multi-head self-attention*, which is a specialized version of the multi-head attention mechanism described earlier. Self-attention, differently from plain attention, uses the same sequence both as input and target. The immediate consequence of this specialization is that $\mathbf{q}, \mathbf{v}, \mathbf{k}$ are computed starting from the same input set. In this case, $T = S$, and the scalar product $\mathbf{q} \cdot \mathbf{k}^\top \in \mathbb{R}^{S \times S}$ is a square matrix encoding the affinity that each sequence element has with all the other elements of the same sequence.

The output from the TE is computed through a simple feed-forward layer applied to the $\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v})$ vectors, with a ReLU activation function. This simple feed-forward layer casts in output a set of features with the same dimensionality of the input sequence. Two residual connections followed by layer normalization are also present around the self-attention and the feed-forward sub-modules. An overview of the Transformer Encoder architecture is shown in Figure 2.13. Given that the input and output interfaces of the TE match, multiple TEs can be stacked together to capture relationships between elements of the sequence at different levels of abstractions.

Although the TE was initially developed to work on sequences, there are no architectural constraints that prevent its usage on sets of vectors instead of sequences. In

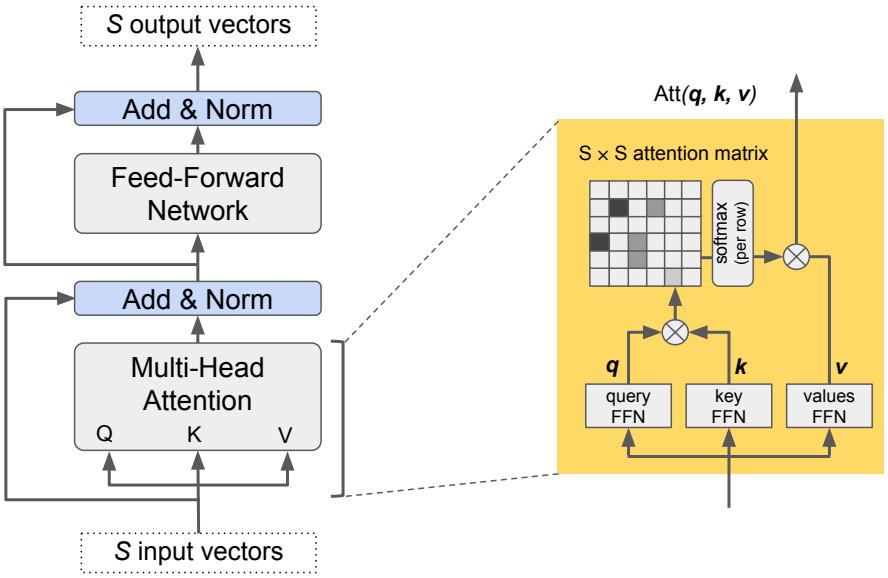


Figure 2.13: *A high-level view of the Transformer Encoder layer. Every arrow carries s fixed-sized vectors.*

fact, the TE module has no built-in sequential priors that consider every vector in a precise position in the sequence, thanks to the permutation invariance property of the multi-head self-attention mechanism.

2.3.3 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) [58] is an architecture exploiting the power of Transformer Encoders to understand and process natural language texts. BERT is a powerful language model that is pre-trained on a number of challenging language problems; the pre-trained model can then be finetuned on a specific downstream task.

BERT is pre-trained on problems that ensure a general understanding of the dependencies between words in a sentence or between the sentences in a paragraph; thus, this model becomes a good language prior that can be further refined during the finetuning phase. In particular, BERT is trained on the Masked Language Modelling (Masked LM) and the Next Sentence Prediction (NSP) tasks. In MLM, some specific tokens from the input sequence are masked out, and the model is asked to predict them. Unlike other unsupervised methods like denoising auto-encoders [224], this approach only predicts the masked words rather than reconstructing the entire input. On the other side, the NSP task consists in predicting if two input sentences are consecutive or not, and it is therefore framed as a binary classification problem.

To solve these tasks, the TE is arranged as shown in Figure 2.14. The two sequences for NSP are both given as input to the Transformer Encoder. A special [SEP] token is inserted between them to signal the end of the first and the beginning of the second, and a start-of-sequence token, called [CLS], is added to the head of the first sequence.

The NSP head — a binary classification head — is constructed on top of the output [CLS] token, which is therefore used to collect global information from the two sentences. On the other hand, many MLM heads are attached to the masked output tokens,

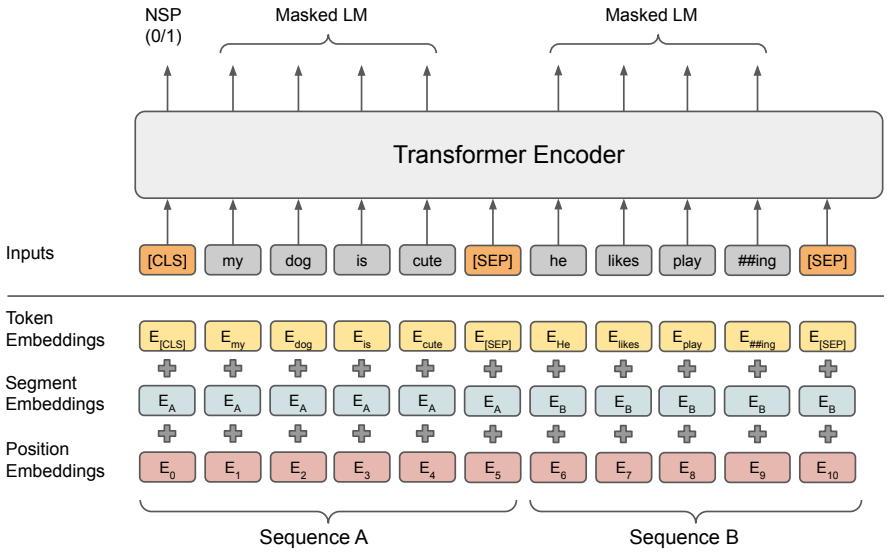


Figure 2.14: *The BERT architecture. NSP stands for Next Sentence Prediction, and Masked LM stands for Masked Language Modelling, and they refer to the meta-tasks on which BERT is pre-trained.*

and they are optimized on a standard token prediction problem.

With these simple yet effective pre-training schemes, BERT is able to outperform other language models like ELMo [177] and GPT [190] pre-trained on the same data corpus, on many downstream tasks such as textual similarity prediction, sentiment analysis, and question answering [226, 192].

2.3.4 Vision Transformers

Given the massive engagement of the Transformer architecture in the NLP community, these models have recently acquired much interest also in Computer Vision.

Transformers, and in particular Transformer Encoders, are able to process sets of input vectors that do not necessarily come from textual sequences. In fact, it is possible to subdivide images into several *patches*, which can be fed as input to a TE for further processing. The self-attentive mechanism of TEs can discover long-range dependencies between image patches, overcoming the limits of the local-processing performed by CNNs. The patches are usually extracted either by subdividing the image with a regular grid or by running an object detector like Faster-RCNN [198] to find the most salient regions (Figure 2.15).

On these simple concepts, many different architectures have been proposed. Some of these, like Cross Transformers [60] or DETR [39] use the regular grid of features from the last feature map of a CNN as visual tokens. Particularly, DETR uses the entire Transformer pipeline to approach the object detection problem, reaching comparable results with fully-convolutional architectures [195, 198]. More recently, fully-Transformer architectures, first among which the Vision Transformer (ViT) [61], have taken root. For the first time, no convolutions are used to process the input image. In particular, the ViT architecture divides the image in patches using the grid approach; the RGB pixel values from every patch are concatenated, and they are linearly projected

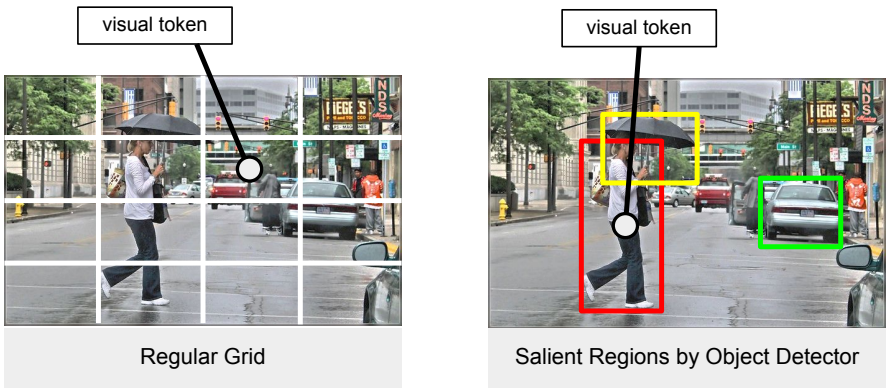


Figure 2.15: *The two approaches for obtaining visual tokens from an image. Every visual token is transformed into a vector, and the resulting set of vectors is fed to the Transformer Encoder network.*

to a lower-dimensional space to be used as visual tokens. The BERT-like [CLS] token is then used as the classification head. On the same wave of ViT, the TimeSformer [33] redefined attention both in space and time to understand long-range space-time dependencies in videos.

Vision Transformers recently outperformed ResNets and ResNeXts on image classification on ImageNet. Nevertheless, the major downside of ViTs is the large amount of data needed to train them. In fact, differently from CNNs, ViTs lack the inductive biases deriving by the local spatial processing of the convolution operation. ViTs, with their global self-attention mechanism, must learn the important inductive biases directly from data. For this reason, there is particular interest in mixing CNNs with Transformers for creating effective hybrid architectures [53, 78]. An example is the recently-released CoAtNet [52], the current state-of-the-art architecture for image classification on Imagenet, reaching a Top-1 accuracy of 90.88%.

2.3.5 Multi-modal Transformers

A multitude of methods has been recently proposed to pre-train Transformer Encoder architectures on visual-textual tasks. In fact, in the last years, multi-modal Deep Learning acquired a fundamental role, and there was an increasing need for models able to process visuals and natural language texts jointly. Given the vast amount of visual-textual tasks, the research focused on ways to pre-train visual-language models that can be further specialized on a specific downstream task during the finetuning phase.

Most of the proposed architectures follow the same wave of BERT [58], with the additional consideration that, as in Vision Transformers [61], image patches can also be treated as tokens, as much as words. First among all, the ViLBERT architecture [145] proposed a two-stream visual and textual pipeline able to jointly reason on the two modalities using co-attentional Transformer layers. Concurrently, VisualBERT [127] proposed an architecture much more in line in spirit with BERT: a Transformer Encoder is fed with both textual and visual tokens, and the output CLS and SEP tokens are used as heads for the downstream tasks, such as Visual Question Answering (VQA) or Visual Commonsense Reasoning (VCR). However, differently from BERT, VisualBERT is not pre-trained on unsupervised tasks such as Masked LM. Pre-training on this task would help such models to acquire a global a-priori knowledge of both visual and textual

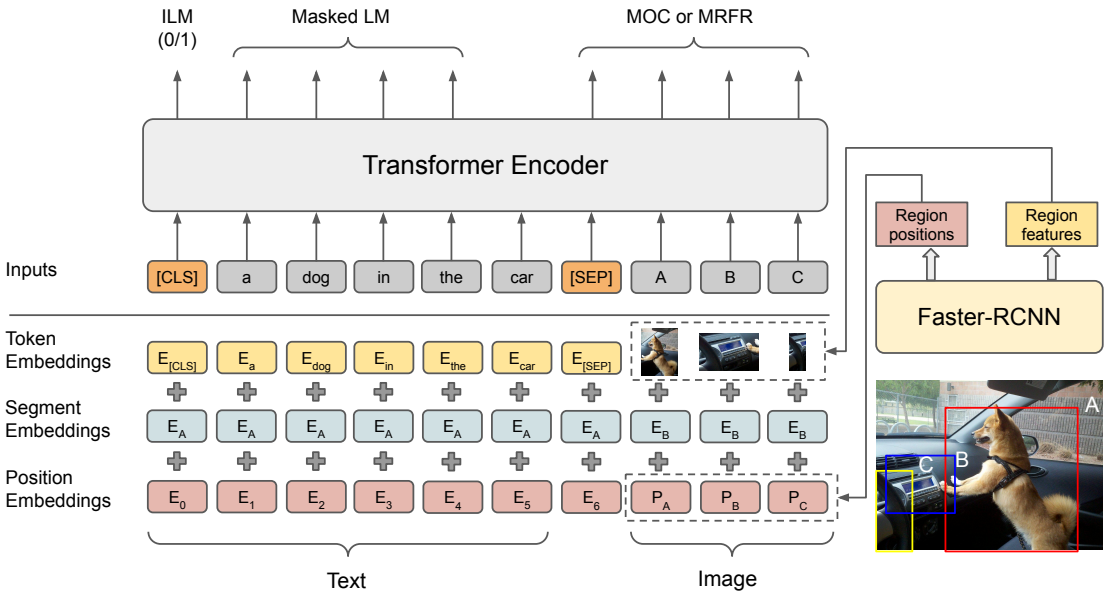


Figure 2.16: *The blueprint of multi-modal BERT-like architectures for jointly processing images and texts. ILM stands for Image Text Matching, while MOC and MRFR stand for Masked Object Classification and Masked Region Feature Regression, respectively, and they refer to some of the meta-tasks on which these architectures are usually pre-trained.*

worlds.

To this end, a new generation of multi-modal Transformers — ImageBERT [184], PixelBERT [91], VL-BERT [215], and OSCAR [131] — extended the Masked LM pre-training objective also to the visual world. In particular, ImageBERT introduced *Masked Object Classification* (MOC), where the goal is to infer the correct label for the masked image region, and *Masked Region Feature Regression* (MRFR), which consists in regressing the visual feature of the masked image region to the ground-truth region vector — the one produced by the object detector. The underlying blueprint for the architectures listed above is shown in Figure 2.16.

Similarly to BERT, these architectures can spot hidden semantic links between tokens, thanks to the power of the self-attention mechanism. For this reason, it is possible to visualize the internal multi-modal learned attentions to understand the fine-grained alignments between image regions and words. In particular, the various Transformer Encoder layers and heads capture relationships at different abstraction levels, creating a powerful relational model. An example of attention visualization is shown in Figure 2.17. All the presented methods use Faster-RCNN to detect objects and use the ROI-pooled features in output from the first stage of the detector as visual tokens. They are pre-trained on many datasets, mainly Conceptual Captions [210], which contains 3M images with descriptions harvested from the Alt-text HTML attribute of the web pages, and SBU Captions [171], which consists of 1M images with user-associated captions.

Many of these multi-modal Transformer architectures are pre-trained also on the noteworthy *Image Language Matching* (ILM) task. The objective of ILM is to decide if

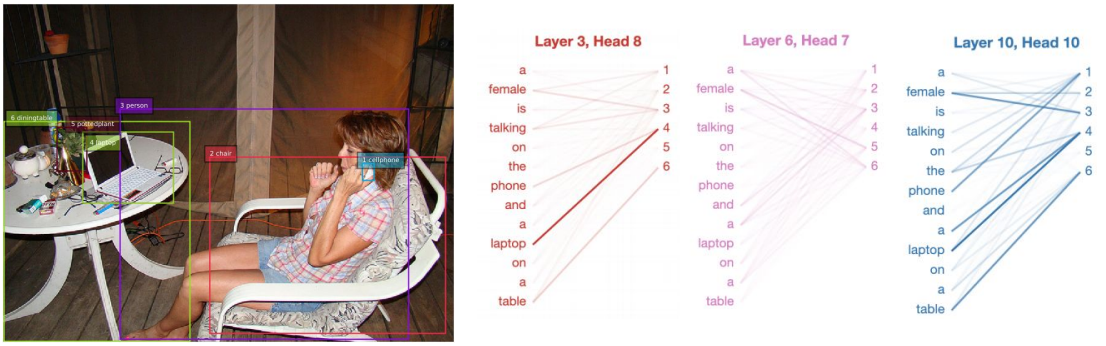


Figure 2.17: *An example of attention visualization from VL-BERT. Image by Su et al. [215].*

the image and the text in input are related or not. The problem is usually faced as a binary classification task, which returns the matching likelihood during inference. In this dissertation, the ILM task covers an important role in Chapter 4, where we discuss efficient image-text matching. Although these architectures are very effective and able to produce astonishing semantic groundings, they cannot produce a meaningful common space where efficient and scalable K-nearest neighbors search can be performed.

2.4 Content-Based Multimedia Information Retrieval

In this section, we review some of the key concepts related to multimedia content retrieval, which are of high relevance in Chapter 3 and Chapter 4.

From a broad perspective, *Multimedia Information Retrieval* is a wide area of computer science involved in the research and development of methods for effectively and efficiently searching the huge variety of media available in different kinds of repositories — digital libraries, social networks, multimedia databases. Given the massive amount of multimedia content uploaded every day to the web, it is essential to develop systems able to organize all these data to be easily retrieved. Albeit there are today many different paradigms for making the human-machine interface during the search phase as simple as possible, the core idea in Multimedia Information Retrieval remains unchanged: given a user-provided *query*, we are requested to retrieve the most relevant *items* available in a given collection (or *database*). Historically, the first large-scale retrieval systems were built for searching textual documents only; for this reason, in the early days of information retrieval, databases contained only textual files, usually referred to as *documents*. Today, the possibility of having large collections of images, videos, and audio besides text significantly extended the limits and areas of competence of Information Retrieval, giving rise to Multimedia Information Retrieval. In a multimedia database, queries and items usually pertain to different modalities, meaning that the query is expressed in a modality different from that of the retrieved items. This scenario is also called *cross-modal* retrieval, and some examples are *text-to-image*, *image-to-text*, *text-to-video*. This paradigm extends the famous *query-by-example* one, in which the user provides a query with the same modality of the items to be retrieved. In this scenario, we find Content-based Image Retrieval (CBIR), where both queries and retrieved items are images.

Early multimedia retrieval systems employed metadata associated with the items of

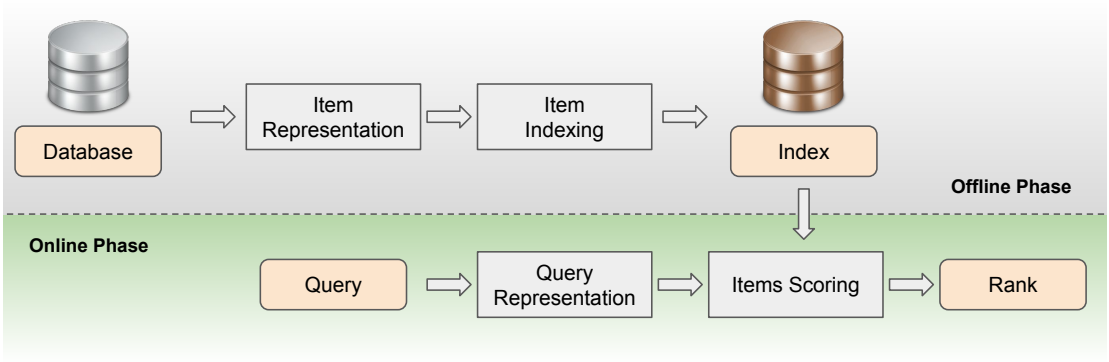


Figure 2.18: *The schema behind any retrieval system. Notice that items and queries are not constrained to have the same modality.*

a collection to perform indexing — human annotations, surrounding text in web documents, or descriptions. Metadata either enables the use of standard relational databases or is composed of unstructured text, which is fairly more manageable than the content itself (e.g., an image). However, metadata is hard to collect, and they are often absent, incorrect, or misleading. For this reason, the research attention moved towards *content-based* retrieval, which exploits the high-level concepts hidden in the media to automatically index huge collections of data without relying on metadata.

In order to be manageable, the raw media content is usually transformed into some vectorial representation which enables similarity search using text-based approaches [2, 6, 40], or the formalism of metric spaces [251]. The schema representing the search system is shown in Figure 2.18. During the *offline* phase, the vectorial representations are computed for each element of the collection through the *item representation* module and stored in an *index*. The *online* phase begins whenever a user issues a query to the system; in this case, the *query representation* is asked to produce the vectorial description of the query, while the *scoring* module computes the similarities between the query representation and the ones stored in the index. Finally, the *ranking* module orders the results by decreasing scores.

The most critical elements in the schema are the *item representation* and *query representation* modules, also called generically *feature extraction* modules. They are responsible for creating suitable and rich vectorial representations for the media, and therefore have an high impact on the system effectiveness. Understanding all the different layers hidden in a media object and condensing all the information inside a single vectorial description is anything but simple. For example, we could be interested in catching low level details in an image — e.g., the style or the distributions of colors of a painting — or we can instead consider high level features, such as the abstract relationships between the actors in a picture — e.g., a *dog* which is running in the *garden*, towards his *master*. This limitation of the employed content descriptors to represent high-level concepts is usually known as *semantic gap* [54]. Although it has been widely studied for CBIR [22], it is especially critical in cross-modal search engines. In fact, different modalities can be compared only at a higher abstraction level, given that they do not share the same low-level structure.

2.4.1 Feature Extraction

Before the advent of Deep Learning, feature extraction methods for images relied on low-level local descriptors such as SIFT [143], SURF [24], and used handcrafted aggregation methods such as VLAD [96] or Fisher Vectors [176].

Today, thanks to the unprecedented ability of Deep Neural Network to create high-level representations of the input contents, the *item representation* and *query representation* modules of Figure 2.18 are implemented by ad-hoc trained DNNs. In this dissertation, we pay particular attention to the features extraction modules for Relational Content-based Image Retrieval in Chapter 3 and cross-media retrieval in Chapter 4, with the intention of reducing the semantic gap. In literature, the approaches for producing suitable features from DNNs can be roughly divided into two main strands: *transfer learning* and *metric learning*, better detailed in the following paragraphs.

Transfer Learning In this scenario, features for retrieval are extracted from an architecture originally trained for a different task. In CBIR, for example, it is fairly common to extract visual features from methods trained on *image classification*. In particular, many works [209, 15, 225] used the activations from the last fully-connected layers of CNN classifiers trained on large image classification datasets like ImageNet. Since fully-connected features struggle to capture localized details in images, many works tried to directly employ the feature maps in output from the last convolutional layer by spatially pooling them using sum or max-pooling [13, 194, 14]. In this context, the Regional Maximum Activations of Convolutions (R-MAC) [219] proposes a very efficient method for pooling convolutional features at different scales to improve the overall retrieval effectiveness. Similar transfer learning methodologies have been applied in the context of text retrieval — by using the representations of sentences [58] or individual words [166, 173] learned in weakly supervised scenarios — as well as in content-based audio retrieval [183].

Metric Learning Distance metric learning (or simply *metric learning*) aims at automatically constructing task-specific distance metrics from (weakly) supervised data in a machine learning manner [26]. The key idea of distance metric learning is to learn an optimal metric that minimizes the distance between similar items and simultaneously maximizes the distance between dissimilar ones. In DL, this objective has been successfully achieved through the triplet-based loss functions already introduced in Section 2.1.4. In many cases, this objective is not directly used to learn a parametric distance function (e.g., Mahalanobis distance); instead, the metric is fixed (e.g., L2-distance), and the network is asked to produce vectorial representations able to satisfy the triplet constraint with the provided metric. The metric learning objective can be easily confused with a classification objective. Nevertheless, cross-entropy only ensures that the features are linearly *separable* in some high-dimensional space. Triplet loss, instead — similarly to other ad-hoc designed objectives such as center loss [234] — also learns *discriminative* features, ensuring intra-class compactness, and giving a meaning to the concept of *nearest neighbor* (Figure 2.19). In the field of computer vision, this learning technique showed its effectiveness in siamese networks [153] and triplet networks [227] for tasks like face verification, image matching and retrieval, and one-shot classification. In particular, metric learning using siamese or triplet loss

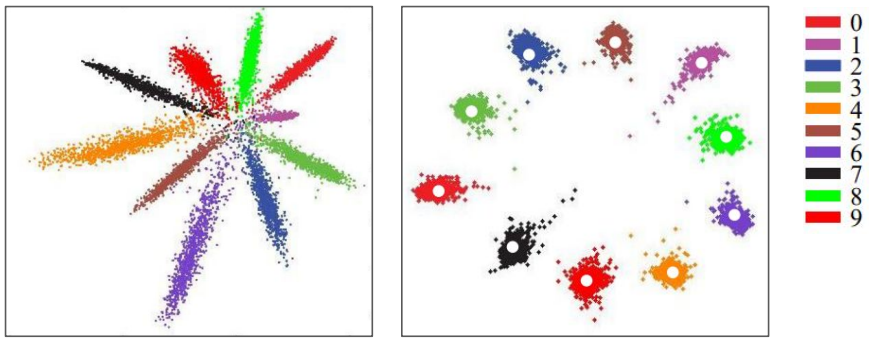


Figure 2.19: *Separable features vs. discriminarive features. Figures by Wen et al. [234].*

was largely used recently to improve CBIR features. In particular, the basic R-MAC pooling scheme was extended in [74] with a triplet loss to learn more discriminative features for instance retrieval. More recently, using a similar learning framework, the Generalized Mean (GeM) features [189, 199] obtained state-of-the-art results on this task. Recent works use triplet loss objectives to project items having different modalities (e.g., images and texts) to the same common space, in which K-nearest neighbors search can be efficiently performed [65, 125]. This kind of architecture is largely used to learn representations independent from the source modality. Therefore, this is the core mechanism behind cross-modal search engines and, in particular, behind *semantic* retrieval.

2.4.2 Evaluation Metrics

In this section, we briefly review the most commonly-used metrics to evaluate the effectiveness of retrieval systems.

First of all, it is worth introducing some useful notation. Given a data collection $\mathbf{X}$, we define $\mathbf{R}_q \subset \mathbf{X}$ the set of items retrieved by the system for a given query q, and $\mathbf{R}_q^\star \subseteq \mathbf{X}$ the actual relevant set of documents for the same query (i.e., the ground-truth). Many of the presented metrics are also used in the evaluation of binary classifiers, given that they deal with *binary relevance* as well. Some examples are the evaluation of object detectors — in which a detection either is or is not correct — or the multi-label binary classification, an example of which is the social network persuasion detector presented in Chapter 4. In information retrieval, binary relevance arises since usually a document either is or is not relevant to the provided query.

Precision and Recall The *precision* is the fraction of retrieved documents that are relevant to the query q:

$$\text{Precision} = \frac{|\mathbf{R}_q| \cap |\mathbf{R}_q^\star|}{|\mathbf{R}_q|}, \tag{2.44}$$

where we indicate with $|\cdot|$ the set cardinality. Differently, the *recall* is the fraction of relevant documents that are actually retrieved by the system:

$$\text{Recall} = \frac{|\mathbf{R}_q| \cap |\mathbf{R}_q^\star|}{|\mathbf{R}_q^\star|}. \tag{2.45}$$

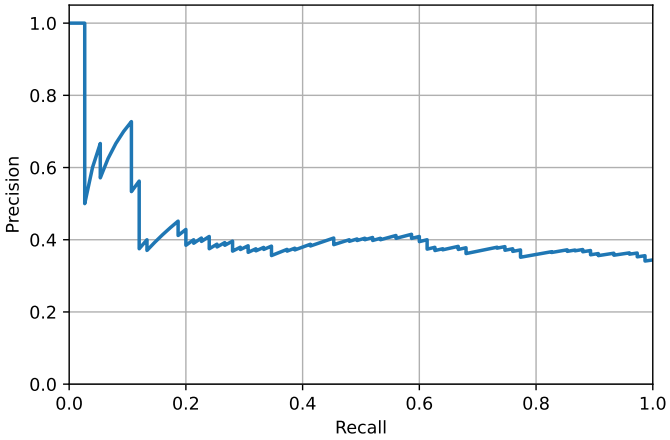


Figure 2.20: *An example of precision-recall curve, showing the precision of the retrieval at given recall values.*

Usually, the elements in the $\mathbf{R}_q$ are chosen by applying a threshold on the relevance score provided by the system for each retrieved item. If the threshold is increased, it is less probable that a retrieved item is classified as relevant; for this reason, by gradually increasing the threshold, the precision increases while the recall decreases. Vice-versa, if the threshold is decreased, the precision decreases — there is a higher probability of false positives in the retrieved set — and the recall increases. Obviously, if $\mathbf{R}_q = \mathbf{X}$, the recall is 1 since we certainly retrieved all the relevant documents; nevertheless, in this case, the precision is very low as we also retrieved all the non-relevant items. Precision and recall are both 1 only when $\mathbf{R}_q = \mathbf{R}_q^\star$. The trend of precision and recall are usually reported in the so-called *precision-recall* curve, an example of which is reported in Figure 2.20.

Mean Average Precision (mAP) Although the precision-recall curve provides a lot of useful information, it is usually difficult to use to compare different systems. For this reason, AP summarizes the precision-recall curve in a single value that provides a quantitative measure for the global effectiveness of the system. Formally, the Average Precision is defined as follows:

$$\text{AP} = \sum_r \text{Precision}(r), \tag{2.46}$$

where the index r spans the different values for the recall, and $\text{Precision}(r)$ is the precision computed at recall r. In other words, the AP is a measure approximating the area under the precision-recall curve. Usually, a system is evaluated on multiple queries, and the mean Average Precision (mAP) is simply the AP averaged over the different queries.

F-measure Similarly to mAP, the *F-measure* provides a single-value characterizing the retrieval effectiveness. The F-measure is defined as the weighted harmonic mean of precision and recall:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where} \quad \beta^2 = \frac{1-\alpha}{\alpha}, \tag{2.47}$$

where P is precision, R is recall, $\alpha \in [0, 1]$, and thus $\beta \in [0, \inf]$. The balanced version of the F-measure is obtained when precision and recall are equally weighted. This happens when $\alpha = 1/2$, or equivalently, when $\beta = 1$. For this reason, the most common version of the F-measure is known as F_1, which is short for $F_{\beta=1}$. When $\beta = 1$, the formula simplifies to:

$$F_1 = F_{\beta=1} = \frac{2PR}{P + R} \tag{2.48}$$

The F_1-score gives values in the interval [0, 1]; hence it is often a good way of summarizing the performance of binary classifiers. The harmonic mean between precision and recall usually gives fair results over the simple arithmetic mean: if we return all the documents (recall $= 1$), the arithmetic mean returns 0.5, which seems very unfair. When the values of two numbers differ greatly, the harmonic mean is instead closer to their minimum than to their arithmetic mean. Therefore it better correlates with user satisfaction.

Recall at K In the literature related to this dissertation, the Recall@K (R@K) is, unfortunately, an overloaded metric: it is defined in two different ways depending on the specific research context in which it is used. In the broad context of *information retrieval*, the Recall@K is defined as the recall metric defined above, but only considering the top-K results. Therefore, R@K measures how many correct results there are within the first K returned elements. Formally, it is defined as:

$$R@K = \frac{|\mathbf{R}_{q,:k}| \cap |\mathbf{R}^{\star}_{q,:k}|}{K}, \tag{2.49}$$

where we indicate as $\mathbf{R}_{q,:k}$ and $\mathbf{R}^{\star}_{q,:k}$ the top-K retrieved elements and the top-K ground-truth elements, respectively. Differently, in *metric learning*, and, more specifically, in papers concerning *cross-modal retrieval*, the Recall@K is defined as the percentage of queries that retrieve at least one correct result within the top-K items. This latter definition will be extensively used for evaluating cross-modal retrieval architectures in Chapter 4, where usually only one item is relevant for every given query.

Spearman's ρ The Spearman's rank correlation coefficient, also known as Spearman's ρ, is a statistical measure of the correlation between two rankings. In information retrieval, it can be therefore used to quantify the affinity between the rankings obtained from the retrieved items and the ground-truth rankings. Mathematically, it is defined as the Pearson correlation coefficient between the rank variables [167]:

$$\rho = \frac{\text{cov}(\text{rg}(\mathbf{R}_q), \text{rg}(\mathbf{R}^{\star}_q))}{\sigma_{\text{rg}(\mathbf{R}_q)} \sigma_{\text{rg}(\mathbf{R}^{\star}_q)}}, \tag{2.50}$$

where $\text{rg}(\cdot)$ returns the rankings, and σ is the standard deviation.

Given n samples, if all the n ranks are distinct integers, the Spearman's ρ can be computed using the formula

$$\rho = 1 - \frac{6 \sum_i d_i^2}{n(n^2 - 1)}, \tag{2.51}$$

where $d_i = \text{rg}(\mathbf{R}_{q,i}) - \text{rg}(\mathbf{R}^{\star}_{q,i})$ is the difference between each pair of ranks taken respectively from the result set and from the ground-truth.

Discounted Cumulative Gain (DCG) Unlike the previous metrics, the Discounted Cumulative Gain (DCG) employs the score returned by the system, walking out of the binary relevance framework used so far. In DCG, every item is associated a *relevance*; the top-K results are considered, and their relevance is cumulated. However, the relevance for each item is penalized logarithmically with respect to its rank in the returned list:

$$\mathrm{DCG}_K = \sum_{i=1}^{K} \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)}. \tag{2.52}$$

The relevance at the exponent places a stronger emphasis on retrieving relevant documents, although other variants weigh it linearly. The values assumed by DCG_K are heavily influenced by the particular value of K used. Usually, to minimize this problem, the normalized version of the DCG, called Normalized Discounted Cumulative Gain (NDCG), is employed instead:

$$\mathrm{NDCG}_K = \frac{\mathrm{DCG}_K}{\mathrm{IDCG}_K}, \tag{2.53}$$

where IDCG_K is the best possible ranking, so that the resulting NDCG_K has values in the range $[0, 1]$.

2.5 Datasets

In this section, we briefly describe the datasets which are relevant — either directly or indirectly — for this thesis.

CLEVR Compositional Language and Elementary Visual Reasoning (CLEVR) [104] is a synthetic dataset composed of 3-D rendered scenes, and it is specifically designed to challenge DNNs on the Relational Visual Question Answering (R-VQA) task. There are 100k rendered images subdivided among training (70k), validation (15k), and test (15k) sets. The total number of questions is 865k, again split among training (700k), validation (150k), and test (15k). The acronym CLEVR was forged from assonance with a bizarre character who lived in the early 1900s, a horse named Hans Clever. It was said that Hans was capable of performing simple arithmetic operations. Careful observation revealed that Hans correctly answered questions by simply reacting to invisible body cues from the people assisting his shows. This anecdote explains quite well the main goals of the CLEVR dataset: (a) trying to understand how an architecture internally pursues the reasoning process and (b) avoiding data biases so that architectures have no chance to behave like Hans. In order to reach such objectives, CLEVR has been designed meticulously. The main concept behind this dataset is the *scene*. A scene contains different simple-shaped objects with mixtures of colors, materials, and sizes. There are cubes, spheres, cylinders, each one of which can have a color chosen among eight; they can be big or small, and they can be made of one of two different materials, metal or rubber. The scene is fully and uniquely described by a *scene graph*. The scene graph describes in a formal way all the relationships between objects. The question is formulated under the form of a *functional program*, which is a declarative formulation where elementary functions such as `count()`, `exists()`, `filter_size()` are connected to form a query. These basic functions potentially take as input attributes,

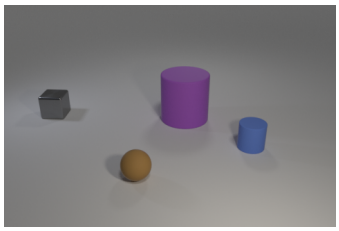


Figure 2.21: *Example of CLEVR image with the associated question: "Are there fewer metallic objects that are on the left side of the large cube than cylinders to the left of the giant shiny block?". Expected answer: "Yes".*

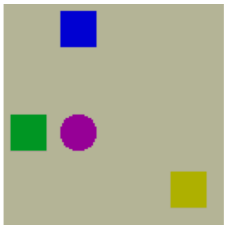


Figure 2.22: *Example of Sort-of-CLEVR image with the associated question: "What is the shape of the object that is farthest from the gray object?". Expected answer: "The yellow square"*

.

e.g.: `filter_size(small)`. The question is therefore described in the form of a graph. The answer to a question represented by its functional program on a scene is simply calculated by executing the functional program on the scene graph. There are 28 different possible answers, ranging from binary responses (*yes* or *no*) to quantities (whole numbers in the range $\{0 \dots 9\}$). Scene graphs are rendered to photo-realistic 3-D scenes using Blender, a free 3-D software; instead, functional programs are converted to natural language expressions compiling some *templates* embedded in the dataset and written in English. A sample image from this dataset is shown in Figure 2.21.

Sort-of-CLEVR Sort-of-CLEVR consists of a simplification of the original CLEVR dataset. It is created mainly for testing and debugging architectures that are designed to work with CLEVR. Images are simpler than 3-D renders provided with the original dataset; they instead carry simple 2-D scenes, consisting of a certain number of shapes. Shapes can be circles or squares and come in different colors. Every object, however, is uniquely identified by its color. Differently from the CLEVR dataset, this one splits the questions into two different subsets:

- **relational questions**, asking for the color or shape of the farthest or the nearest object with respect to the given one. Example: *"What is the shape of the object that is farthest from the gray object?"*;
- **non-relational questions**, involving specific attributes that characterize a single object, in particular the shape or the absolute position of the object with respect to the overall scene. Example: *"What is the shape of the gray object?"*.

Questions are directly encoded into 11-dimensional state vectors, so there is no need for LSTM modules processing natural language. Although this dataset seems extremely

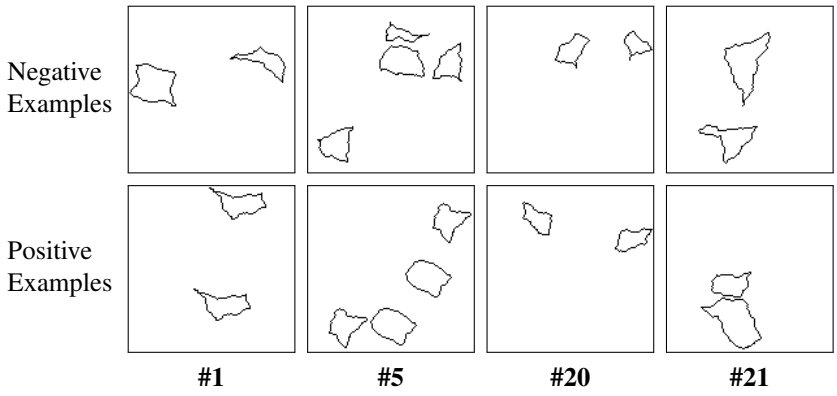


Figure 2.23: *Positive and negative examples from four different SVRT visual problems.*

simple, it can help spot some architectural problems that inhibit the network from thinking relationally. A sample image from this dataset is shown in Figure 2.22.

SVRT The Synthetic Visual Reasoning Test (SVRT) dataset [67] is an extensive benchmark designed to test some abstract reasoning capabilities of machine learning algorithms. It is in principle very similar to Sort-of-CLEVR, although it is designed to diagnose very specific relational shortcomings of Computer Vision algorithms. This dataset is comprised of 128×128 images containing simple black closed curves on a white background, and it is organized as a collection of 23 different visual problems. Every visual problem in SVRT is divided into two classes: the set of positive examples, which are the images that satisfy a specific rule, and the set of negative examples which do not satisfy the rule. The SVRT dataset is designed so that the two categories can be perfectly separated once the underlying rule is understood. In Figure 2.23 are shown positive and negative examples from different visual problems. The highly-irregular closed contours make the number of possible combinations huge, and brute-force memorization of those already seen serves no purpose. For this reason, although SVRT is comprised of images that grossly oversimplify the natural world, this dataset is also free of biases that can be employed to guess the correct class.

MS-COCO The MS-COCO dataset [134] comprises images harvested from the web and containing contextual relationships and noniconic object views. It comes with a total of about 123k images, and it contains 91 common object categories, with 82 of them having more than 5k labeled instances. Unlike the popular ImageNet dataset [56], MS-COCO has fewer categories but more instances per category. Every image has associated a set of 5 human-written captions describing the image so that it can be used for image-text matching and captioning tasks. Usually, in image-text matching literature, the splits introduced in [109] are commonly used. According to these splits, 113k images are reserved for training, 5k for validation, and 5k for testing.

Crisscrossed Captions The Crisscrossed Captions (CxC) dataset extends the development and test splits of MS-COCO with semantic similarity ratings for image-text, text-text and image-image pairs. The rating criteria are based on Semantic Textual Similarity, an existing and widely-adopted measure of semantic relatedness between pairs

of short texts. For semantic similarity, they mixed Universal Sentence Encoder(USE) [42] and Bag-of-Words (BoW) with Glove embeddings [173] on the image captions for obtaining sentence similarity, and they extended it to include judgments about images as well. Overall, CxC contains human-derived semantic similarity ratings for around 267k pairs, derived from more than 1 million independent judgments. The scores range from 0 (completely irrelevant) to 5 (fully relevant). For the evaluation, they proposed to use the Recall@k metric, following the literature on text-to-image or image-to-text on the MS-COCO dataset [65, 125]. They set the threshold for the various tasks by empirically looking at the distribution of the scores. This dataset constitutes a milestone for evaluating image retrieval — and Semantic Content-based Image Retrieval (S-CBIR) in particular, explored in Chapter 4 — as well as for obtaining human-level judgments during the evaluation of cross-modal retrieval methods.

Flickr30k Flickr30k is often used as an alternative to MS-COCO. Flickr30k consists of 31k images and 158k English texts, harvested from the Flickr website [1]. The textual captions are cleaned up to avoid spelling mistakes, eliminate ungrammatical or non-descriptive sentences. Like MS-COCO, each image is annotated with five captions. Following the splits by [109], 29k images are used for training, 1k images for validation, and the remaining 1k images for testing.

Visual Genome Similar to MS-COCO and Flickr30k datasets, Visual Genome [116] is an extensive collection of images scraped from the web and manually annotated. Visual Genome accompanies every image with very precise annotations: it carries bounding boxes for every relevant visual element, together with the class, and a graph describing the inter-dependencies between the actors in the image. For this reason, it can also be used as a vast and rich semantic knowledge base. More in detail, it is comprised of about 108k images, 40k unique attributes, and 40k unique relationships. It also provides five question-answer pairs for each image so that it can also be used to solve real-world Visual Question Answering (VQA) tasks. In this sense, Visual Genome is in spirit very similar to CLEVR, although it is defined on real-world pictures and challenges DNNs to understand real-world situations. It is widely employed to pre-train object detectors or for training architectures on the task of Visual Relationship Detection (VRD). In particular, the bottom-up visual features [10] used in many visual downstream tasks are extracted using a Faster-RCNN model trained on bounding boxes, attributes, and relationships from the Visual Genome dataset. This pre-trained model acquired strong visual relational priors, and it is therefore suitable for use in VQA, image-text matching, or captioning architectures.

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

CHAPTER 3

Relational Content-Based Image Retrieval

Recent advances in Deep Learning technologies brought to light the remarkable potential of Deep Neural Networks. In particular, focusing on the computer vision world, one of the aims of Deep Learning architectures consists in understanding the content of an image at a high level of abstraction. In this respect, some specific tasks have been defined to test the capabilities of newly proposed architectures to cope with high-level reasoning. Understanding relationships between entities is considered a difficult task since it requires complex and non-local reasoning skills. For this reason, some challenging tasks such as Relational Visual Question Answering (R-VQA) and Visual Relationship Detection (VRD) have been introduced as reference tasks for probing the relational abilities of Deep Learning solutions. R-VQA consists of answering questions related to difficult inter-object relationships in an image; on the other hand, VRD tries to recover relationships between couples of objects in the images by coding the information under the form of triplets *subject*, *predicate*, *object*. R-VQA and VRD underlined some of the difficulties that current deep-learning approaches present when it comes to reasoning about relationships between different objects: plain convolutional architectures showed incredible results in tasks such as image classification or object recognition; however, they exhibit some limitations in relational contexts.

In this chapter, we analyze the possibility of applying relational understanding capabilities to the CBIR task. In standard CBIR, the relevance score between two images is determined by focusing solely on a single element, such as an animal, a person, or a famous building. Here, instead, we are interested in a novel sub-field of CBIR, called Relational Content-based Image Retrieval (R-CBIR), in which we care not only about elements in themselves but also about the relationships linking them together. In other words, R-CBIR aims to retrieve images with given relationships among objects. This study is focused on bringing image retrieval a step further with respect to current ap-

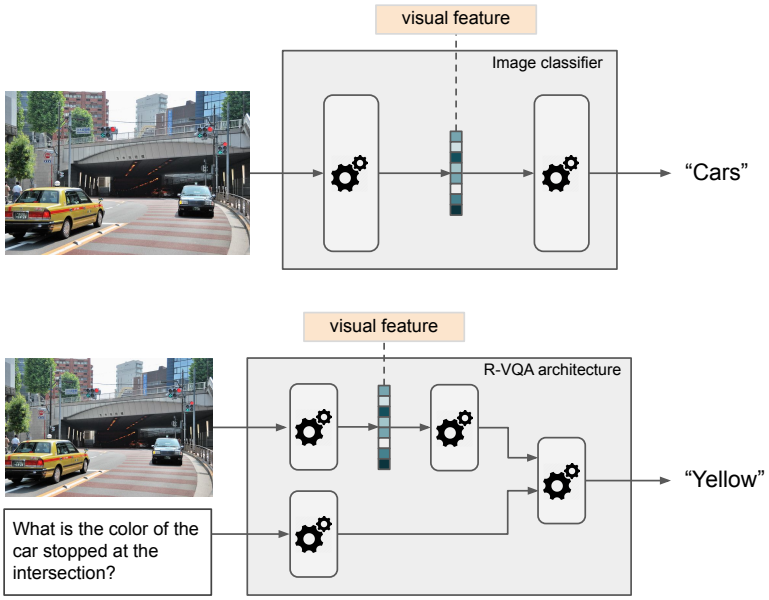


Figure 3.1: *Difference between the visual feature extracted using a pre-trained image classification network (top) and the visual feature extracted using an architecture trained on R-VQA (bottom). In the latter case, the visual feature must embed some relational information that can produce the correct answer when merged with the textual question.*

proaches, keeping the basic idea untouched. In fact, the similarity between two images is always measured as the affinity among some high-level features extracted from the image. In this context, our objective consists of extracting a relationship-aware descriptor able to embed relational information; this novel relational descriptor is easily comparable using standard distance metrics to be used in standard indexing engines.

This chapter investigates the possibility of learning features from networks trained on the task of R-VQA. The transfer-learning methodology is not a novel approach for CBIR, as pointed out in Section 2.4.1. In this scenario, features are often extracted from architectures trained, for example, on image classification tasks. Image classification, however, does not require the architecture to learn difficult relational concepts. Hence, as far as R-CBIR is concerned, relationship-aware features can be extracted from architectures trained on a task that requires high-level reasoning capabilities, and the R-VQA task perfectly fills this need. In other words, we rely on the assumption that architectures that can correctly answer questions on complex inter-object relationships have internally learned some relational concepts that can be later extracted and compared. A simple schematic of the feature extraction pipeline in both these cases is shown in Figure 3.1.

We perform this study in a fully controlled environment, using the images and scene graphs provided by the CLEVR and Sort-of-CLEVR synthetic datasets introduced in Section 2.5. Being highly controlled environments, these datasets are valuable to test in fine detail the relational shortcomings of DNNs.

The chapter is organized as follows. In Section 3.1, we review some of the works belonging to the relational learning world. In Section 3.2, we describe in detail the models proposed to extract features suitable for the R-CBIR task. Given that we need

an appropriate benchmark for evaluating the retrieval abilities of the extracted features, in Section 3.3 we describe in detail the creation of the relational ground-truth, obtained by extending the CLEVR dataset. In Section 3.4, we describe our experimental setup, we collect the results, and we discuss the obtained results, also considering baseline architectures present in the literature. Finally, in Section 3.5, we recap our contribution.

This chapter collects the research published in the following papers:

- Learning Relationship-aware Visual Features. *Proceedings of the European Conference on Computer Vision Workshops*. 2018. [156];
- Re-implementing and Extending Relation Network for R-CBIR. *Italian Research Conference on Digital Libraries*. 2020. [159];
- Learning Visual Features for Relational CBIR. *International Journal of Multimedia Information Retrieval*. 2020. [158].

3.1 Understanding Relations in Images

In this section, we review some of the works related to relational learning in particular related to Relational Visual Question Answering (R-VQA) and Visual Relationship Detection (VRD) tasks. Afterward, we review some of the existing approaches that strongly relate to the novel Relational Content-based Image Retrieval (R-CBIR) task.

Visual Relationship Detection (VRD) Recent work has addressed the problem of VRD in images in the form of triplets (*subject*, *predicate*, *object*), where *subject* and *object* are common objects present in an image, and *predicate* indicates a relationship between them out of a set of possible relationships containing verbs, prepositions, comparatives, etc. Several datasets comprised of a large set of visual relationships, such as the ones introduced in [116, 144, 178], opened the way to approaches aimed to detect inter-object relationships in images [144, 178, 51]. A common approach to VRD employed by many [144, 242, 185, 247] consists at first in proposing entities using region proposal networks, such as Faster-RCNN [198]. Then, once the entities have been located, a network tries to reason on the relationships occurring between them. Notwithstanding approaches that solve VRD are able to detect relationships, they usually do not encode the learned information in a compact representation: all possible relationships are combinatorially tested on prediction time.

Relational Visual Question Answering (R-VQA) R-VQA comes from the basic task of Visual Question Answering (VQA). Plain VQA consists in giving the correct answer to a question asked on a given picture, so it requires connecting together different entities coming from heterogeneous representations (text and visuals). Some works [256, 245] proposed approaches to standard VQA problems on datasets such as VQA [12], DAQUAR [150], COCO-QA [196]. Recently, there is a tendency to conceptually separate VQA and R-VQA. In R-VQA, in fact, images contain difficult inter-object relationships, and question are formulated in a way that it is impossible for deep architectures to answer correctly without having understood high-level interactions between the objects in the same image. Some datasets, such as CLEVR [104], RVQA [146], FigureQA [107], move the attention towards this new challenging task. On the CLEVR

dataset, the authors in [205] and [193] proposed a novel architecture specialized to think in a relational way. They introduced a particular layer called Relation Network (RN), which is specialized in comparing pairs of objects. Objects representations are learned by means of a four-layer CNN, and the question embedding is generated through an LSTM. The overall architecture, composed of CNN, LSTM, and the RN, can be trained fully end-to-end, and it is able to reach superhuman performances. Other solutions [85, 106] introduce compositional approaches able to explicitly model the reasoning process by dynamically building a reasoning graph that states which operations must be carried out and in which order to obtain the right answer. These architectures are internally split into two different sub-components: a *generator network* that produces an execution graph based on the question embeddings, and an *execution network* that executes the graph produced by the generator network taking in input the image features and outputting the answer. In order to close the performance gap between interpretable architectures and high performing solutions, [152] proposed a set of visual-reasoning primitives that are able to perform complex reasoning tasks in an explicitly interpretable manner.

R-CBIR While standard CBIR captured a lot of attention even before the deep-learning era, R-CBIR involves complex reasoning skills and current deep-learning approaches have shown promising results in this direction. We take as reference the work by [219] that introduced R-MAC features — one of the state-of-the-art non-relational image descriptors for image instance retrieval. This descriptor encodes and aggregates several regions of the image in a dense and compact global image representation exploiting a pre-trained fully convolutional network for feature map extraction. The aggregated descriptor is obtained by max-pooling the feature map over different regions and scales and summing them together. Concerning the work carried out on R-CBIR, there was some experimentation using both CLEVR and real-world datasets. The work in [102] introduced a CRF model able to ground relationships given in the form of a scene graph to test images for image retrieval purposes. However, this model is not able to produce a compact feature. They employed a simple dataset composed of 5000 images and annotated with objects and their relationships. More recently, using the Visual Genome dataset, the authors in [254] implemented a large-scale image retrieval system able to map textual triplets into visual ones (object-subject-relation inferred from the image) projecting them into a common space learned through a modified version of triplet-loss. The work in [25] exploits the graph data associated with every image in order to obtain ranking goodness metrics, such as Normalized Discounted Cumulative Gain (NDCG). Their objective was evaluating the quality of the ranking produced for a given query, keeping into consideration the relational content of every scene.

3.2 Features Extraction Architectures

In order to extract relationship-aware visual features, we build upon a state-of-the-art architecture designed for R-VQA on the CLEVR dataset, the Relation Network (RN) architecture [205] introduced in Section 2.2.3. The RN takes an image and a natural language question and handles the R-VQA task as a classification problem, outputting the probabilities over the 28 possible answers. In our setup, the R-VQA is only a proxy task with which the architecture is trained. During inference, only the visual

pipeline is forwarded to obtain the visual relational descriptors. However, extracting visual features from the plain RN architecture is not a trivial operation, as visual features should be extracted before the question conditioning. This is not directly possible in the original model, as question features are early fused with the ones from the visual pipeline. For this reason, we introduced two variations of the RN, called respectively Two-stage Relation Network (2S-RN) and Aggregated Visual Features Relation Network (AVF-RN), whose objective is to obtain rich visual relational descriptors before they are conditioned on the input question.

3.2.1 The Relation Network (RN)

The RN obtained impressive results on relational tasks and in particular on CLEVR. It combines input objects forming all possible pairs and applies a common transformation to them, producing activations aimed to store information about possible relationships among input objects. For the specific task of R-VQA, authors used a four-layer CNN to learn visual object representations, which are then fed to the RN module and combined with the textual embedding of the question produced by an LSTM, conditioning the relationship information on the textual modality. As anticipated in Section 2.2.3, the core of the RN module is given by the following equation:

$$\mathbf{r} = \sum_{i,j} g_\theta(\mathbf{o}_i, \mathbf{o}_j, \mathbf{q}) \tag{3.1}$$

where g_θ is a parametric functions whose parameters θ can be learned during the training phase. Specifically, it is a multi-layer perceptrons (MLP) network; $\mathbf{o}_i$ and $\mathbf{o}_j$ are the objects forming the pair under consideration, and $\mathbf{q}$ is the question embedding vector obtained from the LSTM module. The output $\mathbf{r}$ vector is a multi-modal descriptor carrying information about both visuals and text.

Relation-aware features useful for R-CBIR should be extracted from a stage inside the network still not conditioned by the question. Hence, valid visual features can be extracted from the original RN module only at the output of the convolutional layer, since, after that, questions condition entirely the remaining pipeline. Inspired by the state-of-the-art works on CBIR [219, 73], we obtain an overall description for the image aggregating all object pair features in output from the CNN. These features extracted from the plain RN model serve as a relational baseline for better comparing the introduced 2S-RN and AVF-RN modules.

More in detail, we considered extracting $H_{i,j}([\mathbf{o}_i, \mathbf{o}_j])$, where $\mathbf{o}_i$ is a vector extracted from the i-th position of the last flattened convolutional layer, $[\cdot, \cdot]$ denotes concatenation, and $H_{i,j}(\cdot)$ is an arbitrary aggregation function over all object pairs indexed by (i, j). However, in this work, we aim at producing an R-CBIR baseline for the introduced benchmark by exploiting only two simple aggregations, namely $\max_{i,j}$ and $\text{avg}_{i,j}$. We can notice that for these aggregations the following property holds: $H_{i,j}([\mathbf{o}_i, \mathbf{o}_j]) = [H_i(\mathbf{o}_i), H_j(\mathbf{o}_j)]$. This reveals that the resulting vector is constructed by concatenating two identical aggregated representations. This is mainly because these simple aggregation functions process each single object descriptor component independently. Hence, in this scenario, we can simply discard half of each resulting vector and consider only the aggregation $H_i(\mathbf{o}_i)$. This, in the end, consists in simply computing the

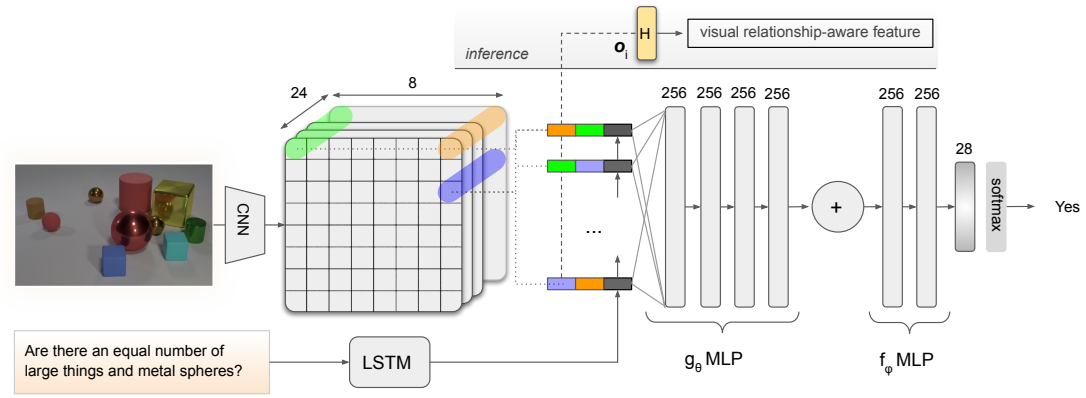


Figure 3.2: *Relation Network (RN) architecture.*

global max or avg pooling from the last layer of the convolutional module. The RN architecture, together with the visual feature extraction endpoint, is shown in Figure 3.2. Although this descriptor represents our relational baseline, we show in Section 3.4.1 that these features already embed relational knowledge able to defeat state-of-the-art CBIR solutions on this task.

3.2.2 Two-stage RN (2S-RN)

The two-stage pipeline is aimed at decoupling visual relationships processing (*first-stage*) from the question elaboration (*second-stage*) so that activations from a layer in the first stage can be employed as visual relationship-aware features.

Our contribution consists in the following: first, we consider all possible relations between objects $\mathbf{v}_{i,j} = g_\theta(\mathbf{o}_i, \mathbf{o}_j)$ in the image. This is what we denoted as *first-stage*. The output from this stage is a representation of the relationships between objects in the image not conditioned on the question. Then, we combine the obtained visual relational representations $\mathbf{v}_{i,j}$ with the query embedding $\mathbf{q}$ as follows:

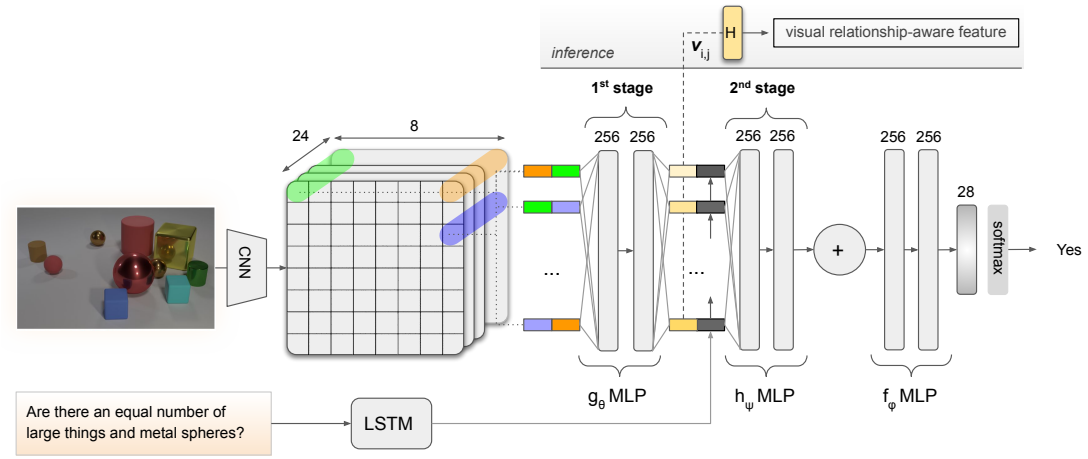
$$\mathbf{r} = \sum_{i,j} h_\psi(\mathbf{v}_{i,j}, \mathbf{q}) = \sum_{i,j} h_\psi(g_\theta(\mathbf{o}_i, \mathbf{o}_j), \mathbf{q}) \tag{3.2}$$

where h_ψ is the second-stage, implemented as a MLP with parameters ψ. Using this solution, we constrained the network to learn relational concepts without considering the questions, at least during the first stage, before the h_ψ function evaluation. Therefore, the relationship-aware features for the images can be extracted from the output of any layer of the g_θ function, and then aggregated at inference time computing $H_{i,j}(\mathbf{v}_{i,j})$.

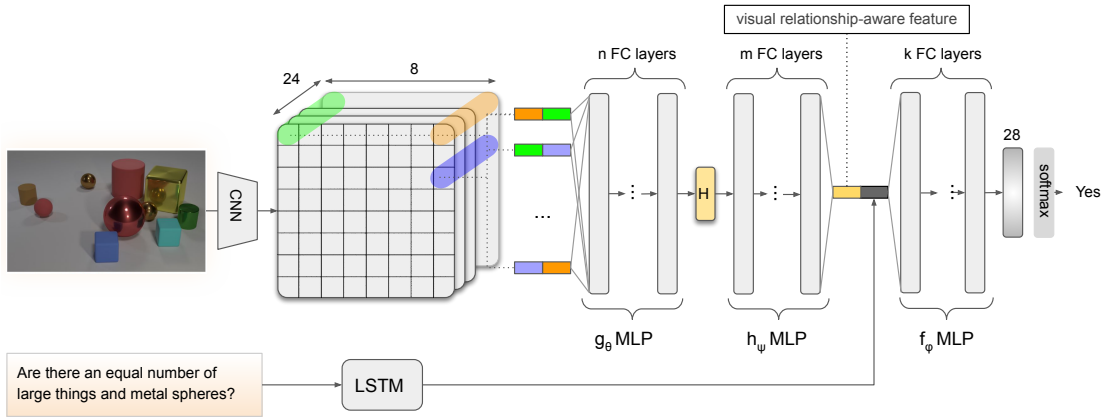
The overall architecture, named 2S-RN, is shown in Figure 3.3a. For training, we stick to the procedure reported in [205].

3.2.3 Aggregated Visual Features Relation Network (AVF-RN)

The 2S-RN approach is able to extract the relational content from the visual pipeline before it is conditioned by the question embedding. Nevertheless, features extracted from the 2S-RN are still not aggregated and contain all the descriptions from every pair of objects. For this reason, these features are aggregated only at inference time



(a) *Two-stage Relation Network (2S-RN) architecture.*



(b) *AVF-RN architecture overview. The number of fully-connected layers is fully customizable, as well as the aggregation function.*

Figure 3.3: *Detailed 2S-RN and AVF-RN architectures with layers configuration.*

by simply averaging over all the visual pairs. To solve this problem, we introduced the Aggregated Visual Features Relation Network (AVF-RN), which is able to learn an aggregated visual representation directly inside the network. Differently from the previous architectures, in this scenario we consider H to be the sum function, effectively bringing the Aggregated Visual Features Relation Network (AVF-RN) closer to the original RN formulation. The AVF-RN network can be described by the following equation:

$$r = [\mathbf{q}, h_\psi(H_{i,j}(\mathbf{v}_{i,j}))] = [\mathbf{q}, h_\psi(H_{i,j}(g_\theta(\mathbf{o}_i, \mathbf{o}_j))] \tag{3.3}$$

with the same naming conventions used for 2S-RN. Although $H = \sum$ in this case, we leave H indicated in the formula to stick with the concept that features are extracted after the application of H, which in this case takes part to the training process. Differently from 2S-RN, h_ψ is not evaluated for every pair; instead, it is evaluated once, on the already aggregated visual features. The architecture has been designed so that each function g_θ, h_ψ and f_ϕ can be customized with any number of fully-connected layers with any number of neurons each. More in detail:

- g_θ comprises the n layers before the aggregation operation;

- h_ψ comprises the m layers between the aggregation and the question insertion;
- f_ϕ comprises the k layers after question insertion; they are aimed at processing the joint visual aggregated features and the textual ones to obtain the information needed to predict the answer.

The overall architecture is reported in Figure 3.3b.

3.2.4 Detailed Configurations and Hyper-parameters Tuning

All the presented architectures are trained on the R-VQA task on the CLEVR dataset. Concerning the RN network, we use the very same setup described by the authors. In particular, the CNN is composed of 4 convolutional layers each with 24 kernels, ReLU non-linearities, and batch normalization; both g_θ and f_ϕ are composed by 256-dimensional fully-connected layers, with ReLU non-linearities after every layer, with four and two layers respectively. The final linear layer with 28 units produces logits for a softmax layer over the answers vocabulary; finally, the learning rate follows an exponential step increasing policy, that doubles it every 20 epochs, from 5e-6 up to 5e-4. Features are extracted directly at the end of the CNN and are aggregated using global average pooling. 2S-RN follows a very similar setup to the one of the original RN. Differently from the RN, g_θ and the novel h_ψ are both composed by 2 fully-connected layers. In this case, features are extracted at the end of the g_θ layer, immediately before the question concatenation. Both RN and 2S-RN reaches very high performances when trained and tested on CLEVR R-VQA: they obtain 93,6% and 93,8% accuracy on the test set respectively.

In the case of RN and 2S-RN, the concatenation of the question with all the object pairs works as a simple but quite effective attention mechanism. The novel AVF-RN model, instead, introduces the question embedding after the aggregation function. With this modification, the model gains in feature relational expressiveness but, on the other hand, the attention-like effect is lost. For this reason, we obtain an overall lower accuracy with respect to the other architectures. Furthermore, AVF-RN is substantially different from the other models. For this reason, we would need to re-settle the hyperparameters for accomodating the architectural changes. There are several hyper-parameters that should be tuned and an extensive search is infeasible with the available hardware, so we concentrate on the most relevant ones. In this network, the most critical hyperparameters are the number of fully-connected layers for every function g_θ, h_ψ, and f_ϕ, namely n, m, k, and the output size for each one of these layers. For the remaining hyper-parameters, we try to stick to successful configurations observed when training the RN and the 2S-RN architectures. In Table 3.1 we collect some of the hyper-parameters experimentation we performed on this architecture, together with the reached accuracy on the CLEVR R-VQA task. The best result is obtained using weighted-sum as aggregation, with weights learned during training, one layer of h_ψ and three layers of f_ϕ. The aggregation is positioned after the 4-th fully-connected layer of g_θ, while the question is inserted after a single fully-connected layer of h_ψ. The 4-th layer of g_θ is larger in order to augment the expressiveness of the aggregated feature. In order to speed up the convergence, we initialize the weights for the CNN and the first two fully-connected layers of g_θ with the weights coming from the respective

g_θ config.	h_ψ config.	f_ϕ config.	Aggr. Type	Accuracy(%)
256, 256, 512	256	256, 256	sum	53.8
256, 256, 256, 512	256	256, 256	sum	53.2
256, 256, 256, 256	256	256 ,256, 256	sum	54.0
256, 256, 256, 512	256	256, 256, 256	sum	54.2
256, 256, 256, 1024	-	512, 1024	weighted-sum	55.7
256, 256, 256, 512	**256**	**256, 256, 256**	**weighted-sum**	**64.5**

Table 3.1: *The accuracy values of different fully-connected layer configurations for every function g_θ, h_ψ and f_ϕ. Each configuration includes the output size for every fully-connected layer.*

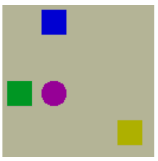
layers of the 2S-RN architecture (they are the only ones to maintain the same role and the same interface with respect to the AVF-RN).

Although the final accuracy is quite far from the performance reached by the RN and the 2S-RN architectures, we claim that this is enough for learning relationship-aware visual features, and we confirm this with targeted experiments on our downstream retrieval task.

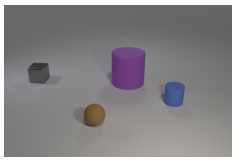
3.3 Constructing a R-CBIR Ground-truth

In order to quantitatively evaluate the performance of the introduced architectures on the downstream R-CBIR task, we need to construct a suitable ground-truth. A R-CBIR ground-truth can be constructed by defining, for each query image, which are the most relevant ones according to some relationship-aware distance metric defined on the available scene-graphs.

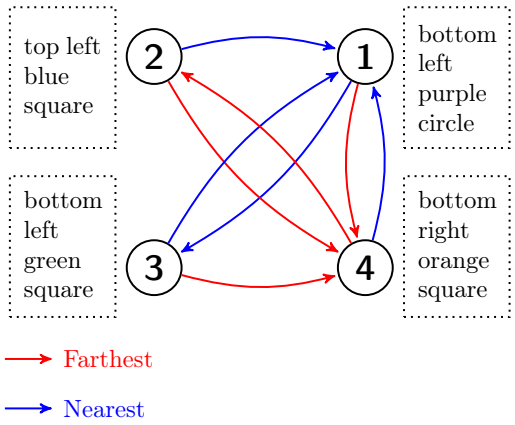
Scene graphs The best way to formally describe relationships among objects inside a scene is by using *scene graphs*, already available both in CLEVR and Sort-of-CLEVR. More in detail, a scene graph contains *nodes*, that account for objects occupying the scene and *edges*, that describe relations occurring among them. Every node or edge can be assigned a set of attributes that fully describe them. For Sort-of-CLEVR, nodes carry information regarding objects *color*, *shape* together with their absolute positions (*left/right* or *up/down* with respect to the scene). An edge, instead, carries information about the kind of relationship it is describing. In Sort-of-CLEVR, an edge can refer to *farthest* and *nearest* relations. Unlike the Sort-of-CLEVR case, CLEVR object attributes do not include absolute positions since CLEVR deals uniquely with relational questions, and the possible attributes are the *color*, the *shape*, the *material* and the *size*. CLEVR also includes an higher number and more detailed spatial relations: *to the left of*, *to the right of*, *in front of*, *behind*. In Figure 3.4, we report an example image for each dataset together with the associated scene-graphs. Notice how, although CLEVR graph is complete, half of the edges can be removed without losing information, since *to the right of* implies an opposite edge *to the left of* and *in front of* implies an opposite edge *behind*.



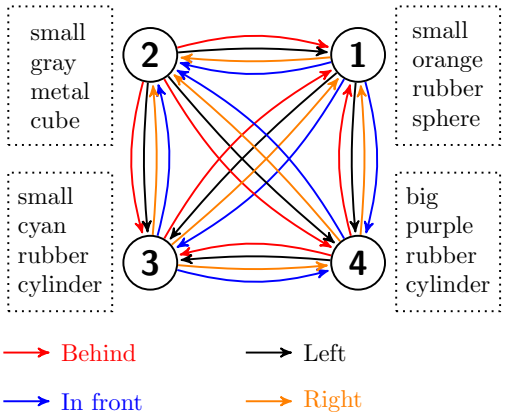
(a) *Sort-of-CLEVR scene*



(b) *CLEVR scene*



(c) *Sort-of-CLEVR scene graph*



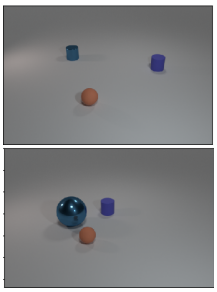
(d) *CLEVR scene graph*

Figure 3.4: *Example of scenes with associated scene-graphs*

3.3.1 Ground-truth Generation

We define a ground-truth for retrieving images with similar relations among objects relying on the similarity between scene graphs. Two scene graphs should be similar if they can depict almost the same relations between the same objects. However, evaluating the similarity between two graphs is not trivial; it is often a subjective task as there are aspects of the graph — for example, the attributes associated to nodes — that weight differently, depending on the specific application.

Although many solutions have been proposed in literature for defining distances between graph-structured data [38], in this particular use case we decided to employ the Graph Edit Distance (GED), that is an extension of the well-known edit distance working on strings. The edit distance is defined as the shortest number of *delete*, *insert*, *substitute* operations needed in order to transform the source string into the target one. Differently from strings, edit operations on graphs include *delete*, *insert*, *substitute* for both nodes and edges, for a total of six edit operations. The problem is faced as an optimization problem. Since the GED problem is known to be computationally hard, in this work we employ two different implementations [1, 200]. The proposal in [1] describes an exact, non-approximated version of the GED algorithm. We reference it as *Exact-GED*. While execution times are acceptable for Sort-of-CLEVR graph data, they become easily unworkable on CLEVR, even when removing the redundant *behind* and *left* edges. For this reason, the relaxation proposed in [200] is able to perform an efficient approximation of the algorithm. We refer to this as *Approx-GED*. Approx-GED does not consider the entire span of solutions. Instead, it looks for a tiny subset of edit sequences, obtained by first matching similar nodes using linear assignment and then matching edges on the ruled node pairing. Nevertheless, during experimentation, we



Steps	Cost
1. Substitute node **small-cyan-metal-cylinder** with **big-cyan-metal-sphere** (change 2 attributes)	0.5
2. Substitute edge **small-cyan-metal-cylinder behind small-blue-rubber-cylinder** with **big-cyan-metal-sphere in front of small-blue-rubber-cylinder**	1.0

Figure 3.5: *Example of computation of the Graph Edit Distance between two scene-graphs, using the soft-match policy. The returned cost is 1.5 in this case.*

empirically measured that the resulting approximated ground-truth is perfectly comparable with the exact one. Both implementations allow for the customization of the node-edge edit costs on the basis of their attributes. We applied the following policies:

- nodes-edges insertion or deletion has always a cost of 1;
- edge substitution cost is 1 if edges do not belong to the same kind of relation, 0 otherwise;
- node substitution cost can be driven by two different policies:
  - *soft-match*: all attributes of a node weight equally during a substitution. So, considering a total of 4 attributes, if three attributes match the substitution cost is $3/4 = 0.75$. This is the fairest and most neutral solution since it does not prefer any attribute over all the others;
  - *hard-match*: the cost is 1 if at least one attribute value differs. It is 0 only if all attributes match.

To clarify the functioning of the GED algorithm using our cost policies, we report in Figure 3.5 an example on CLEVR with *soft-match*. In the light of this, given a query, we compute the ground-truth ranking of the dataset by sorting all scenes using the GED distances computed between the scene graph of the query image and the graphs from all the others.

3.4 Experiments

In our extensive experimental setup, we aim at verifying if visual features extracted from our improved architectures are better at describing visual relationships than state-of-the-art CBIR features. We generate image rankings from relational features by normalizing all the vectors obtained in output from the H function, and then computing the Euclidean distance between the query feature and all the others. From the ranking point of view, this is equivalent to ranking by decreasing cosine similarity. We use the Spearman's ρ correlation index for evaluating the ranking goodness against our ground-truth. Spearman's ρ is a common ranking similarity measure often employed in information retrieval scenarios [154], as explained in Section 2.4.2.

As a baseline, we computed the ranking obtained with one of the state-of-the-art non-relational image descriptors for image instance retrieval, namely the R-MAC descriptor [219]. With R-MAC, the similarity score between two images is obtained by

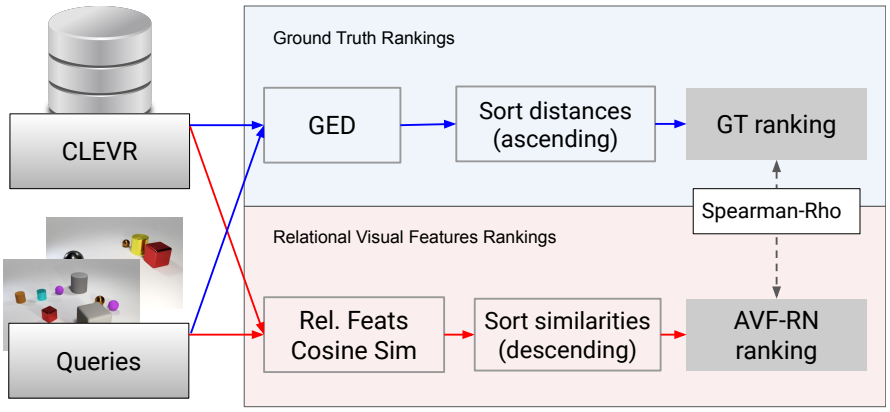


Figure 3.6: *High-level overview of the R-CBIR evaluation pipeline.*

	Sort-of-CLEVR		CLEVR	
GT policy	soft-match	hard-match	soft-match	hard-match
R-MAC [73]	0.49±0.03	0.07±0.03	−0.15±0.02	−0.18±0.02
RN [205] *max*	0.36±0.02	0.14±0.03	−0.24±0.02	−0.25±0.03
RN [205] *avg*	0.64±0.02	0.34±0.04	0.08±0.05	0.06±0.05
2S-RN *max*	**0.70**±0.02	**0.58**±0.03	−0.19±0.03	−0.21±0.03
2S-RN *avg*	0.24±0.02	0.18±0.02	**0.15**±0.04	**0.13**±0.04

Table 3.2: *Spearman's ρ correlation index for existing features and our novel two-stage extracted features (2S-RN), both using CLEVR and Sort-of-CLEVR. We report the 95% confidence intervals for the mean over 500 queries.*

computing the cosine similarity between image descriptors. In our experiments, we adopted the R-MAC descriptor extracted from the pre-trained model proposed in [73].

First of all, for the RN and the 2S-RN architectures which do not internally aggregate the visual features, we perform a preliminary analysis for understanding what is the effect of different aggregation functions at inference time and for evaluating the different variants of the GED algorithm used to generate the baseline. In this preliminary analysis, we target both CLEVR and Sort-of-CLEVR datasets. Afterwards, we compare in great detail all the three RN architectures — the plain RN, the 2S-RN, and the AVF-RN — targeting carefully-designed subsets of the CLEVR dataset. CLEVR, in fact, contains the most challenging images and launches this work towards more realistic use cases.

3.4.1 Preliminary Experiments on RN and 2S-RN

We evaluate both convolutional and 2S-RN features against the generated ground-truth. In our experiments, features from 2S-RN are extracted from the last layer of g_θ. We employ features extracted from the convolutional layer of the original RN as baseline for evaluating features from the first-stage of our novel 2S-RN approach. Table 3.2 reports values of Spearman's ρ for the two considered datasets. CLEVR results can be

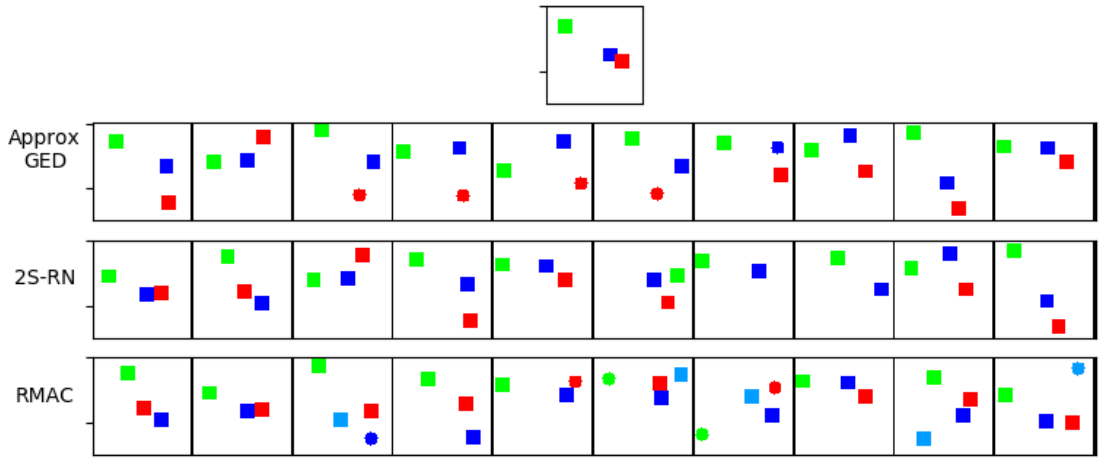


Figure 3.7: *Top 10 Sort-of-CLEVR images using our solution (2S-RN) and R-MAC against our ground-truth for a given query (on top).*

reproduced using the code publicly available on GitHub[1]. Spearman's ρ correlations are relative to the two generated ground-truths, *soft-match* and *hard-match* obtained by ranking images using Approx-GED. Notice that the Spearman's ρ correlation between rankings obtained with exact and approximated versions on Sort-of-CLEVR dataset over 500 queries gives a value of 0.89, using the *soft-match* policy. Given this result, we can empirically claim that the employed GED approximation is legitimate in this particular scenario.

Correlation index has been evaluated over the rankings generated from 500 query images, in order to produce statistically meaningful results. We can notice that, with a 95% confidence interval on the mean, convolutional relational features definitely defeat R-MAC features on this relational task. Furthermore, relational features extracted from the 2S-RN are noticeably better than convolutional relational features extracted from the plain RN. These results are reasonable since the original RN has problems when processing the image alone without having also the question as input, while R-MAC tends to retrieve images containing the very same objects present in the query disregarding relative size, order or position.

Depending on the dataset, different aggregation methods can produce diverse optimal results. In particular, the $\max$ aggregation seems to work better on Sort-of-CLEVR dataset, while avg obtain the best results on CLEVR. This could be explained considering that the avg aggregation is more sensible to the number of identical relations happening inside the scene. However, the number of relations involved among objects having same attributes is quite important when considering CLEVR since, unlike in Sort-of-CLEVR, it is more likely to find multiple instances of the same relationship insisting on similar objects. Thus, discriminating the CLEVR images by their cardinality can be relevant for obtaining an overall better ranking. Moreover, the $\max$ aggregation becomes unstable and sensible to outliers when the number of samples increases; a single huge activation in one of the 4,096 features in CLEVR can significantly affect the aggregation results. This is in line with findings in aggregation techniques for CNN features [219, 73], where sum and avg aggregations are preferred.

[1] https://github.com/mesnico/learning-relationship-aware-visual-features

Although it is quite difficult to give an objective evaluation of RN features and R-MAC ones by only looking at the first more relevant images, visual evaluation reported in Figure 3.7 are useful for giving an intuition beyond statistics. Visual results for the CLEVR dataset are reported in the next paragraph, where we evaluate all the proposed architectures and the baselines on different splits of the more realistic CLEVR dataset.

3.4.2 Detailed Evaluation on CLEVR

In this section, we extend the previous study on the CLEVR dataset, by including the AVF-RN architecture and considering different slices of the dataset to derive more precise conclusions. Furthermore, in these experiments we probe another non-relational baseline. In particular, we train a simple architecture on a multi-classification task, where the objective consists in correctly classifying all the objects inside every CLEVR scene. This baseline is aimed at understanding if a CNN trained to recognize objects and their attributes, without focusing on relationships, can obtain good results on this task. This simple architecture consists of the CNN already used in the original RN architecture and 2 fully-connected layers with ReLU non-linearities for use as multi-label classifier. Similarly to the basic RN architecture, features are extracted by average-pooling the CNN activations. We call this architecture *multi-label CNN*.

All the architectures are trained on the CLEVR training set, while the features are always extracted on the test-set in order to evaluate the generalization capabilities of the system. All the architectures are trained on an RTX 2080Ti, with a batch size of 640. During the experiments, we observe that the training time was almost the same for all the RN-derived architectures. We train for about 300 epochs, and we pick the model having the highest validation accuracy among all the training epochs. The average training speed is about 25 minutes per epoch, while the feature extraction requires only about 1 minute for the whole test set. Questions are not needed at extraction time, so the entire architecture is considerably lighter.

We use the following three different dataset setups for evaluating the results:

1. **CLEVR-Full** — we use the entire CLEVR test set. Any image can be selected as query and any image could be eligible for being retrieved;
2. **CLEVR-Filtered-Queries** — we select as queries only the images containing at most N objects, while any image remains eligible for being retrieved;
3. **CLEVR-Subset** — we filter the entire CLEVR test set with images containing at most N objects. Therefore, both queries and retrieved images contain at most N objects.

CLEVR-Full is the same scenario used for evaluating 2S-RN performances in the previous section. However, the approximated GED algorithm we employ presents some notable differences with the exact version when graphs have a large number of nodes. For this reason, during experimentation, we explore also the simpler scenarios *CLEVR-Filtered-Queries* and *CLEVR-Subset*. CLEVR comes with rendered images containing no more than 10 objects. In our experiments we set N equal to 5.

Table 3.3 reports values of Spearman's ρ correlation index for all the experiments on all the three versions of the CLEVR datasets. As before, Spearman's ρ correlations are relative to the ground-truth generated as explained in Section 3.3 and obtained by

	CLEVR Full	CLEVR Filtered Queries	CLEVR Subset
R-MAC [73]	-0.15 ± 0.02	0.02 ± 0.02	0.09 ± 0.01
Multi-label CNN	0.05 ± 0.05	0.64 ± 0.04	0.18 ± 0.04
RN [205]	0.04 ± 0.05	0.64 ± 0.03	0.20 ± 0.03
2S-RN [156]	0.15 ± 0.04	0.65 ± 0.02	0.26 ± 0.02
AVF-RN	$\mathbf{0.28\pm0.04}$	$\mathbf{0.72\pm0.02}$	$\mathbf{0.34\pm0.02}$

Table 3.3: *Spearman-Rho correlation index for the explored methods. We report the 95% confidence intervals for the mean over 500 queries.*

ranking images using the approximated version of the GED algorithm. The Spearman's ρ correlation index is evaluated over the rankings generated using 500 query images, in order to produce statistically meaningful results.

Discussion The AVF-RN features reach the state-of-the-art on the R-CBIR task, defeating both non-relational baseline methods (R-MAC and multi-label CNN) and the RN and 2S-RN relationship-aware techniques. It is interesting to notice the similarity among the results of the RN and the multi-label CNN features. These results confirms that the features extracted from the last CNN layer of the RN, without further processing, can probably discriminate among object and attribute instances, leaving the relationships out. On the *CLEVR-Full* scenario, our AVF-RN features obtain an almost doubled Spearman's ρ value with respect to the 2S-RN one. This suggests that the novel AVF-RN architecture is able to correctly order complex relevant scenes in terms of their relational content. However, due to the approximation introduced by *Approx-GED* in case of large number of objects, it is difficult to strongly confirm this claim in this scenario.

On the other hand, in the *CLEVR-Filtered-Queries* scenario, the images with few objects are privileged by the ground-truth. Hence, standard approaches like R-MAC or simple CNN features behave quite well since they can exploit their capability of retrieving images having a similar number of objects with respect to the query. Besides counting, they are in any case unable to catch intrinsic inter-object relationships. Instead, these details are well captured by AVF-RN and 2S-RN features. The aggregation learned inside the AVF-RN network obliges the layers after the aggregation to learn compact and smart scene descriptions. Consequently, AVF-RN captures more detailed scene-information with respect to the aggregated features computed at inference time in 2S-RN.

In the *CLEVR-Subset* scenario, instead, all the retrieved images are forced to contain a small number of objects. In this case, as all the images contain few objects, the only way to obtain good results is to understand the intrinsic relational content of the scene. This explains why there is a great improvement of AVF-RN features over the other methods in this case.

As for the Sort-of-CLEVR case, we report a visual evaluation in Figure 3.8 for giving a qualitative feedback and an intuition beyond statistics. We collect these visual results from the challenging *Full CLEVR* experiment. In particular, we can see that both R-MAC and multi-label CNN features always try to find the very same objects as the query, in any position inside the image. Instead, it seems that 2S-RN and AVF-RN

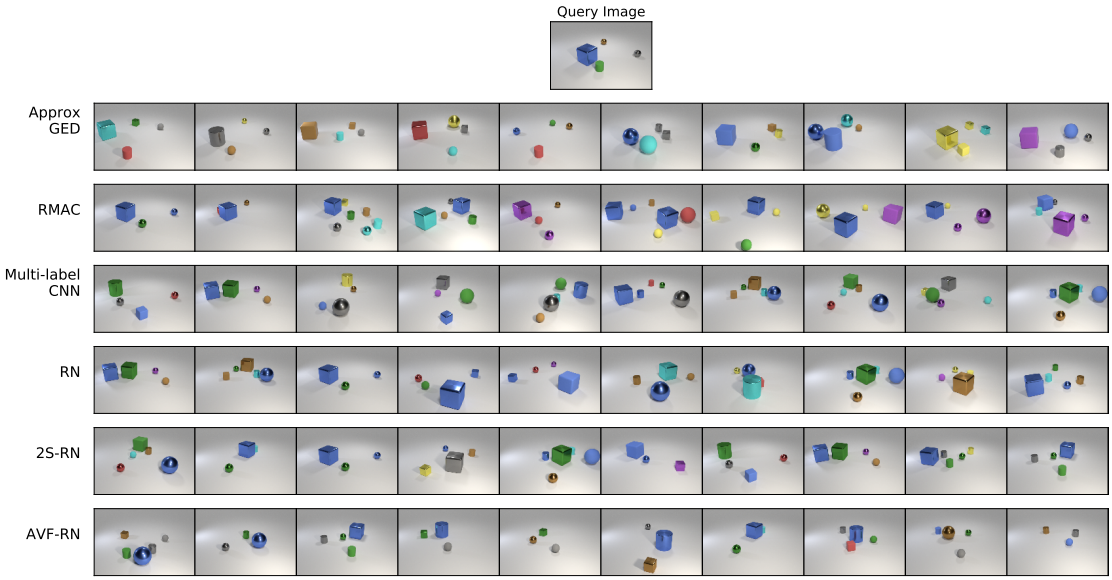


Figure 3.8: *Most relevant images for the proposed query from Full CLEVR experiment, using both non-relational approaches (R-MAC, Multi-Label CNN) and relational ones (RN, 2S-RN, AVF-RN). The first row belongs to the ground-truth generated as explained in Section 3.3.*

are interpreting the scene from an high-level perspective by finding all the images containing a big object (better if a metallic blue cube) surrounded by other smaller objects. On our website[2] it is possible to find an interactive browsing system for exploring the R-CBIR results from the proposed methods for different query images.

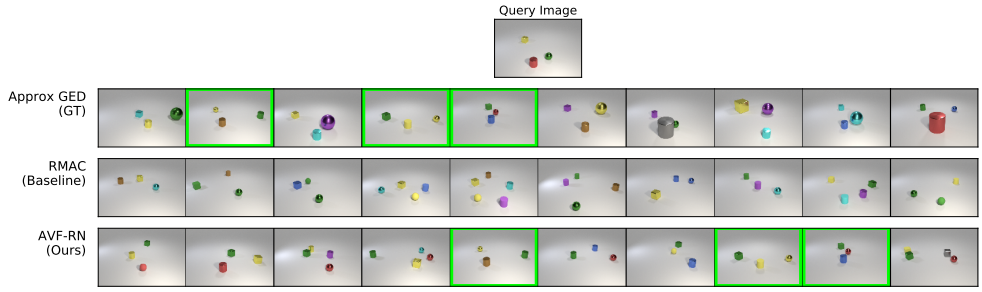
3.4.3 Success/Failure Analysis

In Figure 3.9 we report simple cases of success and failure of the top-performing method AVF-RN against the two baselines R-MAC and multi-label CNN. We assume the result as successful if our AVF-RN features can retrieve more ground-truth images with respect to the baselines; otherwise, the experiment is considered failed for the examined query. For the sake of simplicity, we analyze only the top 10 results.

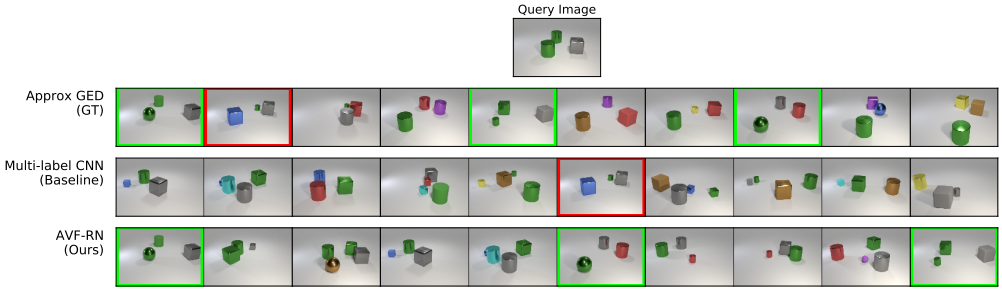
We can notice that successful retrieved images (Figure 3.9a and Figure 3.9b) are well approximating the ground-truth scene graphs. This is probably because AVF-RN features exhibit some scene-wide image understanding that is not tailored to the features of single objects. On the other hand, R-MAC features are quite good at catching the key visual features of the single objects, such as their size, but they have troubles to focus the attention on the global scene arrangement.

Failure cases demonstrate that AVF-RN features cannot always catch the relational content of the scene. In particular, in the failure example of Figure 3.9c, the AVF-RN features seem to be always triggered by a yellow object, that is a not so important characteristic when considering the whole scene arrangement. Instead, Figure 3.9d demonstrates that is difficult to catch objects arranged in precise configurations (in this case, placed on the same line). In this example, both the multi-label CNN baseline and our AVF-RN features fail.

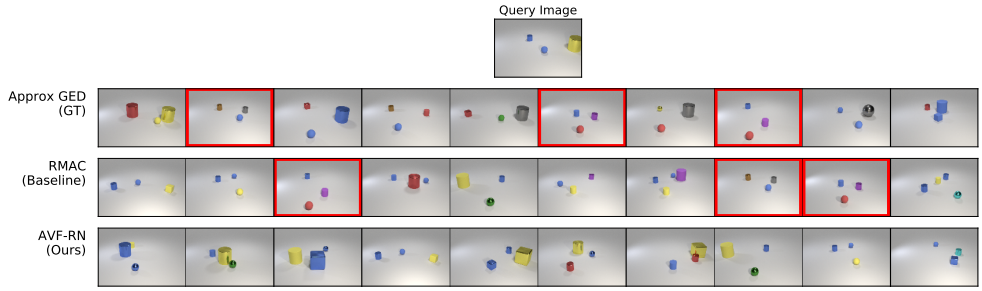
[2] `www.rcbir.org`



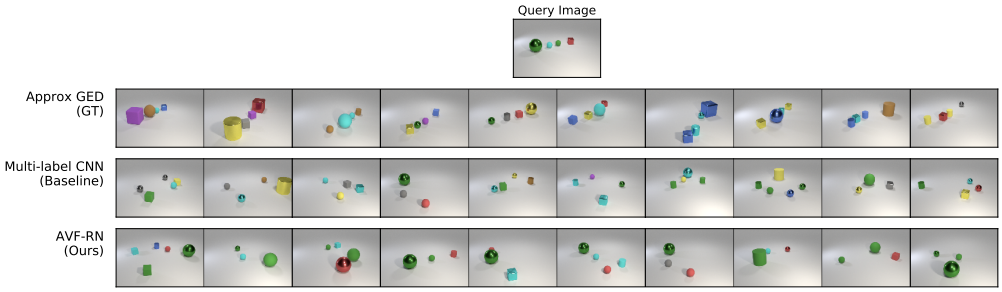
(a) *Success against R-MAC features*



(b) *Success against Multi-label CNN features*



(c) *Failure against R-MAC features*



(d) *Failure against Multi-label CNN features*

Figure 3.9: *Success (a)(b) and failure (c)(d) cases for AVF-RN compared to the baselines, R-MAC (a)(c) and Multi-Label CNN (b)(d). Matches among GT and AVF-RN are marked in green, while matches among GT and the baselines in red.*

3.5 Summary

In this chapter, we defined the sub-task of Relational Content-based Image Retrieval (R-CBIR) in which retrieved images should be similar to the query in terms of relationships among objects. This was motivated by the fact that current image retrieval systems, performing traditional CBIR, are not able to infer relations among the query and the retrieved images.

We developed some variations of the Relation Network (RN) trained on Relational Visual Question Answering (R-VQA) on the CLEVR dataset, from which we extracted some meaningful relationship-aware visual features. We extended the CLEVR dataset adding information about scene-graph similarities for measuring the retrieval performance on the novel R-CBIR task. On this dataset, we obtained very promising results, and we discussed upon the limits of current image retrieval descriptors.

In the end, this chapter presented a preliminary study on the possibility of learning relationship-aware visual descriptors on very restricted and controlled datasets. Although this research cannot be directly applied to realistic scenarios including real-world pictures, it paves the way towards a deeper understanding of relationship-aware architectures for use in retrieval applications. In the next chapter, many of the introduced concepts reoccur, although in a realistic scenario with extremely prominent use cases.

CHAPTER 4

Visual-Textual Matching and Retrieval

In Chapter 3, we showed how the relation network could be employed for extracting relationship-aware visual features, producing descriptors that embed spatial relationships between visual entities. In this chapter, we try to further grow this idea by (a) moving to real-world images and (b) enabling highly semantic cross-modal retrieval. Concerning the point in (a), considering real-world images enables the developed tools to be usable in real use cases. To discuss the importance of the point in (b), we need to discuss an intrinsic limitation of the approach proposed in Chapter 3. The AVF-RN network was designed to solve the R-VQA upstream task and did not include any targeted objective for producing retrieval-effective features. This was in line with previous works that used features from networks pre-trained on image classification [219]. Nevertheless, the features extracted using this approach suffer from the lack of discriminative power, as described in Section 2.4.1. Direct correspondence between images and scene-graphs to produce relational discriminative features is in principle possible by enforcing image features and graph features to lay into the same common space, using a metric learning objective. While this is feasible using computer-generated data, it is difficult on real-world images, as we would require a manually annotated scene-graph for each image. Although some publicly available datasets carry this kind of annotations — OpenImages [120] or Visual Genome [116] — this approach is definitely not scalable in the long term, as it requires too complex human-provided annotations.

There is another type of data able that carries very high-level information and describes relationships between entities: *textual data*. When we caption an image with some natural language text, we are implicitly encoding the image with symbols carrying high-level semantics. In turn, we are creating a structured description of that image. This is what scene-graphs would natively encode; nevertheless, natural language texts have been used by humans for thousands of years to spread ideas, knowledge and re-

lationships. Thanks to this, texts are natively widespread in our society. It is relatively straightforward to find images with associated textual descriptions on the web, and it is easier to ask annotators to caption an image instead of encoding its full scene-graph. Furthermore, dealing with two well-established modalities — images and natural language texts — enables *cross-modal* applications, in which the user, for example, can search images feeding the system a textual query and vice-versa. In the first sections of this chapter, we exactly tackle the cross-modal retrieval task, focusing on both effectiveness and efficiency aspects.

From a broader perspective, in this chapter, we try to fill the semantic gap between images and texts by employing the Transformer architecture, which recently demonstrated groundbreaking relational abilities. In fact, the non-local Transformer's attention mechanism can relate distant and heterogeneous entities, creating a powerful cross-modal reasoning pipeline.

This chapter is organized as follows. In Section 4.1 we present the framework for creating and evaluating cross-modal features for use in large-scale visual-textual retrieval applications. In particular, we introduce two Transformer Encoder based architectures, able to generate highly semantic cross-modal features. In Section 4.2, we discuss about some preliminary results for quantizing and sparsifying the features obtained in Section 4.1, for indexing them using off-the-shelf text-based retrieval tools. Furthermore, in Section 4.3, we explore the abilities of these features to perform semantic image retrieval. The results obtained from these researches are applied to a real use case in Section 4.4, where we present a tool for large-scale video retrieval. In Section 4.5, instead, we leave the retrieval framework, and we employ the same visual-textual Transformer-powered technologies to demonstrate their effectiveness on another critical application: the detection of disinformation campaigns in social networks. Finally, in Section 4.6, we summarize the work presented in this chapter and draw the conclusions.

This chapter collects the research published in the following papers:

- Transformer Reasoning Network for Image-text Matching and Retrieval. *International Conference on Pattern Recognition (ICPR)*. 2021. [163];
- Fine-grained Visual Textual Alignment for Cross-modal Retrieval using Transformer Encoders. *Trans on Multimedia Computing Communications and Applications (TOMM)*. 2021. [161];
- AIMH at SemEval-2021 Task 6: Multimodal Classification using an Ensemble of Transformer Models. *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval)*. 2021. [164];
- VISIONE at Video Browser Showdown 2021. *International Conference on Multimedia Modeling (ICMM)*. 2021. [8];
- Towards Efficient Cross-Modal Visual Textual Retrieval using Transformer-Encoder Deep Features. *International Conference on Content-Based Multimedia Indexing (CBMI)*. 2021. [162];
- Relational Visual-Textual Information Retrieval. *International Conference on Similarity Search and Applications*. 2020. [155].

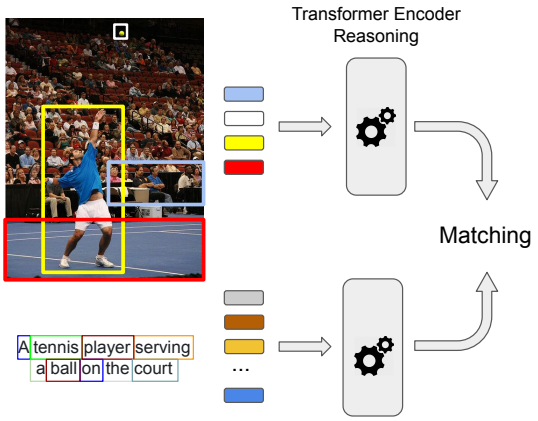


Figure 4.1: *High-level overview of the Transformer-based image-text matching. Image and text are seen respectively as sets of image regions and sequences of words, and they are processed using a Transformer-based reasoning engine.*

4.1 Transformers for Effective and Efficient Visual-Textual Retrieval

Joint processing of images and natural language text is crucial nowadays, as it enables very interesting and challenging tasks. One of the most interesting problems is cross-modal retrieval, in which we aim at retrieving images given a textual description. This task is intrinsically challenging due to the semantic gap that visual data entails. Furthermore, the matching should be scalable and efficient to be deployed in real cross-modal retrieval systems.

Inspired by state-of-the-art works in cross-modal matching, in this section we propose two architectures able to efficiently and effectively match images and natural language texts. We are to tackle this important problem by employing the recently introduced Transformer models. Unlike current multi-modal Transformer methods — presented in Section 2.3.5 and further discussed in the following section — the introduced architectures are guided to produce compact and semantic descriptions of images and sentences. Compact and informative descriptions are required in the context of large-scale retrieval systems, where image and text embeddings can be compared and indexed using a simple similarity function (e.g., cosine similarity) defined on a common embedding space. A rough overview of the approach is shown in Figure 4.1.

The first introduced network is called Transformer Encoder Reasoning Network (TERN) [163]. Transformer Encoder Reasoning Network (TERN) is able to produce compact visual and textual features laying in the same common space by attending to all the words from the sentence and the regions from the image. The second network is called Transformer Encoder Reasoning and Alignment Network (TERAN) [161], and it extends TERN by forcing a fine-grained alignment loss between image regions and sentence words, using supervision only at the global image-sentence level.

This section also discusses the shortcomings of the evaluation metrics used to measure the cross-modal retrieval effectiveness. In particular, the current literature usually employes binary relevance to evaluate the cross-modal retrieval task, constrained by the lack of fine-grained annotations in currently available datasets. In many cases, these metrics cannot capture the real user satisfaction. In the light of this, we propose to use

the Normalized Discounted Cumulative Gain (NDCG) metric with ad-hoc constructed relevance scores, accounting for non-exact yet relevant matches.

Before diving into the details of the proposed architectures and evaluation protocols, in the following two sections we better frame the problem reviewing the relevant literature and discussing the training and the evaluation protocols.

4.1.1 Visual-Textual Matching using Transformers

In this section, we review some of the previous works related to image-text joint processing for cross-modal retrieval and alignment, and high-level relational reasoning, on which this work lays its foundations. Also, we briefly summarize the evaluation metrics available in the literature for the cross-modal retrieval task.

Image-Text Processing for Cross-Modal Retrieval Image-text matching is often cast to the problem of inferring a similarity score among an image and a sentence. Usually, one of the common approaches for computing this cross-domain similarity is to project images and texts into a common representation space on which some kind of similarity measure can be defined (e.g.: cosine or dot-product similarities). Images and sentences are preprocessed by specialized architectures before being merged at some point in the pipeline.

Concerning image processing, the standard approach consists in using CNNs, usually pre-trained on image classification tasks. In particular, [115, 223, 135, 88, 63] use VGGs, while [139, 65, 77, 89] use ResNets. Concerning sentence processing, many works [41, 145, 109, 65, 125, 122, 89] employ GRUs or LSTMs recurrent networks to process natural language, often considering the final hidden state as the only feature representing the whole sentence. The problem with these kinds of methodologies is that they usually extract extremely summarized global descriptions of images and sentences. Furthermore, in many works the image embeddings are extracted from standard image classification networks, such as ResNet or VGG, by employing the network activations before the classification head. Usually, descriptions extracted from CNN networks trained on classification tasks are able to only capture global summarized features of the image, ignoring important localized details. Therefore, a lot of useful fine-grained information needed to reconstruct inter-object relationships for precise image-text alignment is permanently lost.

For these reasons, many works try to employ region-level information, together with word-level descriptions provided by recurrent networks, to understand fine-grained alignments between words and localized patches in the image. Recent works [86, 136, 137, 87, 230, 43, 122] exploit the availability of pre-computed region-level features extracted from the Faster-RCNN [197] object detector. An alternative consists in using the features maps in output from ResNets, without aggregating them, for computing fine-grained attentions over the sentences [239, 90, 228, 231, 79, 99].

Recently, the Transformer architecture [220] achieved state-of-the-art results in many natural language processing tasks, such as next sentence prediction or sentence classification. The results achieved by the BERT model [58] demonstrated the power of the attention mechanism to produce accurate context-aware word descriptions. For this reason, some works in image-text matching use BERT to extract contextualized word embeddings for representing sentences [235, 207, 187, 232]. In Section 2.3.5 we reviewed

some works that employ BERT-like processing on both visual and textual modalities, such as ViLBERT [145], ImageBERT [184], Pixel-BERT [91], VL-BERT [215]. These latest works achieve state-of-the-art results in sentence and image retrieval, as well as excellent results on the downstream word-region alignment task [44]. However, they cannot produce separate image and caption descriptions; this is an important requirement in real-world search engines, where usually, at query time, only the query element is forwarded through the network, while all the elements of the database have already been processed by means of an offline feature extraction process. These architectures model a function $s = \phi(I, C)$ that measures the affinity between an image and a caption, where I is an image, C is the caption and s is a normalized score in the range $[0, 1]$. Following this path, at query time we would need to evaluate $\phi(I, C)$ for every element in the database. This is infeasible for scalability issues and timing constraints.

Some architectures have been designed so that they are natively able to extract disentangled visual and textual features. In particular, in [65] the authors introduce the VSE++ architecture. They use VGG and ResNets visual features extractors, together with an LSTM for sentence processing, and they match images and captions exploiting hard-negatives during the loss computation. With their VSRN architecture [125], the authors introduce a visual reasoning pipeline built of Graph Neural Networks (GNNs) and a GRU to sequentially reason on the different image regions. Furthermore, they impose a sentence reconstruction loss to regularize the training process. The authors in [90] use a similar objective, but employing a pre-trained multi-label CNN to find semantically relevant image patches and their vectorial descriptions. Differently, in [207] an adversarial learning method is proposed, where a discriminator is used to learn modality-invariant representations. The authors in [79] use a contextual attention-based LSTM-RNN which can selectively attend to salient regions of an image at each time step, and they employ a recurrent canonical correlation analysis to find hidden semantic relationships between regions and words.

The works closer to the TERAN setup presented in the next sections are SAEM [235] and CAMERA [187]. In [235] the authors use triplet and angular loss to project the image and sentence features into the same common space. The visual and textual features are obtained through Transformer Encoders (TEs) modules. Differently from the TERAN proposal, they do not enforce fine-grained alignments, and they pool the final representations to obtain a single-vector representation. Instead, in [187] the authors use BERT as language model and an adaptive gating self-attention module to obtain context-enhanced visual features, projecting them into the same common space using cosine similarity. Unlike our work, they specifically focus on multi-view summarization, as multiple sentences can describe the same images in many different but complementary ways.

High-Level Reasoning Another branch of research from which this work draws inspiration is focused on the study of relational reasoning models for high-level understanding. The work in [206] proposes an architecture that separates perception from reasoning, namely the Relation Network (RN), discussed in Section 2.2.3. Other solutions try to stick more to a symbolic-like way of reasoning. In particular, [84, 106] introduce compositional approaches able to explicitly model the reasoning process by dynamically building a reasoning graph that states which operations must be carried out and

in which order to obtain the right answer. Recent works employ Graph Neural Networks (GNNs) to reason about the interconnections between concepts. The authors in [246, 244, 128] use GNNs to reason on the image regions for image captioning, while [241, 132] use GNNs with attention mechanisms to produce the scene graph from plain images.

Cross-Modal Retrieval Evaluation Metrics All the works involved with image-caption matching evaluate their results by measuring how good the system is at retrieving relevant images given a query caption (image-retrieval) and vice-versa (caption-retrieval). Usually the Recall@K metric is used [65, 125, 184, 145, 123], where typically $K = \{1, 5, 10\}$. On the other hand, [41] introduced a novel metric able to capture non-exact results by weighting the ranked documents using a caption-based similarity measure. In this work, we extend the metric introduced in [41], giving rise to a powerful evaluation protocol that handles non-exact yet relevant matches. Relaxing the constraints of exact-match similarity search is an important step towards an effective evaluation of search engines.

4.1.2 Training and Evaluation Protocols

The neural networks that produce suitable vectorial representations for cross-modal retrieval are usually trained on a specific objective called *cross-modal matching*. Given an image-text pair $\{I, T\}$, this objective consists in learning a normalized relevance score $s \in [0, 1]$ which is high if I is described by T (or vice-versa), and low otherwise. Usually the relevance score is constructed as a similarity score between image and text vectors produced respectively by image and text encoders f_θ and g_ψ:

$$s = \cos(f_\theta(I), g_\psi(T)) = \frac{f_\theta(I) \cdot g_\psi(T)}{\|f_\theta(I)\|^2 \|g_\psi(T)\|^2}, \tag{4.1}$$

where $\cos(\cdot)$ is the cosine similarity and f_θ and g_ψ are the text and image encoders parametrized respectively by θ and ψ. This kind of construction for the similarity scores defines a metric space where the similarity between images and text is measured through a dot product between normalized vectors (i.e., cosine similarity). A nice structure of this space can be obtained using a triplet loss objective, by getting closer the related $\{I, T\}$ pairs while pulling away unrelated ones. More in details, recent approaches in cross-modal matching [65, 125, 109, 258] use the so-called *hinge-based triplet ranking loss* objective. In particular, this loss considers an image as an anchor and tries to attract a positive sentence while pushing away a negative one, very similarly to the triplet loss described in Section 2.1.4. Concurrently, it performs the symmetric operation: it takes a sentence as an anchor and samples two images, a positive and a negative one, to be pushed and pulled away respectively. An overview of this dynamic is shown in Figure 4.2. In datasets where every image I_i is accompanied by k human written sentences T_i^k (e.g. MS-COCO or Flickr30k), it is possible to construct the related pairs $\{I_i, T_i^k\}$, while all the negative pairs can be constructed as $\{I_i, T_j^k\}$ $\forall i, j$ s.t. $i \neq j$.

During the evaluation phase, the best way to probe the constructed space is to search all the images related to a given text or vice-versa. If the space has the desired structure, it is possible to perform KNN search to find the images more similar to a given sentence or vice-versa (Figure 4.3).

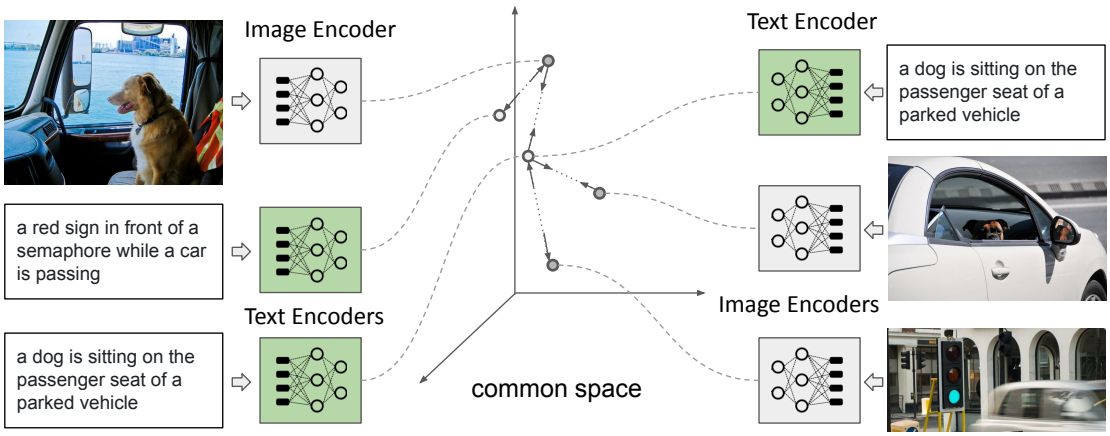
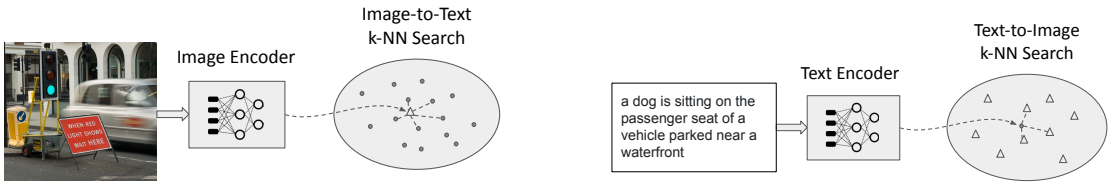


Figure 4.2: *The dynamics operated in the common space by the hinge-based triplet ranking loss.*



(a) *Text-to-Image retrieval.*

(b) *Image-to-Text retrieval.*

Figure 4.3: *The evaluation is performed by measuring the quality of a k-nn search between the query and the vectors from the other modality. Triangles are image vectors, while dots are sentence vectors laying in the same common-space.*

Notice that in literature the retrieval task is often used as a proxy task for evaluating how well an architecture is able to match related images and sentences. In this chapter, we consider it the main goal, as this research is geared towards the study of effective and efficient cross-modal search engines that could be deployed in real scenarios. Therefore, the metrics employed during the evaluation phase take inspiration from the information retrieval literature. Evaluating the effectiveness of a method performing information retrieval is often difficult since different metrics usually capture non-overlapping features of interest, all possibly relevant. As of now, many works in the computer vision literature treating image-text matching measure the retrieval abilities of the proposed methods by employing the well known Recall@K metric. The Recall@K measures the percentage of queries able to retrieve the correct item among the first K results. In datasets where images come with a list of sentences written by human annotators, this metric is very simple to evaluate: given a textual query chosen among all the sentences in the dataset, the exact matching image is the one that in the dataset is associated with that sentence (in case of sentence-retrieval) and vice-versa for the image-retrieval scenario. This is a metric perfectly suitable for scenarios where the query is very specific and thus we expect to find the elements that match perfectly among the first search results. However, in common search engines, the users are not asked to input a very detailed query, and they are often not searching for an exact match. They expect to find in the first retrieved positions some relevant results, with relevance defined using some pre-defined and often subjective criterion. In these

scenarios, the first retrieved elements could be relevant without being the images/sentences originally paired with the query sentences/images. For these reasons, it would be nice to weight the intrinsic semantic similarity between an image and a text during the evaluation phase; in this way, the strict matching/non-matching binary criterion is relaxed, and an element can be relevant to a certain degree. A graphical representation of the two situations can be observed in Figure 4.4.

Inspired by the work in [41], we employ a common metric often used in information retrieval applications, the Normalized Discounted Cumulative Gain (NDCG). The NDCG is able to evaluate the quality of the ranking produced by a certain query by looking at the first p positions of the ranked elements list. The premise of NDCG is that highly relevant items appearing lower in a search result list should be penalized as the graded relevance value is reduced proportionally to the position of the result.

As anticipated in Section 2.4.2, the NDCG until position p is defined as follows:

$$\text{NDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}, \qquad \text{where} \quad \text{DCG}_p = \sum_{i=1}^{p} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}; \tag{4.2}$$

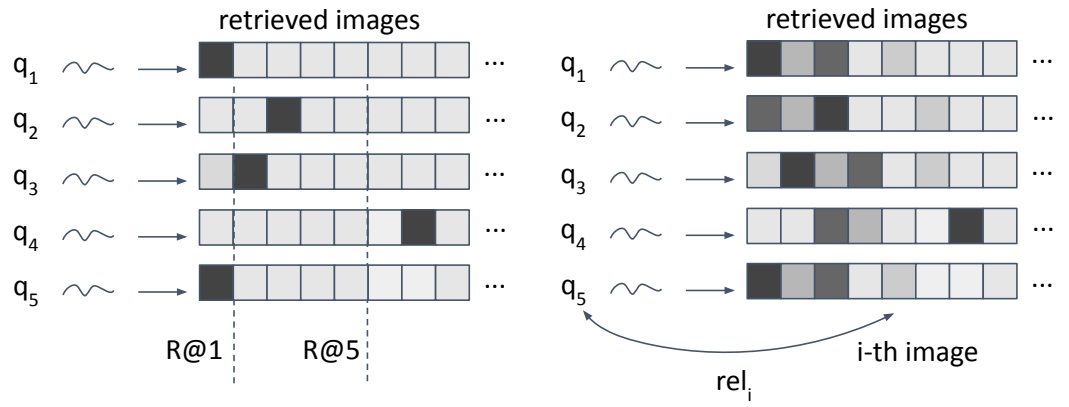
rel_i is a positive number encoding the affinity that the i-th element of the retrieved list has with the query element, and IDCG_p is the DCG_p of the best possible ranking. Thanks to this normalization, NDCG_p acquires values in the range $[0, 1]$. The NDCG_p is computed by normalizing the DCG_p with respect to the Ideal Discounted Cumulative Gain (IDCG), that is defined as the DCG of the list obtained by sorting all its elements by descending relevance:

The rel_i values can be computed using well-established sentence similarity scores between a sentence and the sentences associated with a certain image. Being a cross-modal retrieval setup, the relevance should be a value obtained from a function operating on an image I_i and a caption C_j. In principle, it could be possible to use the $\phi(I_i, C_j)$ learned by other networks as the ones in [145, 184]. The problem is that ϕ is a complex DNN, and I_i, C_j are drawn from a dataset of thousands of elements, in the best case. This means that constructing a $N_c \times N_i$ relevance matrix is computationally unfeasible, where N_c is the number of total captions and N_i is the total number of images in the dataset.

Usually, in the considered datasets, images come with a certain number of associated captions. This important consideration can be exploited for efficiently producing our relevance scores: instead of computing $\phi(I_i, C_j)$, we could think of computing $\text{rel}_i = \tau(\bar{C}_i, C_j)$, where $\bar{C}_i$ is the set of all captions associated to the image I_i, and $\tau : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ is a similarity function defined over a pair of sentences returning their normalized similarity score. With this simple expedient, we could efficiently compute quite large relevance matrices using similarities defined over captions, which are in general computationally much cheaper than similarities computed between images and sentences directly.

We thus compute the rel_i value in the following ways:

- $\text{rel}_i = \tau(\bar{C}_i, C_j)$ in case of image retrieval, where C_j is the query caption
- $\text{rel}_i = \tau(\bar{C}_j, C_i)$ in case of caption retrieval, where $\bar{C}_j$ is the set of captions associated to the query image I_j.



(a) *The Recall@K evaluation approach.*

(b) *Semantic-based relevance approach; rel_i is a a-priori known score defining the affinity between the query and the i-th image.*

Figure 4.4: *(a) only the exact-matching images are used for evaluating the quality of the retrieval, given sentences as queries; (b) a more "colorful" approach would consider the intrinsic semantic relevance between a query and the retrieved images. The same reasoning applies for image-to-text retrieval.*

In our work, we use ROUGE-L[133] and SPICE[9] as sentence similarity functions τ for computing caption similarities. These two scoring functions capture different aspects of the sentences. In particular, ROUGE-L operates on the longest common sub-sequences, while SPICE exploits graphs associated with the syntactic parse trees, and has a certain degree of robustness against synonyms. In this way, SPICE is more sensitive to high-level features of the text and semantic dependencies between words and concepts rather than to pure syntactic constructions.

4.1.3 Transformer Encoder Reasoning Network (TERN)

Inspired by many recent works that achieved state-of-the-art results in image-text matching, we developed an architecture able to project images and sentences in the same common-space. Differently from many, however, the developed architecture incorporates Transformer Encoders (TEs) for processing both the visual and textual pipelines, and it is designed with to be used in large-scale cross-modal retrieval scenarios. TERN relies almost entirely on the TE architecture, both for the visual and the textual data pipelines. The TE takes as input sequences or sets of entities, and it can reason upon these entities disregarding their intrinsic nature. In particular, we consider the salient regions in an image as visual entities, and the words present in the caption as language entities. More formally, the input to our reasoning pipeline is a set $I = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_n\}$ of n image regions representing an image I and a sequence $C = \{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_m\}$ of m words representing the corresponding caption C.

The overall architecture is shown in Figure 4.5. The reasoning module continuously operates on sets and sequences of n and m objects respectively for images and captions. The objective is to produce a compact representation of the n processed regions and of the m processed words suitable for the downstream task of image-text matching in a common space with fixed dimensionality. One of the easiest ways to proceed is to pool the elements of the set/sequence using symmetric functions like sum or avg, or, like

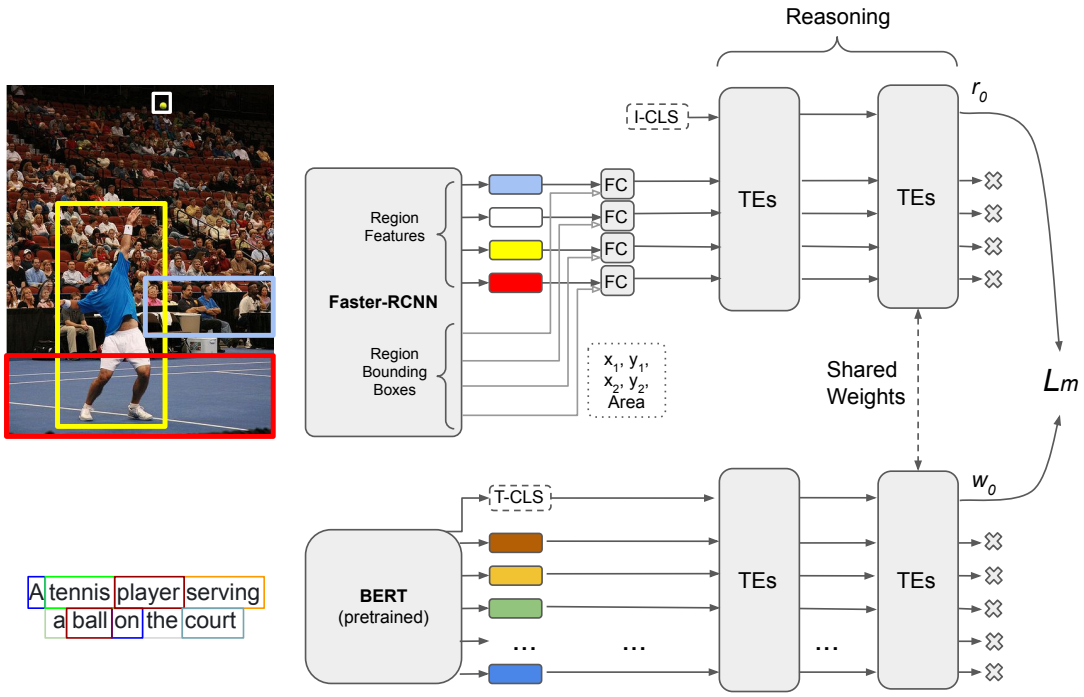


Figure 4.5: *The proposed TERN architecture. Region and words are extracted through a bottom-up attention model based on Faster-RCNN and BERT respectively. BERT already employs positional encoding for representing the sequential nature of words, therefore this step is not reported in the figure. Concerning regions, the extracted bottom-up features are conditioned with the information related to the geometry of the bounding-boxes. This is done through a simple fully connected stack in the early visual pipeline, before the reasoning steps.* L_m *is the matching loss.*

in [125], growing a meaningful aggregated representation inside the hidden state of a recurrent network (GRU or LSTM).

Our method, instead, follows the approach by BERT [58]: we reserve a special token both at the beginning of the regions set and of the words sequence (I-CLS and T-CLS) devoted to carrying global information along the two pipelines. For this reason, we effectively expand the number of image regions to $n + 1$ and the number of words to $m + 1$, with r_0 and w_0 reserved for this purpose. Initially, $\mathbf{w}_0$ is set to the T-CLS BERT token, while $\mathbf{r}_0$, i.e., I-CLS, is a zero vector. At every reasoning step, this information is updated attentively by the self-attention mechanism of the TEs. In the end, our final image and caption features are $\mathbf{r}_0$ and $\mathbf{w}_0$ in output from the last Transformer Encoder layer. In the last layers of the TERN architecture, the abstracted representations of the visual and textual pipelines should be comparable. To enforce this constraint and together apply regularization during training, we share the weights of the last layers of the TEs before computing the matching loss $\mathcal{L}_m$ on the common space.

If we use only bottom-up features without any spatially related information, the visual reasoning engine is less responsive to spatial relationships. This is a fairly important aspect to capture since lot of textual descriptions contain spatial indications (e.g. *on top of* or *above*). In order to include spatial awareness also in the visual reasoning process, we condition the early visual pipeline with the bounding-boxes coordinates. To this aim, we compute the normalized coordinates and the normalized area for each

region, as follows:

$$\mathbf{c} = \left\{ \frac{x_1}{W}, \frac{y_1}{W}, \frac{x_2}{H}, \frac{y_2}{H}, \frac{(x_2 - x_1)(y_2 - y_1)}{WH} \right\}. \tag{4.3}$$

Then, we concatenate $\mathbf{c}$ with the original bottom-up feature. In the end, we forward this information through a simple Linear-ReLU-Linear stack (sharing weights among all the n regions) to obtain the final spatial-aware bottom-up feature.

Learning In order to match images and captions in the same common space, we use a hinge-based triplet ranking loss, focusing the attention on hard negatives, as in [65, 125]. Therefore, we use the following loss function:

$$\begin{aligned} \mathcal{L}_m(i, c) = & \max_{\mathbf{c}'}[\alpha + S(\mathbf{i}, \mathbf{c}') - S(\mathbf{i}, \mathbf{c})]_+ + \\ & \max_{\mathbf{i}'}[\alpha + S(\mathbf{i}', \mathbf{c}) - S(\mathbf{i}, \mathbf{c})]_+, \end{aligned} \tag{4.4}$$

where $[x]_+ \equiv max(0, x)$. The hard negatives $\mathbf{i}'$ and $\mathbf{c}'$ are computed as follows:

$$\begin{aligned} \mathbf{i}' &= \arg\max_{\mathbf{j} \neq \mathbf{i}} S(\mathbf{j}, \mathbf{c}) \\ \mathbf{c}' &= \arg\max_{\mathbf{d} \neq \mathbf{c}} S(\mathbf{i}, \mathbf{d}), \end{aligned} \tag{4.5}$$

where $(\mathbf{i}, \mathbf{c})$ is a positive pair. S is the similarity function between image and caption features, which in our setup is the cosine similarity. As in [65], the hard negatives are sampled from the mini-batch and not globally, for performance reasons.

Region and Word Features The $I = \{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_n\}$ and $C = \{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_m\}$ initial descriptions for images and captions come from state-of-the-art visual and textual pre-trained networks, Faster-RCNN with bottom-up attention and BERT respectively.

Faster-RCNN [197] is a state-of-the-art object detector. It has been used in many downstream tasks requiring salient object regions extracted from images. Therefore, Faster-RCNN is one of the main architectures implementing human-like visual perception. The work in [10] introduces bottom-up visual features by training Faster-RCNN with a ResNet-101 backbone on the Visual Genome dataset [116]. Using these features, they can reach remarkable results on the two downstream tasks of image captioning and visual question answering.

Concerning text processing, we use BERT [58] for extracting word embeddings. BERT already uses a multi-layer Transformer Encoder (TE) to process words in sentences and capture their functional relationships through the same powerful self-attention mechanism. BERT embeddings are trained on some general natural language processing tasks such as sentence prediction or sentence classification and demonstrated state-of-the-art results in many downstream natural language tasks. BERT embeddings, unlike word2vec [165] or GloVe [173], capture the context in which each word appears. Therefore, every word embedding carries information about the surrounding context, that could be different from caption to caption.

4.1.4 Transformer Encoder Reasoning and Alignment Network (TERAN)

As explained in the previous paragraph, TERN processes visual and textual elements, exploring and reasoning on the relationships among image regions and sentence words. However, its main objective is to match images and sentences as atomic, global entities, by learning a global representation of them inside special tokens (I-CLS and T-CLS) processed by the Transformer Encoder (TE). This usually leads to performance loss and possibly poor generalization since fine-grained information useful for effective matching is lost during the projection to a fixed-sized common space.

For this reason, in this section, we propose an extension to TERN, called Transformer Encoder Reasoning and Alignment Network (TERAN) in which we force a fine-grained word-region alignment. Fine-grained matching deals with the accurate understanding of the local correspondences between image regions and words, as opposed to coarse-grained matching, where only a summarized global descriptions of the two modalities is considered. In fact, differently from TERN, the objective function is directly defined on the set of regions and words in output from the architecture, and not on a potentially lossy global representation. Using this objective, TERAN tries to individually align the regions and the words contained in images and sentences respectively, instead of directly matching images and sentences as a whole. The information available to TERAN during training is still coarse-grained, as we do not inject any information about word-region correspondences. The fine-grained alignment is thus obtained in a semi-supervised setup, where no explicit word-region correspondences are given to the network.

As TERN, TERAN is built using a stack of TE layers, both for the visual and the textual data pipelines. The TE takes as input sequences or sets of entities, and it can reason upon these entities disregarding their intrinsic nature. In particular, we consider the salient regions in an image as visual entities, and the words present in the caption as textual entities.

The TERN architecture in [163] produces summarized representations of both images and words by employing special I-CLS and T-CLS tokens that are forwarded towards the layers of the TEs. In the end, the processed I-CLS and T-CLS tokens gather important global knowledge from both modalities. Contrarily, TERAN does not produce aggregated fixed-sized representations for images and sentences. For this reason, it does not employ the global features constructed inside the I-CLS and T-CLS tokens. Instead, it tries to impose a global matching loss defined on the variable-length sets in output from the last TE layers that is able, as a side effect, to produce also good and interpretable region-word alignments.

The price to pay to obtain this fine-grained alignment is that we can no more directly rely on standard indexing techniques to scale up the search to millions of items, as the network output in this case is no more a single vector, but instead a set of vectors, one for each contextualized word or region. We try to handle this problem in Section 4.2, where we introduce the Bag of Concepts model.

The TERAN architecture is shown in Figure 4.6. We left in the scheme the I-CLS and T-CLS tokens connections for comparison with the TERN architecture. These tokens are still used for a targeted experiment that exploits the combination of the TERN and TERAN losses (more details in Section 5.2.3). However, they are not used in the main TERAN experiments. The two linear projection layers within the TE modules are

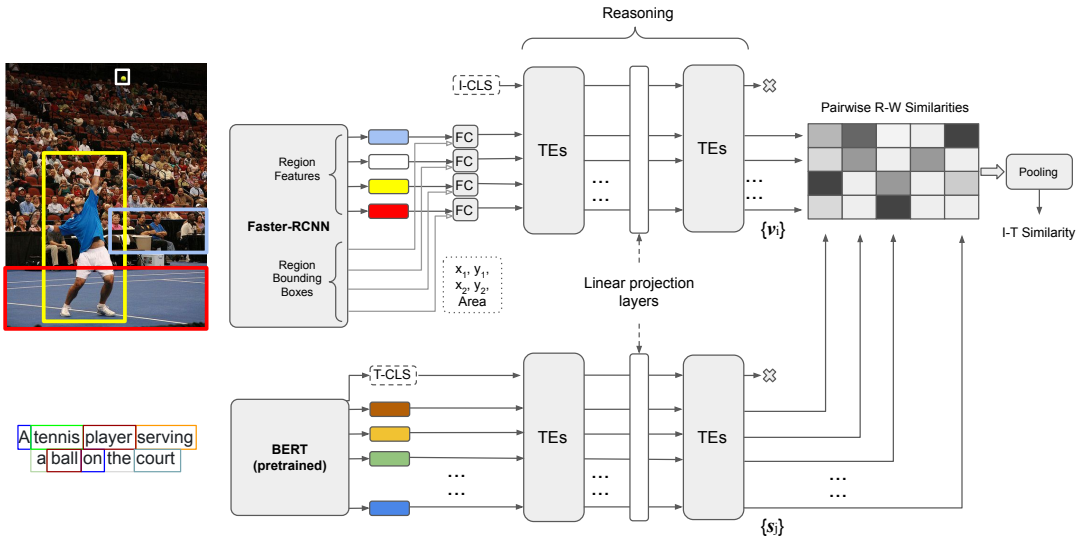


Figure 4.6: *The proposed TERAN architecture. TEs stands for Transformer Encoders. Region and word features are extracted through a bottom-up attention model based on Faster-RCNN and BERT, respectively. The final image-text (I-T) similarity score is obtained by pooling a region-word (R-W) similarity matrix. Notice that the special I-CLS and T-CLS are not used in the basic formulation of TERAN.*

used to project the visual and textual concepts in spaces having the same dimensionality. Then, the latest TE layers perform further processing before outputting the final features that are used to compute the final alignment loss. Differently from TERN, we initially do not share the weights of the last TE layers. We discuss the effect of weight sharing in our ablation study (Section 4.1.7).

In our novel TERAN architecture, the features in output from the last TE layers are used to compute a region-word alignment matrix $\mathbf{A} \in \mathbb{R}^{|\mathbf{g}_k| \times |\mathbf{g}_l|}$, where $\mathbf{g}_k$ is the set of indexes of the region features from the k-th image and $\mathbf{g}_l$ is the set of indexes of the words from the l-th sentence. We use the cosine similarity for measuring the affinity between the i-th region and the j-th word. If $\{\boldsymbol{v}_i\}$ and $\{\boldsymbol{s}_j\}$ are the sets of contextualized region and word vectors in output from the network for the k-th image and the l-th sentence respectively, then $\mathbf{A}$ is constructed as:

$$A_{ij} = \frac{\boldsymbol{v}_i^\top \boldsymbol{s}_j}{\|\boldsymbol{v}_i\| \|\boldsymbol{s}_j\|} \qquad i \in \mathbf{g}_k, j \in \mathbf{g}_l \tag{4.6}$$

At this point, the global similarity S_{kl} between the k-th image and the l-th sentence is computed by pooling this similarity matrix through an appropriate pooling function. Inspired by [109] and [122], we employ the max-sum pooling, which consists in computing the max over the rows of A and then summing or, equivalently, *max-over-regions sum-over-words* ($M_r S_w$) pooling. We explore also the dual version, as in [122], by computing the max over the columns and then summing, or *max-over-words sum-over-regions* ($M_w S_r$) pooling:

$$S_{kl}^{M_r S_w} = \sum_{j \in \mathbf{g}_l} \max_{i \in \mathbf{g}_k} A_{ij} \qquad or \qquad S_{kl}^{M_w S_r} = \sum_{i \in \mathbf{g}_k} \max_{j \in \mathbf{g}_l} A_{ij} \tag{4.7}$$

Since both these similarity functions are not symmetric due to the diverse outcomes we obtain by inverting the order of the sum and max operations, we introduce also the symmetric form, obtained by summing the two:

$$S_{kl}^{\text{Symm}} = S_{kl}^{M_r S_w} + S_{kl}^{M_w S_r} \tag{4.8}$$

Learning Given the global image-sentence similarities S_{kl} computed through alignments pooling, we can proceed as in TERN as well as in previous works [65, 125], using the contrastive learning method with emphasis on hard negatives:

$$\begin{aligned} \mathcal{L}_{kl} = & \max_{l'}[\alpha + S_{kl'} - S_{kl}]_+ + \\ & \max_{k'}[\alpha + S_{k'l} - S_{kl}]_+ \end{aligned} \tag{4.9}$$

where $[x]_+ \equiv max(0, x)$ and α is a margin that defines the minimum separation that should hold between the truly matching word-region embeddings and the negative pairs; k' and l' are the hard negative examples.

4.1.5 Experimental Setup

We trained the TERN and TERAN architectures and we measured their performance on the MS-COCO [134] and Flickr30k datasets [248], computing the effectiveness of our approach on the image retrieval and sentence retrieval tasks. We compared our results against state-of-the-art approaches on the same datasets, using the introduced NDCG and the already-in-use Recall@K metrics.

The MS-COCO dataset comes with a total of 123,287 images. Every image has associated a set of 5 human-written captions describing the image. We follow the splits introduced by [109] and followed by the subsequent works in this field [65, 77, 125]. In particular, 113,287 images are reserved for training, 5,000 for validating, and 5,000 for testing. Differently, Flickr30k consists of 31,000 images and 158,915 English texts. Like MS-COCO, each image is annotated with 5 captions. Following the splits by [109], we use 29,000 images for training, 1,000 images for validation, and the remaining 1,000 images for testing. For MS-COCO, at test time the results for both 5k and 1k test-sets are reported. In the case of 1k images, the results are computed by performing 5-fold cross-validation on the 5k test split and averaging the outcomes.

We computed the relevance scores for the NDCG metric using `ROUGE-L`[133] and `SPICE`[9], as explained in Section 4.1.2, and we set the NDCG parameter $p = 25$ as in [41] in our experiments. We employed the NDCG metrics measured during the validation phase for choosing the best performing model to be used during the test phase.

Implementation Details We employ the BERT model pre-trained on the masked language task on English sentences, using the PyTorch implementation by HuggingFace[1]. These pre-trained BERT embeddings are 768-D. For the visual pipeline, we extracted the bottom-up features from the work by [10], using the code and pre-extracted features provided by the authors[2]. Specifically, for MS-COCO we used the already-extracted

[1] `https://github.com/huggingface/transformers`
[2] `https://github.com/peteanderson80/bottom-up-attention`

bottom-up features, while we extracted from scratch the features for Flickr30k using the available pre-trained model. In the experiments, we used the bottom-up features containing the top 36 most confident detections, although our pipeline already handles variable-length sets of regions for each image by appropriately masking the attention weights in the TE layers. Concerning the reasoning steps, we used a stack of 4 TE layers for visual reasoning. We found the best results when fine-tuning the BERT pre-trained model, so we did not add further reasoning TE layers for the textual pipeline. The final common space, as in [65], is 1024-D. We linearly projected the visual and textual features to a 1024-D space and then we processed the resulting features using 2 final TEs before computing the alignment matrix. All the TEs feed-forward layers are 2048-dimensional and the dropout is set to 0.1. We trained for 30 epochs using Adam optimizer with a batch size of 40 and a learning rate of $1e{-}5$ for the first 20 epochs and $1e{-}6$ for the remaining 10 epochs. The α parameter of the hinge-based triplet ranking loss is set to 0.2, as in [65, 125].

4.1.6 Results

We compare our methods against the following baselines: JGCAR [228], SAN [97], VSE++ [65], SMAN [99], M3A-Net [98], AAMEL [231], MRNN [109], SCAN [122], SAEM [235], CASC [239], MMCA [232], VSRN [125], PFAN [230], Full-IMRAM [43], and CAMERA [187]. We clustered these methods based on the visual feature extractor they use: VGG, ResNet, or Region CNN (e.g., Faster-RCNN). To obtain a better comparison with our method, we also annotated in the tables whenever they use BERT as the textual model, or if they use disentangled visual-textual pipelines for efficient feature computation. For better comparing with TERAN with TERN, we included three more targeted experiments. In the first two, called *TERN* M_wS_r *Test* and *TERN* M_rS_w *Test* we used the best-performing TERN model, trained as explained in Section 4.1.3, testing it using the M_wS_r and M_rS_w alignments criteria respectively. TERN is effectively able to output features for every image region or word; however, it is never constrained to produce meaningful descriptions out of these sets of features; hence, this experiment is aimed at checking the quality of the alignment of the concepts in output from the previous TERN architecture. In the third experiment, called *TERN w. Align*, we tried to integrate the objectives of both TERN and TERAN during training, by combining their losses using the uncertainty weighting method proposed in [110], and testing the model using the TERN inference protocol. Thus, in this experiment, we effectively reuse the I-CLS and T-CLS tokens as global descriptions for images and sentences, as described in Section 4.1.3. This experiment aimed to evaluate if the TERAN alignment objective can help TERN learn better fixed-sized global vectorial descriptions. Also notice that many of the listed methods report the results using an ensemble of two models having different training initialization parameters, where the final similarity is obtained by averaging the scores in output from each model. Hence, we reported also our ensemble results, for a better comparison with these baselines. In the tables, we indicate ensemble methods postponing *(ens.)* to the method name. We used the original implementations from their respective GitHub repositories to compute the NDCG metrics for the baselines, where possible. In the case of missing pre-trained models, we were not able to produce consistent results with the original papers. In this case, we do not report the NDCG metrics.

	Image Retrieval					Sentence Retrieval				
	Recall@K			NDCG		Recall@K			NDCG	
Model	K=1	K=5	K=10	ROUGE-L	SPICE	K=1	K=5	K=10	ROUGE-L	SPICE
(VGG)										
JGCAR [228]	40.2	74.8	85.7	-	-	52.7	82.6	90.5	-	-
SAN [97]	60.8	90.3	95.7	-	-	74.9	94.9	98.2	-	-
(ResNet)										
VSE++ [65] †	52.0	84.3	92.0	0.712	0.617	64.6	90.0	95.7	0.705	0.658
SMAN [99]	58.8	87.4	93.5	-	-	68.4	91.3	96.6	-	-
M3A-Net [98]	58.4	87.1	94.0	-	-	70.4	91.7	96.8	-	-
AAMEL [231]	59.9	89.0	95.1	-	-	74.3	95.4	98.2	-	-
(Region CNN)										
MRNN [109]	27.4	60.2	74.8	-	-	38.4	69.9	80.5	-	-
SCAN (ens.) [122]	58.8	88.4	94.8	-	-	72.7	94.8	98.4	-	-
SAEM (ens.)[235] §†	57.8	88.6	94.9	-	-	71.2	94.1	97.7	-	-
CASC [239]	58.9	89.8	96.0	-	-	72.3	96.0	**99.0**	-	-
MMCA [232] §	61.6	89.8	95.2	-	-	74.8	95.6	97.7	-	-
VSRN [125] †	60.8	88.4	94.1	0.723	0.621	74.0	94.3	97.8	0.737	0.690
VSRN (ens.) [125] †	62.8	89.7	95.1	0.732	0.637	76.2	94.8	98.2	0.748	0.704
PFAN (ens.) [230]	61.6	89.6	95.2	-	-	76.5	96.3	**99.0**	-	-
Full-IMRAM [43]	61.7	89.1	95.0	-	-	76.7	95.6	98.5	-	-
CAMERA [187] §†	62.3	90.1	95.2	-	-	75.9	95.5	98.6	-	-
CAMERA (ens.) [187] §†	63.4	90.9	95.8	-	-	77.5	96.3	98.8	-	-
TERN	51.9	85.6	93.6	0.725	0.653	63.7	90.5	96.2	0.716	0.674
TERN M_rS_wTest	51.5	84.9	93.1	0.722	0.642	26.6	70.3	86.3	0.568	0.530
TERN M_wS_rTest	51.2	84.6	92.9	0.722	0.643	61.9	88.9	95.7	0.713	0.666
TERN w. Align	54.5	86.9	94.2	0.724	0.643	65.5	91.0	96.5	0.720	0.675
TERAN Symm.	63.5	91.1	96.3	0.739	0.666	76.3	95.3	98.4	0.741	0.701
TERAN M_wS_r	57.5	88.4	94.9	0.730	0.658	70.8	93.5	97.3	0.725	0.681
TERAN M_rS_w	65.0	91.2	96.4	0.741	0.668	77.7	95.9	98.6	0.746	0.707
TERAN M_rS_w (ens.)	**67.0**	**92.2**	**96.9**	**0.747**	**0.680**	**80.2**	**96.6**	**99.0**	**0.756**	**0.720**

Table 4.1: *Results on the MS-COCO dataset, on the 1k test set.*

§ Uses BERT as language model

† Uses disentangled visual-textual pipelines

	Image Retrieval					Sentence Retrieval				
	Recall@K			NDCG		Recall@K			NDCG	
Model	K=1	K=5	K=10	ROUGE-L	SPICE	K=1	K=5	K=10	ROUGE-L	SPICE
(ResNet)										
VSE++ [65] †	30.3	59.4	72.4	0.656	0.577	41.3	71.1	81.2	0.597	0.551
M3A-Net [98]	38.3	65.7	76.9	-	-	48.9	75.2	84.4	-	-
AAMEL [231]	39.9	71.3	81.7	-	-	51.9	84.2	91.2	-	-
(Region CNN)										
MRNN [109]	10.7	29.6	42.2	-	-	16.5	39.2	52.0	-	-
SCAN (ens.) [122]	38.6	69.3	80.4	-	-	50.4	82.2	90.0	-	-
VSRN [125] †	37.9	68.5	79.4	0.676	0.596	50.3	79.6	87.9	0.639	0.598
VSRN (ens.) [125] †	40.5	70.6	81.1	0.684	0.609	53.0	81.1	89.4	0.652	0.612
Full-IMRAM [43]	39.7	69.1	79.8	-	-	53.7	83.2	91.0	-	-
MMCA [232] §	38.7	69.7	80.8	-	-	54.0	82.5	90.7	-	-
CAMERA [187] §†	39.0	70.5	81.5	-	-	53.1	81.3	89.8	-	-
CAMERA (ens.) [187] §†	40.5	71.7	82.5	-	-	55.1	82.9	91.2	-	-
TERN	28.7	59.7	72.7	0.665	0.599	38.4	69.5	81.3	0.601	0.556
TERN M_rS_wTest	28.3	59.1	72.2	0.663	0.592	6.8	28.4	46.7	0.406	0.372
TERN M_wS_rTest	28.1	58.6	71.8	0.663	0.592	35.5	67.5	78.9	0.600	0.551
TERN w. Align	31.4	62.5	75.3	0.667	0.597	40.2	71.1	81.9	0.606	0.561
TERAN Symm.	41.0	71.6	82.3	0.680	0.607	54.8	82.7	90.9	0.641	0.601
TERAN M_wS_r	34.1	65.7	77.8	0.669	0.596	45.3	76.3	86.2	0.611	0.564
TERAN M_rS_w	42.6	72.5	82.9	0.682	0.610	55.6	83.9	91.6	0.643	0.606
TERAN M_rS_w(ens.)	**45.1**	**74.6**	**84.4**	**0.689**	**0.622**	**59.3**	**85.8**	**92.4**	**0.658**	**0.624**

Table 4.2: *Results on the MS-COCO dataset, on the 5k test set.*

§ Uses BERT as language model

† Uses disentangled visual-textual pipelines

Although TERN can not compete with current state-of-the-art architectures as far as Recall@K metric is concerned, it can challenge other very efficient methods like VSE++ or VSRN in terms of NDCG. Due to the high-level abstraction nature of the SPICE relevance, this result confirms the ability of TERN to understand complex patterns and abstract concepts both in the visual and textual inputs. On both the 1k and 5k test sets, the TERAN approach reaches state-of-the-art results on almost all the metrics. Concerning the results reported in Table 4.1 regarding 1k test set, the best performing TERAN model is the one implementing the max-over-regions sum-over-words (M_rS_w) pooling method, although the model using the symmetric loss reaches comparable results. We chose the same TERAN M_rS_w model to evaluate the ensemble, reaching an improvement of 5.7% and 3.5% on the Recall@1 metric on image and sentence retrieval respectively, with respect to the best baseline using ensemble methods, which is CAMERA [187]. Notice, however, that even the basic TERAN model without ensemble is able to surpass CAMERA in many metrics. This confirms the power of the TERAN model despite its overall simplicity.

Table 4.2 reports the results for the 5k test set, which confirm the superiority of TERAN M_rS_w over all the baselines also on the full test set. In this scenario, we increase the Recall@1 performance by 11.3% and 7.6% on image and sentence re-

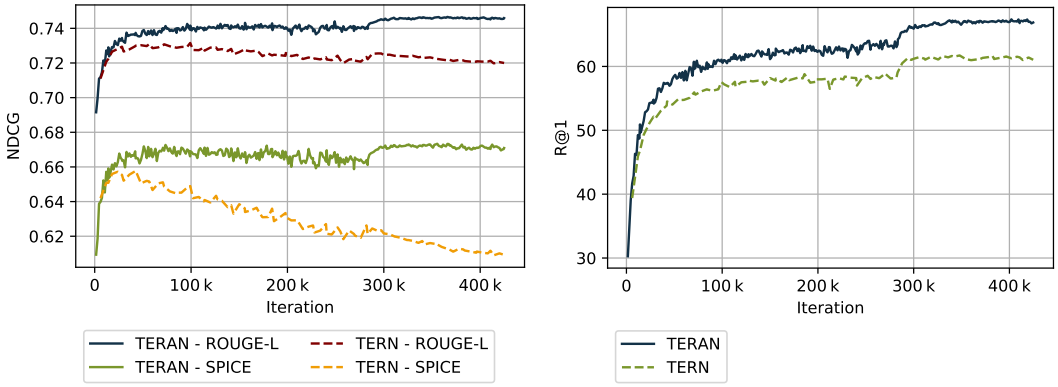


Figure 4.7: *Validation metrics on sentence-to-image retrieval, measured during the training phase, for the average-over-sentences scenario. TERN overfits on the NDCG metrics, while Recall@1 still improves. TERAN instead generalizes better on both metrics.*

trieval with respect to the CAMERA approach. On the other hand, the max-over-words sum-over-regions (M_wS_r) method loses around 10% on the Recall@1 metrics with respect to the best performing TERAN non-ensemble model. In this case, the Recall@K metric does not improve over the top results obtained by the current state-of-the-art approaches. Nevertheless, this model loses only about 1.5% during image-retrieval and about 3.5% during sentence-retrieval as far as the NDCG with the `SPICE` relevance is concerned, reaching perfectly comparable results with our state-of-the-art method. In light of these results, we deduce that the M_wS_r model is not so effective in retrieving the exact-matching elements; however, it is still very good at retrieving the relevant ones.

As far as image retrieval is concerned, in the *TERN* M_wS_r*Test* and *TERN* M_wS_r*Test* experiments we can see that the TERN architecture performs fairly good when the similarity is computed as in the novel TERAN architecture, using the region and words outputs and not the I-CLS and T-CLS global descriptions. In particular, the use of max-over-words sum-over-regions similarity still works quite well compared to the similarity computed through I-CLS and T-CLS global visual and textual features as it is in TERN.

Notice instead that on the sentence retrieval task, the *TERN* M_rS_w *Test* experiment obtains a very low performance. This is the consequence of the fact that TERN is trained to produce global-scale image-sentence matchings, while it is never forced to produce meaningful fine-grained aligned concepts. This is further supported by the evidence that if we visualize the region-words alignments as further explained in Section 4.1.7 we obtain random word groundings on the image, meaning that the concepts in output from TERN are not sufficiently informative.

In order to better compare TERAN with the TERN approach, in Figure 4.7 we report the validation curves for both NDCG and Recall@1 metrics, for both methods. We can notice how the NDCG metric overfits in the TERN model, especially when using the SPICE metric, while the Recall@K keeps increasing. On the other hand, TERAN demonstrates better generalization abilities on both metrics. This is a clear indication that TERAN is better at retrieving relevant items in the first positions, as well as exact matching elements. Instead, TERN is more prone to overfitting to the SPICE metric,

	Image Retrieval					Sentence Retrieval				
	Recall@K			NDCG		Recall@K			NDCG	
Model	K=1	K=5	K=10	ROUGE-L	SPICE	K=1	K=5	K=10	ROUGE-L	SPICE
					(VGG)					
JGCAR [228]	35.2	62.0	72.4	-	-	44.9	75.3	82.7	-	-
SAN [97]	51.4	77.2	85.2	-	-	67.0	88.0	94.6	-	-
					(ResNet)					
VSE++ [65] †	39.6	70.1	79.5	0.631	0.494	52.9	80.5	87.2	0.601	0.514
TIMAM [207] §†	42.6	71.6	81.9	-	-	53.1	78.8	87.6	-	-
SMAN [99]	43.4	73.7	83.4	-	-	57.3	85.3	92.2	-	-
M3A-Net [98]	44.7	72.4	81.1	-	-	58.1	82.8	90.1	-	-
AAMEL [231]	49.7	79.2	86.4	-	-	68.5	91.2	95.9	-	-
					(Region CNN)					
MRNN [109]	15.2	37.7	50.5	-	-	22.2	48.2	61.4	-	-
SCAN (ens.) [122]	48.6	77.7	85.2	-	-	67.4	90.3	95.8	-	-
PFAN (ens.) [230]	50.4	78.7	86.1	-	-	70.0	91.8	95.0	-	-
SAEM (ens.)[235] §†	52.4	81.1	88.1	-	-	69.1	91.0	95.1	-	-
VSRN [125] †	53.0	77.9	85.7	0.673	0.545	70.4	89.2	93.7	0.676	0.592
VSRN (ens.) [125] †	54.7	81.8	88.2	0.680	0.556	71.3	90.6	96.0	0.688	0.606
Full-IMRAM [43]	53.9	79.4	87.2	-	-	74.1	93.0	96.6	-	-
MMCA [232] §	54.8	81.4	87.8	-	-	74.2	92.8	96.4	-	-
CASC [239]	60.2	78.3	86.3	-	-	68.5	90.6	95.9	-	-
CAMERA [187] §†	58.9	84.7	90.2	-	-	76.5	**95.1**	97.2	-	-
CAMERA (ens.) [187] §†	60.3	85.9	91.7	-	-	78.0	**95.1**	**97.9**	-	-
TERN	41.1	71.9	81.2	0.647	0.512	53.2	79.4	86.0	0.624	0.529
TERAN Symm.	55.7	83.1	89.3	0.678	0.555	71.8	90.5	94.7	0.676	0.603
TERAN M_wS_r	49.4	78.3	85.9	0.664	0.536	60.5	85.1	92.2	0.651	0.558
TERAN M_rS_w	59.5	84.9	90.6	0.686	0.564	75.8	93.2	96.7	0.687	0.614
TERAN M_rS_w (ens.)	**63.1**	**87.3**	**92.6**	**0.695**	**0.577**	**79.2**	94.4	96.8	**0.707**	**0.636**

Table 4.3: *Results on the Flickr30k dataset.*

§ Uses BERT as language model
† Uses disentangled visual-textual pipelines

meaning that at a certain point in training, the network still searches for the top matching element, but with a tendency to push away possible relevant results. However, looking at the results from the *TERN w. Align* experiment, we can notice that by augmenting the TERN objective with the TERAN alignment loss, we can slightly increase the TERN overall performance. This confirms that a more precise and meaningful region-word alignment works as a nice regularization term, that has a visible effect on the quality of the fixed-sized global embeddings.

In Table 4.3 we report the results on the Flickr30k dataset. Our single-model TERAN M_rS_w method outperforms the best baseline (CAMERA) on the image retrieval task while approaching the single-model CAMERA performance on the sentence retrieval task. Nevertheless, even on Flickr30k our TERAN M_rS_w method with model ensemble obtains state-of-the-art results with respect to all the baselines on all the metrics, gaining 4.6% and 1.5% on the Recall@1 metric on the image and sentence retrieval tasks respectively.

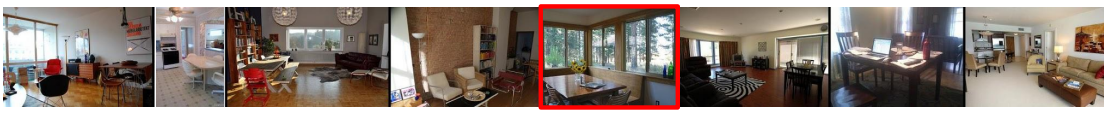
On the MS-COCO dataset, our system powered by a single GTX 1080Ti can compute a single image-to-sentence query in $\sim 0.12s$ on 5k sentences of the test split; in the sentence-to-image scenario, it can produce scores and rank the 1k images in $\sim 0.02s$. Although the TERAN features are not directly indexable using standard vector indexes, these timings allow TERAN to be effectively used, for example, in a re-ranking phase, where the first items have been previously retrieved using a faster descriptor (e.g., TERN).

Qualitative Analysis for Image Retrieval The visualization of image retrieval results is a good way to qualitatively appreciate the retrieval abilities of the proposed TERAN model. Figures 4.8 and 4.9 show examples of images retrieved given a textual caption as a query, with scores computed using the max-over-regions sum-over-words method. In particular, Figure 4.8 shows image retrieval results for a couple of non specific query captions. The red-marked images represent the exact-matching elements from the ground-truth. The retrieved images in these examples are incorrect results for the Recall@1 metric (and for the first query even for Recall@5). Nevertheless, in the very first positions, we find non-matching yet relevant images, due to the ambiguity of the query caption. These are common examples where NDCG succeeds over the Recall@K metric since we need a relaxed evaluation for generic query captions. Figure 4.9 reports instead image retrieval results for a couple of very specific query captions. For the first two queries, the network succeeds in positioning the only really relevant image in the first position (*a dog sitting on a bench* on the upper query, and *Pennsylvania Avenue*, uniquely identifiable by the street sign, on the lower query). In this case, the Recall@1 metric also succeeds, given that the query captions are very selective. The third example of Figure 4.9, instead, evidences a failure case where the model cannot deal with very subtle details. The (only) correct result is ranked 6th in this case; in the first ranking positions, the model can find images with a vase used as a centerpiece, but the table is not often visible, and when it is visible, it is not in the corner of the room.

4.1.7 Ablation Study

The Effect of Weight Sharing We tried to apply weight sharing for the last two layers of the TERAN architecture, those after the linear projection to the 1024-D space. Weight sharing is used to reduce the size of the network and enforce a structure able to perform common reasoning on the high-level concepts, possibly reducing the overfitting and increasing the stability of the whole network. We experimented with the effects of weight sharing on the MS-COCO dataset with 1k test set, for both the max-over-words sum-over-regions and the max-over-regions sum-over-words scenarios. Results are shown in the 2-nd and 6-th rows of Table 4.4. It can be noticed that the values are perfectly comparable with the TERAN results reported in Table 4.1, suggesting that at this point in the network the abstraction is high enough that concepts coming from images and sentences can be processed in the exact same way. This result shows that vectors at this stage have been freed from any modality bias and they are fully comparable in the same representation space. Also, in the max-over-words sum-over-regions scenario (6-th row), there is a small gain both in terms of Recall@K and NDCG. This confirms the slight regularization effect of the weight sharing approach.

Query: A large jetliner sitting on top of an airport runway.



Query: An eating area with a table and a few chairs.

Figure 4.8: *Example of image retrieval results for a couple of generic query captions. These are common examples where NDCG succeeds over the Recall@K metric. The ground-truth matching image is not among the very first positions; however, the top-ranked images are also visually very relevant.*

Query: A large white dog is sitting on a bench beside an elderly man.

Query: An old black and white photo of Pennsylvania Avenue.

Query: Table situated in corner of room with a vase for a center piece.

Figure 4.9: *Example of image retrieval results for a couple of very specific query captions.*

Averaging Versus Summing We tried to compute the average instead of the sum during the last pooling phase of the alignment matrix. We consider only the case in which we average-over-sentences; in fact, since in our experiments the number of visual concepts is always fixed to the 36 more influent ones during the object detection stage, average-over-regions and sum-over-regions do not differ substantially. Thus, we considered the case of max-over-regions average-over-words ($M_r\text{Avg}_w$):

$$S_{kl} = \frac{\sum_{j \in \mathbf{g}_l} \max_{i \in \mathbf{g}_k} A_{ij}}{|\mathbf{g}_l|}. \tag{4.10}$$

If we compute the average instead of the sum in the max-over-regions sum-over-words scenario, the final similarity score between the image and the sentence is no more dependent on the number of concepts from the textual pipeline: the similarities are averaged and not accumulated. In the 3-rd row of Table 4.4 we can notice that by averaging we lose an important amount of information with respect to the max-over-regions sum-over-words scenario (1-st row). This insight suggests that the complexity of the query is beneficial for achieving high-quality matching. Another side effect of using average instead of the max is the premature clear overfitting on the NDCG metrics as far as image-retrieval is concerned. The effect is shown in Figure 4.10. The clear overfitting of the NDCG metrics resembles the training curve trajectories of TERN (Figure 4.7).

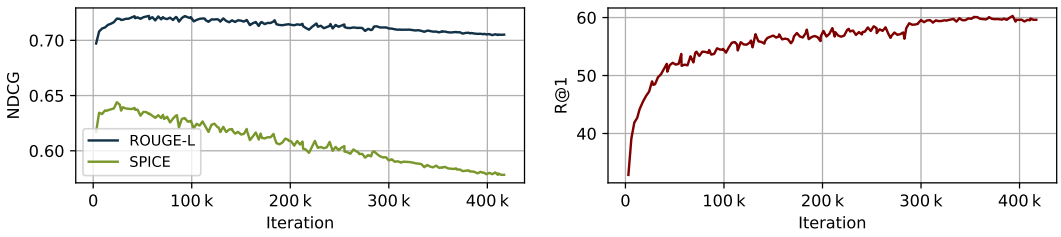


Figure 4.10: *Validation metrics measured during the training phase, for the average-over-sentences scenario. This model overfits on the NDCG metrics on the image-retrieval task, while Recall@1 still improves.*

This result demonstrates that although this model can correctly perform exact matching, it is pulling away relevant results from the head of the ranked list of images, during the validation phase.

	Image Retrieval					Sentence Retrieval				
	Recall@K			NDCG		Recall@K			NDCG	
Model	K=1	K=5	K=10	ROUGE-L	SPICE	K=1	K=5	K=10	ROUGE-L	SPICE
M_rS_w *(Table 4.1)*	65.0	91.2	96.4	0.741	0.668	77.7	95.9	98.6	0.746	0.707
M_rS_w Shared-W	64.5	91.3	96.3	0.740	0.667	77.3	95.9	98.4	0.746	0.706
$M_r\text{Avg}_w$	57.2	87.6	93.6	0.705	0.587	68.6	92.4	96.7	0.721	0.671
M_rS_w StopWordsFilter	64.2	91.1	96.3	0.737	0.658	76.8	95.9	98.6	0.745	0.705
M_rS_w Bi-LSTM	55.6	86.9	93.9	0.734	0.666	67.4	92.5	96.9	0.717	0.677
M_rS_w Bi-GRU	56.3	87.1	94.0	0.735	0.666	69.1	93.4	97.1	0.720	0.678
M_wS_r *(Table 4.1)*	57.5	88.4	94.9	0.730	0.658	70.8	93.5	97.3	0.725	0.681
M_wS_r Shared-W	58.1	88.4	95.0	0.730	0.657	71.1	93.1	97.7	0.728	0.683

Table 4.4: *Results for the ablation study experiments. We organize the methods in the table clustering them by the pooling method, for an easier comparison (max-over-regions methods in the upper part and max-over-words methods on the lower part). In the first row of both sections we report the TERAN results from Table 4.1. Experiments are computed on the MS-COCO dataset, 1k test set.*

Removing Stop-Words During Alignment Some words may carry no substantial meaning by themselves, such as articles or prepositions. These words with a high and diffuse frequency of use are typically called *stop-words* and are usually removed in classical text analysis. In this context, removing stop-words may help the architecture to focus only on the important concepts. Doing so, the training process is simplified as the noise introduced by possibly irrelevant words is removed. Results are reported in the 4-th row of Table 4.4. The overall performance, both in terms of Recall@ and NDCG is comparable, yet with a small decrease, with the one obtained without stop-words removal (1-st row of the table). This suggests that in this context stop-words are linguistic elements that bring some useful information to distinguish ambiguous scenes. Prepositions and adverbs often indicate the spatial arrangement of objects, thus "chair *near* the table" is not the same as "chair *over* the table". Distinguishing these fine-grained differences is beneficial for obtaining a precise image-text matching.

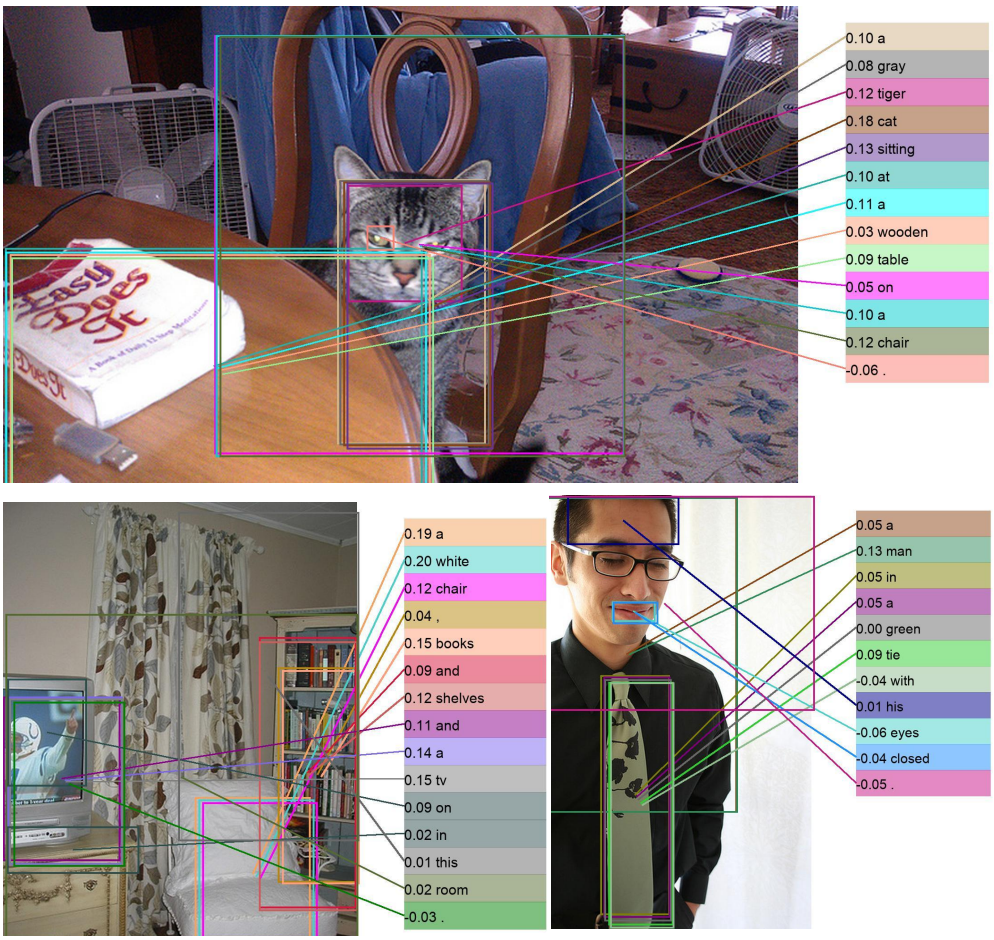


Figure 4.11: *Visualization of the word-region alignments. Near each word, we report the cosine similarity computed between that word and the top-relevant image region associated with it. We slightly offset the overlapping bounding-boxes for a better visualization.*

Using Different Language Models Despite the power of BERT [58] for obtaining contextualized representations for the words in a sentence, many works use recurrent bidirectional networks instead, such as Bi-GRU or Bi-LSTMs. In the 5-th and 6-th row of Table 4.4 we report the results for the TERAN model with Bi-GRU and Bi-LSTM in substitution of the BERT model for language processing. We used 300-D word embeddings, and a hidden size of 512 so that the final bi-directional sentence feature is a 1024-D description; we used the same training protocol and hyper-parameters used for the main experiments. The results suggest that BERT is an essential ingredient for reaching top results on the Recall@K metrics, especially when K={1, 5}. In particular, Bi-LSTM and Bi-GRU lose around 14% on image retrieval and 12% on sentence retrieval on the Recall@1 metric compared to the TERAN $M_r S_w$ single-model method. However, we can notice that TERAN with these recurrent language models still maintains a comparable performance with respect to the NDCG metric, especially on the image retrieval task.

Visualizing the Visual-Word Alignments Inspired by the work in [109], we try to visualize the region-word alignments learned by TERAN on some images from the test set of

MS-COCO dataset. We recall that no supervision was used at the region-word level during the training phase. In Figure 4.11, we report some figures where every sentence word has been associated with the top-relevant image region. The affinity between visual concepts (region features) and textual concepts (word features) has been measured through cosine similarity, just as during the training phase. We can see that the words have overall plausible groundings on the image they describe. Some words are really difficult to ground, such as articles, verbs, or adjectives. However, we can notice that phrases describing a visual entity and composed of nouns with the related articles and adjectives (e.g. *"a green tie"*, or *"a wooden table"*) are often grounded to the same region. This further confirms that the TERAN architecture can produce meaningful concepts, and it is also able to cluster them under the form of complete reasonable phrases. We can notice some wrong word groundings in the images, such as the phrase *"eyes closed"* that is associated with the region depicting the closed mouth. In this case, the error seems to lie on some localized misunderstanding of the scene (in this case the noun *"eyes"* has probably been misunderstood since the mouth and the eyes are both closed). Overall, however, complex scenes are correctly broken down into their salient elements, and only the key regions are attended.

4.2 Towards Large-scale Cross-modal Retrieval

The results presented so far addressed the effectiveness of image-to-text and text-to-image retrieval scenarios. In this paragraph we discuss some methods for scaling up this research to possibly millions of images or sentences. In particular, we try to sparsify and quantize the TERN features so that it is possible to use them in text-based indexing engines, using Surrogate Text Representation (STR).

4.2.1 Surrogate Text Representation (STR)

There are many possibilities to index a vectorial representation for efficiently performing KNN search. Although there are many efficient libraries for efficient retrieval in vector spaces, like FAISS[3], recent approaches [70, 6] proposed to use off-the-shelf text-based indexing tools, transforming a real-valued vector into a textual representation. The latter approach has the enormous advantage of exploiting already-implemented very efficient text-based indexing engines, which are nowadays very stable and effective for searching using the *vector space* document representation model [47].

The approach used in [70, 6, 40] to generate a document from a real-valued vectorial representation is called Surrogate Text Representation (STR). The STR of an object is a space-separated concatenation of some alphanumeric codeword selected from a pre-defined dictionary. The concept is quite simple: the original vector $\mathbf{x} \in \mathbb{R}^n$ is transformed into a positive-integer vector $\mathbf{z} \in \mathbb{N}^n$ by using a certain function f, such that $\mathbf{z} = f(\mathbf{x})$. Given a set of symbols $\{\tau_1, \tau_2, \ldots, \tau_n\}$, the textual document $\text{STR}_f(\mathbf{x})$ for the input feature $\mathbf{x}$ is obtained considering the i-th integer value of $\mathbf{z} = f(\mathbf{x})$ as the term frequency of the codeword τ_i. For example, given $\mathbf{z} = f(\mathbf{x}) = [3, 0, 1, 2]$ and a codebook $\{\tau_1 = \text{"A"}, \tau_2 = \text{"B"}, \tau_3 = \text{"C"}, \tau_4 = \text{"D"}\}$, we have that $\text{STR}_f(\mathbf{x}) =$ "A A A C D D". In this way, the text engine creates an internal representation of this document by counting the number of occurrencies of each word — the term frequencies

[3] https://github.com/facebookresearch/faiss

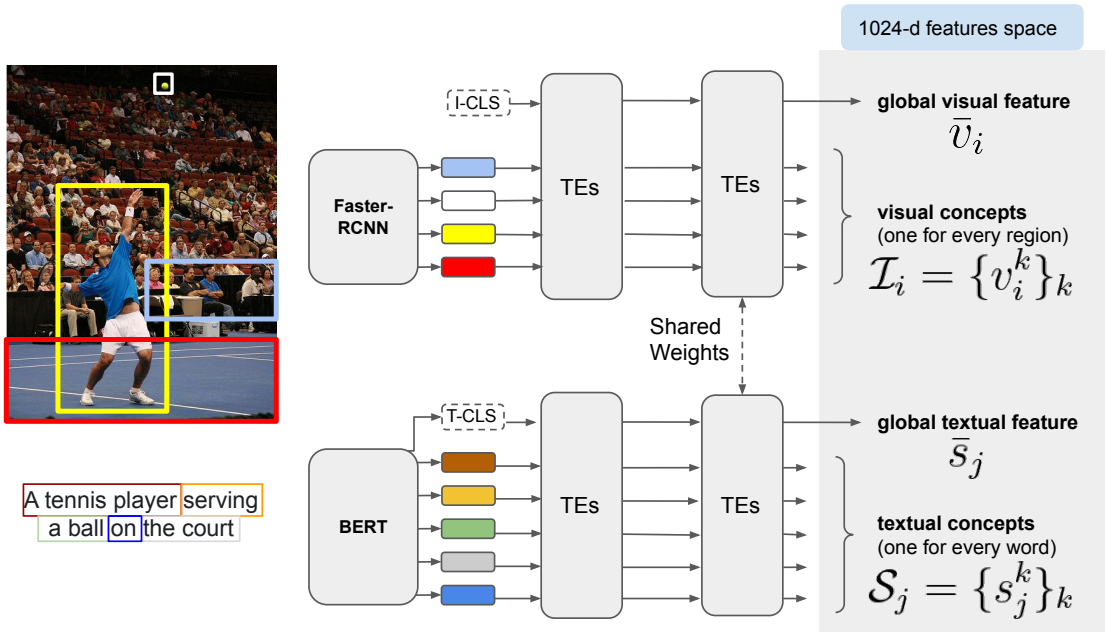


Figure 4.12: *The output from the TERN and TERAN architecture consists of both global features summing up whole images and whole sentences, as well as contextualized concepts lying in the same common space.*

— and it uses the well-known TF scheme and the document vector model for easily and efficiently compute dot product similarities.

Given the above described method, there are possible choices of f for performing the real-to-positive-integer vector transformation. Ideally, f should be (a) *order-preserving*, so that the ranked results do not change when the document $\text{STR}_f(\mathbf{x})$ is used instead of the original vector $\mathbf{x}$, and (b) *sparsifying*, which means that the resulting vector $\mathbf{z}$ should be sparse, so that every document does not contain all the symbols.

From this point on, we take for granted the Surrogate Text Representation procedure, and we focus on the possibilities we have for implementing f. The first one deals with the sparsification of the $\bar{\boldsymbol{v}}_i$ and $\bar{\boldsymbol{s}}_j$ global vectors (the ones from TERN) for images and sentences respectively, using approaches like deep permutations [3] or scalar quantization [6]. The second possibility concerns the use of the fine-grained concepts extracted from the image regions $\mathcal{I}_i$ and the sentence words $\mathcal{S}_j$ (the ones from TERAN). In particular, to handle the set of TERAN vectors, we propose to construct a Bag-of-Concepts (BoC) model, where images and sentences are described as sets of elementary components. This model can be used to create a sparse description of images and sentences over a fixed-sized dictionary of reference concepts. Following, we describe in detail these two methodologies.

4.2.2 Dealing with Global Descriptions

The 1024-D vectors $\bar{\boldsymbol{v}}_i$ and $\bar{\boldsymbol{s}}_j$ describing whole images and sentences can be transformed into a suitable vector indexable with inverted lists by applying the procedures described by [3, 6]. Inverted lists and surrogate text representation need and sparse and quantized representations, as they work with frequencies of appearance of terms in a

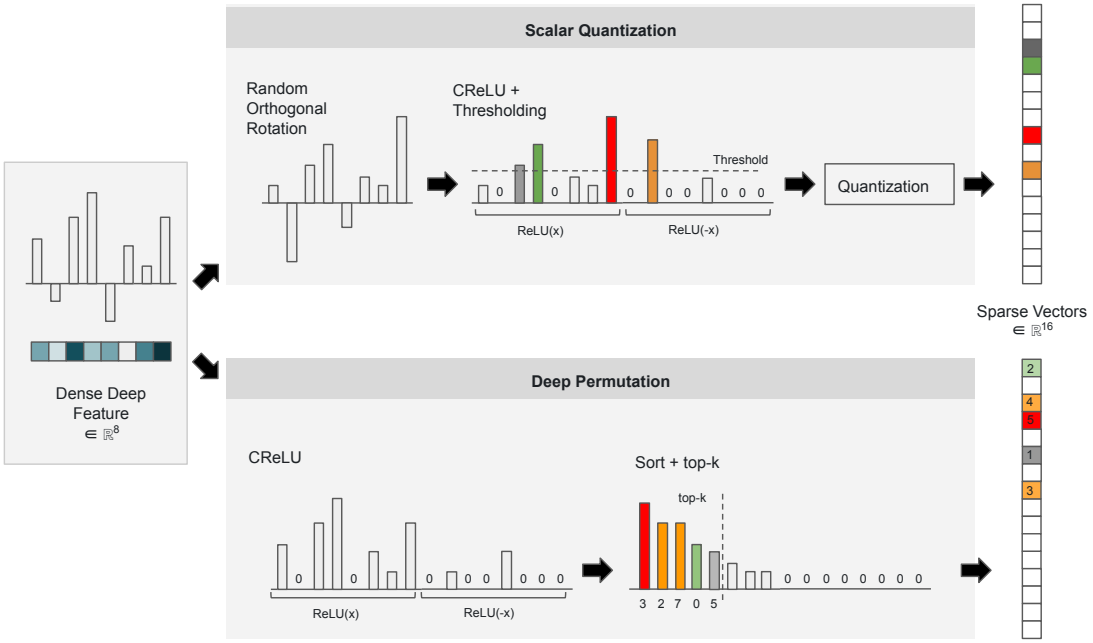


Figure 4.13: *Scalar quantization and deep permutation approaches for producing integer-valued vectors, on a toy 8-D input vector.*

dictionary. Both the scalar quantization and the deep permutation approaches try to obtain such representations from dense vectors of real numbers produced by DNNs.

Specifically, in the deep permutation approach every feature vector $\boldsymbol{v} \in \mathbb{R}^n$ is transformed by sorting the indexes of the elements of $\boldsymbol{v}$ in descending order with respect to the corresponding element values. In this way, we can construct the permutation $\Pi_{\boldsymbol{v}} = [\Pi_{\boldsymbol{v}}(1), \ldots, \Pi_{\boldsymbol{v}}(n)]$ of the feature vector $\boldsymbol{v}$ with respect indexes $\{1, ldots, n\}$ such that:

$$\forall i = 1, \ldots, n-1, \qquad \boldsymbol{v}(\Pi_{\boldsymbol{v}}(i)) \geq \boldsymbol{v}(\Pi_{\boldsymbol{v}}(i+1)), \tag{4.11}$$

where $\boldsymbol{v}(j)$ is the j-th element of $\boldsymbol{v}$. For example, if we have $\boldsymbol{v} = [0.2, 0.4, 0.1, 0.3, 0.6]$ the resulting permutation vector would be $\Pi_{\boldsymbol{v}} = [5, 2, 4, 1, 3]$.

On the other hand, the scalar quantization approach applies the following transformation to the original feature vector: $\boldsymbol{v} \rightarrow \lfloor s\boldsymbol{v} \rfloor$, where s is a scale factor and $\lfloor \cdot \rfloor$ is the floor operation.

Notice that these representations are as dense as the original feature vectors, while inverted indexes need sparse representations to be efficient. For this reason, we sparsify the vectors obtained with deep permutation and scalar quantization approaches by keeping only the first z higher values, while forcing all the others to be zero, as in [6]. A graphical representation for this two methods is shown in Figure 4.13. Notice that in the original formulation, the scalar quantization method keeps the vector components higher than a certain value $1/\gamma$; in this work, for an easier comparison as well as for a major control over the number of zeroed-out dimensions, we act as in the deep permutation by keeping the top-K active dimensions.

In both cases, we measure the similarity between the transformed vectors using the standard cosine similarity.

4.2.3 The Bag-of-Concepts Model

Although the research of indexing methods for sets of concepts deserves more attention, this novel method opens the doors to some interesting applications in large-scale cross-modal retrieval. In fact, a descriptor having each dimension carrying the grade of presence of each abstract concept would enable fine-grained control on the retrieval. It would be possible, for example, to weight differently the many words of the sentence to move attention on certain nouns or verbs, at query time. The Bag-of-Concepts (BoC) model is aimed at moving the first steps in this direction.

The TERN architecture provides us a set of concepts describing images and sentences. In the case of images, every salient region carries a concept, while for sentences the concepts are associated with single words. We are given a variable set of concepts $\mathcal{I}_i$ for every image i, and a variable sequence of concepts $\mathcal{S}_j$ for every sentence j. The key idea is to produce a codebook of concepts so that both images and sentences can be described as a set of codewords drawn from a common dictionary.

The codebook can be produced by collecting a large amount of visual and textual concepts from the training set and then performing clustering as in the standard Bag of Visual Words model. For this reason, we produce a large set of mixed visual and textual concepts:

$$\mathcal{C} = \bigcup_i \mathcal{I}_i \cup \bigcup_j \mathcal{S}_j \tag{4.12}$$

We downsample $\mathcal{C}$ so that $|\mathcal{C}| \simeq$ 100k concepts.

At this point, k-means is used to produce p clusters. The p centroids represent our codebook of concepts. Given that the word and the visual word spaces correspond, it is also possible to create a common codebook by using only the textual words from all the sentences. If we follow this methodology, we can consider the top p most common words appearing in all the $\mathcal{S}_j$ of the training set that are also present in the English dictionary and which are not stop-words.

Given this codebook, it is possible to produce the proper encoding for the images and sentences by describing them using the degree of presence of each concept from this dictionary. In the following discussion, we refer to the set of concepts $\mathcal{I}_i$ coming from images, although the same holds symmetrically for the set of concepts $\mathcal{S}_j$ from sentences. There are two possible paths for accomplishing this objective: the *hard* and the *soft* assignment methods.

Hard assignment If we use hard assignment, we can proceed as follows: given the set of concepts $\{\boldsymbol{v}^k\}_{k\in 1\ldots n}$ from an image[4], we can encode the image by finding, for every concept $\boldsymbol{v}^k$, the index of the nearest centroid (using L2 distance). Thus, in output, we obtain a set of k codewords $\tilde{\mathcal{I}} = \{\tilde{v}^k\}_{k\in 1\ldots n}$, where every element $\tilde{v}^k$ of this set is no more a 1024-D vector but an integer representing the index of the nearest centroid. The final representation for the image is obtained by computing the histogram $\boldsymbol{h}$ over the integer values contained in $\tilde{\mathcal{I}}$. The histogram $\boldsymbol{h}$ has p buckets and it is therefore a p-dimensional vector. It is already very sparse, with a maximum number n of non-zero elements, where $n \ll p$.

[4]Again, we consider an image for simplicity, but this procedure remains valid if we symmetrically consider sentences.

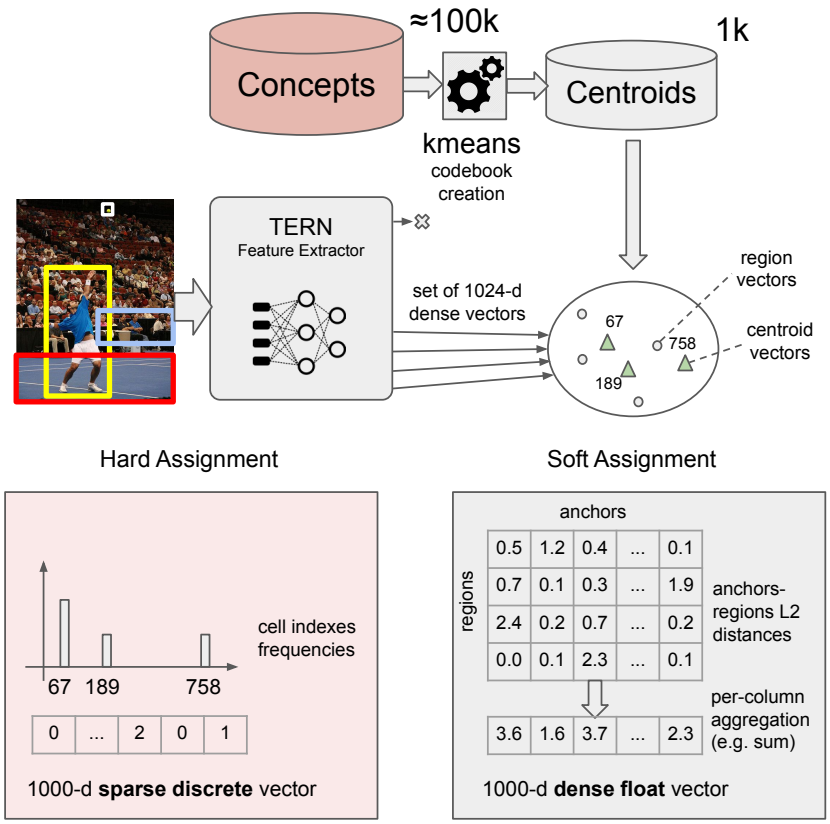


Figure 4.14: *Overview of the Bag of Concept (BoC) model, in the case of image concepts inference. Both soft and hard assignment versions of BoC obtain fixed-sized descriptions from sets of possibly contextualized regions extracted from the image. Notice that hard assignment already produces highly sparsified and discretized vectors.*

Soft assignment The hard assignment methodology performs heavy approximations on the original concept vectors, due to the discretization phase that transforms a 1024-D vector to a single discrete codeword. We use the soft assignment methodology to mitigate this problem. Given an image, soft assignment is performed by replacing the 1024-D vector for the k-th concept $\boldsymbol{v}^k$ with a p-dimensional vector of distances computed against all the p centroids. In this way, we preserve the information regarding all the distances between every concept and all the centroids. The result of this operation is a matrix $\boldsymbol{D}$ of L2 distances with shape $n \times p$. We convert L2 distances to similarities by applying the common transformation $\boldsymbol{S} = \frac{1}{1-\boldsymbol{D}}$. In order to produce a fixed-length p vector $\boldsymbol{a}$ describing the image we can aggregate the columns of $\boldsymbol{S}$, thus constructing $\boldsymbol{a}$ as $a^k = H_l(S_{lk})$, where $H_l(\cdot)$ is a symmetric aggregation function over the index l. An example of the overall inference procedure in the case of an image is reported in Figure 4.14. In our setup we experiment with both $\max$ and sum aggregation functions. By using sum we are summing together the similarity contributions for each image concept with respect to a certain centroid. If all the image concepts are near a given centroid the sum is very high, meaning that probably that particular concept is highly present in the image. Instead, if we employ $\max$, we gather information only from the nearest image concept for each given centroid. In this case, the maximum concept sim-

	MS-COCO (5k images, 25k sentences)						**Flickr30k** (10k images, 50k sentences)					
	Image Retrieval			Sentence Retrieval			Image Retrieval			Sentence Retrieval		
Model	K=1	K=5	K=10	K=1	K=5	K=10	K=1	K=5	K=10	K=1	K=5	K=10
Global Features												
TERN [163]	28.7	59.7	72.7	38.4	69.5	81.3	13.1	30.1	39.5	17.0	37.1	47.8
Deep Permutation	28.7	59.8	72.7	38.5	69.6	81.3	13.2	30.1	39.5	17.1	37.1	47.9
Scalar Quantization	28.7	59.8	72.7	38.4	69.6	81.3	13.1	30.1	39.5	17.0	37.1	48.0
Bag of Concepts - Hard Assignment												
W/o Context (WoC)	3.5	14.0	24.2	4.0	14.7	23.1	0.8	2.9	4.9	1.1	3.9	6.1
W Context (WC)	7.5	29.9	43.6	8.3	27.3	41.4	1.4	5.5	9.3	1.8	6.2	9.5
No stop-words WoC	3.4	13.6	23.6	4.0	14.2	22.3	0.8	2.9	4.9	1.1	3.6	5.8
No stop-words WC	6.7	27.5	42.0	6.9	25.0	39.0	1.3	5.1	8.8	1.6	5.8	9.1
No stop-words WoC (infer.)	3.1	12.8	21.8	3.7	12.9	20.9	0.8	2.7	4.5	0.9	3.0	5.0
English dict.	2.7	9.4	16.4	2.8	10.0	16.5	1.1	4.2	7.1	1.3	4.7	7.7
Bag of Concepts - Soft Assignment												
MAX-aggr WoC	25.4	54.2	67.4	32.0	63.6	76.2	10.1	24.1	32.6	13.2	30.3	40.5
SUM-aggr WoC	25.7	54.5	67.4	32.7	64.4	77.0	10.3	24.7	33.1	14.1	32.5	42.5
MAX-aggr WC	26.8	57.0	70.4	35.1	65.1	77.9	10.6	25.5	34.1	14.1	32.4	42.3
SUM-aggr WC	27.2	57.4	70.4	34.9	65.9	78.4	10.7	25.8	34.4	14.6	33.1	43.5

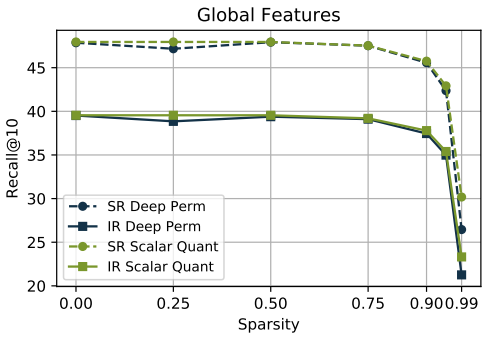
Table 4.5: *Recall@K metrics for our experiments on both MS-COCO and Flickr30k datasets*

ilarity to a given centroid is low only if all the concepts are far away from that specific centroid. The problem with soft-assignment is that the vector $\boldsymbol{a}$ is still dense and hence very inefficient from the point of view of an inverted index. We can think of sparsifying this vector by acting on the rows of $\boldsymbol{S}$ before computing the aggregation: for every row of $\boldsymbol{S}$, we keep only the z higher values, by setting to zero all the others. In the end, we use $\boldsymbol{h}$ and the sparsified $\boldsymbol{a}$ vectors as features from the hard and soft assignment Bag of Concepts model, and they are compared using the cosine similarity.

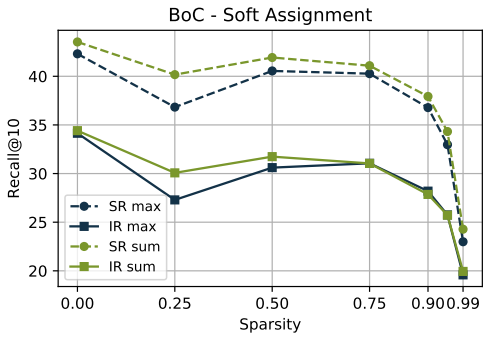
4.2.4 Experimental Setup

In our experiments, we evaluated the retrieval effectiveness of the produced features. We experimented both with the I-CLS and T-CLS global features (transformed using deep permutation or scalar quantization) and local contextualized set of features, processed using the Bag of Concepts model. We measured the retrieval effectiveness using the MS-COCO and Flickr30k datasets. For MS-COCO, we used the 5k test images from the split introduced by [109], together with the associated 25k sentences. Regarding Flickr30k, we sampled 10k images and the corresponding 50k sentences from the training set, given that Flickr30k was not used for the training procedure of the TERN feature extractor.

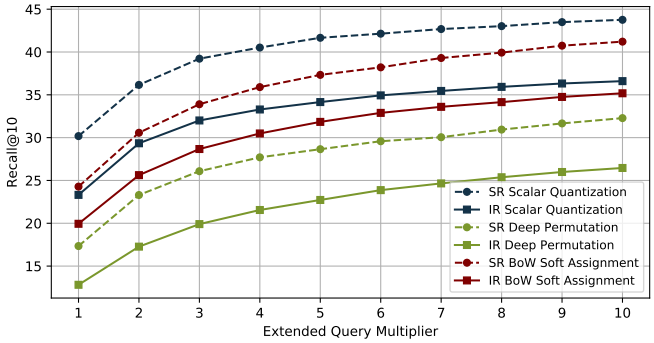
We used the Recall@K metric for evaluating the retrieval abilities of the collected features. The Recall@K is employed in many previous image-text retrieval works [65, 125, 163], and it measures the percentage of queries able to retrieve the correct item among the first k results.



(a) *Sparsity on Global Features*



(b) *Sparsity on BOW Soft Ass. Features*



(c) *Re-ranking effect with sparsity=0.99*

Figure 4.15: *(a) and (b): Trend of the Recall@10 metric while increasing the vector sparsity. Sparsity is reported as the fraction of vector components that are zeroed out ($\frac{z}{p}$ for BoC and $\frac{z}{2d}$ for global features, where $p = 1000$ and $d = 1024$ in our experiments). (c): Reranking results in terms of Recall@10 varying the extended query multiplier R_m. We search for the first $R_m \cdot 10$ items using the approximated method and then we rerank them using the original feature vectors. All the charts report the curves for both image retrieval (IR) and sentence retrieval (SR), and they are collected on the 10k images (50k sentences) from the Flickr30k dataset.*

Concerning deep permutations and scalar quantization approaches, we followed [6]: we first pre-processed the feature by applying a CReLU operation, which concatenates the vector to its opposite, and then we clipped all negative values to zero. In this way, we obtained a vector containing only positive or zero integer elements. For the scalar quantization, we used a scale of 1,000. Regarding the Bag of Concepts setup, we used a codebook of $p = 1{,}000$ elements, either by clustering or by using the most frequent 1,000 words from the training set present also in the English dictionary.

There are little variants of the Bag of Concepts that worth exploring. Remember that the TERN architecture produces contextualized concepts that vary among different scenarios. If we want to avoid the contextualization effect during the clustering phase, we can forward regions and words one at a time inside TERN, so that it is impossible for the network to discern the different contexts. The de-contextualized scenario produces region and word features more similar in spirit to the hand-crafted local features like SIFT, which were highly de-contextualized. Furthermore, it is possible to exclude the stop-words during the clustering and/or indexing phases. This is often performed to avoid the noise produced by not-so-informative sentence words. The results of these experiments are reported in Table 4.5.

4.2.5 Results

As it can be noticed from the first three rows from Table 4.5, the application of deep permutations and scalar quantization does not change the essence of the ranking, as far as the features are not sparsified. This confirms the order-preserving property of both quantization approaches in this context. Figure 4.15a and Figure 4.15b show how the features sparsification affects the overall performances on the 10k images from Flick30k. In particular, Figure 4.15a shows that the results begin to diverge from the original TERN performance only when a massive sparsification is applied. In that case, the scalar quantization method produces features which are more resilient to a strong sparsification, both during image- and sentence-retrieval.

Overall, the Bag of Concepts model can obtain very similar performances to the values reached by deep permutation and scalar quantization approaches, when soft-assignment is being used. Instead, the strong sparsification put in place by the hard assignment mechanism drastically lowers the overall effectiveness. Nevertheless, it can be noticed that we can always improve the overall Bag of Concepts performance by employing contextualized features. This is an important finding since it confirms that the context is a fundamental building block for understanding complex scenes: contextualized representations seem to have an important role even when considered as unordered sets without any structure, exactly like in the Bag of Concepts model. Also, it is worth noting that the exclusion of stop-words both during the clustering and the indexing phases lowers the overall performances, indicating that they probably have a non-negligible role in the pipeline.

When English words are used instead of the centroids from the k-means clusters, we obtain performances worse than the non-contextualized k-means case. This happens probably because these words natively lack context, as they are drawn from a dictionary and not computed from a representative training set. In fact, they could not be representative of the overall distribution of words in the MS-COCO dataset, although they are chosen among the 1,000 more frequent ones.

In Figure 4.15c we report the results after the re-ranking of the first $R_m \cdot K$ retrieved elements using the original global feature vectors in output directly from TERN, where R_m is the extended query multiplier. For example, to build the Recall@5 metrics when $R_m = 10$ we first retrieve the first $5 \times 10 = 50$ elements using the approximated method, and then we re-order them by computing the distances using the original feature vectors. The features used for the approximated search are sparsified with a sparsity factor of 0.99. The BoC with soft-assignment features cannot improve the results obtained by the deep permutation and scalar quantization ones, in case the sparsification is not applied (as shown in Table 4.5). However, Figure 4.15c demonstrates that in case the sparsification factor is very high (0.99) the BoC soft-assignment features are more resilient and can almost bridge the performance gap with the scalar quantization features when the reranking multiplier is progressively increased.

These results, in the end, show that the Bag of Concepts method has some interesting potential for efficient cross-modal retrieval, and further studies need to be performed in the direction of matching multi-modal elements as complex sets of concepts. This, in turn, would enable interesting functionalities at query time, such as the possibility to weight differently sentence words for having major control on the search results.

4.3 Semantic CBIR using Visual-Textual Features

The TERN visual features are learned by mapping them in a space where nearby images have similar textual descriptions. If the model learned the right abstraction level, the extracted visual features probably carry high-level semantics, and they are powerful enough to solve the so-called Semantic Content-based Image Retrieval (S-CBIR). S-CBIR perfectly recalls what in Chapter 3 we defined as Relational Content-based Image Retrieval (R-CBIR). From the point of view of this thesis, we can say that these two tasks are perfectly equivalent. They both deal with high-level semantic retrieval, and they consider not single instances or classes but a complex arrangement of entities interconnected by abstract relationships. Nevertheless, in this section, we explicitly refer to this image retrieval task using the *Semantic* prefix instead of the *Relational* one to align with some literature recently developed around this promising and challenging problem.

This section aims at closing the loop started in Chapter 3. We present some exciting achievements obtained on R-CBIR considering real-world images while using features learned using a cross-modal learning setup.

4.3.1 Instance vs Semantic Image Retrieval

Instance retrieval is the most studied in the recent literature. It consists of searching images that depict the same instance of an object — i.e., a person or a certain building. Thanks to this straightforward formulation, it is a well-defined task with clear ground-truths [95, 179, 180, 188], and it has been widely studied in literature [211, 175, 96, 94, 93, 15, 14, 74, 189, 199, 36, 219]. Instead, as stated in [22], semantic retrieval is a much more challenging task for many reasons. First of all, semantic retrieval is prone to subjective interpretations, as it aims to retrieve images belonging to the same *categories* as the query. Nevertheless, an image can simultaneously pertain to many categories, each one having a graded relevance. Furthermore, if we also consider the entity-entity relationships, the problem analogously becomes finding the most appropriate parameters to instantiate a fair scene-graph comparison. Because of this, another important problem is the lack of fair benchmarks to evaluate S-CBIR. In principle, we could consider an object-classification dataset and evaluate the retrieval effectiveness by employing precision and recall measures. Nevertheless, this would treat images as simple lists of keywords, not accounting for context, entity relationships, and, in the end, human taste. For this reason, some works [57, 21, 142] employed word taxonomies, such as WordNet [66], with metric learning approaches, to learn fine-grained semantics. Only recently, textual descriptions associated with images came into play. DEVISE [68] and HUSE [168] used pre-trained word embeddings to map visual features to label embeddings, evaluating their effectiveness on hierarchical datasets. More recently, many works for cross-modal retrieval processed images and natural language text for embedding them into a common space; nevertheless, they never evaluated their performance on CBIR, probably for the lack of a valid benchmark.

Only very recently, a novel very promising ground-truth, called Crisscrossed Captions (CxC) [172] from Google Research, has been released. It provides image similarities by computing and refining similarities among the corresponding natural language captions, offering high-level similarity scores suitable for S-CBIR. Therefore, in this

Method	mAP
Deep R-MAC	0.282
Listwise	0.285
GeM	0.287
TERN	0.303
MAC	0.311
R-MAC	0.323

(a) *Results on MIRFlickr25k. Methods are ordered by increasing mAP.*

Method	R@1	R@5	R@10	medr	NDCG
GeM	14.1	34.8	45.7	13	0.358
R-MAC	20.4	49.1	63.7	6	0.385
VSE++	38.8	74.2	85.2	2	0.518
VSRN	**47.3**	**81.3**	**90.3**	2	0.557
TERN	43.6	80.3	89.7	2	**0.560**

(b) *Results on MS-COCO, using Crisscrossed Captions annotations.*

Table 4.6: *Semantic CBIR results on MIRFlickr25k and CxC datasets.*

section, we aim at quantitatively evaluating the TERN features and other visual-textual models against this novel and promising benchmark.

4.3.2 Experimental Setup

Datasets and Evaluation Metrics In order to perform a comprehensive comparison, we evaluate the TERN visual features on two different benchmarks. The first one is the dataset used in [22], which proposed to use MIRFlickr25k [92] and its labels. Although this dataset comprises many categories, the evaluation is performed only on label matching, and therefore in a binary relevance fashion (mAP is used). Furthermore, for simplicity, only single-label images are used as queries. This evaluation scheme is very restrictive for the high-level TERN features and can easily hide their actual potential. For this reason, we also use the recently released Crisscrossed Captions (CxC) [172] annotations for the MS-COCO dataset. The Crisscrossed Captions (CxC) dataset, already introduced in Section 2.5, extends MS-COCO with semantic similarity ratings for image-text, text-text, and image-image pairs. The rating criteria are indeed very similar to the semantic evaluation performed in Section 4.1, except that they mixed Universal Sentence Encoder(USE) [42] and Bag-of-Words (BoW) with Glove embeddings [173] for obtaining sentence similarities. For image-to-image retrieval — that they call Semantic Image Similarity (SIS) — they use the Recall@k metric. They set the positive threshold on the relevance score to 2.5, defined empirically by observing the distribution of the scores. Given the availability of continuous relevance scores between pairs of images in the CxC annotations, we additionally avoid reducing the evaluation to a binary retrieval problem by using the NDCG metric, as already done in Section 4.1.

Features Extraction We used the visual pipeline of the TERN architecture for extracting the visual features. Concerning the baseline methods we used in the comparison, we extracted the features using the code and the pre-trained models already available online. All the features are compared using the cosine similarity.

4.3.3 Results

We can notice how the text-driven methods (TERN, VSRN) perform similarly to other instance retrieval methods (like GeM or R-MAC) on the MIRFlickr25k dataset in Table 4.6a. This is mostly due to the keyword-based image annotation of this dataset,

which gives too coarse information about the semantics contained in the images. Although this is a coarse-grained semantic dataset, TERN can still perform better than other recent architectures specifically designed for image retrieval.

The results are visibly different when using the novel Crisscrossed Captions annotations for MS-COCO (Table 4.6a). The features developed for solving the instance retrieval task here cannot handle the high-level semantics. In particular, the recent GeM features perform worse than R-MAC features on this benchmark, confirming the claim by [22], which states that more recent image retrieval methods — maybe focusing on local pattern matching for maximizing the effectiveness on instance retrieval — often perform worse in semantic retrieval. The best results are obtained by recently developed cross-modal matching models, that ground natural language texts and visual data into the same common space. In particular, VSRN and TERN outperform VSE++, with TERN obtaining better results on the NDCG metric.

These insights suggest that a significant effort is needed to fill the semantic gap in image retrieval. The most critical challenge remains the design of fair benchmarks to evaluate the semantic retrieval abilities of modern architectures. Nevertheless, TERN performs reasonably well on the novel Crisscrossed Captions annotations; the quantitative results align with the qualitative outcomes measured on the V3C1 dataset, employed in the next section to perform large-scale video retrieval. For this reason, we redirect the reader to Section 4.4.4 for some S-CBIR qualitative results.

4.4 Application to Large-scale Video Retrieval

All the previous studies on TERN features effectiveness and resilience to sparsification have been recently deployed on a recently-developed tool for large-scale video retrieval, called VISIONE.

4.4.1 VISIONE and Video Browser Showdown (VBS)

VISIONE [5] is a tool for large-scale video search developed at the AIMH laboratory, at the ISTI-CNR in Pisa, for participating in the Video Browser Showdown (VBS) annual competition. VBS [49, 140] is an international video search competition whose aim is to evaluate the performance of interactive video retrievals systems on the following tasks:

- *Known item search (KIS)*, where the objective is to retrieve an already seen videoclip;
- *Textual KIS*, a variation of the KIS task where the target videoclip is not shown to the participants; instead, only a very detailed natural language description of it is available;
- *Ah-hoc Video Search (AVS)*, where a very generic textual sentence is shown to participants — for example, *"A person walking near his dog on the sidewalk"* — and the objective is to find as many matching videoclips as possible.

The challenge is powered by the V3C1 dataset [32], a collection of around 1000 hours of videos in the wild.

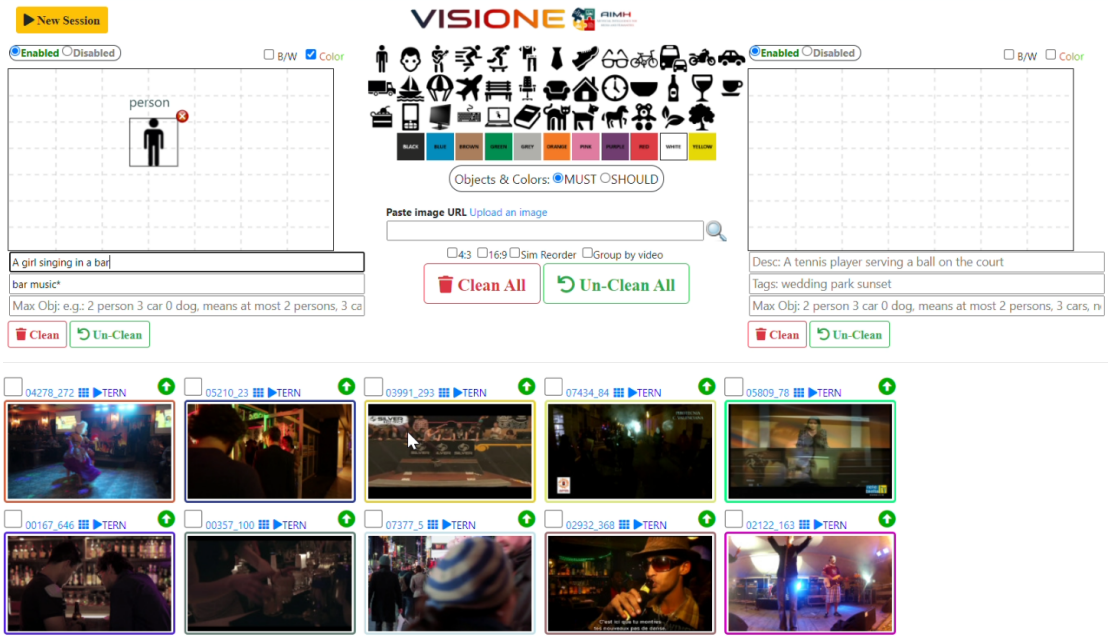


Figure 4.16: *VISIONE main interface.*

The main interface of VISIONE is shown in Figure 4.16. The first version of the tool, developed in 2019, was designed to face the aforementioned tasks using a mix of off-the-shelf and novel techniques for large-scale content-based retrieval. More in details, VISIONE is keyframe-based: it extracts relevant information from the 1M keyframes automatically collected by the VBS organizers; then, a mix of state-of-the-art techniques from image analysis and understanding are used to automatically extract useful information from these keyframes (e.g., R-MAC features [219], color features, bounding boxes and respective labels using the YOLO [195] object detector). All these different-level vectorial descriptions of the various keyframes are indexed in Lucene[5], a text-based indexing engine, using the Surrogate Text Representation (STR) [6, 4], together with the scalar quantization approach, both already described in Section 4.2.

VISIONE 2019 dealt with AVS and textual KIS tasks by querying the system with the keywords extracted from state-of-the-art object detectors. However, this is a more BoW-like model of proceeding: relationships between words is completely lost, and the spatial and abstract interactions between actors and objects cannot be explicitly specified in the textual query. VISIONE 2019 partially solved this shortage by introducing a novel canvas-based search tool that enabled positional queries and an interface for constraining the number of instances for each concept.

The novel VISIONE 2021 [8] integrates the TERN visual-textual features, turning the keyframe retrieval using natural language sentences into a straightforward task. In particular, we use the TERN features regularized with the TERAN loss, which achieved better Recall@K values (Table 4.1). Being real-valued features like the already-in-use R-MAC, the TERN features can be seamlessly integrated into the 2019 system without any substantial engineering effort. R-MAC features are only used for internal queries (i.e., visual similarities are computed only between V3C1 keyframes). Therefore they

[5] `https://lucene.apache.org/core/`

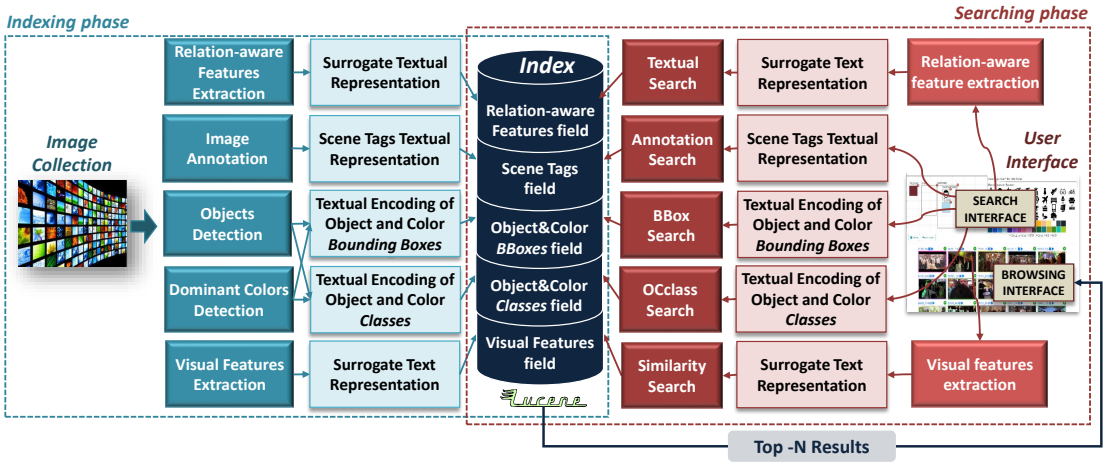


Figure 4.17: *The VISIONE 2021 architecture.*

should not be extracted on the fly from a never-seen external image. Instead, the textual query is necessarily external since the user inputs a never seen text at query time. For this reason, the text features must be extracted from the textual pipeline of the TERN architecture during the online phase. The inference of the textual path is generally very fast, requiring only a few milliseconds even without GPU acceleration. The resulting overall architecture is shown in Figure 4.17.

4.4.2 Fine-tuning Scalar Quantization on Textual KIS and AVS

In order to be compliant with the already existing text-based indexing, we needed to quantize and sparsify the real-valued vectors. Following the results in Section 4.2, we used the scalar quantization approach to perform such transformation, as it was already done for the R-MAC features. In a large-scale scenario, it is preferable to accurately tune the parameters of the scalar quantization to obtain the best index performance while maintaining good retrieval effectiveness. In the scalar quantization method, the main parameter is the threshold $\frac{1}{\gamma}$, which directly controls the sparsity of the features and, therefore, the number of posting lists accessed at query time. For the scale factor which controls the quantization, we used $s = 100$. We empirically observed that the scale factor has a limited impact on the final rankings, so we decided to omit it from the parameter search.

Initially, we considered some of the textual queries extracted from some preliminary logs of VISIONE. We observed that the sentences formulated by VISIONE users are indeed very similar to the textual descriptions present in the MS-COCO and Flickr30k datasets. Therefore, they seem to have a distribution similar to the ones that TERN can understand (more on this in Section 4.4.3). Specifically, after a preliminary data cleaning step, we obtained 384 natural language sentences from the logs to be used as queries. We then measured the retrieval effectiveness of the scalar-quantized features with respect to the original TERN ones varying γ. The retrieval effectiveness is measured using Recall@K, as we are interested in understanding the intersection between the original TERN rankings and the ones obtained after applying scalar quantization. We used $K = 500$, as the user usually ignores deeper results.

To run these experiments, we encoded the features using the Surrogate Text Repre-

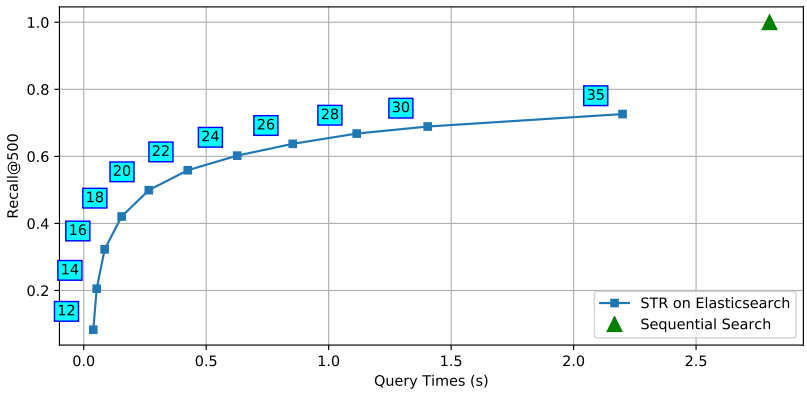


Figure 4.18: *Recall@500 vs. query times. The numbers in cyan above each point denote the factor γ used for sparsifying the features.*

sentation, and we employed the Elasticsearch[6] textual search engine to index them. The results are shown in Figure 4.18. The numbers above each point in the chart represent the scalar quantization thresholds used for each experiment run. As we can notice, the query time increases non-linearly with respect to the factor γ. For $\gamma > 30$, the query times easily become unworkable in real interactive search scenarios, and for even higher thresholds, the search times become greater than the sequential search ones. This is because textual indexes are overloaded when most lists have to be accessed. For these reasons, we were mainly interested in the half-plane of the graph satisfying $\gamma < 28$, where query times are below 1s and recall values near 70%.

To better fix the threshold, we further considered the textual KIS task. The textual queries, in this case, are more complex and extremely more detailed than those in the AVS task. They are built of three separate sentences, overall describing the whole target shot. An example:

> *A slow pan up from a canyon, static shots of a bridge and redrock mountain.*
> *A river is visible at the ground of the canyon.*
> *The bridge is a steel bridge, there is a road right to the mountain in the last shot.*

We know in advance that TERN cannot handle temporal information, therefore querying the system with the whole 3-sentences description can bring misleading results. Furthermore, in a real use case, the user very likely enters only one or two sentences from the original full-length description, probably choosing among the more discriminative ones. We tried to emulate this real-world scenario by feeding the system with one, two, or three sentences to measure the resulting system effectiveness under different scalar quantization thresholds. In particular, we had at our disposal eight different AVS queries from VBS 2019, with the relative ground-truth — i.e., the list of relevant keyframes from the target shot. Given that the task is solved if we can find at least one of the keyframes, we measured the position in the ranked list where the first valid keyframe is found; in other words, we searched for the minimum rank for each query. In Figure 4.19, we reported the minimum ranks averaged among all the queries.

[6] https://www.elastic.co/elasticsearch/

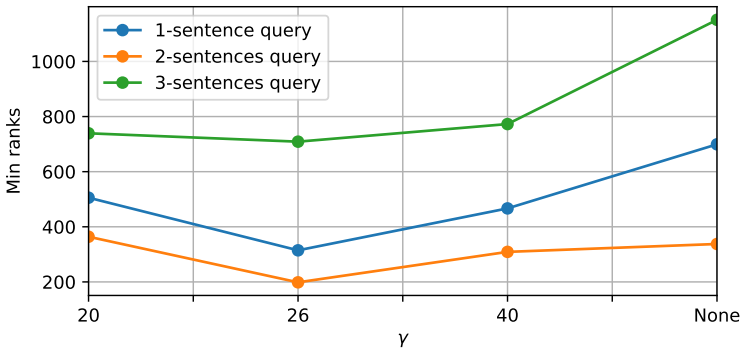


Figure 4.19: *Minimum ranks when querying the system with one, two, or three of the sentences from the original shot description. None in the x-axis refers to the case of disabled scalar quantization (retrieval is performed using the original vectors).*

Notably, we can notice that the curves reach a minimum when the factor $\gamma = 26$. This value corresponds to a good tradeoff position in the effectiveness-efficiency curve in Figure 4.18. Therefore, this is the final value that we chose as a threshold for deploying the text-to-image module in VISIONE. Interestingly, the scalar quantization approach also seems to have a positive effect on the retrieval effectiveness since it can surprisingly reach better results than original TERN features ($\gamma = None$). This highly desirable effect can be caused by the implicit noise removal performed by zeroing out the less relevant vector dimensions. Furthermore, Figure 4.19 suggests that the system is better on average when feeding two out of the three sentences from the original full-length description. This result validates our hypothesis that the full textual description is too rich to be correctly embedded into the current TERN feature. On the other hand, using only one of the sentences can be too simplistic.

4.4.3 Features Analysis

In this section, we briefly inspect the features learned by TERN when the inference is performed on the data from the VBS challenge. As in the previous performance analysis, we used 384 textual captions extracted from the VISIONE logs, together with a subset of the visual features extracted from the keyframes of V3C1.

In Figure 4.20, we reported the PCA[7] data visualization of both visual and textual TERN features. Specifically, images are from V3C1 keyframes, while captions come either from MS-COCO or the VISIONE logs. As we can notice, many of the MS-COCO captions lie out of the V3C1 keyframes distribution, as many of them specifically describe a target MS-COCO image. This evidences the sensitivity of the learned feature space to the specific application context. On the other side, the textual queries from the VISIONE logs are more in-distribution with respect to the searched V3C1 images, probably because they are specific to a V3C1 keyframe (during KIS searches) or general enough to lie in the middle of the distribution (AVS searches).

Similar conclusions can be drawn by visualizing the image-text and image-image distance distributions (Figure 4.21). In this case, we considered only the captions from

[7]We did not report the T-SNE visualization, as it seems to capture wrong data distributions for the VISIONE queries (most likely due to their small number with respect to the number of keyframes).

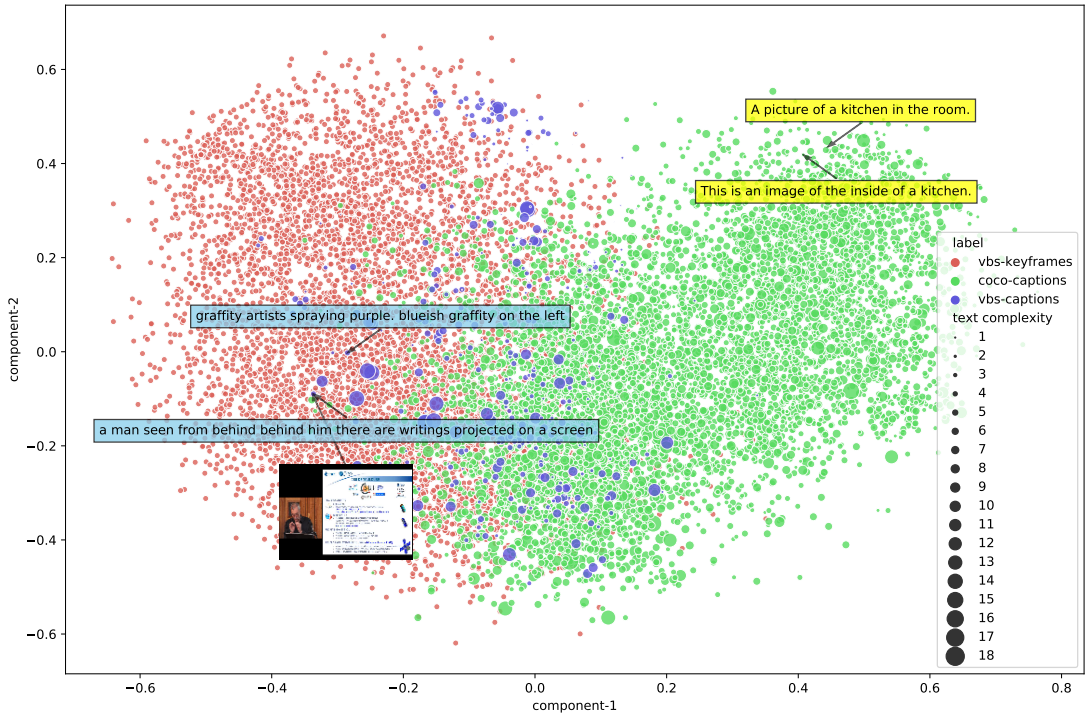


Figure 4.20: *PCA of the TERN visual and textual features from VBS and MS-COCO data. The marker size is related to the text complexity, measured as the number of non-stopwords occurring in the phrase (the marker size carries no information for the keyframe features).*

the VISIONE logs. We can notice how the two distributions overlap, confirming that a significant portion of textual features is embedded into the distribution defined by the visual features. Furthermore, the fact that the image-image distance distribution is entangled with the text-image one gives an intuitive explanation of why the Semantic Content-based Image Retrieval (S-CBIR) introduced in Section 4.3 works nicely with the visual features extracted from the TERN architecture.

4.4.4 Qualitative Results

In this section, we issue some targeted queries to the VISIONE browsing system using the TERN features processed as explained earlier, and we briefly comment on the outcomes.

As a first test, we probe the system with four targeted textual sentences, reporting the top-5 results. The queries and the respective results are reported in Figure 4.22a. We can immediately notice that the queries are very similar considering the involved actors — there is always one or more *dogs* and one or more *persons*. Nevertheless, the overall formulation is very different in each query. Notably, we vary the *verb* between the two actors to force very specific relationships between them; furthermore, we also vary the number of actor instances to challenge the system with counting skills, crucial in relational scenarios as shown in Chapter 3. Looking at the results, we can notice how well the system can cope with all four different shades. In particular, the difference between the first and the second query is the substitution of the verb caress with stand.

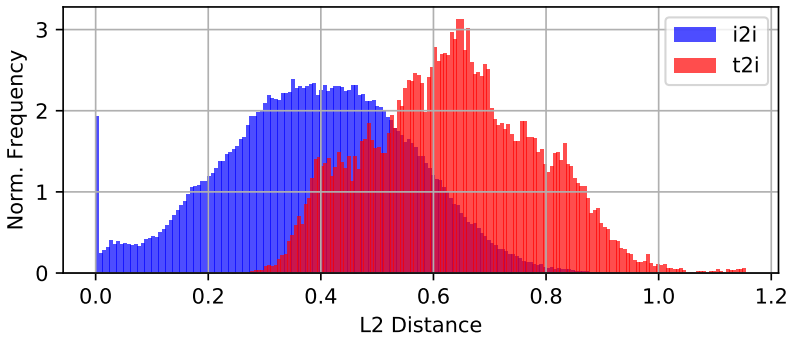


Figure 4.21: *Distance distributions for image-image and text-image using TERN features on the VBS data.*

Notice that there is a common image between these two queries; notably, it shows a man with a woman in the background who *caresses* the dog while *standing*. Similarly, the 3-rd query correctly captures the *many dogs*, while the fourth associates the *running dog in the field with a man* to a dog training session, which is the place where it is most likely to find these actors in this configuration.

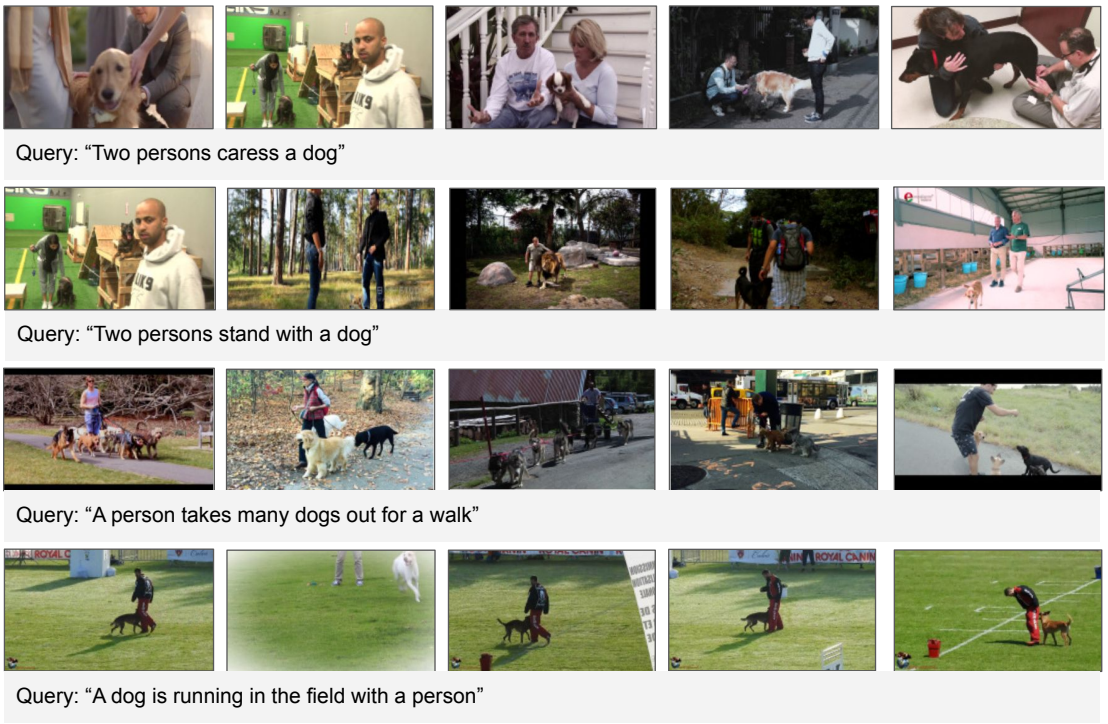
Secondly, we evaluate the system by looking at the results for Semantic Content-based Image Retrieval using the TERN features. As mentioned in Section 4.3, where we reported some quantitative results for the semantic image similarity, the TERN features are much more semantically expressive than low-level features like the R-MAC or the newer GeM features, mainly forged for solving the *instance* retrieval task. Using the same scalar quantized TERN and GeM features (both already indexed in VISIONE), in Figure 4.22b, we show an example of image retrieval using these features. Given that GeMs are best suited for instance retrieval, they can most likely retrieve low-level matching images, like images with persons or with similar patterns in the background. Differently, the TERN descriptors can embed very high-level information, correctly capturing the concept of *dancing people*.

4.5 Persuasion Detection in Memes using Multi-modal Transformers

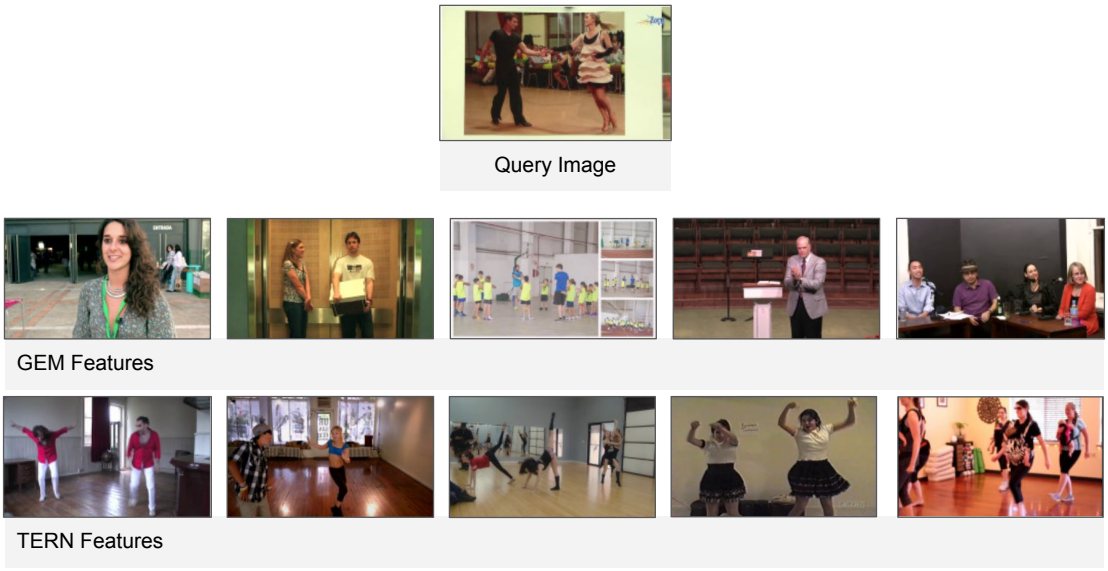
Although the previous sections were oriented towards effective and efficient retrieval, in the last section of this chapter we briefly discuss another important critical real-world use case where multi-modal Transformers could have a key role. In particular, the task we discuss is the detection of persuasion techniques in social network images and, more specifically, in *memes*. Notably, this work describes the system with which we participated to the *SemEval 2021 Task 6* challenge [8] [59], kept in conjunction with the *International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*.

Social networks play a critical role in our society. Nowadays, most of the ideas, thoughts, and political beliefs are shared through the internet using social platforms like Twitter, Facebook, or Instagram. Although these online services enable information to be spread efficiently and effectively, it is non-trivial to understand if the shared contents are free of subtle meanings altering people's judgments.

[8] https://propaganda.math.unipd.it/semeval2021task6/index.html



(a) *Text-to-Image Retrieval.*



(b) *Image-to-Image Retrieval.*

Figure 4.22: *Qualitative results from (a) four different textual queries and (b) an image query. Only the top-5 images are shown, for ease of viewing.*

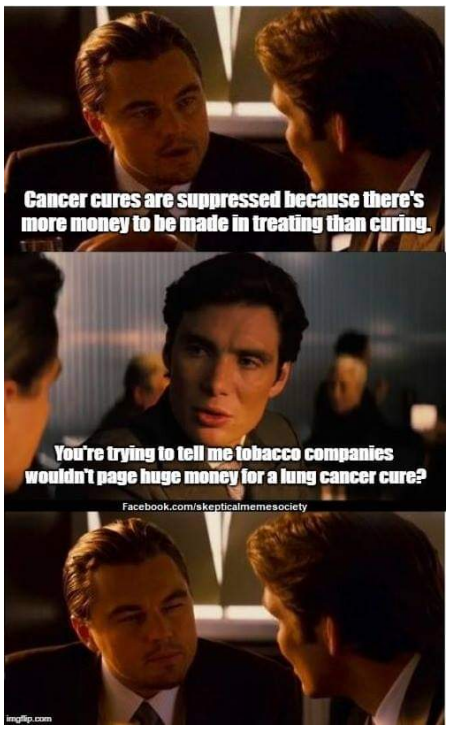
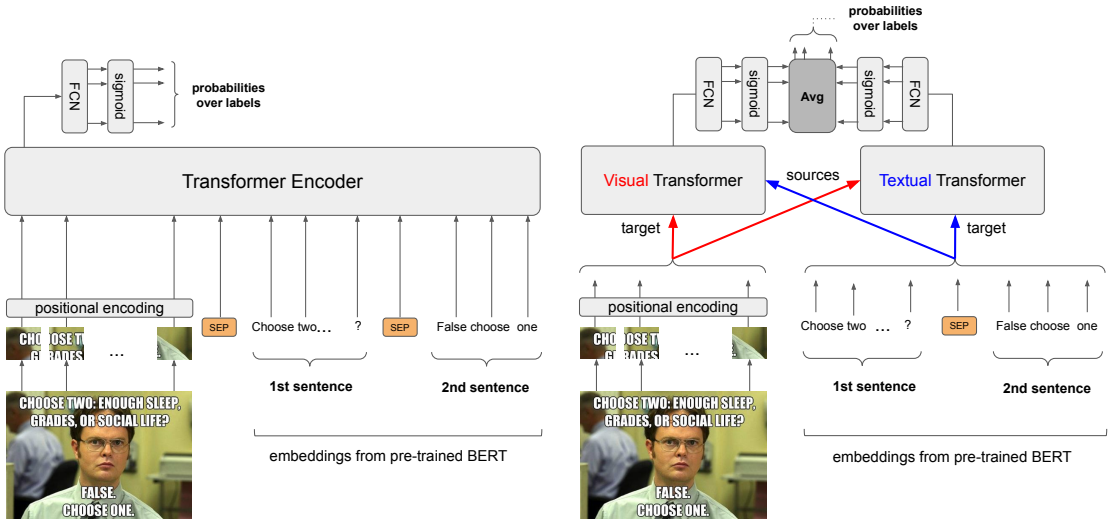


Figure 4.23: *An example of meme with evident causal oversimplification, loaded language to catch more attention, and built on top of dialogues from a very popular film with famous actors to acquire even more strength. The added dialogues, however, perfectly fit the facial expressions of the actors and the overall context of the scene.*

Among all the types of content living in a social network, memes acquire a significant role. Memes are small yet effective units of information able to spread cultural ideas, symbols, or practices and usually exist under the form of pictures, possibly with overlayed text. An example is shown in Figure 4.23. Memes are created so that they can propagate rapidly and reach a large number of users; for this reason, they are one of the most popular types of content used in an online disinformation campaign, influencing the users through several rhetorical and psychological techniques, such as causal oversimplification, name-calling, or smear. The automatic detection of these memes and the disinformation techniques they are possibly employing is a challenging yet crucial task for the proper management of a social network. All the milestones reached in the last few years in automatic content extraction and reasoning from multimedia data acquire a fundamental role in large-scale analysis of social networks.

In this work, we tackle the problem of recognizing which kind of disinformation technique is used to forge memes for a disinformation campaign. In particular, we propose an architecture based on the Transformer architecture model [220] for processing both the textual and visual inputs from the meme. This architecture, which we call DVTT (Double Visual Textual Transformer), comprises two full Transformer networks working respectively on images and texts; however, each of these Transformers is conditioned on the other modality. We consider this task as a multi-label classification problem, where text and/or images from the meme are processed, and probabilities of presence of each possible persuasion technique are returned as a result.



(a) *VTTE* **(b)** *DVTT*

Figure 4.24: *The proposed architectures: (a) the Visual-Textual Transformer Encoder model (VTTE) used as a baseline, and (b) the novel Double Visual Textual Transformer model (DVTT). The shown meme is taken from* `https://engineermemes.blogspot.com`*, and it is licensed under the Creative Common license.*

We tackle subtasks 1 and 3 of the *SemEval 2021 Task 6* challenge [59]. Subtask 1 consists of identifying which of 20 possible persuasion techniques are used in it given only the textual content; subtask 3 is very similar to subtask 1, but both textual and visual contents of the meme are used, and there are 22 possible persuasion techniques. Our proposed models could reach the 5th position for subtask 1 and the 4th position for subtask 3 on the publicly available leaderboard. The code for replicating our results is available on GitHub[9].

4.5.1 Double Visual-Textual Transformer

Mainstream multi-modal Transformers employ the Transformer Encoder (TE) to jointly process image patches and words [145, 184, 215]. Inspired by these models, we initially defined a baseline model which jointly feeds image patches from the meme and the words from the text to a TE. An overview of this approach is presented in Figure 4.24a. We refer to this baseline as *VTTE* (Visual-Textual Transformer Encoder).

In this work, instead, our main proposal consists of an architecture that can exploit the full Transformer architecture to jointly reason on visual and texts and producing label probabilities as output. We call this model *DVTT* (Double Visual Textual Transformer). DVTT is composed of two different Transformer networks able to process visual and textual inputs concurrently. The important aspect of DVTT is that each Transformer is conditioned on the other modality so that it is possible for the whole architecture to jointly reason on the two modalities following two different paths: in the first, the text is the key aspect, and images integrate the reasoning performed on the text; conversely, in the second, the images are the primary modality and the text intervene to enrich the visual features. The DVTT architecture is shown in Figure 4.24b.

[9] `https://github.com/mesnico/MemePersuasionDetection`

For each of the two Transformers, the final head is a multi-classification head constructed on the first token of the output sequence. In particular, a linear layer outputs the logits over each possible persuasion technique, and the final softmax operator converts logits into probabilities, exactly like in the VTTE baseline model. The two Transformers outputs are merged by averaging the inferred probabilities for each possible label, and the overall network is trained end-to-end with a binary cross-entropy loss.

4.5.2 Experimental Setup

We used the data provided by the *SemEval 2021 Task 6* challenge organizers to train and validate our model. Although we mainly concentrated on subtask 3 (images + texts), we also tackled subtask 1, which is essentially equivalent to subtask 3, except that only the text is available.

Dataset The provided dataset comprises 687 memes for training, 63 memes for validating on the so-called development set, and 200 memes for the final testing. All the memes carry textual captions written in English. Notice that, in the end, we were allowed to use the annotations for the development set, so we had at our disposal a total of 750 annotated memes to use for the training and validation phases. The annotations consist of a list of persuasion techniques for every meme. In subtask 1 there are 20 possible persuasion techniques and 22 in subtask 3.

Metrics The official metrics for computing the model performance are the Micro-F_1 and Macro-F_1 scores; The F_1-score is defined as the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}. \tag{4.13}$$

The F_1-score gives values in the interval [0, 1], hence it is often a good way of summarizing the performance of binary classifiers. More details on this metric are given in Section 2.4.2. The difference between Micro-F_1 and Macro-F_1 scores lies in the way *precision* and *recall* are computed: in Micro-F_1, they are computed from all the true positives, false positives, and false negatives over all the labels; for this reason, Micro-F_1 gives each sample the same weight, thus giving more emphasis to the most frequent labels. On the other hand, Macro-F_1 is computed as the mean value among the F_1-scores computed on the different labels: Macro-$F_1 = \frac{1}{N}\sum_1^N F_1^i$, where N is the number of labels and F_1^i is the F_1-score computed among the samples having label i. In this case, all the classes contribute equally regardless of how often they appear in the dataset.

Model Setup For subtask 3, we used the proposed DVTT model (Figure 4.24b). We used a learning rate of $5 \cdot 10^{-5}$ and a batch size of 8. We trained the models for 40 epochs in all the experiments, decreasing the learning rate after 30 epochs to $5 \cdot 10^{-6}$. The Transformer is composed of 4 encoder layers and 4 decoder layers, with 1024-dimensional feed-forward networks for producing queries, keys, and values. As a baseline for subtask 3, we used the VTTE architecture (shown in Figure 4.24a), composed of a 4-layer Transformer encoder module, with a multi-label classification head on top, exactly like the one in DVTT. For subtask 1, instead, we used the VTTE architecture

(Figure 4.24a) with the same setup used for the subtask 3 baseline, except that the visual input is not fed to the network.

Features Extraction For all the conducted experiments, we obtained suitable visual and textual features from pre-trained state-of-the-art networks. Concerning images, we re-scaled them to 256×256, and we took a 224×224 crop (a random crop during training and a center crop during inference). We also normalized the images using the pixels mean and standard deviation computed on the whole dataset. In order to input an image to the Transformer, we had to encode it as a set of features. We used a ResNet50 pre-trained on image classification, as it is characterized by a good performance at low computational costs compared to deeper backbones; we down-sampled the features maps from the last convolutional layer to a 7×7 spatial grid of 2048-D features. The resulting flattened 49 visual features were then augmented with their spatial positions by appending the normalized coordinates of the chunk to the 2048-D visual feature. Another possibility consisted of using visual features extracted from state-of-the-art object detectors, like Faster-RCNN. However, images carried in memes are not homogeneous: they show possible stacked scenes and overlayed text, making it very difficult for an object detector to identify the most critical regions. Concerning text processing, we used a pre-trained BERT model [58] for extracting word embeddings. BERT embeddings are trained on some generic language processing tasks such as sentence prediction or sentence classification and demonstrated state-of-the-art results in many downstream tasks. Every meme can carry one or more sentences, encoded in the same string and separated by "\n\n". For this reason, during the string tokenization phase, we simply replaced "\n\n" with the SEP token. In the basic DVTT model, we trained only the Transformer models, leaving the feature extractor fixed. In the Experiments section, we also report the results from a fine-tuning of the feature extractors.

Validation The test-set annotations were hidden to the participants, so the model should be validated using a split of the available annotated data. Given that the available annotated memes are relatively few, we validated our model using cross-validation. In particular, we split the training data into six different folds, training six different models by using five out of six data folds and validating them using the remaining fold. We selected the model having the best sum of Micro-F_1 and Macro-F_1 scores on the validation fold. All the performance measures reported in the Results section are an average of the metrics from this 6-fold validation procedure. For participating in the final competition on the test set, we prepared an ensemble model composed of all the six trained models, and we produced the final probabilities by soft-voting. We used a final binary-classification threshold of 0.3 over the label probabilities.

4.5.3 Results

Concerning subtask 3, we studied the performance of our DVTT model by comparing the F_1-scores against the VTTE baseline; furthermore, we tried also to train the model using a balanced sampling of the labels and to fine-tune the feature extractors (BERT and the ResNet-50), using a learning rate of $1/10$ with respect to the one used for training the Transformer models. Using a lower learning rate during the fine-tuning process is a common procedure to avoid model overfitting. We also report the results of

Model	Macro-F_1	Micro-F_1
VTTE (Baseline)	0.327	0.596
DVTT	0.336	**0.601**
DVTT - Balanced	0.300	0.489
DVTT - Finetuned	**0.341**	0.592
DVTT - 2 layers	0.310	0.596
DVTT - 6 layers	0.325	0.583

(a) *Results for subtask 3.*

Model	Macro-F_1	Micro-F_1
VTTE	0.372	0.566
VTTE - Balanced	0.361	0.490
VTTE - Finetuned	0.386	**0.581**
VTTE - 2 layers	0.365	0.565
VTTE - 6 layers	**0.389**	0.569

(b) *Results for subtask 1.*

Table 4.7: *Ablation results, reported on the validation set.*

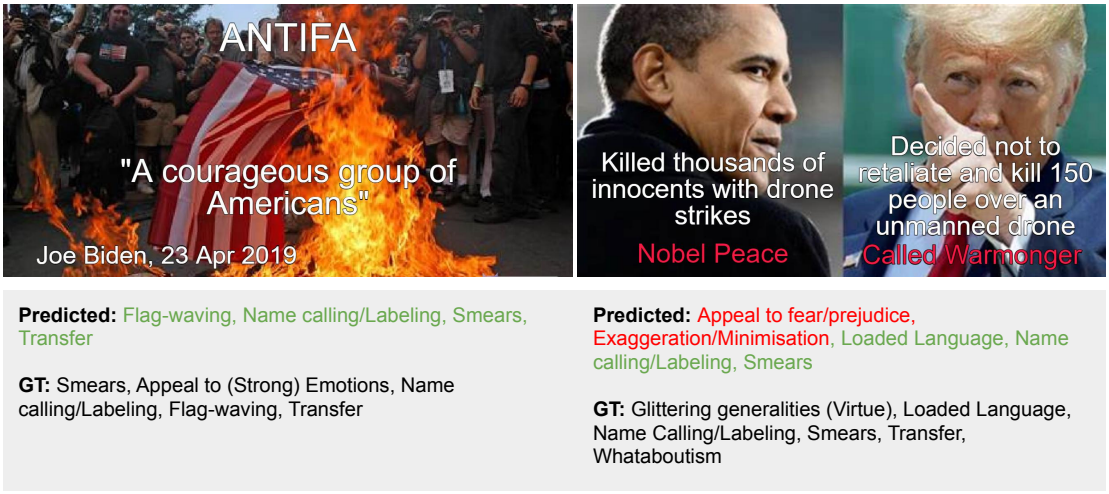


Figure 4.25: *Example of predictions from the DVTT model for subtask 3. In green, the true positives labels; in red, the false positives labels. Images obey to the Creative Common license and they are searched on the Bing image-search engine using "free to modify, share and use" license filtering.*

slightly different variants of the DVTT model obtained by increasing and decreasing the number of the Transformer's encoder / decoder layers: the base architecture contains four layers; we also experimented with two and six. The results of these experiments are reported in Table 4.7a.

For subtask 1, instead, we used the VTTE model without visual input, trying out the same experiments performed for subtask 3. In this case, when varying the number of layers, we only considered the Transformer Encoder ones (there is no decoder in the VTTE model). The ablation results on subtask 1 are reported in Table Table 4.7b.

Discussion Looking at the subtask 3 results in Table 4.7a, we can notice that the proposed DVTT model can achieve slightly better results than the VTTE baseline. In particular, the DVTT with fine-tuned BERT and ResNet50 modules achieve the best results on the Macro-F_1 metric. Also, the choice of using four encoder and decoder layers seems to lead to the best compromise on both the metrics. Concerning the results of subtask 1 in Table 4.7b, fine-tuning the BERT model is even in this case a good

Rank	Team	Macro-F_1	Micro-F_1
1	MinD	.290	.593
2	Alpha	.262	.572
3	Volta	.266	.570
4	mmm	.303	.548
5	**AIMH**	**.245**	**.539**
6	LeCun	.227	.512
7	WVOQ	.227	.511
8	TeamUNCC	.236	.510
9	NLyticsFKIE	.140	.498
10	TeiAS	.187	.497
11	DAJUST	.187	.497
12	YNUHPCC	.263	.493
13	CSECUDSG	.185	.489
14	TeamFPAI	.115	.406
15	NLPIITR	.126	.379
	Maj. Baseline	.033	.374
16	TriHeadAttention	.024	.184
	Rand. Baseline	.044	.064

(a) *Results on subtask 1.*

Rank	Team	Macro-F_1	Micro-F_1
1	Alpha	.262	.581
2	MinD	.244	.566
3	1213Li	.228	.549
4	**AIMH**	**.207**	**.540**
5	Volta	.189	.521
6	CSECUDSG	.121	.513
7	aircasMM	.200	.511
8	LIIR	.188	.498
9	CAU731NLP	.084	.481
10	WVOQ	.240	.478
11	YNUHPCC	.096	.446
12	TriHeadAttention	.062	.442
13	NLyticsFKIE	.118	.423
	Maj. Baseline	.036	.354
14	LT3UGent	.264	.332
15	TeamUNCC	.124	.224
	Rand. Baseline	.052	.071

(b) *Results on subtask 3.*

Table 4.8: *Leaderboard of the competition evaluated on the test set. The systems are ordered by Micro-F_1 scores. Majority Baseline selects the most abundant class, while Random Baseline simply assigns labels at random.*

choice. Fine-tuning the feature extractors, in fact, enables the model to slightly adjust the weights of the backbones pre-trained on generic tasks to align them to the specific domain.

Figure 4.25 reports some examples of predictions from our model for subtask 3. We evidenced in green the true positives and in red the false positives. The model can correctly identify most of the persuasion techniques. However, there are cases where it is probably necessary to access more contextual information to solve the most complex labels. For example, in the second meme from the left, the model outputs the label *Exaggeration/Minimisation* probably due to the presence of vague quantities (*Killed thousands of innocents*). It would be necessary to access external data to effectively reason on the complex common sense and historical facts hidden behind these complex memes. In the end, the final leaderboards for both subtasks are reported in Table 4.8. We placed 5-th and 4-th respectively, maintaining relative small distances from the podium.

In the future, we plan to improve our visual feature extraction pipeline, using face expression detection and classification and possibly employing ad-hoc trained object detectors suitable for meme images. Also, it would be interesting to study the effective reasoning abilities of the proposed models, by leveraging the attention mechanisms of the Transformer, possibly integrating the data with a knowledge base of historical facts that helps to create a more suitable context.

4.6 Summary

In this chapter, we used Transformer Encoders to jointly process images and texts, mainly for semantic and relational information retrieval. Specifically, we initially presented two novel approaches called Transformer Encoder Reasoning Network (TERN) and Transformer Encoder Reasoning and Alignment Network (TERAN). Both these approaches use the Transformer self-attention mechanism to relate visual and textual concepts to create highly semantic features, embedding also entity-entity relationships. These architectures try to overcome some of the efficiency limitations of current multimodal Transformer architectures, very effective but unusable in large-scale retrieval scenarios. TERN tries to obtain a vectorial description of the two modalities into the same common space, aggregating the image region and word elements using the Transformer attention mechanism. Nevertheless, TERN does not enforce fine-grained alignment between image regions and words for a more grounded correspondence between the two modalities. TERAN tries to solve this drawback by enforcing a fine-grained match between image regions and words to preserve the informative richness of both modalities. Both TERN and TERAN features have been explored for applications in large-scale scenarios, employing quantization and sparsification methodologies with the aim of using off-the-shelf textual indexing tools. In particular, we introduced the Bag-of-Concept model as a tool for producing sparse and quantized representations from sets of contextualized features in output from the proposed architectures. The TERN features, regularized with TERAN loss, have been employed for enabling text-to-image searches in VISIONE, an internally developed tool for large-scale video retrieval. These features also demonstrated excellent results in S-CBIR, compared to other state-of-the-art image features using ad-hoc designed semantic retrieval benchmarks. Finally, deviating a little from the retrieval task, we applied the cross-modal processing abilities of the Transformer Encoder architecture to detect persuasion techniques in memes. This is a critical task in modern social networks, where images and texts are improperly employed to perpetuate disinformation campaigns.

CHAPTER 5

Solving the Same-Different Visual Problems

In the previous chapters, we discussed how high-level relational understanding can improve retrieval effectiveness, both in uni- and cross-modal scenarios. Nevertheless, the research for methods for relational understanding is interesting on its own. Humans always think by relating distant and abstract concepts through complex analogical reasoning. Therefore, it is interesting to understand to which extent a machine learning algorithm — and a DNN in particular — can solve apparently simple yet challenging tasks requiring distant comparisons. Given the importance of images in our world, we are especially interested in tackling *abstract visual reasoning* problems that require this kind of relational intelligence to be correctly solved.

This chapter aims to throw some light and partially solve apparently trivial visual reasoning tasks, known as the *same-different* tasks. In short, the same-different tasks consist in understanding if two shapes in an image satisfy a certain rule. In the simplest case, the rule is merely that *the two shapes must be equal*; however, the rule is not known a priori and must be internally understood from the provided positive and negative examples. An example is given in Figure 5.1. It is a challenging set of tasks for machine learning algorithms. In fact, it is required not to learn specific shape patterns to solve the problem; instead, they require to grow some abstract internal representation that is powerful enough to draw a logical conclusion on a fact hidden in the image (e.g., the shapes in the images are the same even if they are orientated in different ways).

Humans perceive the world as a complex set of patterns composite together to form higher-level structures, such as the repeating chorus in a song. By tackling the same-different concept, we can better understand the abstract abilities of current deep neural network models, even outside of the computer vision world. The long-term results from these studies can be applied in a wide range of disciplines, from robotics and intelligent video surveillance to cultural heritage preservation.

In Section 5.1 we briefly introduce the current literature behind abstract visual reasoning, and we present the same-different visual problems. In Section 5.2 we probe state-of-the-art CNNs on these challenging visual tasks to check their ability to converge and generalize. Then, in Section 5.3 we propose a Transformer-based recurrent architecture that can solve these problems while being conceptually simpler and more data-efficient.

This chapter collects the research published in the following papers:

- Testing Deep Neural Networks on the Same-different Task. *International Conference on Content-Based Multimedia Indexing (CBMI)*. 2019. [157];
- Solving the Same-different Task with Convolutional Neural Networks. *Pattern Recognition Letters*. 2021. [160];

5.1 Abstract Visual Reasoning

Abstract visual reasoning problems are designed to probe the reasoning abilities of machine learning algorithms, as much as IQ tests are used to measure logical and mathematical intelligence in humans. In some studies, abstract and high-level reasoning abilities are probed using some on-purpose generated complex visual puzzles. Among the most interesting ones, we find benchmarks such as CLEVR and Sort-of-CLEVR [103], that probed neural networks on the complex R-VQA task, which consists in answering questions about complex dispositions of simple 3-D shapes. An interesting research direction in relational and abstract visual understanding is undertaken by [105] and [151], which developed upon the idea of dynamically assembling an explainable program conditioned on the image-question pair, able to infer the correct answer by performing multiple reasoning steps. They reached more than 99% accuracy on the CLEVR test set. Other than the recently developed CLEVR dataset, other benchmarks were introduced to test the relational and abstract reasoning abilities of artificial vision systems. Some works tackled the abstract reasoning abilities of neural networks by using Raven's Progressive Matrices (RPM). RPMs consist of visual geometric designs with a missing part. The test taker is given a small number of different choices to pick from and fill in the missing piece. In particular, [253] tried to establish a semantic link between vision and reasoning by employing hierarchical representations suitable for relational and analytical thinking. Differently, [19] introduced Procedurally Generated Matrices (PGMs), similar to RPMs but procedurally generated using a detailed algorithm to create a fully controlled environment. They introduced a novel architecture that defeated popular state-of-the-art models like ResNets. In [240] a synthetic dataset has been introduced to test the abilities of a network of memorizing configurations. Although it is similar in essence to 2-D synthetic datasets like Sort-of-CLEVR, it is specifically designed to study the behavior of working memories.

A simple yet powerful dataset was introduced in [67]. They introduced the Synthetic Visual Reasoning Test (SVRT) dataset, composed of simple images containing closed shapes. It was developed to test the relational and comparison abilities of artificial vision systems. In [214] the authors first showed, using the SVRT dataset, that the tasks involving comparisons between shapes were difficult to solve for convolutional architectures like LeNet and GoogLeNet [218].

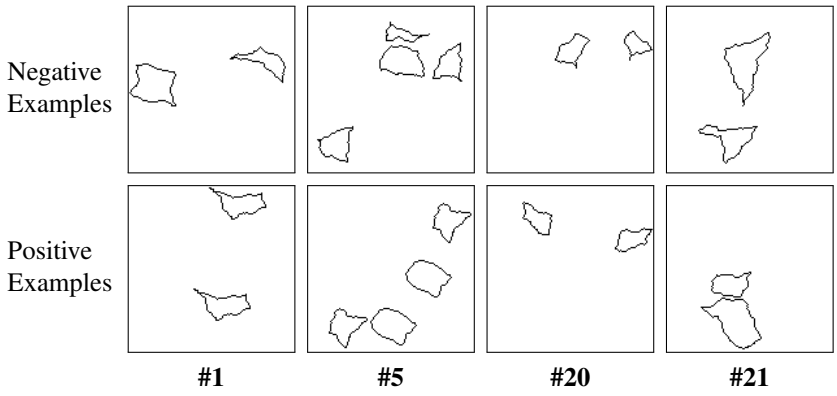


Figure 5.1: *Positive and negative examples from four same-different problems from the SVRT dataset.*

5.1.1 The Same-different Problems

The *same-different* problems are a particular subset of the SVRT dataset. In fact, SVRT collects a total of 23 different visual problems, that can be further divided into two clusters: problems related to the spatial arrangement of shapes, and problems regarding comparisons between shapes. In this work, we are not interested in understanding if the network is able to understand relative or abstract positions of the shapes in the image. In this scenario, we are only interested in probing DNNs on shapes comparisons. The latter set of problems pertain to the *same-different* challenge, and this is the set that we are interested in. Every same-different problem comes with a certain rule that should be discovered by solely looking at some images. Despite having different rules, the same-different problems have a common underlying difficulty: discerning if two objects in the same image are the same or not, under different sets of geometric transformations. The images in SVRT are visually very simple: they contain simple black closed curves on a white background. Every visual problem in SVRT is divided into two classes: the set of positive examples, which are the images that satisfy the specific rule, and the set of negative examples, which do not satisfy the rule.

In [112] the authors proposed an exhaustive evaluation of simple CNN-based networks on all the 23 different sub-tasks of the SVRT dataset. According to their findings, the most difficult same-different problems are the ones related to shape comparison under different geometric transformations (problems no. 1, 5, 20, 21). For this reason, from this point on, we tackle in great detail only these four challenging problems. In particular, to solve them we are requested to handle the following challenges: **Problem 1 (P.1)** - detecting the very same shapes, randomly placed in the image, with the same orientation and scale; **Problem 5 (P.5)** - detecting two pairs of identical shapes, randomly placed in the image. The two images inside every pair have the same orientation and scale; **Problem 20 (P.20)** - detecting the same shape, translated and flipped along a randomly chosen axis; **Problem 21 (P.21)** - detecting the same shape, randomly translated, orientated, and scaled. Figure 5.1 shows examples of positive and negative samples for each of these problems.

5.2 Solving using CNNs

CNNs constitute now a standard approach to transform a raw matrix of pixels into some higher-level representation, and they are used in many downstream tasks. Despite their success, there are many open problems with current deep architectures, and in particular with their abstract reasoning abilities. They seem still unable to distill high-level general concepts that can be transferred to different domains. This brings to low generalization abilities and often to an overfit to the specific domain on which the network is trained. Conversely, humans can recognize some shape patterns never seen before, and they can deduce some general properties of a never seen shape (e.g., *is it a closed shape? Is it the same shape as another one but rotated?*). In this work, we probe many state-of-the-art CNNs, to understand if they are able to solve these challenging visual task. We also introduce little variations to the presented architectures to better understand the role of the architectural features in the convergence or generalization abilities. In particular, we try to remove residual connections from ResNets or residual and/or recurrent connections from the CorNet-S architectures, to appreciate their role in the four same-different tasks. We show that the older VGG-19 and AlexNet architectures, in some specific cases, are able to move away from pure chance accuracy on the test set, although they cannot reach state-of-the-art results on the presented problems. In the end, we also perform final zero-shot generalization tests on the converged architectures, and we show that residual and recurrent connections can have possibly strong impacts on the final test accuracies. Despite the underlying difficulty in discerning what are the key architectural factors driving the convergence of these networks, we think that this work evidences in a systematic way the current weaknesses of current CNN-based vision models.

5.2.1 Method

This work is aimed at probing state-of-the-art architectures forming the basis of modern computer vision models, and at measuring their abilities to intrinsically perform abstract visual reasoning. The basic pool of architectures that we probe is composed of the following state-of-the-art CNNs: AlexNet [117], VGG-19 [138], three variants of the ResNet [80]: in order of increasing complexity ResNet-18, ResNet-34 and ResNet-101, and a recently introduced biologically inspired network called CorNet-S [119].

One of our objectives is trying to draw some better conclusions on the architectural factors that contribute to solving the same-different problems. For this reason, we probe also two versions of the DenseNet architecture, DenseNet-121 and DenseNet-201, which implement non-residual skip connections, and we explored the Batch Normalized version of the VGG-19. Furthermore, we try to remove important architectural building-blocks from promising architectures to try to isolate the important architectural factors triggering the convergence. In this regard, we try ResNet-18 and ResNet-34 without residual connections, and three variations of the CorNet-S obtained by removing recurrent and/or residual connections. Following, we recall the details behind the architectures that we manage to probe.

VGGs VGG-19 is a simple convolutional architecture comparable with the AlexNet structure, although it is significantly deeper. The original VGG-19, however, does not

include in its convolutional modules a batch normalization layer. This could be quite an important detail for reaching network convergence and better stability, especially if the input image is non-normalized. For this reason, we experimented also with the *VGG-19-BN* architecture, which is the Batch Normalized version of the VGG-19. It simply includes a `BatchNorm` layer after each `Conv2D`, before the ReLU activation.

ResNets ResNets introduce residual connections. This kind of skip connection helps the model to produce incremental differences in the hidden representations, dynamically refining the data passing through the network until it is sufficiently informative for the downstream task. The experiments on residual-connection removal from ResNet-18 and ResNet-34 (namely ResNet-18-WS and ResNet-34-WS, where WS = *Without Skip-connections*) can highlight the role of residual connections in solving the same-different task.

DenseNets The novel experiments conducted on DenseNets can spot the differences between a residual network and a network based on generic skip connections. In fact, the DenseNets, differently from ResNets, introduce multiple non-residual skip connections moving lower-level information to each one of the higher-level layers.

CorNet-S The authors in [108] introduced this biologically-inspired network which evolves the ResNet architecture by introducing recurrent connections. CorNet-S is inspired by some experimental evidence reported by the authors on primates brain, claiming that the visual cortex could be comprised of recurrent connections. CorNet-S is designed to mimick four different brain cortical areas involved with vision; each one of these four blocks is composed of a recurrent connection together with a residual skip connection. Thanks to the weight-sharing among timesteps, this architecture has considerably less learnable parameters than ResNets, and it is therefore an overall cheaper architecture. The introduced architecture CorNet-S-WR, where WR = *Without Recurrent-connections*, is identical to the CorNet-S model but has recurrent connections removed. This modified network is aimed at spotting out what is the influence of recurrent connections as far as the same-different problems are concerned. Similarly, CorNet-S-WS removes the residual connections from the basic CorNet-S model. We try also to remove both the recurrent and the residual links, giving rise to the CorNet-S-WR-WS model. Note that CorNet-S-WR, when unrolled, constitutes a very shallow network. To preserve the original depth, we stack in sequence the internal modules a number of times equal to the original timesteps proposed by the authors. In this way, CorNet-S-WR is effectively an unrolled version of CorNet-S, with non-shared weights among timesteps.

5.2.2 Training

All the previously described architectures come with a final classification head since they have been mostly used for the task of image classification. In our scenario, instead, the output must be a binary value indicating if the shapes in the figure correctly satisfy the rules of the specific SVRT problem or not.

For this reason, we replace the final classification head with a fully-connected layer outputting a single scalar value, normalized in the range $[0, 1]$ using a final sigmoid

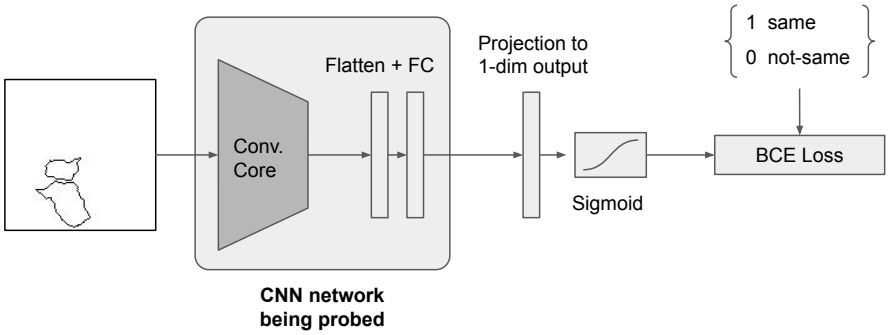


Figure 5.2: *Overview of the network for training on the same-different problems. The architecture of the network in the large light-gray box depends on the specific convolutional network being probed. It is usually composed of a core built of CNN layers plus final FC layers with ReLU activations outputting a fixed-sized vector. We linearly project the output to a single scalar value using a single FC layer. We then normalize this value in the range* $[0, 1]$ *with a sigmoid activation function before computing the Binary Cross-Entropy (BCE) loss.*

activation function. The whole network is then trained end-to-end using a Binary Cross-Entropy (BCE) loss (Figure 5.2).

5.2.3 Experimental setup

For each one of the four same-different benchmarks and every model described in Section Section 5.2.1, we use 400k images for training, 100k for validation, and 100k for testing. All the generated images in the SVRT dataset have a size of 128×128 pixels. The images were generated with the SVRT original code, available online[1]. Using 400k training examples could in principle bring to overfitting since the variability of synthetic datasets is often limited. Nevertheless, we claim that being the figures randomly generated in a 128×128 pixels grid, the probability of generating the same image twice is very low. Furthermore, humans can indeed learn the proposed problems using only a few samples. However, humans use a lot of pre-learned priors to solve these tasks, like the rotation/scale invariance, the concept of shape and closed shape, or the mirroring invariance. Humans, differently from this setup, acquire this knowledge from the experience acquired on a multitude of other tasks, usually by performing transfer learning. We try to partially solve this data hunger problem in the next section, introducing a novel hybrid CNN-Transformer architecture.

All the positive and negative examples are perfectly balanced in all the training, validation, and test sets. For all the probed models, we use SGD as the optimization algorithm, with a momentum of 0.9, weight decay of 1e-4, and a learning rate of 0.1. We do not use pre-trained weights if they are available. In fact, since these networks are trained on image classification from real-world images, the pre-trained weights cannot deal with the very different distributions of the synthetic SVRT dataset. The input images are resized to the standard 224×224 for ResNets and DenseNets architectures. We perform more experimentation on the VGG-19 and AlexNets models since they are the ones demonstrating major strain. In particular, for these architectures, we try both to resize the input images to 224×224 and to keep the original 128×128 dimensions.

[1] `https://fleuret.org/git-tgz/svrt`

	Problem 1		Problem 5		Problem 20		Problem 21	
Model	Acc.	CE	Acc.	CE	Acc.	CE	Acc.	CE
LeNet [214]	57	n.a.	54	n.a.	55	n.a.	51	n.a.
GoogLeNet [214]	50	n.a.	50	n.a.	50	n.a.	51	n.a.
AdaBoost [67]	98	n.a.	87	n.a.	70	n.a.	50	n.a.
AlexNet	50.0	-	50.0	-	50.0	-	50.0	-
AlexNet 224x224	50.0	-	50.0	-	50.0	-	50.0	-
AlexNet norm.input	80.1	-	50.0	-	76.1	-	84.1	-
VGG-19	50.0	-	50.0	-	50.0	-	50.0	-
VGG-19 224x224	50.0	-	50.0	-	50.0	-	50.0	-
VGG-19-BN	50.0	-	50.0	-	50.0	-	50.0	-
VGG-19-BN 224x224	93.8	1.5	93.1	6.0	50.0	-	50.0	-
ResNet-18	99.2	0.5	**99.9**	2.5	95.5	2.0	96.2	17.5
ResNet-18-WS	98.9	0.5	99.5	2.0	95.7	1.0	96.7	8.5
ResNet-34	98.2	4.5	98.7	1.5	93.8	6.5	96.9	13.0
ResNet-34-WS	98.6	1.0	97.6	1.5	93.6	1.0	90.8	17.5
ResNet-101	99.1	3.5	96.0	3.5	**95.8**	4.0	91.1	20.5
CorNet-S	96.9	1.0	96.8	2.0	95.0	2.0	96.9	17.0
CorNet-S-WS	95.6	1.5	97.1	2.0	92.7	3.0	90.7	18.5
CorNet-S-WR	94.2	1.5	91.0	7.5	91.5	4.0	88.3	-
CorNet-S-WS-WR	93.5	1.5	92.7	8.0	91.3	7.5	86.5	-
DenseNet-121	**99.6**	1.0	98.2	2.5	94.2	1.5	95.1	7.0
DenseNet-201	99.5	0.5	99.3	1.5	94.3	1.5	**97.5**	17.0
Human [67]	98		90		98		83	

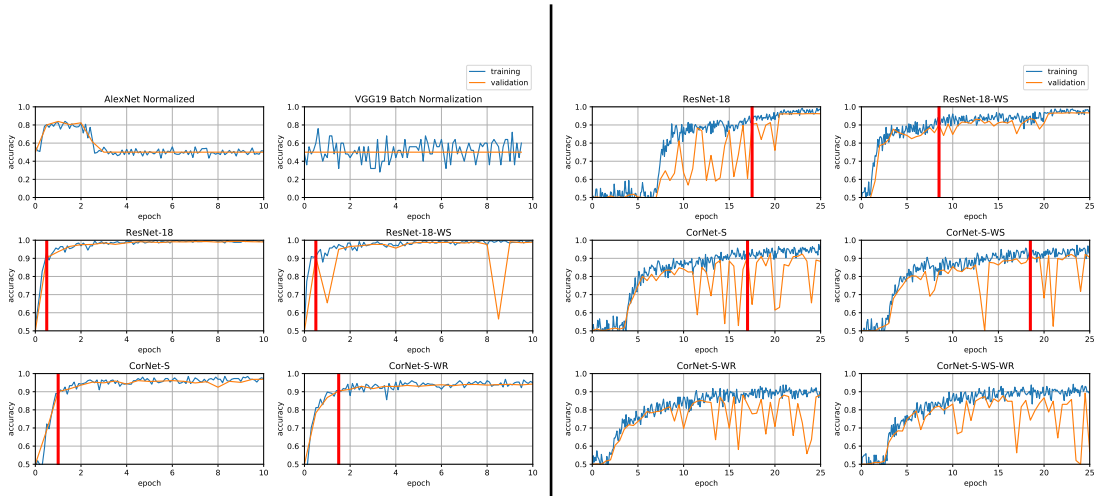
Table 5.1: *Accuracy values (%) measured on the test set of the probed architectures, for each of the four SVRT problems. Experiments reaching a perfect chance accuracy are reported in gray. The values reported from [214, 67] did not report any convergence information (CE is n.a.); also, these values are reported with the same number of significant digits as in the original papers.*

Furthermore, since AlexNet is designed with no batch-normalization layers, we tried to normalize the input images by subtracting the mean and dividing by the standard deviation of the SVRT dataset.

5.2.4 Results

Experiment 1: Convergence

In this first experiment, we aim to understand if the explored networks can converge on the four same-different problems. Thus, in this setup, we train the various models, measuring their accuracy on the test set of the same same-different problem. In this experimental scenario, we are also interested in measuring what is the strain perceived by the network during the training phase. For this reason, we desire to capture fine-grained insights during the training phase, for understanding what is the effect of different architectures — or small modifications of them — on the training curves. Reporting the training curves for all the explored networks is infeasible; however, we can extract some relevant information from the training curves and summarize them under the form of a simple metric. To this aim, similarly to [112], we extract from the training curves the point (expressed in epochs or fraction of epochs) in which the validation accuracy



(a) **(b)**

Figure 5.3: *Training curves for some of the networks trained on P.1 (a) and P.21 (b). The red vertical line is placed in the correspondence of the convergence epoch (CE). Note that it is not present for the methods that do not reach at least 90% accuracy on the validation set. For performance reasons we validate the model every half epoch, so we can provide the CE with a resolution of 0.5 epochs.*

reaches 90%. We assume that the more the network is strained, the more examples it needs during the training phase to reach a good accuracy. We call this particular point *convergence epoch* (CE). Together with our measurements, we also report the values as measured by [214, 67] on LeNet, GoogLeNet, and AdaBoost (using feature group 3).

Looking at Table 5.1, it is clear that most of the configurations derived from AlexNet and VGG-19 architectures are unable to learn or are particularly strained. More in detail, almost all the VGG-19 configurations remain on the chance level accuracy of 50%, apart from the VGG-19-BN resized to 224x224. Nevertheless, this configuration can converge on only two out of four problems, and with accuracies far below the state-of-the-art reached with ResNets and DenseNets. This is the case even for the AlexNet with normalized input images. Residual networks, as well as DenseNets, are always able to converge obtaining state-of-the-art performances on the proposed tasks. Also, residual and dense networks defeat humans on three of the four tasks. The fact that both residual and dense networks behave so well suggests that there is no significant difference between the residual connections and the DenseNet-like skip connections in this scenario. Furthermore, the ResNets without recurrent connections, (ResNet-18-WS and ResNet-34-WS) reach almost the same accuracies of the full ResNets, except in the P.21 where ResNet-34-WS obtains a lower accuracy with a higher CE, indicating a little strain with respect to the full ResNet-34. Overall, these results suggest that residual connections may be architectural building-blocks with little impact on the final test accuracy, as far as the convergence is concerned.

The comparison among the CorNet-S-WS and CorNet-S-WR shows that the lack of recurrent connections in CorNet-S has a stronger impact than the lack of the residual ones. Without recurrent connections, CorNet-S cannot reach 90% validation accuracy on P.21, leaving the CE for this experiment uncharted. Furthermore, the CorNet-S-WS-WR experiment, which lacks both residual and recurrent connections, reaches accura-

cies on the test set perfectly comparable with the CorNet-S-WR network, suggesting that the removal of residual connections has not a strong impact on the overall convergence accuracies. We claim that recurrent connections, which implies sharing the weights among timesteps, help in regularizing and stabilizing the network, and they impact considerably on the overall test accuracy when removed. In Figure 5.3a and Figure 5.3b we report also some detailed training insights for some of the architectures present in Table 5.1 and trained on P.1 and P.21 respectively. In particular, Figure Figure 5.3b shows how P.21 looks immediately more difficult when considering the training curves. The convergence epochs are very noisy and visibly shifted to the right. It is worth to mention that P.20 is the only problem that, as of now, is still overtaken by humans (last row of Table 5.1). This can be due to the intrinsic difficulties of CNNs networks to discern flipped shapes, on which humans instead are very good.

Experiment 2: Generalization

Following, we test the generalization abilities of the converged models by measuring their performance on the test set of other problems. We probe the models trained on P.21 and P.1. In particular, P.21 should force the models to learn most of the required invariances needed to solve all the other problems (translation, scale, rotation). It is interesting to understand if the networks trained on P.21 are also able to solve P.1 and P.5 and to measure their generalization abilities to shapes mirroring (P.20). On the other hand, P.1 only requires the networks to learn translation invariance; therefore, it is interesting to understand how well networks trained on this task can deal with rotation, scale, or mirroring invariance (P.20, P.21). It is also worth trying the P.1 generalization abilities to multiple shapes (P.5). Table 5.2 reports the accuracies for the most promising models trained on P.1 on the test sets of P.5, P.20, and P.21. Instead, Table 5.3 provides the accuracies for the most prominent networks trained on P.21 on the test sets of P.1, P.5, P.20.

Training on P.1 and Testing on the Others Looking at Table 5.2 it turns out that none of the models obtaining almost-perfect test accuracy on P.1 can understand P.21. This is reasonable since P.21 requires rotation and scale invariances, difficult to acquire for an architecture trained on P.1. On the other hand, ResNet-34 can generalize quite well to P.5. Note that P.5 requires the models to understand that objects should be clustered into two pairs of possibly identical shapes, and it is not trivial to deduce this information by learning only from single pairs figures. The architectural changes made to ResNets and CorNet-S have a visible impact on this generalization scenario, especially when recurrent and residual connections are removed. For example, when testing on P.5, ResNet-34 without residual connections (ResNet-34-WS) loses around 10% with respect to the basic architecture, while CorNet-S-WS and CorNet-S-WR lose 13% when compared to CorNet-S. The lack of both recurrent and residual connections in this scenario brings to a huge loss in accuracy (18%). ResNet-18 seems to be an outlier to this trend: on P.20, it obtains a better accuracy when the residual connections are removed. On P.20, the higher accuracy is reached by CorNet-S. Although approaching a test accuracy far below the optimal one, the clear deviation from chance accuracy suggests that this architecture can partially understand flipped shapes. Remarkably, the DenseNet networks cannot generalize very well to any of the three test problems.

Model	Test P.5	Test P.20	Test P.21
ResNet-18	56.5	55.6	51.6
ResNet-18-WS	56.4	58.4	51.2
ResNet-34	**84.4**	61.6	51.5
ResNet-34-WS	75.4	61.3	51.5
CorNet-S	73.6	**78.7**	52.0
CorNet-S-WS	64.6	76.8	51.7
CorNet-S-WR	63.9	71.3	**52.5**
CorNet-S-WS-WR	60.7	76.2	52.4
DenseNet-121	58.8	55.3	51.2
DenseNet-201	56.2	54.5	51.3

Table 5.2: *Accuracy values (%) measured on the probed architectures, by training on P.1 and testing on the test sets of the other three problems.*

Training on P.21 and Testing on the Others Table 5.3 shows how the models trained on P.21 can understand also P.1. This is expected since P.1 is a subset of P.21 that does not deal with scales and orientations of the shapes. A remarkable result can be observed on the networks tested on P.20: the great part of the networks trained on P.21 can solve this problem almost perfectly, although P.21 does not carry the concept of shape mirroring. In particular, DenseNet-201 can reach state-of-the-art results on both P.1 and P.20. In this generalization scenario, CorNet-S and ResNets suffer from the removal of residual and recurrent connections only when addressing P.1 and P.20, reaching the minimum accuracy on these problems with the CorNet-S-WS-WR, where both the residual and recurrent links are missing. ResNet-18-WS defines an exception to this trend as it performs better than the full ResNet-18 on these two problems. However, there is an interesting trend when comparing P.1 or P.20, with P.5. If we focus on the various CorNet-S versions, we notice that there is a decreasing accuracy trend on P.1 and P.20 when the recurrent and skip connections are gradually removed. Instead, an increasing trend is visible for P.5, although the absolute values for P.5 remain very low. The same thing happens for ResNets and DenseNets. The low absolute accuracy values obtained in P.5 suggest that it is not sufficient to be invariant to rotation, scale, or translation to understand this problem.

5.3 Solving using Recurrent Transformers

Recently, the Transformer architecture took hold in the field of image processing, as shown in Section 2.3.4. Initially developed for solving natural language processing tasks, it found its way into the computer vision world, capturing the interest of the whole community. Thanks to the ability of the Transformer of relating distant image patches through the power of the self-attention mechanism, we aim at experimenting this recent architecture on the challenging same-different problems.

As in-depth discussed in Section 5.2, the more challenging tasks have been partially solved with state-of-the-art convolutional architectures, particularly with ResNets [69, 182, 34, 160]. From these studies, it has been observed that (a) deep CNNs are needed, with lots of free parameters, to relate distant zones of the image in search of

Model	Test P.1	Test P.5	Test P.20
ResNet-18	97.9	54.2	96.0
ResNet-18-WS	98.3	53.3	96.6
ResNet-34	98.3	59.4	96.6
ResNet-34-WS	94.2	**63.4**	91.7
CorNet-S	98.6	54.2	97.0
CorNet-S-WS	95.6	59.1	91.7
CorNet-S-WR	92.4	61.4	89.9
CorNet-S-WS-WR	91.7	62.4	87.9
DenseNet-121	96.9	55.7	95.1
DenseNet-201	**98.9**	50.8	**97.4**

Table 5.3: *Accuracy values measured on the probed architectures, by training on P.21 and testing on the test sets of the other three problems.*

matching patterns, and (b) usually, a lot of data is needed to learn the underlying rule, while humans can spot it with only a few samples. Furthermore, some works [108] emphasized the role of recurrent connections, which can iteratively refine the visual input until an optimal and stable conclusion is drawn. In the light of these observations, we introduce a novel architecture, called Recurrent Vision Transformer (RViT), for solving the same-different problems. It is inspired by both the recent Vision Transformer (ViT) model [61] and by a recurrent version of the Transformer architecture, the Universal Transformer [55]. The introduced architecture can understand and relate distant parts in the image using the powerful Transformer's attentive mechanism and iteratively refine the final prediction using feedback connections. Notably, we find that the base ViT model cannot learn any of the same-different tasks, suggesting that both a hybrid architecture (upstream CNN + downstream Transformer) and feedback connections can be important features for solving the task. The code for reproducing our results is publicly available[2].

5.3.1 Recurrent Connections and Reasoning

Recurrent models – LSTMs [82] and GRUs [46], to name a few – have been widely used for dealing with variable-length sequences, especially in the field of Natural Language Processing (NLP). However, recently, many neuroscience and deep-learning works claimed the importance of recurrent connections outside the straightforward text processing, as they could have an essential role in recognition and abstract reasoning. The work in [108] claimed that the visual cortex could be comprised of recurrent connections, and the visual information is refined in successive steps. Recurrent architectures are used to generate executable programs for compositional reasoning [106, 11]. Differently, many works in deep learning tried to achieve Turing-completeness by creating recurrent architectures with dynamic halting mechanisms [75, 55, 18]. Iterative computation is a key paradigm in algorithmics, but it has been poorly studied in the past literature. Nevertheless, very recent and pionieering works [238, 222] try to align classical computation theory — dynamic programming in particular — with deep learning

[2] https://github.com/mesnico/recurrent_vision_transformer_visual_reasoning

architectures, showing that Graph Neural Networks can learn graph algorithms with a better generalization margin. The long-term return from this whole research is to make neural architectures able to deal with logical and algorithmic thinking, an essential requirement for abstract reasoning.

This work aims at transforming the multi-layer Transformer Encoder (TE) into a recurrent TE, in which the information from each image patch is propagated to all the neighbors, as in a GNN with a complete-graph structure. This has the effect of updating the internal representations using shared computational steps at each iteration; the message passing is performed until a confident result is obtained.

5.3.2 The Recurrent Vision Transformer Model

The proposed model is based on the recent Vision Transformer – in particular, the Vision Transformer (ViT) model [61]. The drawback of CNNs in solving the same-different problems is that sufficiently deep networks are needed to correlate distant zones in the image. The Transformer-like attention mechanism in ViT helps in creating short paths between image patches through the self-attention mechanism. Furthermore, inspired by the role of recurrent connections in the human's visual cortex [108], we modify the ViT Transformer encoder module by sharing the encoder weights among all the T layers (i.e., along the depth dimension), effectively creating a recurrent Transformer encoder model, similar to [55]. This has the effect of sharing weights not only in the sequence dimension as in standard Transformers, but also in the depth dimension, further constraining the model complexity. Differently from the base ViT which uses simple linear projections to obtain features from pixels, we instead use a CNN feature extractor, more capable of understanding low level image features like contours or edges. For this reason, as a feature extractor, we use a small upstream CNN that outputs a $N \times N$ grid of features that are then used as visual tokens in input to the Transformer encoder. The overall architecture is shown in Figure 5.4.

By leveraging the recurrent nature of the architecture, we avoid explicitly tuning the depth of the network (i.e., the total number of recurrent iterations) by forcing the architecture to perform a prediction at each time step, using the CLS token. The most likely outcome among the predictions from all the time steps is then taken as the final prediction. More in detail, the model comprises T binary classification heads, one for each time step. During training, the Binary Cross-Entropy (BCE) loss at each time step is computed as

$$\mathcal{L}_t = \text{BCE}(y_t, \hat{y}), \tag{5.1}$$

where y_t is the network output from the t-th time step, and $\hat{y}$ is the ground-truth value. The various losses are then aggregated to obtain the final loss $\mathcal{L}_{\text{total}}$. We noticed that a simple average $\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{\text{t}}$ already led to good results. However, we obtained the best results by using the automatic loss-weighting scheme proposed in [110]:

$$\mathcal{L}_{\text{total}} = \frac{1}{2}\sum_{t}^{T}\left(\frac{1}{e^{s_t}}\mathcal{L}_t + s_t\right), \tag{5.2}$$

where s_t is a free scalar parameter that encodes the predicted uncertainty of the classification at the t-th time step, and the model automatically learns it during the training phase. We refer readers to [110] for more detailed derivation and discussion.

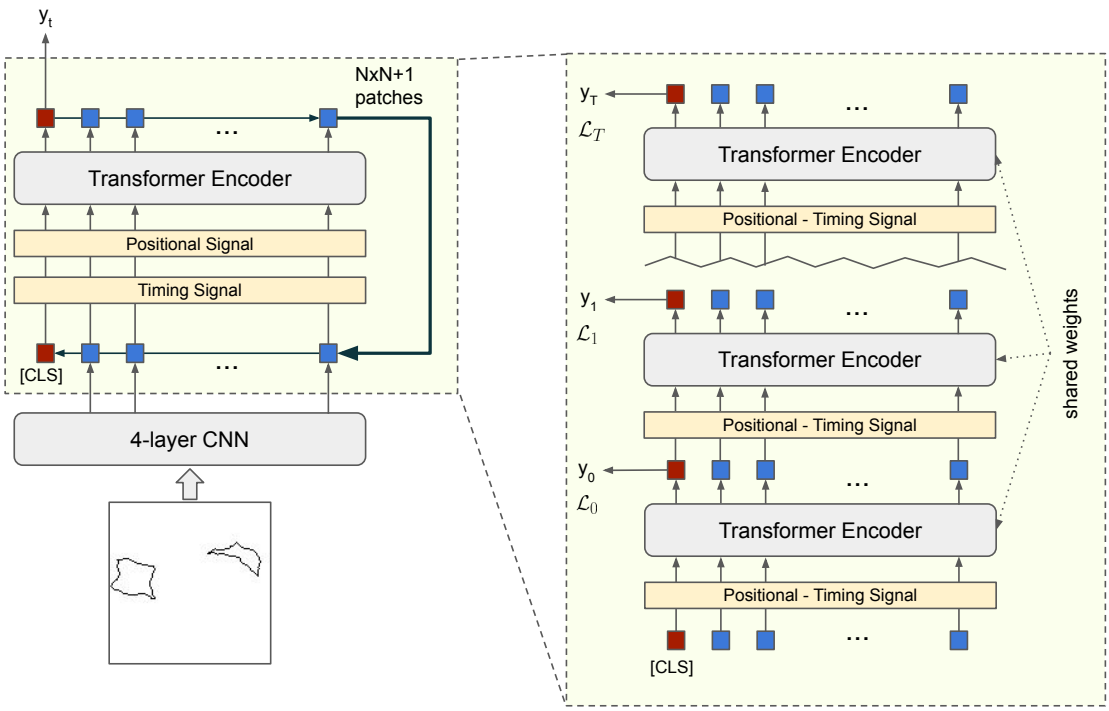


Figure 5.4: *The RViT architecture. The image is processed by a 4-layer CNN, outputting a 8×8 grid of visual features. The CLS token is added to this set, and the tokens are processed multiple times by the recurrent Transformer Encoder module. At each time step, the binary cross-entropy loss is computed against the ground-truth labels.*

During inference, the maximum-likelihood prediction is taken as the final network output. In particular, the time step $\bar{t}$ at which the network reaches the maximum confidence is the one where the output probability is farthest from the pure chance in a binary classification setup (p=0.5):

$$\bar{t} = \arg\max_t |y_t - 0.5|. \tag{5.3}$$

At this point, the final output is simply $y = y_{\bar{t}}$.

5.3.3 Experimental Setup

For the upstream CNN processing the pixel-level information, we used a 4-layer *Steerable CNN* [233]. A Steerable CNN describes E(2)-equivariant (i.e., rotation and reflection equivariant) convolutions on the image plane $\mathbb{R}^2$; in contrast to conventional CNNs, E(2)-equivariant models are guaranteed to generalize over such transformations other than simple translation and are therefore more data-efficient. In the ablation study in Section 5.3.5, we give more insights on the role of Steerable CNNs over standard CNNs in solving the same-different task.

We forged two different versions of the RViT, a *small* and a *large* version, having the same structure but a different number of hidden neurons in the core layers: the small RViT produces 256-dimensional keys, queries, and values and outputs 256-dimensional visual features from the CNN, while the large RViT has these two parameters set to 512. We used the Adam optimizer; after a minor hyper-parameter tuning, we set the learning

Model	400k training samples				28k training samples				#pars
	P.1 ↑	**P.5** ↑	**P.20** ↑	**P.21** ↑	**P.1** ↑	**P.5** ↑	**P.20** ↑	**P.21** ↑	↓
RN [204]	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	0.4M
ViT [61]	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	26M
ResNet-18 [160]	99.2	**99.9**	95.5	96.2	99.2	98.4	93.7	50.0	11M
ResNet-50 [34]	-	-	-	-	95.4	89.9	92.9	72.6	23M
DenseNet-121 [160]	99.6	98.2	94.2	95.1	73.9	54.7	94.4	**85.8**	6.9M
CorNet-S [160]	96.9	96.8	95.0	**96.9**	98.8	97.1	92.3	82.5	52M
RViT-small	**99.9**	99.4	**98.9**	95.7	**99.6**	98.0	93.9	78.6	**0.9M**
RViT-large	**99.9**	99.0	98.8	96.4	**99.6**	**99.3**	**95.3**	77.8	3.1M

Table 5.4: *Accuracy (%) of our method, trained from-scratch, with respect to the baselines. #pars indicate the number of free parameters of the model.*

Model	**P.1** ↑	**P.5** ↑	**P.20** ↑	**P.21** ↑	#pars ↓
ResNet-50 [34]	99.5	98.7	98.9	**92.5**	23M
RViT ResNet-50/11	99.6	98.6	94.5	91.6	**2.3M**
RViT ResNet-50/23	**99.7**	**99.7**	**99.4**	85.2	9.5M

Table 5.5: *Accuracy (%) of RViT-small, with the first layers of a ResNet-50 pre-trained on ImageNet, with respect to the full ResNet-50 baseline. In ResNet-50/11 we kept the first 11 layers, while in ResNet-50/23 the first 23.*

rate for all the experiments to 1e-4, and the number of attention heads to 4; we let the models train for 200 epochs, decreasing the learning rate to 1e-5 after 170 epochs. We tested the models using the snapshot with the best accuracy measured on the validation set.

In order to better compare with the ResNet-50 experiments in [34], we also tried to use as up-stream CNN the first two or three layers of a ResNet-50 pre-trained on ImageNet. For the image resolution, we mainly used $N = 16$, outputting 16×16 visual tokens from the CNN. During the pre-training experiments, instead, we used $N = 8$ for accommodating the output feature map resolution of the pre-trained model and also for performance reasons. During training, we set the maximum time steps $T = 9$.

We collected results using both 28k training images, following [34], and 400k training images, for comparing our proposed architectures with convolutional networks previously trained as explained in Section 5.2. We used 18k images both for validation and testing. As the previous experiments on CNNs, the images were generated using the generation code provided by the authors of the SVRT dataset [67].

5.3.4 Results

We compared our model with other key architectures: the Relation Network (RN) [204] which by design should be able to correlate distant zones of the image; the Vision

Transformer (ViT) [61] which recently achieved remarkable performance on classification tasks, although it is very data-hungry, and some state-of-the-art convolutional models — ResNet-18, ResNet-50, CorNet-S and DenseNet-121 — trained on the same task in [34] and Section 5.2. Notably, CorNet-S also implements feedback connections, although it is much more complex, in terms of number of parameters, than our RViT architecture.

Looking at Table 5.4, we can see how neither the Relation Network nor the ViT converges on the four visual problems, for both 400k and 28k data regimes. The ViT probably needs more architectural inductive biases to understand the rules, while the relational mechanism of Relation Network is probably too simple for understanding the objects in the image and their relationships. Instead, our RViT model can obtain very competitive results on all tasks and on both data regimes, often outperforming the baselines. Noticeably, the RViT-small can learn all the four problems using only 0.9M free parameters, about 8 times fewer parameters than the smallest convolutional network able to solve the task (DenseNet121). This suggests that the model has the correct structure for understanding the visual problems, without having the possibility to memorize the patterns.

In Table 5.5, we instead report the accuracy of the small RViT model, where the upstream path is pre-trained on the classification task on ImageNet, following the work in [34]. Even in this case, the RViT achieves competitive results, but with much fewer free parameters and using only a slice – the first 11 and 23 layers – of the pre-trained ResNet-50 architecture.

5.3.5 Ablation Study

Following, we report some in-depth analysis of the RViTs performed with 28k training images.

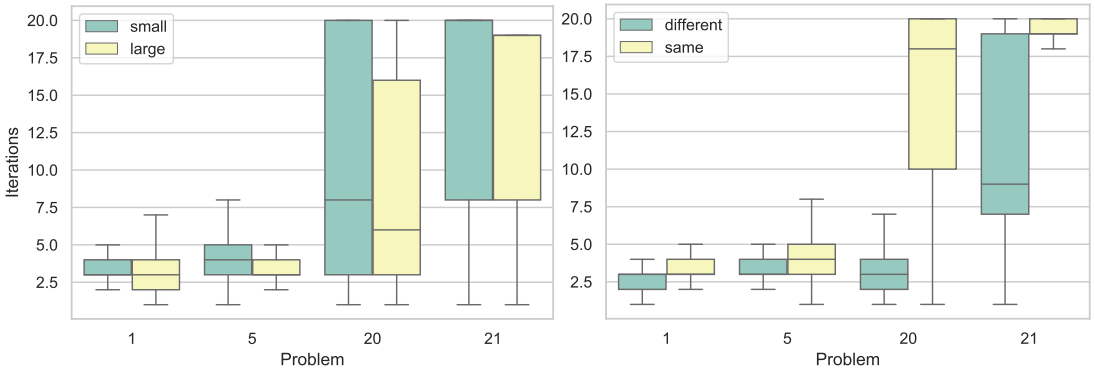
The role of Recurrent Connections and Steerable Convolution

In Table 5.6, we experimented with some variations of the RViT to understand the roles of recurrent connections and the employed 4-layers steerable CNN. The basic configuration is Conv. ViT, which is the same as the standard ViT from [61] but with an upstream CNN as the visual feature extractor, without any recurrent connection. In contrast to the original ViT formulation, the Conv. ViT can improve significantly on P.1, P.20, and P.21, moving away from the chance accuracy. However, the most significant jump in accuracy happens when recurrent connections are introduced (Conv. RViT). In this case, the same model can learn all the visual problems, with an improvement of 67% on P.1 and 7% on P.21. Another improvement is obtained when using the Steerable CNNs [233]. This kind of CNN produces features equivariant to rotations and reflections. For this reason, it has a wider impact on P.20 and P.21, where shapes are reflected and rotated, respectively.

Recurrent connections seem to have critical importance. They highly regularize the model, making it more data-efficient and performing a dynamic iterative computation that procedurally refines both the previous internal representations and the previous predictions. To better appreciate this aspect, in Figure 5.5 we show the mean time step $\bar{t}$, for each problem, where the model reaches the maximum confidence. Interestingly, P.1 and P.5 reach the best confidence in few iterations, while the more challenging

Model	P.1 ↑	P.5 ↑	P.20 ↑	P.21 ↑
Conv. ViT	59.5	50.0	88.5	62.5
Conv. RViT	99.9	99.0	93.9	66.8
Eq. Conv. RViT	99.8	99.4	95.6	77.3

Table 5.6: *Ablation study on Convolutional ViT (Conv. ViT), on Convolutional Recurrent ViT (Conv. RViT), and Equivariant Convolutional Recurrent ViT (Eq. Conv. RViT). The last one is the model effectively employed in Table 5.4 and Table 5.5. Accuracy (%) is in this case measured on the validation set.*



(a) **(b)**

Figure 5.5: *The distribution of the best time step $\bar{t}$ grouped by (a) the two different RViT sizes (small, large), and (b) by the same-different label.*

P.20 and P.21 need much more pondering before stabilizing. More in detail, it can be noticed that although there is not too much difference considering the size of the models (Figure 5.5a), the network seems a bit more strained when the shapes are the *same* (Figure 5.5b). This is reasonable: it is heavier to be sure that shapes coincide in every point, while it takes little to find even a single non-matching pattern to output the answer *different*.

Visualizing the Attention

In Figure 5.6, we reported a visualization of the self-attention maps learned by the trained models, computed in specific points (marked with red dots) in the image, and by averaging the four attention heads. The 16×16 grid allows us to appreciate fine details; in particular, we can see what parts of the shapes the model is attending to for producing the final answer. In most cases, the model correctly attends the other shape in search of the corresponding edges. In some instances, the attention map is not so neat (e.g., in (d) and (f)), emphasizing the intrinsic complexity of the tasks. Furthermore, in Figure 5.7 we report the evolving attention maps at different time steps. The map is initially very noisy, but it is slowly refined as the number of iterations increases to create a stable representation.

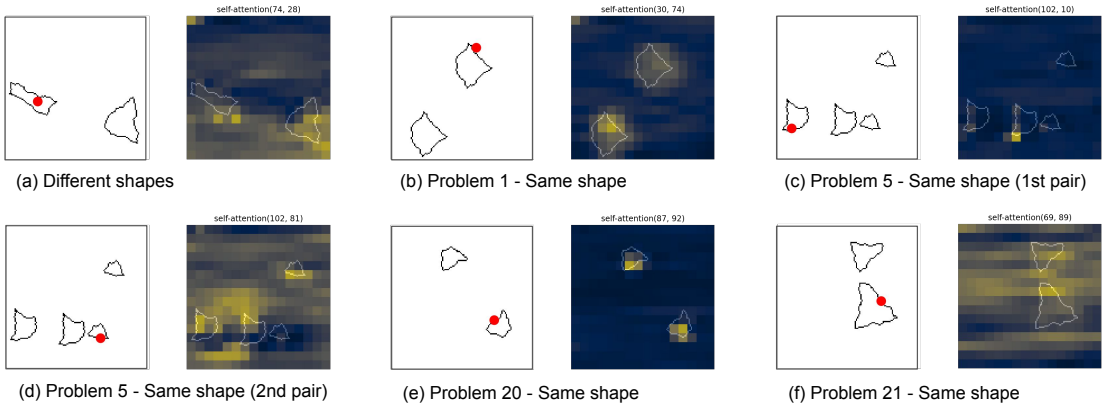


Figure 5.6: *Attention visualization on the different visual problems. The red dot shows the point in space with respect to which the self-attention is computed.*

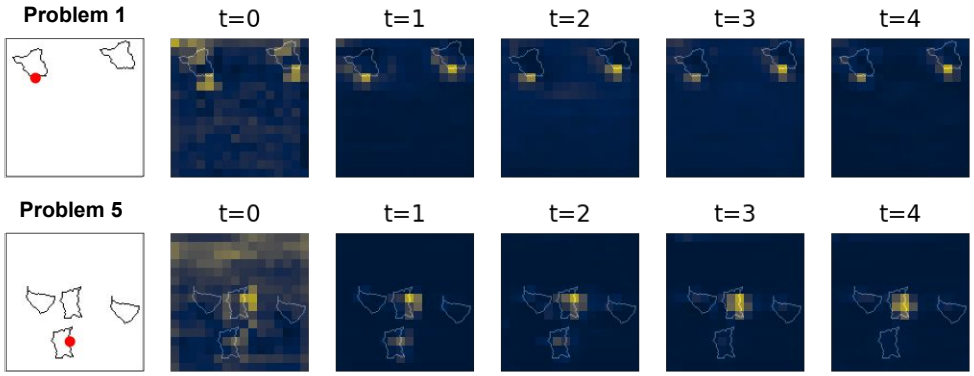


Figure 5.7: *Evolving attention maps at different time steps.*

5.4 Summary

In this chapter, we explored the ability of DNNs to solve the *same-different* tasks, a specific class of visual abstract reasoning problems. We initially probed some state-of-the-art CNN architectures, finding that they usually require to see many data samples before converging and an unplanned huge representation power ($\sim 1 \times 10^7$ free parameters). Driven by these concerns and inspired by the recent success of the Transformer network in computer vision, we then introduced the Recurrent Vision Transformer (RViT) model. Thanks to the impact of recurrent connections and spatial attention, this network achieves competitive results on the four referenced same-different problems. The weight-sharing both in spatial and depth dimensions regularizes the model, allowing it to learn using less than 1M free parameters, with only 28k training samples. In the end, this study lays the basis for a deeper understanding of the role of attention and recurrent connections for solving visual abstract reasoning tasks. Furthermore, it can drive the research towards deep learning methods for solving slow-thinking logical problems, requiring multiple iterated basic steps to converge to a solid conclusion. In the future, we plan to transfer the seeds of this research to real use cases, where multiple possibly distant inputs need to be related and analyzed to draw a conclusion — for example,

judge if two similar walking pedestrians caught from two different city cameras are the same person or not, in surveillance applications.

CHAPTER 6

Conclusions

Relationships shape our world, as they enable us to link multiple — and apparently unrelated — entities creating a new layer of knowledge from sensory data. For example, we can conclude that a *ball* can be *kicked* if we relate the ball with the *football player* which is interacting with it. This would be a challenging conclusion to draw by solely looking at the ball. In this case, in fact, we would be able to derive some low-level evidence (e.g., the ball is similar to a sphere, and it has some nice tiled textures) without being able to understand what a ball is used for or who or what may use it. Guided by these concerns, in this thesis, we studied relational shortages of current Deep Neural Networks in multiple application scenarios. In particular, sticking to the Computer Vision domain, we noticed that the local processing nature of Convolutional Neural Networks does not enable enough interaction among distant zones in the image, severely disadvantaging the discovery of important high-level relationships between actors — objects, animals or persons — in a scene. We explored the application of recently proposed non-local architectures able to discover long-range relationships, such as Relation Network (RN) [205] and Transformers [220], in the following scenarios:

- we used these novel architectures as feature extractors for obtaining visual and textual descriptors able to perform very semantic and relationship-aware Content-based Image Retrieval and Cross-modal Visual-textual Retrieval;
- we explored their application in specific visual abstract reasoning tasks, where CNNs seem to struggle.

In Chapters 1 and 2, we provided an introduction to this thesis, and we discussed the relevant background, introducing the Deep Learning framework and the current state-of-the-art architectures for processing images and natural language texts. We introduced Relation Network, Transformers, and Graph Networks, as the main recently

proposed architectures able to perform non-local visual and textual processing.

In Chapters 3 and 4, we presented novel relationship-aware networks in the context of semantic information retrieval. In particular, in Chapter 3 we introduced the challenging task of Relational Content-based Image Retrieval (R-CBIR), which consists in retrieving images with similar spatial relationships among objects. The original Content-based Image Retrieval (CBIR) task usually deals with instance retrieval, which does not require a high-level understanding of the content. Instead, it has a high sensitivity to low-level pattern matching, as it requires a good understanding of colors, shapes, textures. However, apart from inferring the object class, no high-level semantic is needed to solve the task. In this chapter, we demonstrated how features extracted from an architecture trained on Relational Visual Question Answering (R-VQA) can defeat very effective features for instance retrieval, like R-MAC [219], on this novel R-CBIR task. More in detail, we proposed some modifications to the original Relation Network formulation, and we extracted visual relationship-aware features from the resulting architectures. We validated these novel features for R-CBIR both quantitatively — introducing a novel ground-truth based on the CLEVR dataset — and qualitatively, demostrating the effectiveness of our approach.

We further extended this preliminary work on semantic retrieval in Chapter 4, where we tackled cross-modal visual-textual retrieval. An alternative way to obtain visual semantic features is to match them with natural language descriptions of the scene, which usually carry very high-level details, and encode relationships into predicates (e.g., "The kid *is running* in the field"). This, in turn, enabled us to perform visual-textual retrieval, where we used natural language queries to find images and vice-versa. In this chapter, we used the novel Transformer architecture to obtain a relational representation for images and texts. Unlike many state-of-the-art Transformer-based methodologies for effective image-text matching, this research focused on producing compact representations that could be used for efficient and scalable cross-modal retrieval. The visual features obtained by forwarding only the visual pipeline were also able to perform well on Semantic Content-based Image Retrieval (S-CBIR) on a recently released benchmark for Semantic Image Similarity, called Crisscrossed Captions. With this result, we demonstrated that architectures jointly trained on images and texts could be used to extract semantic and relationship-aware image descriptions, closing the loop initiated in Chapter 3. Furthermore, in Chapter 4, we applied the multi-modal Transformers to another important real-world scenario, namely the detection of persuasion techniques in social networks. We tackled this problem by using both images and texts from *memes*, the most widespread tool used for propagating misinformation, placing in a good position in the SemEval 2021 competition leaderboard.

In Chapter 5, we followed a more fundamental line of research, trying to solve visual abstract reasoning problems requiring the comparison of multiple shapes in an image. This set of tasks is also known as the *same-different* problems, introduced with the SVRT dataset [67]. Although this research is less directly transferable to real-world applications, it enabled us to understand better what are the limits of current CNNs when they have to face native relational problems. In particular, the same-different tasks require the comparison between possibly distant shapes in the image, and the local processing performed by CNNs usually cannot discover distant relationships. For this reason, CNNs usually require a very deep architecture that increases the overall

complexity and, in turn, requires many data. In this analysis, we tried to shed some light on the role of residual, skip, and recurrent connections. In particular, thanks to some biological evidence, recurrent connections seemed to have an essential role in visual reasoning tasks. Given all these clues, we developed a recurrent CNN-Transformer architecture that could solve the problems with less free parameters and, in turn, higher data efficiency. This architecture enabled us to inspect attention weights learned by the model at different reasoning timesteps, suggesting that attention and recurrent connections could be the key to reaching better results on abstract reasoning problems.

In the end, in this thesis, we demonstrated that approaches based on relational Deep Learning brought many improvements in semantic image retrieval, cross-modal retrieval, and visual reasoning. We argued how relationship-aware descriptors, extracted from Relation Networks and Transformer-based architectures, could defeat non-relational descriptors for instance retrieval, obtaining excellent results in R-CBIR and S-CBIR. In particular, our TERN features obtained an improvement of about 50% with respect to the R-MAC ones on the novel CxC dataset for Semantic Image Similarity. We also demonstrated that Transformer Encoders contribute to creating relationship-aware representations for efficient cross-modal retrieval. At the time of writing, TERAN defined the state-of-the-art approach on visual-textual retrieval, among the approaches that used late-fusion methods for large-scale search. Furthermore, the hybrid CNN-Transformer architecture used to solve the same-different problems demonstrated how perception and brain-inspired reasoning could cooperate to solve abstract visual problems. With this architecture, comprised of less than 1M free parameters, we obtained state-of-the-art results on the most challenging tasks from the SVRT dataset, using only 28k training samples.

6.1 Further Activities

During the PhD period, I also participated in other activities that are weakly coupled with the main topics presented in this thesis. In particular, in the last few years, there has been an increasing interest in the use of synthetic data to overcome the lack or scarcity of data for training Deep Neural Networks on particular tasks. An example is the presented CLEVR dataset, artificially generated to avoid data biases, thus obtaining a perfect test sandbox. However, synthetic data is also employed to avoid the costly manual annotation or produce a multitude of real-world scenarios, possibly with custom lighting or weather conditions.

To this end, I contributed to two works in which we studied the effect of using synthetic data to train pedestrian detectors for surveillance applications. In the first work [7], we presented a real-time pedestrian detection system trained using a virtual environment. In particular, we introduced ViPeD[1], a new synthetically generated set of images extracted from a realistic 3D video game where the labels can be automatically generated exploiting 2D pedestrian positions extracted from the graphics engine. We exploited this new synthetic dataset fine-tuning the state-of-the-art YOLO [195] object detector. A preliminary experimental evaluation, compared to the performance of other existing approaches trained on real-world images, showed promising results on the generalization abilities of the trained model on real-world scenarios. In the second work

[1] http://aimh.isti.cnr.it/viped/

[48], we further extended these results. We proposed two different domain adaptation techniques, and we tested our claims on more real-world data, also using another state-of-the-art object detector, Faster-RCNN [198].

These secondary lines of research are currently active. In the near future, we plan to merge the results from surveillance camera analysis and relational deep learning to solve tasks like multi-camera tracking and anomaly detection, handling spatial and temporal relationships using the tools researched and developed in this thesis.

6.2 Future Work

The research carried out in the main dissertation chapters inspired some interesting further studies, in light of very recent and fast developments in these fields. In the following paragraphs, we report some of the most promising extensions to the contributions presented in this thesis.

R-CBIR using cross-modal features In Chapter 3, we introduced some modifications to the Relation Networks (RNs) to extract meaningful relationship-aware image descriptors for solving the challenging Relational Content-based Image Retrieval (R-CBIR) task. In the last few years, we witnessed a growing interest in high-semantic features which might be natively able to capture spatial relationships like, for example, the CLIP [61] or the ALIGN [100] features. In this regard, it could be interesting to study their ability to handle different kinds of object-object spatial relationships and measure how much their L2 or angular distances correlate with distances between the underlying scene graphs. Following the results obtained in semantic image retrieval in Section 4.3, it might be worth conducting the same exploration for the TERN or the TERAN features.

End-to-end learning of the Bag-of-Concepts model In Chapter 4, we introduced the Bag-of-Concept model, whose aim is to create compact semantic descriptors for cross-modal retrieval. The obtained vectors have an immediate and explainable interpretation, as each dimension encodes the presence of each of the abstract concepts present in the image or in the text. Nevertheless, the creation of the concept codebook and the sparsification of the final vector are implemented as post-processing operations after the features extraction process. It would be interesting to develop an architecture equipped with an internal memory working as a learned codebook, automatically refined during the training process. Also, it would be possible to automatically sparsify the generated vectors so that the training process can automatically find the best compromise between sparsity — and, in turn, efficiency — and effectiveness of the produced cross-modal features.

Fine-grained scalable cross-modal search The use of large-scale retrieval frameworks — either vector-based indexes such as FAISS[2] or text-based engines like Lucene used to index multimedia data like explained in Chapter 4 — is constrained to a simple concept: the item that we want to index, as well as the query that we submit to the system, should be represented as single fixed-sized vectors, laying in some $\mathbb{R}^n$ manifold. In this way, cosine similarity and its properties can be leveraged to enable effective and large-scale

[2] https://github.com/facebookresearch/faiss

search. In most of cases, this interface is just the right compromise between efficiency and effectiveness. Nevertheless, it can be constrictive for fine-grained search, especially in cross-modal scenarios. For example, there are cases in which we would like to weigh the words differently in the query sentence to acquire more control over the image retrieval process at query time. This is unfeasible unless we consider fine-grained region-words associations during the indexing and the search phases. Following this line of research, a very recent work [147] proposed to use a weighted Bag-of-Words (BoW) to describe each image as a set of words, obtaining remarkable results on MS-COCO and Flickr30k, while being very efficient when implemented in inverted lists. This approach seems similar to the Bag-of-Concept model, except that it uses words from the original sentence and not high-level concepts automatically learned by the network. Another possibility would be to use metric-space approaches defined on graphs or set structures to obtain fine-grained similarity scores. In particular, we could rely on the work in [16], which proposed a fast way to compute the Wasserstain distance between two different sets of vectors.

Text-to-Video search In Chapter 4, we used the developed text-to-image search tools for addressing large-scale *video* retrieval. Videos were considered a set of independent keyframes so that video retrieval could be trivially cast to an image retrieval problem. Nevertheless, the exploitation of the temporal domain is required to understand actions or camera movements, as we noticed while discussing the Textual KIS Task from the VBS challenge. Given these current limitations, we manage to extend our work on cross-modal retrieval by taking into account also the time domain. Recently, many effective text-to-video features have been proposed, like the W2VV++ features [130], or the CLIP image features [191] intelligently aggregated among consecutive frames to work on videos [181, 148, 45]. It would be interesting to follow this research direction to enable, as in TERAN, a closer interaction between *tubes* — the extension of *image regions* to spatio-temporal *video regions* — and words from the query sentence.

Interactive cross-modal search Another promising extension to the research presented in Chapter 4 would be the modification of the search paradigm to include more *interactive* search strategies. Specifically, we consider the search system interactive if it employs user feedback to refine the search results in a continuous feedback loop. In this sense, it could be interesting to extend TERN to the task of retrieving images given an (image, text) pair instead of relying on the textual query alone. The problem could be framed as in [76], where a deep neural network is trained to alter the semantics of an input image using a *modification text*. In this way, we could initially retrieve images using some natural text (e.g.: *"A pink dressed woman"*). Then, we could refine the results by feeding back the images retrieved at the previous search step, integrating them with some other textual insights (e.g.: *"The woman wears black shoes"*). This interactive search paradigm based on user feedback is gaining increasing attention in information retrieval [141, 252], and it may be worth assessing the deployment of similar methods in VISIONE [8].

Abstract reasoning in more challenging scenarios We in-depth studied the behavior of Deep Neural Networks in very specific and challenging abstract visual reasoning problems,

the *same-different* problems, in Chapter 5. However, it would be interesting to probe the proposed networks on more challenging abstract reasoning problems, such that the one proposed in the more recent PGM [19] or in the RAVEN [253] datasets. These benchmarks collect matrices of figures similar to those proposed in IQ tests for humans, and the objective is to find the missing image from a set of candidate answers. These puzzles require high-level, abstract, and analogical reasoning capabilities to be solved. It would be interesting to test some variations of the recurrent Vision Transformer proposed in Chapter 5 on these challenging tasks to check if the conclusions drawn on the same-different task also apply to these scenarios. Furthermore, it would be interesting to study these networks' analogical and relational skills in real-world settings, where perception and reasoning work together to solve critical tasks like multi-camera anomaly detection or multi-target tracking in surveillance applications.

Bibliography

[1] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. 1, 01 2015.

[2] Giuseppe Amato, Claudio Gennaro, and Pasquale Savino. Mi-file: using inverted files for scalable approximate similarity search. *Multim. Tools Appl.*, 71(3):1333–1362, 2014.

[3] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. Deep permutations: Deep convolutional neural networks and permutation-based indexing. In *SISAP 2016*, volume 9939, pages 93–106, 2016.

[4] Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. Large-scale image retrieval with elasticsearch. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 925–928, 2018.

[5] Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Franca Debole, Fabrizio Falchi, Claudio Gennaro, Lucia Vadicamo, and Claudio Vairo. Visione at vbs2019. In *International Conference on Multimedia Modeling*, pages 591–596. Springer, 2019.

[6] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. Large-scale instance-level image retrieval. *Information Processing & Management*, page 102100, 2019.

[7] Giuseppe Amato, Luca Ciampi, Fabrizio Falchi, Claudio Gennaro, and Nicola Messina. Learning pedestrian detection from virtual worlds. In *International Conference on Image Analysis and Processing*, pages 302–312. Springer, 2019.

[8] Giuseppe Amato, Paolo Bolettieri, Fabrizio Falchi, Claudio Gennaro, Nicola Messina, Lucia Vadicamo, and Claudio Vairo. Visione at video browser showdown 2021. In *International Conference on Multimedia Modeling*, pages 473–478. Springer, 2021.

[9] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer, 2016.

[10] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[11] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Deep compositional question answering with neural module networks. *CoRR*, abs/1511.02799, 2015. URL `http://arxiv.org/abs/1511.02799`.

[12] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015. URL `http://arxiv.org/abs/1505.00468`.

[13] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 36–45, 2015.

[14] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 1269–1277, 2015.

[15] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European conference on computer vision*, pages 584–599. Springer, 2014.

[16] Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Scalable nearest neighbor search for optimal transport. In *International Conference on Machine Learning*, pages 497–506. PMLR, 2020.

[17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[18] Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

[19] David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 511–520, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[20] David GT Barrett, Ari S Morcos, and Jakob H Macke. Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Current opinion in neurobiology*, 55:55–64, 2019.

[21] Björn Barz and Joachim Denzler. Hierarchy-based image embeddings for semantic image retrieval. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 638–647. IEEE, 2019.

[22] Björn Barz and Joachim Denzler. Content-based image retrieval and the semantic gap in the deep learning era. In *International Conference on Pattern Recognition*, pages 245–260. Springer, 2021.

[23] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL `http://arxiv.org/abs/1806.01261`.

[24] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[25] Eugene Belilovsky, Matthew B. Blaschko, Jamie Ryan Kiros, Raquel Urtasun, and Richard Zemel. Joint embeddings of scene graphs and images. *ICLR*, 2017.

[26] Aurélien Bellet, Amaury Habrard, and Marc Sebban. Metric learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 9(1):1–151, 2015.

[27] Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[28] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[29] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34 (5):1–41, 2007.

[30] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[31] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. Deep learning for ai. *Communications of the ACM*, 64 (7):58–65, 2021.

[32] Fabian Berns, Luca Rossetto, Klaus Schoeffmann, Christian Beecks, and George Awad. V3c1 dataset: an evaluation of content characteristics. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 334–338, 2019.

[33] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[34] Judy Borowski, Christina M Funke, Karolina Stosio, Wieland Brendel, TS Wallis, and Matthias Bethge. The notorious difficulty of comparing human and machine perception. In *2019 Conference on Cognitive Computational Neuroscience*, pages 2019–1295, 2019.

[35] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[36] Andrew Brown, Weidi Xie, Vicky Kalogeiton, and Andrew Zisserman. Smooth-ap: Smoothing the path towards large-scale image retrieval. In *European Conference on Computer Vision*, pages 677–694. Springer, 2020.

[37] Charles F Cadieu, Ha Hong, Daniel LK Yamins, Nicolas Pinto, Diego Ardila, Ethan A Solomon, Najib J Majaj, and James J DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS Comput Biol*, 10(12):e1003963, 2014.

[38] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *CoRR*, abs/1709.07604, 2017. URL `http://arxiv.org/abs/1709.07604`.

[39] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[40] Fabio Carrara. Deep learning for image classification and retrieval: Analysis and solutions to current limitations. 2019.

[41] Fabio Carrara, Andrea Esuli, Tiziano Fagni, Fabrizio Falchi, and Alejandro Moreo. Picture it in your mind: Generating high level visual representations from textual descriptions. *Information Retrieval J.*, 21(2-3): 208–229, 2018.

[42] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, 2018.

[43] Hui Chen, Guiguang Ding, Xudong Liu, Zijia Lin, Ji Liu, and Jungong Han. Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12655–12663, 2020.

[44] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019.

[45] Xing Cheng, Hezheng Lin, Xiangyu Wu, Fan Yang, and Dong Shen. Improving video-text retrieval by multi-stream corpus alignment and dual softmax loss. *arXiv preprint arXiv:2109.04290*, 2021.

[46] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[47] Gobinda G Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010.

[48] Luca Ciampi, Nicola Messina, Fabrizio Falchi, Claudio Gennaro, and Giuseppe Amato. Virtual to real adaptation of pedestrian detectors. *Sensors*, 20(18):5250, 2020.

[49] Claudiu Cobârzan, Klaus Schoeffmann, Werner Bailer, Wolfgang Hürst, Adam Blažek, Jakub Lokoč, Stefanos Vrochidis, Kai Uwe Barthel, and Luca Rossetto. Interactive video search tools: a detailed analysis of the video browser showdown 2015. *Multimedia Tools and Applications*, 76(4):5539–5571, 2017.

[50] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[51] Bo Dai, Yuqi Zhang, and Dahua Lin. Detecting visual relationships with deep relational networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3298–3308. IEEE, 2017.

[52] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *arXiv preprint arXiv:2106.04803*, 2021.

[53] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021.

[54] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)*, 40(2):1–60, 2008.

[55] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[57] Jia Deng, Alexander C Berg, and Li Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *CVPR 2011*, pages 785–792. IEEE, 2011.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics, 2019.

[59] Dimiter Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. Task 6 at semeval-2021: Detection of persuasion techniques in texts and images. In *In Proceedings of the 15th International Workshop on Semantic Evaluation*, 2021.

[60] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. *arXiv preprint arXiv:2007.11498*, 2020.

[61] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[62] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[63] Aviv Eisenschtat and Lior Wolf. Linking image and text with 2-way nets. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 4601–4611, 2017.

[64] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[65] Fartash Faghri, David J. Fleet, Jamie Ryan Kiros, and Sanja Fidler. VSE++: improving visual-semantic embeddings with hard negatives. In *BMVC 2018*, page 12. BMVA Press, 2018.

[66] Christiane Fellbaum. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer, 2010.

[67] François Fleuret, Ting Li, Charles Dubout, Emma K Wampler, Steven Yantis, and Donald Geman. Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences*, 108 (43):17621–17625, 2011.

[68] Andrea Frome, Greg S Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: a deep visual-semantic embedding model. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 2121–2129, 2013.

[69] Christina M Funke, Judy Borowski, Karolina Stosio, Wieland Brendel, Thomas SA Wallis, and Matthias Bethge. Five points to check when comparing visual perception in humans and machines. *Journal of Vision*, 21(3):16–16, 2021.

[70] Claudio Gennaro, Giuseppe Amato, Paolo Bolettieri, and Pasquale Savino. An approach to content-based image retrieval based on the lucene search engine library. In *International Conference on Theory and Practice of Digital Libraries*, pages 55–66. Springer, 2010.

[71] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[72] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[73] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *arXiv preprint arXiv:1610.07940*, 2016.

[74] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254, 2017.

[75] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[76] Chunbin Gu, Jiajun Bu, Xixi Zhou, Chengwei Yao, Dongfang Ma, Zhi Yu, and Xifeng Yan. Cross-modal image retrieval with deep mutual information maximization. *Neurocomputing*, 2022.

[77] Jiuxiang Gu, Jianfei Cai, Shafiq R Joty, Li Niu, and Gang Wang. Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 7181–7189, 2018.

[78] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021.

[79] Yawen Guo, Hui Yuan, and Kun Zhang. Associating images with sentences using recurrent canonical correlation analysis. *Applied Sciences*, 10(16):5516, 2020.

[80] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE CVPR*, pages 770–778, 2016.

[81] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

[82] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[83] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[84] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proc. of the IEEE International Conference on Computer Vision*, pages 804–813, 2017.

[85] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[86] Feiran Huang, Xiaoming Zhang, Zhonghua Zhao, and Zhoujun Li. Bi-directional spatial-semantic attention networks for image-text matching. *IEEE Transactions on Image Processing*, 28(4):2008–2020, 2018.

[87] Yan Huang and Liang Wang. Acmm: Aligned cross-modal memory for few-shot image and sentence matching. In *Proc. of the IEEE International Conference on Computer Vision*, pages 5774–5783, 2019.

[88] Yan Huang, Wei Wang, and Liang Wang. Instance-aware image and sentence matching with selective multimodal lstm. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2310–2318, 2017.

[89] Yan Huang, Qi Wu, Chunfeng Song, and Liang Wang. Learning semantic concepts and order for image and sentence matching. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6163–6171, 2018.

[90] Yan Huang, Qi Wu, Wei Wang, and Liang Wang. Image and sentence matching via semantic concepts and order learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[91] Zhicheng Huang, Zhaoyang Zeng, Bei Liu, Dongmei Fu, and Jianlong Fu. Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. *arXiv preprint arXiv:2004.00849*, 2020.

[92] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43, 2008.

[93] Syed Sameed Husain and Miroslaw Bober. Improving large-scale image retrieval through robust aggregation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1783–1796, 2016.

[94] Hervé Jégou and Andrew Zisserman. Triangulation embedding and democratic aggregation for image search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3310–3317, 2014.

[95] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008.

[96] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010.

[97] Zhong Ji, Haoran Wang, Jungong Han, and Yanwei Pang. Saliency-guided attention network for image-sentence matching. In *Proc. of the IEEE International Conference on Computer Vision*, pages 5754–5763, 2019.

[98] Zhong Ji, Zhigang Lin, Haoran Wang, and Yuqing He. Multi-modal memory enhancement attention network for image-text matching. *IEEE Access*, 8:38438–38447, 2020.

[99] Zhong Ji, Haoran Wang, Jungong Han, and Yanwei Pang. Sman: Stacked multimodal attention network for cross-modal image-text retrieval. *IEEE Transactions on Cybernetics*, 2020.

[100] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

[101] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[102] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015.

[103] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of IEEE CVPR*, pages 2901–2910, 2017.

[104] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, 2017.

[105] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of IEEE CVPR*, pages 2989–2998, 2017.

[106] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[107] Samira Ebrahimi Kahou, Adam Atkinson, Vincent Michalski, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *CoRR*, abs/1710.07300, 2017. URL `http://arxiv.org/abs/1710.07300`.

[108] Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B Issa, and James J DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature neuroscience*, 22(6):974–983, 2019.

[109] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[110] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[111] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[112] Junkyung Kim, Matthew Ricci, and Thomas Serre. Not-so-CLEVR: Visual relations strain feedforward neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[113] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

[114] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[115] Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. Associating neural word embeddings with deep image representations using fisher vectors. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4437–4446, 2015.

[116] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1), 2017.

[117] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[118] Jonas Kubilius, Stefania Bracci, and Hans P Op de Beeck. Deep neural networks as a computational model for human shape sensitivity. *PLoS computational biology*, 12(4):e1004896, 2016.

[119] Jonas Kubilius, Martin Schrimpf, Aran Nayebi, Daniel Bear, Daniel LK Yamins, and James J DiCarlo. Cornet: Modeling the neural mechanisms of core object recognition. *BioRxiv*, page 408385, 2018.

[120] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *CoRR*, abs/1811.00982, 2018. URL `http://arxiv.org/abs/1811.00982`.

[121] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[122] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 201–216, 2018.

[123] Kuang-Huei Lee, Hamid Palangi, Xi Chen, Houdong Hu, and Jianfeng Gao. Learning visual relation priors for image-text matching and image captioning with neural scene graph generators. *arXiv preprint arXiv:1909.09953*, 2019.

[124] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[125] Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. Visual semantic reasoning for image-text matching. In *Proc. of the IEEE International Conference on Computer Vision*, pages 4654–4662, 2019.

[126] Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. Relation-aware graph attention network for visual question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10313–10322, 2019.

[127] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[128] Xiangyang Li and Shuqiang Jiang. Know more say less: Image captioning based on scene graphs. *IEEE Transactions on Multimedia*, 21(8):2117–2130, 2019.

[129] Xiangyang Li and Shuqiang Jiang. Know more say less: Image captioning based on scene graphs. *IEEE Transactions on Multimedia*, 21(8):2117–2130, 2019.

[130] Xirong Li, Chaoxi Xu, Gang Yang, Zhineng Chen, and Jianfeng Dong. W2vv++ fully deep learning for ad-hoc video search. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1786–1794, 2019.

[131] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[132] Yikang Li, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. Factorizable net: an efficient subgraph-based framework for scene graph generation. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 335–351, 2018.

[133] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[134] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[135] Xiao Lin and Devi Parikh. Leveraging visual question answering for image-caption ranking. In *European Conference on Computer Vision*, pages 261–277. Springer, 2016.

[136] Chunxiao Liu, Zhendong Mao, An-An Liu, Tianzhu Zhang, Bin Wang, and Yongdong Zhang. Focus your attention: A bidirectional focal attention network for image-text matching. In *Proc. of the 27th ACM International Conference on Multimedia*, pages 3–11, 2019.

[137] Chunxiao Liu, Zhendong Mao, Tianzhu Zhang, Hongtao Xie, Bin Wang, and Yongdong Zhang. Graph structured network for image-text matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10921–10930, 2020.

[138] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*, pages 730–734. IEEE, 2015.

[139] Yu Liu, Yanming Guo, Erwin M Bakker, and Michael S Lew. Learning a recurrent residual fusion network for multimodal matching. In *Proc. of the IEEE International Conference on Computer Vision*, pages 4107–4116, 2017.

[140] Jakub Lokoč, Werner Bailer, Klaus Schoeffmann, Bernd Münzer, and George Awad. On influential trends in interactive video retrieval: video browser showdown 2015–2017. *IEEE Transactions on Multimedia*, 20(12): 3361–3376, 2018.

[141] Jakub Lokoč, Patrik Veselỳ, František Mejzlík, Gregor Kovalčík, Tomáš Souček, Luca Rossetto, Klaus Schoeffmann, Werner Bailer, Cathal Gurrin, Loris Sauter, et al. Is the reign of interactive search eternal? findings from the video browser showdown 2020. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(3):1–26, 2021.

[142] Teng Long, Pascal Mettes, Heng Tao Shen, and Cees GM Snoek. Searching for actions on the hyperbole. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1141–1150, 2020.

[143] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[144] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 2016.

[145] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23, 2019.

[146] Pan Lu, Lei Ji, Wei Zhang, Nan Duan, Ming Zhou, and Jianyong Wang. R-vqa: Learning visual relation facts with semantic attention for visual question answering. In *SIGKDD 2018*, 2018.

[147] Xiaopeng Lu, Tiancheng Zhao, and Kyusong Lee. Visualsparta: An embarrassingly simple approach to large-scale text-to-image search with weighted bag-of-words. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5020–5029, 2021.

[148] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of clip for end to end video clip retrieval. *arXiv preprint arXiv:2104.08860*, 2021.

[149] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2018.

[150] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1682–1690. Curran Associates, Inc., 2014.

[151] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of IEEE CPVR*, pages 4942–4950, 2018.

[152] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[153] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 378–383. IEEE, 2016.

[154] Massimo Melucci. On rank correlation in information retrieval evaluation. *SIGIR Forum*, 41(1):18–33, June 2007. ISSN 0163-5840. doi: 10.1145/1273221.1273223.

[155] Nicola Messina. Relational visual-textual information retrieval. In *International Conference on Similarity Search and Applications*, pages 405–411. Springer, 2020.

[156] Nicola Messina, Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. Learning relationship-aware visual features. In *Computer Vision – ECCV 2018 Workshops*, pages 486–501, Cham, 2018. Springer International Publishing.

[157] Nicola Messina, Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. Testing deep neural networks on the same-different task. In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE, 2019.

[158] Nicola Messina, Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. Learning visual features for relational cbir. *International Journal of Multimedia Information Retrieval*, 9(2):113–124, 2020.

[159] Nicola Messina, Giuseppe Amato, and Fabrizio Falchi. Re-implementing and extending relation network for r-cbir. In *Italian Research Conference on Digital Libraries*, pages 82–92. Springer, 2020.

[160] Nicola Messina, Giuseppe Amato, Fabio Carrara, Claudio Gennaro, and Fabrizio Falchi. Solving the same-different task with convolutional neural networks. *Pattern Recognition Letters*, 143:75–80, 2021.

[161] Nicola Messina, Giuseppe Amato, Andrea Esuli, Fabrizio Falchi, Claudio Gennaro, and Stéphane Marchand-Maillet. Fine-grained visual textual alignment for cross-modal retrieval using transformer encoders. *To appear in Trans on Multimedia Computing Communications and Applications (TOMM)*, 2021.

[162] Nicola Messina, Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Stéphane Marchand-Maillet. Towards efficient cross-modal visual textual retrieval using transformer-encoder deep features. In *18th International Conference on Content-Based Multimedia Indexing, CBMI 2021, Lille, France, June 28-30, 2021*, pages 1–6. IEEE, 2021. doi: 10.1109/CBMI50038.2021.9461890. URL https://doi.org/10.1109/CBMI50038.2021.9461890.

[163] Nicola Messina, Fabrizio Falchi, Andrea Esuli, and Giuseppe Amato. Transformer reasoning network for image-text matching and retrieval. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5222–5229. IEEE, 2021.

[164] Nicola Messina, Fabrizio Falchi, Claudio Gennaro, and Giuseppe Amato. Aimh at semeval-2021 task 6: multimodal classification using an ensemble of transformer models. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 1020–1026, 2021.

[165] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013*, 2013.

[166] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[167] Jerome L Myers, Arnold Well, and Robert Frederick Lorch. *Research design and statistical analysis*. Routledge, 2010.

[168] Pradyumna Narayana, Aniket Pednekar, Abishek Krishnamoorthy, Kazoo Sone, and Sugato Basu. Huse: Hierarchical universal semantic embeddings. *arXiv preprint arXiv:1911.05978*, 2019.

[169] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[170] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[171] Vicente Ordonez, Girish Kulkarni, and Tamara Berg. Im2text: Describing images using 1 million captioned photographs. *Advances in neural information processing systems*, 24:1143–1151, 2011.

[172] Zarana Parekh, Jason Baldridge, Daniel Cer, Austin Waters, and Yinfei Yang. Crisscrossed captions: Extended intramodal and intermodal semantic similarity judgments for ms-coco. *arXiv preprint arXiv:2004.15020*, 2020.

[173] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[174] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. *CoRR*, abs/1709.07871, 2017. URL http://arxiv.org/abs/1709.07871.

[175] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3384–3391. IEEE, 2010.

[176] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision*, pages 143–156. Springer, 2010.

[177] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[178] Julia Peyre, Ivan Laptev, Cordelia Schmid, and Josef Sivic. Weakly-supervised learning of visual relations. In *ICCV 2017- International Conference on Computer Vision 2017*, Venice, Italy, October 2017. URL https://hal.archives-ouvertes.fr/hal-01576035.

[179] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[180] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[181] Jesús Andrés Portillo-Quintero, José Carlos Ortiz-Bayliss, and Hugo Terashima-Marín. A straightforward framework for video retrieval using clip. In *Mexican Conference on Pattern Recognition*, pages 3–12. Springer, 2021.

[182] Guillermo Puebla and Jeffrey S Bowers. Can deep convolutional neural networks learn same-different relations? *bioRxiv*, 2021.

[183] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.

[184] Di Qi, Lin Su, Jia Song, Edward Cui, Taroon Bharti, and Arun Sacheti. Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *arXiv preprint arXiv:2001.07966*, 2020.

[185] Mengshi Qi, Weijian Li, Zhengyuan Yang, Yunhong Wang, and Jiebo Luo. Attentive relational networks for mapping images to scene graphs. *CoRR*, abs/1811.10696, 2018. URL http://arxiv.org/abs/1811.10696.

[186] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[187] Leigang Qu, Meng Liu, Da Cao, Liqiang Nie, and Qi Tian. Context-aware multi-view summarization network for image-text matching. In *Proc. of the 28th ACM International Conference on Multimedia*, pages 1047–1055, 2020.

[188] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5706–5715, 2018.

[189] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

[190] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

[191] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[192] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[193] David Raposo, Adam Santoro, David G. T. Barrett, Razvan Pascanu, Timothy P. Lillicrap, and Peter W. Battaglia. Discovering objects and their relations from entangled scene representations. *CoRR*, abs/1702.05068, 2017. URL http://arxiv.org/abs/1702.05068.

[194] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*, 4(3):251–258, 2016.

[195] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[196] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2953–2961. Curran Associates, Inc., 2015.

[197] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[198] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[199] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5107–5116, 2019.

[200] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2008.04.004. 7th IAPR-TC15 Workshop on Graph-based Representations (GbR 2007).

[201] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[202] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[203] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3859–3869, 2017.

[204] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[205] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4967–4976. Curran Associates, Inc., 2017.

[206] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. pages 4967–4976, 2017.

[207] Nikolaos Sarafianos, Xiang Xu, and Ioannis A Kakadiaris. Adversarial representation learning for text-to-image matching. In *Proc. of the IEEE International Conference on Computer Vision*, pages 5814–5824, 2019.

[208] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102, 2008.

[209] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[210] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.

[211] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, IEEE International Conference on*, volume 3, pages 1470–1470. IEEE Computer Society, 2003.

[212] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

[213] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

[214] Sebastian Stabinger, Antonio Rodríguez-Sánchez, and Justus Piater. 25 years of cnns: Can we compare to human abstraction capabilities? In *International Conference on Artificial Neural Networks*, pages 380–387. Springer, 2016.

[215] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*, 2020.

[216] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[217] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[218] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of IEEE CVPR*, pages 1–9, 2015.

[219] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. In *ICLR 2016*, 2016.

[220] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[221] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[222] Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SkgKO0EtvS`.

[223] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *4th International Conference on Learning Representations, ICLR*, 2016.

[224] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[225] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166, 2014.

[226] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[227] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1386–1393, 2014.

[228] Shuhui Wang, Yangyu Chen, Junbao Zhuo, Qingming Huang, and Qi Tian. Joint global and co-attentive representation learning for image-sentence retrieval. In *Proc. of the 26th ACM international conference on Multimedia*, pages 1398–1406, 2018.

[229] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[230] Yaxiong Wang, Hao Yang, Xueming Qian, Lin Ma, Jing Lu, Biao Li, and Xin Fan. Position focused attention network for image-text matching. *arXiv preprint arXiv:1907.09748*, 2019.

[231] Kaimin Wei and Zhibo Zhou. Adversarial attentive multi-modal embedding learning for image-text matching. *IEEE Access*, 2020.

[232] Xi Wei, Tianzhu Zhang, Yan Li, Yongdong Zhang, and Feng Wu. Multi-modality cross attention network for image and sentence matching. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10941–10950, 2020.

[233] Maurice Weiler and Gabriele Cesa. General E(2)-Equivariant Steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[234] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016.

[235] Yiling Wu, Shuhui Wang, Guoli Song, and Qingming Huang. Learning fragment self-attention embeddings for image-text matching. In *Proc. of the 27th ACM International Conference on Multimedia*, pages 2088–2096, 2019.

[236] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[237] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[238] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rJxbJeHFPS`.

[239] Xing Xu, Tan Wang, Yang Yang, Lin Zuo, Fumin Shen, and Heng Tao Shen. Cross-modal attention with semantic consistence for image-text matching. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[240] Guangyu Robert Yang, Igor Ganichev, Xiao-Jing Wang, Jonathon Shlens, and David Sussillo. A dataset and architecture for visual reasoning with a working memory. In *European Conference on Computer Vision*, pages 729–745. Springer, 2018.

[241] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proc. of the European conference on computer vision (ECCV)*, pages 670–685, 2018.

[242] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph R-CNN for scene graph generation. *CoRR*, abs/1808.00191, 2018. URL `http://arxiv.org/abs/1808.00191`.

[243] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019.

[244] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.

[245] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. Stacked attention networks for image question answering. *CoRR*, abs/1511.02274, 2015. URL `http://arxiv.org/abs/1511.02274`.

[246] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proc. of the European conference on computer vision (ECCV)*, pages 684–699, 2018.

[247] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. *CoRR*, abs/1809.07041, 2018. URL `http://arxiv.org/abs/1809.07041`.

[248] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

[249] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[250] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[251] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

[252] ChengXiang Zhai. Interactive information retrieval: Models, algorithms, and evaluation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2444–2447, 2020.

[253] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. RAVEN: A Dataset for Relational and Analogical Visual rEasoNing. *arXiv e-prints*, art. arXiv:1903.02741, Mar 2019.

[254] Ji Zhang, Yannis Kalantidis, Marcus Rohrbach, Manohar Paluri, Ahmed M. Elgammal, and Mohamed Elhoseiny. Large-scale visual relationship understanding. *CoRR*, abs/1804.10660, 2018. URL `http://arxiv.org/abs/1804.10660`.

[255] Jin Zhang, Xiaohai He, Linbo Qing, Luping Liu, and Xiaodong Luo. Cross-modal multi-relationship aware reasoning for image-text matching. *Multimedia Tools and Applications*, pages 1–23, 2021.

[256] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple baseline for visual question answering. *CoRR*, abs/1512.02167, 2015. URL `http://arxiv.org/abs/1512.02167`.

[257] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[258] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.