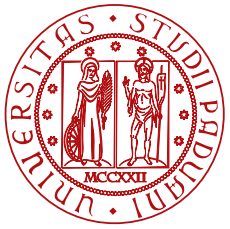


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITY OF PADUA
DEPARTMENT OF INFORMATION ENGINEERING
MASTER THESIS IN ICT FOR INTERNET AND MULTIMEDIA

Semantic Aware Image Search with Scene Knowledge Graphs

MASTER CANDIDATE

Giacomo Loreggia

Student ID 2005798

SUPERVISOR

Prof. Gianmaria Silvello

University of Padua

CO-SUPERVISOR

Prof. Matteo Lissandrini

University of Aalborg

Padua, 17/10/2022

ACADEMIC YEAR
2021/2022

To all my family,
and friends

Abstract

Scene Graphs (SGs) are Knowledge Graphs representing the contents of an image in terms of its elements, e.g., people, objects, and attributes, as well as their relationships. Hence, they can capture the scene's structural and semantic organization and express it in a machine-readable model. Thanks to their expressive power, SGs have been applied in important Image Processing tasks such as Image Captioning, Visual Question Answering, and Image Search. In this work, we focus on Image Search, which, given a query, is the task of retrieving the images that best match the text given as input. The task of determining which images are the best matches for a given query becomes more and more challenging as the details and the complexity of the query increase. While several approaches have been proposed to tackle the Image-Search task by adopting pre-trained language models, the opportunity to use both a Scene Graph and a language model has not been studied yet. In this work, we employ an SG representation and a pre-trained language model with the purpose of improving the Image Search performance when dealing with complex textual queries.

Sommario

I grafi di scena (Scene Graphs, SGs) sono grafi di conoscenza (Knowledge Graphs, KGs) che rappresentano il contenuto dell'immagine attraverso i suoi elementi, come per esempio persone, oggetti, attributi, assieme anche alle rispettive relazioni. Per questo, essi riescono a catturare l'organizzazione semantica e strutturale della scena e ad esprimerla in un formato adatto per una macchina. Grazie alla loro abilità espressiva, gli SG sono stati applicati in importanti ambiti di ricerca dell'elaborazione delle immagini, come la descrizione didascalica automatica delle immagini, risposta a domande che comprendono una componente visuale, e ricerca di immagini. In questo lavoro, ci concentriamo sulla ricerca di immagini, che data una richiesta testuale, è il compito di trovare le immagini che meglio si associano a quel testo. Questo problema diventa più difficile se la richiesta contiene più dettagli ed è quindi più complessa. Mentre diversi approcci sono stati proposti per risolvere il problema della ricerca delle immagini utilizzando modelli di linguaggi pre-allenati, l'opportunità di utilizzare entrambi SG e un modello di linguaggio non è ancora stata studiata. In questo lavoro, implementiamo entrambi questi elementi con il fine di migliorare le prestazioni nell'ambito della ricerca di immagini quando si utilizzato richieste testuali complesse.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

DL Deep Learning

SGG Scene Graph Generation

KG Knowledge Graph

SG Scene Graph

NN Neural Network

FFNN Feed-Forward Neural Network

IS Image Search

ISS Image Search System

NLP Natural Language Processing

CV Computer Vision

R Recall

CLIP Contrastive Language-Image Pre-train

CLIGT Contrastive Language-Image-Graph Train

1

Introduction

In the last years, Deep-Learning (DL) for Image Processing has attracted considerable attention [3, 19, 15, 48, 7, 36]. Among these approaches, Convolutional Neural Networks (CNNs) have been offering the possibility to process pictures to capture meaningful patterns inside them. CNNs paved the way to more complex Deep Learning architectures capable of improving image understanding, thus enabling different forms of image reasoning [7, 12, 36]. Thanks to these technologies, we have witnessed substantial advancements in the performance of models for Visual Question Answering [1], Image Retrieval [42], Image Captioning [16], and Image Generation [33]. Among these, in the literature, *Image Retrieval* refers to the general task of retrieving an image, no matter what input is used [3, 18, 42, 32]. In this work, we will refer to *Image Search* as text-based Image Retrieval, i.e., the task of using a text query (e.g., a keywords query) to retrieve a set of images matching its intent (Figure 1.1). We are going to discuss this formally in Section 2.2.

On the other hand, we have also witnessed an increasing adoption of the Knowledge Graph (KG) model [10] when in need of a data model able to represent data with a rich semantic structure. A KG is a semantic network, i.e., a graph, composed of entities, relationships, and attributes. In its most common form, it represents data as subject-predicate-object triples. Entities can be both real-world objects as well as abstract concepts.

Scene Graphs (SGs) are Knowledge Graphs representations of the content of an image describing both the entities present in the image as well as the interactions between them [3]. In Figure 1.1.b we see 2 examples of Scene Graphs created from 2 images. Due to their semantic ability to describe images, SGs are becoming more and more popular.

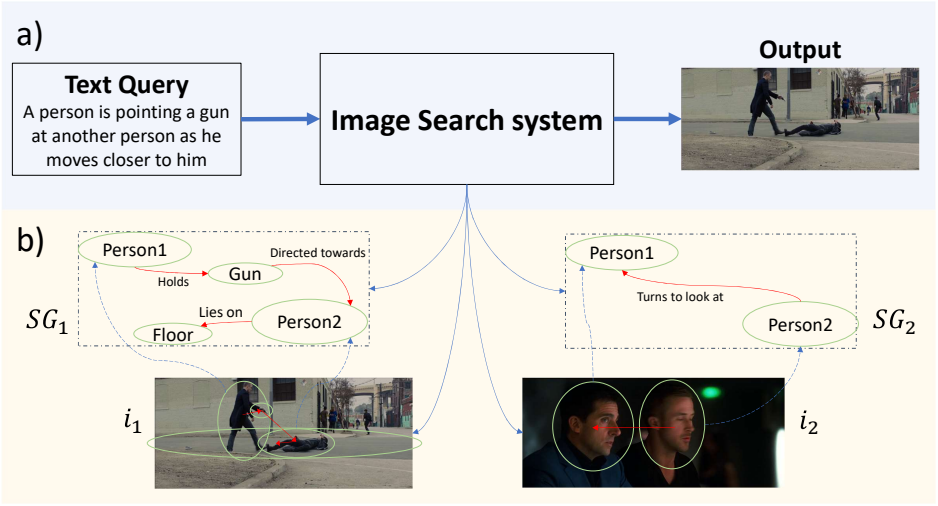


Figure 1.1: Example of Image Search by text comparing also Scene Graphs. Text query and images taken from VisualCOMET dataset [29]. a) depicts just the general Image Search flow. b) shows 2 examples of Scene Graphs associated with a collection of 2 images and their connection with the Image Search task.

Researchers have implemented them for several tasks, such as Scene Graphs-based Image Retrieval [18], Image Captioning [15], and Visual Question Answering [3].

In this work, we implement SGs in the context of Image Search. As we will see better in Section 2.4, researchers have already obtained good results on the Image Search task. However, the performance can still be improved, in particular when complex and richer-in-details queries are used. At the same time, there is no work involving both SGs and language models to solve Image Search. Hence, we hypothesize that by integrating these image representations into the framework, we can improve the performance of an Image Search system. In particular, we delve into one of the techniques developed very recently, that is, Contrastive Language-Image Pre-train [32]: the targeted neural network is firstly pre-trained on a dataset composed of images and associated text to shape a multi-modal embedding space comprising both images and text; the more the images and the text are related, the higher their similarity [32].

2

Background

In this section we briefly describe the fundamental concepts of our research, that is: Neural Networks (NNs), Image Search (IS), Scene Graphs (SGs) and the evaluation metrics we use in this work. We are also mentioning the state of the art.

2.1 Neural Networks

The history of the Neural Networks is quite recent, as it dates back just a few decades. It actually started with a neuroscience discover, the one by Hodgkin and Huxley, who developed a biophysical model to describe neurons' action potential, in 1952 [14].

Then, Frank Rosenblatt, in 1958, proposed the first algorithm to train a single neuron, called the Perceptron model [38]. We are going to see this in detail in a bit.

From that time on, both neuroscientists and computer scientists worked on the NNs development, reaching several milestones along the decades: from fundamental discovers such as the backpropagation algorithm [39], the first Recurrent NN [4], the concept of the Convolutional NN [20] to more recent and complex findings such as Generative Adversarial Networks [8] and Transformers [45].

In this section we describe the concept of Neural Network in a general setup. This is meant to be a very brief overview and to help us show the basic idea behind a complex neural network, we start from the basic piece, that is, the Rosenblatt's Perceptron model. Then, we delve a bit deeper on the Feed-Forward Neural Network, the model we will implement in our context .

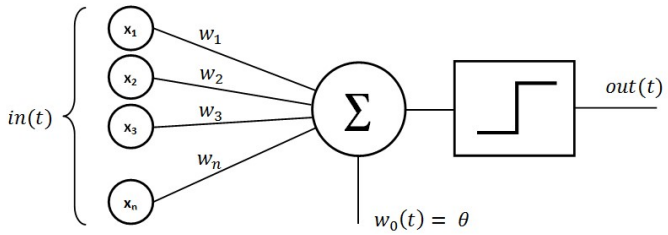


Figure 2.1: Rosenblatt Perceptron model

2.1.1 Perceptron Model

In Figure 2.1 the Rosenblatt's Perceptron model is depicted. At time t a series of n inputs $x_1, x_2, ..., x_n$ are fed into the neuron depicted with a circle containing the sigma sign, meaning the sum operation. The inputs are weighed according to the values $w_1, w_2, ..., w_n$ and biased with $w_0(t)$. The output of the sum is fed into a so-called non-linear Activation Function, here denoted as g(), to produce the final output $out(t)$.

Hence, $out(t)$ depends on $w_1, w_2, ..., w_n$ and on the input $in(t) = (x_1, x_2, ..., x_n)$ according to the following rule:

$$out(t) = g(\sum_{i=1}^{n} w_i x_i + w_0). \tag{2.1}$$

According to Rosenblatt, the learning consists in changing the bias w_0 and the weights w_i, $i = 1, ..., n$ so that the overall function returns the wanted result.

In particular, the update rule is the following one:

$$\Delta w = \eta(target - out)x, \tag{2.2}$$

where $target$ is the ideal value, and out is the value computed by the perceptron as seen in the Equation 2.1.

2.1.2 Feed-Forward Neural Networks

Roughly speaking, a Feed-Forward Neural Network (FFNN) is the scaled version of a Perceptron, that is why this type of network is also called multi-layer Perceptron.

In Figure 2.2 we see an example of FFNN. The *input* depicted here is the same of

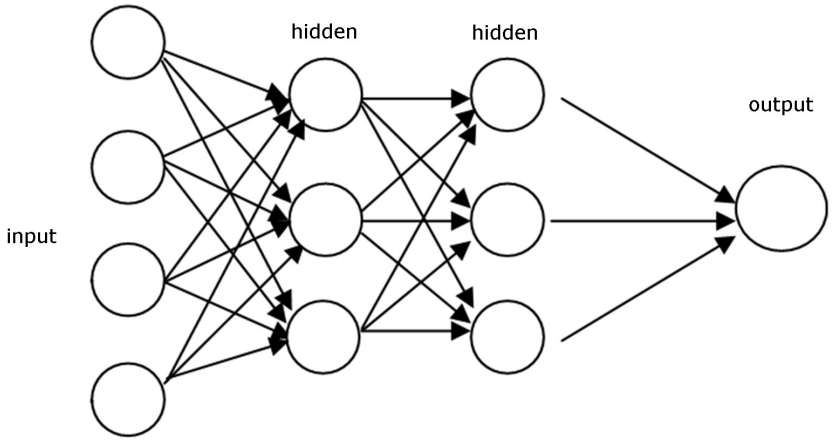


Figure 2.2: Example of Feed-Forward Neural Network

the one depicted in Figure 2.1 as $in(t)$. However, in this model, more other nodes are present, and in particular we identify the so-called hidden layers, the two intermediate ones, and the output layer, the latter formed just by one neuron in the case of the figure.

The hidden layers' task is to learn intermediate characteristics, also called features or patterns, that may be useful in computing the output.

A complete overview of an FFNN train is out of scope of this work. Anyway, we just mention that the train of such a network is way more complex that the one of the perceptron, and in this context the already mentioned Backpropagation algorithm comes handy in propagating the error to all the weights of the network. Together with this, the concept of optimizer appears, which is the algorithm utilized to actually update the weights once the error has been back-propagated.

2.2 Image Search

In this work, we tackle the problem of Image Search (IS), i.e., text-based Image Retrieval. It refers to an interdisciplinary task that involves both Natural Language Processing (NLP) and Image Processing, in that it involves textual queries and a collection of images. In Figure 1.1.a we see an example of IS flow. In the case of the figure, the collection of images is made of i_1 and i_2 and the query is *A person is pointing a gun at another person as he moves closer to him.*

Formally, given a word dictionary $\mathcal{D}$, $\mathcal{D}^*$ denotes sequences of any length that can be built with words belonging to $\mathcal{D}$. Hence, a query is a text sequence $q = (w_1, w_2, w_3, ...) \in \mathcal{D}^*$. The collection of images is denoted as $\mathcal{I}$, and, in the IS task, the goal is to find the

so-called relevant images $\mathcal{I}_{r,q} \subseteq \mathcal{I}$, i.e., the ones that best match the query q. An Image Search system (ISS) is a system that solves an Image Search task. When determining the expected results of an ISS, we may look at the IS task from 2 different point of views: either we consider it a binary classification problem or a relevance prediction problem.

In the former, we just wonder whether the images recommended by the ISS are relevant or not, whereas in the latter, we consider the image-relevance degree, i.e., how much an image is relevant. Given this, we are going to first define the binary-classification IS problem, which we call simply *Binary Image Search*; then we define the *Ranked Image Search*.

Problem 2.2.1 (Binary Image Search) *Given a set of images $\mathcal{I}$, a vocabulary $\mathcal{D}$ and a query $q = (w_1, w_2, ...w_l) \in \mathcal{D}^*$, where $\mathcal{D}^*$ is the set of sequences of any length built from words $w \in \mathcal{D}$, the Binary Image Search problem is the classification problem that requires to determine a function $f : \mathcal{I} \times \mathcal{D}^* \rightarrow \{\bot, \top\}$, such that $\forall i \in \mathcal{I}_{r,q} \subseteq \mathcal{I}$, $f(i, q) = \top$ if image i is a relevant match for the query q, and $f(i, q) = \bot$ otherwise.*

Following what we mentioned earlier, in this definition we are just interested on whether the relevant images are actually classified as relevant or not according to the function f. The set $\{\bot, \top\}$ is indeed composed by only two elements, which forces the classification to be binary. Moreover, $\mathcal{I}_{r,q}$ can be of any size, which means that there may be many pictures that can be judged somewhat relevant to a query q. We will come back to this aspect later, when we define the evaluation metrics in use in this work.

To study the *Ranked Image Search* problem, the codomain of the previous function f must express the degree of relevance of the images with respect to the query, i.e., a relevance score, usually in $[0, 1] \subset \mathbb{R}$. In some cases, this can also be seen as the probability $P_q(i)$ that i is a relevant image for q. Hence, the definition of the *Ranked Image Search* problem is the following:

Problem 2.2.2 (Ranked Image Search) *Given a set of images $\mathcal{I}$, a vocabulary $\mathcal{D}$ and a query $q = (w_1, w_2, ...w_l) \in \mathcal{D}^*$, where $\mathcal{D}^*$ is the set of sequences of any length built from words $w \in \mathcal{D}$, the Ranked Image Search Problem is the relevance prediction problem that requires to determine a function $\hat{f} : \mathcal{I} \times \mathcal{D}^* \rightarrow [0, 1]$, such that the output of $\hat{f}(i, q)$ is 0 if i is irrelevant for the query q and given i_1, i_2, $\hat{f}(i_1, q) \geq \hat{f}(i_2, q)$ if and only if i_1 is a better match than i_2 for the query q.*

2.3 Scene Graphs

We presume that by integrating the SGs in the process, the overall performance of the system can be improved. To define the Scene Graphs, we take inspiration from the definition given in the SG Survey by Chang et al. [3]:

Definition 2.3.1 (Scene Graph) *A Scene Graph is a directed graph, defined as* $G = (O, R, E)$ *where:*

- $O = \{o_1, o_2, ..., o_N\}$ *is the set of objects detected in the images; each object has the form* $o_i = (c_i, A_i)$ *where* c_i *and* A_i *represent the category and the attribute of the object respectively;*
- R *is the set of relations;*
- $E \subseteq O \times R \times O$ *set of directed edges that expresses the relationship between objects.*

2.4 State of The Art

In this work we advance the state-of-the-art by studying how to improve the performance of a text-image matching model by incorporating the information of a scene graph representation for an image.

2.4.1 Text-Image Matching

Recently, there has been an increased interest in cross-modal retrieval, which takes one type of data as the query and retrieves relevant data from another type. Text and image matching is an example of cross-modal retrieval model where a query in text format retrieves a set of images relevant to that query. One of the early approaches for such a model relied on projecting the images and text models representations into a shared, embedded space for similarity computation or learning a matching score [6, 21, 34]. Another line of research adopts pre-training techniques applied in Computer Vision (CV) and NLP. These proposed approaches have achieved promising performance on different tasks, including text-image matching [46, 43, 22]. Contrastive Language-Image Pre-training (CLIP) [32] is among the most known example of multi-modal models for joint learning representations of images and text. CLIP achieves superior retrieval results in open-domain image-text matching tasks [41]. Text-to-image matching using the CLIP

model can be conducted by calculating the similarities in the embedding space. The main challenge of these matching approaches is extracting semantic information from the images and mapping it to the textual query. Most of the current approaches rely on detecting certain regions from the images. For instance, Faghri *et al.* [5] employed a CNN-based image encoder to extract visual features of images and an RNN-based textual module to extract those from captions. These features are then projected into the same vector space to perform the matching comparison. Although multi-modal models achieved good accuracy, previous work ignored the vital information of objects interactions within images and sentences, that led the researchers to direct their focus on extracting non-obvious information from the images. As an example, researchers showed that the relative position of detected objects within an image could be useful when matching with the caption [47]. In our research, we incorporate the semantic knowledge about the objects and their interactions in the images using generated scene graphs from the images and incorporate this scene graph information in the representation of the images in the CLIP framework.

2.4.2 Scene Graph

The scene graph was first proposed by Johnson, Justin, *et al.* for image caption and retrieval [18]. A scene graph describes objects, their attributes, and relationships in images with a graph. In follow-on work, Johnson, Justin, *et al.* used the ground-truth scene graph as the query for image retrieval [18]. However, this approach relied on human-generated scene graphs. On other direction, other attempts to use the graph structure generally to represent both the textual and visual data, such as [44]. Unlike our approach, the employed graphs in this work include no semantic relations and are not scene graphs. With the developments of scene graph generations [51, 49], the performance of many visual cross model tasks are improved, such as Visual Question Answering [13] and Image Captioning [50]. Most of these approaches benefit from the scene graph to retrieve images. For example, Yao *et al.* [50] designed a GCN-based captioning model that employs scene graph generators to propose possible connections between objects. Other methods are proposed to parse text into a scene graph [40]. These approaches discard the representations of visual relations generated from the scene graph generators, and instead implicitly infer relationships from caption data. Directly incorporating both the scene graph objects and relationships in the multi-modal (text and image) representation distinguishes our approach from this line of work.

Dataset	Descriptions	# Images	Year
VisualCOMET [1] [29]	Large-scale dataset of Visual Commonsense Graphs for reasoning about the dynamic context of static images for visual scene understanding	59,356	2020
CUHK-PEDES [23]	Images of pedestrians accompanied by textual descriptions	40,206	2017
Flowers [34]	Images of flowers with 10 descriptions for each image	8189	2016
Caltech-UCSD Birds (CUB) [34]	Images of birds with 10 descriptions for each image of them	11,788	2015
Flicker30K [30]	A wide variety of images (humans, animals, objects, scenes) in addition to 5 different descriptions for each image	31,783	2015
MSCOCO [25]	Contains 123,287 images, and each image is annotated with five text descriptions	123,287	2014

Table 2.1: Dataset for Text-Image Matching Evaluation.

Model	Descriptions	Year
CLIP [32]	Joint learning representations of image and text	2021
Data efficient CLIP (DeCLIP) [24]	An efficient CLIP with less training time and training data. Similar to CLIP, evaluated on (text, image) pair. Mostly classification accuracy	2021
GXN [9]	Generative models	2018
SCO [17]	Multi-regional multi-label CNN	2018
HM-LSTM [28]	Hierarchical multimodal LSTM embedding model	2017
VSE++ [5]	Visual-Semantic Embeddings	2017
DVSA [6]	Deep Visual-Semantic Alignments	2013

Table 2.2: Models used for Text-Image Matching Tasks.

Table 2.1 shows the most known public datasets used in the literature to evaluate the text to image matching and retrieval. Table 2.2 summarizes different models developed to improve different visual downstream tasks, including text to image matching.

2.5 Evaluation Metrics

To perform the evaluation, similarities between images and texts must be computed, and then, on these values, the metrics of interest are computed, i.e., Hits and Recall

[1] 9https://visualcomet.xyz

(R). Both these metrics are often used on binary classification problems. In the case of Image Search, they are then used when just one image is relevant. However, we will also see that with our dataset they are enough to evaluate our models, even though multiple images could be relevant in principle. Before going on, we report briefly what we mean by *recommended* images following the notation of Problem Definition 2.2.1:

$$\{f(i,q) = \top; i \in \mathcal{I}\}, \tag{2.3}$$

that is, the images classified as relevant by the ISS. Given this, in the next sections, we define Hits and Recall.

2.5.1 Hits

Hits is a binary measure, so it has been mainly used for binary classification problems. Looking at Figure 2.3, we see the possible outcomes of a binary classification. Hits counts the number of *True positives*.

We define it formally in our context, i.e., Image Search, and given the fact that it is a binary measure, we will use the notation in Problem Definition 2.2.1 and in [31].

Definition 2.5.1 (Hits) *Given a Binary Image Search task defined as in Problem Definition 2.2.1 we define the Hits measure as the number of* True Positives $TP = |\{f(i,q) = \top; i \in \mathcal{I}_{r,q}\}|$*, i.e., the number of relevant images recommended by the system.*

2.5.2 Recall

Recall is a binary classification measure as well. Roughly speaking, it expresses the fraction of relevant elements that were indeed "selected" by the system. It builds upon Hits measure.

Looking again at Figure 2.3. Recall expresses the fraction of True Positives with respect to all the Real Positives.

We again define it formally in our context and we will use the notation in Problem Definition 2.2.1 and in [31].

The Recall definition follows:

Definition 2.5.2 (Recall) *Given a Binary Image Search task defined as in Problem Definition 2.2.1 we define the Recall measure at that ratio R such that:*

$$R = \frac{TP}{RP}, \tag{2.4}$$

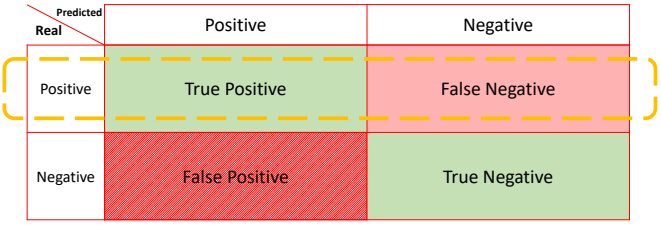


Figure 2.3: Table reporting the possible outcomes of a binary classification. We use Hits and Recall metrics, which deals with the row indicated by the dashed rectangle. Hits in particular deals with just the green square of *True Positives*.

where TP *are the* True Positives*, and* RP*,* Real Positives*, is the cardinality* $|\mathcal{I}_{r,q}|$ *of relevant images associated to query* q*.*

3 Architectures

In this chapter we show the initial architecture together with the model considered and the proposed architecture, the latter integrating the Scene Graphs into the input data.

3.1 CLIP

In this section, we delve deeper into both the approach and the model we consider, named in the same way, that is, CLIP: Contrastive Language-Image Pre-train.

In Figure 3.1, we have an idea of how CLIP works. The motivation behind the Language-Image part in the acronym is because the architecture entails an image encoder and a text encoder, which receive as input respectively an image and a text and produce in the output the associated features vector [32].

Concerning the image encoders, CLIP authors work with different model architectures [32]: they train 5 slightly modified ResNets and 3 Vision Transformers. The text encoder is a modified Transformer. According to the authors' results, the image encoder that performs better is the one reported with the code *ViT-L/14@336px*, which corresponds to the architecture *ViT-L/14* pre-trained one epoch more at a higher pixel resolution. Hence, this is the model that we are also going to use.

The encoders are jointly pre-trained so that the associated image and text features maximize the cosine similarity, and the non-associated pairs have their similarity minimized [32]. More formally, using the notation of Figure 3.1, given a batch of N $(image, text)$ positive pairs such that $I_s = (i_{s,1}, i_{s,2}, ..., i_{s,M})$ is the s-th image feature vector and $T_s = (t_{s,1}, t_{s,2}, ..., t_{s,M})$ is the associated text feature vector, the encoders are

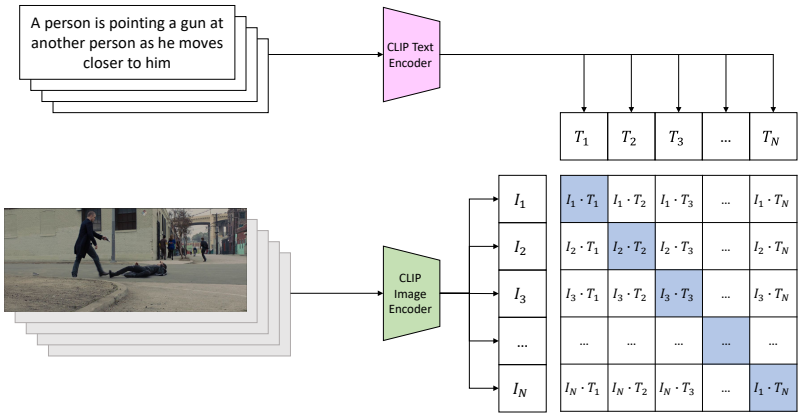


Figure 3.1: Image depicting CLIP workflow.

jointly trained to maximize the following:

$$CS(T_s, I_s) = \frac{\sum_{j=1}^{M} t_{s,j} i_{s,j}}{\sqrt{\sum_{j=1}^{M} (t_{s,j})^2 \sum_{j=1}^{M} (i_{s,j})^2}}, \tag{3.1}$$

for $s = 1, ..., N$, and also to minimize the cosine similarity between all the other possible pairs of the batch, i.e.,

$$CS(T_s, I_k) = \frac{\sum_{j=1}^{M} t_{s,j} i_{k,j}}{\sqrt{\sum_{j=1}^{M} (t_{s,j})^2 \sum_{j=1}^{M} (i_{k,j})^2}}, \tag{3.2}$$

for $s, k = 1, ..., N$, $s \neq k$. This is how Contrastive training works: in general the measure considered for associated samples is *contrasted* with the non-associated ones.

In Figure 3.1, the similarity values are represented by the matrix $N \times N$, in which the value in row s and column k correspond to the similarity value between text features T_s and image features I_k, with $s, k = 1, ..., N$. In the figure, the contrastive learning is represented by the different colours of the matrix cells: on the diagonal, that is, with $s = k$, the value are maximized, whereas all the others lying outside the diagonal are minimized.

3.2 Proposed Architecture - CLIGT

The proposed architecture comprising both CLIP encoders and the SGs is depicted in Figure 3.2. Given a text and an image, they are encoded through CLIP's blocks. At

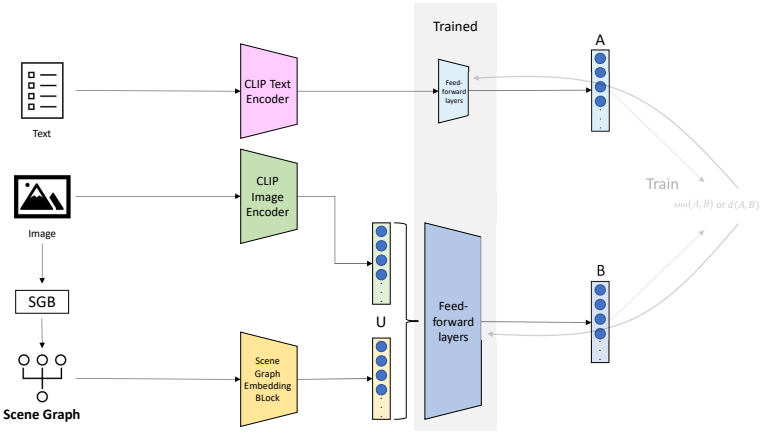


Figure 3.2: Proposed model integrating Scene Graphs.

the same time, SGB (Scene Graph Benckmark) is a code that is used to produce SGs. The obtained Scene Graphs are then put as input to a Scene Graph Embedding block, which outputs the associated embedding vector. Then, the image feature and the SG embedding are joined in a so-called *visual vector* and put as input to a FFNN, which outputs the *visual features*. Also the text feature is put as input to a FFNN.

The goal is to integrate the information coming from the SGs into the vector which also comprises the image information, thus making the associated image-text vectors closer.

We called the architecture CLIGT: Contrastive Language-Image-Graph Train. We refer to it even if, as we will se in Section 5.3.3, we also follow an approach that is not contrastive.

3.2.1 Scene Graph Embedding Block

The yellow block in Figure 3.2 is the Scene Graph Embedding block, which takes an SG as input and produces as output an embedding vector, thus projecting all the scene graphs into a vector space. In this section, we delve deeper on the internal structure of this block, which is depicted in Figure 3.3.

Given an SG, a Ristoski's RDF2Vec-like Graph Walk [37] is performed, to obtain sentence-like sequences from the SG nodes and relationships. Then, every sentence is encoded through BERT [35] model. Finally, all the sentence features are aggregated through an aggregator function, which, in our case, we chose to be the average function. The result is an SG embedding vector. Notice that not a single part of the aforementioned

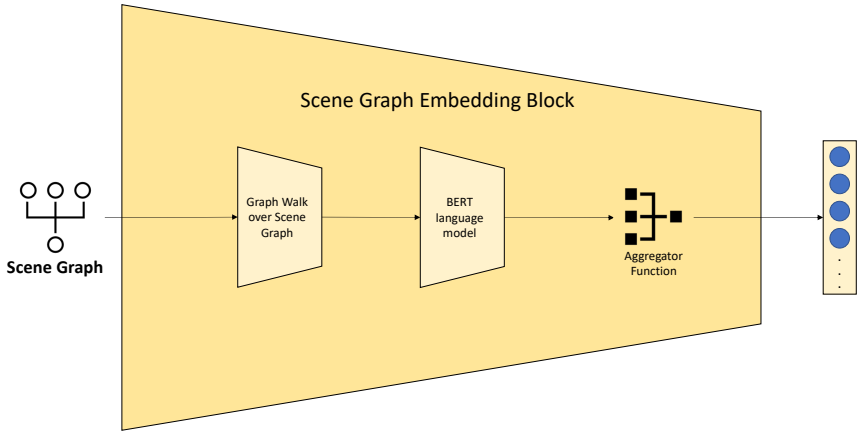


Figure 3.3: The SG Embedding Block functioning.

block needs to be trained. In fact, BERT, which is the only DL model in this block, is already trained [35]. In the next paragraph we present the Graph Walk we implemented in detail, with an example.

Graph Walk To generate the graph walks, we followed closely the related paragraph in Ristoski's RDF2Vec work [37]. We think it is then meaningful to report the paragraph:

> *In this approach, for a given graph $G = (V; E)$, for each vertex $v \in V$ we generate all graph walks P_v of depth d rooted in the vertex v. To generate the walks, we use the breadth-first algorithm. In the first iteration, the algorithm generates paths by exploring the direct outgoing edges of the root node v_r. The paths generated after the first iteration will have the following pattern $v_r \rightarrow e_{1i}$, where $i \in E(v_r)$. In the second iteration, for each of the previously explored edges the algorithm visits the connected vertices. The paths generated after the second iteration will follow the following patter $v_r \rightarrow e_{1i} \rightarrow v_{1i}$. The algorithm continues until d iterations are reached. The final set of sequences for the given graph G is the union of the sequences of all the vertices $\bigcup_{v \in V} P_v$.*

We report in Algorithm 1 the recursive algorithm utilized to generate all the graph walks starting from a root node v_r. Notice that the output returned by the algorithm does not include the source node class, which must be prepended to every output element.

Once the algorithm is executed for every node in the graph, we obtain all the graph walks, i.e., all the sentences.

Algorithm 1 Single graph walk performed on a node.

Input:

- starting node $start$
- visited nodes set $visited$
- current depth c_depth
- maximum depth m_depth

```
1:  visited.append(start)
2:  extension_paths ← []
3:  for edge ∈ start.outgoing() do
4:      dest ← edge.dest
5:      if c_depth + 2 = m_depth then
6:          extension_paths.append(concatenate(edge.label(), dest.label()))
7:      else
8:          if c_depth + 1 < m_depth then
9:              if dest ∉ visited then
10:                 dest_paths ← graph_walk(dest, visited, c_depth + 2, m_depth)
11:                 extension_paths.add_all(merge(edge.label(), dest.label(), dest_paths))
12:             end if
13:         else
14:             extension_paths.append(edge.label())
15:         end if
16:     end if
17: end for
18: return extension_paths
```

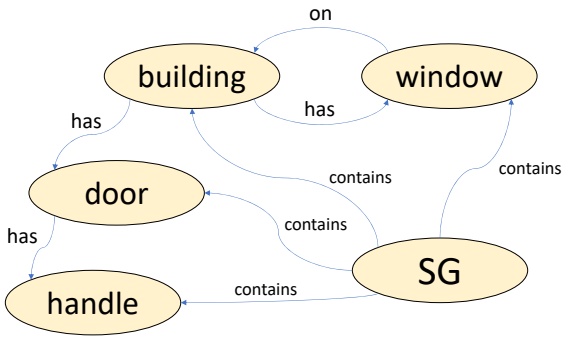


Figure 3.4: Artificial example of SG to present the output of the graph walks

As an example of the output produced, Figure 3.4 shows an artificially-built SG on which all the graph walks are retrieved. We consider here $max\_depth = 6$, which then comprises at most a path containing seven elements: the starting node plus the six consecutive elements, i.e., 3 nodes and 3 relantions, as Ristoski's paragraph states. The output list of sentences in this case is the following:

- 'window on building has door has handle'
- 'building has door has handle'
- 'door has handle'
- 'SG contains window on building has window'
- 'SG contains window on building has door'
- 'SG contains building has door has handle'
- 'SG contains door has handle'
- 'SG contains handle'

3.2.2 FFNN Structure

The architecture of the FFNNs depicted in Figure 3.2 comprises some hidden layers. The number of layers and the number of nodes per layer are hyperparameters, of which we talk in Section 5.3.5. Here we just report in Tables 3.1 and 3.2 the structure per approach.

	textual net	visual net
input	768	1536
hidden layer 1	576	512
hidden layer 2	384	960
output	192	192

Table 3.1: FFNNs architecture for the classification approach

	textual net	visual net
input	768	1536
hidden layer 1	448	384
hidden layer 2	-	960
output	512	512

Table 3.2: FFNNs architecture for the energy-based approach

4 Data

The purpose of this chapter is to present the ground-truth that we have chosen. From the previous chapter we have seen that we work with 3 types of data: images, texts and Scene Graphs. VisualCOMET dataset [29] provides the first two types, and the latters are computed through the code developed by Han *et al.* [11]. In the next sections, we first present VisualCOMET and the modifications done on it, supported by some statistics. Then we present the Scene Graphs retrieved and some of their characteristics observed.

4.1 VisualCOMET Dataset

We decided to use VisualCOMET[29] perform our experiments. The reason behind this choice is that this dataset contains more than fifty thousand images, and every image content is described by at least a couple of sentences [29]. The descriptions are indeed more than a hundred thousand. Beyond that, every image is also provided with some annotations about the intents of the people in the image and what events might have happened before and after [29]. However, we focus on the description itself, as it provides a semantic textual representation of the content.

We recall we are working on the Image Search context, so we consider every description as the query to retrieve the image it is paired with. Following the Problem 2.2.1 defined previously, given a description d_j of an image i_j, d_j can be seen as the query q and i_j can be considered as being part of the relevant images of q, i.e., $i_j \in \mathcal{I}_{r,q}$. In this way, we obtain a ground-truth dataset for our task. As a side note, we stress that what makes VisualCOMET even more suitable for our study case is that the image descriptions are enough complex to require a deeper understanding of the scene.

In studying the structure of the dataset descriptions, we discovered an important characteristic: every description contains the so-called person-groundings, which are references contained in the description to the persons in the images [29]; these references are in particular numbers.

In the example of Figure 1.1, the image i_1 description in the VisualCOMET dataset is the following one [29]:

"1 is pointing a gun at 8 as he moves closer to him".

While useful to identify the persons in the image, this is not a Natural Language query, i.e., we would never insert a sentence like this to search for image i_1. Hence, we opt for the substitution of the numbers with the simple expression *a person*.

However, this introduces another issue. First of all, if we substituted *a person* every time we found a number, the example above would become this:

"A person is pointing a gun at
a person as he moves closer to him",

which is now in natural language, but can still be improved. To do it, instead of using *a person* for the second number, we can substitute *another person*. The aforementioned example thus becomes the following sentence:

"A person is pointing a gun at
another person as he moves closer to him".

This can actually be done for every sentence that contains exactly 2 numbers associated with persons.

However, the cases in which there are 3 or more people must still be considered. These can not be modified in a suitable automatic way. To understand why, take the following clarifying example:

"1 sits on the couch with 2 and 4 on her lap".

Clearly, as we have seen for the case of 2 persons, it is not elegant to simply substitute *a person*, even more, because in this example there are 3 persons. If we substituted *a*

	Train	**Dev**	**Test**	**Total**
# Images/Places	47,595	5,973	5,968	59,356
# Events at Present	111,796	13,768	13,813	139,377

Table 4.1: Part of Table 1 taken from VisualCOMET paper, reporting statistics about the Visual Commonsense Reasoning dataset, whose images are utilized in the VisualCOMET dataset.

person and *another person* it would not be elegant no longer as well:

"A person sits on the couch with
another person and another person on her lap".

If we used this approach for all the sentences with more than 3 persons in general, the resulting structure would introduce a non-negligible bias over the distribution of the names in the sentences, which in turn will affect the performance of the model.

In conclusion, what we opt for, is to remove all the sentences that involve 3 or more people. In analysing the dataset, we also discovered that there are some pairs repeated which are also removed after the filtering. We wondered how the filtering and the modification operations affected the dataset. In particular, we wondered whether we could still utilize the data after the change. To answer this, we computed some statistics before and after the operations, which we are going to show in the following section.

As a side note, the remaining descriptions, as well as the associated pairs, are addressed as *valid*.

4.1.1 Statistics

The dataset comes split into 3 subsets: training, validation and test. For the rest of this work, they are kept divided as they are, for convenience. Hence, in this section, 3 statistics are retrieved, one for each split.

In Table 4.1 we see the part we care about Table 1 of the VisualCOMET paper. We see the number of images and event descriptions per dataset split and their total.

As we said, the operations will affect the descriptions, but we can not avoid studying other quantities that regard the $(image, description)$ pairs. Different quantities are computed before and after the filtering.

The precise list of quantities retrieved follows:

Before Filtering

1. Number of duplicate $(image, description)$ pairs
2. Number of unique event descriptions that appear in more than one pair and relative percentage w.r.t. the unique number of descriptions
3. Average number of times that a duplicate event appears (with standard deviation)
4. Average number of persons per event (with standard deviation)
5. Median number of persons per event
6. Max number of persons per event

After Filtering

1. Number of valid $(image, description)$ pairs and relative percentage w.r.t. the initial pairs number
2. Number of images without description and relative percentage w.r.t. the total number of images
3. Average number of persons per event (with standard deviation)
4. Median number of persons per event

After substituting the strings *a person* and *another person*

1. Number of unique event descriptions appearing in more than one pair at least once and relative percentage w.r.t. the unique number of descriptions
2. Average number of times that a duplicate events appears (with standard deviation)

Statistics Results and Considerations

Following the order in the lists of the previous section, we report here the results for each split.

Looking at Table 4.3, maybe the most important value is the number of valid pairs, i.e., the number of remaining pairs after the filtering process: we have similar percentages across the splits, which float around 92% and 93% of the initial pairs. Also, we notice that the number of repeated pairs in Table 4.2, which are also filtered out, is pretty low.

What we deduce, then, is the following: firstly, the major part of the whole dataset contains at maximum 2 persons per description, and in considering just these descriptions, we do not affect the dataset in an important way; secondly, the descriptions were

Split	Training	Validation	Test
Duplicate pairs	28	3	1
Duplicate events (and %)	966 (0.88%)	37 (0.27%)	29 (0.21%)
Average appearance (and stdev)	2.66 (2.01)	2.05 (0.23)	2.10 (0.30)
Average persons x sentence (and stdev)	1.53 (0.78)	1.59 (1.09)	1.55 (1.04)
Median persons x sentence	1	1	1
Max persons x sentence	25	30	27

Table 4.2: Statistics before filtering

Split	Training	Validation	Test
Valid pairs (and %)	103677 (92.74%)	12688 (92.16%)	12861 (93.11%)
Images w/o descriptions (and %)	570 (1.20%)	74 (1.24%)	68 (1.14%)
Average persons x sentence (and stdev)	1.39 (0.53)	1.40 (0.50)	1.38 (0.51)
Median persons x sentence	1	1	1

Table 4.3: Statistics after filtering

distributed in a weighted way among the splits, which means that removing the events we do not care does not make any split too small to be utilized.

Both these considerations can be also confirmed by looking at the mean and at the related standard deviation before and after the filtering, which, in all the splits, are respectively around 1.5 and no more than 1.10 before the filtering, and 1.40 and 0.5 after the filtering. We notice also that the median is always 1, both before and after the filtering. This indeed tells us that on average before the filtering we may find 1 or 2 persons in the sentence and also that this statistic is not affected in an important way after the filtering, as the values do not change a lot.

Another effect of the filtering can be seen in accessing the values that refer to the images without descriptions in Table 4.3. Indeed, in applying the aforementioned filtering, it may be that all the pairs $(image, description)$ referring to an image that contains more than 3 persons have been removed. In this way, some images are not considered at all. Again, looking at the percentages in the same table, which are all a bit more than 1%, we do not think this can be a major problem.

One interesting filtering effect is observable by looking at the duplicated events

Split	Training	Validation	Test
Duplicate events (and %)	2152 (2.17%)	146 (1.17%)	124 (0.98%)
Average appearance (and stdev)	3.05 (3.86)	2.37 (0.99)	2.24 (0.71)

Table 4.4: Values retrieved after the substitution of the strings *a person* and *another person* in the descriptions

change after the filtering and the substitution of the strings *a person* and *another person*, Tables 4.2 and 4.4. As we already mentioned, this replacement is also guided by the will to generalize the concept of "person", without referring to specific instances. The loss of information due to this generalization of the person's identities causes descriptions that were initially different to be now the same. Considering the training split, we notice indeed that from the 0.88% before the filtering, the number of unique events that are present in more than one pair increases to 2.17% after the substitution of the strings, even if we had removed descriptions. To understand better why this happens, take the following description examples:

"1 is playing the piano",
"2 is playing the piano";

both of them appear just once in the validation set, but when we replace the number with *a person*, they end up being the same description.

In conclusion, we can state safely that the filtering process has not corrupted the dataset in such a way to make it no more usable for our purpose, in that we are still provided with more than 92% of $(image, description)$ pairs, 98% of the initial images are still associated with at least one description. Moreover, after the substitution of the numbers with the strings, it is true that we experience an increase in the events appearing more than once, thus having a certain redundancy, but we still have the reasonable amount of unique descriptions of more than ninety-five thousand.

4.2 Scene Graphs

VisualCOMET provides us with text and images, but we also need Scene Graphs associated to those images. We thus compute them through the code based on the work of Han *et al.* [11].

The Scene Graph Generation (SGG) methods available are several, but we utilize the RelDN one [11, 52]. The object detector is Mask R-CNN [11, 26]. The dataset on which the SGG is trained is Visual Genome [11, 19].

In using the code provided by Han *et al.*, we first discover that the attributes are not considered in the resulting SGs, and also, that one can choose an entity threshold and a relationship threshold, which sets a minimum so-called confidence for the respective graph elements, that is, they express how likely is that a certain element, node or relationship, is right according to the algorithm.

Graph element	List of classes
Entity	airplane, animal, arm, bag, banana, basket, beach, bear, bed, bench, bike, bird, board, boat, book, boot, bottle, bowl, box, boy, branch, building, bus, cabinet, cap, car, cat, chair, child, clock, coat, counter, cow, cup, curtain, desk, dog, door, drawer, ear, elephant, engine, eye, face, fence, finger, flag, flower, food, fork, fruit, giraffe, girl, glass, glove, hair, hand, handle, hat, head, helmet, hill, horse, house, jacket, jean, kite, lamp, laptop, leaf, leg, letter, light, logo, man, men, motorcycle, mountain, mouth, neck, nose, number, orange, pant, paper, paw, people, person, phone, pillow, pizza, plane, plant, plate, player, pole, post, pot, racket, railing, rock, roof, room, screen, seat, sheep, shelf, shirt, shoe, short, sidewalk, sign, sink, skateboard, ski, snow, sock, stand, street, surfboard, table, tail, tie, tile, tire, toilet, towel, tower, track, train, tree, truck, trunk, umbrella, vase, vegetable, vehicle, wave, wheel, window, windshield, wing, wire, woman, zebra
Relationship	at, behind, carrying, eating, hanging from, has, holding, in, in front of, near, of, on, riding, sitting on, under, using, wearing

Table 4.5: List of entity and relationship classes in the SGs.

As we will see in the SGs-statistics section, even though we tried to balance during the choice of the thresholds, there is a considerable number of edgeless graphs. To make these graphs a bit more useful, we decided to add to all the graphs a node called "SG", which connects to all the nodes through a relationship of type "contains". In this way there is no edgeless graph.

In Table 4.5 we present the lists of entity and relationship classes that we encountered in the SGs.

After delving a bit deeper on the choice of the thresholds, we present some statistics we observe by retrieving the SGs on the VisualCOMET images. The statistics concern the graphs before the insertion of the node "SG" and the relations "contain". They will be useful when we analyze the results.

4.2.1 Nodes and Relationships Thresholds

The minimum-confidence thresholds values range from 0.0 to 1.0 for both nodes and relations. In our case, we choose empirically 0.7 for the nodes and 0.5 for the relationships by inspecting the SGG output. In fact, an exhaustive evaluation of the thresholds is orthogonal to this work. We chose those values because they seemed a reasonable trade-off between quality and quantity of information. In fact, as shown in

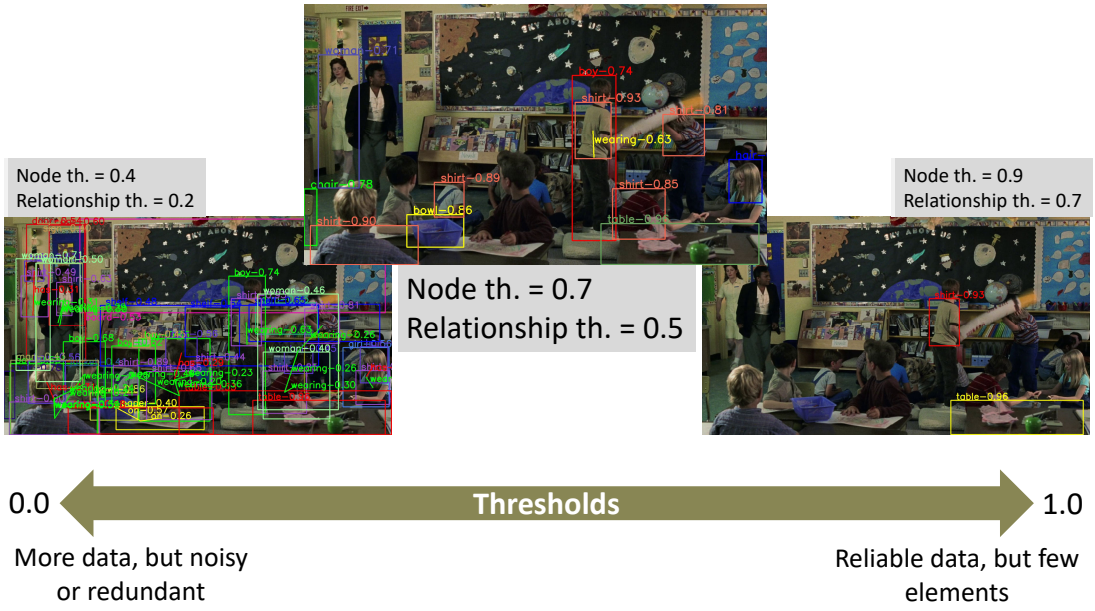


Figure 4.1: Han *et al.* [11]-SGG threshold-effect dimostration on a VisualCOMET image. The images' rectangles represent the entities detected, and the straight lines represent the connections between the nodes. Notice the big amount of boxes in the left image, and the nearly-total absence in the right image. The center image is obtained through the thresholds we utilized for all the images.

Figure 4.1, by putting the thresholds lower, we noticed that the graph could count a bigger number of nodes and relationships, thus having also tens more connections, but that they were not necessarily right or they could be redundant. On the other hand, by putting the thresholds higher, we noticed that the SGs were formed by just a small number of disconnected nodes, transforming what should have been a graph into a list of objects detected inside the image. From now on, one must keep in mind that the values that depend on the SGs can change according to these two parameters: for example, the embeddings computed through the SG Embedding system (Section 3.2.1) can vary according to the number of sentences that can be retrieved from the SG through the Graph Walk algorithm that we presented earlier in Section 3.2.1; so, smaller threshold values than the ones we chose implies more sentences and higher threshold values implies less sentences

4.2.2 Statistics

In this section we report some statistics about the SGs retrieved, which will help us in Chapter 6 to understand better the results. The purpose of these statistics is to explore

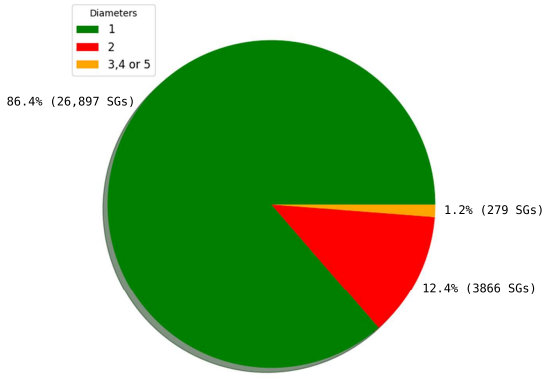


Figure 4.2: Graph-diameters distribution along the train split excluding the edgeless graphs.

how much useful the SGs can be in adding their information to the other data.

The values reported here concern the SGs before the insertion of the "SG" node and the relation "contains" and they are computed just on the VisualCOMET train split, since we have seen that they do not change in a meaningful way in the other splits.

In Figures 4.2, 4.3 and 4.4 we report the graphs diameter, the number of nodes distribution compared with the isolated nodes and the relations number distribution along the graphs, respectively. All the figures refer to the graphs that present at least one relation; the edgeless graphs are 15841, i.e., nearly 1 third of the initial number of graphs.

Looking at Figure 4.2, we notice that 86% of that considered graphs present a diameter of 1, i.e., the longest shortest path can be just a triple of type "subject - predicate - object". This means that once we insert the node "SG" and the relations, for these 86% of the graphs the diameter becomes 2. Hence, in walking the graph as described in 3.2.1, in these 86% of the cases, we will be able to produce paths that consider at most 3 nodes and 2 relations, no matter what higher length we choose, and, every path of these, will consist of initial part of type "SG contains".

Looking then at Figure 4.3, we see that the most frequent number of nodes in the considered graphs ranges from 6 to 11. The white dots correspond to the half of the value of those bins, and thanks to this we see that for the most frequent bins, at least half of the graphs have at least half of their nodes that are isolated. The consequence

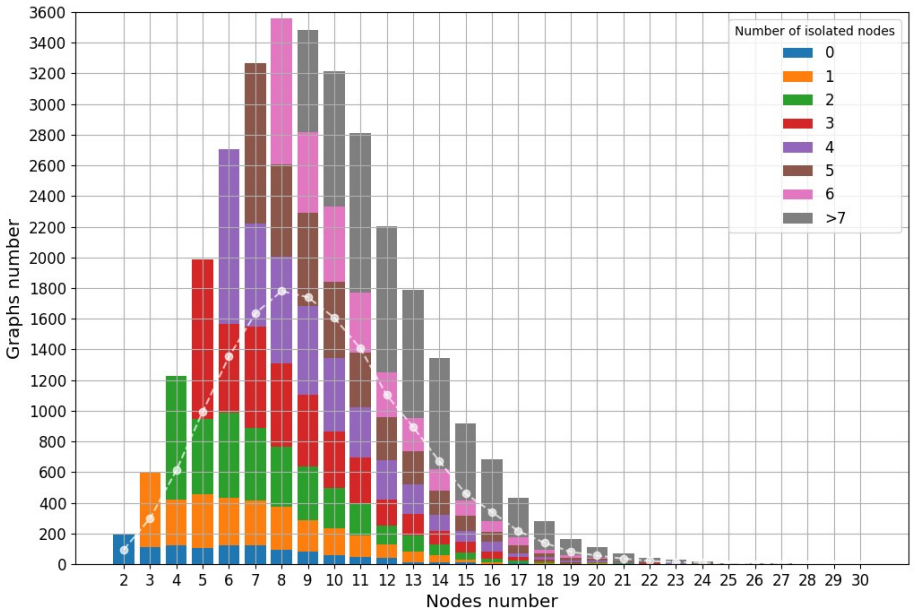


Figure 4.3: Number of nodes occurrence compared with the isolated nodes along the train split excluding the edgeless graphs.

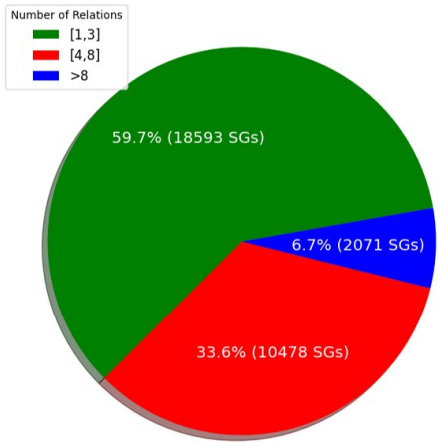


Figure 4.4: Graph relation numbers along the train split excluding the edgeless graphs.

of this is that the graph are submerged by the isolated nodes, and, without replacing the elements aforementioned, few meaningful relations are present in the graphs with respect to the number of detected nodes. This is also confirmed by Figure 4.4 concerning the relations-number distribution: in more than a half of the cases, the graphs comprise at most 3 relations, despite the higher number of nodes.

5

Experimental Part

In this chapter we first evaluate the original CLIP and then the architecture depicted in Figure 3.2, comprising both CLIP and SGs.

To evaluate the performance of both the models in the context of the Image Search task, we need to compare every text t considered with all the images in the collection, and then look at the most recommended ones (Equation 2.3) by the system.

We work on the VisualCOMET test split and we consider 2000 queries, picked randomly over all the split texts. For comparison reasons, the queries are kept the same for the models evaluation. Some of the queries utilized are reported in Table 5.1 together with their ids.

From the ground-truth, over these 2000 queries, 1986, more than 99%, have associated just one image, 13 have 2 images associated and just 1 has 3 images associated, increasing the total number of relevant images to 2015. Hence, we are safe in considering as if every query has associated just one image. This assumption will be exploited when presenting the results in the next sections. We sum up these values in Table 5.2.

Furthermore, when presenting the results, we consider just a portion of the most recommended images. This amount is indicated with K. Therefore, we will present $Hits@K$ and $Recall@K$.

5.1 Evaluation Preprocessing

The first step in the evaluation is to retrieve the data, that is, both the annotations and the images. Since the images are split in a random way across the splits, to retrieve the $(image, description)$ pairs we must start from the annotation document and from

Id	Query
5	A bunch of boys are in a motel room having fun and making a mess of it
47	A group of cyclists ride down the hill
113	A group of soldiers stands behind a person and watches him
425	a person & another person argue on the rooftop of the bulding
560	a person a robber is crouching among the customers in a grocery store aisle
746	a person and another person are arguing
3092	a person comes out of a room
13178	a person is handing another person a hand towel
17486	a person is looking up at another person as he risks his life to fix something

Table 5.1: Examples of the 2000 queries picked randomly to evaluate the models

Number of relevant Images	Number of queries
1	1986 (99.3%)
2	13 (0.65%)
3	1 (0.05%)

Total number of relevant images = 2015.

Table 5.2: Number of relevant images of the 2,000 queries utilized for the evaluation.

that gather the paths to the test-split images. Moreover, as we have seen in Section 4.1, the annotation texts require some modifications, which involve a filtering and some substitutions. After modifying the texts and loading all the images, we need to encode all of them, which means to calculate the features through the neural networks. Finally, the evaluation could take place, computing the similarities between the features and retrieving the metrics.

Directly performing all the steps just described during the evaluation software development is a computationally expensive process, both in time and memory terms. Hence, we used some pre-processing techniques, which are depicted in Figure 5.1.

The steps are as follows:

1. First, after having accessed VisualCOMET annotations, map the images' paths into ids, to avoid using strings;
2. Use both image ids and the annotations to assign the images' ids to the related description; in this step both the filtering of the annotations and indexing of the descriptions take place; hence, two documents are produced: a table of text-id relations and a map text id $\rightarrow$ images' ids;
3. Encode all the descriptions into feature tensors and store a dictionary in which the key is the text id and the value is the associated feature tensor;
4. Use image ids and the image database to encode the images and assign to every image id the associated tensor. Since working on all the images is too expensive in terms of memory, batches of images are preprocessed time by time;

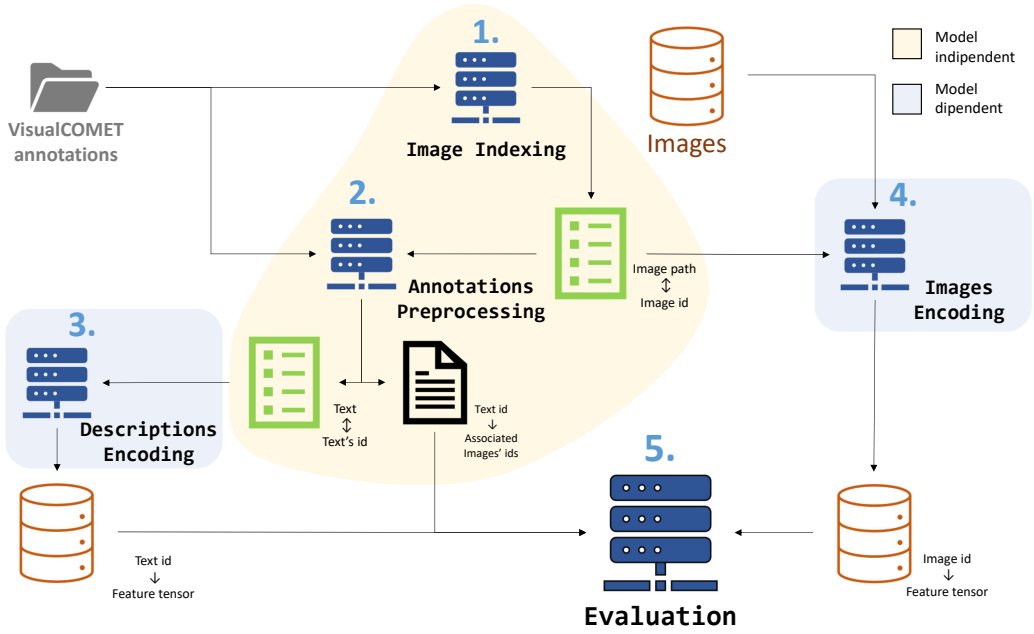


Figure 5.1: Pre-processing on images and annotations.

5. Use the produced text id→ images' ids map, the description feature tensors and the image feature tensors to perform the evaluation.

As we can notice, steps 1. and 2. do not depend on the architecture utilized. On the other hand, steps 3. and 4. depend on the architecture considered, because it is in these two in which we retrieve the model features. Hence, we utilize the data produced through steps 1. and 2. in both CLIP and CLIGT evaluation. In the next section, we describe the evaluation performed on CLIP, and we report the test-split results.

5.2 CLIP Evaluation - Results

Recalling briefly what we have seen in Section 3.1, CLIP encodes images and texts and returns features of both of them, belonging to the same multi-modal vector space. Once the features are retrieved, we need to compare them using the Cosine Similarity function seen in Equations 3.1 and 3.2. OpenAI GitHub repository permits us to make use of the model[1].

[1]https://github.com/openai/CLIP

Hits sum k=10	Recall avg k=10	Hits sum k=100	Recall avg k=100
448	0.2223	985	0.4900

Table 5.3: CLIP-evaluation results

The *Hits@K* sum and *Recall@K* average of all the 2,000 queries are presented in Table 5.3 for $K = 10$ and $K = 100$.

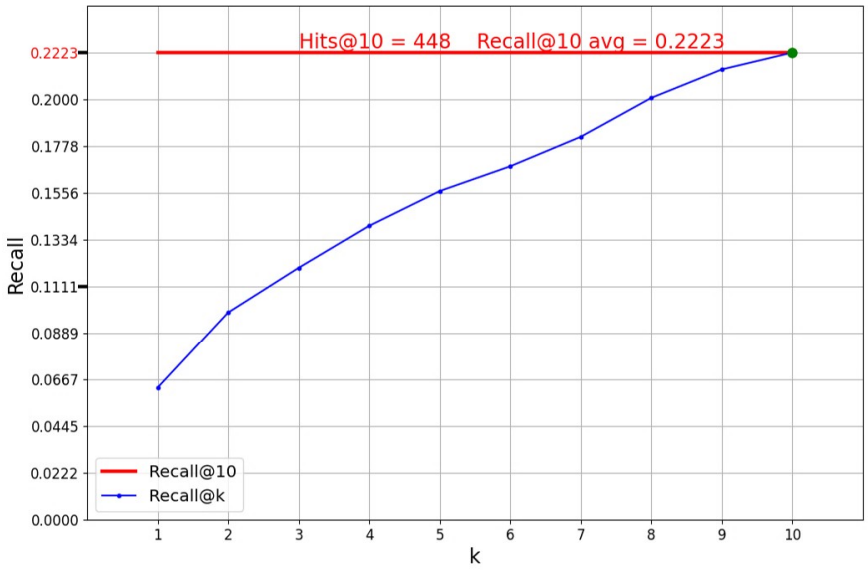


Figure 5.2: Recall Trend clipped at $k = 10$

Following the assumption done previously, the table values mean that those 448 images represent the nearly 22.2% of the cases in which the ISS is able to return the relevant image within the first 10 recommended ones. For the other 77% of the queries, the ISS is not able to return the relevant image in that range of recommended ones. Concerning $K = 100$, the *Recall@100* is nearly 0.5, which means that on average in almost half of the cases, the ISS is able to rank the relevant image within the first 100 recommended images, i.e., given a query, its relevant image can be found in one of the first 100 positions. We have just talked about ISS, and in our case it means that CLIP computed features that, when compared through the Cosine Similarity function, provide those evaluation metrics. In this way, we have just given an answer to the Binary Image Search Problem defined in Problem 2.2.1.

Concerning the Ranked Image Search problem, we report Figures 5.2 and 5.3. For

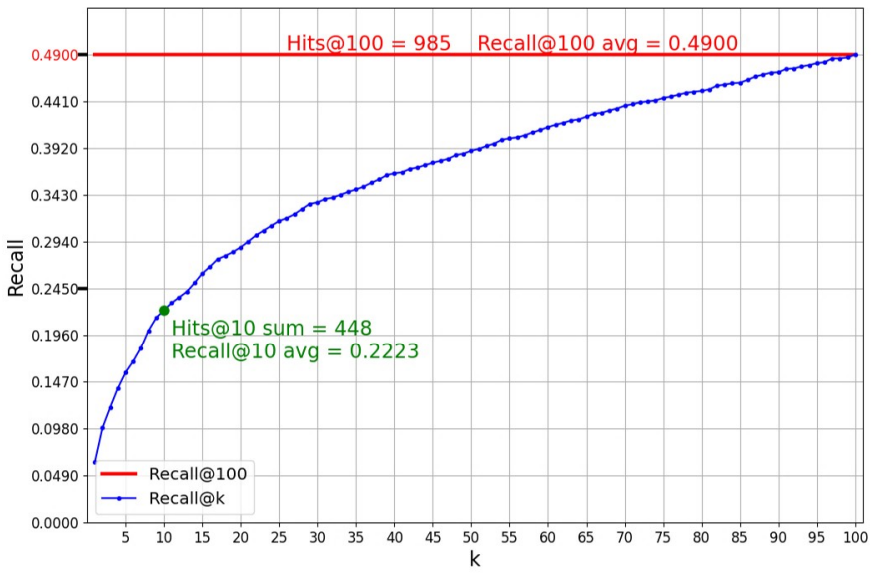


Figure 5.3: Recall Trend clipped at $k = 100$

every image, the red horizontal line corresponds to the maximum recall reachable by the function. If the blue line reaches the red one for a value $k_0 < K$, it means that all the recommended relevant images lie in the first k_0 most recommended images. Looking at the figures, we can understand how the relevant images are ranked by the ISS, i.e., how far away they are placed with respect to the first position, in those cases in which they are present in the most K recommended ones.

In particular, from Figure 5.2, which stops at the maximum of $K = 10$, we notice that the recall increases more or less linearly up to that value of K. This means that the relevant images are distributed in a nearly uniform way in the first 10 positions. Figure 5.3 instead depicts the recall trend up to $K = 100$ and we see that the distribution is not uniform, and in particular it we see that the relevant images are found in more than half of the cases in the first twenty images.

5.3 CLIGT Train

The purpose of the FFNN blocks is to project the incoming vectors into a new vector space. In this way visual vectors and textual vectors lie on the same multi-modal space. In this section we present the work done on the FFNNs.

5.3.1 Data Preprocessing

Before inputting the data into the networks, we wondered whether we needed to rescale the components or not. To answer this question, we observed the distribution of the feature components values, which we report in Figure 5.4.

First of all, notice that the y-scale of the 3 plots is logarithmic. We observe a big difference in the occurrence of values close to zero and other relatively far from 0: for all the 3 types of data, we have more then 10^7 values ranging from -0.5 to 0.5 down to less than 10 in the cases of text and SG features for values around -16 and -5 respectively. This shows the presence of outliers in the values and makes the distribution resemble the bell shape of a Gaussian one, even though we are kind of far from a real Gaussian curve.

We thought to choose between the normalization and the standardization operation and since the normalization is not suitable when there are the conditions just mentioned, because it squeezes the values between a specific range[27], and also we do not care about forcing the range to be a specific one, we adopted the standardization, which means to substract all the data by its mean and divide it by its standard deviation.

The next step is deciding whether grouping the data during the standardization, and if yes, how. Due to the features nature, we decided to standardize together the outputs of CLIP and to standardize the SG ones on their own. The data thus pre-processed is utilized as input to the FFNNs.

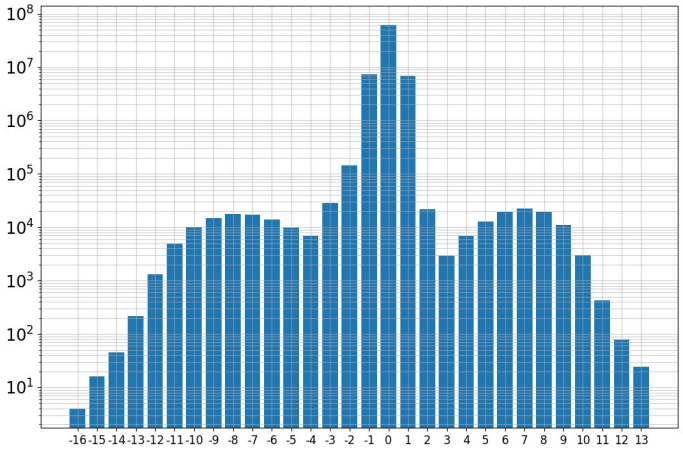
We trained the FF layers in 2 different ways, which are presented in the next two sections.

5.3.2 Binary Classification

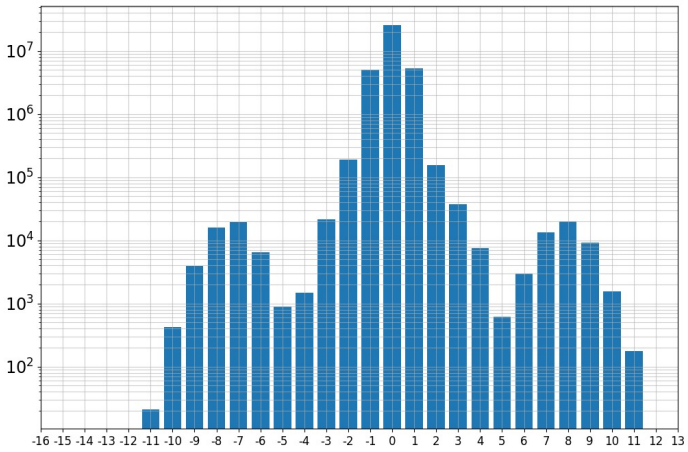
In considering the networks' train as a binary classification task, we identify 2 classes (labels): one that corresponds to maximum similarity, i.e., when the cosine similarity returns the value 1, and one that corresponds to the similarity due to orthogonal feature vectors, i.e., when the cosine similarity returns 0.

Building on this, we present the pseudocode of a train iteration in Algorithm 2.

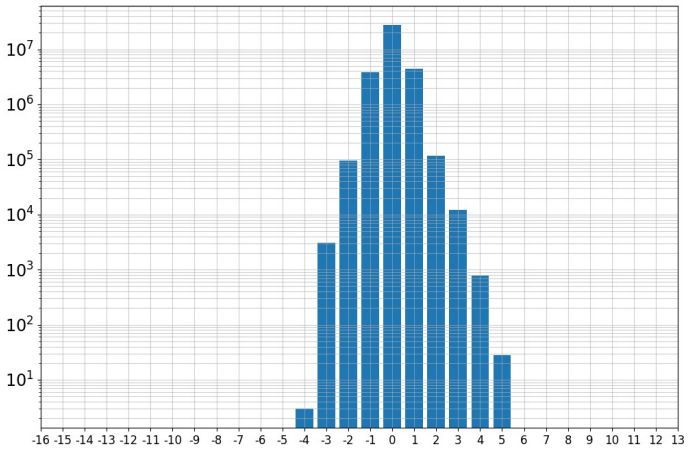
In the first part of the train iteration, the ground-truth labels are computed. Indeed, in our case, we cannot consider that just the diagonal contains 1s, i.e., maximum similarity elements, because, as we have seen in Section 4.1.1, dedicated to the VisualCOMET statistics, on average every image is associated with at least 2 descriptions, and some descriptions are associated to more than one image. Therefore, we must assume that



(a) Text feature components distribution



(b) Image feature components distribution



(c) SG embedding components distribution

Figure 5.4: Feature-components distribution. Notice the logarithmic scale along the y axis

Algorithm 2 FF networks $TtxNN$ (Text NN) and $VisNN$ (Visual NN) train-loop single iteration

Input:

- N text ids id_t and N visual ids id_v, such that ($id_{i,t}$,$id_{i,v}$) is a positive pair, for $i = 1, ..., N$
- text-features data-structure DS_t with text id → text feature mapping
- visual-features data-structure DS_v with visual id → visual feature mapping
- text-images mapping Ann (annotations)

Output: batch loss

1: initialize $gtMatx$ as $N \times N$ zero matrix{Matrix containing the ground-truth labels}
2: replace $gtMatx(i, j) = 1$ if $id_{j,v}$ is in $Ann(id_{i,t})$, for $i, j = 0, ..., N - 1$
3: retrieve $weights$ for the positive and negative examples
4: $f_t \leftarrow DS_t(id_t)$ {Dimension $N \times M$}
5: $f_v \leftarrow DS_v(id_v)$ {Dimension $N \times (M + S)$}
6: $f'_t \leftarrow TtxNN(f_t)$ {Dimension $N \times F$}
7: $f'_v \leftarrow VisNN(f_v)$ {Dimension $N \times F$}
8: $SimMatx \leftarrow CS(f'_t, f'_v)$ {Dimension $N \times N$}
9: $loss \leftarrow binary\_cross\_entropy(SimMatx, gtMatx, weights)$
10: **return** $loss$

some elements outside the main diagonal are one as well.

In the third line of the algorithm we retrieve the weights for positive and negative samples. In fact, we tried first with an unweighted train and we discovered that the network simply learnt to build feature vectors such that their cosine similarity is always near to 0. This is due to the high number of negative samples that is created by computing all the possible pairs in a N-sized batch. In this case, indeed, the positive samples are almost N, $P \approx N$, and the negative samples are $N^2 - P \approx N^2 - N$. This is called normally a class imbalance.

The algorithm then proceeds by retrieving all the features from the relative data structures and by computing the similarity values through the application of the Cosine Similarity function to all the possible feature vectors pairs.

The loss returned is the binary cross entropy[2] between the ground-truth label matrix $gtMatx$ and the similarity matrix $SimMatx$, weighed by the parameters found before.

[2]https://pytorch.org/docs/stable/generated/torch.nn.functional.binary_cross_entropy.html

5.3.3 Alternative approach - Energy-based Train

As we will see in Section 5.4, the results concerning the binary classification model are worse than the original CLIP. The idea presented in this section is an attempt to train the network with a different approach, to study the problem from a different perspective, thus hoping to understand better the issues.

The approach builds up on the idea of Bordes *et al.* TransE [2] and consists in the minimization of a loss function called energy function. In this context, given one positive pair we always select just a negative one: more formally, given a positive pair $(id_{i,t}, id_{i,v})$, we select $id_{i',v}$ such that it is not associable with $id_{i,t}$, which means that the similarity of $id_{i,t}$ and $id_{i',v}$ should be minimized, or alternatively, their dis-similarity should be maximized. From now on, following the notation used in Algorithm 2, f is a feature vector retrieved from the respective data structure DS_t or DS_v.

The loss function on a batch of N pairs is the following:

$$loss = \frac{1}{N}\sum_{i=0}^{N}[\gamma + d(f_{i,t}, f_{i,v}) - d(f_{i,t}, f_{i',v})]_+, \tag{5.1}$$

where $\gamma > 0$ is the so-called margin parameter [2], $[x]_+$ is the positive part of x and $d(x, y)$ is the distance between x and y, computed as the Squared Euclidean Distance:

$$d(x, y) = (x - y)^2 = ||x||_2^2 + ||y||_2^2 - 2x^T y \tag{5.2}$$

As explained in the work of Bordes *et al.* [2], putting the additional constraint of $||x||_2^2 = ||y||_2^2 = 1$ is preferable.

Hence, in the case of 5.2, we can rewrite the equation in the following way:

$$\frac{1}{N}\sum_{i=0}^{N}[\gamma - 2f_{i,t}^T(f_{i,v} - f_{i',v})]_+, \tag{5.3}$$

which is the loss function minimized.

5.3.4 Ablation Study

We will see that also the alternative approach does not even reach CLIP performance. For this reason, we decide to delve deeper on the problems of the designed architectures: we propose an ablation study based on the removal of the standardization effect described in Section 5.3.1 and additionally, on top of the standardization removal, we experiment

replacing the SGs embeddings with 0 vectors. The results concerning the ablation study are presented in the dedicated part in Section 5.4.

5.3.5 Finding FFNNs Best Architecture - Hypertparameter Optimization

The hyperparameters present in these trains are a lot: the number of epochs for which we train the networks, the batch size, the number of layers per NN, the number of nodes per layer, the learning rate, the regularization factor.

In order to find the best FFNN architecture, we conducted an optimization through *Optuna* module, but we fixed a couple of parameters to decrease the dimension of the search space: following what CLIP researchers did, for the classification train we fixed a big batch size of $N = 20,000$ and we utilized the small number of epochs of 8 [32]. Notice that in this way the resulting similarity matrix is made of $N^2 = 4 \cdot 10^8$ values. This has the effect of reducing the overfitting risk [32]. For the energy-based case we optimized fixing the batch size to 64 and the same epochs number of 8.

5.4 CLIGT Evaluation Results

In this section we present the results concerning both the approaches and also the results retrieved through the ablation study.

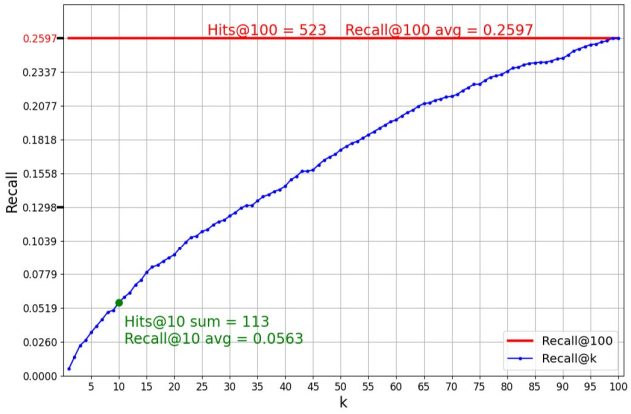
In Table 5.4 we present the results of the evaluations carried on the same 2000 queries used for the CLIP evaluation in Section 5.2.

Approach	Variation	Hits sum k=10	Recall avg k=10	Hits sum k=100	Recall avg k=100
Classification	WS	113	0.0563	523	0.2597
	WoS	164	0.0820	628	0.3127
	WoS & Zeros	174	0.0870	664	0.3295
Energy based	WS	184	0.0917	674	0.3350
	WoS	**206**	**0.1030**	746	0.3708
	WoS & Zeros	**206**	**0.1030**	**794**	**0.3950**

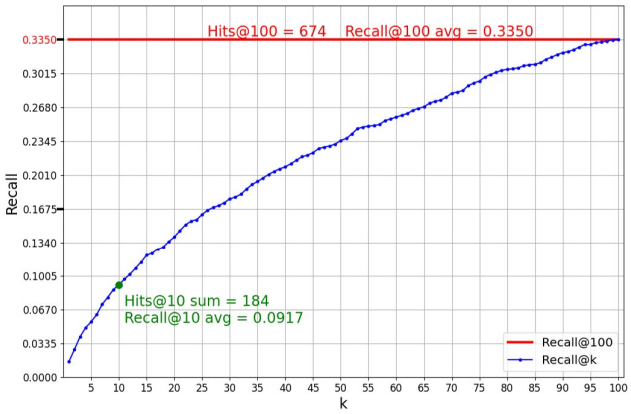
Table 5.4: Summary of the results of the evaluations carried on the different models. In the Table, WS means With Standardization and WoS means Without Standardization.

In Figure 5.5 we show the Recall trends graphs corresponding to the two approaches applied. For every image, the red horizontal line corresponds to the Recall@100, and

the blue line corresponds to the trend of the Recall@K, with $K = 1, ..., 100$. Just the recall trends corresponding to the approaches adopting classification are depicted, without those referring to the ablation study: this is because the curves do not show a substantial difference. Indeed, we do see by looking at Figure 5.5 that already the curve shape of the two approaches are very similar. Referring to the Ranked Image Search problem, we cannot infer anything but the fact that there is no meaningful difference in the distribution of the recommended relevant images.



(a) Recall trend for the Classification case.



(b) Recall trend for the energy-based case.

Figure 5.5: Recall trends for the evaluations carried on the different models.

All the recall trends are reported in Figure 5.6 in a comparison graph with respect to CLIP. Thanks to it, we can appreciate how the recalls evolve with respect to K. In particular, notice that CLIP model outperformes the others already when considering $K = 1$. CLIGT metric values are always worse than CLIP, but there are some models that

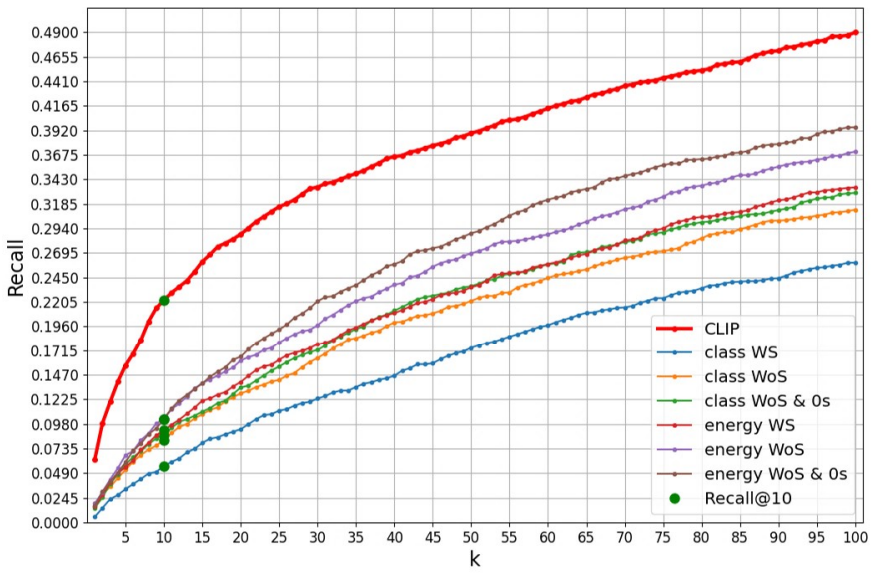


Figure 5.6: Comparison graph of the recall trends

perform worse than others. For instance, notice how the classification model adopting the standardization always performs worse. On the other hand, the energy-based model retrieved through the ablation study that eliminates both the standardization and the SGs, replacing them with 0s vectors, is the best one after CLIP from $K = 20$. Furthermore, we discover that all the metric values retrieved on the energy-based models are always nearly better than those of the classification models.

We discuss some hypothesis of the outcome of these results in the next chapter, where we delve on the different parts of the architecture.

6

Failure Analysis and Future Works

In this chapter, we give an explanation of the results and we investigate the possible failure reasons. In fact, by looking at Figure 5.6 we understand that the evaluation metric values are worse than the original CLIP. Yet, some of them allow us to infer interesting features, which in turn permits to hypothesize something on the possible failure causes.

We identified some possible problems, so we divide this chapter in sections according to the cause: we first analyze the effect of the standardization; then, just the loss utilized in the two approaches is examined; afterwards, we consider the information carried by the SGs; the Scene Graph Embedding block and finally the FFNN architecture are then taken into consideration.

What we propose in this chapter are hypothesis built on the results and on some exploratory statistics analysis we performed on the data we utilized. They should be confirmed by further analysis, which could highlight what are the real source of the unsatisfactory performance of these models.

6.1 Standardization Effect

From the results, we can infer that the standardization we performed has not got the desired effect. Instead, it worsened the performance. This comes by looking at Table 5.4 in the rows "WS" and "Wos" in both the approaches: for example, in the classification case with $K = 10$, we pass from a 113 hits sum adopting standardization to a 164 without standardization; also looking at all the other cases we confirm an increase in the metric values without standardization.

We speculate that the main problem in this context is the fact that when we performed

the standardization on the data, we kept CLIP descriptions and images features together, computing the mean and standard deviation taking into account all of them, even though the two types of features were used as input to two different blocks.

Removing the standardization we performed increases the performance, but we wonder whether there can be a more appropriate standardization for which we can instead improve the performance. Since grouping the data has not led to an improve, we suppose that by executing the standardization on the single data-type features, the evaluation metrics could be higher. This can be part of a future work.

6.2 Approach Effects

6.2.1 Classification Approach

In Table 5.4 the results concerning the WoS + Zeros classification case, i.e., the classification ablation study in which also the SGs data is replaced with 0s, are the best with respect to the other classification variations. Since the only difference with respect to the classification WoS is the substitution of the 0s to the SG embeddings, we infer that the cause of the improvement is the removal of the embeddings.

Furthermore, notice that in this way, we are just using VisualCOMET data without variation, which resembles the original CLIP evaluation. Why then don't we obtain the original metric values retrieved in Section 5.2?

The only difference with CLIP is the insertion of the FFNN blocks, trained with the classification approach. This means that the classification loss, paired with the architecture, are detrimental for the performance, since we get nearly one third of the original metric values.

We suppose the problem here is the label discretization: we impose the Cosine Similarity to be either 0 or 1, forcing the associated-content features to be "similar", i.e., to have $CS = 1$, and all the other non-associated contents to be completely non-correlated. The problem relies on this last category. In fact, given two feature vectors x and y, forcing $CS(x, y) = 0$ means forcing x and y to be orthogonal, which in turns implies that they must not share any information at all. This is not possible in our case, since, for example, given two descriptions, they may share an object word and they may not refer to the same image in VisualCOMET.

As a specific example, we take the following description:

Figure 6.1: VisualCOMET image from the movie "Robin Hood" depicting, among the other things, a horse in the foreground.

"A person is leaving quickly after he sells the **horse** to another person in secret."

This description refers to a frame depicted in Figure 6.1 taken from the movie "Robin Hood". However, the entity "horse" appears also in many other VisualCOMET images and descriptions. For instance, the image taken from "The Lord Of The Rings - The Return Of The King" depicted in Figure 6.2 also contains a horse and the associated description also contains the word "horse":

"A person is leading the **horse** in back of him"

In the dataset, the description of the Robin Hood image is not associated with the image of The Lord of The Rings, and neither the opposite happens.

Several other examples could be made directly from the dataset. This means that there is several information that is shared among the descriptions and the images, and one can't simply force two non-associate feature vectors to be orthogonal, because they may share some concepts and their ideal Cosine Similarity could have a value $-1 < CS < 1$, $CS \neq 0$.

Then, this kind of approach is not suitable for this specific task.

6.2.2 Energy-based Approach

The results concerning the energy-based train WS and WoS lead to a better performing model than the classification one. Yet, we are far from even reaching the performance

Figure 6.2: VisualCOMET image from the movie "The Lord Of The Rings - The Return Of The King" also depicting a horse.

of CLIP. We think that the main reason why the model still performs this poorly is the removal of the contrastive feature in the train loss: as we said in the previous chapter, given a positive pair we select just a single negative pair, collecting just two values of dis-similarity. Instead, CLIP authors selects $N = 32,768$ negatives given one positive [32], and also in our classification approach we considered $N = 20,000$ negative pairs per positive one, resulting respectively in $32,768$ and $20,000$ similarity values per positive in the case of CLIP.

To confirm this aspect, one could perform an analysis in which instead of using just one negative pair per positive one, more negative pairs could be taken and multiple losses could be computed keeping always the same positive pair. If what we have just hypothesized is right, the performance of the model thus trained should be better.

Despite the less solid train per description, the loss utilized comprises two interesting characteristics which are, in our opinion, what really make the difference with respect to the classification case:

1. the energy-based loss is built upon differences of dis-similarities, rather than similarities;

2. the single distance is not forced to take a specific value, but rather, it suffices that the positive-pair distance and the negative-pair distance differ for a value γ.

In the case for example of a positive pair, this is important because we are not forcing the feature vectors to be parallel (which corresponds to CS=1), we are simply requiring their distance to be smaller of a value γ than the distance between a negative pair. This allows for a bit more flexibility when training the network.

6.3 Scene Graph Benchmark

The point of this work was to show that by introducing the Scene Graphs semantic information, we are able to improve the performance of CLIP in the context of Image Search.

Apparently, our model does not seem to be able to appropriately explore the information of the SGs to produce a better ranking, but it does not mean that the problem is the original idea. At this point, we wonder whether the SGs contained low-quality information. From the analysis we show below, we speculate that this is actually the main reason.

Before going on, we recall the fact that the results can vary also with respect to the minimum-confindence thresholds used for the entity and for the relation during the SGG. In fact, as we have described in Section 4.2.1, we chose empirically one specific value for each threshold based on the SGs output. However, by changing them, the SG embeddings vary, influencing also the train on the FFNN.

Looking back at Section 4.2.2, we can understand that the information brought by the SGs is quite noisy for different aspects that we are going to describe below.

First of all, the length of the possible paths does not permit a meaningful description of the image, in that in most of the cases, we can just create paths that comprise two relations, one of which is "contains" that starts from the node "SG". The useful description of the image derives from longer paths, so that the elements are put more in relation one another. On top of this, we have seen from the pie chart in Section 4.2.2 that most of the SGs comprise a low number of relations. One third of the SGs don't even entail any relation. Beyond losing the interaction between the entities, having few relations makes also the nodes-class appearance crucial, in that to differentiate the images we need more objects or more-specific labels of the objects detected. We take as an example some of the images whose SGs are edgeless and contain at least once the three nodes "man", "horse" and "tree", depicted in Figure 6.3. A part from the fact that the second image does not even contain horse but dromedaries, wrongly detected during the SGG, in the scenes we can see very different contents that can't be simply differentiated by the words "man", "horse" and "tree". What could help in this is the use of different relations from "man" to "horse" such as "spurring" for the first image, "calming" or more generally "next to" for the third image, and "on" in the fourth. By using just a set of isolated nodes, we are categorizing the images just by those elements, without further specifying how these images are different.

Keeping the focus on the entities, we show Figure 6.4, where we depict the number

"man wearing shirt"	13566
"man has hair"	10358
both	6412

Table 6.1: Table showing the correlation between the triples "man wearing shirt" and "man has hair".

of graphs in which the relative entity appears at least once. The first thing we see in this graph is that the distribution is peaked, so there some words that appear a lot more than others. It follows from the information theory that the most frequent words are less informative than other less frequent words. In this context, it means that the most frequent words are less discriminative, i.e., they do not help in differentiating the content of the image. Paradoxically, the two most-frequent categories, "hair" and "man", are more informative when they are not present in the graph, since in nearly two thirds of the SGs those categories are present.

Moreover, we discover also some correlations: we computed the number of times that the triple "man wearing shirt" appears, the number of times that "man has hair" appears, and the number of times in which both of them appears in the same SG. The results are depicted in Table 6.1. We understand that in more than half of the cases in which appear "man has hair", also "man wearing shirt" is present. The information theory comes again handy in saying that the information brought by "man wearing shirt" includes a good part of the information brought by "man has hair", which is thus less useful. Notice that we already said that we have few relations and small diameter. If, on top of these characteristics, we add also the fact that some relations are not useful, the SGs become even less effective.

We think that a cause of the peaked distribution and of the correlation is due to the generality of labels such as "man", "woman", "building" and "shirt". That is why they appear in a lot of SGs. One future work could focus on a SG generation model able to compute more descriptive and specific SGs.

The source of the noise is then due to all these aspects, since a lot of data is shared among the SGs without making a real difference. This is valid in general, but the results on this context differ slightly for the two approaches, so we are going to comment in the next two small sections our idea of why the results differ.

Classification Case

We have seen that the classification loss requires the use of two labels, which shape a stricter training process, which in turn does not perform well when dealing with shared

content among the data. In this case, in fact, the architecture is more sensible to noisy data, making the SGs we utilize more a source of noise than a source of information.

ENERGY-BASED CASE

In considering $K = 10$, the energy-based metric values for the ablation study don't change in Table 5.4. However, when looking at the recall trend in Figure 5.6, we see that the ablation study comprising also 0s vectors for this approach is the best overall. Hence, also in this case the SGs worsen the results, but they have less impact than in the classification approach. In fact, the metric results for $K = 10$ are more important than those with $K = 100$. We speculate that the less negative impact of the SGs is due to the more flexibility by the energy-based loss, which is apparently enough to deal with the noise inside the SGs, so that they happen to effectively bring some positive contribute to the system.

6.4 SCENE GRAPH EMBEDDING SYSTEM

The designed embedding system also has some faulty features that we are going to discuss in this section.

As we said, given an SG, we first retrieve all the possible walks of a certain length in the SG, we then code them through BERT, and then we average all the feature vectors. However, we recall also that the sentences we create through the graph walk we described in Section 3.2.1 are not really natural language sentences. We then wonder whether a more appropriate encoder could be utilized in this context.

An evaluation of the embedding system could be carried out, thus examining the SG embeddings produced. One future study could be thus performed to explore the SGs embeddings information, evaluating at this point the similarity between SGs embeddings that refer to more related images with respect to other non related.

Furthermore, we suppose that the utilized aggregator function, a simple average, is a not appropriate for this case, since it takes all the sentence embeddings computed and aggregates them in a lossy way. In fact, in principle, different combinations of feature vectors could also result in the same average feature vector. We wonder if there is more effective way of encoding an entire SG, and this is part of the future work.

6.5 Feed-Forward Blocks

Once we obtained all the data, we trained the simplest possible DL architecture, i.e., the Feed-Forward neural network.

We wonder whether there are more appropriate neural networks for this task. Indeed, for example, CLIP researchers utilized encoders such as transformers [32].

On top of this, we need to count that the optimization of the network always requires time since we work with several hyperparameters, and, in our case, we decided to fix some of them to decrease the search space dimension. Hence, there is still the possibility of exploring more extensively the full search space of hyperparameters, with the purpose of designing a better performing network.

A deeper study on this aspect is thus part of the future work.

(a) Image from "Gandhi".

(b) Image from "Gandhi".

(c) Image from "The Lord of the Rings the Return of the King".

(d) Image from "The Lord of the Rings the Return of the King".

Figure 6.3: Images taken from VisualCOMET from different movies, whose SGs are edgeless and contain "man" "horse" and "tree".

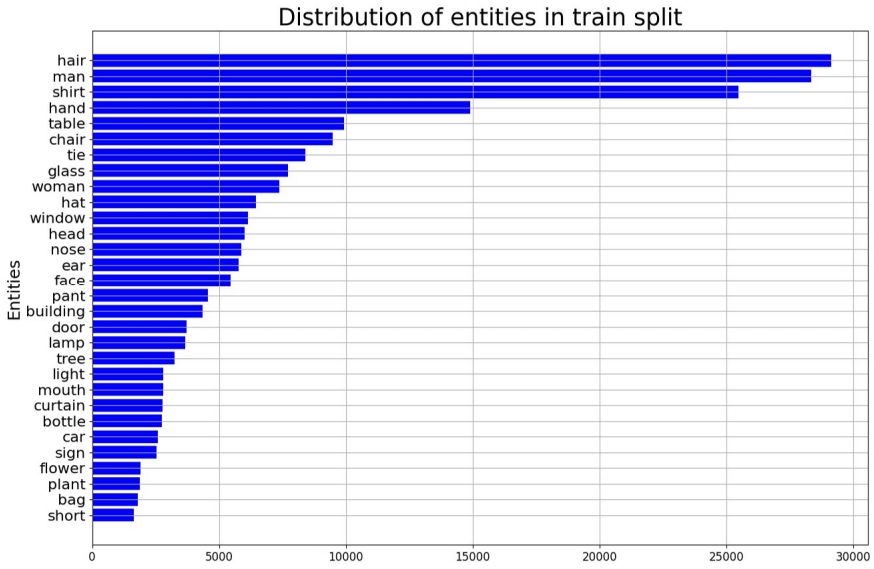


Figure 6.4: Number of graphs in which the relative entity appears at least once.

7

Conclusions

In this work, we have hypothesized that the introduction of the Scene Graphs in an Image Search solution could improve its performance. In particular, we worked with Contrastive Language-Image Pre-train [32], with the dataset VisualCOMET [29]. We have seen that what we envisioned does not happen, but we have also thought about different problem sources. The original idea, then, should not be discarded completely before performing some further analysis that may help in discovering the real problems and may give an idea of what one could do to try to improve the results present in this work.

To sum up, we hypothesize that the major problems reside in the train approaches, in the SGs structure and in the SG embedding system. In particular, we have seen that among the two approaches utilized, i.e., classification and energy-based, the less detrimental is the latter. Future works could also be conducted to understand what standardization should be carried out, if any, and what different NN, if any, could be implemented in this context to better project the 3 kind of features into a single vector space.

References

[1] Silvio Barra et al. "Visual question answering: Which investigated applications?" In: *Pattern Recognition Letters* 151 (2021), pp. 325–331. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2021.09.008. URL: https://www.sciencedirect.com/science/article/pii/S0167865521003147.

[2] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[3] Xiaojun Chang et al. "Scene Graphs: A Survey of Generations and Applications". In: *ArXiv* abs/2104.01111 (2021).

[4] Jeffrey L. Elman. "Finding Structure in Time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. DOI: https://doi.org/10.1207/s15516709cog1402_1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1402_1. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1.

[5] Fartash Faghri et al. "Vse++: Improving visual-semantic embeddings with hard negatives". In: *arXiv preprint arXiv:1707.05612* (2017).

[6] Andrea Frome et al. "Devise: A deep visual-semantic embedding model". In: *Advances in neural information processing systems* 26 (2013).

[7] Ross Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[8] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[9] Jiuxiang Gu et al. “Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7181–7189.

[10] Claudio Gutierrez and Juan F. Sequeda. “Knowledge Graphs: A Tutorial on the History of Knowledge Graph’s Main Ideas”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020, pp. 3509–3510. ISBN: 9781450368599. DOI: 10.1145/3340531.3412176. URL: https://doi.org/10.1145/3340531.3412176.

[11] Xiaotian Han et al. *Image Scene Graph Generation (SGG) Benchmark*. 2021. DOI: 10.48550/ARXIV.2107.12604. URL: https://arxiv.org/abs/2107.12604.

[12] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: June 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[13] Marcel Hildebrandt et al. “Scene graph reasoning for visual question answering”. In: *arXiv preprint arXiv:2007.01072* (2020).

[14] A. L. Hodgkin and A. F. Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of Physiology* 117.4 (1952), pp. 500–544. DOI: https://doi.org/10.1113/jphysiol.1952.sp004764. eprint: https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1952.sp004764. URL: https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1952.sp004764.

[15] MD Zakir Hossain et al. “A comprehensive survey of deep learning for image captioning”. In: *ACM Computing Surveys (CsUR)* 51.6 (2019), pp. 1–36.

[16] MD. Zakir Hossain et al. “A Comprehensive Survey of Deep Learning for Image Captioning”. In: *ACM Comput. Surv.* 51.6 (Feb. 2019). ISSN: 0360-0300. DOI: 10.1145/3295748. URL: https://doi.org/10.1145/3295748.

[17] Yan Huang et al. “Learning semantic concepts and order for image and sentence matching”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6163–6171.

[18] Justin Johnson et al. “Image retrieval using scene graphs”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3668–3678. DOI: 10.1109/CVPR.2015.7298990.

[19] Ranjay Krishna et al. “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations”. In: *International Journal of Computer Vision* 123.1 (Feb. 2017), pp. 32–73. DOI: 10.1007/s11263-016-0981-7. URL: https://doi.org/10.1007/s11263-016-0981-7.

[20] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[21] Kuang-Huei Lee et al. “Stacked cross attention for image-text matching”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 201–216.

[22] Gen Li et al. “Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 11336–11344.

[23] Shuang Li et al. “Person search with natural language description”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1970–1979.

[24] Yangguang Li et al. “Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm”. In: *arXiv preprint arXiv:2110.05208* (2021).

[25] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[26] Francisco Massa and Ross Girshick. *maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch*. https://github.com/facebookresearch/maskrcnn-benchmark. 2018.

[27] Peshawa Muhammad Ali and Rezhna Faraj. *Data Normalization and Standardization: A Technical Report*. Jan. 2014. DOI: 10.13140/RG.2.2.28948.04489.

[28] Zhenxing Niu et al. “Hierarchical multimodal lstm for dense visual-semantic embedding”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1881–1889.

[29] Jae Sung Park et al. “VisualCOMET: Reasoning About the Dynamic Context of a Still Image”. In: *ECCV*. 2020.

[30] Bryan A Plummer et al. “Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2641–2649.

[31] David M. W. Powers. “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation”. In: 2008.

[32] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *ICML*. 2021.

[33] Aditya Ramesh et al. “Zero-Shot Text-to-Image Generation”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8821–8831. URL: https://proceedings.mlr.press/v139/ramesh21a.html.

[34] Scott Reed et al. “Learning deep representations of fine-grained visual descriptions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 49–58.

[35] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: http://arxiv.org/abs/1908.10084.

[36] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 91–99.

[37] Petar Ristoski and Heiko Paulheim. “RDF2Vec: RDF Graph Embeddings for Data Mining”. In: *SEMWEB*. 2016.

[38] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65 6 (1958), pp. 386–408.

[39] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[40] Sebastian Schuster et al. “Generating semantically precise scene graphs from textual descriptions for improved image retrieval”. In: *Proceedings of the fourth workshop on vision and language*. 2015, pp. 70–80.

[41] Sheng Shen et al. “How Much Can CLIP Benefit Vision-and-Language Tasks?” In: *arXiv preprint arXiv:2107.06383* (2021).

[42] Arshiya Simran, P.S Shijin Kumar, and Srinivas Bachu. “Content Based Image Retrieval Using Deep Learning Convolutional Neural Network”. In: *IOP Conference Series: Materials Science and Engineering* 1084.1 (Mar. 2021), p. 012026. DOI: 10.1088/1757-899x/1084/1/012026. URL: https://doi.org/10.1088/1757-899x/1084/1/012026.

[43] Weijie Su et al. “Vl-bert: Pre-training of generic visual-linguistic representations”. In: *arXiv preprint arXiv:1908.08530* (2019).

[44] Damien Teney, Lingqiao Liu, and Anton van Den Hengel. “Graph-structured representations for visual question answering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1–9.

[45] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[46] Nam Vo et al. “Composing text and image for image retrieval-an empirical odyssey”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6439–6448.

[47] Yaxiong Wang et al. “Position focused attention network for image-text matching”. In: *arXiv preprint arXiv:1907.09748* (2019).

[48] Qi Wu et al. “Image captioning and visual question answering based on attributes and external knowledge”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2017), pp. 1367–1381.

[49] Jianwei Yang et al. “Graph r-cnn for scene graph generation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 670–685.

[50] Xu Yang et al. “Auto-encoding scene graphs for image captioning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10685–10694.

[51] Rowan Zellers et al. “Neural motifs: Scene graph parsing with global context”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5831–5840.

[52] Ji Zhang et al. “Graphical Contrastive Losses for Scene Graph Generation”. In: *CoRR* abs/1903.02728 (2019). arXiv: 1903.02728. URL: http://arxiv.org/abs/1903.02728.