FINAL MASTER'S DEGREE THESIS

MASTER IN ARTIFICIAL INTELLIGENCE (MAI)

COMBINING ONTOLOGIES AND SCENE GRAPHS TO EXTRACT ABSTRACT ACTIONS

Author: Miquel Buxons Vives

Director: Javier Vázquez Salceda

Co-Director: Sergio Álvarez Napagao

Defense: October 2022

FACULTAT D'INFORMÀTICA DE CATALUNYA (FIB)

UNIVERSITAT ROVIRA i VIRGILI

UPC

UNIVERSITAT DE BARCELONA

The deductive method is the mode of using knowledge, and the inductive method the mode of acquiring it.

HENRY MAYHEW

Abstract

Keywords: Scene Graph Generation, Common-sense knowledge, WordNet, Computer Vision, Inductive reasoning, Deep Learning, Generalization, Knowledge Graphs.

Make a machine to understand what is happening on a visual recording is one of the most ambitious goals pursued by the Computer Vision research field. This incredibly complex task requires the application of several different techniques, from the object detection, through the definition of the relationships that these objects can have in the scene, including the application of knowledge that allows adding different sets of relationships to compose abstract and compound actions (such as washing clothes or getting ready to go out on the street).

In this context, Scene Graphs techniques have been proposed in literature. Their approach is to capture the different relations that appear in a scene with the aim of aggregating them inside a graph which allow us to define a visual scene. Nowadays, the state-of-the-art methods hardly rely on prior knowledge extracted from the training step, this knowledge is clearly biased into the training set. Because of that, Scene Graph Generation models have a hard time correctly defining relationships between previously unseen objects. In recent years, a branch of models has emerged that attempt to apply common-sense knowledge techniques to try to lower the dependency of Scene Graph Generation models on prior bias.

This project describes and tests the most recent Common-sense techniques applied to scene graph generation, and then proposes a new technique: Generalized Action Graphs (GAG). The work also implements a recently published metric that allows measuring the generalization of a Scene Graph Generation model.

Acknowledgements

Firstly, I would like to thank my tutors Javier Vázquez-Salceda and Sergio Álvarez for all the support they have been giving me during the project, helping me to get the focus of the project, as well as helping me with the problems that have arisen in the technical part of the project. Thanks to them, I have also had access to the resources of the Barcelona Supercomputing Center (BSC) that have been vital for the experimental part of the project.

Also, I want to thank all the lecturers who have taught us during these years of master's degree. I can assure you that each one of them have been great teachers who put a lot of effort into making their subjects interesting for us. At the same time, I would also like to thank my colleagues on the master's degree, who made up a great working group, without whom everything would have been much more difficult.

To finish, I would like to thank my family and friends, who through their unconditional support helped me overcome the many difficulties that have been occurring. But above all, I would like to especially thank Camila Ramírez, whose tireless support has been essential, really very grateful.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

aGCN Attentional Graph Convolutional Network.

AI Artificial Intelligence.

ANN Artificial Neural Network.

AP Average Precision.

BiLSTM Bidirectional Long Short-Term Memory.

CNN Convolution Neural Network.

Faster R-CNN Faster Region-based Convolutional Neural Network.

FPN Feature Pyramid Network.

GAG Generalized Action Graphs.

GB-NET Graph Bridging Network.

GNN Graph Neural Network.

HICO-DET Humans Interacting with Common Objects for Detection.

HOI Human-Object Interaction.

HOMAGE Home Action Genome.

HORL Human-Object Relation LSTM.

IoU Intersection over Union.

KERN Knowledge-Embedded Routing Network.

LSTM Long Short-Term Memory.

MLP Multi-layer Perceptron.

MOO Multi-objective optimization.

mPD mean Precision Degradation.

NLP Natural Language Processing.

PredCls Predicate Classification.

RelDN Relationship Detection Network.

RelPN Relation Proposal Network.

RNN Recurrent Neural Network.

ROI Region of Interest.

RPN Region Proposal Network.

SgCls Scene Graph Classification.

SgDet Scene Graph Detection.

SGG Scene Graph Generation.

ST-SGG Spatio-Temporal Scene Graph Generation.

TCB CogTree-based class balanced loss.

VDS Visual Distant Supervision.

VGG Visual Geometry Group.

Chapter 1

Introduction

Understanding what happens around us is a vital capacity that both animals and human beings constantly use. This activity, although it seems trivial due to the constant use we make, requires a set of extremely complex processes.

This complexity level comes to the fore when we try to incorporate this capacity into a machine. It is at that moment when we realize that processes such as object detection, based on images or videos, or the determination of relationships between the objects, are basic tasks so that the machine can develop the ability to explain what its environment is like.

In this context, Scene graph is a structural representation which can capture a semantic description by explicitly representing objects ('person', 'window', 'banana', etc.), objects attributes (such as the material of an object), and the relations between paired objects ('jumping over', 'walk through', etc.) [70]. Scene Graph Generation (SGG) refers to the task of automatically mapping an image or a video into a semantic structural scene graph.

Scene graph generation (SGG) models take a visual input (e.g., an image or a video) and generate a scene graph which explains the scene appeared in the visual input. This generated scene graph can be used, for example, to automatically generate the image caption, or giving a query, use the information about the objects (nodes) and the relations between them (edges) to respond the question answered.

SGG, as a field of research is very recent, but currently it is one of the Computer Vision fields that the research community has the spotlight on, having big advances in the last years. Being SGG a quite recent topic, sometimes relevant aspects when we are evaluating an Artificial Intelligence (AI) model can be neglected in pursuit of an accuracy improvement. We could consider one of these aspects the **generalization**, which is, in SGG field, the capacity of a model to relate an action that is usually applied to

a specific object, to another object with a certain degree of semantic, or visual, similarity.[70]

Nowadays, the current state-of-the-art methods do not generalize appropriately [32]. We can associate this lack of generalization to the fact that some of the current techniques contain a large prior bias, since they tend to directly use as previous knowledge, the probability between actions and objects directly extracted from the relationships specified as ground truth in the training dataset, either in the form of a probability matrix [69], as in graph form [45].

1.1 Project scope

This project seeks to demonstrate that this generalization problem exists, analysing some of the most recent models to evaluate this capacity. We also aim to demonstrate that, by using previous knowledge stored in external knowledge bases and the ground truth labels available for training, we can infer a certain degree of generalization that will allow the reduction of the previous model bias.

The project's approach belongs to the category of common-sense knowledge applied to SGG (explained in depth in section 2.2.2). Briefly, these techniques try to reduce the bias of the current models by applying different techniques or heuristics that can use both the data from the training samples and use external knowledge databases. More concretely, our approach tries to generate the action-associated objects hierarchy tree by extracting the relevant super classes of the objects.

1.2 Objectives

The aim of this work is the proposal of a new common-sense knowledge generalization method to avoid the current previous knowledge bias that exists in some methods, adapting it to RelDN method [69], trying to give a better reasoning to the model. RelDN is one of the SGG state-of-the-art. We want to compare directly the effect of the proposed method over the model's performance and capacity of generalization.

Another objective of this work is the analysis of the generalization. In order to apply a qualitative analysis on the knowledge generalization in SGG, in this work the implementation of a recently published metric by [32] is used (the metric is deeply explained in 4.7)

1.3 Hypothesis

Therefore, having in clear the project scope and objectives, the main questions that the thesis wants to study are:

- **Hypothesis 1:** *The state-of-the-art SGG methods have a low level of object generalization.* In other words, the current models work with a huge bias over the most used objects giving an action in Human-Object Interaction (HOI).
- **Hypothesis 2:** *Using the semantic hierarchy of the objects in the dataset, this object generalization per action can be improved, not reducing the model accuracy.* The usage of semantic information, in our case the inherited hyperonyms, can help the model to tackle rare, or unseen relations between a person and objects.
- **Hypothesis 3:** *Giving a richer input data to GAG method in each action analysed, it better generalizes the final SGG model.* Combining the graphs generated in GAG method for each dataset increases the HOI examples analysed per action, producing a better generalization to the final model.
- **Hypothesis 4:** *Our technique can help to the model generalization, such as other state-of-the-art common-sense techniques.* Apart from the evaluation of the technique presented in this project, we will compare the new technique results to the ones obtained by other common-sense well known techniques.

1.4 Contributions

The master's thesis main contributions are:

- A new common-sense technique, Generalized Action Graphs (GAG) which uses WordNet object hierarchies and the dataset training ground truth labels to build a generalization graph per action with the purpose to add extra previous information to feed the model. An analysis of the effects that this new technique has on the SGG methods is done.
- An implementation of the mean Precision Degradation (mPD) proposed metric ([32]) to use it for the analysis of the object generalization techniques in SGG

A minor contribution of the thesis is the analysis of the currently most used Common-sense methods effects in the SGG metrics results.

In addition, this project is inspired by another method proposed in January 2022 by another classmate, Kevin Rosales. The method proposed by Rosales (HORL)[1], focuses on

[1]Kevin Rosales' HORL method is described in Section 3.2.5 of this document

ST-SGG methods, which are based on videos as inputs, having better results than other state-of-the-art methods, and this thesis uses it as a baseline to see the current generalization of this method.

1.5 Overview

This first chapter of the thesis has introduced the the field of study and motivations of the work, its objectives and expected contributions to the computer vision and knowledge community.

In Chapter 2, a detailed explanation of SGG is provided, jointly with an explanation of other important concepts such as Knowledge Databases. In Chapter 3, a deeper study of the related work is done, analysing the important methods for the main concept of the thesis, such as SGG methods or state of the art common-sense methods.

Chapter 4 describes the methodology followed by the TFM, e.g., datasets used in each part of the work, baselines, methodology followed by to create the GAG technique or the methodology followed to run the object detector and Scene Graph Generator. In Chapter 5 all the results extracted from the evaluated methods are exposed, giving a quantitative analysis of them.

Finally, in Chapter 6, the conclusions extracted after the realization of the work are described, adding an after work hypothesis achievement analysis and a future work section with some issues and lines of work that either are out of the scope or there was no time to tackle them properly, but would be of interest..

1.6 Chapter Summary

In this first chapter, a brief introduction to the thesis topics is done, explaining the project scope over the lack of generalization that currently affect the Scene Graphs field of research (1.1), probably due to the short time life that has this field. Then, the main motivations of the thesis are explained (1.2), giving rise to the hypothesis that this work wants to analyse (1.3). Finally, the contributions of the work to the SGG field of research are exposed and a general overview of the document chapters is done (1.4).

Now, it is time to explain the technical background that gives the basis to this work, explaining the SGG field of research more in deep and describing the methods in which generalization is studied.

Chapter 2

Technical background

To completely understand the project motivation and work done, firstly it is necessary to explain the technical background of each research field relevant for this work. This chapter is devoted to explain the related fields of research, starting with an extensive definition of the Scene Graph Generation (SGG) and all its subparts (Section 2.1), Knowledge Databases (Section 2.4), work-context important Deep Learning concepts (Section 2.3), and a brief context of multi-objective optimization (due to it being used in GAG, see section 4.2).

2.1 Scene graphs

A scene graph is a structured representation of an image, which can capture the semantics that describe the picture by modelling objects, subjects, relations between objects and the possible attributes that can distinguish an object from another [70].

Subject and Objects are considered the core block of the scene graphs, and they can be directly located in the image, normally inside a bounding box. Each object can be described with zero or more attributes (such as colour, material, position, etc). Relations can be actions (e.g., swim), spatial positioning (e.g., is behind), prepositions(e.g. "with"), descriptive verbs(e.g., wear), etc.

We can distinguish two main formats of scene graph, both of them represented as the composition of different triplets formats: [*subject, relation, objects*] or [*object, is, attribute*]. The verb "*is*", between objects and attributes, is for uniformity, a convention. Specifically, we can consider Human-Object Interaction (HOI), as a scene graph's subtype in which the subject of each relation is human, forming a triplet such as [*"person-like label", relation, objects*], e.g. labels such as woman, man, person or child are labels that can be considered as person-like labels. Figure 2.1 shows a simple example of scene graphs, concretely the

scene graph generated by the composition of three Human-Object interactions.

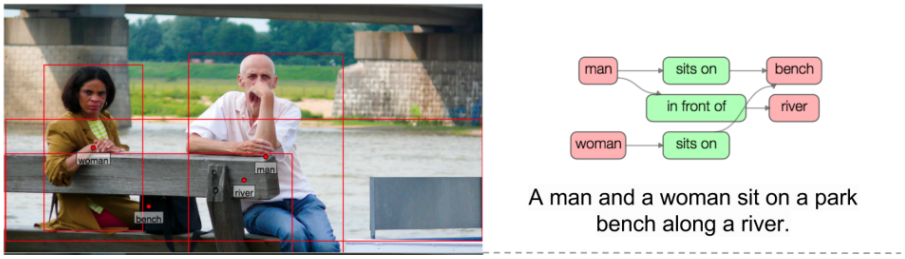


Figure 2.1: Simple scene graph example. [28]

From the point of view of graph theory, a scene graph is a directed graph with three types of nodes: **Objects**, **Subject** and **Attributes** (represented as *nodes* in the graph). The nodes are connected through **Relations** (represented as *edges* in the graph). In some representations, objects' attributes are not represented as nodes, instead they are incorporated as a description list of elements inside the Object node. This fact only affects the final representation of the graph and it makes no difference in the scene graph building process.

Due to the fact that a scene graph usually aims to semantically establish a representation of a real world image, we can fall in the erroneous association between Scene Graphs and Knowledge Graphs. While Knowledge Graphs are representations of multi-relational data with huge amount of fact triplets, Scene Graphs aim to semantically represent the visual relationships of an image. The knowledge extracted from these visual representations highly differs from the ones found in social networks and knowledge bases, as the relationships generated from an image are incidental, image-specific and are not intentionally constructed.

Focusing on the possible sources from which we can generate scene graphs, we can distinguish three main sources: firstly, and the most usual, we can generate a scene graph from a 2D image, which is the representation of a 3D space, this fact adds a difficulty level, the perspective of the photo, which can effect for example in the detection of the elements of the scene if we find occlusions in it. Another source is a 3D mesh that captures a static scene in 3D. Finally, we have videos where the visual relationships are not instantaneous, but may change through the time. In fact, we can see digital videos as a set of images (also called frames), which means relations can be spanned over different frames. The scene graphs build based on videos is also referred to as *Spatio-Temporal Scene Graph*.

2.2 Scene graph generation

Scene Graph Generation is the task of generating a visually-grounded scene graph that most accurately correlates with an image. It aims to parse images, or sequences of images (in the case of videos), to generate a structured representation that bridges the gap between visual and semantic perception, with the intent to build a complete understanding of visual scenes [70].

The generation of scene graphs is usually treated as a bottom-up process, where entities are grouped into triplets, and these triplets are joined to other triplets to form the complete scene graph. The starting point in the Visual Relationship Detection is the article presented by *Lu et al.* [33]. Another catalyst for the research in the field was the release of the first large-scale scene graph dataset Visual Genome [28].

Nowadays there exist two ways to build an SGG system. The mainstream approach is the one that uses a two-step pipeline: a first one that detects and locates inside bounding boxes the objects of a scene, and then solves a classification task to determine the relations between two objects. The alternate approach is to join the objects inferring and their relationships based on the object region proposals. But both approaches have in common the sequence of actions to be done:

1. Detect all the objects in the scene.
2. group them into pairs.
3. use the features of their union area.
4. infer the predicate (relation) between them.

In the case of SGG methods for 2D input that use a two-step pipeline, this step sequence can be is applied (as shown in Figure 2.2), and the following sections 2.2.1, 2.2.2, 2.2.3 and 2.2.4 will explain each step. In the case of SGG using videos as input, the pipeline is different, and therefore section 2.2.5 will introduce Spatio-Temporal Scene Graph Generation.

2.2.1 Object Detection

Object Detection is composed by an Off-the-shelf object detector. It is responsible for detecting the different objects, subjects, and relations inside the image, placing them within a bounding box (asually defined by the coordinates of two opposite corners) and, therefore, locating said elements within the image.

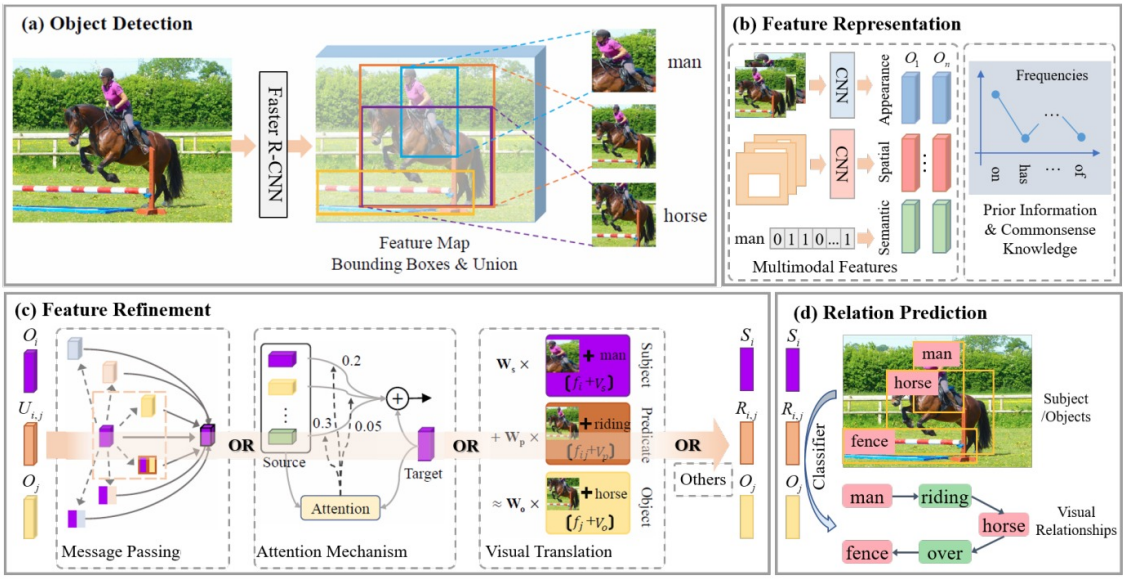


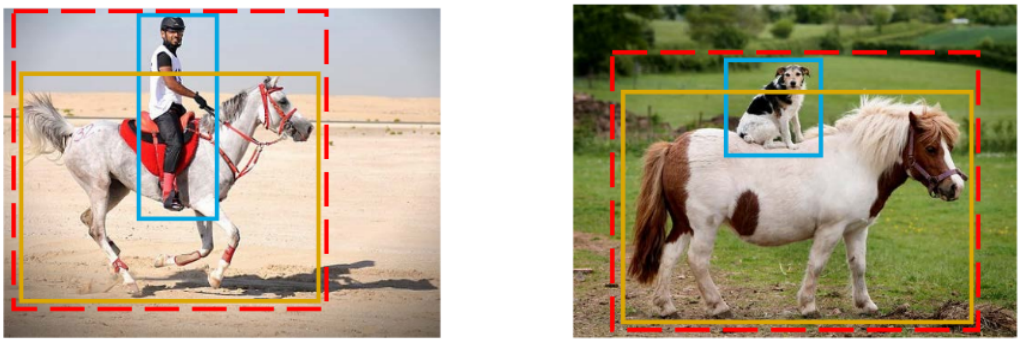
Figure 2.2: Scene graph generation methods framework overview [70]

Nowadays, the most widely used model is Faster-RCNN, which generates such object detection through Regions of interests (ROIs) [63]. Faster RCNN is the architecture chosen for the project's object detector, and it is further explained in section 3.1.

2.2.2 Feature Representation

The Feature Representation step aims to transform the labelled ROIs received as input, into the form of bounding boxes with associated feature labels. Transforming the bounding box content to a visual feature by, e.g., using a Convolution Neural Network (CNN) would not be enough to build a reliable relation predictor most of times. In order to build sufficiently consistent labels, we need to build a set of different features, also called **Multimodal Features**. In Scene Graph Generation, the mainstream feature types used are:

- Appearance features: these features are the ones mentioned just above, they are extracted from the part of the image inside the object/subject bounding boxes.
- Spatial features: features extracted to analyse the relative position of the object and subject inside the image.
- Contextual features: encoded using the information extracted from the rest of the photo outside the object/subject bounding boxes.
- Semantic features: these features differ from the rest as they are not visual features, but features that are extracted from the labels of each ROI gathered.



(a) man-riding-horse (b) dog-sitting on-horse

Figure 2.3: different ground truth triplets with similar spatial features. [70]

In Scene Graph Generation, data balance plays a key role in the performance due to the use of Deep Neural Networks (which are data-dependant structures). However, because of the long list of relations of the training sets, collecting enough training data images and labelling all the objects and relations becomes unfeasible due to its cost in time and resources. To tackle this issue, *Prior Knowledge* and *Common-sense Knowledge* strategies are used.

Prior Knowledge strategies

This Prior Knowledge strategies can be seen as "previous experiences" in human point of view. In this type of approaches the model is helped with data extracted from previous situations, it is widely used by the community, and we can distinguish two main categories:

- Statistical Priors: it is the simple way to use prior knowledge, the methods that use it calculate the possibility of a triplet, e.g., in [69], the statistical co-occurrences between pairs of objects and relations were calculated, another example can be found in [4], where authors train their semantic module using the entire triplet absolute frequency in the training data.

- Language Priors: the approaches behind these techniques use the semantics behind the triplet elements to calculate the probability of the triplet, in other words, even if the visual features extracted from Figure 2.3a and Figure 2.3b can be similar, we know that the verb "*ride*" is used in the case of humans as subject, but when in Figure 2.3b we see a dog on a horse, semantically the best relation could be "*sitting on*".

Common-sense Knowledge strategies

On the other hand, Common-sense Knowledge strategies search to include information about events that occur in time, about the effects of the actions, about objects and the perception of these objects. The research community has proposed different approaches to extract common-sense knowledge in order to refine object and relation features to improve generalizability of scene graph generation.

In the case of Common-sense Knowledge strategies, we do not want to add "previous experiences", as in Prior Knowledge strategies, we want to add "generalization capacity" to the model. There are three fundamental sub-issues of common-sense knowledge applied to SGG (see an overview schema in Figure 2.4):

- Information source: the source of the common-sense knowledge can be internal (extracted from the training samples), external (using external knowledge databases such as ConceptNet 2.4.2).
- Formulation: tackle the way to incorporate the knowledge in a more efficient and comprehensive manner. For example, using co-occurrence matrix, word embedding vectors or adding common-sense relationships triplets, that can be used to build a Common-sense graph.
- Usages: how we use the acquired structured knowledge to refine the original extracted features, the mainstream approach is to use this knowledge as a guidance in the original feature refinement, but, e.g., in [62], the authors demonstrated a framework which can train the scene graph models in an unsupervised manner, based on the knowledge bases extracted from the triplets of Web-scale image captions (Deeply explanation in section 3.3.3).

2.2.3 Feature Refinement

To achieve high-quality predictions it is necessary to achieve two objectives: on one side, information of the elements (subject, relation, object) must be integrated in order to refine the features, but on the other side, it is important to fuse the contextual information because the relations can be influenced by their surroundings. In [70], the authors propose a set of three different techniques used for this task:

- Message Passing: in this approach, the features and messages are passed between elements of a scene graph, including objects and relationships. The Message Passing technique can be applied locally, or **Local Message Passing**, where the propagation is inside the triplet elements. On the other hand, there are the global, or **Global Message Passing**, where the propagation is across all the elements detected in an image, in order to give a triplet context.

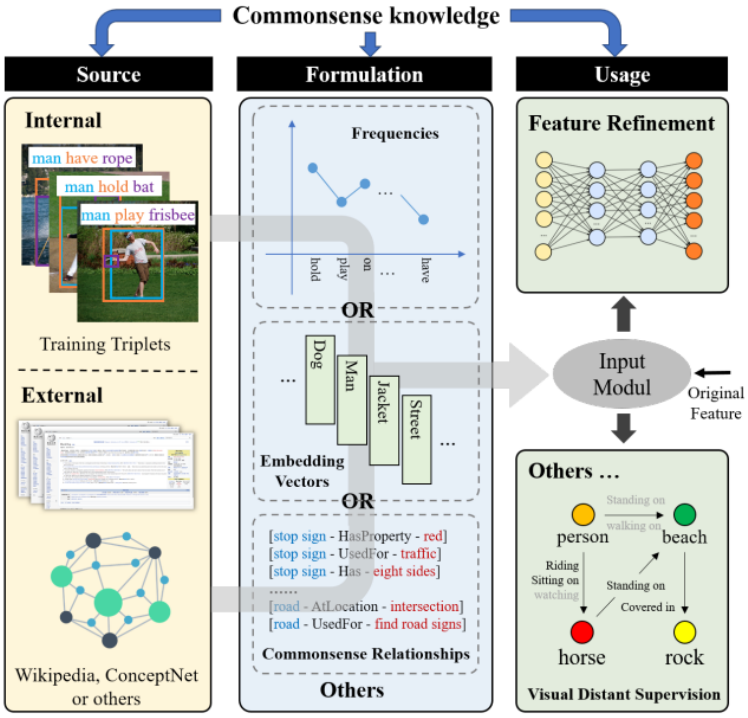


Figure 2.4: Common-sense Knowledge strategies graphical overview. [70]

- Attention Mechanism: attention can be used to update each object and relation representation, or integrating the contextual information. For this reason, we can distinguish between two types of attention mechanisms for SGG; **Self-Attention mechanism** which aggregates multimodal features of an object to generate a more comprehensive representation, and **Context-Aware Attention** which learns the contextual features using graph parsing.

- Visual Translation Embeddings: in this approach the objective is to learn a compositional representation for each element of the triplet (subject, object and predicate) by learning separate visual-language embeddings spaces. Each triplet element is mapped close to the language embedding of the associated label. Generating each triplet element visual-semantic embedding, the architecture can learn a visual translation vector in order to the prediction.

2.2.4 Relation Prediction

After feature refinement, we have to train a classifier in order to make it capable to use the features, extracted from feature refinement step, to predict the relation between each pair of objects.

2.2.5 Spatio-Temporal Scene Graph Generation

Compared to images, videos provide a more natural set of features for detecting visual relations, due to the dynamic nature of the interactions between objects in the real world. [70]

Using videos as input, we are able to detect dynamic actions as "*put down*", or even recognize correlated actions, such as in the context of a supermarket, the triplets [*woman*, *pay*, *money*] can have a correlated action "*buy*", e.g. [*woman*, *buy*, *orange*]. Additionally, videos can allow us to disambiguate easily between similar actions in images, e.g., "*run*" and "*walk*".

From the technical point of view, nowadays Spatio-Temporal Scene Graph Generation (ST-SGG) relies on video object detection, which is a composition between image-based object detection and multi-object tracking. After obtaining the object/subject trajectory, e.g. in [47], a relationship feature extraction step is done in order to obtain the object features over the time. After it, a relationship modelling technique is applied. An example of video visual relations can be seen in figure 2.5.

But there are some problems that raise when we are using videos as input (that in images are not frequent), which downstream video relationship detection, such as blur, dynamic occlusions or camera motions.

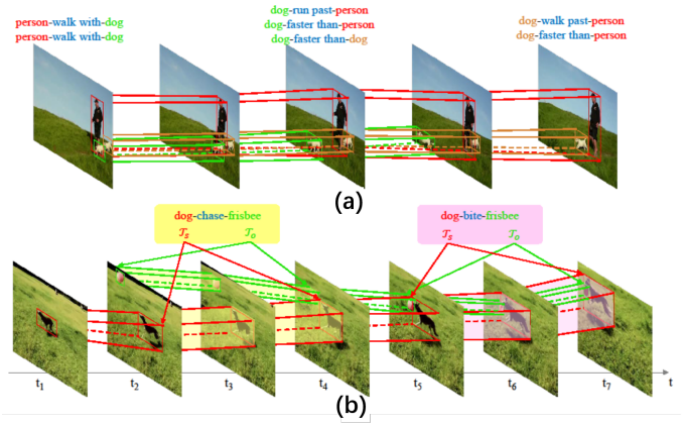


Figure 2.5: Example of video visual relations. [47]

2.3 Deep Learning for SGG

This section is devoted to define some Deep Learning (DL) concepts related to some of the project's key points. It begins with the basic ANN (see in Section 2.3.1) to set the bases of the other described Neural networks, CNN (Section 2.3.2) and LSTM (Section 2.3.3). Then it continues with the attention mechanism (Section 2.3.5) and Message Passing technique (Section 2.3.6).

2.3.1 Artificial Neural Network

Artificial neural networks [59] are a technology based on studies of the brain and nervous system. In fact, these networks only use a reduced set of concepts from biological neural systems, simulating the electrical activity of the brain. As a network, it is a composition of simple elements "*neurons*", also called perceptrons. These "*neurons*" are connected between them, having inputs received from other neurons and output that they generate after the application of a weighted sum (see Equation 2.1) where n is the number of inputs, x_i are the inputs and w_i are the weights.

$$f = (\sum_{i=0}^{n} w_i x_i) \tag{2.1}$$

ANN have become popular due to their capability of learning representations of data by learning their set of weights. Specifically, a type of ANN is the Multi-layer Perceptron (MLP)[41] which is a feedforward ANN where the neurons are organized in layers, and the output of the neurons of a layer is the input of the next one.

Backpropagation algorithm [42] is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous iteration in a supervised manner. [25]

2.3.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) [38] is a Deep Learning (DL) algorithm architecturally-based on Artificial Neural Networks (ANN) that is commonly used for image-based DL. CNNs are widely used in image based DL, instead of regular ANN, because CNN are able to extract local patterns of data, using kernels [43].

These kernels are matrix filters applied on the images, understanding images as a big 2D, or 3D (counting the colour channels) matrix. After applying a set of convolutional layers, we extract the features of the structures that appeared in the image. Finally, we apply a few fully connected layers in order to build a classifier, that uses the features created to assign a category over a set of categories (see Figure 2.6).

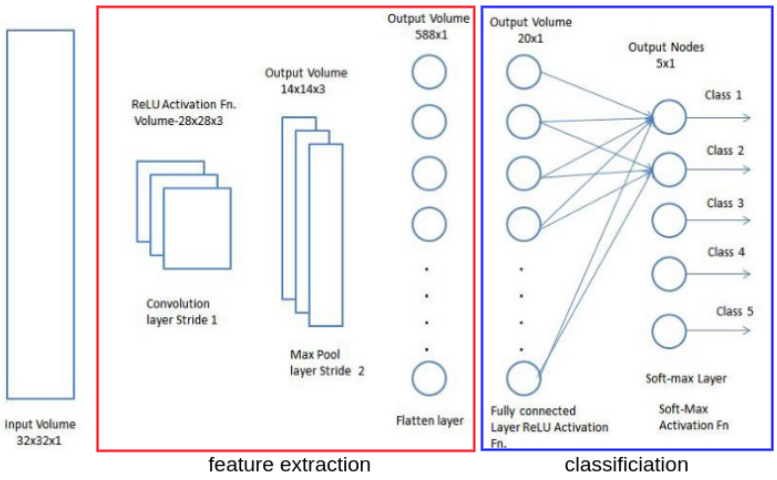


Figure 2.6: CNN basic structure example. [38]

2.3.3 Long Short-Term Memory networks

Recurrent Neural Networks (RNN) are a subclass of ANN where the output of the previous time step is used as part of the input of the next iteration. They keep information about past inputs for an amount of time (not fixed) depending on the weights of the input data. [7]
These properties make RNN very useful when processing sequences (text, video), as it

is able to take into account previous perceptions (previous words or sentences, previous video frames) to process the current perception and also give a certain resistance to the noise.

However, RNN are algorithms that require a big ammount of time and data to train. Furthermore, they suffer from the *vanishing gradient problem*, which occurs when, after some training iterations, the gradient will be vanishingly small.

Long Short-Term Memory (LSTM)[21] networks are a type of RNN that solves the two main issues that vanilla RNN have. On one side, LSTM reduce the *vanishing gradient problem* and, on the other side, LSTM are more computationally efficient.

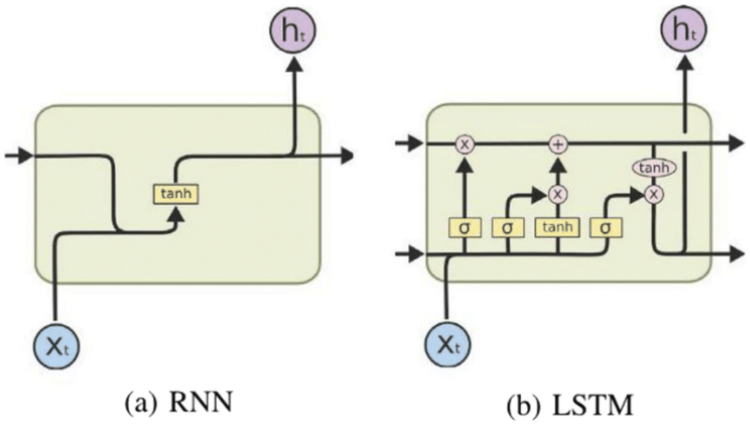


Figure 2.7: Differences between RNN and LSTM cell structure. [5]

As it can be seen in Figure 2.7, RNN and LSTM has similar architecture, having an input gate, an output gate and a forget gate. The main change is internal calculation flow structure, which is much more complex in the case of LSTM.

In the case of a RNN cell (Figure 2.7(a)), it basically consists of an activation function:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{2.2}$$

Where W is weight, h is the single-hidden vector, W_{hh} is the weight at previous hidden state, W_{hx} is the weight at current input state, $tanh$ is the Activation function, that implements a Non-linearity that squashes the activations to the range[-1,1]. The structural simplicity of RNN cell has as drowbck that it struggles to remember long term data.[6]

On the other hand, each LSTM cell (Figure 2.7(b)) has a more complex internal structure that can be divided in three main stages[6] (which are show in more detail in Figure 2.8):

- Forget stage, or gate: It seeks to discover what details can be discarded from memory

using a sigmoid function. Look at the previous state, at the input and generate an output with a range between 0 (can be omitted) or 1 (important data, save this).

- Input stage, or gate: It seeks to discover the input values that should be saved, using a sigmoid function. The *tanh* function determines the weights of each value in a range between -1 and 1 depending on its importance.
- Output stage, or gate: To decide the output, the input received and the current state of the block are used. A sigmoid function decides which values to pass, the result is multiplied by the result obtained in a *tanh* function that gives the weights to the input values defining their importance, thus obtaining the output.

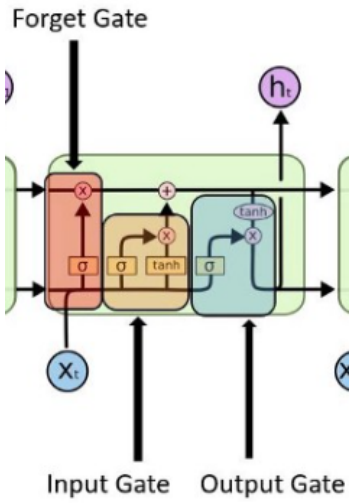


Figure 2.8: LSTM cell structure, divide in 3 different stages. [6]

2.3.4 Encoder-Decoder architecture

Encoder-Decoder architectures are based on three main actors[35]: (1) the encoder is the actor that receives the input and generates an abstract representation of the input, (2) hidden state, is the abstract representation of the input received, and (3) the Decoder that will transform the abstract representation into the desired output. This structure is widely used, e.g., in tasks such as sentiment analysis, image captioning or language translation.

If we go deeper into the concepts explained above, to design the architecture of the encoder we would have to see what input we want to process, e.g., in the case that the input is an image, a possible architecture would be a CNN to transform the image into a vector of features. This vector of features would fulfil the role of abstract representation,

which the decoder would later transform into the output.

Finally, to define the architecture of the decoder it is necessary to define the output that is required, e.g., if we want to process that vector of features and convert it into a caption of the image used as input, our architecture could be an RNN that generates a sequence of words using the features.

2.3.5 Attention Mechanism

The attention mechanism[3] was introduced to improve the performance of the encoder-decoder model(see in section 2.3.4) in the machine translation field. But a vanilla encoder-decoder, with only one encoded feature vector, suffers with long vectors of sequences. Intuitively, it can be seen that, as more updates are made to the same vector, the higher the chances are the earlier inputs and updates are lost.[12]

The main idea behind attention is to allow the decoder to utilize the most relevant parts of the input sequence. For this reason, the attention approach encodes each input element of the vector separately, weighting each input with the relevance, with the most relevant vectors being attributed the highest weights, being different for each output step. (see in the equation 2.3)

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) * V \tag{2.3}$$

Where Q is the Query, K is the Key and V is the value. We can see attention as a function that maps a Query to Key-Value pairs, and the result is a set of weighted values that describe which key-value is more important to query. The result is normalized using a softmax function, finally a dot product is applied between normalized weights and values.[1]

2.3.6 Message Passing Mechanism

Message passing mechanism [14] was born in the context of graph neural networks (GNN). These networks receive as input graphs composed by nodes (with a feature vector) and edges between them. Graphs differ from other data processed by the other neural networks (e.g., images or sequences of words) because in graphs the nodes are connected, this means that the input data is dependant. This issue makes unfeasible the graphs processing by other kind of neural networks because input data independence is a huge assumption that they require. [30]

Duvenaud et al. (2015) [14] propose the use of Message Passing to extract valuable information from graph molecules and then transform it into a single feature vector. Message passing is used to exchange information between adjacent nodes, this information from

other nodes are summed and then mixed with the current state of the node. This procedure is repeated over a certain number of steps, finalizing with the creation of a final feature vector describing the whole graph. This feature vector can be then used as input to a standard machine learning model.

2.4 Knowledge Databases

As we mentioned in section 2.2.2, Common-sense Reasoning techniques can use external knowledge databases to infer, or generate, new knowledge to feed the model with. In this section we will describe the main knowledge databases which are used in several different approaches, including Natural Language Processing (NLP) approaches, including the knowledge database used in our project (WordNet).

2.4.1 WordNet

WordNet [34] [36], is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct lexicalized concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser [58]. WordNet is a free service that can be consumed thwough an API, and it can also be downloaded directly from the official web [57] to use a local instance.

WordNet superficially resembles a dictionary, grouping its words together based on their meanings. However, there are some important distinctions. First, WordNet is not focused on connecting the words (the letter strings), instead it focuses on interconnecting the senses of the words, inducing a semantic disambiguation. Second, WordNet presents these relations between words with a variety of semantic labels.

WordNet's most important relation is synonymy (e.g., box and package). These synonyms, or words that have the same meaning in a sentence, are grouped in sets (synsets). Each one of these sets are linked to other synsets using conceptual relations, such as, hyperonymy, hyponymy or domain term categories. Additionally, each synset has a brief definition and, almost all of them, have a small sentence as usage example.

In image 2.9, we can see an example of a search in WordNet browser [58]. It distinguishes the different synsets that the word car can belong, and inside the first synset, and we can observe the different relations available, jointly with the definition and usage example sentence.

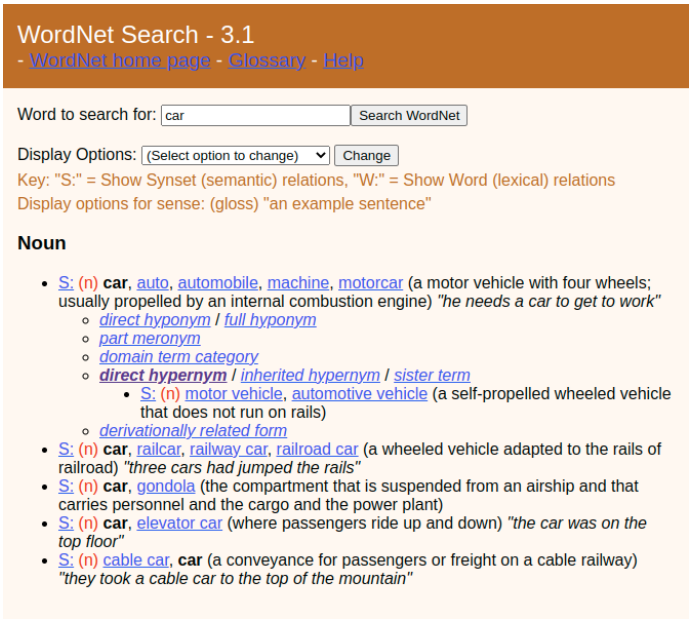


Figure 2.9: Example of WordNet browser retrieved information.

2.4.2 ConceptNet

ConceptNet is a knowledge graph that connects words and phrases of natural language (internally called *terms*) with labelled, weighted edges (internally named as *assertions*). This freely-available semantic network has the aim to helps the computers understand the meanings of words that people use.

ConceptNet is originated from the crowdsourcing project Open Mind Common Sense, launched by MIT Media Lab in 1999. It has grown to include knowledge from other resources: [1]

- **Open Mind Common Sense (OMCS):** the knowledge extract of this project was the first source of knowledge.
- **WordNet:** a large lexical database of English. Described previously in section 2.4.1.
- **Wiktionary:** a free-content multilingual dictionary that aims to describe all words of all languages using definitions and descriptions in English.
- **Games with a purpose:** a website that hold a set of "*game with a purpose*" in it.
- **JMDict:** a Japanese-multilingual dictionary.

[1] List of sources extract from *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge* [51].

Figure 2.10: Example of ConceptNet browser retrieved information.

- **OpenCyc:** a system that represents common-sense knowledge in predicate logic.
- **DBPedia:** a network of facts extracted from Wikipedia infoboxes.

Like WordNet, ConceptNet has been distributed in different ways: it can be downloaded as GitHub repository ([16]), directly as a Python library that can be installed using PyPi ([26]), or can be accessed in an online browser ([11]). Image 2.10,shows the information retrieved after questioning for the concept "*Dictionary*".

2.4.3 Google Knowledge Graph

Google Knowledge Graph is a knowledge database base directly structured as a graph. It allows to build relevant information in an infobox beside its search results (see an infobox retrieve example in Figure 2.11). The data is generated automatically from a variety of sources covering a set of categories, such as books, events, persons, movies, etc. (see the complete entity list in [23]).

This database allows the user to find the required information even before entering the source site, but this fact, and other database key points, have been the target of some criticism of the project, as we will see below. Google Knowledge graphs grew exponentially since 2012, the year of database release, starting with the coverage of 570 million entities and 3.5 billion facts [52]. By May 2020, the database had grown to cover 500 billion facts on 5 billion entities [53].

Although Google gives public access to query the database, its structure is unknown and cannot be downloaded. This fact is just one of the criticisms of the Google Knowledge Graph [37], the main ones being:

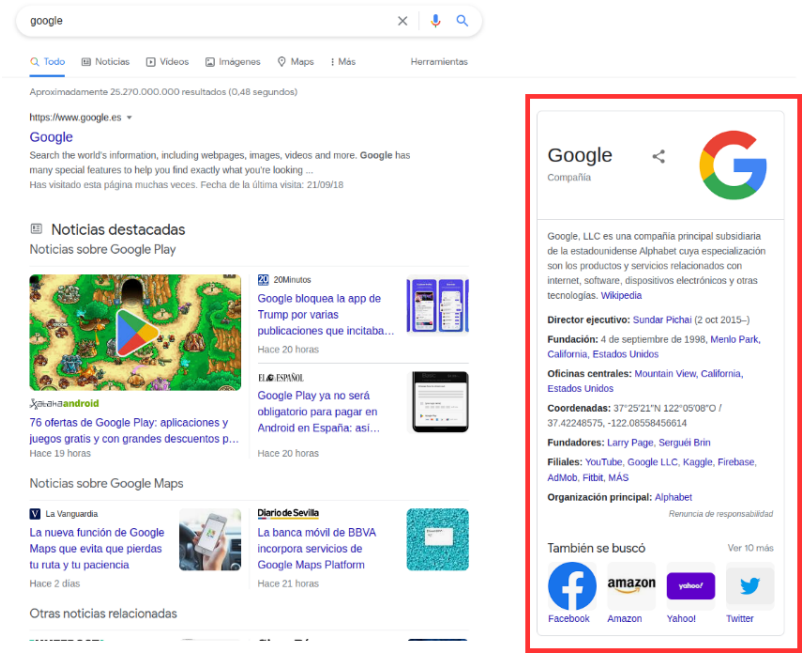


Figure 2.11: Example of Google Knowledge Graph usage, generating the infobox associated to a Google search engine query.

- Lack of publicly available structure information.
- Lack of source attribution, Google does not link the sources of the information retrieved.
- Declining Wikipedia article readership, Google search results caused significant readership declines for Wikipedia, from which the panels obtained some of their information, even Wikipedia is one of the main source of information of the database.
- Biased information, usually because of sourcing information from websites with high search engine optimization.

2.5 Multi-objective optimization

Multi-objective optimization (MOO) is an area of multiple criteria decision-making. It has been applied in many fields, such as economy, logistics, manufacturing or car industries. MOO is an optimization problem that, instead of having only one single function to optimize, it this case, more than one function have to be optimized [49].

In the case of single-objective optimization problem, the quality of a solution can be easily determined by comparing their values in the objective functions. On the other hand, in multi-objective optimization, we cannot compare directly the values of the different objective functions. We have to determine a solution goodness, determining the dominance. We can understand the dominance using a brief example; having two solutions A and B, A is no worse than B in all objective and A is strictly better than B in at least one objective.

Having the above definition in mind, we can create 2 different sets of solutions, *dominated* solutions (which have a better solution in the set of possibles solutions), and *non-dominated* ones. This *non-dominated* set of solutions is call **Pareto-optimal set**: inside this solution set, there is no *best solution*, if a solution A has better value than a solution B in a certain criterion ($c1_A > c1_B$), it is worst than B solution in other criterion ($c2_B > c2_A$). Often, the Pareto-optimal solutions can be joined by a surface, this boundary is called **Pareto-optimal front**. A representation of the explained concepts in a bi-dimensional space can be seen in Figure 2.12.

2.6 Chapter Summary

In this chapter, the relevant theoretical framework for this thesis is explained. Starting with an explanation of scene graph generation and its differentiable conceptual parts, which is the main topic framework of this TFM (2.2). Followed by the description with some well known Knowledge databases, such as WordNet (used in this work), ConceptNet or Google

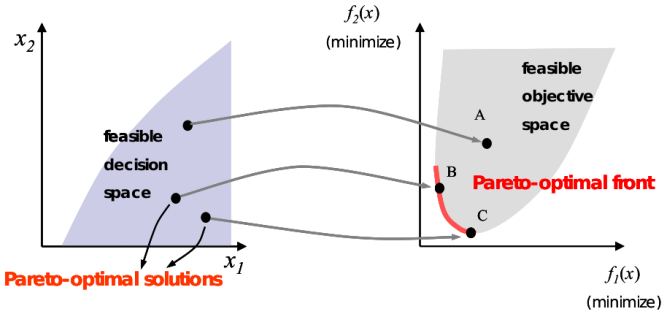


Figure 2.12: Pareto-optimal front in a bidimensional space representation. [49]

Knowledge Graph. (2.4) Finally, an introduction to Pareto front theory is presented (2.5), as it is used in the Generalized action graphs proposed technique.

In the next chapter, an analysis of the specific implemented methods for this thesis is given. More information on the SGG state-of-the-art methods (3.2) and the latests common-sense techniques used in SGG (3.3) will be provided, in order to give a better framework definition to the reader.

Chapter 3

Related work

This chapter is devoted to explain a set of state-of-the-art approaches to deal with Scene Graph Generation (SGG) and the latest Common-sense knowledge methods applied in SGG. All the approaches covered in this Chapter are relevant to the work made in the thesis (described in Chapter 4) to the experiments (described in Chapter 5), or both.

Section 3.1 introduces the Faster R-CNN algorithm. Then Section 3.2 provides an overview about the research on SGG for static environments. After that, an overview of the latest research in Common-sense knowledge methods is given in Section 3.3.

3.1 Faster RCNN

Region-based Convolutional Neural Network (R-CNN) [18] was an evolution of Convolutional Neural Networks (CNN)[1] based on a Selective Search, which generates around 2000 region proposals for each image, and then each region proposal is fed to the underlying network architecture. This approach was promising but highly time, power and storage consuming.

Fast Region-based Convolutional Neural Network (Fast R-CNN) [17] maintain the Selective Search, but it only fed the CNN once per image. In Faster R-CNN the region proposals generated by Selective Search are projected on to the feature map (one per image) generated by the CNN.

Faster Region-based Convolutional Neural Network Faster R-CNN, [40] wants to tackle the main Fast R-CNN bottleneck, the Region Proposer based on Selective Search. The new approach substitutes the Selective Search procedure with the application of the Region

[1] Convolutional Neural Networks are described in Section 2.3.2 of this document.

Proposal Network (RPN) approach.

Going more in detail, Faster R-CNN uses a pretrained VGG16 [50] network to generate the ROI and the image feature map, and, the RPN module is the responsible to generate the ROIs (in form of anchor boxes inside a blue box in Figure 3.1).

Firstly a set of fixed size slide windows are applied on the image in order to get a set of anchor boxes, then a binary classifier is used in order to distinguish the foreground boxes from the background boxes, eliminating the background boxes (pointed with a **1** in Figure 3.1). Then, a regressor is used in order to obtain the confidence classification of each region proposal and rank them (pointed with a **2** in Figure 3.1). Finally getting the top-K most confident anchor boxes.

After merging the ROIs generated by RPN, and the feature map, an object classificator is trained with the ground-truth object labels (pointed with a **3** in Figure 3.1). And a regressor is trained in order to generate the final object's bounding boxes (pointed with a **4** in Figure 3.1).

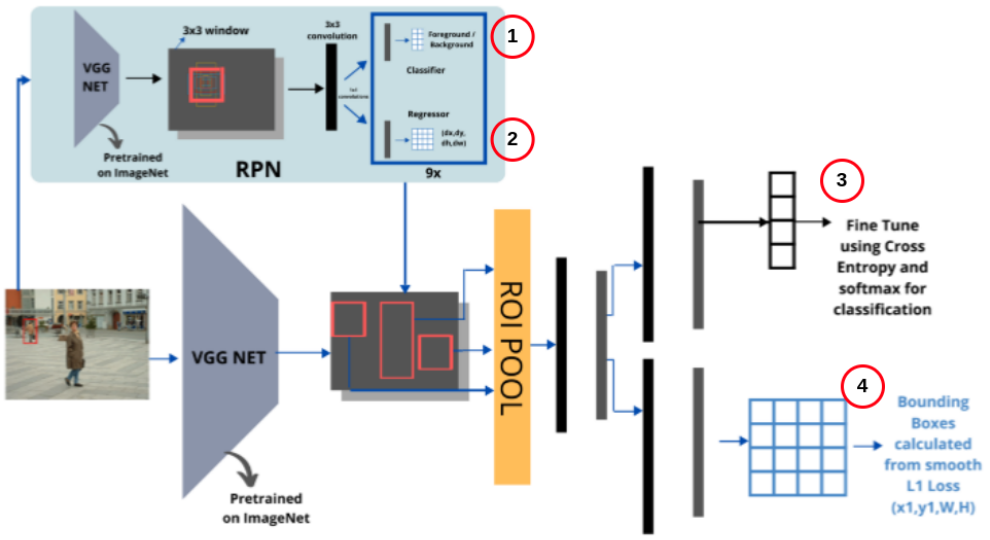


Figure 3.1: Faster R-CNN architecture overview. With the four losses generated pointed in red. [63]

3.2 Scene graph generation methods

In this section, the most relevant approaches of generating scene graphs from images and videos will be explained. Three image-based techniques are described: G-RCNN (Section 3.2.1), Neural Motifs (Section 3.2.2), Knowledge-Embedded Routing Network (Section 3.2.3) and Graphical Contrastive Losses (Section 3.2.4). Then a video-based method (HOLR) is described (Section 3.2.5).

3.2.1 Graph-RCNN for Scene Graph Generation

Yang et al. presented Graph R-CNN application for Scene Graph Generation (SGG) [61]. The method is based on the two-step structure for SGG methods [2]:

- Firstly an object detector step. In this case the authors used a Faster R-CNN as object detector[3].

- The second step is devoted to the SGG construction method, which is the one which stablishes the relations between the objects detected. The authors propose a three-phase technique, based on: (1) object extraction, (2) relationship pruning, and (3) graph context integration. (see the method overview in Figure 3.2)

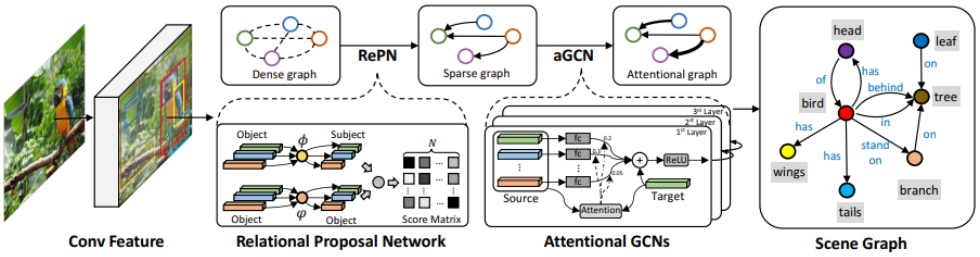


Figure 3.2: Graph-RCNN architecture overview. We can distinguish the three phases of the method jointly, with the Relation Proposal Network (RelPN) used to prune the graph connections and the Attentional Graph Convolutional Network (aGCN) used to capture the contextual information between object and relations. [61]

The first stage consists on the construction of a dense graph using the detected objects as nodes connected between them. After that, the authors propose the usage of a Relation Proposal Network (RelPN)[68] which aims to model the likely relationships between objects.

[2] SGG's two-step pipeline is described in Section 2.2 of this document.
[3] Faster R-CNN is described in Section 3.1 of this document.

To exploit these likely relations, they propose the usage of the soft class-relationships priors. In other words, they generate a 2D matrix with the probability of each pair of objects to be related, this probability can be calculated as:

$$f(p_i^0, p_j^0) = \langle \Phi(p_i^0), \Psi(p_j^0) \rangle, i \neq j \tag{3.1}$$

where $\Phi(\mathring{u})$ and $\Psi(\mathring{u})$ are projection functions for subjects and objects in the relationships respectively. They use two multi-layer perceptrons MLP with identical architecture for $\Phi(\mathring{u})$ and $\Psi(\mathring{u})$ followed by a matrix multiplication to extract the probability values. With this relation probability matrix, the dense graph is transformed to a sparse graph with the non-likely relations (edges in this case) pruned.

After obtaining the sparse directed graph, an Attentional Graph Convolutional Network (aGCN) [27] is used in order to integrate contextual information from neighbouring nodes and relations using the message passing technique [4]. In order to be able to apply the aGCN, a new graph is built where the nodes are the objects, the subjects and the relationships, adding connections between them using the previously created spare graph, additionally connections between objects are set to allow information to flow directly between object nodes. These additional connections are based in a previous work [22] which shows that reasoning about object correlation can improve detection performance.

Finally, the final method loss can be calculated as in Equation 3.2:

$$P(S|I) = P(V|I) * P(E|V, I) * P(R, O|V, E, I) \tag{3.2}$$

where S is the Scene Graph, I represents an image, V is the set of nodes corresponding to localized object regions in image I, E denotes the edges between objects, and O and R denote object and relationship labels respectively.

3.2.2 Neural motifs: Scene Graph Parsing with Global Context

In 2018, *Zellers et al.* presented a new approach in [67] where they propose the usage of neural motifs to generate SGG. They describe motifs as regularly appearing substructures in the scene graph.

Their approach, *MotifsNet*, uses a two-step schema (as the one explained in section 2.2), in which we can find a first step which consists of an object detector that extracts the object locations inside the images (extracting the bounding boxes). In the *MotifsNet* case, a Faster R-CNN[40] was chosen for this task.[5] The second step is the relation predictor, where *MotifNet* uses the different regions proposed previously by Faster R-CNN, these

[4]The Message Passing technique is described in Section 2.3.6 of this document.
[5]Faster R-CNN is described in Section 3.1 of this document.

regions are inserted into an architecture composed of a Bidirectional LSTM, a LSTM and a Bidirectional LSTM again. A complete method overview can be seen in Figure 3.3.

Going step by step, the first BiLSTM is used to codify the regions proposed by Faster R-CNN, this extracted regions are organized as a sequence, and then they are codified. The LSTM is used to decodify the object context vector created previously. With it, the method obtains the object class commitments.

Finally, the second BiLSTM is used to codify the edge context of each possible relation between the objects detected, it uses the object class commitments of each object of the pair for this task. With the pair context obtained, the probability of each object pair to be related is calculated.

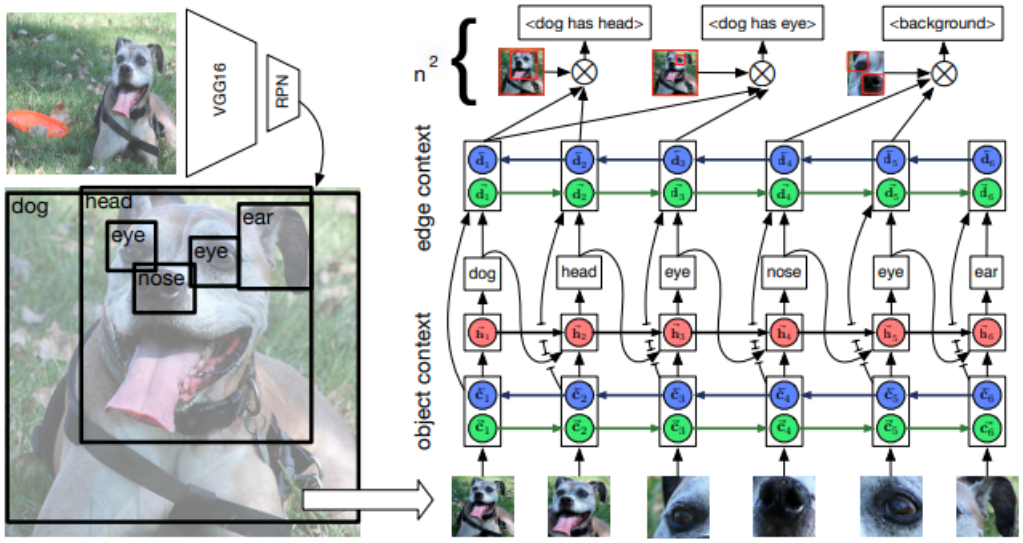


Figure 3.3: MotifNet overview. [67]

3.2.3 Knowledge-Embedded Routing Network for Scene Graph Generation

In 2019, Tianshui Chen et al. described in [10] another approach based on the generation of a knowledge graph in order to describe the co-occurrences between actions and objects.

Knowledge-Embedded Routing Network (KERN) is a two-step SGG method, which is based on the usage of an external object detector to receive a set of region proposals. These region proposals are used to build a graph, trying to learn a contextualized representation to

predict the class label for each region. For each object pair with predicted labels, the authors build another graph to correlate the given object pair with all the possible relationships and employ a graph neural network to infer their relationship.

In KERN the statistical information of object co-occurrence is stored in a graph, which associates the regions detected in the image with the predicted object label. For their approach, the authors build a co-occurrence matrix $M_c \epsilon \mathbb{R}^{C \times C}$ where C is the number of object categories. Then, given two regions b_i and b_j, the authors duplicate b_i C times to obtain C nodes, with node b_{ic} denoting the correlation of region b_i with category c. Finally, a Message Passing mechanism[6] is used to propagate the information between nodes. This built graph, is used in the green labels' step in Figure 3.4.

In the case of the relationships, and after obtaining the objects labels, another set of graphs is created in order to map the correlations' context of each *subject*, *object*, *relationship* co-occurrence. Finally, inferring the most feasible relationship. This step is denoted with blue labels in Figure 3.4.

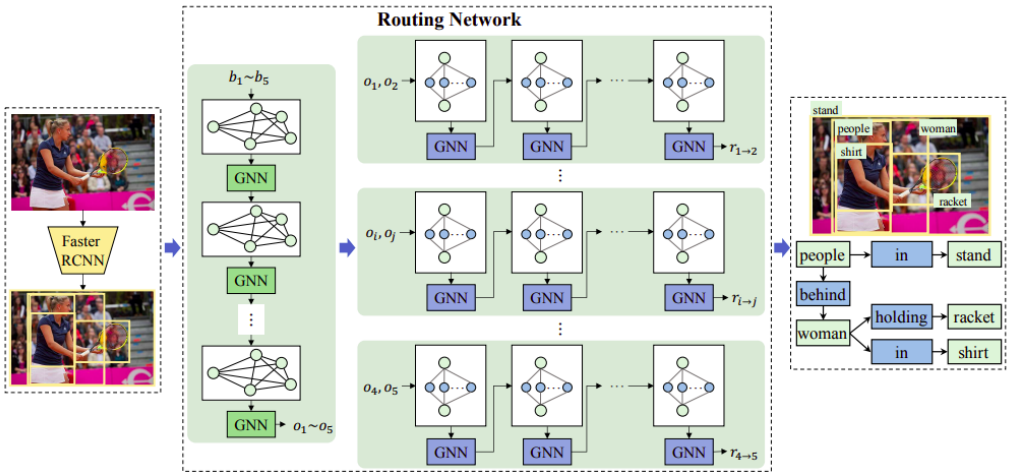


Figure 3.4: KERN overview. [10]

3.2.4 Graphical Contrastive Losses for Scene Graph Parsing

In [69] *Ji Zhang et al.* released the Relationship Detection Network (RelDN), a SGG method for static inputs (e.g., images). With this new method, the authors tackle two main errors that contemporaneous SGG methods fall in:

[6]The Message Passing technique is described in Section 2.3.6 of this document.

- Proximal Relationship Ambiguity: is the problem that occurs when multiple subject-object pairs appear very close one to each other in the image and interacting with very similar actions, (e.g., a musical band in which each member plays the instrument close to each other).

- Entity Instance Confusion: occurs when the subject or object is related to one or many instances of the same class, and the model fails to distinguish between the target instance and the others.

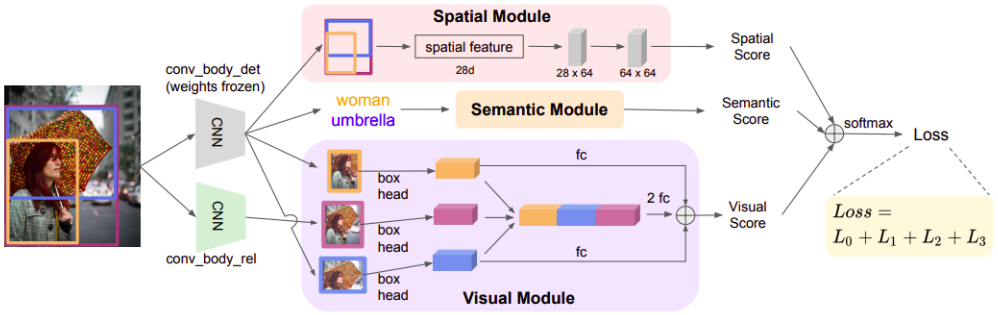


Figure 3.5: RelDN method overview. [69]

Coming up next, the Relation Proposal Network (RelPN) method functionality can be described with the following steps (see the overview in Figure 3.5):

1. The first stage of the RelDN exhaustively returns bounding box regions containing every pair of elements detected in the image.

2. In the second stage is devoted to the feature extraction. For this task RelDN uses two similar CNN (same estructure but with different parameters) each one used to: (1) detect the features of each entity of the pair (conv_body_det), and (2) learn features of the regions that strongly imply relationships. (conv_body_rel).

3. The third stage computes three types of features for each relationship proposal: semantic, visual, and spatial:

    - Semantic features: it conditions the predicate predictions on subject-object class co-occurrence frequencies. These frequencies are created using the ground truth annotations, creating a 3D matrix where for each pair subject-object the probability of a certain predicate is specified. It is inspired by Zeller, et al. [67] work.

- Spatial features: it conditions the predicate class predictions on the relative positions of the subject and the object. In order to encode the spatial information, the method uses the box coordinates from subject-object pair using the box delta and normalized coordinates.
- Visual features: it uses the subject and object ROI features from (conv_body_det) CNN, and the extracted predicate ROI features from (conv_body_rel). The three feature vectors are concatenated and passed through an MLP to attain the predicate class logits.

Finally, the method obtains the final probability distribution over predicate classes by using the three scores followed by softmax normalization:

$$p^{pred} = softmax(f_{vis} + f_{spt} + f_{sem}) \tag{3.3}$$

In order to tackle the above problematic cases, the paper propose a set of four contrastive losses which explicitly force the model to disambiguate the related and unrelated instances. Concretely, this set of losses are:

- Predicate classes loss (L_0): it is the simplest loss, and it consists on the cross-entropy loss over the predicate classes.
- Class Agnostic (L_1): It aims to maximize the affinity of the lowest scoring positive pairing and minimize the affinity of the highest scoring negative pairing.
- Entity Class Aware (L_2): it aims to address the Entity Instance Confusion by focusing entities with the same class. This loss maximizes the margins between instances of the same class.
- Predicate Class Aware (L_3): it aims to address the Proximal Relationship Ambiguity by focusing on entity pairs with the same potential predicate. This loss maximizes the margins within groups of instances determined by their associated predicates.

To conclude, the final loss is expressed as:

$$L = L_0 + \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3 \tag{3.4}$$

where L_0 to L_3 are the above-mentioned losses, and the λ_i are weights to adjust the influence of each loss in the final loss.

3.2.5 HORL

Human-Object Relation LSTM (HORL)[44] is a Spatio-Temporal Scene Graph Generation (ST-SGG) method, designed by Kevin Rosales during its final master thesis in January 2022. An old masters' classmate presented his video-based ST-SGG method which uses the

context extracted from the scene graph generated in K-frames around the frame analysed in order to refine these scene graphs using BiLSTM structure. Rosales' HORL methodology uses the SGG method results.

The first step is to divide each video in frames (images). After this, we have to run on each frame the object detector, Rosales uses Faster R-CNN[7], and later a SGG method is used in order to obtain the triplets of the scene graph for each frame. Then, HORL is used to improve each frame F_i generated triplets using the context obtained using an attentional BiLSTM on the adjacent frames $F_{i\pm k}$, where k is the number of frames towards and backwards used in BiLSTM. A complete overview is depicted in Figure 3.6.

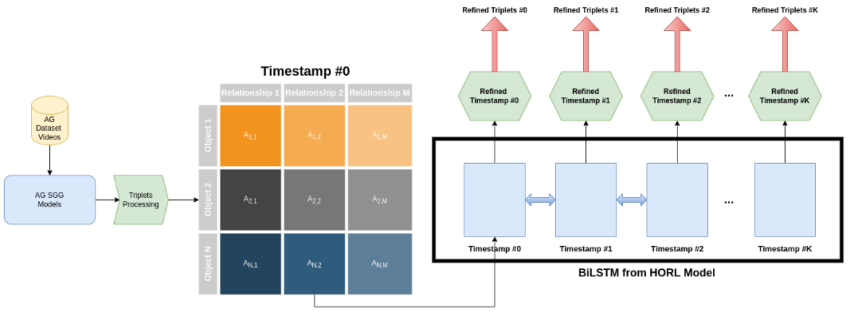


Figure 3.6: HORL method Overview. [44]

It is important to note that Rosales' work is focused on the Action Genome dataset[8], a video-based dataset that focuses on Human-Object Interaction (HOI) relations only. In HOI relations the subject is always a *person*, so it used its premise to only work with the *relation* and *object* of each triplet. With them, he constructed a 2D matrix where each triplet generated in a frame is mapped into the matrix by its relation and object combination. Then Rosales applies the attentional BiLSTM layer to generate a set of refined triplets based on the context extracted from the other adjacent frames.

Kevin Rosales built two different HORL architectures:

- CNN-based HORL: where the main idea behind the architecture is the construction of an encoder-decoder similar architecture.
- Linear HORL: where the key idea behind it is a implementation similar to a general regression method. A PCA transformation is used between the frame associated SGG matrix and attentional BiLSTM in order to reduce the dimensionality of the matrix. After attentional BiLSTM, the features are reshaped to the original matrix dimensions.

[7]Faster R-CNN is described in Section 3.1 of this document.
[8]The Action Genome dataset is described in Section 4.1.3 of this document.

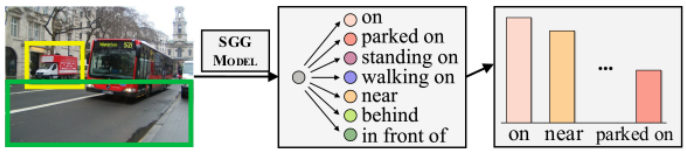
3.3 Common-sense Knowledge methods

This section reviews the most relevant Common-sense knowledge approaches[9] applied in SGG: CogTree (section 3.3.1), GB-NET (Section 3.3.2) and Visual Distant Supervision (Section 3.3.3).

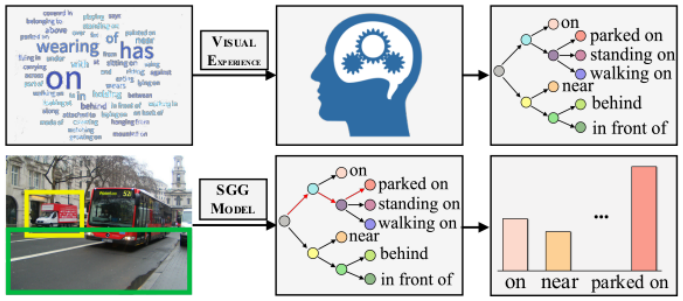
3.3.1 CogTree

CogTree [64], or Cognition Tree loss for SGG, is a common-sense knowledge approach based on the creation of a hierarchical tree, which aims to decrease the model bias caused by a poor relation-space differentiation.

In the current state-of-the-art SGG methods, a single loss is used to distinguish a long-tailed set of actions (as can be seen in Figure 3.7(a)), and usually a complete exploration through the relation space may require a huge ammount of training data that are not available. CogTree proposes the construction of a hierarchical tree that allows the usage of a 3-level loss (shown in Figure 3.7(b)), which allows better action space exploration with fewer data.



(a) Flat thinking.



(b) Cognition-based hierarchical thinking.

Figure 3.7: (a) SGG model with conventional loss. (b) SGG model with CogTree proposed cognition tree loss. [64]

[9] In Section 4.2 we will present our own proposal, the Generalized Action Graphs (GAG), which will be compared with these methods in Chapter 5.

In order to build the hierarchical graph, CogTree does not require external data from other sources. A 3-step methodology is proposed:

1. using a trained SGG method, they predict the relations over all training images. For each relation R_i, the method extracts the predicted relation R_j when the ground-truth relation is R_i, obtaining the frequently predicted classes (step 1 in Figure 3.8);

2. using this statistics created based on the biased predictions, they associate each relation with a node, and create an edge between the relation R_i and its most predicted relation (step 2 in Figure 3.8) obtained previously (as the most predicted relation is with itself, they actually use the second most predicted relation);

3. finally, in order to aggregate all the subtrees generated, the authors propose a structured aggregation in four layers (step 3 in Figure 3.8):

   - Root Layer: is a virtual general node.
   - Concept Layer: under each node of this level, there is one previously generated sub-graphs, so it is used to distinguish the concepts.
   - Coarse-fine Layer: this layer separate the root of the generated sub-graphs from its leafs.
   - Fine-grained Layer: where the sub-graphs leafs are placed.

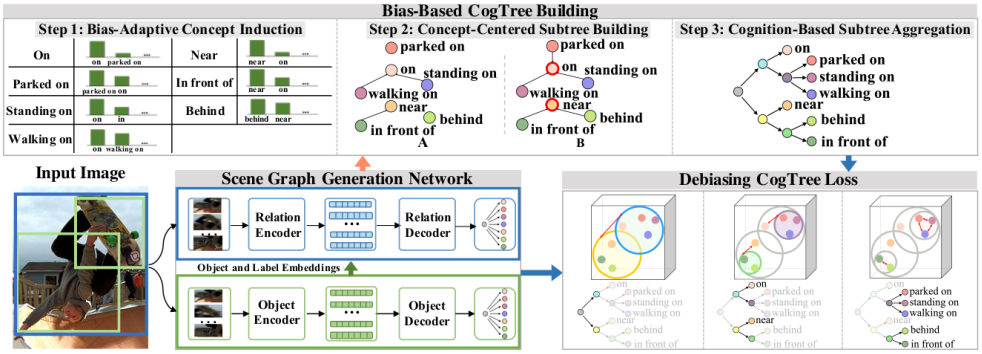


Figure 3.8: The overview of CogTree loss applied to SGG models. [64]

In order to define the new loss, given a sample with the ground-truth path L_{path}, as illustrated in Figure 3.9, the authors compute the class-balanced softmax cross-entropy loss on each layer of CogTree:

$$\zeta_{TCB} = \frac{1}{|L_{path}|} \sum_{i \epsilon L_{path}} = -w_i log(\frac{exp(z_i)}{\Sigma_{z_j \epsilon B(i)} exp(z_j)}) \tag{3.5}$$

where B(i) means the brothers of node i. This TCB loss forces the network to surpass inter-concept relation noises, learning concept-specific embeddings. Finally, TCB loss is aggregated to model loss.

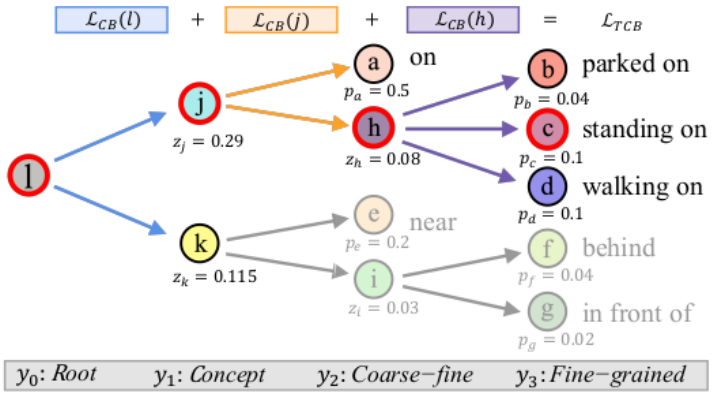


Figure 3.9: CogTree-based class balanced (TCB) loss. [64]

3.3.2 GB-NET

Graph Bridging Network (GB-NET) [66] is a method that, given an image, it initializes the object/subject (entity) and predicate as nodes of a graph, and then classifies each node by connecting it to its class node in a common-sense graph, establishing a special edge call *bridge*. This premise generates a connection between visual knowledge and generic and common-sense knowledge.

To incorporate this combination of visual and common-sense information in the SGG process, the authors propose a GNN, that using the message passing technique[10] allows the exchange of information between the scene and common-sense graph, as well as within each of them. The external knowledge databases used to obtain the common-sense knowledge are ConceptNet[11] and WordNet[12].

Firstly, it is important to define the main differences between each graph type mentioned in this section:

- Knowledge graphs: defined as a set of entity and predicate nodes, with a semantic label and a set of weighted edges of a predefined set of different types (representing different conceptual relations).

[10]The Message Passing technique is described in Section 2.3.6 of this document.
[11]ConceptNet is described in Section 2.4.2 of this document.
[12]WordNet is described in Section 2.4.1 of this document.

- Common-sense graphs: it is a subtype of Knowledge graphs proposed in by GB-NET authors, in which each node represents both, entities and predicates but edges only encode a subset of specific relational fact, concretely the following ones:
  - *SimilarTo* (extracted from WordNet)
  - *PartOf* (extracted from ConceptNet)
  - *RelatedTo* (extracted from ConceptNet)
  - *IsAMannerOf* (extracted from ConceptNet)
  - *UsedFor* (extracted from ConceptNet)

The approach proposed by GB-NET starts with an intial step where an object detector, Faster R-CNN, initializes the scene graph with entities, predicates and their connections in the image. Then the *bridges* from each of the generated entity and predicate node and its representation in the common-sense graphs are set. Then the BG-NET propagates messages with each node features to its behaviours, including intra-graph edges and *bridges* between scene and common-sense graph (as can be seen in Figure 3.10).

This process is repeated for a predefined number of steps. The final state of the bridge determines which class each node belongs to, resulting in the output scene graph.

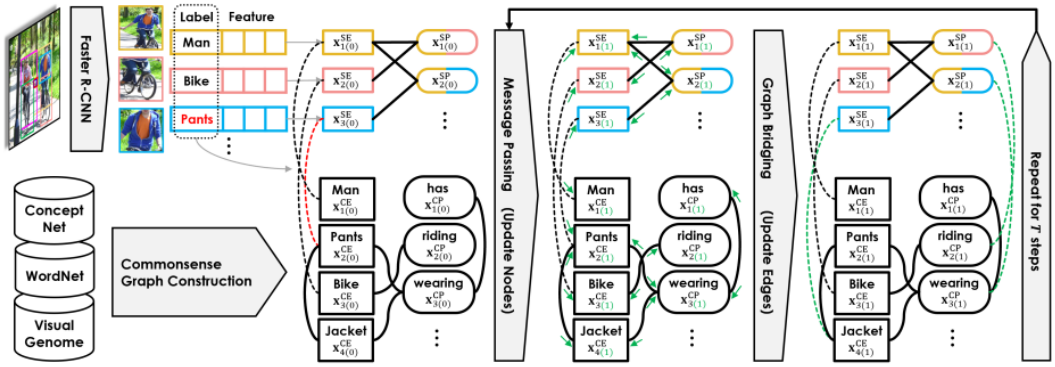


Figure 3.10: Bridging Knowledge Graphs (GB-NET) method overview. [66]

3.3.3 Visual Distant Supervision

Visual Distant Supervision (VDS) [62] aims to allow Scene graph Generation (SGG) models to be trained without human-labelled data, using the common-sense knowledge database in order to encode the possible relation candidates between objects.

The approach proposed also alleviates the long-tail problem, which is the lack of enough training samples for every relation category due to the long list of possible relations.

The main insight of VDS is the usage of visual relational triplets to directly train SGG model when there is not human-labelled data, or in the case of having some training human-labelled relations, use this visual distant relations to build a semi-supervised approach. To build the triplets the authors did not use a public knowledge database (e.g., WordNet or ConceptualNet), they propose the usage of Conceptual Captions [48], a dataset with 3.3 Million captions, in addition of a rule-based textual parser.

First, the authors propose the use of an object detector to locate the different objects and these labels in the scene. In order to associate the object label to an element of the common-sense knowledge base, the authors propose a distance calculation between object label and visual supervised label.

In the case of distantly supervised framework (see the green path in Figure 3.11), after obtaining the objects' association, the visual distance triplets are used in order to train the predicate loss function directly.

On the other hand, in semi-supervised framework (see the red path in Figure 3.11), the authors propose a 2 steps methodology: first, a model is pretrained using distantly supervised framework, as in the first case, then the human-labelled data is used to fine-tune the model obtaining better results than other fully supervised methods. (the comparison can be seen in [62])

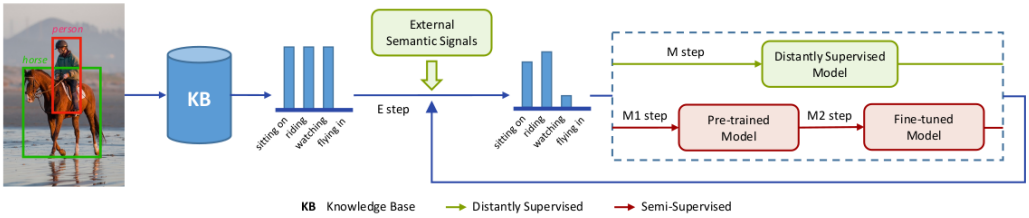


Figure 3.11: Visual Distant Supervision (VDS) method overview. [66]

3.4 Chapter Summary

In Section 3.2 we have reviewed some SGG methods that are most broadly used in the state-of-the-art. These are the models used in the experiments in Chapter 5. Additionally, in Section 3.1 the Faster R-CNN model is explained, which is the model used as Object

Detector in our Scene Graph Generation models experiments.

Section 3.3 describes the three currently used Common-sense techniques applied in SGG. These techniques are important because they establish a reference to compare our proposed Common-sense technique: the Generalized Action Graphs (GAG) which is explained in Section 4.2.

In the following chapter, the methodology followed to build the data preprocessing, the new proposed technique and the training of the different SGG methods are described.

Chapter 4

Methodology

This chapter the procedure used in each of the points that this work aims to explore. From the data has been used for its development and the new model that we have created to improve generalisation to the trained algorithms and the exploration of results obtained by our model and other models to be able to compare them in Chapter 6.

Specifically, section 4.1 describes the datasets used for the development of the project. Then Section 4.2 introduces the Generalized Action Graphs (GAG), which is our proposed Common-sense technique to increse the generalization of SGG's. Section 4.3 develops the data preprocessing required later in sections 4.4 and 4.5 for its correct training. Section 4.6 is devoted to explain the sources used to extract the resutls of the statw-of-the-art methods. Finally, section 4.7 explains the implementation of a recently published metric, which seeks to measure the generalizability of an SGG model.

4.1 Used Datasets

In this section, all the datasets used will be briefly presented. Since this work is made up of different tasks, it is important to define for which task each dataset has been used:

- Visual Genome (section 4.1.1): is used to train static SGG methods, to evaluate them in order to compare the results. Additionally, its HOI training ground-truth triplets are used to fill graphs in our proposed GAG method[1].

- Action Genome (section 4.1.3): is used to train and evaluate the HORL method, which is a ST-SGG video-based method. As in Visual Genome, its ground-truth triplets are used also to generate graphs in GAG method.

[1]The Generalized Action Graphs method is described in section 4.2.

- Humans Interacting with Common Objects for Detection (section 4.1.2): is only used to generate triplets in order to fill action graphs in GAG method.
- Home Action Genome (section 4.1.4): is used uniquely to extract its triplets from the training ground truth annotations to generate action graphs in GAG method.

4.1.1 Visual Genome

The Visual Genome dataset [28] is an image-based dataset presented in 2015. It is considered one of the most important datasets in the Computer Vision field in general and in the Scene Graph Generation (SGG) field in particular. This dataset has provided an important base of human-labelled images with the relations between the objects of each image.

Visual Genome contains 108077 images where each image has an average of 35 objects, 26 attributes, and 21 pairwise relationships between objects. It presents the following characteristics set:

- 5.4M region descriptions.
- 1.7M visual question answers.
- 3.8M object instances.
- 2.8M attributes.
- 2.3M relationships.
- Everything Mapped to WordNet Synsets.

This last characteristic has relevance in our work because it provides the associated WordNet synset of each relation and object label which is an important issue in GAG method, as we will explain in section 4.2.

An example of Visual genome scene graph can be seen in Figure 4.1, where we are able to distinguish the three types of triplets represented in this dataset: (*subject*, *relation*, *object*), (*object*, *relation*, *attribute*) and (*subject*, *relation*, *attribute*)

4.1.2 Humans Interacting with Common Objects for Detection

Humans Interacting with Common Objects for Detection (HICO-DET) is a dataset for detecting Human-Object Interaction (HOI) in images [9] released in 2018. It contains 47,776 images (38,118 in the train set and 9,658 in the test set), it is composed by 80 object categories and 117 verb classes. HICO-DET provides more than 150k annotated human-object pairs.

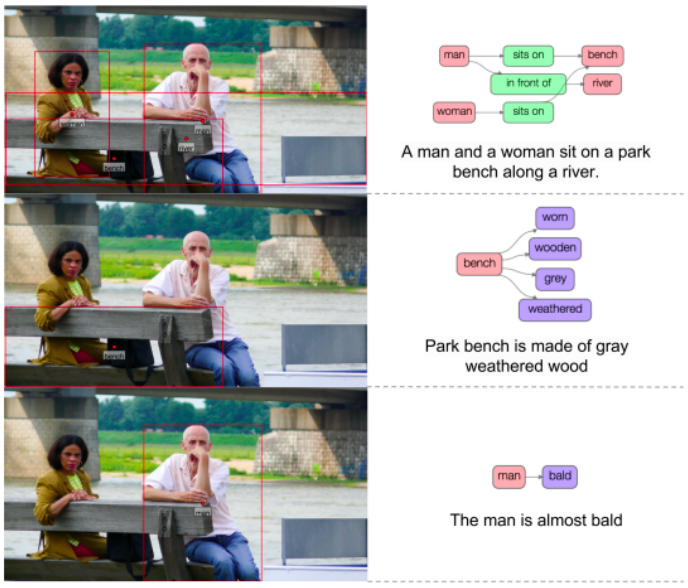


Figure 4.1: Simple examples of scene graph generated in Visual Genome. [28]

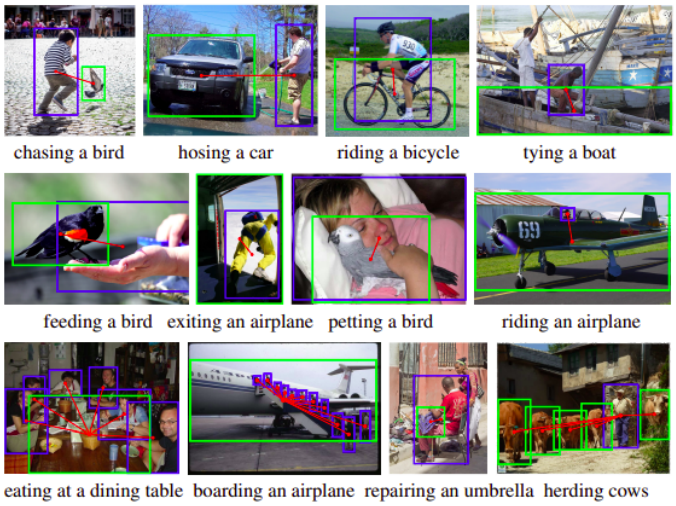


Figure 4.2: Examples of sample annotations in HICO-DET. [9]

As can be observed in Figure 4.2, only a relationship (action) is assigned to each image, but there can be more than one subject or more than one object.

4.1.3 Action Genome

The Action Genome dataset [24] is a video-based dataset, which is focused on the action decomposition into spatio-temporal scene graphs. Action Genome provides the scaffold to study the dynamics of actions as changing relationships between people and objects, so it is centred in Human-Object Interaction (HOI). It is build on the Charades dataset [29], which is a video-based dataset of daily indoors activities.

Action Genome contains 10000 videos and 234000 frames extrated from those videos, where the 25% approximately are devoted to the test set. It provides 400000 objects of 36 classes (including the subject class, in this case always is *person*) annotated with their bounding boxes. Moreover, it contains the annotation of 1.7M visual relationships with their 26 action categories. The distribution of the database categories is shown in Figure 4.3.

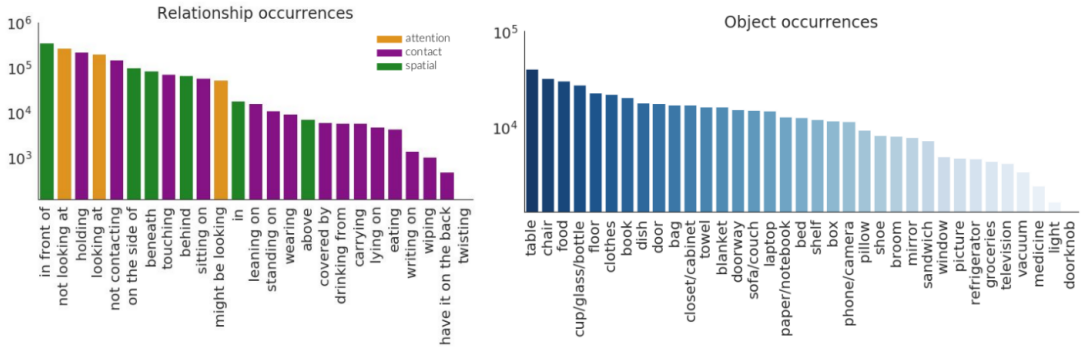


Figure 4.3: Object/Relation occurrences distribution inside Action Genome dataset. [24]

This dataset contains low-quality videos, with poor illumination and videos with blur or moving-camera information. Their purpose is to add difficulties to the scene graph generation task, including very common problems of video-datasets to assist the creation of techniques with resistance to difficult video inputs.

4.1.4 Home Action Genome

Home Action Genome (HOMAGE) [39] is a multi-view action video dataset with multiple modalities and view-points, supplemented with hierarchical activity and atomic action labels and scene graphs, released in 2021 by *Nishant Rai1 et al.*, in collaboration with

Panasonic corporation.

This dataset presents a set of different views for each different scene, plus an *Ego-view* which presents a first person view of the same scene. Additionally, all the different view videos of a same scene are synchronized between them, having a total number of at most six different and synchronized views per scene, (see examples in Figure 4.4).

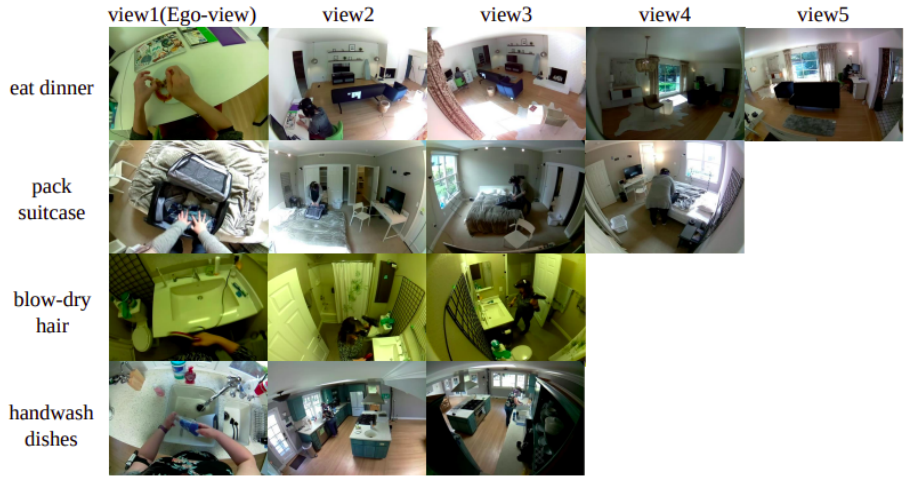


Figure 4.4: Multiple Views of HOMAGE dataset. Each sequence has one ego-view video as well as at least one or more synchronized third person views. [39]

The authors provide labelled spatio-temporal scene graphs, with its respective bounding boxes, for one of the views in the scene. In the same way, they provide a video-level activity labels with its temporally localized atomic activity labels for each scene.

In general, HOMAGE presents 1,752 synchronized sequences (scenes), with 5,700 videos in total. The sequences are split in three: 1,388 train sequences and two test splits containing 198 and 166 sequences each. For the scene graph task, there are 86 object classes (excluding "person"), and 29 relationship classes in the dataset. Overall, there are annotations of 497,534 bounding boxes and 583,481 relationships. For atomic actions composition task, there are 20,039 training, 2,062, and 2,468 atomic action sequences in the three splits (train, validation and test).

4.2 New method: Generalized Action Graphs

Generalized Action Graphs (GAG) aim to increase the SGG method's generalization, reducing the knowledge bias generated by prior knowledge usage of hierarchical graphs based

on the object inherited hyperonyms in WordNet[57][2]

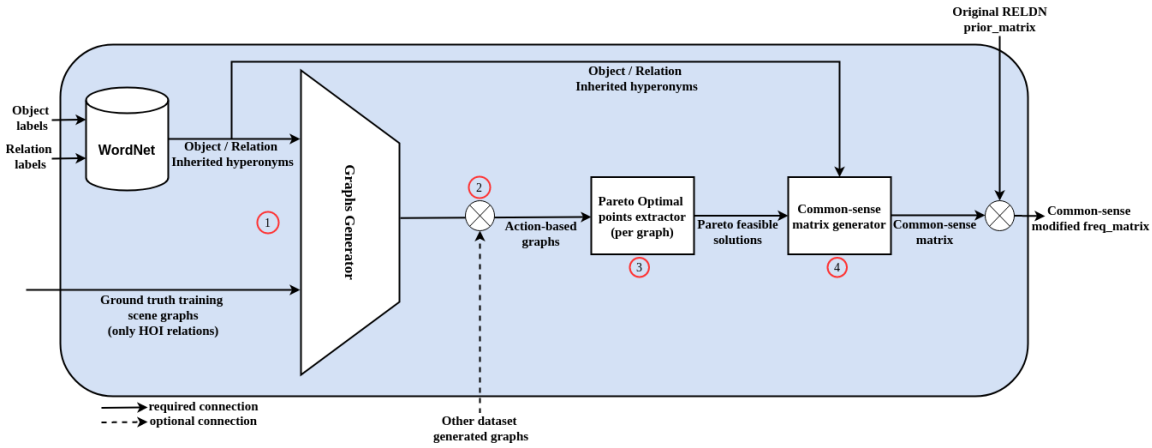


Figure 4.5: GAG method overview. Each main step is highlighted: (1) Action graphs generation, (2) Graphs fusion between datasets, (3) Pareto Optimal Points Generator and (4) Common-sense matrix generation and Prior_freq Matrix Aggregation

With this generated graph for each action, it is possible to fuse the knowledge obtained from different databases. After the graphs construction is done, the Pareto optimal points[3] are searched using the two attributes of each node: (1) distance from "entity" node, which is the root node for objects in WordNet. (2) the number of inherited apparitions in the ground-truth data (i.e., for each time that a tuple $\langle predicate, object \rangle$ appear, the apparition is counted in the node n_{object} and in all the hyperonyms of this label, in the graph $G_{predicate}$).

After obtaining the Pareto optimal points, the sub-graph under each Pareto optimal point is gathered. For each sub-graph the graph's leafs are extracted, each leaf node is compared to the dataset labels and the intersections between the two sets are finally selected.

Finally, the selected labels are highlighted to a common-sense matrix, which is a 2D-matrix that represents the most suitable objects for a specific action. In the case evaluated, this common-sense matrix is integrated to RelDN (3.2.4) prior knowledge matrix. Finally, with the aggregated matrix, RelDN method is applied.

This method is centred in Human-Object Interaction HOI, in this case only the relations that have human-centred label a subject are considered, $\langle$"*person-like*", "*man*", "*women*"$\rangle$, etc. A technique overview can be seen in Figure 4.5.

[2]WordNet is described in Section 2.4.1 of this document.
[3]A more detailed description of the Pareto Optimal Points Generator is provided in Section 4.2.3).

In the following sub sections, the steps mentioned before are explained more in detail. Starting with the action graph generation (Section 4.2.1), followed by the action graphs fusion between different datasets (Section 4.2.2), then the Pareto Optimal points search (Section 4.2.3) and ending with the aggregation between both, common-sense matrix and prior knowledge matrix (Section 4.2.4).

4.2.1 Graph Generation

In order to build the action graphs it is important to identify which inputs are required for the task. Firstly, the method requires the complete lists of objects and relations in the dataset (see an example in Figure 4.6) and, the SGG training ground-truth scene graphs.

```
### Action labels ###
lookingat
notlookingat
unsure
above
beneath
infrontof
behind
onthesideof
in
carrying
coveredby
drinkingfrom
eating
haveitontheback
holding
leaningon
lyingon
notcontacting
otherrelationship
sittingon
standingon
touching
twisting
wearing
wiping
writingon
```

```
### Object labels ###
person
bag
bed
blanket
book
box
broom
chair
closetcabinet
clothes
cupglassbottle
dish
door
doorknob
doorway
floor
food
groceries
laptop
light
medicine
mirror
papernotebook
phonecamera
picture
pillow
refrigerator
sandwich
shelf
shoe
sofacouch
table
television
towel
vacuum
window
```

Figure 4.6: Object and relationship labels for Action Genome dataset.

The process to build the action graphs list can be divided into four steps, the first three of them are input processing and the last one is the graph generation itself:

1. Generate the associated WordNet synset: for each object and relation label a WordNet synset has to be associated, if the dataset does not bring these associations, they have to be done manually.

2. Generate WordNet associated hyperonyms paths: for each label-associated synset, a recursive search is done, getting all the possible paths from the label-associated synset to the further hyperonym (the most general concept).

3. Generate the train set triplets: using the training set ground-truth scene graphs, standardize them in the form of triplets $[subject, relation, object]$. This step has to be adapted for each dataset used as the ground truth scene-graph structure depends on the dataset.

4. Generate Action graphs: for each action in the dataset, and using all the previously generated data, the graph is built as can be seen in algorithm 1.

Algorithm 1: Action graph generation.

```
Data: sgTriplets, labelSynsetObjectMap, inheritedHyperonyms, relationLabels
Result: graphList
for action in relationLabel do
    graphList_action ← emptyDirectedGraph();
    for triplet in sgTriplets do
        if triplet.subject = person & triplet.relation = action then
            objHierarchy ← labelSynsetObjectMap_triplet.object;
            for hierarchy in objHierarchy do
                for object in hierarchy do
                    if labelSynsetObjectMap_object exists in graphList_action then
                        graphList[action, object].appearances+ = 1;
                    else
                        objNode ← newNode(labelSynsetObjectMap_object);
                        objNode.appearances ← newAttribute(appearances, 1);
                        objNode.rootD ←
                          newAttribute(rootD, lenght(hierachy[object :]));
                        graphList_action ← appendNewNode(objNode);
```

An important aspect of the proposed GAG method is that only takes into account the triplets that have a person as the subject of the triplet, in other words Human-Object Interaction HOI.

We have chosen to perform the graph generation for each action (obtaining the graph generated by the inherited hyperonym of each object affected by the specific relation). We explored the option of building graphs for each object (using the inherited hyperonyms of

each action that affects the object) but, this option was discarded due to the poorly informative graph generated (see an example in Figure 4.7). The graphs created are not always fully-connected, and this issue can be explained due to the poor verb-to-verb connections specified in WordNet.

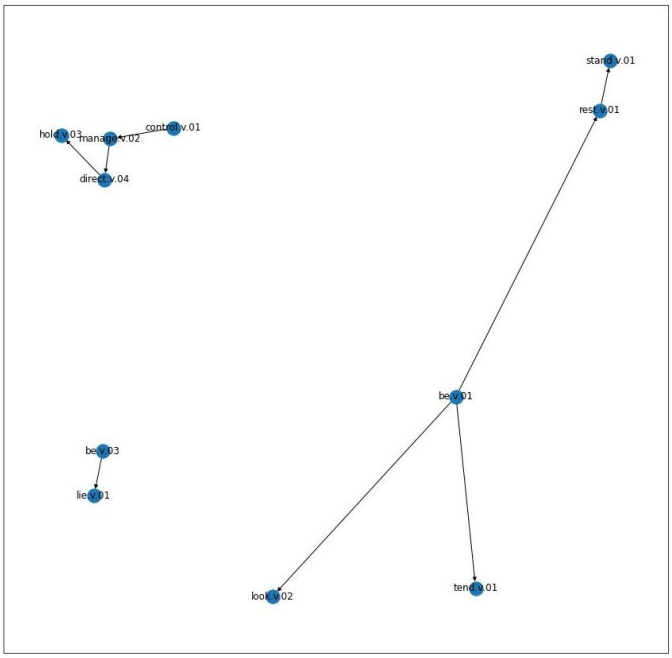
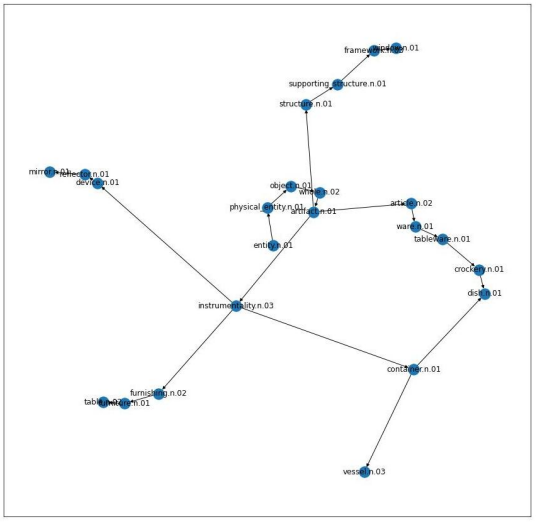


Figure 4.7: Generalized Action graph generated using a chair (as object) as a base, in GAG method

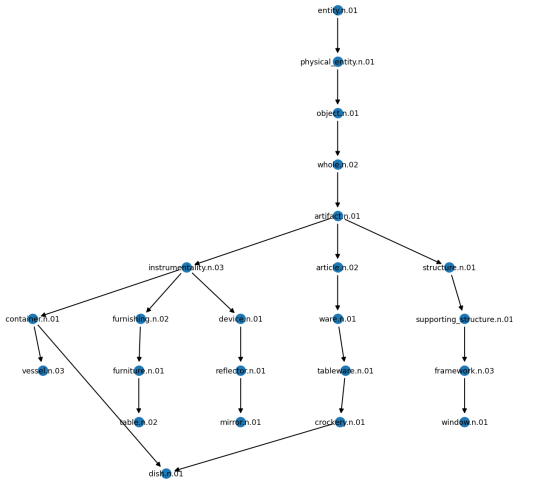
In the case of action-based graph, the graph generated are fully connected always. This property is ensured because all the object synsets in WordNet have a common root synset *entity.n.01*. Therefore, due to this property, we can see the generated graphs as hierarchies with a common root node, as shown in Figure 4.8.

Another important GAG insight is the usage of ground-truth triplets in the graph construction, allowing to count occurrences, such as in prior frequency matrix (the method behind its construction is explained in 4.3.1), and additionally extending these occurrences to all the inherited hyperonyms. But, as we will see in the following GAG steps, these occurrences would not be directly used to extract the usage frequencies.

In addition to the number of occurrences per node (also called appearances), we are able to set the distance from the root for each node, which will be an important factor in



(a) Generalized *wiping* action graph example



(b) *wiping* action graph represented as a hierarchy.

Figure 4.8: Action graph example, with its representation as hierarchy.

order to stabilize the generalization, due to the crescent nature of the occurrences variable over the hierarchy.[4]

4.2.2 Action graph fusion

After obtaining the list of graphs for the actions of a dataset, the next step is optional: the aggregation of other datasets' action graphs which allow us to fuse and extend several dataset information.

The aim of this step is that, by aggregating knowledge from different graphs, we can obtain a bigger scope of different objects in which we can apply the actions which may give us the opportunity to increase the generalization capacity[5] of the GAG method.

Algorithm 2 shows our proposal to implement graph fusion. From all the list of different datasets from which we want to aggregate the information, we use one of them as a base and then one dataset by one we aggregate each graph. The important insight here is that for each graph that have to be aggregated we check if it already exists in the graph list, to evaluate if a node exists we check the synset of the word in order to disambiguate it.

[4]We will further describe this in section 4.2.3.

[5]In section 5.3 we show how generalization capacity is evaluated.

In the case that the graph to be aggregated does not exist, we append this new graph to the list as new graph. On the other hand, if it already exists in the aggregated graph list, the new graph is fused and the *appearance* attribute of each node repeated in both fusing graphs are summed.

Algorithm 2: Different datasets' graph list aggregation.

```
Data: datasetNames
Result: aggregatedGraphList
aggregatedGraphList ← d1Graphs ← loadDatasetGraphs(datasetNames_1);
d1ActionSynsets ← loadSynsets(datasetNames_1);
for xName in datasetNames_{2-N} do
    dXGraphs ← loadDatasetGraphs(xName);
    dXActionSynsets ← loadSynsets(xName);
    for graph in dXGraphs do
        if dXActionSynsets_graph in d1ActionSynsets then
            fuse(d1Graphs_{dXActionSynsets_graph}, dXGraphs_{dXActionSynsets_graph};
        else
            append(d1Graphs, dXGraphs_{dXActionSynsets_graph});
            d1ActionSynsets_graph ← dXActionSynsets_graph;
```

4.2.3 Pareto Optimal Points Generator

With the final list of action graphs (mixing different datasets or not), now it is time to extract the nodes that allow us to generalize, but that, at the same time, they are still sufficiently informative. If we have in mind the hierarchical structure of each graph, in order to obtain these informative nodes, a first approach could be to select the nodes with more appearances, but as this attribute always increases when we go up through the hierarchy (see Figure 4.9), choosing the nodes with higher appereances will tend to select nodes so high in the hierarchy (e.g., person, action) that may be too general, causing an excessive lose of information.

The solution to the problem explained above is a compromise between the number of appearances and the distance from the most general node. In other words, we have to search the nodes which maximizes both attributes of each node: (1) number of appearances and, (2) number of nodes from the root.

In order to obtain this set of nodes which maximize both node attributes, we decided:

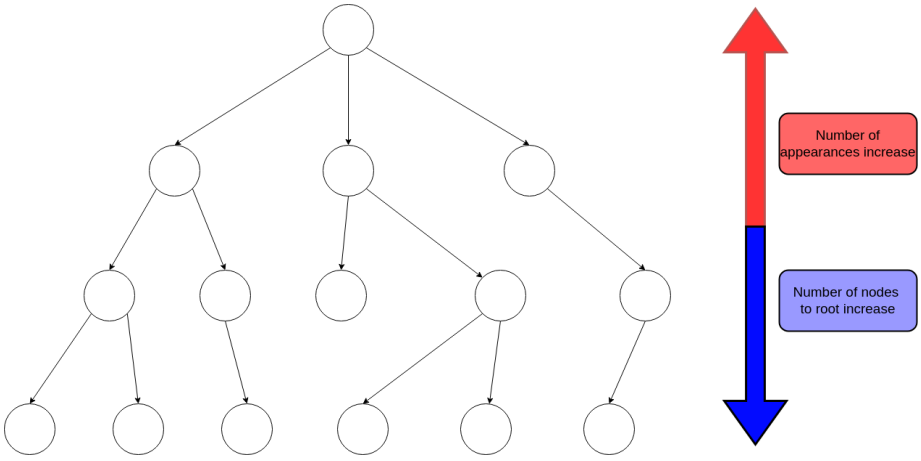


Figure 4.9: Action graphs, node attributes tendencies.

- to represent each graph node in 2-dimensional space, where each dimension is each one of the node attributes normalized,
- to apply on them Pareto optimization which give me the dominant points of the graph

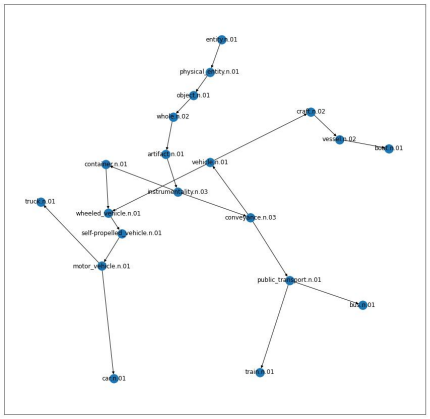
For each graph, the Pareto optimal points (in this case each node is represented with a synset) are extracted and will be processed, as can be seen in Algorithm 3:

Algorithm 3: Pareto optimal points extraction from action graph.

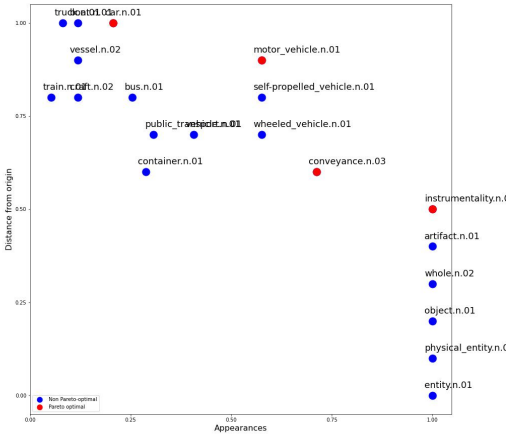
Data: graphList
Result: paretoVectorAction
$paretoVectorAction \leftarrow Dictionary(empty)$;
$actNames = names(graphList)$;

for *action in act_names* **do**
  $nodeList \leftarrow GetNodes(graphList_{action})$;
  $normNodeList \leftarrow NormalizeAttributes(nodeList)$;
  $paretoOptimal \leftarrow ExtractParetoNonDominated(normNodeList)$;
  $paretoVectorAction_{action} \leftarrow paretoOptimal$;

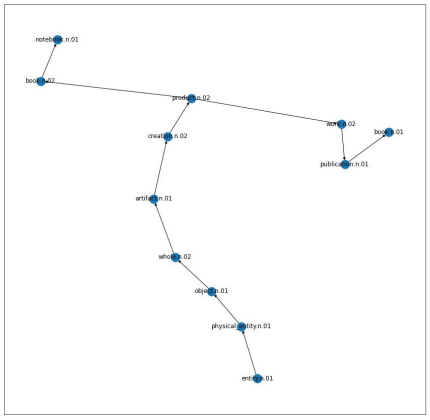
Figure 4.10 shows two different examples of the Pareto optimal points on two different images. We can see that in the case of the **drive** action (see Figure 4.10a(a) and (B)) some of the selected nodes are very informative nodes (e.g., *car* and *motor bicycle*). In the case of the **writtingOn** action (see Figure 4.10c(c) and (d)), a smaller graph is generated, with selected nodes that are far more specific (e.g., *book* or *notebook*).



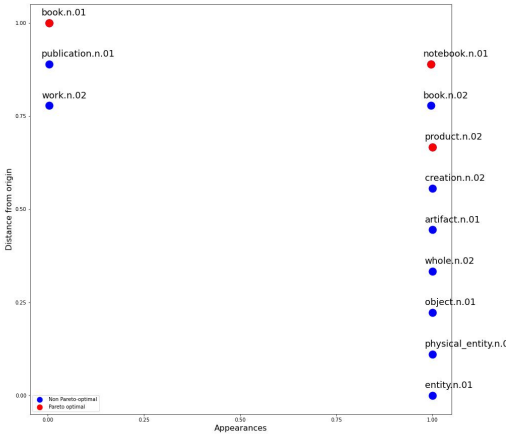
(a) Drive action generated graph



(b) Drive action Pareto optimal points



(c) WritingOn action generated graph



(d) WritingOn action Pareto optimal points

Figure 4.10: Examples of Pareto optimal points extraction.

4.2.4 Common-sense matrix generation and Prior_freq Matrix Aggregation

After obtaining the Pareto optimal nodes (which are synset representations), now it is time to generate a matrix, which we call common-sense matrix. This step is specific for each dataset, because the matrix generated will have the dataset label dimension ($dim = object\_labels \times relationship\_labels$), initializing it full of zeros in the beginning of the process.

For each action, the highlighted synsets as Pareto optimal solutions are used, jointly with the *Inherited hyperonyms* extracted in the graph construction (as explained in section 4.2.1), to map all the graph leaf nodes under a synset denoted as Pareto optimal. For each Pareto optimal synset, the sub-graph leafs under it are extracted.

The leaf synsets that are in the dataset are mapped in the matrix summing 1 for each time that it appears as a leaf (see Algorithm 4). Finally, for each relation (mapped as columns in the matrix), the values obtained are normalized, using $L2$ normalization.

Algorithm 4: Common-sense matrix construction.

```
Data: paretoNodesAction, inheritedHyperonyms, objectLab, relLab
Result: csMatrix
csMatrix ← EmptyMatrix(dim : lenght(objectLab) * lenght(relLab));
for action in relLab do
    paretoSynsets ← paretoNodesAction_action;
    for synset in paretoSynset do
        synsetLeafs = getLeafs(inheritedHyperonyms_action, synset);
        for leaf in synsetLeafs do
            if leaf in objectLab then
                csMatrix_action,leaf = +1;
    csMatrix_action ← L2Norm(csMatrix_action, axis = relation);
```

In order to use the generated common-sense matrix into RELDN (a SGG method, explained in section 3.2.4), we decided to aggregate both the prior_matrix generate by RELDN method and the Common-sense matrix in the preprocessing data step. This new matrix has the same dimension of the original prior_matrix ($dim \leftarrow (objects \times objects \times relations)$).

The aggregation is done as a sum of equally dimensioned matrix in the dimensions where a person is the subject of the relation, due to the fact that the Common-sense matrix only affects (Human-Object Interactions). Finally, the obtained 2D matrix is normalized

using $L1Norm$ in the object axis. The precise process is described in Algorithm 5, where *personal* are the set of object labels that represent humans in a dataset (e.g. person, man, women) and λ is a parameter with range [0,1], useful to set the importance of the common-sense matrix in the aggregation.

Algorithm 5: Common-sense and Frequency matrix aggregation.

Data: commonMatrix, freqMatrix, personal, λ
Result: mixMatrix
$mixMatrix \leftarrow freqMatrix$;
for *p in personal* **do**
  $resMatrix \leftarrow freqMatrix_p + \lambda * commonMatrix$;
  $normMatrix \leftarrow L1Norm(resMatrix, axis = object)$;
  $mixMatrix_p \leftarrow normMatrix$;

4.3 Data Preprocessing

Data preprocessing is one of the most important steps when training a machine learning algorithm, if not the most. In our experiments there is not a common preprocessing step for all the SGG models due to the fact that different repositories were used to run the different SGG models and that some runs included common-sense reasoning techniques and some did not.

But we can highlight that in all of the repositories used, both the images and annotations did not undergo major changes, they were simply regrouped in intermediate files that standardized the data from different datasets for each of the repositories.

Next sections focus on the process to create the base *freq_prior.npy* file (section 4.3.1), and its variant *cs_freq_prior.npy* (generated through the proposed GAG method, presented in the previous section). These files are used for the RelDN method[6] training in order to feed its Semantic module.

4.3.1 Prior_freq file Generation

This file contains the subject-object class co-occurrence frequencies matrix, which are necessary for RelDN method. In order to build it, it uses the training set ground-truth triplets. For each image, the occurrences of predicate class *pred* given subject and object classes are counted. s and o are the labels in the ground-truth annotations for subject

[6]The Relationship Detection Network is described in in section 3.2.4 of this document.

and object. This gives us an empirical distribution $p(pred|s, o)$ in a 3D-matrix. This matrix is stored in a NumPy or Pickle file (depending on method implementation requirements).

This distribution created has an obvious bias on the training set annotation triples, though. Therefore, in order to decrease this bias, we propose to apply the GAG method which generates a new matrix where common-sense abstractions and prior knowledge are aggregated, as we will see in the next section.

4.3.2 Commonsense_Prior_freq file Generation

This file contains the aggregation of the common-sense knowledge obtained after the application of GAG method presented in this work and the co-occurrences matrix, explained just in the previous section.

As explained in Section 4.2, the GAG method extracts some common-sense knowledge, searching for relevant object hyperonyms for each predicate. After finding them, these relevant hyperonyms are decomposed into all their hyponyms inside the object labels. With the obtained relevant object labels for each predicate, a 2D-matrix is created and finally aggregated to person-like subject dimensions in the frequency prior matrix. Figure 4.11 summarizes the process to build the common-sense prior knowledge matrix.

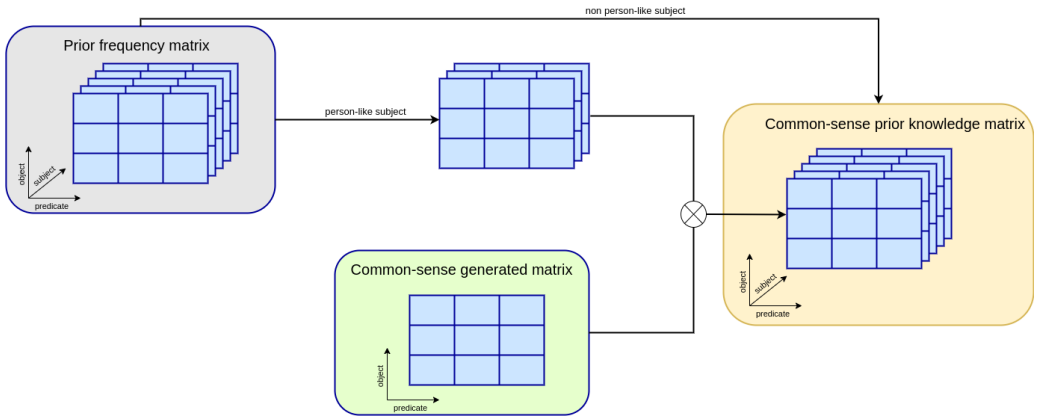


Figure 4.11: Common-sense prior knowledge matrix construction overview

The result of this procedure is a matrix with the same dimension as the frequency prior matrix, in which the person-like subject sub-matrices are an aggregation of the prior-knowledge and the common-sense knowledge. As in the previous case, the matrix is stored in a NumPy, or Pickle file (depending on method implementation requirements).

4.4 Faster R-CNN Training

Once all the data preprocessing is done, it is time to start the training of the model. Our approach is a two-step SGG model, which consists of a first object detection model that will be responsible for locating and classifying the objects in each scene. This first model is trained individually in the same dataset that will later be used to train the model that will determine the relationships between objects.

The model chosen for our project is Faster Region-based Convolutional Neural Network (Faster R-CNN)[63][7] because the object detector step is the most time-consuming step and Faster R-CNN is the mainstream object detector model in the community, with many examples of this model already pretrained (which causes a drop in estimated training time).

In the case of Faster R-CNN object detectors trained into Visual Genome[28] there are examples of pretrained Faster Region-based Convolutional Neural Network (Faster R-CNN) available in many repositories (e.g., [15][8][2]). But our work also aims to make some analysis in the case of HORL, which is a ST-SGG method that uses videos as input from the Action Genome[24]. This last dataset has a lack of pretrained object detectors on it, so to tackle this problem, in our work we train four object detectors (two for each dataset) based on **ResNet-50** and **ResNet-101**, all of them used the Feature Pyramid Network (FPN). All the four trained models use as backbone a *ResNet*[20] model which is pretrained using the ImageNet dataset[13] and the COCO dataset[31]. In the case of the trained Faster R-CNN models in Visual Genome, the one that gets better results will be compared with the one that is already pretrained by the community, and the better one will be chosen as the object detector for the final model.

In this stage one more data preprocessing has to be done for Action Genome dataset annotations data, in order to completely align the data provided by the dataset and the one required by the *Detectron 2*[56], which is the repository where Faster R-CNN is implemented. This last preprocessing only requires the change of some parameters of each object bounding box. This was because of the format of the mentioned bounding box: the ones provided by Action Genome have the **xyhw** format (top left coordinates, width and height) and Faster R-CNN requires the **xyxy** format (top left coordinates and bottom right coordinates).

In order to set all the hyperparameters required by the model training, for each trained model they are store in *YAML* files, which later are parsed by the repository. The parameters set for the models based on **ResNet-50** are the same for both datasets, and the same happens in the case of the models based on **ResNet-101**:

[7](Faster Region-based Convolutional Neural Network (Faster R-CNN) is described in Section 3.1 of this document.

- In the case of ResNet-101 models, the base learning rate and the epochs were set to 10^{-3} and 100000, respectively. The learning rate was decreased utilizing a weight decay of 10^{-4} at epochs 30000 and 60000. Each batch of data consists of two images and a checkpoint of the model is stored every 20000 epochs.

- In the case of ResNet-50 models, the base learning rate and the epochs were set to 10^{-3} and 150000, respectively. The learning rate was decreased utilizing a weight decay of 10^{-4} at epochs 30000 and 60000. Each batch of data consists of two images and a checkpoint of the model is stored every 20000 epochs. Since the validation step is highly resource demandant, it was performed every 20000 epochs in both cases of ResNet-101 and ResNet-50.

The results obtained are shown in Figure 4.12 for Visual Genome trained models and in Figure 4.13 for Action Genome trained models. In both datasets, the best results are obtained by the ones that have **ResNet-101** as backbone. The final AP50 (50 is because the IoU threshold is set to 0.5) for each model can be seen in table 4.1. In the case of Visual Genome the trained model trained with *Detectron 2*[56] results was incompatible with the repository used to train the SGG models, for this reason a third Faster R-CNN model was trained with the same hyperparameters as the previous ones but through another repository called *Maskrcnn_benchmark*, an older version of *Detectron 2*. As we had already done an analysis In this case, only one model was trained with a ResNet-101 as backbone. The resulting training loss progression can be seen in Figure 4.14.

	Models	AP50
VG	w/ ResNet-50 backbone	24.688
	Self-trained w/ ResNet-101 backbone	**26.284**
AG	w/ ResNet-50 backbone	25.094
	w/ ResNet-101 backbone	**28.246**

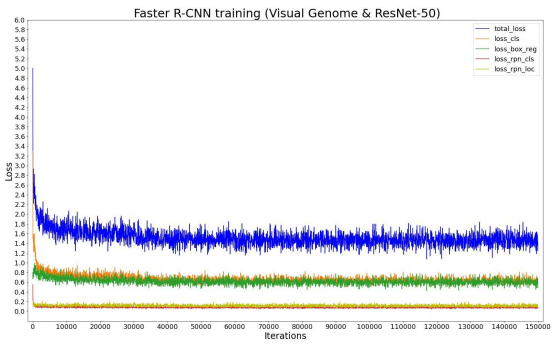
Table 4.1: AP50 obtained by each Faster R-CNN, Action genome (AG) and Visual Genome (VG)

After obtaining each model result metrics we are able to, at first glance, draw a couple of conclusions:

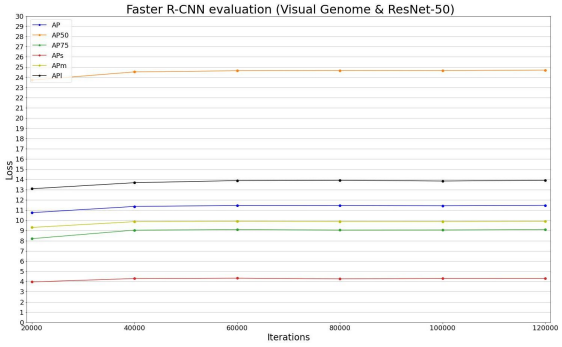
1. models that have ResNet-101 as their backbone outperform models that are based on ResNet-50,

2. there is substantial variability in the mean accuracy obtained for each object.

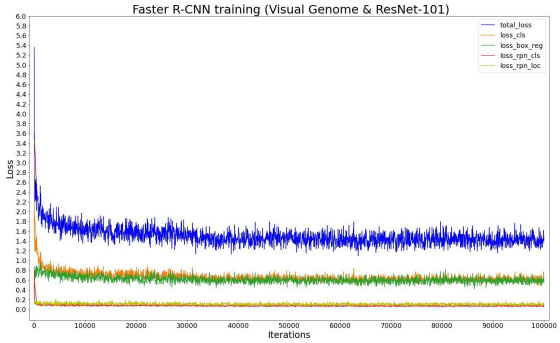
From this point on, we are only going to use the models with ResNet-101 to carry out the subsequent the results' analysis.



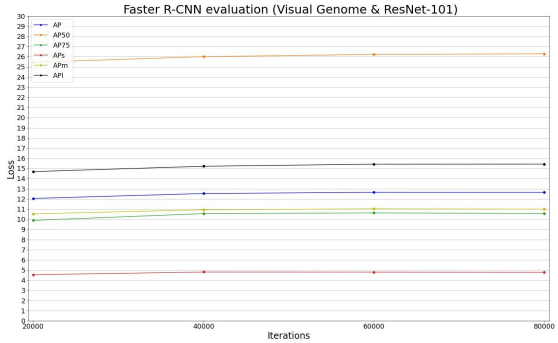
(a) Training losses (ResNet-50)



(b) Validation Average Precision (ResNet-50)

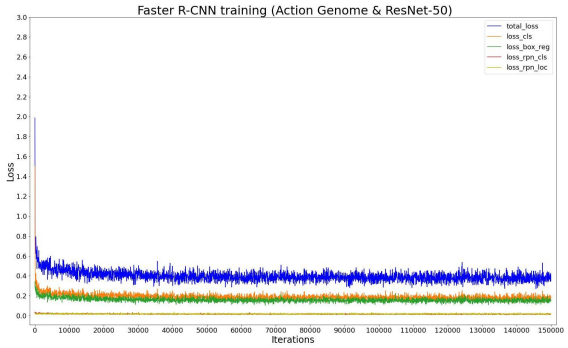


(c) Training losses (ResNet-101)

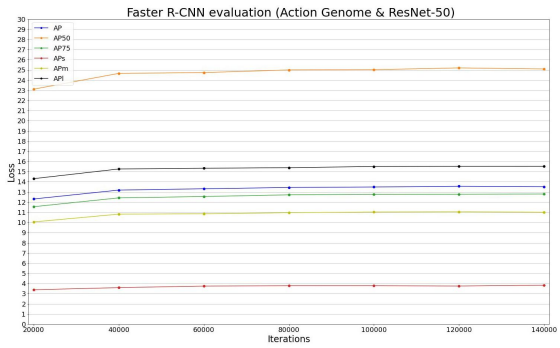


(d) Validation Average Precision (ResNet-101)

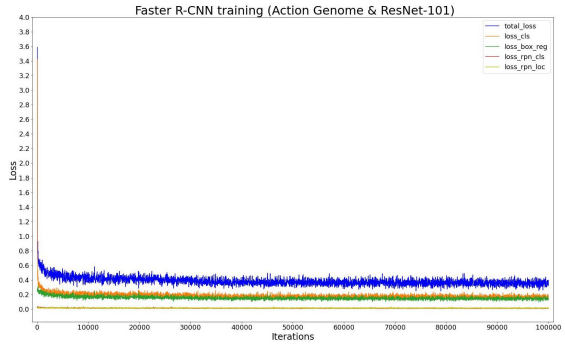
Figure 4.12: Faster R-CNN training and validation results for Visual Genome dataset



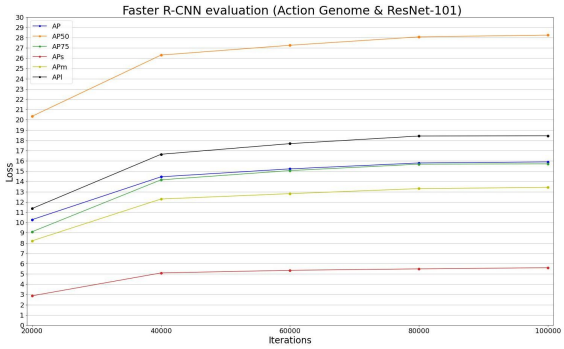
(a) Training losses (ResNet-50)



(b) Validation Average Precision (ResNet-50)



(c) Training losses (ResNet-101)



(d) Validation Average Precision (ResNet-101)

Figure 4.13: Faster R-CNN training and validation results for Action Genome dataset

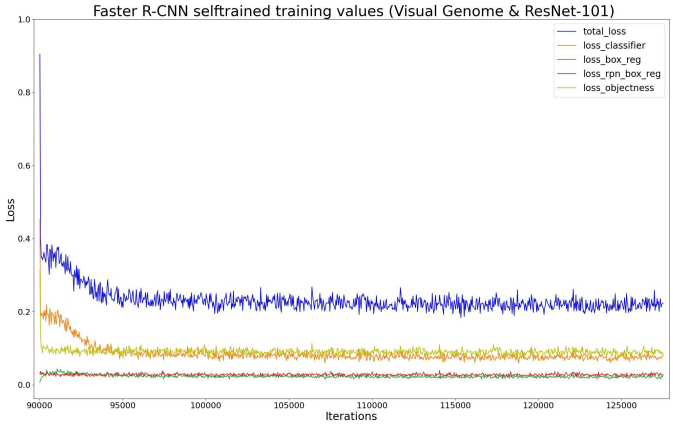


Figure 4.14: Training loss progression for Faster R-CNN model trained using *Maskr-cnn_benchmark repository.*

To be able to observe this last conclusion graphically, we are going to use a simplification of the metric implemented in this work, mean Precision Degradation (mPD) (more information in section 4.7). In this case, we are not going to look for the average between all the Performance Degradation, but we are only going to look for the Performance Degradation between all the average precisions obtained. This Performance degradation can be formulated as in 4.1:

$$PD = \frac{AP_{\langle o_{max} \rangle} - \bar{AP_o}}{AP_{\langle o_{max} \rangle}} \tag{4.1}$$

Where $AP_{\langle o_{max} \rangle}$ is the maximum Average Precision obtained overall object categories and $\bar{AP_o}$ is the mean of all objects Average Precision. In this metric, the higher the result, the greater the difference between the average precision of each object, therefore, it will be a marker to take into account to analyse the results of the SGG method, since it could be affected by the lack of precision of the object detector.

The Performance degradation obtained for Visual Genome categories is $PD = 0.6448$, since the range of the metric is $[0, 1]$ denotes what we expected from the Average Precision per object observation. In the case of Action Genome, the Performance Degradation $PD = 0.653$, which is similar to the Visual Genome one. In Action Genome, we decide to quit the *person* category due to the fact that the dataset focuses on Human-Object Interaction (HOI)s and, therefore, this category had a high prevalence compared to the other categories in number of appearances and caused a notable distortion in the results.

The Performance Degradation can be represented graphically as the area percentage of a rectangle (where the upper left point corresponds to the maximum AP and its lower right point is the minimum AP) and the line generated by the entire set of APs. So, the Performance Degradation graphical representation for Visual Genome can be seen in Figure 4.15 and, for Action Genome, can be seen in Figure 4.16.

After seeing the results obtained in the Performance Degradation, and to try to explain the difference between the Average Performances obtained for each object, we looked for a correlation between the number of occurrences and the Performance obtained. To do this, first a tour of all the annotations of the two datasets was made, noting how many appearances each object had. Subsequently, the Pearson correlation [46] between the vectors of the number of appearances and AP for each object category. The results can be seen in table 4.2

Dataset	Pearson coefficient
Visual Genome	0.0619
Action Genome	0.3801

Table 4.2: Pearson correlation results obtained for each dataset.

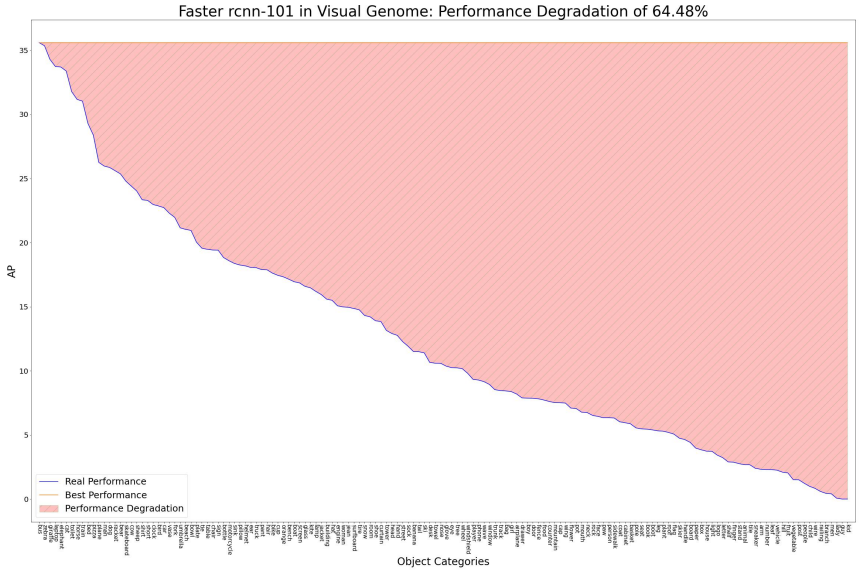


Figure 4.15: Performance Degradation graphical representation for Visual Genome.

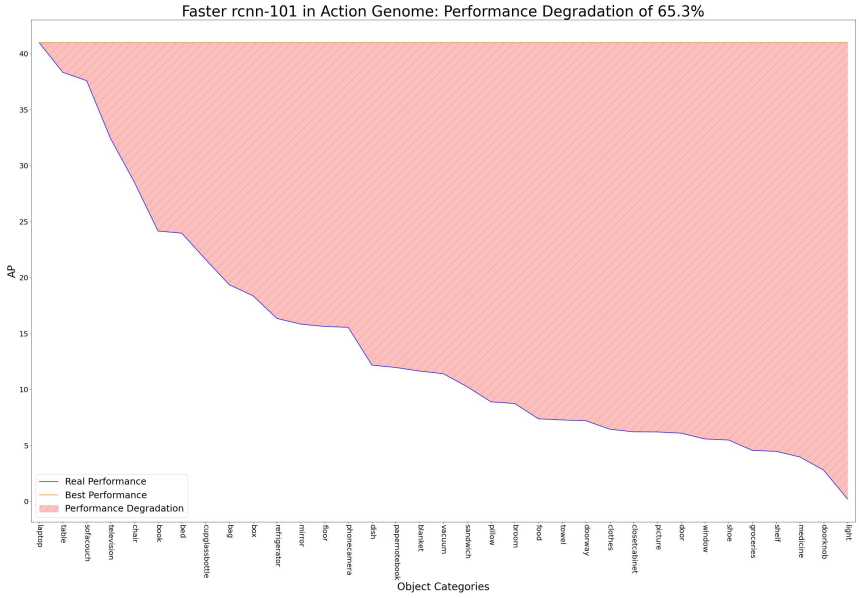
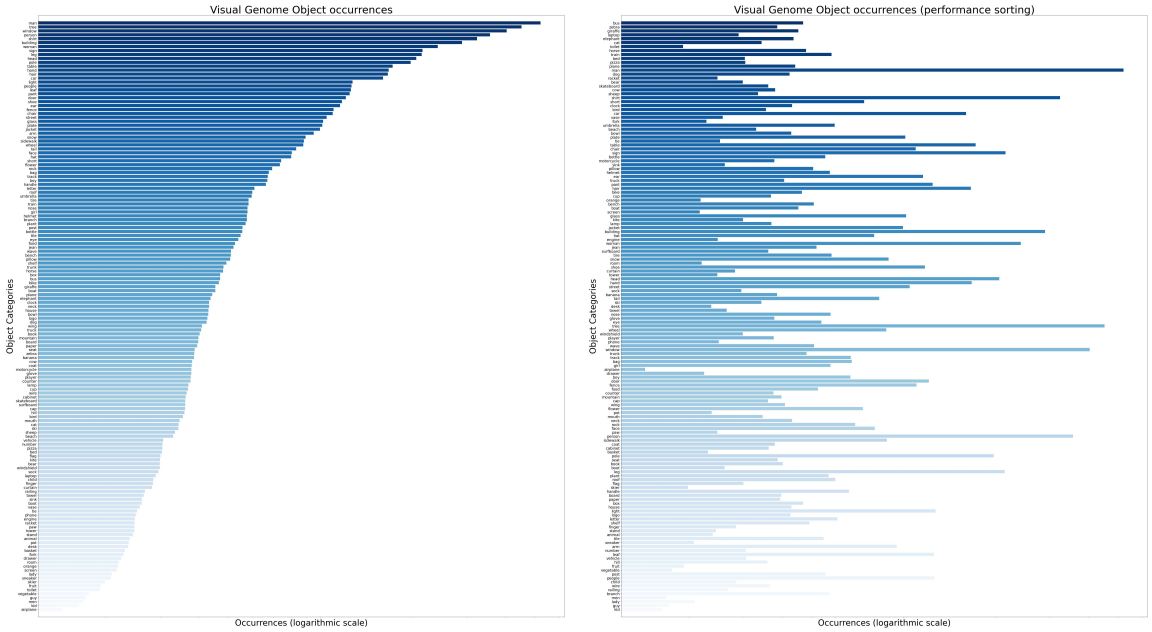


Figure 4.16: Performance Degradation graphical representation for Action Genome.

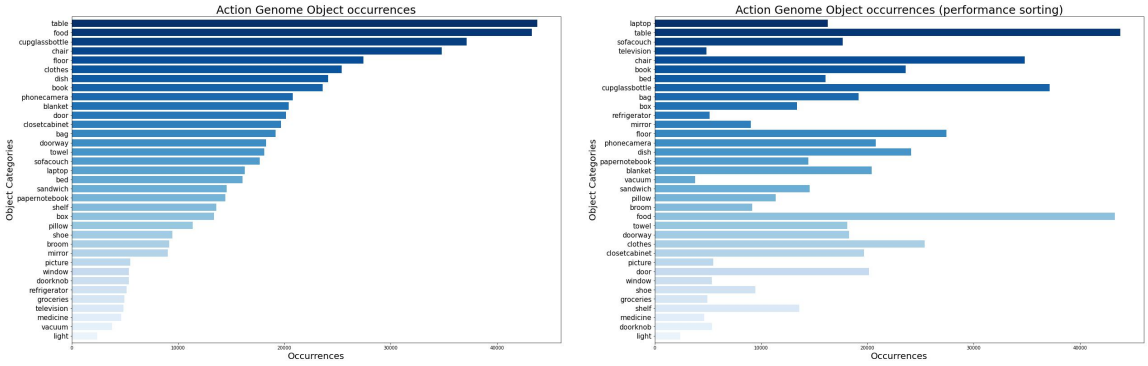
If we focus on the Pearson correlation results obtained, we can extract that in Visual Genome there is no significant correlation between the number of occurrences and the average precision obtained for each object category. On the other hand, in the case of Action Genome, we can extract a significant positive correlation between occurrences and average precision. So we can infer that, in the case of Action Genome dataset, we could be able to obtain better detection results by adding more training instances of each object category.

Additionally, In order to graphically observe the results obtained by Pearson correlation, in figure 4.17 (for Visual Genome) and in figure 4.18 (for Action Genome) we propose the following structure. For each dataset, the occurrences, and the average precision, of each object category are counted and sorted in a decreasing order. After the data extraction step, the sorted occurrences of each object categories is plotted. Finally, next to the previous plot, the occurrences of each object category are plotted, but with the object class ordering by the average precision results.



(a) Object occurrences (descending sorting) (b) Object occurrences (AP sorting)

Figure 4.17: Object occurrences and Average Precision correlation for Visual Genome



(a) Object occurrences (descending sorting) (b) Object occurrences (AP sorting)

Figure 4.18: Object occurrences and Average Precision correlation for Action Genome

4.5 RelDN Training

With the object detection model trained (in our case a Faster RCNN model), it is time to train the model that will establish the relationships between the objects detected by the object detector. Our work bases its analysis on the model in the structure presented by *Ji Zhang et al.* in [69] (already explained in section 3.2.4), the RelDN model.

It is important to mention that the scene graph models are computationally less expensive to train than the detector object, that is, they will not require as many resources. Even so, RelDN, being based on a structure with four losses, is located in a range of high iterations to be trained within the generation models of scene graphs.

To better understand the tests proposed in our approach, it is important to highlight a couple of key aspects of the basic design of the RelDN model. The first important aspect is that said model proposes a structure based on three branches: semantic, spatial and visual (full explanation in section 3.2.4). In the semantic branch, RelDN proposes the use of a three-dimensional matrix (subject, object, relationship) where the probability of co-occurrence of said elements is established.

In previous approaches we used to use the same convolutional layers used for the detector object to train the relationship generator. But in a recent study [19], it has been shown that the use of different backbones helps in the final performance. In the project case, this RelDN backbone is a ResNet-50 [20]. This structure is usually named as Decoupled SGG model, a conceptual example of the difference between both structures can be seen in the following Figure 4.19.

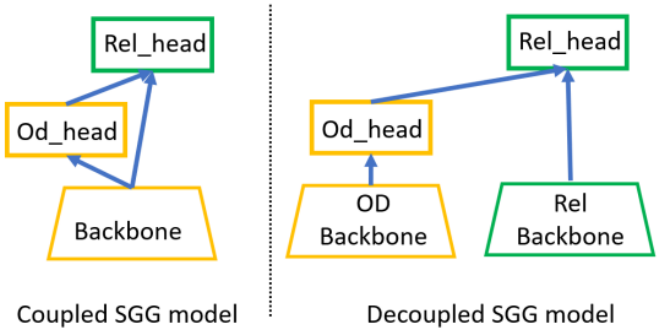


Figure 4.19: Illustration of coupled and decoupled SGG model architecture. [19]

In relation to the treatment given to the prior knowledge matrix of the semantic branch of the RelDN model, we proposes a set of three different RelDN models to train:

- A first model is trained with the prior knowledge matrix without any variation, that is, as the original authors presented the structure. This model will serve as a basis to see the influence the changes proposed by our work have.
- A second model will be trained with the modified prior knowledge matrix using our proposed GAG technique. The GAG used in this method is only trained using instances of the Visual Genome dataset itself. This model seeks to measure the effect of the GAG technique.
- Finally, a third RelDN model will be trained using the prior knowledge matrix modified with the GAG technique proposed in this work. But in this case the matrix generated by GAG will be the one obtained through the analysis of different datasets (e.g., HICO, Visual Genome, Action Genome or Homage).

The three models exposed above will be trained using the same hyperparameters in all cases, only changing the prior knowledge matrix. In order to be able to evaluate the effect of changes in it without possible noise generated by other parameters. All of them are trained using a learning rate and the epochs were set to 10^{-3} and 10000, respectively. The learning rate was decreased utilizing a weight decay of 10^{-4} at epoches 2500 and 6000. Each batch of data consists of two images and a checkpoint of the model is stored every 500 epochs. Since the validation step is highly resource demandant, it was performed not performed during the training stage. Additionally, for each model, a batch size of 4 images for training and 1 images for testing stage is used.

After training each RelDN model, we were able to observe that the losses during training remained extremely constant and low, except for the loss of the relationship classifier, which

maintained a sustained decline. We have also observed that this type of models do not require long training times, unlike object detectors.

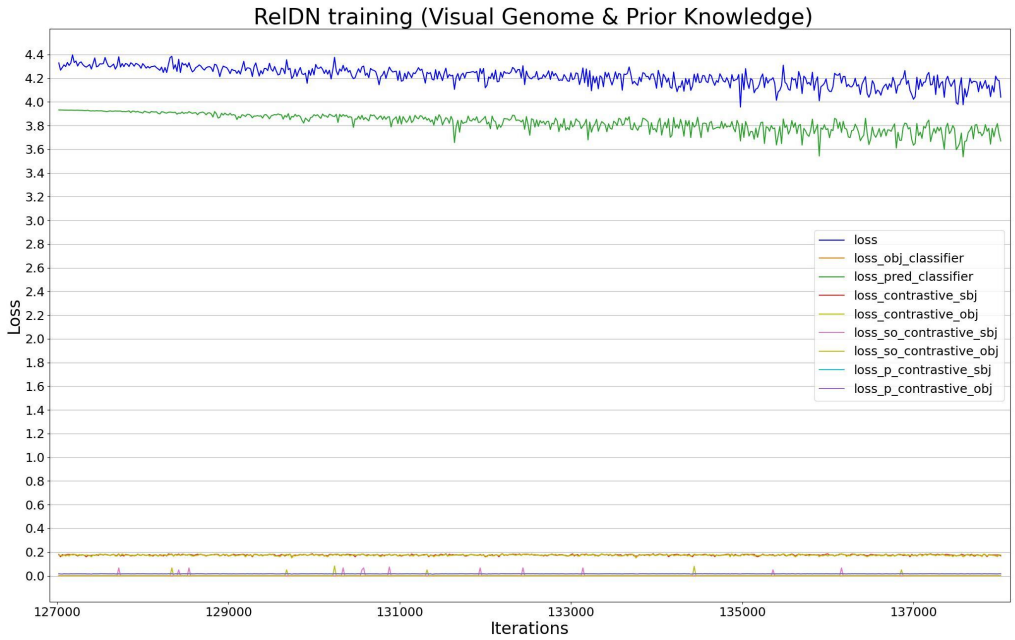


Figure 4.20: RelDN with Prior Knowledge matrix Training losses

So, the results of each model are shown in Figures 4.20 to 4.22: Figure 4.20 for the RelDN trained with the prior knowledge matrix, Figure 4.21 for the RelDN model trained with locally extracted Common-sense knowledge, and Figure 4.22 for the RelDN model trained with globally extracted Common-sense knowledge. It is important to mention that the epochs of each graph do not start from epoch 1, since the iterations performed during the training of the object detector are not shown in the figures.

In each plot a set of losses are displayed. In order to understand what mean each one it is important to remember that RelDN is composed by four losses (as it is explained by Equation 3.4, in section 3.2.4). Therefore, each loss can be understood as:

- `loss_obj_classifier` is the one extracted from object detector.
- `loss_pred_classifier` represents RelDN L_0 loss.
- `loss_contrastive_sbj` and `loss_contrastive_obj` compose RelDN L_1 loss.
- `loss_so_contrastive_sbj` and `loss_so_contrastive_obj` describe RelDN L_2 loss.

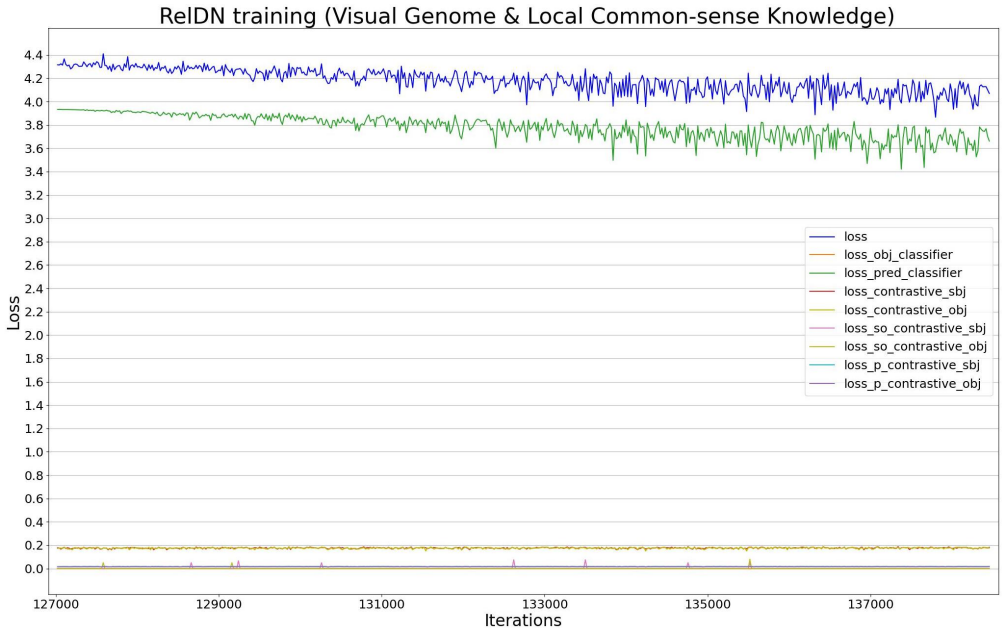


Figure 4.21: RelDN, with Locally extracted Common-sense matrix, Training losses

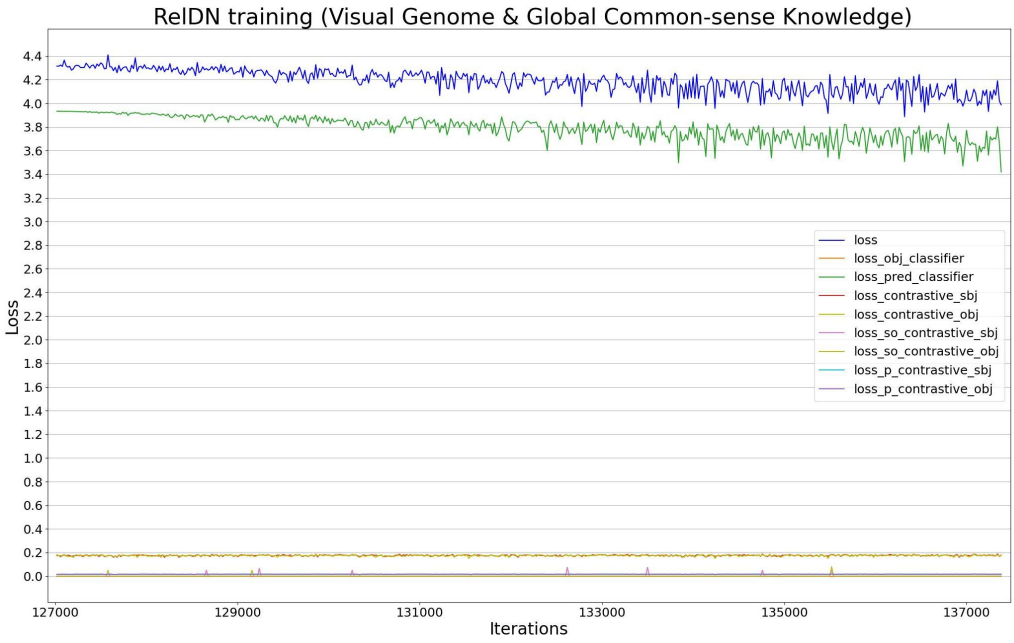


Figure 4.22: RelDN, with Globally extracted Common-sense matrix, Training losses

- `loss_p_contrastive_sbj` and `loss_p_contrastive_obj` represents RelDN L_3 loss.
- `loss` is the sum of all the other losses.

In the three images we can observe very similar shapes. In general, we can highlight the stability suffered by all the losses related to L_1, L_2 and L_3 in which the values remain almost constant throughout the training, with a very slight descent. On the other hand, the loss L_0, or los_pred_classifier, maintains a notable but sustained decline, upon reaching epoch 8000, we can see that the loss begins to oscillate nobly, due to this fact we decided to stop training at 10000 epochs.

It is also observable that the proposed method, GAG, does not greatly modify the behaviour of losses during training. In any case, we can denote a very slight increase in the oscillation of the loss_pred_classifier.

4.6 Obtaining results from State-of-the-art methods

In order to compare the results obtained in the exploration proposed for the project, a set of state-of-the-art methods have been chosen with the objective of said comparison. To classify them, we could distinguish them into two main groups: (1) vanilla state-of-the-art methods (all of them explained in section 3.2), (2) state-of-the-art methods with modifications to apply common-sense knowledge (explained in section 3.3).

Models	Other metrics results origin	Could mPD be analysed?
Neural Motifs	[67]	X
Graph R-CNN	[61]	X
KERN	[15]	✓
motifs + CogTree	[64]	X
motifs + VDS (DS)	[62]	X
motifs + VDS (SS)	[62]	X
GBNET	[65]	✓

Table 4.3: Compared state-of-the-art SGG methods. *Other metric results origin* column provides the link to the origin of the metric results used for the comparison; *Could mPD be analysed?* column shows for which models we are able to measure the mPD metric.

Table 4.3 summarizes the availability of results for the chosen State-of-the-art methods. Due to the fact that during the analysis of some state-of-the-art methods we have had problems accessing the checkpoints or/and the repositories provided, which they required deprecated libvailability of raries, some results have been directly extracted from the methods' original papers. At the same time, due to the above casuistry, together with the fact

that the mPD metric was recently published, in some cases it has not been possible to calculate the mPD metric.

4.7 mPD metric implementation

Mean Performance Degradation (mPD) [32], was proposed by *Liu et al.* in February 2022, to evaluate the performance gap among compositions of different objects and the same verb. mPD reveals that the current state-of-the-art SGG do not generalize properly.

The aim of this metric is to evaluate the degradation of the Average Precision (AP) of each object given a relation. This particularity, allow us to compare the gap from the better classified object to the worst. In mPD is a HOI centred metric, so it was created to only evaluate this kind of relations between humans (subject) and objects.

As can be seen in Figure 4.23, in the plot, we are able to observe how the AP highly decrease through the different objects, given the relation *ride*. Another key point that can be observed is that even the best classified object and the worst (horse and cow, in this case) are very close visually, and semantically, the model is not able to generalize properly.

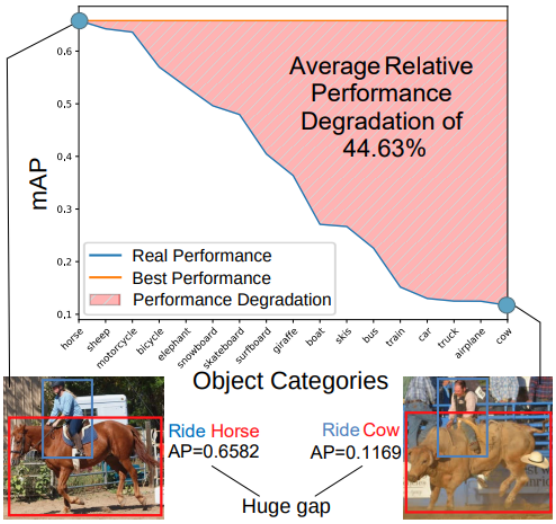


Figure 4.23: mPD output example. Performance degradation of *ride* relation overall the predictions

So, mPD can be defined as in the following equation 4.2, extracted from [32]:

$$mPD = \frac{1}{|V|} \sum_{v \epsilon V} \frac{AP_{\langle v, o_{max} \rangle} - \bar{AP}_v}{AP_{\langle v, o_{max} \rangle}} \tag{4.2}$$

where, for each verb $v \epsilon V$:

- the subset object categories available for v is defined as O_v,
- the average precision for a HOI composition $\langle v, o \rangle$ is defined as $AP_{\langle v,o \rangle}$,
- the object with better average precision is defined as o_{max}, or in other words $o_{max} = argmax_o AP_{\langle v,o \rangle}$,
- and, $\bar{AP}_v = \frac{1}{|O_v|} \sum_{o \epsilon O_v} AP_{\langle v,o \rangle}$ define the mean average precision (AP) for compositions $\langle v, o \rangle$ where o is inside O_v.

We have built an implementation of this metric to perform the calculation exposed in the formula explained above. Additionally, two possibilities were also added that did not come in the version given by the authors in their GitHub repository ([60]); (1) possibility of specifying a set with the K-best predictions per image to be selected. (2) The option of not only processing HOI, but also other types of relationships in datasets containing other types of relationships (e.g., Visual Genome) has been enabled. Our implementation is described in Algorithm 6.

The input parameters are composed of the ground truth triplets and bounding boxes, the predicted triplets and bounding boxes which are separated by images. Additionally, the k values set is required, this variable keeps the different values of K, which are the predicted triplets' subset, by image, that will be selected to evaluate.

The outputs extracted are the average precision matrix where for each $\langle v, o \rangle$ tuple the AP is calculated. Additionally this matrix has a third dimension which corresponds to the results extracted from each K. The following parameter called $pdMatrix$ is generated by calculating the Performance Degradation for each action separately with additional k dimensions. Finally, mpd is the mean of the $pdVector$ for each k.

4.8 Chapter Summary

In this chapter, the methodology followed to build the experiments and all its parts are explained. Starting with the datasets, following with the methodology used to build prior_frequence matrix and Generalized Action Graphs (GAG). Later, the Faster-RCNN and SGG models training methodology are explained, and ending with a description of the

Algorithm 6: Mean Performance Degradation Algorithm

Data: gtTriplets, gtBboxes, predTriplets, predBboxes, kSet, imageSet
Result: apMtx, pdMtx, mpd
$Objects \leftarrow Load(DatasetObjectList)$;
$Predicates \leftarrow Load(DatasetPredicateList)$;
$hitMatrix \leftarrow EmptyMatrix(dim : K * dim(Objects) * dim(Predicates))$;
$nPredMatrix \leftarrow EmptyMatrix(dim : K * dim(Objects) * dim(Predicates))$;
for *image in imageSet* **do**
  for *k in kSet* **do**
    $predTriplets_{image,k} \leftarrow predTriplets_{image}[: k]$;
    for *predTriplet in* $predTriplets_{image,k}$ **do**
      $nPredMatrix[k, predTriplet[v], predTriplet[o]] += 1$;
      if *predTriplet in* $gtTriplets_{image}$ **then**
        $gtTriplet \leftarrow getGtTripletEqual(image, predTriplet)$;
        $IoU \leftarrow calcIoU(gtBboxes_{image,}, predBboxes_{image,predTriplet})$;
        if $IoU = 0.5$ **then**
          $hitMatrix[k, predTriplet[v], predTriplet[o]] += 1$;
$apMtx = hitMatrix/nPredMatrix$;
$pdMtx \leftarrow EmptyMatrix(dim : K * dim(Predicates))$;
$mpd \leftarrow EmptyMatrix(dim : K)$;
for *k in kSet* **do**
  for *prd in Predicates* **do**
    $pdMtx_{k,prd} \leftarrow Max(pdMtx_{k,prd}) - Mean(pdMtx_{k,prd})/Max(pdMtx_{k,prd})$;
  $mpd_k = Mean(pdMatrix_k)$;

mean Precision Degradation (mPD) metric algorithm that has been implemented.

The following chapter is devoted to show the experimentation results and the analysis that we can extract from them, both quantitatively. Additionally, an explanation of the used metrics and experiments setup is given.

Chapter 5

Experiments and results

This chapter has the objective of showing and analysing the results obtained with the proposal presented in this project, as well as comparing the results with the methods of the current state of the art.

To do this, in sections 5.1 and 5.2 a brief description is made of the hardware setup used to train the different parts of the SGG model and to run the experiments, followed by the metrics used to analyse the data. Subsequently, in sections 5.3, a quantitative analysis of the results obtained is carried out, together with the results extracted from other models. This analysis aims both to look at the performance of each proposal, as well as to analyse its generalization capacity.

5.1 Experiments Setup

All the experiments were run on Mininostrum, which is a server under the Barcelona Super Computer group domain. Mininostrum ihardware setup is composed by:

- CPU: Intel core i7 5820k
- 2 NVIDIA 3090 GTX GPU Cards.
- RAM: 64GB DDR4.

This amazing equipment, without any doubts can help with the training procedure, and maybe can influence the final results obtained, due to the major capacity of training at our disposal.

5.2 Used Metrics

During the realization of the work, many different measures are used to evaluate each part of the work. In literature the evaluation of SGG methods is made though a variety of metrics such as Recall@K, Top@K Accuracy or No Graph Constraint Recall@K (see the complete list in [54]), in the case of this work the evaluation is based on Recall@K, Mean Recall@K and mPD.

Foremost, the following three predictions are used:

- Predicate Classification (PredCls): it takes the ground-truth object labels and bounding boxes for relationship prediction, it only measures the predicate prediction.
- Scene Graph Classification (SgCls): it takes ground-truth bounding boxes for object and relationship prediction.
- Scene Graph Detection (SgDet): it predicts scene graphs from scratch, without the usage of prior knowledge, prediction the object/subject location, its classification and predicting the predicate between them. mean Performance Degradation (mPD) uses the same predictions as SgDet, but mPD differs on the metric building.

After the predictions are performed, for PredCls, SgCls and SgDet, the recall is calculated (Equation 5.1):

$$recall = \frac{|groundTruth_{triplets} \cap predictedtriplets|}{groundTruth_{triplets}} \tag{5.1}$$

Recall@K is the mostly used over all the papers but it can suffer from bias (e.g., in the Visual Genome (4.1.1) dataset it is biased towards dominant predicates). If the 10 most frequent predicates are correctly classified, the accuracy would reach 90% even if the rest 40 predicates are all wrong. So *Chen et al.* [10] and *Tang et al.* [55] in 2019, propose the Mean Recall@K, which calculates Recall@K for each predicate category independently then report their mean. (Equation 5.2)

$$meanrecall = \frac{1}{P}\sum_{p=1}^{P} \frac{|groundTruth_{triplets_p} \cap predictedtriplets_p|}{groundTruth_{triplets_p}} \tag{5.2}$$

where p are each one of the predicate in the dataset. In the case of mPD@K, the precision is calculated for all the predicates independently, as can be seen in Equation 5.3, after it, the main mPD@K equation is used (see a complete explanation of mPD in Section 4.7):

$$meanprecision = \frac{1}{P}\sum_{p=1}^{P} \frac{|groundTruth_{triplets_p} \cap predictedtriplets_p|}{predicted_{triplets_p}} \tag{5.3}$$

where p are each one of the predicate in the dataset. Top k of predicted triplets is used in order to avoid inserting every possible combination as a result. Usually, this k is a fix set of [20, 50, 10], which means that only the most reliable 20, 50 or 100 predicted triplets are used in the recall/precision formula.

In the case of the predictions SgDet and mPD that evaluate the correct localization of the object bounding boxes, an IoU of 0.5, between predicted and ground truth bounding boxes, is performed in order to measure that the system is identifying correctly the position.

5.3 Quantitative Analysis

After having exposed all the metrics that are going to be used to measure the results of the different SGG algorithms in the previous section, in this section the results obtained are going to be exposed and analysed. For this purpose, we are going to use two different tables. In the first table, for each model included in the comparison, its results in the most standardized metrics for the analysis of SGG models will be exposed. Subsequently, in a second table, the results obtained in the metric implemented in this project, mPD, will be analysed.

In table 5.1, we can see the results obtained for each model. This project has proposed to group these models into three categories: (1) state-of-the-art models used as baselines, (2) proposed models that contain modifications related to the field of common-sense knowledge, and (3) models proposed in this work. A particular case is the HORL, which, although it was proposed by Kevin Rosales[15], has also been retrained in order to obtain more results. The results obtained for the Bases and Common-sense adaptations categories were directly extracted from the original papers, shown in table 4.3. For each model, its development is analysed in a total of three tasks; *PredClass*, *SGClass* and *SgDetection*. For each of them, the results are shown after applying Recall and the mean Recall (which differs from the first one in that the Recall is calculated for each relationship separately and then the average is made). For each metric, the values of $K = 50$ and $K = 100$ are used.

In general terms, in the metrics analysed during the comparison of results, we can observe a notable difference between the results obtained by the recall and the mean recall. Since the mean recall is calculated by calculating the mean of the recall obtained by each relation individually, we can observe that this value is much lower than recall due to a very poor distribution of correct cases. In other words, in some relationships (the minority) the value of the recall is much higher than the rest. This fact denotes the importance of each action, the action cases available in the images, and differentiation from the rest of the actions. Another possible explanation for this differentiation between relations may be the training cases available in the dataset, in other words, the Long Tail problem [70], which is

	Models	PredClass				SGClass				SgDetection			
		R@50	R@100	mR@50	mR@100	R@50	R@100	mR@50	mR@100	R@50	R@100	mR@50	mR@100
Bases	Neural Motifs[67]	65.2	67.1	14.0	15.3	35.8	36.5	7.7	8.2	27.2	30.3	5.7	6.6
	Graph R-CNN[61]	54.2	59.1	-	-	29.6	31.6	-	-	11.4	13.7	-	-
	KERN[10]	65.8	67.6	17.7	19.2	36.7	37.4	9.4	10.0	27.1	29.8	6.4	7.3
CS adapt	motifs + CogTree[64]	35.6	36.8	26.4	29.0	21.6	22.2	14.9	16.1	20.0	22.1	10.4	11.8
	motifs + VDS (DS)[62]	53.40	56.54	37.68	41.98	26.12	27.46	17.20	18.39	23.69	25.59	13.84	15.23
	motifs + VDS (SS)[62]	76.28	77.98	60.20	63.61	35.93	36.47	28.07	30.09	33.94	37.26	23.90	28.06
	GBNET[66]	66.6	68.2	22.1	24.0	37.3	38.0	12.7	13.4	26.3	29.9	7.1	8.5
GAG	RelDN[69]	20.54	24.66	0.94	1.31	11.23	12.99	1.41	1.91	9.10	10.32	0.10	0.05
	RelDN + GAG (VG)	14.12	17.80	1.71	2.57	7.63	9.31	0.94	0.49	7.63	9.31	0.94	0.49
	RelDN + GAG (global)	13.80	17.27	1.64	2.46	7.45	9.10	0.85	1.23	1.64	2.39	0.20	0.38
	HORL	91.59	91.80	-	-	54.13	54.93	-	-	47.80	56.64	-	-

Table 5.1: Quantitative comparison between, the state-of-the-art SGG methods (Bases), with Common Sense variations (CS adapt) and the ones proposed in this work (GAG). All the results are based on Visual Genome dataset [28], except the HORL method, which is based on Action Genome[24]. *The results are extracted from the original papers of each one of the methods.

suffered by the datasets most used by the state of the art. This problem may occur when a dataset has a long set of options, in this case relations, and do not present enough training cases, or not sufficiently representative cases, to allow the models train with it, a good classification between all of them. In short, the current models prove to have a long way to go before obtaining promising results.

Another noteworthy aspect is that the results obtained by models that use Commonsense knowledge are higher than in the case of the vanilla models. This fact may be due to the fact that they are modifications of the vanilla model, they start from improving what is already established. But we can also observe that they show a smaller difference between the results obtained by means of the recall and the mean recall.

If we focus on the analysis of the tasks (*SgDet*, *PredClass* and *SgClass*) we can observe better results in *PredClass*, slightly lower results in *SgClass* and the lowest results are always those of *SgDet*. This can be explained by the parameters used to calculate each task. In the case of *PredClass*, only the prediction of the relationship (label) between the objects is evaluated, in the case of *SgClass*, apart from what was previously evaluated, the classification of the objects in the scene made by the object detector is added. Finally, *SgDet* evaluates both the relationship between the objects in the scene, as well as the object class classification and their location in the image. With these results, it can be observed that the accuracy of the object detector has an influence on the results of the SGG models.

Looking at the results obtained in this project we can see a large decrease in the values obtained in the metrics, this fact may be due to human error due to the complexity of the repository used as a base to carry out these experiments and a lack of material time to execute solutions properly.

In any case, the results obtained can be analysed internally between the base model and the variants proposed by this work. During the experiments we have been able to observe that the values obtained by the base model exceed those obtained by the approximations of the work in terms of recall, but in the ore recall we see the opposite case.

In the following table 5.2, the results obtained in mean Precision Degradation (mPD) metric will be exposed. This metric uses the Performance for each relation category instead of the Recall (this element will be analysed later), for this project the same K values have been used as in the previous table. Finally, mention that for some of the models included in the previous comparison it has been impossible to carry out an evaluation of said metric, due to the code obsolescence of the repositories made available by the authors of the papers.

	Models	mPD ↓	
		AP@50	AP@100
mPD Evaluated	KERN[10]	42.43	47.35
	GBNET[66]	51.60	58.71
	RelDN[69]	9.10	10.32
	RelDN + GAG (VG)	12.62	14.11
	RelDN + GAG (global)	10.47	10.55

Table 5.2: Quantitative mPD results comparison between the state-of-the-art SGG methods, with Common Sense variations, the ↓ means that in mPD lower is better.

In the case of the evaluation of the mPD metric, we must consider that the lower the value in the metric is, the better is the result. A lower value in mPD means that the distance between the best accuracy obtained by an object in each relation and the average accuracy in that relationship is smaller. In our case, we can observe that both RelDN and the approximations made through GAG obtain a much better result than in comparison to the other methods, indicating a better generalization for these models.

Entering to analyse the results between the approaches presented by this work and its base model, we see that our approaches do not manage to improve the results obtained by the RelDN model. Although the difference is small, especially in the case of *GAG (global)* whose results are very close to those obtained by the base model. It is necessary to mention that this result may have been affected by the results obtained by those methods in table 5.1.

After having studied the theory behind the metric, implementing it and using it in practical cases, we can extract some positive and negative aspects of the use of the mPD metric.

Positive aspects include:

- the metric tries to focus on the generalization directly, explicitly evaluating that property.
- it is easy to understand and extract interesting data from the generalizability of a model
- the graph provided by the authors allows it to be understandable even for people outside the field of research, a fact that would increase the interpretability of the results for non-expert people.

On the other hand, after using the metric we can mention some of its limitations:

- being based on precision, is directly affected by the number of predictions that your model generates. This last fact collides with the usual procedure in the SGG, in which the models generate a fixed and high number of predictions that are later ordered by their score, after that, a subset is selected to be evaluated (usually as *K@20*, *K@50* and *K@100*).
- the metric focuses on the HOI analysis, so the subject of the generated triplets is left out of the calculation, since in HOIs the subject of the relationship is always a person. This fact conditions its use in other disciplines, such as the analysis of general SGG.

5.4 Chapter Summary

In this chapter, we have been able to carry out a complete analysis of the results that we have obtained during the realization of the project. First, the setup where the experiments have run has been analysed (5.1), then the metrics used to perform the different analyses have been explained in detail (5.2). Finally, a quantitative analysis has been analysed based on the metrics explained above, as well as comparing the results obtained with other state-of-the-art methods (5.3).

Next chapter is devoted to the conclusions drawn after carrying out the work. Distinguishing some general conclusions, an analysis of the final state regarding the initial hypotheses (6.1) and finally a section to expose possible extensions of the research carried out in this project (6.2).

Chapter 6

Conclusions and future work

This chapter aims to present the conclusions of the project after its completion and analysis of the results. These conclusions are separated into two main sections.

First, some general conclusions drawn from the realization of the project are exposed, based on the hypotheses raised at the beginning. Observing if these hypotheses were accomplished or if, on the contrary, they were not fulfilled, it corresponds to section 6. Finally, in section 6.2, possible future investigations are exposed that can continue with the one carried out in this work, as well as concepts that due to lack of time have been left without deepening too much.

6.1 After-work Conclusions

SGG is a fascinating, as well as novel, new field of research. SGG mixes fields such as computer vision, knowledge structuring, trying to enable the first steps to create systems that understand what actually happens in an image or video.

On the other hand, the application of common sense knowledge to SGG is understood by the prior bias problem suffered by the vast majority of current state-of-the-art methods, which often base their prediction too much on knowledge extracted from the same dataset, which generates a bias. The theory that this project has defended, located within the field of common-sense knowledge, is the use of knowledge bases external to the dataset to infer prior knowledge that does not contain this bias.

In particular, the technique presented in this work, GAG, sought to reduce the bias contained in current state-of-the-art methods, reducing their dependence on the cases that appeared directly in the analysed dataset. After the completion of this work, we can see that GAG does not help the purpose for which it was intended, worsening the results in

the different recalls and the mPD. This fact may be due to the fact that GAG modifies the prior knowledge table of the model, making the probability abstractions of that table contain more possibilities than the ones it initially had with only prior knowledge. This fact can cause an ambiguity of different object categories instead of helping to generalize them.

Although the effort has been put into fixing the results obtained, it has been impossible due to the short time to carry out the technical tests and the complexity of the tests to be carried out, as well as the time necessary for each one of them. At the date of this report, the results obtained are far from the desired ones, but they can give us an idea of how the technique presented in this work affects the performance and the generalization of the models modified with it. Serving, at the same time, as a first look at the mPD metric.

Making a final analysis of the hypotheses initially presented by this project, based on the results obtained, we can describe that:

- Hypothesis 1: The state-of-the-art SGG methods have a low level of object generalization. We have been able to demonstrate that this hypothesis could be true since, as we have seen in table 5.2, the state-of-the-art models obtained values greater than 50%, a fact that is not negligible.

- Hypothesis 2: Using the semantic hierarchy of the objects in the dataset, this object generalization per action can be improved, not reducing the model accuracy. This second hypothesis has not been fully demonstrated, since, as has been seen in the results obtained, the GAG technique does not manage to improve the results obtained by the base model in terms of recall. But if we see the results obtained in the mean recall, we can see a great improvement compared to the base model.

- Hypothesis 3: Giving a richer input data to the GAG method in each action analysed, it better generalizes the final SGG model. We have been able to see that given a larger dataset the mPD improves, but the precision is slightly reduced, this hypothesis has not been fully demonstrated.

- Hypothesis 4: Our technique can help to the model generalization, such as other state-of-the-art common-sense techniques. This last hypothesis has been partially proven since the model manages to improve the mean recall of the base model, denoting a better approximation, but the results for the mPD are slightly worse.

Finally, it is important to mention that this field suffers from a serious code staleness problem. Throughout the project we have come across countless times open source repositories that are only two or three years old, due to library updates and hardware incompatibilities that have delayed the progress of this project too much.

6.2 Future work

However, after the completion of the project, some aspects have remained with questions marks due to they require a big deviation from the scope of the project and/or lack of time to go deeper in the investigation. These questions may be future work for future research in the field of SGG. For example, during the project we focused on parsing the hypernyms of each synset-object with respect to an action, but WordNet[57] provides other fields that can also help with generalization. Such as hyponyms, definitions, examples of use or related synsets, information also existing within WordNet[57].

Other aspects that emerged during the project were the discovery of tools that would allow better performance and open the door for new research. In this regard, we would highlight ConceptNet[51] knowledge graph, which stands out over WordNet because, apart from this platform, it brings together knowledge from many different projects and platforms, forming a very powerful knowledge graph. Another case would be the discovery of HOMAGE[39] dataset, whose properties of relating different views of the same scene, including an egocentric view, added to the definition of the scene graph and atomic actions for each scene, position it as a more than likely basis for future work.

Finally, other aspects in which future research could delve deeper are, for example, the use of more sophisticated optimization techniques for multi-objectives, such as evolutionary algorithms, and analysing their performance in the GAG method. Another field in which new avenues of research are opening up is the use of the mPD metric to analyse more SGG methods, as well as possible adaptations so that it can be used for all aspects of possible scene graphs and not just Human-Object Interaction.

Bibliography

[1] Akash Agnihotri. Attending to attention, 2021. URL: https://towardsdatascience.com/attending-to-attention-eba798f0e940.

[2] Natural Language Processing Lab at Tsinghua University. Distant supervision for scene graph generation github repository, 2021. URL: https://github.com/thunlp/VisualDS.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL: http://arxiv.org/abs/1409.0473.

[4] Stephan Baier, Yunpu Ma, and Volker Tresp. Improving visual relationship detection using semantic modeling of scene descriptions. In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 2017. doi:10.1007/978-3-319-68288-4_4.

[5] Seyedmostafa Sheikhalishahi Claudio Modena Venet Osmani Behrooz Mamandipoor, Mahshid Majd. Monitoring and detecting faults in wastewater treatment plants using deep learning. *Environmental Monitoring and Assessment*, (192), 2020. doi:10.1007/s10661-020-8064-1.

[6] Nechu BM. What is an encoder decoder model?, 2020. URL: https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a.

[7] Jason Brownlee. An introduction to long short-term memory networks by the experts, 2017. URL: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/.

[8] Yuan Chai. Scene graph transformer with cogtree loss in pytorch, 2021. URL: https://github.com/CYVincent/Scene-Graph-Transformer-CogTree.

[9] Yu-Wei Chao, Zhan Wang, Yugeng He, Jiaxuan Wang, and Jia Deng. HICO: A benchmark for recognizing human-object interactions in images. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. doi:10.1109/ICCV.2015.122.

[10] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6156–6164, 2019. URL: http://arxiv.org/abs/1903.03326, doi:10.1109/CVPR.2019.00632.

[11] Joshua Chin. ConceptNet, an open, multilingual knowledge graph (online browser), 2019. URL: https://conceptnet.io/.

[12] Stefania Cristina. The attention mechanism from scratch, 2022. URL: https://machinelearningmastery.com/the-attention-mechanism-from-scratch/.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi:10.1109/CVPR.2009.5206848.

[14] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

[15] Francesco Maria Turno Emil Bækdahl, Weihao Yu. KERN github repository, 2022. URL: https://github.com/yuweihao/KERN.

[16] Joshua Chin et al. ConceptNet, an open, multilingual knowledge graph (github repository), 2019. URL: https://github.com/commonsense/conceptnet5.

[17] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, December 2015.

[18] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. doi:10.1109/CVPR.2014.81.

[19] Xiaotian Han, Jianwei Yang, Houdong Hu, Lei Zhang, Jianfeng Gao, and Pengchuan Zhang. Image scene graph generation (SGG) benchmark. *CoRR*, abs/2107.12604, 2021. URL: https://arxiv.org/abs/2107.12604, arXiv:2107.12604.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi:10.1109/CVPR.2016.90.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. doi:10.1162/neco.1997.9.8.1735.

[22] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3588–3597, June 2018. doi:10.1109/CVPR.2018.00378.

[23] Alphabet Inc. Google knowledge graph search api, 2021. URL: https://developers.google.com/knowledge-graph.

[24] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020.

[25] Daniel Johnson. Back propagation in neural network: Machine learning algorithm, 2022. URL: https://www.guru99.com/backpropogation-neural-network.html.

[26] kcarnold. ConceptNet 5.7.0, python library, 2019. URL: https://pypi.org/project/ConceptNet/.

[27] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL: http://arxiv.org/abs/1609.02907, arXiv:1609.02907.

[28] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David Shamma, Michael Bernstein, and Fei-Fei Li. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123, May 2017. doi:10.1007/s11263-016-0981-7.

[29] Kacper Kubara. Charades: A dataset which guides our research into unstructured video activity recognition and common-sense reasoning for daily human activities., 2016. URL: https://prior.allenai.org/projects/charades.

[30] Kacper Kubara. Introduction to message passing neural networks, 2020. URL: https://towardsdatascience.com/introduction-to-message-passing-neural-networks-e670dc103a87.

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Zitnick. Microsoft COCO: common objects in context. In *European Conference on Computer Vision*, volume 8693, April 2014. doi:10.1007/978-3-319-10602-1_48.

[32] Xinpeng Liu, Yong-Lu Li, and Cewu Lu. Highlighting object category immunity for the generalization of human-object interaction detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36(2), pages 1819–1827, June 2022. doi:10.1609/aaai.v36i2.20075.

[33] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, volume 9905, pages 852–869, October 2016. doi:10.1007/978-3-319-46448-0_51.

[34] George A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, November 1995. doi:10.1145/219717.219748.

[35] Aditi Mittal. Understanding rnn and lstm, 2019. URL: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e.

[36] Peter Oram. Wordnet: An electronic lexical database. christiane fellbaum (ed.). cambridge, ma: Mit press, 1998. pp. 423. *Applied Psycholinguistics*, 22(1):131–134, 2001. doi:10.1017/S0142716401221079.

[37] Panamitsu. Google knowledge graph (wiki article), 2022. URL: https://en.wikipedia.org/wiki/Google_Knowledge_Graph.

[38] Maurice Peemen. Convolutional neural network (cnn), 2021. URL: https://developer.nvidia.com/discover/convolutional-neural-network.

[39] Nishant Rai, Haofeng Chen, Jingwei Ji, Rishi Desai, Kazuki Kozuka, Shun Ishizaka, Ehsan Adeli, and Juan Carlos Niebles. Home Action Genome: Cooperative compositional action understanding. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11179–11188, June 2021. doi:10.1109/CVPR46437.2021.01103.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 1–10, January 2016.

[41] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. URL: http://dx.doi.org/10.1037/h0042519, doi:10.1037/h0042519.

[42] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986. URL: http://www.nature.com/articles/323533a0, doi:10.1038/323533a0.

[43] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[44] Kevin David Rosales Santana. Study and training of deep learning models for scene graph generation from non-static environments, 2022. URL: https://upcommons.upc.edu/handle/2117/362208.

[45] Alessio Sarullo and Tingting Mu. Zero-shot human-object interaction recognition via affordance graphs. *CoRR*, abs/2009.01039, 2020. URL: https://arxiv.org/abs/2009.01039, arXiv:2009.01039.

[46] Philip Sedgwick. Pearson's correlation coefficient. *BMJ*, 345:e4483–e4483, July 2012. doi:10.1136/bmj.e4483.

[47] Xindi Shang, Tongwei Ren, Jingfan Guo, Hanwang Zhang, and Tat-Seng Chua. Video visual relation detection. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, page 1300–1308, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3123266.3123380.

[48] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, January 2018. doi:10.18653/v1/P18-1238.

[49] Igor Shvab. Optimization modelling in python: Multiple objectives, 2021. URL: https://medium.com/analytics-vidhya/optimization-modelling-in-python-multiple-objectives-760b9f1f26ee.

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. Google Research technical report, September 2014. URL: https://arxiv.org/abs/1409.1556, doi:10.48550/ARXIV.1409.1556.

[51] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, February 2017. doi:10.1609/aaai.v31i1.11164.

[52] Danny Sullivan. Google's knowledge graph tripled in size in seven months, 2012. URL: https://www.cnet.com/tech/services-and-software/googles-knowledge-graph-tripled-in-size-in-seven-months/.

[53] Danny Sullivan. A reintroduction to our knowledge graph and knowledge panels, 2020. URL: https://blog.google/products/search/about-knowledge-graph-and-knowledge-panels/.

[54] Kaihua Tang. Scene graph benchmark pytorch github repository (metrics), 2022. URL: https://github.com/KaihuaTang/Scene-Graph-Benchmark.pytorch/blob/master/METRICS.md.

[55] Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. Learning to compose dynamic tree structures for visual contexts. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6621, June 2019. doi:10.1109/CVPR.2019.00678.

[56] Facebook team. Detectron 2, 2022. URL: https://github.com/facebookresearch/detectron2.

[57] Princeton University. Wordnet, a lexical database for english, 2022. URL: https://wordnet.princeton.edu/.

[58] Princeton University. Wordnet search - 3.1, 2022. URL: http://wordnetweb.princeton.edu/perl/webwn.

[59] Steven Walczak and Narciso Cerpa. Artificial neural networks. In Robert A. Meyers, editor, *Encyclopedia of Physical Science and Technology (Third Edition)*, pages 631–645. Academic Press, New York, third edition edition, 2003. URL: https://www.sciencedirect.com/science/article/pii/B0122274105008371, doi:https://doi.org/10.1016/B0-12-227410-5/00837-1.

[60] Cewu Lu Xinpeng Liu, Yong-lu Li. Code for "highlighting object category immunity for the generalization of human-object interaction detection", 2022. URL: https://github.com/Foruck/OC-Immunity.

[61] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. *Graph R-CNN for Scene Graph Generation. 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, pages 690–706. September 2018. doi:10.1007/978-3-030-01246-5_41.

[62] Yuan Yao, Ao Zhang, Xu Han, Mengdi Li, Cornelius Weber, Zhiyuan Liu, Stefan Wermter, and Maosong Sun. Visual distant supervision for scene graph generation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15796–15806, October 2021. doi:10.1109/ICCV48922.2021.01552.

[63] Aakarsh Yelisetty. Understanding fast r-cnn and faster r-cnn for object detection, 2020. URL: https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97.

[64] Jing Yu, Yuan Chai, Yujing Wang, Yue Hu, and Qi Wu. CogTree: Cognition tree loss for unbiased scene graph generation. In *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, pages 1274–1280, August 2021. doi:10.24963/ijcai.2021/176.

[65] Alireza Zareian. Graph bridging network (gb-net) github repository, 2020. URL: https://github.com/alirezazareian/gbnet.

[66] Alireza Zareian, Svebor Karaman, and S. Chang. Bridging knowledge graphs to generate scene graphs. In *Computer Vision - ECCV 2020*, pages 606–623, November 2020. doi:10.1007/978-3-030-58592-1_36.

[67] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5831–5840, June 2018. doi:10.1109/CVPR.2018.00611.

[68] Ji Zhang, Mohamed Elhoseiny, Scott Cohen, Walter Chang, and Ahmed Elgammal. Relationship proposal networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5226–5234, 2017. doi:10.1109/CVPR.2017.555.

[69] Ji Zhang, Kevin Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. Graphical contrastive losses for scene graph parsing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11527–11535, June 2019. doi:10.1109/CVPR.2019.01180.

[70] Guangming Zhu, Liang Zhang, Youliang Jiang, Yixuan Dang, Haoran Hou, Peiyi Shen, Mingtao Feng, Xia Zhao, Qiguang Miao, Syed Afaq Ali Shah, and Mohammed Bennamoun. Scene graph generation: A comprehensive survey. *CoRR*, abs/2201.00443, 2022. URL: https://arxiv.org/abs/2201.00443, arXiv:2201.00443.