Politecnico di Torino

Master's Degree Thesis

Exploiting background knowledge for scene graph generation with Logic Tensor Networks

Supervisors

Prof. Fabrizio LAMBERTI

Prof. Lia MORRA

Candidate

Silvia GIAMMARINARO

Master's Degree in Data Science and Engineering

December 2021

Abstract

In novel Deep Learning applications, complex models and algorithms are designed to understand the world around us. Every scene we see in real life can be represented as a set of objects and a set of predicates (actions, prepositions, etc.). Starting from these sets, a graph can be defined with objects as nodes and predicates as links. Every relationship between two objects is a triplet (subject, predicate, object). This task is called scene graph generation and it is divided into two phases: first, locate objects and predict their classes (object detection), then create the set of possible triplets (relationship detection). In the last years, this topic has gained considered attention by the research community as it is part of more challenging machine learning problems.

In this thesis, the entire scene graph generation pipeline is exploited, focusing first on object detection state-of-the-art and then scene graph generation models. As object detection is already a consolidated task, scene graph generation still has some open problems. One of them are biased annotations: as humans, we have a lot of linguistic biases, some generic words are used more often than specific ones. This bias can be seen in the Visual Relationship Detection (VRD) dataset, where the most frequent predicate is the generic place preposition 'on'. In this case, some predicates carry more information than others. As an example, instead of saying (man, on, chair), it is more accurate to say (man, sitting on, chair). This bias will negatively affect the model, as it will be tempted to use generic predicates in most triplets.

To solve this issue, the injection of knowledge into the model can be beneficial. The Logic Tensor Network (LTN) model is considered, in which the training consists of building a knowledge base. The knowledge base is created upon objects and predicates. For objects, bounding boxes coordinates, classes probabilities and geometric features are used. For predicates, a set of positive and negative logical axioms are created starting from the training set distribution. Moreover, LTN

introduces the concept of fuzzy logic, the truthiness of a logical expression is measured with a value between zero and one.

LTN has demonstrated how the use of a knowledge base can improve the performance in the scene graph generation problem. The purpose of this work is to study the LTN pipeline strengths and weaknesses, starting from the object detection task to the relationship detection. First, the impact of the object detection step on relationship detection is discussed. Wrong object classes and locations can significantly impact the final scene graph. Then, different knowledge bases are used to determine the most promising aggregation of logical axioms. The results show that both phases of the pipeline need to be optimized at their best to obtain good scene graphs. The two main datasets present in the literature are analyzed: Visual Relationship Detection (VRD) and Visual Genome (VG). Both datasets have been used by different models in the literature, so this allows us to study which improvements are more promising for LTN.

Alla mia famiglia e ai miei amici che mi hanno accompagnato in questo percorso.
Sempliecemente grazie.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introducing Scene Graph Generation

The definition of scene graph is attributed to Johnson et al. [1]. A scene graph is a data structure that describes the content of a scene. A scene graph encodes object instances, attributes of objects, and relationships between objects. The scene graph $\mathcal{G}(O, P)$ generated from every image is made of $|O|$ nodes representing the objects and $|P|$ edges representing the relationships among the objects. An example can be seen in image 1.1: a *girl* (subject) is *holding* (predicate) a *racket* (object). This results in the triplet $\langle girl, holding, racket \rangle$. All the triplets are merged to form the final scene graph. Notice that this graph is directed.

Scene Graph Generation task has been gaining interest in recent applications as it provides a better visual and semantic representation of an image. Improving such representation is crucial to obtain better results in more complex tasks. Therefore, the main applications can be divided in two categories: in the first, the final goal is to obtain a scene graph, whereas in the second, the generation of the graph is used as an in-between passage.

- **Visual relationship detection**: retrieve objects, attributes and relationships among them from the scene. First, a set of objects is defined and then they are paired and linked together using relationships. This is a Semantic Image Interpretation (SII) task. Most applications involve supervised learning

techniques. However, when dealing with noisy and biased annotation, it can happen that the detected relationship is wrong even if it fits correctly. Consider the most frequent predicate in one of the most used datasets, Visual Relationship Detection [2]: *on*. This preposition can be misused to express a lot of different scenarios. As an example, take the triplet ⟨ *cat, on, sofa*⟩, this is a very generic relationship. Is the *cat* doing something more specific as *sleeping*?

- **Semantic image retrieval**: given a query (a text or an image), the goal is to find the matching images. In this task, having a complete composition of the image is fundamental to perform the right retrieval. So scene graphs are used to have a detailed representation of the images used to compare.
- **Image/video Captioning**: given an image, the task is to describe the actions happening in the image using a sentence. In this case, scene graph generation is used as an intermediate passage to easily construct a full description.
- **Visual question answering**: given a general question about the scene, the system should provide the answer. Also in this application, it is beneficial to create a scene graph of the image to have a strong prior knowledge.

The focus of this work will be on visual relationship detection as it is part of all other applications. The improvement in such scenario is beneficial for the other ones.

The task consists of two main passages: *object detection* and *relationship detection*. Given an image I, an object detector D is used to gather the bounding boxes B from it. From the features maps (also called regions of interest, RoIs), class labels x_i are predicted. After collecting all the objects O in the image, the next passage is to link them using predicates. In the pair, a subject s and an object o are defined because the graph must be directed. So the model is looking for a predicate p to link two objects o_i and o_j, where one of them should be defined as the subject of the triplet. At the end of this passage, a triplet ⟨*subject, predicate, object*⟩ is created. This operation is repeated for all pairs in the image to generate the final scene graph.

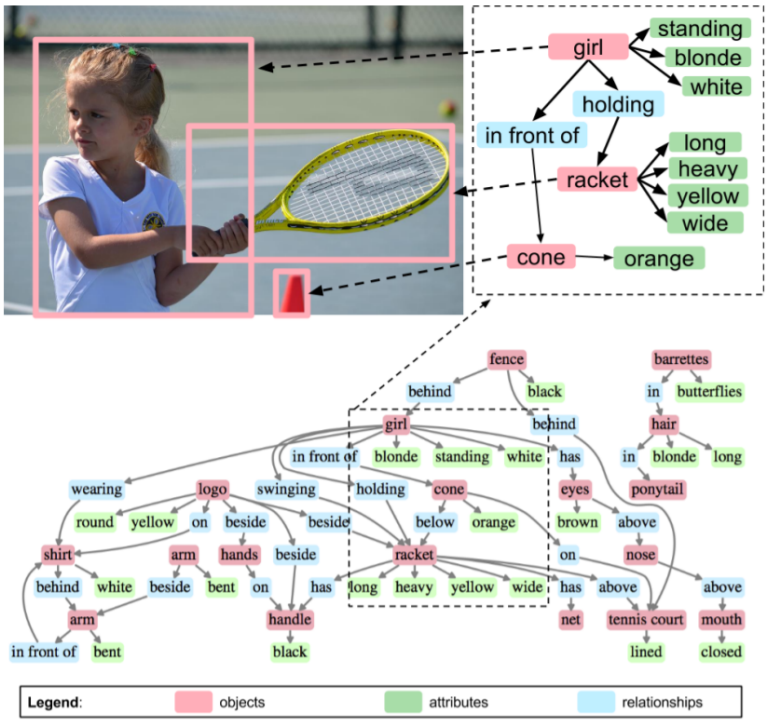


Figure 1.1: Scene graph example [1]. From the image, a scene graph is obtained. It contains subjects, objects, relationships and attributes.

This problem is challenging as it is a union of two subtasks and it still have some major problems to solve to obtain an optimal solution. The main problems related to this task are the following:

- *noisy datasets with missing annotations*: the most used dataset for this task is the Visual Genome Dataset [3]. The first issue of this dataset is related to the missing annotation. More than one predicate could be correct for a pair of objects, but the dataset reports just one possible predicate. This causes problems during training, the model can correctly spot the triplets in the image, but they are considered an error as they do not appear in the dataset.

- *long-tailed relationship distribution*: the relationship distributions of the Visual Relationship Detection dataset [2] and the Visual Genome dataset [3] are exponential distributions, the most frequent relationships have higher values in respect to the other ones. This leads the model to focus on the top frequent

classes rather than the rest. This is the reason why specific metrics have been defined for this task. These metrics will be discussed later.

- *inefficient training when looking for relationship candidates*: the last main issue is the complexity required to find the correct relationship between a given pair of objects. For N objects, the relationship search complexity is $\mathcal{O}(N^2)$. To overcome this, some models define pruning techniques to reduce this complexity and search the best relationship among a smaller list.

The structure of the thesis will be the following:

- Chapter 1 contains an introduction to the thesis.
- Chapter 2 aims to introduce the Logic Tensor Network (LTN) [4] model used for our experiments, the object detection task and the corresponding state-of-the-art models.
- Chapter 3 shows a summary of models presented in the literature to solve the scene graph generation task.
- Chapter 4 reports the methods and the materials used for the experiments.
- Chapter 5 contains the description of the experiments over the Visual Relationship dataset [2] and Visual Genome dataset [3].
- In Chapter 6, a brief discussion and possible future works are proposed.

Chapter 2

Background

2.1 An overview of Logic Tensor Network

When dealing with the Visual relationship detection task, similarities in the training set can be explored and aggregated using a specific statistical learning framework to overcome the problems discussed in Chapter 1. This model is called *Logic Tensor Network* (LTN) and it combines neural networks with logical constraints.

2.1.1 Neuro-symbolic AI

Neuro-symbolic AI is a machine learning field that tries to combine commonsense reasoning and neural networks [5]. The computations are based on symbols, which are the high-level representation of the dataset instances. In this way, the model requires a small amount of data to learn how to solve the task and it becomes interpretable.

In recent years, this approach has been discarded in favor of deep learning, where innate priors are introduced to handle complex tasks as computer vision. To understand why this is happening, we have to step back at the invention of Convolutional Neural Network (CNN) back in the 90s [6]. CNN is a specific architecture designed to produce translational invariance in a visual scene. Translational invariance improves generalization: this is why in simple image classification CNNs are used.

In this work, a Neuro-symbolic architecture, LTN, is analyzed. In this model, logic symbols are embedded into tensors. In this case, the prior is the symbolic information retrieved directly from the data.

2.1.2 Fuzzy Logic

In 1965, Lotfi Zadeh introduced fuzzy logic to describe vague prepositions [7]. It can be interpreted as an extension of the standard boolean logic. In boolean logic, the output is taken from the set $\{0, 1\}$, false or true. Instead, in fuzzy logic, the interval $[0, 1]$ is considered [8]. Doing so, different levels of truth are defined and the extreme values 1 and 0 correspond to the maximum levels of truth and falseness. Logic Tensor Network models use first-order logic expressions which define a language $\mathcal{L}$. One example could be the following:

There exists at least one Silvia such that x is blonde
and Silvia is enrolled at Politecnico di Torino.

Definition 1 *A first-order logic language contains: a set of constant symbols $\mathcal{C}$, a set of functions $\mathcal{F}$ and a set of predicate symbols $\mathcal{P}$.*

In the above expression we have:

- two constants: *Silvia*, which is the subject and *PolitecnicoDiTorino*, the object.
- an unary predicate or class *blonde*;
- a binary predicate or relation *enrolledAt*.

The example can be rewritten using logical symbols:

$$\phi : \exists! Silvia : \text{blonde}(Silvia) \wedge \text{enrolledAt}(Silvia, \text{PolitecnicoDiTorino})$$

Furthermore, we can define *domains* in which our variables are defined. In our case *Silvia* is part of the domain called *People* and *PolitecnicoDiTorino* is part of

Universities. All logic symbols can be used to define *formulas*. Table 2.1 reports the main logic symbols used in LTN formulas.

Symbol	Name	Read as
$\top$	Tautology	truth
$\bot$	Contradiction	falsum
$\rightarrow$	Implication	implies
$\neg$	Negation	not
$\wedge$	Conjunction	and
$\vee$	Disjunction	or
$\forall$	Universal Quantifier	for all
$\exists$	Existential Quantifier	there exists
$\exists!$	Uniqueness Quantifier	there exists one

Table 2.1: Logic symbols and their meaning.

The main advantage of adopting Fuzzy Logic is the use of the *grounding function* $\mathcal{G}$. This function is able to map each constant and formula into numerical features, which will be used in further computations. Each logic formula ϕ will be mapped in the interval [0,1] and each constant x will be encoded into a tensor. Such tensor contains all the features of the object. These mappings allow every data to be expressed as a logical expression. These expressions will form the so called *knowledge base*. the concept of knowledge base will be discussed in Section 2.1.3. Once the knowledge base is defined, LTN's goal is to optimize the grounding function $\mathcal{G}$.

2.1.3 Definitions and semantic

After reviewing the concept of fuzzy logic, we introduce all the theoretical concepts needed to build our LTN. As stated in Definition 1, we define the sets $\mathcal{C}$, $\mathcal{F}$, $\mathcal{P}$. Moreover, we introduce a new set $\mathcal{X}$, the variables and $\mathcal{D}$, the domain which contains terms with the same characteristics. A term t can be a constant c or a variable x. Given a function f, then $f(t)$ is still a term. However given a predicate P, $P(t)$ is an atomic formula. The term t defines the smallest and indivisible unit in a formula ϕ.

Definition 2 *A formula ϕ satisfies one of the following conditions:*

- *if t is a term and P is a predicate, the atomic formula $P(t)$ is a formula;*
- *if ϕ is formula, then $\neg\phi$ is a formula;*
- *if ϕ, ψ are formulas then $\phi \implies \psi$, $\phi \wedge \psi$, $\phi \vee \psi$ are formulas;*
- *if ψ is a formula and x is a variable, $\forall x(\psi)$ and $\exists x(\psi)$ are formulas.*

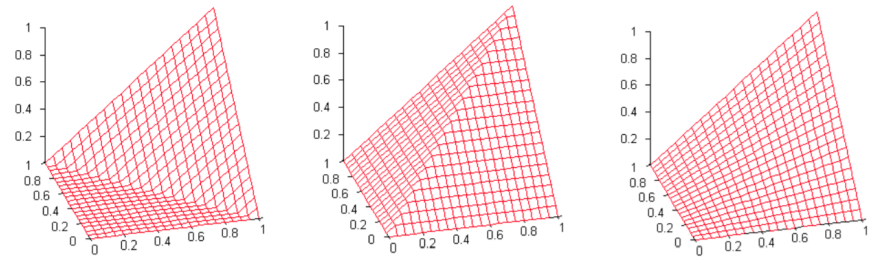
As an example, we define the following sets:

$$\mathcal{C} = \{Silvia,\ PolitecnicoDiTorino,\ Cat\}$$
$$\mathcal{X} = \{x, y\}$$
$$\mathcal{D} = \{People,\ Universities,\ Animals\}$$
$$\mathcal{P} = \{blonde,\ enrolledAt,\ Feedbing\}$$
$$\mathcal{F} = \{Hair,\ FoundedIn,\ Fur\}$$
$$\phi = \{Feeding(Silvia,\ Cat),\ \}$$

For every predicate, a set of positive and negative examples can be defined. With examples, a possible set of triplets *predicate(subject, object)* or couples *predicate(subject)*, *predicate(object)* can be defined. As an example, for the predicate *wearing*, a positive example is *wearing(woman, hat)* and a negative one is *wearing(not table, hat)*. With negative examples, non possible relationships are defined. These two sets will be called *literals*. Once all literals for all the predicates are defined, a specific t-norm and an aggregation operation are performed to obtain a *clause*. In other words, a clause is an merge of literals. In particular, for the positive literals a positive clause is obtained, the same procedure is done for the negative one. About the aggregators, some basic operations between tensors can be performed to fuse them (i.e., minimum, maximum, mean). Possible aggregators will be discussed in Section 5.

T-norm

The main difference with classical logic relies on logical connectives. Logical connectives (conjunction, disjunction, negation) are interpreted in a function from



(a) Łukasiewicz t-norm: $T(x,y) = \max(0, x+y-1)$

(b) Gödel t-norm: $T(x,y) = \min(x,y)$

(c) Product t-norm: $T(x,y) = x \cdot y$

Figure 2.1: Examples t-norm functions.

$[0,1]^2$ to $[0,1]$. This function is the so called *t-norm*.

Definition 3 *A t-norm is a function T from $[0,1]^2$ to $[0,1]$ that satisfies the following conditions:*

- ***commutative*** $T(x,y) = T(y,x)$;
- ***associative*** $T(x, T(y,z)) = T((x,y),z)$;
- ***non-decreasing*** *if* $x \leq y$ *then* $T(z,x) \leq T(z,y)$;
- ***zero and one*** $T(0,x) = 0$ *and* $T(1,x) = x$.

Examples are reported in Figure 2.1.

Grounding

After having defined fuzzy logic and all parts of the formulas, how these formulas can be interpreted by LTN? LTNs link the abstract semantics of the fuzzy logic with a concrete semantics. This interpretation is called *grounding*, which is a subset of $\mathbb{R}^n$. In this way, every term t is mapped into a n-dimensional vector. These features can be manually set: having a bounding box, the features could be the coordinates and the area. Instead, predicates are mapped into real values in the interval [0,1]. Thus, each predicated is associated with a level of trueness: the more the value is higher, the more it is true.

Definition 4 *An grounding function $\mathcal{G}$ satisfies the following conditions:*

1. *for every constant $c \in \mathcal{C}$, $\mathcal{G}(c)$ maps c in a n-dimension latent space;*
2. *for every predicate $P \in \mathcal{P}$, $\mathcal{G}(P) \in \mathbb{R}^{n \cdot \alpha(P)} \rightarrow$ [0,1]. $\mathcal{G}(P)$ is an atomic formula, where α is 1 if the predicate in unary, 2 if binary. This formula is then associated with its level of trueness in the interval [0,1];*
3. *for every function $f \in \mathcal{F}$, $\mathcal{G}(P) \in \mathbb{R}^{n \cdot \alpha(f)} \rightarrow \mathbb{R}^n$. $\mathcal{G}(f)$ is a term, so it is associated with its n features.*

The semantic of atomic formulas with terms $\{t_1, t_2, t_3, ...\}$ is defined as:

$$\mathcal{G}(f(t_1, ..., t_m)) = \mathcal{G}(f)(\mathcal{G}(t_1), ..., \mathcal{G}(t_m))$$
$$\mathcal{G}(P(t_1, ..., t_m)) = \mathcal{G}(P)(\mathcal{G}(t_1), ..., \mathcal{G}(t_m))$$

Whereas, for non-atomic formulas, the semantics is defined according to t-norm formulas:

$$\mathcal{G}(\phi \rightarrow \psi) = \min(1, 1 - \mathcal{G}(\phi) + \mathcal{G}(\psi))$$
$$\mathcal{G}(\phi \wedge \psi) = \max(0, \mathcal{G}(\phi) + \mathcal{G}(\psi) - 1)$$
$$\mathcal{G}(\phi \vee \psi) = \min(1, \mathcal{G}(\phi) + \mathcal{G}(\psi))$$
$$\mathcal{G}(\neg\phi) = 1 - \mathcal{G}(\phi)$$

Knowledge Base

In the previous sections, the grounding for constants, functions and predicates have been defined. However, the grounding has to be learnt from data. At the beginning, a partial grounding $\hat{\mathcal{G}}$ is defined applying the t-norm (Definition 3) to all formulas in $\mathcal{K}$. $\mathcal{K}$ is a set of positive and negative example from data, the goal is to obtain a pair $<\mathcal{K}, \mathcal{G}>$ called *knowledge base* or *grounded theory* to represent the data in terms of logical axioms.

Definition 5 *A knowledge base or grounded theory is defined as the pair $<\mathcal{K}, \hat{\mathcal{G}}>$. A grounding $\mathcal{G}$ satisfies the knowledge base $<\mathcal{K}, \hat{\mathcal{G}}>$ if $\hat{\mathcal{G}} \subseteq \mathcal{G}$ and $\mathcal{G}(\phi) = 1$ for all $\phi \in \mathcal{K}$.*

Thus, the grounding $\mathcal{G}$ is an extension of the starting partial grounding $\hat{\mathcal{G}}$. However, it is not always possible to obtain the grounding $\mathcal{G}$. Then, the aim is to find the best grounding possible $\mathcal{G}^*$ in the set $\mathbb{G}$ such that:

$$\mathcal{G}^* = \underset{\hat{\mathcal{G}} \subseteq \mathcal{G} \in \mathbb{G}}{argmax}\ \mathcal{G}\left(\bigwedge_{\phi \in \mathcal{K}} \phi\right)$$

This is the optimization problem to solve. The aim of the best grounding is to maximize the level of truthiness of the knowledge base. Take as an example the following formula: $\phi = \forall xy\,(\text{hat}(x) \bigwedge \text{wearing}(y, x)) \rightarrow \text{Person}(y)$. According to this, a cat wearing a hat would nullify the formula. With the best grounding problem, LTN is able to handle these cases giving a higher truthiness to the formula according to the amount of examples satisfying it. According to this case, if the triplet (person, wearing, hat) is more frequent than (cat, wearing, hat), then the first formula will be assigned to a higher level of truthiness with respect to the second one.

Moreover, given the following set of parameters to optimize $\Theta = \{M_f, N_f | f \in \mathcal{F}\} \cup \{W_p, V_p, b_p, u_p | P \in \mathcal{P}\}$, the aim is to find the best set:

$$\Theta^* = \text{argmax}_\Theta\ \mathcal{G}\left(\bigwedge_{\phi \in \mathcal{K}} \phi \mid \Theta\right) - \lambda ||\Theta||_2^2$$

where $\lambda ||\Theta||_2^2$ is a smoothing factor. The full implementation of the model will be shown in Chapter 4.

2.2 An overview of Object Detection

The first task for scene graph generation is object detection. The goal of object detection is to locate the object in the image with a bounding box and to classify the object. The main approaches during the last years are analyzed.

There are two types of object detectors: one-stage and two-stage. On the one hand, we have one-stage object detectors (i.e., YOLO) where the problem is treated

as a regression problem. Taking an image, class probabilities and bounding box locations are defined. On the other hand, we have two-stage object detectors (i.e. R-CNN [9]) where region proposals are generated and then they are sent to two different branches: bounding box regression and object classification. The main difference between the two is that two-stage models reach high accuracy but they are more slow. In this section, only two-stage object detectors will be analyzed as they are the ones used for the scene graph generation task.

2.2.1 Two-stage detectors

Object Detection is based on the concept of Region-based CNN. The first model proposed in literature is the so-called **R-CNN** [9]. In R-CNN, 2000 possible RoIs (regions of interest) are extracted from the original image. Then on each region, a CNN is applied as a feature extractor. The features obtained are then passed to a Support Vector Machine (SVM) model for the classification task and the regression task to tighten the bounding boxes. The passages are shown in Figure 2.2. This model is extremely slow due to the high computational cost: to each region a CNN is applied followed by the SVM classification task.

To overcome the issues in R-CNN, the model **Fast R-CNN** has been proposed [10] (Figure 2.3). The architecture is very similar to the R-CNN model. At first, the image is passed to a CNN to extract a feature map. Then a RoI pooling layer and two fully connected layers are used to obtain a RoI feature vector. Then the bounding box and the object class are obtained using two different fully connected layers.

However, both R-CNN and Fast-RCNN use selective search [12] to generate

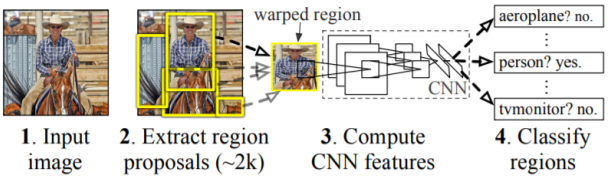


Figure 2.2: R-CNN architecture.

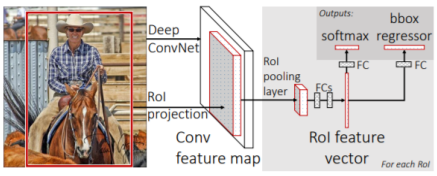


Figure 2.3: Fast R-CNN architecture.

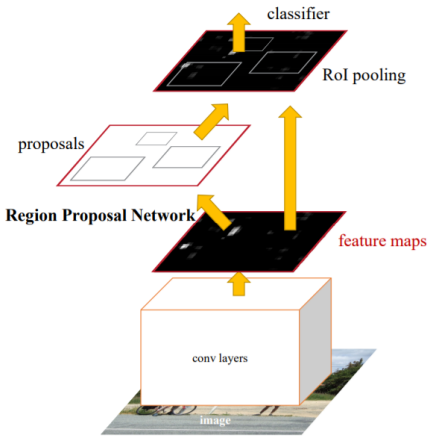


Figure 2.4: Faster-CNN architecture.

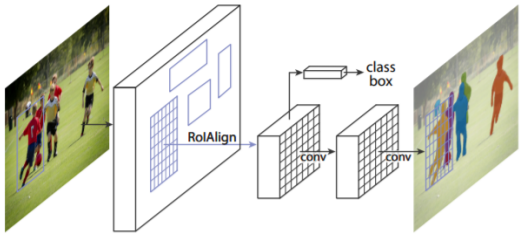


Figure 2.5: Mask R-CNN architecture.

the regions. This method is very time-consuming. The algorithm looks for similar pixels to create the regions. The similarity measure is computed based on four features: color, texture, size, and overlap. At each iteration, similar regions are merged.

Faster R-CNN solves this problem using a new method called Region Proposal Network (RPN) that learns the regions itself. The initial feature map is passed to the network which produces as output the region proposals (Figure 2.4). These regions are then reshaped using a RoI pooling layer. Lastly, the image inside the region is classified and bounding box offsets are defined.

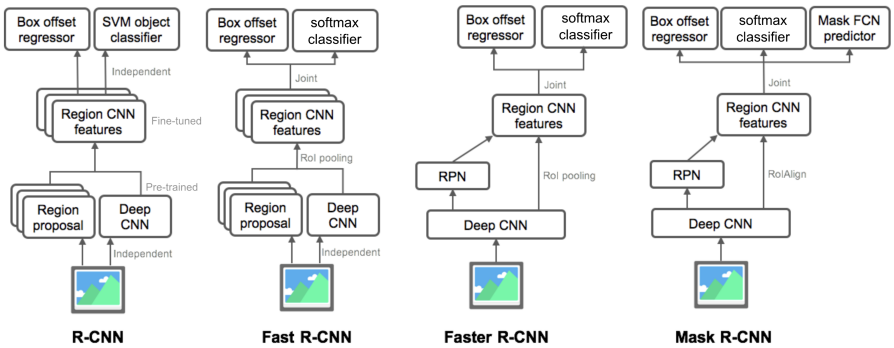


Figure 2.6: Object Detectors comparison. Scheme providing a quick comparison between R-CNN, Fast R-CNN, Faster R-CNN and Mask R-CNN [11].

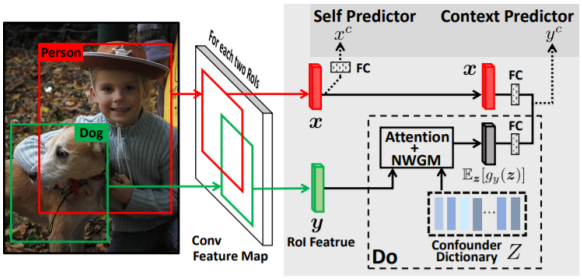


Figure 2.7: Visual Commonsense R-CNN architecture. The model is divided into two main branch: self predictor for the subject and context predictor for the context objects.

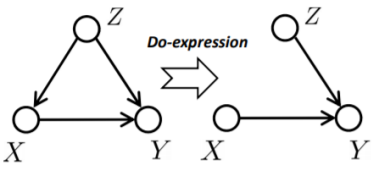


Figure 2.8: Do-expression example. Graphical representation of the difference between $P(Y|X)$ and $P(Y|do(x))$.

A further improvement of the Faster R-CNN is proposed by He et al [13] with the **Mask R-CNN** model. Starting from the Faster R-CNN architecture, a further element is added: pixel-level segmentation. Thus, a mask indicating the object is provided as we can see in Figure 2.5.

Lastly, Wu et al. [14] released **Detectron** which is a complete collection of the state-of-the-art models for object detection. A quick overview of the methods discussed is reported in figure 2.6.

Since object detection is one of the most used task in Computer Vision, a novel approach involving more high-level tasks (i.e. Visual Question Answering) was proposed by Wang et al [15]. The model proposed is called **Visual Commonsense R-CNN**. The novelty introduced is using the causal intervention $P(Y|do(x))$ instead of using the traditional likelihood $P(Y|X)$. The $do(\cdot)$ operation will be further explained in Section 3.1.1. In this way, the network is able to develop a strong commonsense instead of learning common cooccurrences (ex: $P(z = sink|X = toilet)$). The difference can be seen applying the Bayes theorem to both expressions: in the traditional case we obtain:

$$P(Y|X) = \sum_{z} P(Y|X, z)P(z|X) \tag{2.1}$$

Instead, with the causal intervention, the observational bias $P(z|X)$ is removed.

Thus:

$$P(Y|do(X)) = \sum_z P(Y|X,z)P(z) \tag{2.2}$$

The architecture is reported in Figure 2.7. Using a CNN, a feature map is obtained. The following features are extracted: RoI features x and its contextual RoI y. In the example, the subject is *person* and the context object is *dog*. In this model, the Region Proposal Network (RPN) is discarded. Two predictors are present: a Self predictor for the class label and the context predictor to apply the *do* operation. Thus, the loss function contains two terms: one for the self predictor branch $\mathcal{L}_{self}$ and one for the context predictor branch $\mathcal{L}_{ext}$.

$$\mathcal{L}(X) = \mathcal{L}_{self}(p, x_c) + \frac{1}{N}\sum_i \mathcal{L}_{ext}(p_i, y_i^c) \tag{2.3}$$

where $\mathcal{L}_{self}$ is the negative log-likelihood defined for the subject class label and $\mathcal{L}_{ext}$ is the negative log-likelihood for the K context object class labels.

2.2.2 Evaluation metrics

Intersection over Union (IOU)

Given the ground truth bounding box A and the predicted bounding box B, the intersection over union computes the overlap between them. An example is provided in Figure 2.9. It is defined as the total number of pixels in common between the two bounding boxes over the total number of pixels present in both of them. Thus, it is defined as:

$$IOU = \frac{A \cap B}{A \cup B}$$

$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$

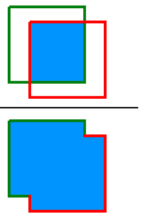


Figure 2.9: IOU computation example [16].

Once the predicted bounding boxes are obtained, it could happen that some of them are detecting the same object and thus they are overlapping. An example is shown in Figure 2.10. To filter out these duplicates, the *non-maximum suppression* technique is applied [17].

Figure 2.10: Non-maximum suppression example. In this image, the objects *bottle* and *table* have been detected twice, the predicted bounding boxes are overlapping. The object detetctor used is R-CNN [9].

Precision and recall

After the predicted bounding boxes are defined, the model has to assign a class label to them. This is a simple classification task and different cases can be identified. Given the truth class y_{true}, the ground truth bounding box $\mathcal{B}_{true}$, the predicted class y_{pred}, the predicted bounding box $\mathcal{B}_{pred}$ and a threshold $thres$, the possible scenarios are:

- *True positive (TP)*: y_{pred} is equal to y_{true} (correct prediction) and $\mathcal{B}_{true}$ and $\mathcal{B}_{pred}$ with $IOU \geq thres$;
- *True Negative (TN)*: it is not used in the object detection task. These are all the possible bounding boxes that were correctly not detected.

- *False Positive (FP)*: wrong detection, $\mathcal{B}_{true}$ and $\mathcal{B}_{pred}$ with $IOU < thres$;
- *False Negative (FN)*: a ground truth object has not been detected.

A graphical representation of such cases is reported in Figure 2.11.

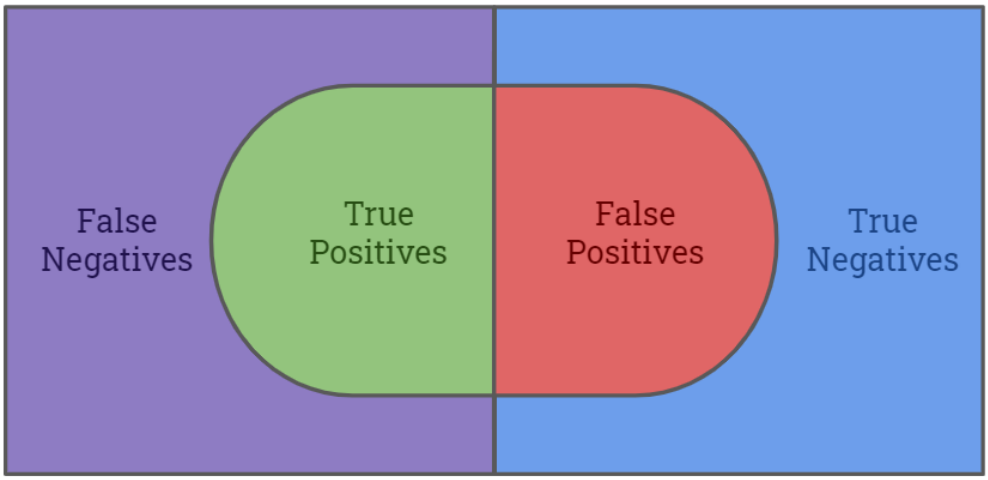


Figure 2.11: Precision and recall. Sets needed to compute precision and recall.

Thus, it is possible to define *precision* which represent the percentage of positives classified correctly:

$$\frac{TP}{TP + FP} \tag{2.4}$$

Then, *recall* is defined as the percentage of positives between the same class:

$$\frac{TP}{TP + FN} \tag{2.5}$$

Average Precision

Precision and recall are evaluated together in the so called *precision-recall* curve. The recall values are reported in the x-axis and the precision in the y-axis. The area underneath is called *average precision* and it varies in the range between 0 and 1. Thus, it is equal to:

$$AP = \int_0^1 p(r)dr \tag{2.6}$$

The mean of the average precision values forms the *mean average precision*.

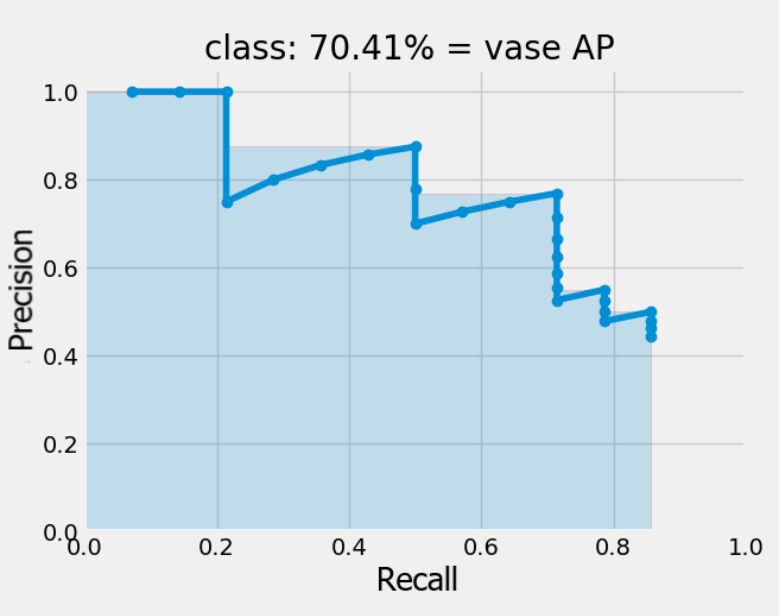


Figure 2.12: Precision-recall curve. The area represents the average precision.

To simplify the integral computation, a new version involving interpolation is used. In 2008, the Pascal VOC2008 Challenge proposes a 11-point interpolation. It is computed as the mean at 11 levels of recall [0, 0.1, 0.2, ..., 1] taking the maximum precision whose recall is greater than r:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{interp}(r) \tag{2.7}$$

where

$$\rho_{interp}(r) = \max_{\widetilde{r} \geq r} \rho(\widetilde{r}) \tag{2.8}$$

From 2010, the interpolation performed by PASCAL VOC challenge uses all n data points for the interpolation instead of the previous 11. Thus, Equation 2.7 can be rewritten as:

$$AP = \sum_{n=0} (r_{n+1} - r_n) \rho_{interp}(r_{n+1}) \tag{2.9}$$

with

$$\rho_{interp}(r_{n+1}) = \max_{\widetilde{r} \geq r_{n+1}} \rho(\widetilde{r}) \tag{2.10}$$

where $\rho(\widetilde{r})$ is the precision measured at $\widetilde{r}$.

Chapter 3

Related Work

3.1 State-of-the-art

In this section, the most recent and relevant models in the literature are analyzed. Every model solves a specific problem related to scene graph generation, so the models have been grouped into different categories:

- *statistical inference methods*: based on the dataset distribution, most common relationships can be retrieved given two objects and removing bias can improve the performances;
- *knowledge as additional resource*, the performances can be improved using external knowledge bases;
- *new innovative losses*: ad-hoc losses have been developed to deal better with this task;
- *efficient graph generation* to improve the final scene graph generation;
- *attention is all you need*: the transformer architecture has completely changed the NLP state-of-art and some applications are rising in computer vision.

Furthermore, the main datasets and metrics used will be discussed.

3.1.1 Statistical Inference methods

Detailed analysis of the most used datasets in the field, Visual Genome [3], has shown two important facts. The first is that the correct relationships between objects can be easily retrieved looking at the relationship distribution in the training set. Thus, the relationship is highly correlated with the objects. The second point is the presence of bias in the dataset. The relationship distribution presents a long-tailed distribution, so the model learns better the most frequent predicates discarding the rarest. This unbalances learning process can impact negatively a few-shot or a zero-shot learning setting.

The first approach to deal with these issues was proposed by Zellers et al. [18] in 2017, the model is the so called **MotifNet**. The main idea behind this approach is that given a pair of object labels $[o_i, o_j]$, this pair is highly predictive of the relationship between them, but not vice-versa. Therefore, to define the graph generation given the image, we can divide the process in three passages:

- generate the bounding box B given the image I ($P(B, I)$);
- predict the class of the object inside the bounding box ($P(O|B, I)$);
- identify the relationship between two objects given their classes ($P(R|B, O, I)$).

Thus, the equation can be defined as follows:

$$P(G|I) = P(B, I)P(O|B, I)P(R|B, O, I) \tag{3.1}$$

These three tasks are divided among three models as we can see in Figure 3.1. The first model is a Faster R-CNN with VGG backbone [19], used as an object detector. Once the bounding boxes are obtained, they are passed to a bidirection LSTM to construct a contextualized representation for the object prediction. The object context C is computed as:

$$\mathbf{C} = \text{biLSTM}([f_i; \mathbf{W}_1\mathbf{l}_i]_{i=1,\ldots,n}) \tag{3.2}$$

where C contains the final LSTM layer's hidden states for every object b in the image and W_1 is the embedded matrix with size d=100 obtained from the

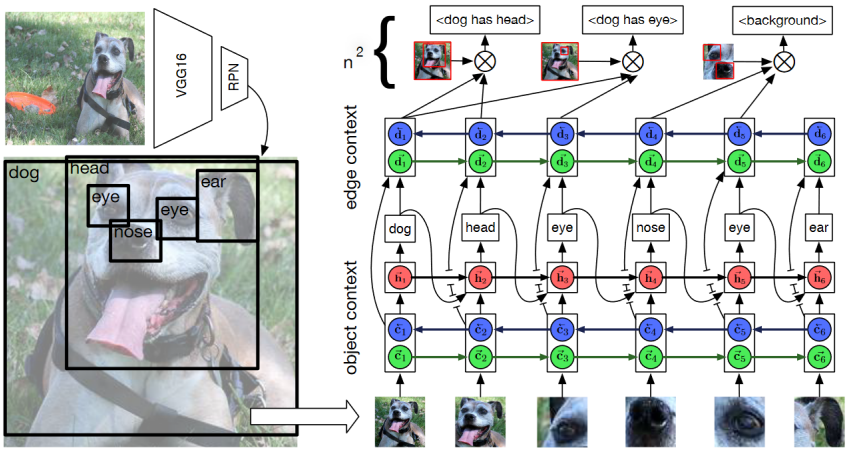


Figure 3.1: MotifNet architecture. The architectures is divided in three models: object detector to detect the boundid boxes, one biLSTM to retrieve the object context and one biLSTM to retrieve the edge context.

distribution of the predicted classes l_1. Then the context is used to decode the labels according to the previous labels:

$$\mathbf{h}_i = LSTM_i([\mathbf{c}_i; \hat{o}_{i-1}]) \tag{3.3}$$

$$\mathbf{o}_i = \text{argmax}(\mathbf{W}_o \mathbf{h}_i) \in \mathbb{R}^{|\mathcal{C}|} \tag{3.4}$$

where $\hat{o}_i$ are the object class commitments.

Then, another bidirectional LSTM is used to extract the edge context D:

$$\mathbf{D} = \text{biLSTM}([c_i; \mathbf{W}_2 \hat{o}_i]_{i=1,\ldots,n}) \tag{3.5}$$

where W_2 is the embedded matrix with size d=100 obtained mapping $\hat{o}_i$.

Lastly, the probability of each edge having label $x_{i \to j}$:

$$\mathbf{g}_{i,j} = (\mathbf{W}_h \mathbf{d}_i)(\mathbf{W}_t \mathbf{d}_j)\mathbf{f}_{i,j} \tag{3.6}$$

$$P(x_{i \to j} | B, O) = \text{softmax}(\mathbf{W}_r \mathbf{g}_{i,j} + \mathbf{w}_{o_i, o_j}) \tag{3.7}$$

where $\mathbf{W}_h$ and $\mathbf{W}_t$ project the head and the tail context into $\mathbb{R}^{4096}$ and $\mathbf{w}_{o_i, o_j}$ is a bias vector specific to the head and tail labels.

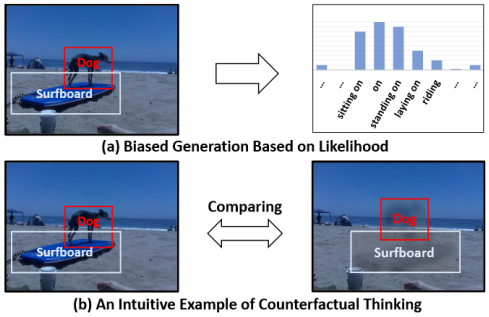


Figure 3.2: Biased vs unbiased approach. Visual representation of how the unbiased approach works using Total Direct Effect.

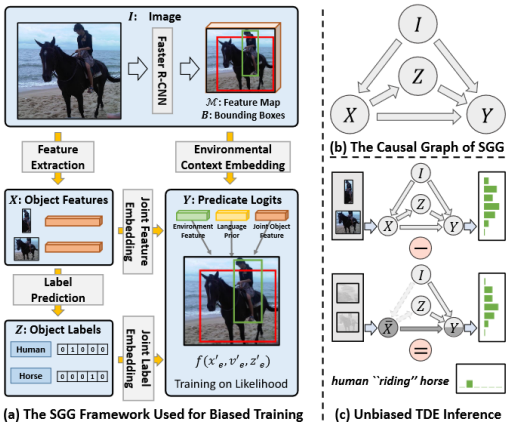


Figure 3.3: TDE architecture. The novel part in respect to previous state-of-the-art model lies on part (c), where unbiased TDE inference is applied. This inference is based on the previously obtained causal graph (b) with the framework shown in (a).

To overcome the problem of biased predicates, Tang et al. proposed a novel unbiased approach called **Total Direct effect (TDE)** [20]. The intuition behind the model is given by the definition of counterfactual causality: *"would the prediction be the same, if I had not seen this context before?"*. To understand this statement, we can refer to Figure 3.2. The model first retrieves biased predicate predications based on 3-relatedwork model (like MotifNet), then the objects in the image are made "invisible" to obtain a new unbiased prediction of the predicate. The reasoning behind this action lies in the counterfactual bias. A graph showing the workflow is shown in Figure 3.3.

In the biased case, the causal graph contains the following nodes: I as image and object detector, X as object features, Z as object class, and Y as final predicate logits. As the object detector, a Faster R-CNN with a ResNetXt101-FN backbone [19] is used.

The loss used for the object label and predicate label prediction is the cross-entropy loss. The y_e logits are predicted by each branch to have a non-dominant

branch.

In the counterfactual setting, an intervention operation is applied. This operation is denoted as $do(\cdot)$. The link between node I and node X is removed and a new value $\bar{x}$ is assigned to node X. The new value can be set as the mean feature of the training set or the zero vector. In this way, the output logits Y becomes:

$$Y_{\bar{x}}(u) = Y(do(X = \bar{x}|u) \tag{3.8}$$

where u is the image.

Thus, the Total Direct effect can be written as

$$TDE = Y_x(u) - Y_{\bar{x}}(u) \tag{3.9}$$

where $Y_x(u)$ is the predicate logits vector obtained from the biased training. With this method, the model is able to balance the biased prediction. In the case of Figure 3.2, the obtained biased prediction is `on` but the dog's straights legs will cause more effect on the predicate `standing on`.

3.1.2 Knowledge as additional resource

Another approach to deal with long-tail distribution and noisy data is to have a knowledge base. A knowledge base can be created gathering sentences from Wikipedia [21], using semantic knowledge graphs such as ConceptNet [22] [23] or manually [4].

A knowledge base is an additional data source used to solve the task we are dealing with. The issue related to it is how to incorporate such information into the model and where to gather it.

Yu et al. [21] propose one of the first models to incorporate knowledge into the scene graph generation task. The 2014 Wikipedia dump is used to extract common triplets. At first the sentences are parsed into text, then scene graphs are generated from such textual descriptions [24]. In this way triplets in the form $\langle$ *subject, predicate, object* $\rangle$ are obtained. The extra knowledge obtained is used to compute statistics about the predicates used for a given pair of object pairs. This additional data is useful to deal with unseen object pairs, but it can be very

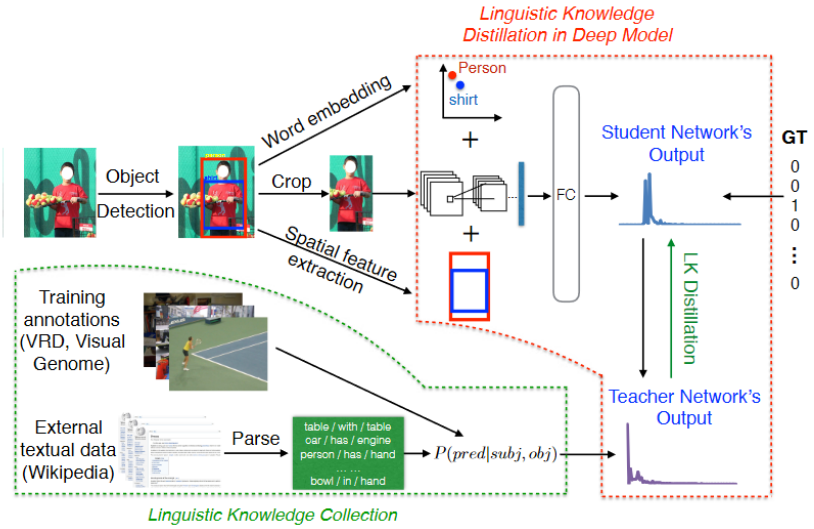


Figure 3.4: Linguistic knowledge Distillation architecture. Student and teacher networks representation.

noisy. This work aims to demonstrate that it is more important to have external knowledge than extend the training data.

The model is composed by two parts: the teacher and the student network. The teacher network aggregates the annotations from two datasets (Visual Relationship Detection [2] and Visual Genome [3]) and the external knowledge obtained from Wikipedia. The statistic extracted is the following: $P(predicate|subject, object)$. Instead, the student network receives three inputs: the cropped image from the bounding boxed union of subject and object, the semantic objects representation, and the spatial features of the bounding boxes. The architecture can be seen in Figure 3.4. The optimization problem for the teacher network is defined as:

$$\min_{t \in T} KL(t(Y)||s_\phi(Y|X)) - C\mathbb{E}_t[L(X|Y)] \tag{3.10}$$

where $t(Y)$ and $s_\phi(Y|X)$ are the outputs of the teacher and the student network and the second term is used as a log barrier to penalize solutions that do not satisfy the constraint $L(X, Y) = logP(pred|sub, obj)$. The KL operation measures the KL-divergence between the student's and the teacher's predictions. The teacher's

predictions can be rewritten as:

$$t(Y) \propto s(Y|X)exp(CL(X,Y)) \tag{3.11}$$

The final objective can be defined as:

$$\min_{\phi \in \Phi} \frac{1}{n} \sum_{i=1}^{n} \alpha l(s_i, y_i) + (1-\alpha) l(s_i, t_i) \tag{3.12}$$

where i is the i-th sample, s_i and t_i are the student's and teacher's predicitions for that sample, y_i is the ground truth label, l is the loss function and α is a balancing term.

Gu et al. [23] propose another called KB-GAN. The knowledge base used a knowledge graph called ConceptNet [22]. The model architecture and the use of ConceptNet are reported in Figure 3.5 and 3.6. Once the object label a_i is obtained from the refined object vector $\bar{o}_i$, the commonsense relationships are retrieved from the knowledge base KB:

$$a_i \rightarrow \langle a_i, a^r_{i,j}, a^o_j, \omega_{i,j} \rangle, j \in [0, K-1] \tag{3.13}$$

where $a^r_{i,j}$ and a^o_j are the possible relationship and objects to form the triplet $\langle a_i, a^r_{i,j}, a^o_j \rangle$. The values of the weights $\omega_{i,j}$ are obtained from the ConceptNet, indicating how common the triplet is. All triplets are then transformed into a sequence of words to map them into a continuous vector space. The embedded vectors are then passed to a RNN-based encoder with bidirectional GRU cells to retrieve the facts F. The most relevant facts are extracted using an episodic memory module.

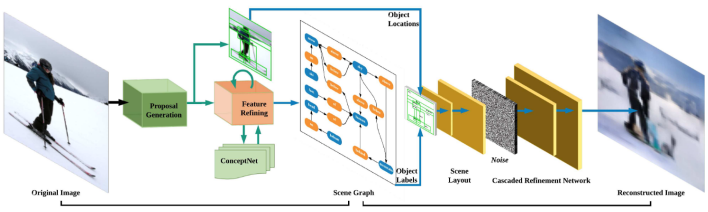


Figure 3.5: KB-GAN architecture [23].

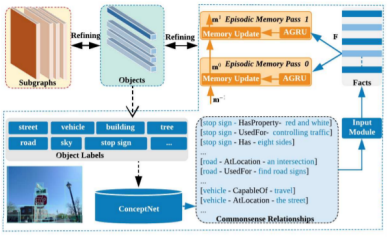


Figure 3.6: ConceptNet usage [23].

Donadello et al. [4] propose a new model called Logic Tensor Network (LTN) to combine prior knowledge and statistical relational learning. This model will be the focus of this work.

3.1.3 New innovative losses

In a complex task such as Scene Graph Generation, it is crucial to define effective loss functions.

In [23] from the bounding boxes locations, and object labels obtained, a new image is created. It is used to compare it with the original one using a Generative Adversarial network architecture. From the object i, the object embedding vector o_i is generated. Then the object embedding vector is wrapped in the object layout o_i^{layout}. To generate the scene layout, all object layouts are summed: $S^{layout} = \sum_i o_i^{layout}$. Given the scene layout, we can generate the image thanks to the generator G. Given the conditional GAN architecture in Figure 3.7, the losses involved are the following:

$$\mathcal{L}_{pixel} = ||\mathbf{I} - \hat{\mathbf{I}}|| \tag{3.14}$$

$$\mathcal{L}_{D_i} = \mathbb{E}_{I \sim p_real}[\log D_i(\mathbf{I})] \tag{3.15}$$

$$\mathcal{L}_{G_i} = \mathbb{E}_{I \sim p_G}[\log(1 - D_i(\hat{\mathbf{I}}))] + \lambda_p \mathcal{L}_{pixel} \tag{3.16}$$

where λ_p is a tuning parameter and $\hat{I} = G_i(z|S^{layout})$.

Zhang et al. [25] introduce three types of losses. These losses address two types of issues. The first one is **entity instance confusion** happens when multiple instances of the same are confused by the model. For example, multiple instances of the object *glass* are present in the picture and the wrong is picked to form the triplet ⟨man, holds, glass⟩.

Then the second issue is **proximal relationship ambiguity**, when multiple triplets present the same predicate and the model fails to infer the the right pair subject-object. For example, when multiple musicians are in the same picture and they are coupled with the wrong instruments. Both examples related to these problems can be seen in Figure 3.8. Each contrastive losses are defined based on the

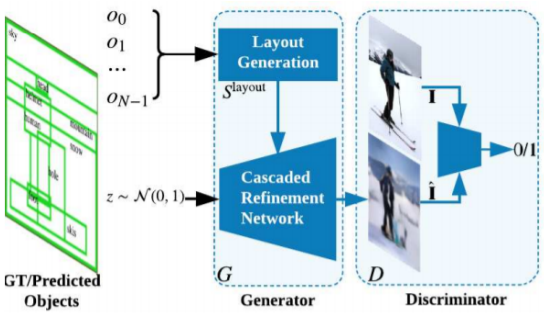
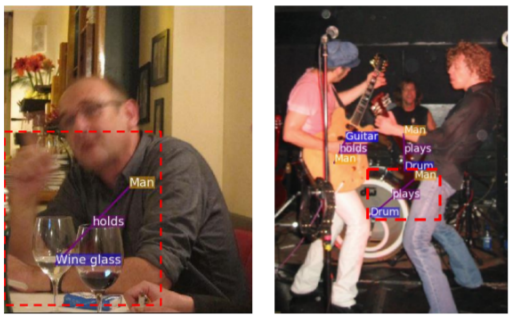


Figure 3.7: Discriminator and generator architecture used in [23]. From the bounding boxes a new image is generated.



(a) Entity Instance Confusion (b) Proximal Rel Ambiguity

Figure 3.8: Entity Instance Confusion and Proximal Relationship Ambiguity. Problems addressed in [25].

affinity term $\Phi(s, o)$. This term indicates the probability of having a relationship between the subject and the object:

$$\Phi(s, o) = 1 - p(pred = \varnothing | s, o) \tag{3.17}$$

where $\varnothing$ represents `no_relationship`.

The losses proposed are the following:

- **1) Class Agnostic** used for contrasting positive/negative pairs regardless of their relation and adds contrastive supervision for generic cases. For a subject s_i and an object o_j, the expressions to maximize are defined as follows:

$$m_1^s(i) = \underset{j \in V_i^+}{min} \Phi(s_i, o_j^+) - \underset{k \in V_i^-}{max} \Phi(s_i, o_k^-) \tag{3.18}$$

$$m_1^s(o) = \underset{j \in V_i^+}{min} \Phi(s_i^+, o_j) - \underset{k \in V_i^-}{max} \Phi(s_i^+, o_k) \tag{3.19}$$

where V_i^+ and V_j^+ represent the objects related or not to subject s_i. In equation 3.24 the aim is to minimize objects related to the subject s_i and to maximize objects not related. The loss is defined as follows:

$$\mathcal{L}_1 = \frac{1}{N} \sum_{i=1}^{N} max(0, \alpha_1 - m_1^s(i)) + \frac{1}{N} \sum_{i=1}^{N} max(0, \alpha_1 - m_1^o(j)) \tag{3.20}$$

where N is the total number of entities and α a tuning parameter. This loss is used among all instances.

- **2) Entity Class Aware:** to deal with entity instance confusion. As before, the margins are defined as:

$$m_2^s(i,c) = \underset{j \in \mathcal{V}_i^{c+}}{min}\, \Phi(s_i, o_j^+) - \underset{k \in \mathcal{V}_i^{c-}}{max}\, \Phi(s_i, o_k^-) \tag{3.21}$$

$$m_2^s(j,c) = \underset{j \in \mathcal{V}_i^{c+}}{min}\, \Phi(s_i^+, o_j) - \underset{k \in \mathcal{V}_i^{c-}}{max}\, \Phi(s_k^+, o_j) \tag{3.22}$$

where $\mathcal{V}_i^{c+}$, $\mathcal{V}_i^{c-}$, $\mathcal{V}_j^{c+}$, $\mathcal{V}_j^{c+-}$ are subsets related to the class $\mathtt{c}$. The loss can be written as:

$$\begin{aligned}\mathcal{L}_2 = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{C}\mathcal{V}_i^+|}\sum_{c \in \mathcal{C}(\mathcal{V}_i^+)} max(0, \alpha_2 - m_2^s(i,c)) \\ + \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{C}\mathcal{V}_j^+|}\sum_{c \in \mathcal{C}(\mathcal{V}_j^+)} max(0, \alpha_2 - m_2^o(j,c))\end{aligned} \tag{3.23}$$

where $\mathcal{C}()$ returns the set of unique classes of sets $\mathcal{V}_i^+$ $\mathcal{V}_j^+$. Compared to the class agnostic loss, this loss maximixes the margins between the instances of the class.

- **3) Predicate Class Aware:** for proximal relationship ambiguity. It maximizes the margins within groups of instances with the same predicate.

$$m_2^s(i,e) = \underset{j \in \mathcal{V}_i^{e+}}{min}\, \Phi(s_i, o_j^+) - \underset{k \in \mathcal{V}_i^{e-}}{max}\, \Phi(s_i, o_k^-) \tag{3.24}$$

$$m_2^s(j,e) = \underset{j \in \mathcal{V}_j^{e+}}{min}\, \Phi(s_i^+, o_j) - \underset{k \in \mathcal{V}_j^{e-}}{max}\, \Phi(s_k^+, o_j) \tag{3.25}$$

where $\mathcal{V}_i^{e+}$ and $\mathcal{V}_j^{e+}$ are the sets of pairs where e is the ground truth predicate between s_i and o_j. Instead $\mathcal{V}_i^{e-}$ and $\mathcal{V}_j^{e-}$ are the sets where the predicate is

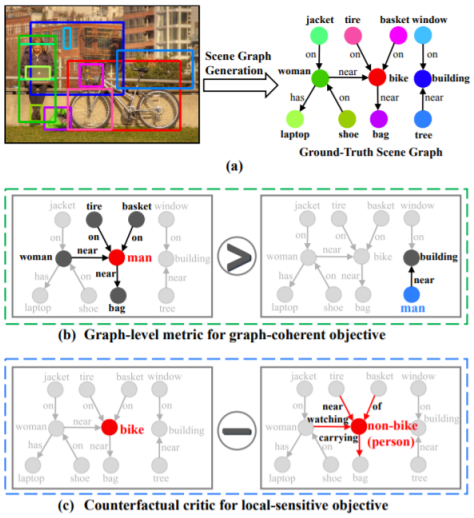


Figure 3.9: Graph Coherency and Local Sensitivity. Problems analyzed in CMAT [26].

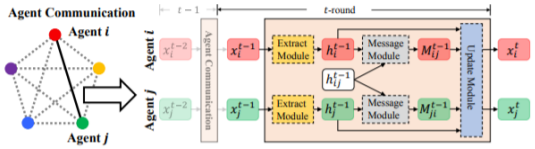


Figure 3.10: Message passing between agents in CMAT [26].

wrongly classified as e. The loss can be written as:

$$\mathcal{L}_3 = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{EV}_i^+|}\sum_{c\in\mathcal{C}(\mathcal{V}_i^+)} max(0, \alpha_3 - m_3^s(i,e)) + \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{EV}_j^+|}\sum_{c\in\mathcal{C}(\mathcal{V}_j^+)} max(0, \alpha_3 - m_3^o(j,e)) \tag{3.26}$$

The final loss can be written as:

$$\mathcal{L} = \mathcal{L}_0 + \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_3 \tag{3.27}$$

where L_0 is the cross-entropy loss computed for predicate classes.

Lastly, Chen et al. [26] propose a reinforcement learning approach to solve the scene graph generation task. The problems addressed in this work are two: **Graph Coherency** and **Local Sensitivity**. For Graph Coherency, objects and relationships should be consistent: a misclassified node with an higher degree should be more penalized than a node with a lower degree. For degree we intend the sum of the incoming and outcoming edges of the node. For Local Sensitivity, counterfactual replacement is used. The node class `bike` is substitute with a `non-bike` instance

to see how the reward changes according to these replacement. These examples are showed in Figure 3.9. Starting from these problems, a novel model called **Counterfactual critic Multi-Agent Training (CMAT)** is proposed. The reinforcement learning expression to define the loss is called Multi-Agent Policy Gradient. The critic is the relationship model and the object classification model serves as a policy network.

In this setting, an action, a policy, and a state are defined. The action space is the set of all possible object classes: the action of agent i is expressed as v_i^t. The state is the hidden state h_i^t of an LSTM cell. It is used to encode the history of each state. The message passing in the LSTM can be seen in Figure 3.10. The policy is the object classifier p_i^T.

Thus, the stochastic gradient to compute is the following:

$$\nabla_\theta J \sum_{i=1}^{n} \nabla_\theta \log \mathbf{p}_i^T(v_i^t|h_i^T;\theta) R(H^T, V^T) \tag{3.28}$$

where $R(H^T, V^T)$ is the real graph-level reward based on the Recall@K metric. The metrics will be discussed in section 3.3. The intuition behind this method is to change the reward function, incorporating the counterfactual baseline. The counterfactual baseline is defined as $CB^i(H^T, V^T) = \sum p_i^T(\widetilde{v}_i^T) R(H^T, (V_{-i}^T, \widetilde{v}_i^T)$ where $\widetilde{v}_i^T$ is new action of agent i and V_{-i}^T are the actions performed by all others agents.

The new reward is called *Advantage* and it is defined as follows:

$$A^i(H^T, V^T) = R(H^T, V^T) - CB^i(H^T, V^T) \tag{3.29}$$

Then the total loss is defined as

$$\begin{aligned} \nabla_\theta J \sum_{i=1}^{n} \nabla_\theta \log \mathbf{p}_i^T(v_i^t|h_i^T;\theta) A^i(H^T, V^T) \\ + \alpha \sum_{i=1}^{n} \sum_{j=1}^{n} \nabla_\theta \log \mathbf{p}_{ij}(r_{ij}) \\ + \alpha \sum_{i=1}^{n} \sum_{j=1}^{n} \nabla_\theta \log \mathbf{p}_i^T(v_i^T) \end{aligned} \tag{3.30}$$

where the second and the third term are cross-entropy (XE) supervised losses for relationships and objects.

3.1.4 Efficient Graph Generation

Yang et al. [27] propose a model called **Graph R-CNN**. This model intelligently prunes the scene graph connection (not random as done in previous work). The process can be defined as follows:

$$P(S|I) = \overbrace{P(V,I)}^{\substack{\text{Object Region}\\ \text{Proposal}}} \underbrace{P(E|V,I)}_{\substack{\text{Relationship}\\ \text{Proposal}}} \overbrace{P(R,O|V,E,I)}^{\text{Graph Labelling}} \tag{3.31}$$

where the object region proposal is a Faster R-CNN [19], and graph labelling is a refinement process to create the graph. The novelty lies in RePN model, which allows the process to be end-to-end. The framework is shown in Figure 3.11.

The relationship proposal network RePN estimate the relatedness of object pairs. Given the object classification distribution P^0, the relatedness between subject i and object j is defined as follows:

$$f(\mathbf{p}_i^o, \mathbf{p}_j^o) = <\Phi(\mathbf{p}_i^o, \Psi(\mathbf{p}_j^o)>, i \neq j \tag{3.32}$$

where Φ and Ψ are projection functions. This relatedness score is computed among all possible pairs. After computing the score matrix, the top K pairs are chosen following a descending order. Then non-maximum suppression (NMS) is applied to filter out the objects that have overlap with others. The overlap between two pairs u, v and p, q is computed as follows:

$$IoU(u,v,p,q) = \frac{I(r_u^o, r_p^o) + I(r_v^o, r_q^o)}{U(r_u^o, r_p^o) + U(r_v^o, r_q^o)} \tag{3.33}$$

where I compute the intersection area between the boxes and U the union.

Once the sparse graph is obtained from RePN, an Attentional GCN (aGCN) is used to generate the final scene graph. The idea behind aCGN is to extend a vanilla GCN [28] adding an attention module for learning how to adjust the weights α. The features are refined in the following way:

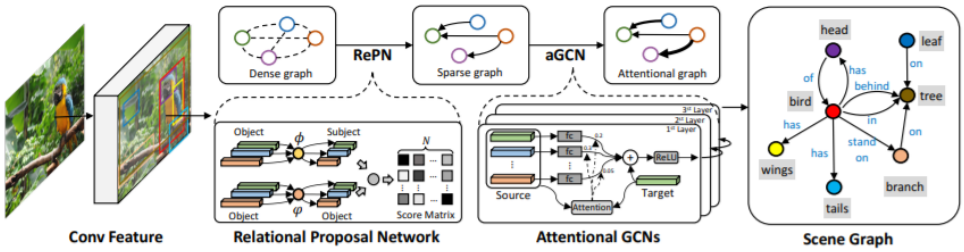


Figure 3.11: Graph R-CNN framework. The architecture is divided in three parts: obejct detector, Relation Proposal Netwrok RePN and Attentional GCN aGCN.

$$z_i^o = \sigma(\overbrace{W^{skip} Z^o \alpha^{skip}}^{\text{Message from others objects}} + \overbrace{W^{sr} Z^r \alpha^{sr} + W^{or} Z^r \alpha^{or}}^{\text{Message from Neighboring Relationships}}) \tag{3.34}$$

$$z_i^r = \sigma(z_i^r + \underbrace{W^{rs} Z^o \alpha^{rs} + W^{ro} Z^o \alpha^{ro}}_{\text{Messages from Neighboring Objects}}) \tag{3.35}$$

where s, o and r are the subjects, objects, and relationships. The linear transformation from a node of type a to a node of type b is written as W^{ab}. Then, the object and relationship features are expressed as Z^o and Z^r.

3.1.5 Attention is all you need

In 2017, Vaswani et al. [29] proposed a new model to overcome recurrent and convolutional neural networks. The model is the so called transformer and since then, it has changed the NLP 3-relatedwork. The main drawback of RNNs is the large number of path links and computations needed to go all over the sequence. Like in figure 3.1, a lot of steps might let you lose some important information. This is where transformers innovate this approach. The architecture is reported in figure 3.13. The model is divided in two parts: encoder (left) and decoder (right).

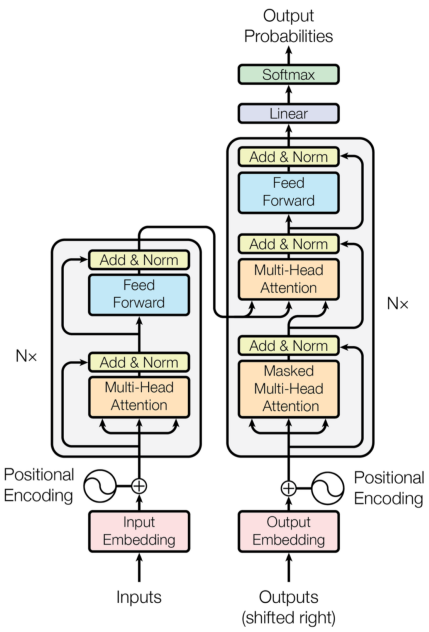


Figure 3.12: Transformer architecture.

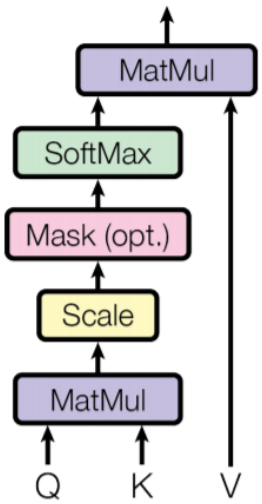


Figure 3.13: Attention block. Diagram showing the operations between values (V), keys (K) and queries (Q) to compute the attention matrix.

The transformer architecture relies on the concept of self-attention (Figure 3.13), defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3.36}$$

where K and V, keys and values, are the two copies of the output of the encoder and Q, queries coming from the decoder. The queries and keys have dimensions d_k and values of dimension d_v.

The main block of the model is the multihead attention, in particular the one connecting the encoder and the decoder. The multihead attention is a concatenation of h heads defined in Equation 3.36:

$$\text{multihead}(Q, K, V) = \text{concat}(\text{head}_1, ..., \text{head}_h)W^o \tag{3.37}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{3.38}$$

where the matrices W are the projection of queries, keys, and values into the embedding of size $d_{model} = 512$. In particular $W_i^Q \in \mathbb{R}^{d_{model} \text{x} d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \text{x} d_k}$,

$W_i^V \in \mathbb{R}^{d_{model} \text{x} d_v}$ and $W_i^O \in \mathbb{R}^{hd_v \text{x} d_{model}}$. In the original paper the number of heads h is set equal to 8 and $d_k = d_v = d_{model}/h = 64$. Note that the multihead attention layer performs computations that are position-independent. To use the order of the sequence, the positional encoding is added after the embedding layers. Sine and cosine functions are used and they are defined as:

$$\text{PE}_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{3.39}$$

$$\text{PE}_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{3.40}$$

where *pos* is the position in the sequence and i is the i-th feauture in the embedding. Note that the wavelenghts form a geometric progression from 2π to $1000 \cdot 2\pi$. Thus, each feature of every word extracted according to the embedding layer is mapped into a different value. This value encodes the position of the word in the sentence and it is summed to the output of the embedding layer.

Sharifzadeh et al. [30] proposed a method based on Graph Transformer [31] called Schemata. After extracting the features of the objects X_i^o and the predicates x_i^p, contextualized representations z_i are obtained. The model uses a Graph Transformer with multihead attention defined as:

$$\mathbf{m}_i^{\mathcal{N}_{(i)}} = \frac{1}{K} \sum_{k=1}^{N} \sum_{j \in \mathcal{N}_{(i)}} \alpha_{ij}^{(l,k)} \mathbf{W}^{(l,k)} \mathbf{z}_j^{(l.t)} \tag{3.41}$$

$$\mathbf{z}_i^{'(l)} = LN(\mathbf{z}_i^{(l,t)} + \mathbf{m}_i^{\mathcal{N}_{in}(i)} + \mathbf{m}_i^{\mathcal{N}_{out}(i)}) \tag{3.42}$$

$$\mathbf{z}_i^{(l+1,t)} = LN(\mathbf{z}_i^{'(l)} + f(\mathbf{z}_i^{'(l)})) \tag{3.43}$$

where $\mathbf{z}_i^{(l,t)}$ is the embedding of node i in the l-th graph convolution layer and t-th assimilation. The number of assimilations is equal to 4. In the first step, we set $\mathbf{z}_i^{(0,t)} = \mathbf{x}_i$. LN is the layer norm. K is the number of attentional heads and $\mathbf{W}^{(l,k)}$ is the weight matrix in the l-th layer of the k-th head. $\mathcal{N}_i$ are the neighbors of the node i. $f(\cdot)$ denote two feed-forward layers divided by a Leaky ReLU. The attention coefficients $\alpha_{ij}^{(l,k)}$ are defined as:

$$e_{ij}^{(l,k)} = \sigma(\mathbf{h}^{(l,k)} \cdot [\text{concat}(\mathbf{z}_i, \mathbf{W}^{(l,k)} \mathbf{z}_j^{(l)})]) \tag{3.44}$$

$$\alpha_{ij}^{(l,k)} = \frac{exp(e_{ij}^{(l,k)})}{\sum_{q \in \mathcal{N}(i)} exp(e_{iq}^{(l,k)}} \tag{3.45}$$

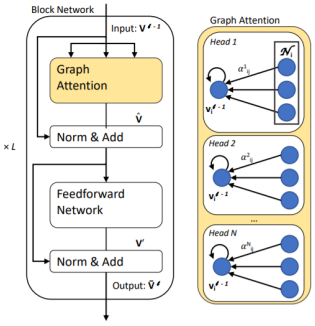


Figure 3.14: Graph Transformer architecture [31].

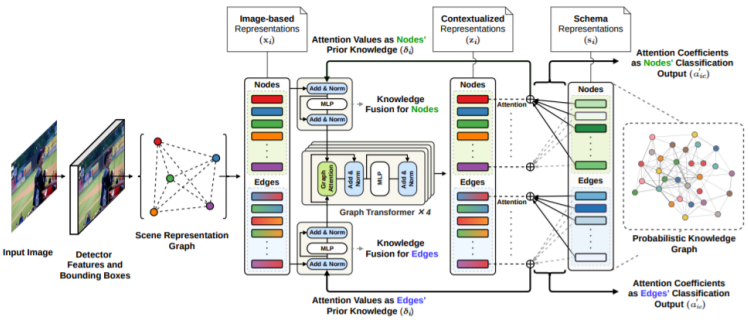


Figure 3.15: Schemata architecture [30].

where $h^{(l,k)}$ is a weight vector and σ is the leaky ReLU function.

The schema of a class c is defined as an embedding vector s_c. The classification outputs $\alpha_{ic}^{'}$ are computed as

$$\alpha_{ic}^{'} = \text{softmax}(\text{attention}(z_i^{(L,t)}, s_c) \tag{3.46}$$

The attentions values δ_i captures the messages and they are used to update the scene representations:

$$\delta_i = \sum_{c \in \mathcal{C}} \alpha_{ic}^{'} s_c \tag{3.47}$$

$$\mathbf{u}_i = LN(\mathbf{x}_i + \delta_i) \tag{3.48}$$

$$\mathbf{z}_i^{(0,t+1)} = LN(\mathbf{u}_i + g(\mathbf{u}_i)) \tag{3.49}$$

where u_i are the new features and $g(\cdot)$ is defined as $f(\cdot)$.

The model proposed by Wang et al. [32] introduce the use of the attention module to deal with the growing complexity of relationship search: $\mathcal{O}(NM)$, where N is the total number of object categories and M the total number of relationships. The model includes two parts: an object detector and a predicate prediction. As an object detector, a Faster-RCCN with VGGNet backbone is used [19]. The two parts are trained separately.

First, the object detector extracts visual features from the image. Then, the second part extracts non-visual features, including location and semantic information.

Given the bounding boxes areas of subject and object, the spatial information can be useful to retrieve the relationship (ex: "above"). Then the semantic information are extracted using the pretrained Word2Vec model as embedding. Thanks to the embedding, the pair subject and object are mapped in a D-space with D=300. The semantic features are crucial in a zero-shot prediction, where objects of the same category present some similarity (ex: animals).

Once we have obtained the visual and nonvisual features (V and u), they are concatenated and passed to the attention mechanism. The attention mechanism is applied recursively. The attention module is able to extract different parts of the image to get useful visual features. Thanks to this mechanism composed by a GRU layer and a CNN, we generate an attention mask M to obtain a new vector defined as follows:

$$v_{k+1} = \sum_i \sum_j M_k(i,j)V(i,j) \tag{3.50}$$

Then v_{k+1} is concatenated with the non-visual features u to begin the new iteration. Thus, the final predictions are:

$$s = \frac{1}{K+1}(s' + \sum_{k=0}^{K-1} W_s[u; v_{k+1}]) \tag{3.51}$$

where K is the total number of iterations and s' are the initial scores $s' = W_s[u; v_o]$.

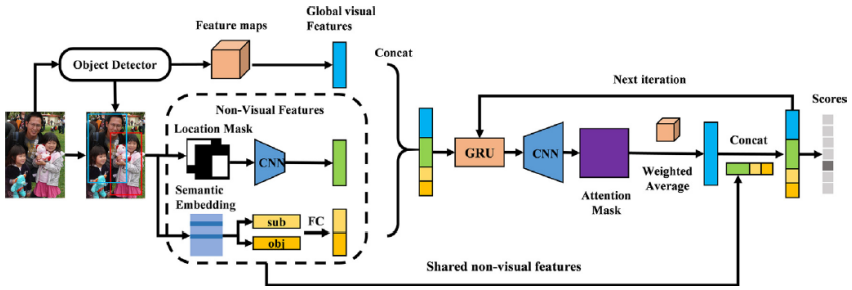


Figure 3.16: Recurrent attention architecture.

3.2 Datasets

The Scene Graph Generation, as other Computer Vision tasks, requires a lot of annotated data [33]. The main datasets used as benchmarks are the following: **Visual Relationship Detection (VRD)** [2] and **Visual Genome (VG)** [3]. Dataset statistics are reported in Table 3.1: unique number of objects, relationships, and attributes are considered. As we can see, the Visual Genome introduces a new type of instance: attributes for objects. Furthermore, a dataset dealing with Visual Relationships in video has been released recently, Action Genome [34]. With Action Genome, it is possible to deal also with the temporal variable, creating spatio-temporal scene graphs. This dataset will not be further addressed in this work as we are planning on working with Visual Relationships in images.

Dataset	Images	# Obj.	# Rel.	# Attr.	Avg Pred. per Obj.	Avg Attr. per Obj.
VRD	5000	100	70	-	24.25	-
VG	108077	80138	40480	40513	17.68	16.08

Table 3.1: Datasets statistics. Statistics on Visual Relationship Detection (VRD) and Visual Genome (VG).

As the VG dataset contains thousands of instances, the split provided by [3] will be used. In this split, the most 150 objects and 50 predicates are considered. Moreover, attributes are discarded.

3.3 Evaluation metrics and comparison

For the scene graph generation task, the first step in the pipeline is to extract the bounding boxes of the objects in the image. Table 3.2 reports the object detection models used by each model cited below. The most used one is the Faster R-CNN [19] with VGG-16 as backbone. All the object detectors used by these models have been pre-trained on the COCO dataset [35] following one of the earliest work done by [18].

Model	Object Detection Model (backbone)				
	R-CNN (VGG)	Fast R-CNN (VGG-16)	Faster R-CNN (VGG-16)	Faster R-CNN (ResNetXt-101-FPN)	Detectron with Faster R-CNN (VGG-16)
MotifNet [18]			✓		
TDE [20]				✓	
LKD [21]		✓			
KB-GAN [23]			✓		
LTN [4]	✓				
Graphical Contrastive Losses [25]					✓
CMAT [26]			✓		
Graph R-CNN [27]			✓		
Schemata [30]			✓		
Recurrent Attention [32]			✓		

Table 3.2: Object detectors used in the state-of-the-art models.

Moreover, specific metrics have been adopted by the research community [33]. Due to the sparse annotation in the datasets, the main metric used is recall. In particular **Recall@n (R@n)** is defined, which measures the fraction of correct predictions that appear among the top N confident predictions. All values are reported in percentage. Usually n is set equals to 20, 50, and 100. For the Visual Relationship Detection [2] and the Visual Genome dataset [3] different tasks are defined. Using the VRD dataset, an hyper-parameter k is added: k stands for the number of chosen predictions per object pair [21]. Notice that if k is not specified, we refer to k=70. It is the special case when all unique relationships of the dataset are considered (as seen in Table 3.1). Values of k used are 1, 10, and 70. The main tasks are:

- *predicate detection (PredDet)*: predicate prediction, given a pair of localized objects (both bounding boxes and labels);
- *phrase detection (PhrDet)*: locate the phrase ⟨subject, predicate, object⟩ in the image with a unique bounding box;
- *relationship detection (RelDet)*: define the triplets ⟨subject, predicate, object⟩ with a pair of bounding boxes.

The metrics are shown in Figure 3.17.

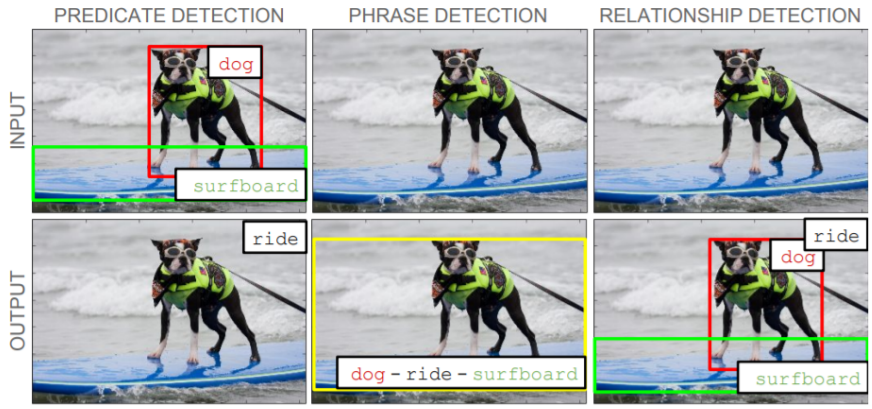


Figure 3.17: Metrics used for the VRD dataset [2]. Visual representation of all the metrics defined for the scene graph generation task applied to the VRD dataset.

Instead, using the VG dataset [3], the hyperparameter k is not used. The main tasks defined are the following:

- *predicate classification (PredCls)*: predict the relationships (edges) among object pairs given a set of ground-truth bounding boxes and labels;
- *phrase classification (PhrCls) or scene graph classification (SGCls)*: predict the triplets ⟨subject, predicate, object⟩ (edges and labels) given a set of localized objects;
- *scene graph generation (SGGEN) or (SGDet)*: predict the bounding boxes and the triplets in the image, an object is considered correct if it has at least 0.5 IoU overlap with the ground-truth bounding box.

Results concerning the VRD dataset [2] are reported in Table 3.3. Donadello et al. [4] report the results only in a zero-shot learning setting, so the results are obtained following their experiment setting. It is clear the model Graphical Contrastive Losses [25] outperforms all the other models expect for the PredDet, where the results are not provided. Furthermore, zero-shot learning experiments are proposed. Zero-shot learning is a type of learning where the ground-truth classes of the test set have few or zero examples in the training data. The results

are reported in Table 3.4. Few models proposed experiments in this setting and it is not clear which is the dominant one.

Model	PredDet		PhrDet						RelDet					
	R@50	R@100	R@50			R@100			R@50			R@100		
	k=70	k=70	k=1	k=10	k=70	k=1	k=10	k=70	k=1	k=10	k=70	k=1	k=10	k=70
LKD (U+W+SF+L: T+S) [21]	**85.64**	**94.65**	23.14	26.47	26.32	24.03	29.76	29.43	19.17	22.56	22.68	21.34	29.89	31.89
LTN ($\mathcal{T}_{prior}$) [4]	81.93	91.24	-	-	21.71	-	-	25.56	-	-	19.48	-	-	22.74
KB-GAN [23]	-	-	-	-	27.39	-	-	34.38	-	-	-	-	-	-
Graphical Contrastive Losses (COCO) * [25]	-	-	15.25	30.47	34.05	20.54	36.96	41.90	12.03	24.88	27.77	15.76	27.77	33.88
Recurrent Attention [32]	-	-	18.45	21.60	24.12	19.67	24.82	28.46	16.16	18.95	21.25	17.10	21.73	25.01

Table 3.3: Results on VRD dataset [2]. Results obtained in the standard setting, splitting the dataset into train and test set. Models marked with * were reimplemented.

Model	PredDet		PhrDet		RelDet	
	R@50	R@100	R@50	R@100	R@50	R@100
LKD (U+W+SF+L: S) [21]	54.20	74.65	**12.96**	**17.24**	**12.02**	**15.89**
LTN ($\mathcal{T}_{prior}$) [4]	**57.34**	**77.16**	11.40	15.74	10.47	14.43
Recurrent Attention (no prior) [32]	-	-	10.85	16.11	9.65	14.73

Table 3.4: Results on VRD dataset [2]. Results obtained with zero-shot learning.

Furthermore, new metrics have been proposed in recent work. In [20] **mean Recall (mR@n)** is used, first R@n is computed on all sample relationship and then averages over all relationships. This metric can balance the long-tailed relationships distribution: if a model performs well with high R@K, it could be not so efficient in all relationships.

The results are divided in **Graph Constraint** and **No Graph Constraint**. The difference between the two is the number of predicates allowed between object pairs: in the graph constraint case, only one is allowed. The models with the best performance are CMAT [26] and Schemata [30]. TDE [20] reports results on mean Recall, using the metric TDE addition overcomes the standard MotiFNet.

Model	Graph Constraint								
	SGGEN/SGDet			PhrCls/SGCls			PredCls		
	R@20	R@50	R@100	R@20	R@50	R@100	R@20	R@50	R@100
MotifNet (leftright) [18]	21.4	27.2	30.3	32.9	35.8	36.5	58.5	65.2	67.1
MotifNet + TDE [20] *	5.9	7.4	8.4	21.8	27.2	29.5	38.7	50.8	55.8
KB-GAN [23]	-	20.31	25.0	-	-	-	-	-	-
CMAT [26]	22.1	**27.9**	**31.2**	35.9	39.0	**39.8**	60.2	66.4	68.1
Graph-RCNN [27]	-	11.4	13.4	-	29.6	31.6	-	54.2	59.1
Schemata [30]	-	-	-	-	**39.1**	**39.8**	-	**66.9**	**68.4**
Recurrent Attention [32]	-	-	-	-	14.1	15.4	-	9.2	10.04

Table 3.5: Results on VG dataset [3] with graph constraint.

Model	No Graph Constraint								
	SGGEN/SGDet			PhrCls/SGCls			PredCls		
	R@20	R@50	R@100	R@20	R@50	R@100	R@20	R@50	R@100
CMAT [26]	23.7	31.6	36.8	41.0	**48.6**	52.0	68.9	83.2	90.1
Schemata [30]	-	-	-	-	48.5	**52.1**	-	**83.8**	**90.5**

Table 3.6: Results on VG dataset [3] without graph constraint.

The introduction of these complex metrics has caused a few errors in recent implementations. RelDN [25] implemented both the evaluation for the Visual Relationship Detection [2] and Visual Genome [3] in the same module mixing the different definitions. This implementation generates **wrong results**. In particular, the main issues are the following:

- *predicate detection PredDet vs predicate classification (PredCls)*: the difference between the two tasks is the presence of the ordered pairs (subject,object). In *PredDet* the pairs are given and in *PredCls* the pairs have to be defined by the model among the set of bounding boxes and labels in the image.
- *scene graph classification (SGCls)*: SGCls is computed based on the entire set of bounding boxes, not ordered pairs of bounding boxes (subject, object).

Using these wrong metrics, the task is made easier as the objects are already paired and divided in subgroups subjects and objects.

Tang et al. [36] define a new metric called **Top@K Accuracy (A@K)** to replace Recall@K. To be consistent with this new definition, the results provided

by RelDN [25] with graph constraints are reported in Table 3.7, along with the reimplemented MotifNet [36]. As we can see, the results are higher using this metric. Moreover, A@50 and A@100 accuracies are the same (A@20 is less as some images have more than 20 relationships). This confirms that the model RelDN [25] does not improve the state-of-the-art results: even a slightly change in the metrics definition can negatively impact the experiments results.

Model	**Graph Constraint**								
	SGGEN/SGDet			PhrCls/SGCls			PredCls		
	A@20	A@50	A@100	A@20	A@50	A@100	A@20	A@50	A@100
MotifNet [18]	-	-	-	40.41	**40.50**	**40.50**	68.87	**69.14**	**69.14**
Graphical Contrastive Losses (RelDN) [25]	21.1	28.3	32.7	36.1	36.8	36.8	66.9	68.4	68.4

Table 3.7: Results on VG dataset [3] with graph constraint and Top@K Accuracy. The results of the MotifNet are provided by [36].

Chapter 4

Methods and material

In this chapter, the usage of Logic Tensor Network applied to Visual Relationship detection is presented. The task is divided into two subtasks: object detection and triplet creation. The aim is to analyze the performance of both passages, looking for possible upsides and downsides in the model. Two datasets will be analyzed: Visual Relation detection dataset [2] and Visual Genome dataset [3]. An overview of the datasets has been reported in Chapter 3.

4.1 Logic Tensor Network Architecture

Following the discussion in Chapter 2, the LTN model can be applied to semantic image interpretation tasks. The training of the model is defined in a different way compared to standard machine learning models to be more robust for this specific task. Two different set of input data are defined, one based on the object detection task and one based on the triplet creation one. First, we consider the bounding boxes and labels. Second, we consider positive and negative examples from the triplets. Both data were taken from the training set. The retrieval and use of these inputs are reported in the following chapters.

4.2 Object Detection Module

The first step is to perform the object detection task. In our experiments, both R-CNN [9], Detectron with Faster R-CNN [14] and Faster R-CNN [19] are used.

During our experiments, a standardized pretraining procedure is followed to compare the results with other state-of-the-art models. For R-CNN, we use the procedure reported in [2], for Detectron with Faster R-CNN [25] and for Faster R-CNN [18]. The experiments are performed with the pre-trained models provided by [2] (R-CNN) and [25] (Detectron with Faster R-CNN and Faster R-CNN).

4.2.1 Rule-based grounding

After defining the object detector used, the images are used to obtain the following outputs:

- the bounding box coordinates $\langle x_0(b), y_0(b), x_1(b), y_1(b)\rangle$ where b is the bounding box, $\langle x_0(b), y_0(b)\rangle$ is the top-left corner, and $\langle x_1(b), y_1(b)\rangle$ is the bottom-right corner;
- the confidence score of the bounding box $score(b)$;
- the object class c, expressed in a label encoding or one-hot encoding format.

Having these results, a *rule-base grounding* can be created. A *rule-base grounding* is used to create features vectors according to the input data. In the case of bounding boxes, we can define two types of grounding.

The grounding $\widehat{\mathcal{G}}(b)$ of each bounding box b creates a feature vector $\mathbf{v}_b \in \mathbb{R}^{\mathcal{C}+4}$ defined as:

$$\mathbf{v}_b = \langle \textit{one-hot}(c),\ x_0(b), y_0(b), x_1(b), y_1(b)\rangle$$

Alternatively, the output softmax vector can be used instead of the one-hot class encoding vector. So for every bounding box b, the feature vector $\mathbf{v}_b$ is composed by its one-hot encoding object class and its coordinates.

However, in the visual relationship detection task, it is crucial to find the right object pairs having a relationship. Thus, for every bounding boxes pair $\langle b_1, b_2 \rangle$, the features vectors $\mathbf{v}_{b_1}$ and $\mathbf{v}_{b_2}$ can be concatenated. Moreover, to fully exploit the features in the visual space, jointly geometrical features are defined. The feature vector $\mathbf{v}_{b_1,b_2} \in \mathbb{R}^{2\cdot(\mathcal{C}+4)+7}$ is defined as:

$$\mathbf{v}_{b_1,b_2} = \langle \mathbf{v}_{b_1} : \mathbf{v}_{b_2}, ir(b_1,b_2), ir(b_2,b_1), \frac{area(b_1)}{area(b_2)}$$
$$\frac{area(b_2)}{area(b_1)}, \text{euclid\_dist}(b_1,b_2), sin(b_1,b_2), cos(b_1,b_2) \rangle$$

where:

- $ir(b_1,b_2) = intersec(b_1,b_2)/area(b_1)$ is the inclusion ratio;
- $area(b)$ is the area of the bounding box b;
- $intersec(b_1,b_2)$ is the area of intersection of bounding boxes b_1, b_2;
- $euclid\_dist(b_1,b_2)$ is the Euclidean distance between the centroids of bounding boxes b_1, b_2;
- $sin(b_1,b_2)$ and $cos(b_1,b_2)$ are the sine and cosine of the angle between the centroids of b_1 and b_2 computed clock-wise.

4.3 Triplet creation Module

After extracting the desired data from the images using the object detection module, the knowledge base used by the LTN can be generated.

4.3.1 Knowledge base usage

We followed the procedure proposed by Donadello et al. [4] to build the knowledge base. As discussed in Chapter 3, several types of knowledge sources can be used. In this case, the knowledge is gathered directly from the training set, looking at possible examples.

The following data is needed:

- domain ontology: given a predicate p, a set of n positive related subjects s^+ and m negative ones s^- are defined. For each predicate in the dataset, we will have the following sequence $< p, (s_0^+, ..., s_n^+), (s_0^-, ..., s_m^-) >$;
- range ontology: given a predicate p, a set of n positive related objects o^+ and m negative ones o^- are defined. For each predicate in the dataset, we will obtain the following sequence $< p, (o_0^+, ..., o_n^+), (o_0^-, ..., o_m^-) >$;
- images features: for each image, the bounding boxes b are extracted. Then a feature vector $\mathbf{v}_b \in \mathbb{R}^{\mathcal{C}+4}$ is computed as reported in Section 4.2.1. Thus, we will have B feature vectors for image, where B is the number of detected objects in the image: $< \mathbf{v}_{b1}, ..., \mathbf{v}_{bB} >$;
- triplets: for each pair subject-object, the in-between predicate is given. For each image, the following sequence is built: $< s_0, p_0, o_0 >, ..., < s_T, p_T, o_T >$ where T is the number of triplets present in the image.

Please note that the domain, range and triplets data is given by the ground truth, instead the image features are extracted used a pretrained object detector.

Following the definitions given in Section 2, a set of tensors are defined: literals and clauses. Clauses are obtained starting from the literals using a *literal aggregator*.

Starting from the domain and range ontology, four different domains are defined: two positive ones (positive subjects and objects) and two negative ones (negative subjects and objects). From these clauses and domain, two different LTN models can be built: $\mathcal{T}_{expl}$ and $\mathcal{T}_{prior}$. $\mathcal{T}_{expl}$ is based just on positive and negative clauses, instead $\mathcal{T}_{prior}$ contains both the clauses and the domains. The corresponding knowledge bases will be called $\mathcal{K}_{expl}$ and $\mathcal{K}_{prior}$. To go along with the setting discussion, the model $\mathcal{T}_{prior}$ will be presented. The model $\mathcal{T}_{expl}$ can be derived removing the use of the domains data.

At this point, the training process can start. Every k steps, a set of examples is sampled from the clauses, the domains and the bounding boxes. The parameter k will be called *frequency example dictionary*. The default number of examples is

100. These examples are then fed to the knowledge base. So at every k steps, the knowledge base is enriched with a new set of data. To group these tensors, a *clause aggregator* is defined. The total number of training iterations is $n * k$. The training stops if the saturation level of the knowledge base overcome a threshold given by the user, this will be called *saturation limit*. The saturation limit can be interpret as the level of fullness of the knowledge base: the more it is full, the more it will function properly.

Lastly, during the evaluation phase, the set of objects and their locations is computed on the test set using the object detection model. For every (subject, object) pair candidate, the probability of every predicate to form a relationship is computed. This vector of probabilities is obtained aggregating the predicate examples in the knowledge base. This aggregator is called *Refine Prediction Aggregator*.

In our experiment, a new aggregation operation is defined: the *focal log-sum* [37]. In this aggregator, a scaling factor γ is added to the standard cross entropy loss. Doing so, the correctly classified predicate will have a very small loss, instead the wrongly classified one have a bigger one. In this way, the focus is more oriented towards the errors made by the model. After the t-norm T is computed for the literal tensor L, the focal log-sum is computed in the following way:

$$FL(T(L)) = (1 - T(L))^{\gamma} log(T(L)) \tag{4.1}$$

4.3.2 Object detection error propagation

As the pipeline starts with the bounding boxes and classes provided by the object detection module, a wrong detection could result is a wrong set of possible relationships. This will be analyzed in the next section.

4.4 Experiments

The experiments are based on the Visual Relationship Detection dataset [2] and the Visual Genome dataset [3]. Information about both datasets are reported in

Section 3.

The Visual Relationship Detection dataset [2] is used by Donadello et al. [4] developing the original LTN model, so the first step is to reproduce their work. To provide a comparison with the original work by Lu et al. [2], the object detector used is a R-CNN with a VGG backbone. The data used to train the knowledge base are provided by the authors.

The experiment is divided into two phases:

- *knowledge base definition*, the experiments are performed using both a knowledge base with constraints $\mathcal{K}_{prior}$ and a knowledge base without constraints $\mathcal{K}_{expl}$.

- *evaluation*, the results are computed according to the tasks *predicate detection (PredDet), phrase detection (PhrDet) and relationship detection (RelDet).* The metric used is recall: Recall@50 and Recall@100. For simplicity, they will be called R@50 and R@100. All the relationships in the dataset are considered (k=70). Those tasks are fully explained in section 3.3.

The complete LTN model with a knowledge base with constraint $\mathcal{K}_{prior}$ will be called $\mathcal{T}_{prior}$. Thus we have two different models: $\mathcal{T}_{prior}$ and $\mathcal{T}_{expl}$.

We report in Table 4.4 the hyper-parameters used by the work of Donadello et al. [4].

LTN Baseline	
Hyperparameters	Value
Learning Rate	1e-2
Optimizer	RMSProp
Decay	0.9
Smooth factor (λ)	1e-15
t-norm	Łukasiewicz
Literal Aggregator	Harmonic mean
Clause Aggregator	Harmonic mean
Refine Prediction Aggregator	Maximum
Batch size examples	100
Training iterations	2500
Frequency example dictionary	250
Saturation threshold	0.96

Table 4.1: Logic Tensor Network baseline hyperparameters. Hyperparameters used in the experiments by Donadello et al. [4] with the VRD dataset [2].

All experiments were implemented with Python 3 and TensorFlow 2 using a shared Tesla V100 GPU. No data augmentation has been applied to the datasets.

Chapter 5

Results

In this section, the results of all the experiments are reported. The experiments are performed on the Visual Relationship Detection [2] and Visual Genome datasets [3].

5.1 Visual Relationship Detection

In this section, results using the Visual Relationship Detection dataset are provided.

5.1.1 Baseline

The first set of experiments reproduces the work by Donadello et al. [4]. His work focuses only on a zero-shot learning setting. Our results are expressed both in a *standard setting* (train and test split with comparable data distributions) and in a *zero-shot learning setting* as discussed in section 3.3. Results are reported in Tables 5.1 and 5.2.

Model	**PredDet**		**PhrDet**		**RelDet**	
	R@50	R@100	R@50	R@100	R@50	R@100
Ours: $\mathcal{T}_{expl}$	81.55	**91.71**	21.55	25.50	19.30	22.72
Ours: $\mathcal{T}_{prior}$	**81.93**	91.24	**21.71**	**25.56**	**19.48**	**22.74**

Table 5.1: Logic Tensor Network baseline results in the standard setting. Results obtained replicating the work by Donadello et al. [4] with the VRD dataset [2].

Model	PredDet		PhrDet		RelDet	
	R@50	R@100	R@50	R@100	R@50	R@100
$\mathcal{T}_{expl}$ [4]	56.25	74.71	11.00	15.91	10.01	**14.65**
$\mathcal{T}_{prior}$ [4]	57.34	**77.16**	11.40	15.74	10.47	14.43
Ours: $\mathcal{T}_{expl}$	55.69	73.79	11.29	14.88	10.44	13.69
Ours: $\mathcal{T}_{prior}$	**58.00**	77.07	**11.89**	**16.08**	11.12	**14.63**

Table 5.2: Logic Tensor Network baseline results in the zero-shot setting. Comparison between our results and the results obtained by Donadello et al. [4] with the VRD dataset [2].

As discussed by Donadello et al. [4], the introduction of logical constraints improves the overall performance of the model. Such improvements are even better in the zero-shot learning setting. This is reasonable as unseen objects and/or relationship are compensated by the facts stored in the knowledge base.

5.1.2 Logical Constraints

Since the introduction of logical constraints shows an improvement on the results, further experiments are reported. In particular, the default aggregators (*literal aggregator, clause aggregator, refine prediction aggregator*) will be changed to observe how they impact on the overall results. Moreover, two models will be used per aggregators configuration: $\mathcal{T}_{expl}$ and $\mathcal{T}_{prior}$. Notice that when logical constraints are used we refer to the $\mathcal{T}_{prior}$ model.

The results obtained can be seen in Table 5.3. The best configuration of aggregators is the one proposed by Donadello et al. [4] with (*harmonic mean, harmonic mean, maximum*) and the use of logical constraints. Furthermore, also the triplet (*harmonic mean, harmonic mean, harmonic mean*) obtains results close to the baseline. Other experiments are performed using the focal log-sum, but the performances drop considerably.

Aggregator	Clause Aggregator	Refine Predicate Aggregator	Logical Constraints	PredDet		PhrDet		RelDet	
				R@50	R@100	R@50	R@100	R@50	R@100
Harmonic mean	Harmonic mean	Maximum	✓	**81.93**	**91.24**	**21.71**	**25.56**	**19.48**	**22.74**
Harmonic mean	Harmonic mean	Maximum		81.55	91.71	21.55	25.50	19.30	22.72
Harmonic mean	Harmonic mean	Minimum	✓	79.98	88.35	21.31	25.10	19.08	22.27
Harmonic mean	Harmonic mean	Minimum		78.87	87.29	21.26	25.19	18.96	22.32
Harmonic mean	Harmonic mean	Harmonic mean	✓	80.62	89.32	21.54	25.61	19.26	22.64
Harmonic mean	Harmonic mean	Harmonic mean		79.43	88.03	21.50	25.29	19.30	22.49
Focal log-sum	Harmonic mean	Maximum	✓	53.00	76.04	1.58	4.46	1.32	3.80
Focal log-sum	Harmonic mean	Maximum		52.56	75.48	1.52	4.35	1.27	3.68
Focal log-sum	Harmonic mean	Minimum	✓	52.99	76.04	1.62	4.45	1.36	3.78
Focal log-sum	Harmonic mean	Minimum		51.56	75.31	1.53	4.35	1.30	3.64
Focal log-sum	Harmonic mean	Harmonic mean	✓	52.97	75.98	1.62	4.46	1.36	3.81
Focal log-sum	Harmonic mean	Harmonic mean		52.36	75.42	1.43	4.34	1.25	3.65

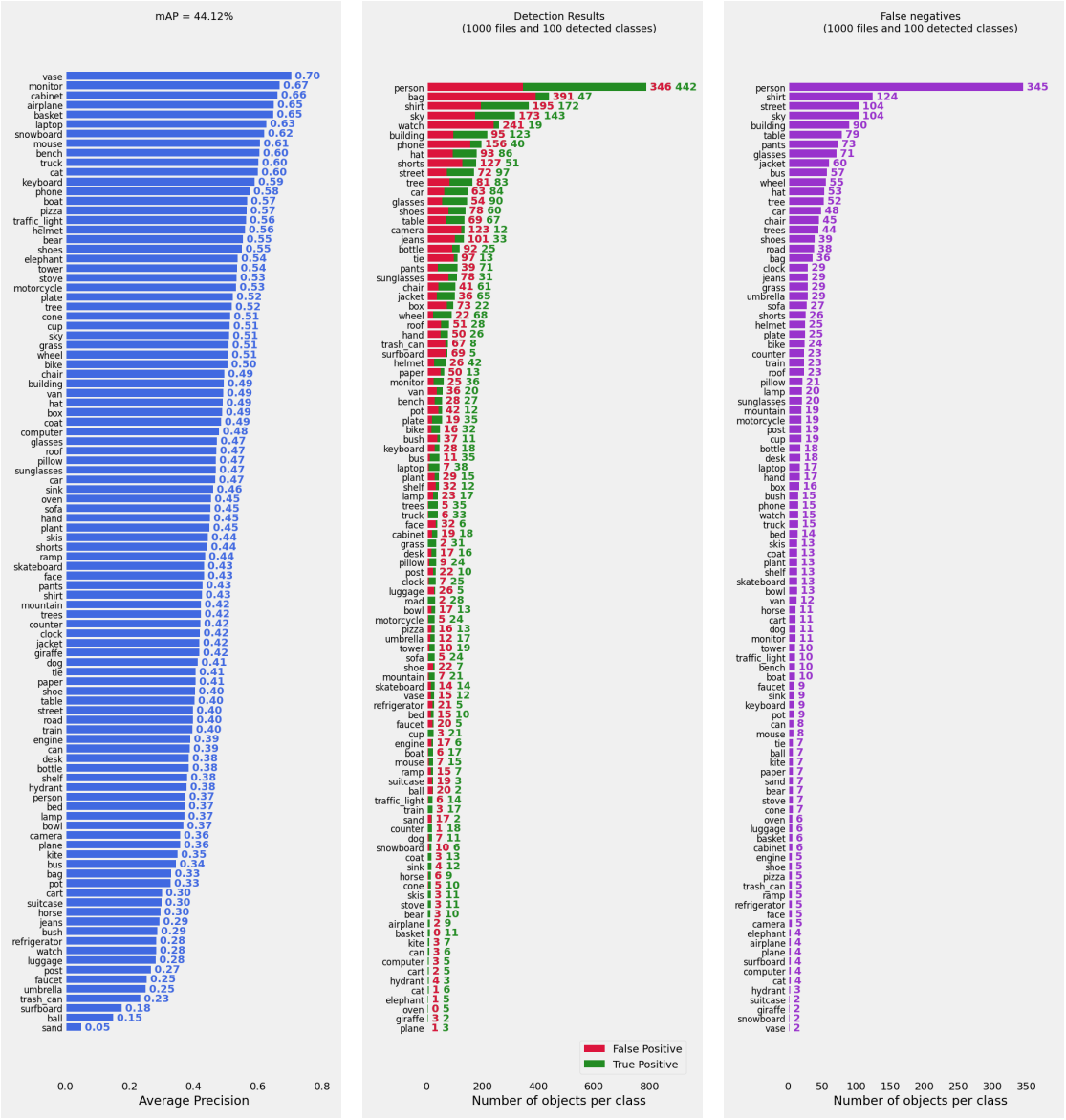
Table 5.3: Results using different aggregators.

5.1.3 Object Detector

Another useful experiment to improve the baseline is to change the object detector. The object detector used is Detectron with Faster R-CNN [14] which is a more recent state-of-the-art compared to R-CNN [9]. To obtain a better output than R-CCN, non-maximum suppression with $IoU = 0.5$ and a confidence threshold equals to 0.6 are applied.

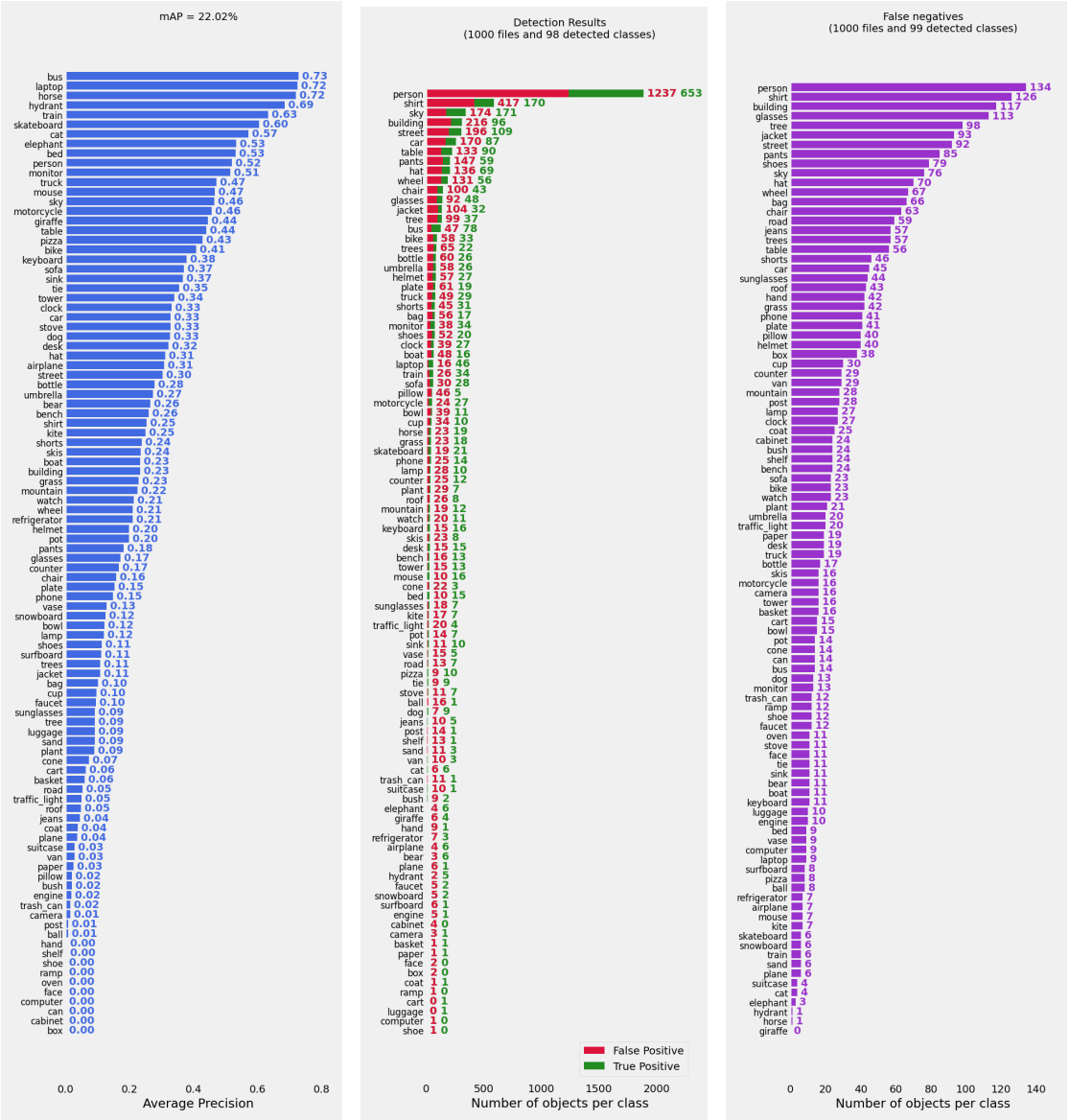
The goal of this setting is to analyze the quality of the object detector output (bounding boxes and class labels) and how it can improve or not the task concerning triplet creation.

First, the output is evaluated using mean Average Precision, mAP, (discussed in Section 2) and the class labels prediction distribution, true positives, false positives, and false negatives. The output is shown is Figures 5.1 and 5.2.



(a) mAP (b) TP and FP (c) FN

Figure 5.1: Object Detection results using R-CNN [9].



(a) mAP (b) TP and FP (c) FN

Figure 5.2: Object Detection results using Detectron [14].

In the general case, there are two types of errors to avoid:

- false positives, they are wrong detected objects that could lead to a set of wrong relationships;
- false negatives, ground-truth objects are not detected, so all relationships related are lost.

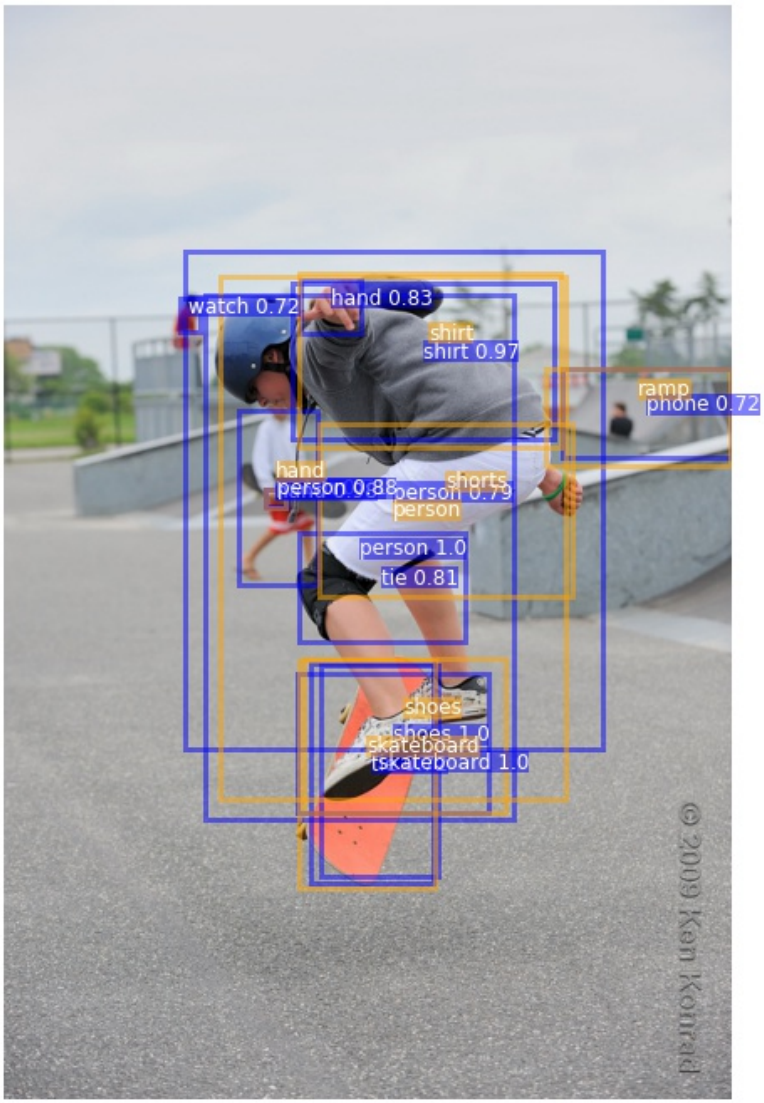


Figure 5.3: False positive example obtained using R-CNN. In this picture, a *tie* has been detected but it is not in the ground-truth objects.

In the R-CNN case, the mean Average Precision is 44.12% and the standard deviation is 11.82%. In this case, objects with borderline AP are not present (close to 100% or close to 0%). However, for some objects, there are more errors than right detections. For example, the class *person* has 442 TP, 348 FP and 345 FN (442 right vs 693 wrong) and the class *shirt* has 172 TP, 183 FP, and 124 FN (172 right vs 307 wrong).

Instead, the Detectron has an mean Average Precision of 22.02% and the standard deviation is 11.82%. In fact, a lot of objects have a very low AP (less than 10%). This can be seen also in the false negatives distribution.

Different errors by both models can result in different impact on the relationship creation. Some examples are provided in Figure 5.3 and 5.4. Orange Bounding boxes are the ground truth ones, instead the blue ones are the output of the detection. Every bounding box is associated with its class, which is located in the center. Detected bounding boxes also report the confidence score. In Figure 5.3, due to the false positive *tie*, the model detected the following wrong relationship related to it: ⟨*tie on skateboard*⟩, ⟨*tie on person*⟩, ⟨*person wear tie*⟩.

Figure 5.4: False negative example obtained using Detectron. In this picture, a *helmet, wheel, bag, building* have not been detected.

Then, the phrase and relationship detection results are reported comparing the two different object detectors. These object detectors are used in the inference phase, while the input of the knowledge base is gathered from the R-CNN detections as the previous experiments. The model configuration used is the baseline reported in section 5.1.1. To have a fully understanding of the analysis, an experiment using ground truth bounding boxes and class labels is added. In this way, it is possible to understand how the model performances in the best-case scenario. The results concerning predicate detection are not reported as they require as input the same ground truth bounding boxes and class labels (these would be the same for all the three object detectors considered). Thus, the recall will not change using different object detectors. Results are reported in Table 5.4. Comparing all three cases, starting from the ground truth bounding boxes provides the best results in both metrics. However, these results are not optimal as expected. The LTN model is not able to create relationships in a superlative way even if the object locations and classes are correct. However, the results increase by more than a factor of 2, showing that a good input is crucial for our model.

Object Detector	**Logical Constraints**	**PhrDet**		**RelDet**	
		R@50	R@100	R@50	R@100
R-CNN [9]	✓	21.71	25.56	19.48	22.74
R-CNN [9]		21.55	25.50	19.30	22.72
Detectron [14]	✓	12.03	16.34	9.43	13.07
Detectron [14]		10.64	14.81	8.51	11.74
Ground Truth	✓	**53.96**	**62.94**	51.77	60.64
Ground Truth		53.61	63.35	**51.82**	**61.21**

Table 5.4: Results comparing different input. The input used by the LTN model (bounding boxes and class labels) is changed according to the model. Notice that ground truth data is also used.

5.2 Visual Genome

A set of experiments has been performed on the Visual Genome dataset [3]. The object detector used is Faster R-CNN [19]. The pretraining of this model is reported by the work of Zhang et al. [25]. The split provided by [3] is used.

The knowledge base is built as done for the Visual Relationship Detection dataset [2]. The hyperparameters used are reported in Table 4.4. Comparing the data to build the VG knowledge base to VRD, 50 more object classes and 20 less predicate classes are considered. Following the work done by Donadello et al. [4], the input files for the knowledge base are created. Regarding the domain and range ontologies, Donadello et al. declare that they have been created manually. This approach is discarded as it could lead to a biased set of examples. Instead, these ontologies are created starting from the triplet distributions in the training set. For each predicate in the training set, the top 25 subjects and objects per frequency are taken. In this way, positive examples are created.

The images features are obtained from the predicted object coming from the Faster R-CNN object detector. This source of knowledge is extremely bigger compared to the VRD dataset as we are analyzing hundreds of thousands of images.

For this experiments, the model $\mathcal{T}_{prior}$ is considered as it reported the best results in all scenarios using the VRD dataset.

In this case, the quantity of data used is 20 times bigger than the VRD case (500 MB vs 25 MB). Due to the huge quantity of data compared to VRD, the knowledge has not reached the fixed saturation limit.

Chapter 6

Discussion

The results obtained from the LTN model show that embedded knowledge as logical constraints can outperforms the standard learning procedure of a general machine learning model. However, it still needs improvements to function in large-scale problems as the Visual Relationship detection task applied to the Visual Genome dataset [3].

One of the possible enhancements that can be done is to fine-tune the rule-based grounding of bounding boxes. Additional features that can be added are the RoI features and the embedding of the object class using models like Word2Vec [38].

Moreover, logical constraints and the positive and negative examples are created upon the objects and predicates frequencies in the training set. This input knowledge can be enriched by external sources where the semantic meaning of words and relationship examples are provided. One of them is the knowledge graph ConceptNet [22]. Once this new set of logical constraints is defined, improved methods for aggregation can be explored.

Furthermore, an end-to-end model can be created to perform both object detection and triplet creation. In this way, the error related to the relationship detection can be backpropagated, improving significantly the results.

Bibliography

[1] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. «Image Retrieval Using Scene Graphs». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015 (cit. on pp. 1, 3).

[2] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. «Visual Relationship Detection with Language Priors». In: *European Conference on Computer Vision*. 2016 (cit. on pp. 2–4, 25, 38–42, 44, 45, 48–52, 59).

[3] Ranjay Krishna et al. «Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations». In: 2016. URL: `https://arxiv.org/abs/1602.07332` (cit. on pp. 3, 4, 21, 25, 38–40, 42–44, 48, 51, 58, 60).

[4] Ivan Donadello and Luciano Serafini. *Compensating Supervision Incompleteness with Prior Knowledge in Semantic Image Interpretation*. 2019. arXiv: `1910.00462 [cs.LG]` (cit. on pp. 4, 24, 27, 39–41, 46, 49–52, 59).

[5] Lus C. Lamb, Artur S. d'Avila Garcez, Marco Gori, Marcelo O. R. Prates, Pedro H. C. Avelar, and Moshe Y. Vardi. «Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective». In: *CoRR* abs/2003.00330 (2020). arXiv: `2003.00330`. URL: `https://arxiv.org/abs/2003.00330` (cit. on p. 5).

[6] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE*. 1998, pp. 2278–2324 (cit. on p. 5).

[7] Lofti A. Zadeh. «Fuzzy Sets». In: *Information and Control* 8 (1965), pp. 338–353. URL: http://www-bisc.cs.berkeley.edu/Zadeh-1965.pdf (cit. on p. 6).

[8] Ivan Donadello, Luciano Serafini, and Artur d'Avila Garcez. *Logic Tensor Networks for Semantic Image Interpretation*. 2017. arXiv: 1705.08968 [cs.AI] (cit. on p. 6).

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV] (cit. on pp. 12, 16, 45, 53, 54, 58).

[10] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV] (cit. on p. 12).

[11] Lilian Weng. «Object Detection for Dummies Part 3: R-CNN Family». In: *lilianweng.github.io/lil-log* (2017). URL: http://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html (cit. on p. 13).

[12] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. «Selective Search for Object Recognition». In: *International Journal of Computer Vision* (2013). DOI: 10.1007/s11263-013-0620-5. URL: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf (cit. on p. 12).

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV] (cit. on p. 14).

[14] Yuxin Wu and Kaiming He. *Group Normalization*. 2018. arXiv: 1803.08494 [cs.CV] (cit. on pp. 14, 45, 53, 55, 58).

[15] Tan Wang, Jianqiang Huang, Hanwang Zhang, and Qianru Sun. *Visual Commonsense R-CNN*. 2020. arXiv: 2002.12204 [cs.CV] (cit. on p. 14).

[16] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. «A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit». In: *Electronics* 10.3 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10030279. URL: https://www.mdpi.com/2079-9292/10/3/279 (cit. on p. 15).

[17] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. *Soft-NMS – Improving Object Detection With One Line of Code*. 2017. arXiv: 1704.04503 [cs.CV] (cit. on p. 16).

[18] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. «Neural Motifs: Scene Graph Parsing With Global Context». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 (cit. on pp. 21, 38, 39, 42, 43, 45).

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV] (cit. on pp. 21, 23, 32, 36, 38, 45, 58).

[20] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. *Unbiased Scene Graph Generation from Biased Training*. 2020. arXiv: 2002.11949 [cs.CV] (cit. on pp. 23, 39, 41, 42).

[21] Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. «Visual Relationship Detection With Internal and External Linguistic Knowledge Distillation». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017 (cit. on pp. 24, 39, 41).

[22] Robyn Speer, Joshua Chin, and Catherine Havasi. *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge*. 2018. arXiv: 1612.03975 [cs.CL] (cit. on pp. 24, 26, 60).

[23] Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. «Scene Graph Generation With External Knowledge and Image Reconstruction». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on pp. 24, 26–28, 39, 41, 42).

[24] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning. «Generating Semantically Precise Scene Graphs from Textual Descriptions for Improved Image Retrieval». In: *Proceedings of the Fourth*

Workshop on Vision and Language. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 70–80. DOI: 10.18653/v1/W15-2812. URL: https://www.aclweb.org/anthology/W15-2812 (cit. on p. 24).

[25] Ji Zhang, Kevin J. Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. *Graphical Contrastive Losses for Scene Graph Parsing*. 2019. arXiv: 1903.02728 [cs.CV] (cit. on pp. 27, 28, 39–43, 45, 58).

[26] Long Chen, Hanwang Zhang, Jun Xiao, Xiangnan He, Shiliang Pu, and Shih-Fu Chang. «Counterfactual Critic Multi-Agent Training for Scene Graph Generation». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019 (cit. on pp. 30, 39, 41, 42).

[27] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. *Graph R-CNN for Scene Graph Generation*. 2018. arXiv: 1808.00191 [cs.CV] (cit. on pp. 32, 39, 42).

[28] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG] (cit. on p. 32).

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention is All You Need». In: 2017. URL: https://arxiv.org/pdf/1706.03762.pdf (cit. on p. 33).

[30] Sahand Sharifzadeh, Sina Moayed Baharlou, and Volker Tresp. *Classification by Attention: Scene Graph Classification with Prior Knowledge*. 2020. arXiv: 2011.10084 [cs.CV] (cit. on pp. 35, 36, 39, 41, 42).

[31] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. *Text Generation from Knowledge Graphs with Graph Transformers*. 2019. arXiv: 1904.02342 [cs.CL] (cit. on pp. 35, 36).

[32] Lei Wang, Peizhen Lin, Jun Cheng, Feng Liu, Xiaoliang Ma, and Jianqin Yin. «Visual relationship detection with recurrent attention and negative sampling». In: *Neurocomputing* 434 (2021), pp. 55–66. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.12.099. URL: https://www.

sciencedirect.com/science/article/pii/S0925231220320117 (cit. on pp. 36, 39, 41, 42).

[33] Aniket Agarwal, Ayush Mangal, and Vipul. *Visual Relationship Detection using Scene Graphs: A Survey*. 2020. arXiv: 2005.08045 [cs.CV] (cit. on pp. 38, 39).

[34] 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*. 2020, pp. 10236–10247 (cit. on p. 38).

[35] Tsung-Yi Lin et al. «Microsoft COCO: Common Objects in Context». In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312 (cit. on p. 38).

[36] Kaihua Tang. *A Scene Graph Generation Codebase in PyTorch*. https://github.com/KaihuaTang/Scene-Graph-Benchmark.pytorch. 2020 (cit. on pp. 42, 43).

[37] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV] (cit. on p. 48).

[38] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL] (cit. on p. 60).