POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Scene Graph Generation in Autonomous Driving: a Neuro-symbolic approach

Supervisors

Prof. Lia MORRA

Prof. Fabrizio LAMBERTI

Candidate

Paolo Emmanuel Ilario DIMASI

December 2023

Abstract

The 2022 study on traffic fatalities in Italy by the Italian National Institute of Statistics (ISTAT) reports 454 daily fatalities and 561 injuries, primarily due to distractions. Then, the success of Autonomous Driving depends on intelligent perception systems to enhance road safety, with vision systems playing a critical role throughout its history.

In the field of Computer Vision, Deep Learning has gained mainstream acceptance for its ability to model complex problems like Object Detection and Instance Segmentation. More recently, Scene Graph Generation has emerged as a novel paradigm, where scenes are depicted as graphs with objects as nodes and their relationships as edges. This area has seen substantial research, but only a limited fraction of it pertains to autonomous driving applications and most of it focuses on specific traffic scenarios, limiting diversity. A comprehensive effort for incorporating all relevant objects in traffic scenarios resulted in the creation of the Traffic Genome dataset. However, it suffers from bias due to uneven relationship frequency, which can lead to the misclassification of rare events. This thesis addresses this issue by infusing prior knowledge into scene graph generation using neuro-symbolic approaches. Relational Transformer network is used as baseline, due to its state-of-art results in the one-stage approaches.

Two methodologies for knowledge injection are adopted. The first involves external knowledge injection through Knowledge Graph Embedding (KGE) techniques, using PandaSet, an autonomous driving dataset released by Hesai and Scale AI, as the foundational knowledge base due to its rich multi-modal information. The second approach utilizes the Logic Tensor Network (LTN) for constraint satisfaction, employing axioms as constraints during training.

Results indicate that both methods improve performance, with the choice depending on the trade-off between deployment speed and accuracy: KGE methods are faster to develop but limited by available relationships in the knowledge base, while LTN potentially can outperform them but requires more time to design optimal axioms based on domain expertise.

Acknowledgements

First and foremost, I would like to express my gratitude to my advisors, Prof. Fabrizio Lamberti, for providing me with this opportunity, and Prof. Lia Morra, for her availability, patience, and guidance throughout this thesis. Her mentorship has been instrumental in introducing me to the fascinating realm of explainable AI systems, which is increasingly vital for the ethical success of AI.

I extend heartfelt thanks to my parents and sisters for their support throughout these years. Their encouragement has been a constant source of motivation.

A big thanks also to all friends and colleagues with whom we have shared joy and suffering during these years of university.

Lastly, but by no means least, I extend my gratitude to the student teams DRAFT PoliTO and Squadra Corse Driverless for providing me with the incredible opportunity to work on cutting-edge technology within the context of perception systems for Autonomous Driving. A special acknowledgment goes to PhD. Simone Godio, PhD. Students Francesco Marino and Stefano Favelli, whose efforts have played a pivotal role in making this opportunity a reality.

Table of Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence
AUQ	Aggregator norm of Universal Quantifier
CEL	Cosine Embedding Loss
CNN	Convolutional Neural Network
CRF	Conditional Random Field
CSA	Coupled Self-Attention
DEA	Decoupled Entity Attention
DVA	Decoupled Visual Attention
EC	Easy Constraints
FNC	First Negative Constraints

FOL
: First Order Logic

HC
: Hard Constraints

HEL
: Hinge Embedding Loss

GCN
: Graph Convolutional Network

GRU
: Gated Recurrent Unit

IMP
: Iterative Message Parsing

IoU
: Intersection over Union

KG
: Knowledge Graph

KGE
: Knowledge Graph Embedding

LiDAR
: Light Detection And Ranging

LSD
: Label Semantic Distributions

LSTM
: Long Short-Term Memory

LTN
: Logic Tensor Network

mAP

mean Average Precision

MHA

Multi-Head Attention

MPNN

Message Passing Neural Network

MLP

Multi-Layer Perception

mR

mean Recall

NAS

Neural Search Architecture

NeSy

Neuro-Symbolic AI

NLP

Natural Language Processing

NMS

Non-Max Suppression

PredCLS

Predicate CLaSsification

RDF

Resource Description Framework

RMP

Relation-aware Message Passing neural network

RNN

Recurrent Neural Network

SAE	Society of Automotive Engineer
SGG	Scene Graph Generation
SLD	Simulated Label Distributions
SNC	Second Negative Constraints
SOTA	State Of The Art
TDE	Total Direct Effect

Chapter 1

Introduction

The scene graph definition is credited to Johnson et al.[1]. A scene graph is a data structure that describes the contents of a scene, i.e. it encodes object instances, attributes of objects, and relationships between objects. Formally, given a set of object classes $\mathcal{C}$, a set of attribute types $\mathcal{A}$, and a set of relationship types $\mathcal{R}$, a scene graph $\mathcal{G}$ is a tuple $\mathcal{G} = (\mathcal{O}, \mathcal{E})$ where $\mathcal{O} = \{o_1, ..., o_n\}$ is a set of objects and $\mathcal{E} \subseteq \mathcal{O} \times \mathcal{R} \times \mathcal{O}$ is a set of edges. Each object has the form $o_i = (c_i, A_i)$ where $c_i \subseteq \mathcal{C}$ is the class of the object and $A_i \subseteq \mathcal{A}$ are the attributes of the object. Note that the graph is directed since symmetric relationship usually does not holds: a nose can belong to a person, but the contrary is impossible in our world. Figure 1.1 shows an example of scene graph: the object instances are people (*girl*), places (*tennis court*), things (*shirt*) and parts of other objects (*arm*). This example demonstrates that scene graph can represent the detailed semantics of a dataset of scene, achieving a high number of visual tasks, including action recognition[2], image captioning[3] and visual relationship detection[4]. Furthermore, this semantic richness could be a game charger in Autonomous Driving, particularly for achieving the SAE-5 level of autonomy[5]. Enhancing the safety of Visual-Controllers by preemptively understanding relationships between objects can significantly improve risk assessment. Consider a scenario where a child rushes onto the road to retrieve a ball unintentionally kicked out of a park; with prior relationship detection between the ball and the child, the controller can swiftly predict the danger and initiate a braking maneuver. Currently, in the Autonomous Driving domain few researches have been conducted in the realm of of Scene Graph Generation (SGG) and the majority of them present several issues: focus on specific sub-task[6, 7, 8, 9] with lack of generalization; usage of spatial relationships derived by manual heuristic[6, 8]. These problems hinder the understanding of the effectiveness of

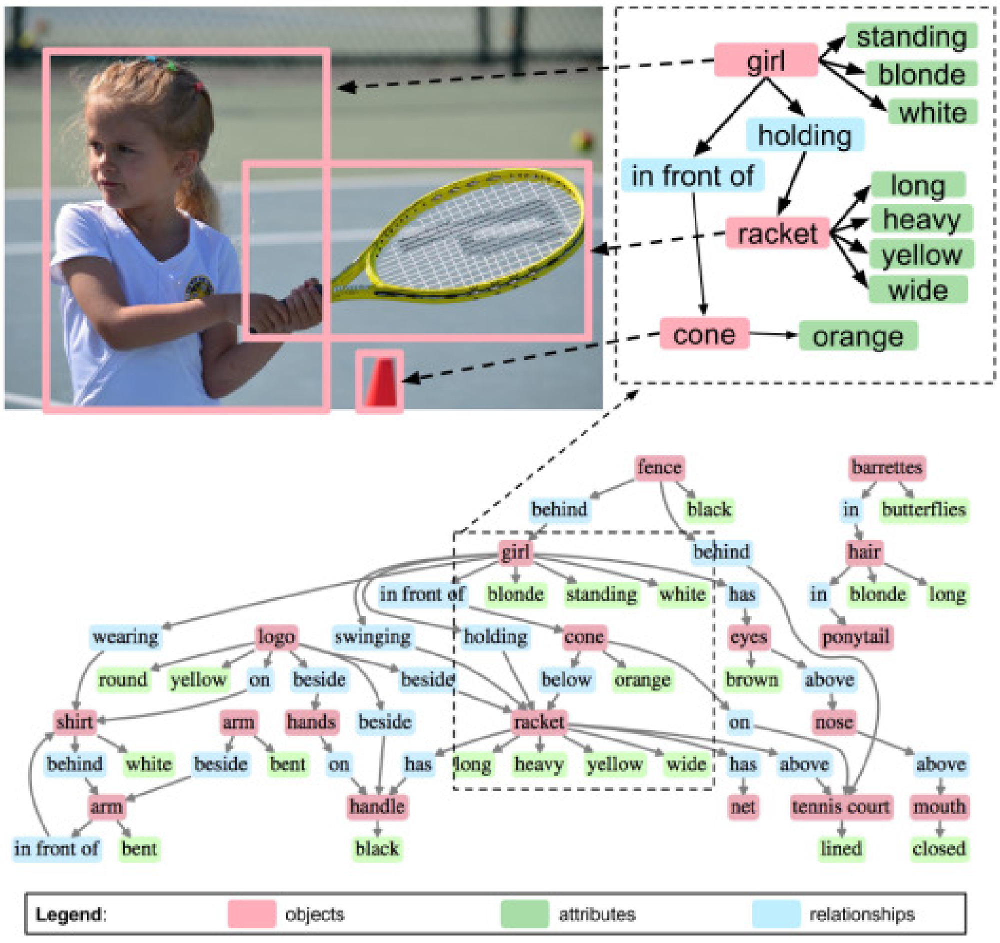


Figure 1.1: An example of a scene graph and its grounding

different methodologies and the applicability of state-of-the-art methods trained on datasets not specifically tailored for Autonomous Driving, like Visual Genome. Only [10] introduces a public benchmark dataset, Traffic Genome, offering a more comprehensive scenario by incorporating a high number of entities and relationships.

The focus of this work will be on visual relationship detection application in Autonomous Driving scenario based on Traffic Genome dataset. The visual relationship detection is mainly composed of two steps: 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, the class labels x_i are predicted. After collecting all the objects O_i in the image, the next passage is to link them using predicates. In the pair, a subject s_i and an object o_i are defined because the graph must be directed. Thus, the model focuses on learning a predicate p capable to link two objects o_i and o_j, where one of them is defined as the subject and the other the object. At the end of this step, a triplet <subject, predicate, object> is created. This operation is repeated for all pairs in the image to generate the final scene graph. In literature are presented two techniques employing these steps: **two-stage** method, where object and relationship detection are done asynchronously by separate networks; **one-stage** method, where the goal is to design a network capable of doing all these passages at once in a synchronous way. Due to the complexity of visual relationship detection task, several issue can occurs during its realization. Hereby, we reported the main ones:

Missing annotations in noisy dataset both Visual Genome Dataset[4], the most used dataset, and Traffic Genome have missing annotations, i.e. dataset reports just one possible predicate even when more than one predicate could be correct for a pair of objects, leading problems during training because while the model could correctly detect the triplets in the image, they will considered wrong since they do not appear in the dataset.

Long-tailed relationship distribution the relationship distributions of available datasets are exponential distributions (Visual Genome Dataset, Visual Relationship Detection[11], Traffic Genome), i.e. 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. To mitigate this problem several prior-knowledge infusion techniques and metric have been introduced.

The structure of thesis is organized as follows:

Chapter 1 contains an introduction to the thesis' topic.

Chapter 2 illustrates Neuro-Symbolic AI and its prior-knowledge approaches.

Chapter 3 reviews of SOTA for Scene graph generation.

Chapter 4 illustrates the datasets used for experiments.

Chapter 5 reports the methods used for experiments.

Chapter 6 shows the results of the experiments over autonomous driving dataset.

Chapter 7 after a brief results' discussion proposes possible future works.

Chapter 2

Embedding prior knowledge in deep learning

2.1 Neuro-Symbolic AI

Neuro-symbolic AI, the so-called 3rd wave[12], is a branch of machine learning that aims to build hybrid AI systems that infuse commonsense reasoning into neural networks exploiting symbols, i.e. high-level representation of the dataset instances. Thanks to symbolic representation, Neuro-symbolic AI can overcome the main issues of deep learning:

Lack of explainability having a formally defined computational semantics increases the trustworthiness of AI systems.

Lack of parsimony requiring far less data and computational power at training time drastically decreasing energy consumption.

2.2 Fuzzy Logic

In the year 1965, Lotfi Zadeh introduced the concept of fuzzy logic as a means to articulate and comprehend imprecise or vague propositions [13]. Fuzzy logic can be viewed as an expansion upon the conventional framework of Boolean logic: in classical Boolean logic, the output is binary, while in fuzzy logic considers a continuous interval, specifically the range [0, 1]. This approach enables the representation of a spectrum of truth values, thereby accommodating varying degrees of veracity. Within the realm of fuzzy logic, the extremities of this spectrum, namely 1 and 0, correspond to the utmost levels of truth and falseness, respectively, providing a nuanced and flexible framework for the analysis of imprecise data and ambiguous scenarios, such as control of complex system whose dynamic is difficult to model in a conventional approach. Hereby are reported the main definition of fuzzy logic.

Definition F1 *Let X be a space of points. A fuzzy set A in X is characterized by a membership function $\mu_A(x)$ which associates with each point $x \in X$ a real number in the interval [0, 1], with the value of $\mu_A(x)$ at x representing the grade of membership of x in A. Thus, the nearer the value of $\mu_A(x)$ to unity, the higher the grade of membership of x in A.*

Definition F2 *Given two fuzzy sets A and B, they are equal if and only if $\forall x \in X\ \mu_A(x) = \mu_B(x)$*

Definition F3 *Given two fuzzy sets A and B, they are equal if and only if $\forall x \in X\ \mu_A(x) = \mu_B(x)$*

Definition F4 *The complement of a fuzzy set A is denoted by A' and is defined by $f_{A'}(x) = 1 - \mu_A(x)$*

Definition F5 *A is contained in B if and only if $\mu_A(x) \leq \mu_B(x)$*

Definition F6 *A fuzzy set is empty if and only if its membership function is identically zero on X*

Definition F7 *The union of two fuzzy sets A and B with respective membership functions $\mu_A(x)$ and $\mu_B(x)$ is a fuzzy set C, written as $C = A \cup B$, whose membership function is related to those of A and B by $f_C(x) = \max(\mu_A(x), \mu_B(x)), x \in X$*

Definition F8 *The intersection of two fuzzy sets A and B with respective membership functions $\mu_A(x)$ and $\mu_B(x)$ is a fuzzy set C, written as $C = A \cap B$, whose membership function is related to those of A and B by $f_C(x) = \min(\mu_A(x), \mu_B(x)), x \in X$*

Definition F9 [14] *Let T be a membership function such that $T : [0,1]^2 \rightarrow [0,1]$. T is a triangular-norm (also called T-norm) if and only if is $\forall x,\ y,\ z \in [0,1]$:*

T1 $T(x,y) = T(y,x)$,

T2 $T(x,T(y,z)) = T(T(x,y),z)$,

T3 $T(x,y) \leq T(x,z)$ whenever $y \leq z$,

T4 $T(x,1) = x$.

Definition F11 [14] *Let S be a membership function such that $T : [0,1]^2 \to [0,1]$. S is a triangular-conorm (also called S-norm) if and only if is* $\forall\, x,\, y,\, z \in [0,1]$:

T1 $S(x,y) = S(y,x)$,

T2 $S(x, S(y,z)) = S(S(x,y), z)$,

T3 $S(x,y) \leq S(x,z)$ whenever $y \leq z$,

T4 $S(x,0) = x$.

Definition F10 *T-norm Łukasiewicz is defined as* $T(x,y) \coloneqq \max(0, x+y-1)$

From the above definitions it is clear that fuzzy logic permits to map each constant and formula into numerical features, which can be used in further computations. Below are summarized the main logical connector formulations using T and S-norms. Note that in this work only strong implications are used, i.e. given the symbol x, y, then $x \to y = \neg x \vee y$

Name	Operations		
	$x \wedge y$	$x \vee y$	$x \to y$
Goedel	$\min(x,y)$	$\max(x,y)$	$\max(1-x,y)$
Goguen/Product	$x \cdot y$	$x + y - x \cdot y$	$1 - x + x \cdot y$
Łukasiewicz	$\max(x+y-1{,}0)$	$\min(x+y,1)$	$\min(1-x+y,1)$

Table 2.1: Common Fuzzy Operators on T and S-norms

Name	$I(x,y)$	S-Implication
Kleene-Dienes I_{KD}	$\max(1-x,y)$	$S = S_M$ $N = N_S$
Reichenbach I_R	$1 - x + xy$	$S = S_P$ $N = N_S$
Łukasiewicz I_{Luk}	$\min(1-x+y,1)$	$S = S_L$ $N = N_S$

Table 2.2: Common Fuzzy implications $I(x,y) =$ and their classes, where N denotes the negation of the S-implication

2.3 Knowledge Graph

The concept of knowledge graphs traces back to diagrammatic knowledge representation methods, with roots dating to ancient greek, notably Aristotle, Euler and Venn. With the advent of digital computers, formal reasoning and knowledge representation shifted to computational programs. Pioneering works by Ritchens (1956) [15], Quillian (1963) [16], and Travers and Milgram (1969) [17] focused on formalizing natural language, information, and knowledge representations. These early efforts faced limitations due to limited computational resources. In the 1970s, significant developments occurred, including Minsky's introduction of frames (1974) [18], Brachman[19] and Woods formalizing semantic networks (1977, 1975) [20], and Sowa proposing conceptual graphs (1979) [21]. These works aimed to integrate formal logic with diagrammatic knowledge representations but faced challenges in providing precise semantics. Thus, it is clear to see the definition of a *knowledge graph* remains contentious; here we reported the one adopted by Hogan et Al. in [22]:

> *A graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities. By knowledge, we refer to something that is known. Such knowledge may be accumulated from external sources, or extracted from the knowledge graph itself.*

Note that knowledge may be composed of simple statements, such as *Nietzsche is a philosopher*, or quantified statements, such as *all dogs are animals*. Simple statements can be accumulated as edges in the data graph. But, when quantified statements have to be represented in Knowledge Graph a formal expression based on ontologies or rules is required. Knowledge Graph has gain a vast popularity in the neuro-symbolic community thanks to easiness of First Order Logic representation. In fact, given a triplet <subject, relationship, object> it can be translated into a edge that goes from a head entity (subject) to an tail entity (object) if there is a link (relationship) between them.

Infusion methods

Although, effectiveness of FOL translation into KG, the resulting symbolic nature lead to difficulty when manipulation is need, specially in case of complex query. To tackle this problem, several techniques have been proposed through the years, in particular Knowledge Graph Embedding methods[23], which have quickly gained a massive consent due to intrinsic simple nature. Suppose it is given a KG consisting of n entities and m relations, where facts observed are stored as a collection of triples $\mathbb{D}^+ = \{(h, r, t)\}$. Let $\mathbb{E}$ denote the set of entities and $\mathbb{L}$ the set of relations, then each triple is composed of a head entity $h \in \mathbb{E}$, a tail entity $t \in \mathbb{E}$, and a relation $r \in \mathbb{R}$ between them, e.g. (Paolo, IsMemberOf, SquadraCorseDriverless). KG embedding aims to embed entities and relations into a low-dimensional continuous vector space to simplify computations on the KG. Most of the currently available techniques use facts stored in the KG to perform the embedding task, enforcing embedding to be compatible with the facts. Typically, a KG embedding technique is generally composed of three steps:

Modelization of entity and relationship to create a suitable form representation in a continuous vector space, whose vectors (or tensors depending to the space dimension) can be either deterministic or stochastic.

Definition of a scoring function to measure the plausibility of both observed and unobserved facts.

Learning entity and relation representation is formulated as an optimization problem that maximizes the total plausibility of observed facts.

Following the classification introduced by [23], embedding techniques are clustered into two macro-groups: **Translational distance models** based on distance scoring function; **Semantic matching models** that employ similarity scoring function. In this work we focus only on *Translational distance models*, specifically on TransE[24], which is able to capture the rule of KG although its simplicity and it can be easily trained.

TransE

Given a training set S of triplets $< h, r, t >$ where h and t are entities from the set $\mathbb{E}$, and r is a relationship from the set $\mathbb{L}$, TransE learns vector representations for these entities and relationships. These representations are vectors in the real space $\mathbb{R}^k$, where k is a hyperparameter of the model. The core concept to model the functional relation induced by the r-labeled edges as translation of the embedding: $h + r \approx t$ when the triplet (h, r, t) holds, meaning that t should be a nearest neighbour to $h + l$ in the vector space; conversely, when (h, r, t) does not hold, $h + r$ to be significantly distant from t. This idea is formalized through an energy-based approach, wherein the energy associated with each triplet d(h + r, t) is defined as a dissimilarity measure, utilizing either the L1 or L2 norm. Here below, it is reported the algorithm related to the learning phase.

Algorithm 1 Learning TransE

```
Require: Training set $S = \{(h, r, t)\}$, entities and relation sets $E$ and $L$, margin
    $\gamma$, embedding dimension $k$, number of epoch $N$.
 1: Initialize $r \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $r \in L$.
 2: $\mathbf{r} \leftarrow \frac{\mathbf{r}}{k} \cdot \frac{\mathbf{r}}{k}$ for each $r \in \mathbb{L}$.
 3: $\mathbf{e} \leftarrow$ uniform(-$\frac{6}{\sqrt{k}}$, $\frac{6}{\sqrt{k}}$) for each entity $e \in \mathbb{E}$.
 4: while epoch<N do
 5:     $\mathbf{e} \leftarrow \frac{\mathbf{e}}{||e||}$ for each entity $e \in \mathbb{E}$.
 6:     $S_{\text{batch}} \leftarrow$ sample$(S, b)$                ▷ Sample a minibatch of size $b$.
 7:     $T_{\text{batch}} \leftarrow \emptyset$                  ▷ Initialize the set of pairs of triplets.
 8:     for $(h, r, t) \in S_{\text{batch}}$ do
 9:         $(h', r, t') \leftarrow$ sample$(S(h, r, t))$         ▷ Sample a corrupted triplet.
10:         $T_{\text{batch}} \leftarrow T_{\text{batch}} \cup \{(h, r, t), (h_0, r, t_0)\}$
11:     end for
12:     Update embeddings w.r.t.
            $$\sum_{(h,r,t),(h',r,t') \in T_{\text{batch}}} [\nabla\gamma + d(\mathbf{h} + \mathbf{r}, \mathbf{t}) - d(\mathbf{h}' + \mathbf{r}, \mathbf{t}')]_+$$
13:     epoch $\leftarrow$ epoch + 1
14: end while
```

2.4 Logic Tensor Network

Firstly introduced by Serafini and Garcez[25], logic tensor networks integrate FOL into a differentiable framework based on Real Logic to ease employment of FOL in deep learning architectures.

2.4.1 Real Logic

Real Logic is based on first order language $\mathcal{L}$ whose signature is composed of set $\mathcal{C}$ of constant symbols, a set $\mathcal{F}$ of functional symbols, a set $\mathcal{P}$ of predicate symbols and a set $\mathcal{X}$ of variable symbols. Real Logic uses the sentences of $\mathcal{L}$ to express relational knowledge based on fuzzy connectors and it interprets (grounds, see definition below) them as tuples of real numbers.

Definition R1 *Let $n > 0$. A grounding $\mathcal{G}$ for a first order language $\mathcal{L}$ is a function from the signature of $\mathcal{L}$ to the real numbers that satisfies the following conditions*:

1. $\mathcal{G}(c) \in \mathbb{R}^n$ for every constant symbol $c \in \mathcal{C}$.

2. $\mathcal{G}(f) \in \mathbb{R}^{n \cdot m} \longrightarrow \mathbb{R}^n$ for every $f \in \mathcal{F}$ and m is the arity of f.

3. $\mathcal{G}(P) \in \mathbb{R}^{n \cdot m} \longrightarrow [0,1]$ for every $P \in \mathcal{P}$ and m is the arity of P.

A novelty introduced by Real Logic is the aggregation semantics (see *Definition R2*) that aggregates the several statements (axioms) formulated. Several aggregators for universal and existential quantification respectively based on T and S-norms can be found in literature, in this works the ones used are reported in Table 2.3.

Definition R2 *Let $\phi(\mathbf{x})$ be a clause of $\mathcal{L}$ that contains a k-tuple $\mathbf{x} = \langle x_1, \ldots, x_k \rangle$ of k distinct variables. Let T be the set of all k-tuples $\mathbf{t} = \langle t_1, \ldots, t_k \rangle$ of closed terms of $\mathcal{L}$, then $\mathcal{G}(\phi(\mathbf{x})) = \alpha_{\mathbf{t} \in T} \mathcal{G}(\phi(\mathbf{t}))$, where α is an aggregation operator from* $[0,1]^{|T|} \rightarrow [0,1]$

Definition R3 *Let ϕ be a clause in $\mathcal{L}$, $\mathcal{G}$ a grounding, and $v \leq w \in [0,1]$. We say that $\mathcal{G}$ satisfies ϕ in the confidence interval $[v, w]$, written $\mathcal{G} \models_v^w \phi$, if $\mathcal{G}(\phi) \in [v, w]$. A partial grounding, denoted by $\hat{\mathcal{G}}$, is a grounding that is defined on a subset of the signature of $\mathcal{L}$. A grounding $\mathcal{G}$ is said to be an extension of a partial grounding $\hat{\mathcal{G}}$ if $\mathcal{G}$ coincides with $\hat{\mathcal{G}}$ on the symbols where $\hat{\mathcal{G}}$ is defined.*

Definition R4 *A grounded theory or knowledge base is a pair $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ where $\mathcal{K}$ is a set of pairs $\langle [v, w], \phi(\mathbf{x}) \rangle$, where $\phi(\mathbf{x})$ is a clause of $\mathcal{L}$ containing the set $\mathbf{x}$ of free variables, and $[v, w] \subseteq [0,1]$, and $\hat{\mathcal{G}}$ is a partial grounding.*

Definition R5 *A grounded theory $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is satisfiable if there exists a grounding $\mathcal{G}$, which extends $\hat{\mathcal{G}}$ such that for all $\langle [v, w], \phi(\mathbf{x}) \rangle \in \mathcal{K}, \mathcal{G} \models_v^w \phi(\mathbf{x})$.*

From the above definitions it easy to understand that constraints' satifiability cannot be just checked using *Definition R5* since its computing power is finite. Thus, a partial sasfiability should be verified limiting to regulars groundings and number of clause instantiations. Then, the best grounding is the one that minimizes the satisfiability error on all the clauses in the knowledge base.

Type	Definition
Minimum T-norm	$A_{T_M}(x_1,\dots,x_n) \coloneqq \min(x_1,\dots,x_n)$
Probabilistic Sum T-norm	$A_{T_P}(x_1,\dots,x_n) \coloneqq \prod_{i=1}^{n} x_i$
Łukasiewicz T-norm	$A_{T_L}(x_1,\dots,x_n) \coloneqq \max\left(\sum_{i=1}^{n} x_i - n + 1{,}0\right)$
Minimum S-norm	$A_{S_M}(x_1,\dots,x_n) \coloneqq \max(x_1,\dots,x_n)$
Probabilistic Sum S-norm	$A_{S_P}(x_1,\dots,x_n) \coloneqq 1 - \prod_{i=1}^{n}(1 - x_i)$
Łukasiewicz S-norm	$A_{S_L}(x_1,\dots,x_n) \coloneqq \min\left(\sum_{i=1}^{n} x_i, 1\right)$

Table 2.3: Fuzzy Aggregator extending T and S-norms

2.4.2 Methodology

Currently, LTN methods can be classified as

Trainer module whose functionality is limited to verification of constraints satisfability during training, in particular for image task relevant works are[26, 27, 28].

Refinement layer where additional layers inject knowledge bases directly into target network[29].

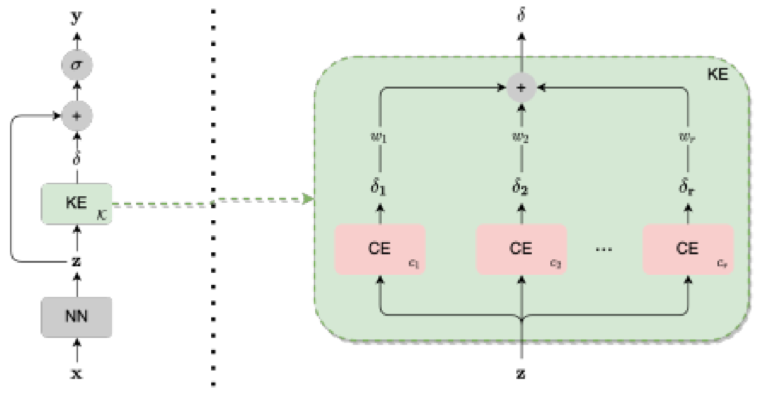


Figure 2.1: KENN: a refinement layer known as Knowledge Enhancer (KE) integrates symbolic knowledge from a clause into the Clause Enhancer module using hyperparameters δ_i and w_i

Chapter 3

Review of the state-of-art

SGG's architecture can be clustered into two main approach: two-stage approach, where the object and relationship detection modules are trained separately; one-stage approach, where object and relationship detection modules are jointly trained. Concerning the relationship detection module, SOTA architecture are based on Conditional Random Field, Graph Convolutional neural Networks (GCN), Recurrent Neural Network (RNN) and Transformers. Due to the complexity of the visual relationship detection task, datasets present several issues. In particular, the main ones are missing annotations and long-tailed relationship distribution. In literature, the methodologies used to solve these issues can be divided into 4 main categories:

Statistical Learning Approach reduces bias by looking on the dataset distribution and exploiting techniques from Statistical Learning such as Conditional Random Field.

External Knowledge integration reduces bias by integrating knowledge from external source, such as Knowledge Graph Embedding, Distillation Knowledge or Visual-linguist Prior.

Novel Losses integration to account the common issues of SGG.

Tailored Message Parsing to improve current message parsing in GCN-based network to enhance relationship predictions.

3.1 Base Architecture

3.1.1 Conditional Random Field

Within the realm of visual relationship triples, denoted as $\langle s - r - o \rangle$, a notable and robust statistical correlation emerges between the relationship predicate and

the object pair. Leveraging this valuable statistical insight holds immense potential in enhancing the accuracy and efficiency of visual relationship recognition.
One of the established methodologies for harnessing these statistical relationships is the Conditional Random Field(CRF)[30]. The CRF offers a sophisticated modelisation method for effectively incorporating statistical relationships into the task of discrimination. In the context of visual relationship, the CRF can be represented as follows[31]:

$$P\left(r_{s \rightarrow o} \mid X\right)=\frac{1}{Z} \exp \left(\Phi\left(r_{s \rightarrow o} \mid X ; W\right)\right), X=x_r, x_s, x_o \tag{3.1}$$

where x_r denotes appearance features and spatial configuration of the object pair, while x_s and x_o denote the appearance features of the subject and object respectively. W is the parameter of the model, Z is a normalization constant and F represents the joint potential.

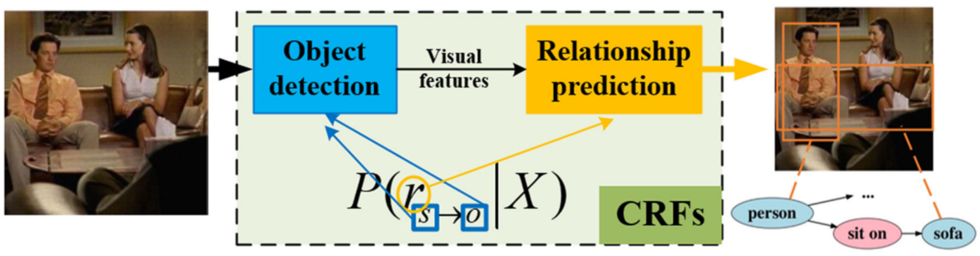


Figure 3.1: The basic structure of CRF-based SGG models[31] has: object detection (s and o) and relationship prediction (r) module

3.1.2 Graph Convolutional Neural Network

The scene graph can be modelled as graph, thus a straightforward approach would be improving the SGG task based on the graph theory, in particular Graph Convolutional Neural Network[32]. Let $\mathcal{G}$ be a graph $\mathcal{G}=(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the node and $\mathcal{E}$ the relationship between them. Each node has a number of predefined features X_i, which are all stacked in a matrix X. Let be adjacency matrix $A \in \mathbb{R}^{N \times N}$, D_{ii} degree matrix and Laplacian Δ of $\mathcal{G}$, then Graph Convolutional Network $Z=f(X, A)$ composed of l-layers has the layer-wise propagation rule:

$$H^{(l+1)}=\sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$$

where $\tilde{A}=A+I_N$ represent the adjacency matrix of the graph $\mathcal{G}$ with added self-connections represented by the identity matrix I_N, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $W^{(l)}$ is trainable weight matrix of the l-layer and $\sigma(\cdot)$ denotes an activation

function. Recall that $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is the spectral graph convolution. In Fig.3.2 is possible to view such schema.
SGG process based on GCN can be factorized into three parts:

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

where V is the set of nodes (objects in images), E is the edges(relationships between objects) in a graph. Based on this basic conditional form, several methods optimize the terms of $P(E|V, I)$ and $P(R, O|V, E, I)$ by designing relevant modules, and GCN-based networks are designed for the graph labeling process $P(< V, E, O, R > |I)$[31].

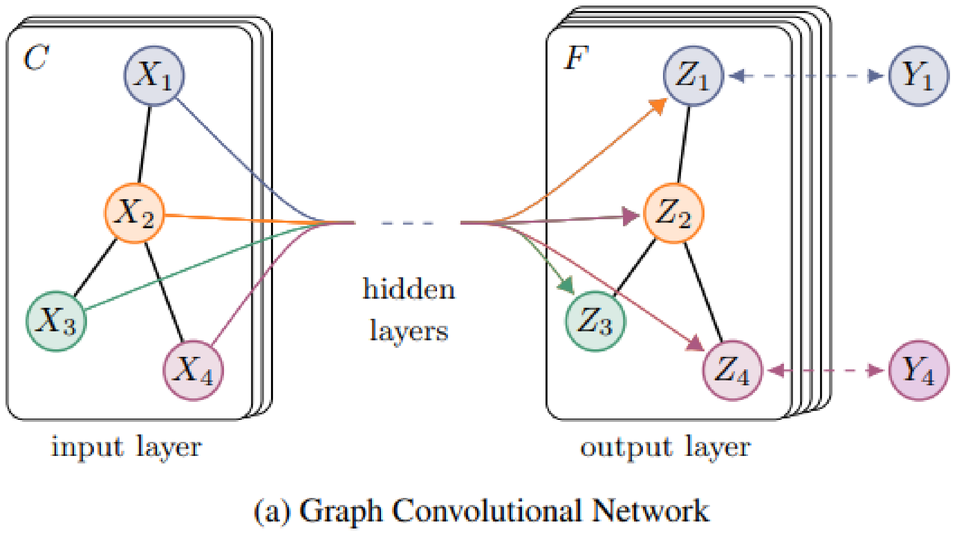


Figure 3.2: The basic structure of GCN, where Y_i, X_i, Z_i are respectively the labels, the inputs and output nodes of the network

3.1.3 Recurrent Neural Networks

RNNs[33] are a class of neural networks specifically designed for sequential data processing. They are characterized by their ability to maintain hidden state information, which is updated at each time step and depends on previous time steps. A RNN consists of an input sequence $\mathbf{X} = (x_1, x_2, \dots, x_t)$, hidden state $\mathbf{h}_t$ updated recurrently and an output sequence $\mathbf{Y} = (y_1, y_2, \dots, y_t)$. Here, the hidden state update formula is reported

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t; \theta) \tag{3.3}$$

RNNs suffer from the vanishing gradient problem, making it challenging to capture long-range dependencies in sequences. To tackle this issue, several improvement such as **LSTM** (Long Short-Term Memory)[34], a variant with gated cells designed to capture long-term dependencies, **GRU** (Gated Recurrent Unit)[35], a simplified version of LSTM with fewer gates.

3.1.4 Transformer

Firstly, introduced in 2017 by Vaswani et al.[36] the Transformer revolutionized the field of Natural Language Processing (NLP) by addressing the limitations of RNN and CNNs: a notable drawback of RNNs is their reliance on sequential processing, which often leads to the loss of important information across lengthy sequences, while CNNs are not capable of retaining temporal information of neural network state; Transformers solve this issue by introducing into a encoder-decoder architecture the concept self-attention modelled as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.4}$$

where Q represents queries from the decoder, while K and V are keys and values derived from the encoder's output. The dimensions of keys and queries are denoted as d_k and the values have dimension d_v. The multihead attention mechanism, which connects the encoder and the decoder, is an aggregation of multiple attention heads:

$$\text{multihead}(Q, K, V) = \text{concat}(\text{head}_1, \ldots, \text{head}_h)W_o \tag{3.5}$$

Each individual attention head $head_i$, is computed using Equation 3.4 with query projections W_i^Q, key projections W_i^K and value projections W_i^V. The multihead attention layer performs computations invariant to the sequence position, loosing one of the major benefit common both to CNNs and RNNs. To address this issue a positional encoding is added after the embedding layers. Sine and cosine functions, as defined in below equations, are employed for positional encoding. These functions vary with the position in the sequence and provide a unique value for each word's position, which is then added to the embedding layer's output.

$$PE(pos{,}2i) = \sin\left(\frac{pos}{10000 \cdot 2^{\left(\frac{i}{\text{dmodel}}\right)}}\right) \tag{3.6}$$

$$PE(pos{,}2i+1) = \cos\left(\frac{pos}{10000 \cdot 2^{\left(\frac{i}{\text{dmodel}}\right)}}\right) \tag{3.7}$$

where *pos* is the position in the sequence and i is the i-th feature in the embedding. Since the wavelengths form a geometric progression from 2π to 2000π each feature of every word extracted according to the embedding layer is mapped into a different

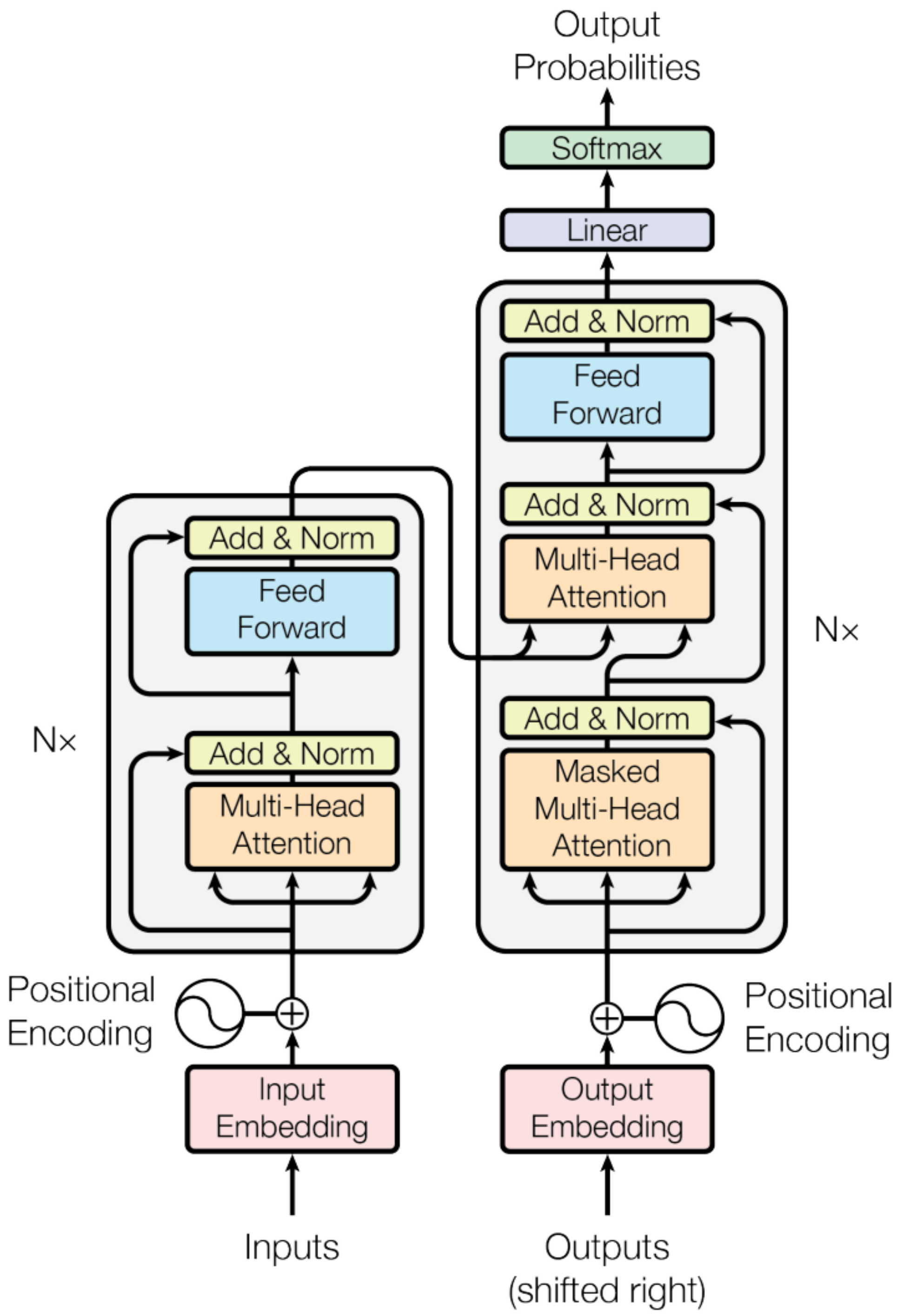


Figure 3.3: The basic structure of Transformer

value. Due to the high information volume captured by transformers, they can achieve SOTA performance both for two and one-stage approaches.

3.1.5 Baseline networks

Here are reviewed the SGG networks, whose performance on Traffic Genome dataset is available in the literature.

MotifNet

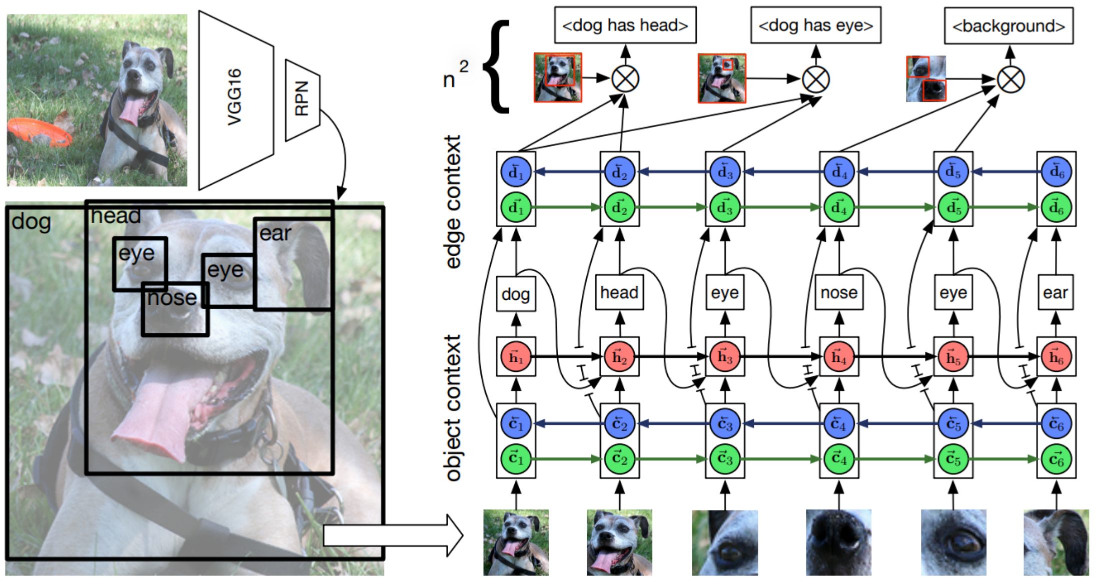


Figure 3.4: The structure of MotifNet

In Fig.3.4 is it possible to visualized the general schema of MotifNet. MotifNet is the first method that tries to solve long-tailed distribution problem. It decomposes the probability of a graph $\mathcal{G}$ (made up of a set of bounding regions $\mathcal{B}$, object labels $\mathcal{O}$, and labeled relations $\mathcal{R}$) into three factors:

$$P(\mathcal{G}|I) = P(\mathcal{B}|I)P(\mathcal{O}|\mathcal{B}, I)P(\mathcal{R}|\mathcal{B}, \mathcal{O}, I) \tag{3.8}$$

The bounding box model ($P(\mathcal{B}|I)$) is a standard object detection model(Faster R-CNN[37]), which retrieves also feature vector $\mathbf{f}_i$ and non-contextualized vector $\mathbf{l}_i$ of object label probabilities for bounding box $\mathbf{o}_i$. The object model ($P(\mathcal{O}|\mathcal{B}, I)$) conditions on a potentially large set of predicted bounding boxes, $\mathcal{B}$. Thus, object detector outputs is linearized into a sequence $[(b_1, f_1, l_1), \ldots, (b_n, f_n, l_n)]$ that an

LSTM can process to create a contextualized representation C of each box using the following equation:

$$C = \mathrm{biLSTM}\left([\mathbf{f}_i; \mathbf{W}_1 l_i]_{i=1,\ldots,n}\right) \tag{3.9}$$

$C = [c_1, \ldots, c_n]$ contains the final LSTM layer's hidden states for each element in the linearization of $\mathcal{B}$, and $\mathbf{W}_1$ is a parameter matrix that maps the distribution of predicted classes, l_i, to $\mathbb{R}^{100}$. This structure allows all elements of $\mathcal{B}$ to contribute information about potential object identities. Then, the context is used to decode the labels according to the previous labels:

$$h_i = \mathrm{LSTM}_i([c_i; \hat{o}_{i-1}]) \tag{3.10}$$

$$\hat{o}_i = \mathrm{argmax}(W_o h_i) \in \mathbb{R}^{|C|}(\text{one-hot}) \tag{3.11}$$

Only the object class commitments $\hat{o}_i$ are used in the relation model (Section 4.3). During the relations modeling ($P(\mathcal{R}|\mathcal{B}, \mathcal{O}, I)$), the set of predicted labeled objects $\mathcal{O}$ is linearized again and processed with another LSTM to create a representation of each object in context:

$$D = \mathrm{biLSTM}([c_i; \mathbf{W}_2 \hat{o}_i]_{i=1,\ldots,n}) \tag{3.12}$$

where the edge context $D = [d_1, \ldots, d_n]$ contains the states for each bounding region at the final layer, and $\mathbf{W}_2$ is a parameter matrix mapping $\hat{o}_i$ into $\mathbb{R}^{100}$. For each possible edge, let say between b_i and b_j, it is computed the probability the edge will have label $x_{i\to j}$ (including BG). The distribution uses global context, D, and a feature vector for the union of boxes, $f_{i,j}$:

$$g_{i,j} = (W_h d_i) \odot (W_t d_j) \odot f_{i,j} \tag{3.13}$$
$$P(x_{i\to j}|\mathcal{B}, \mathcal{O}) = \mathrm{softmax}(W_r g_{i,j} + w_{o_{i,j}}) \tag{3.14}$$

W_h and W_t project the head and tail context into $\mathbb{R}^{4096}$, $w_{o_{i,j}}$ is a bias vector specific to the head and tail label.

VCTree

Introduced by Tang et al.[38] VCTree aims to learn a score matrix S, which approximates the task-dependent validity between each object pair by memorizing the object correlation by all pair of objects. Each element of the S matrix is formulated to be complaint with the following principles:

1. Inherent object correlations should be maintained.
2. Task related object pairs have higher score than irrelevant ones.

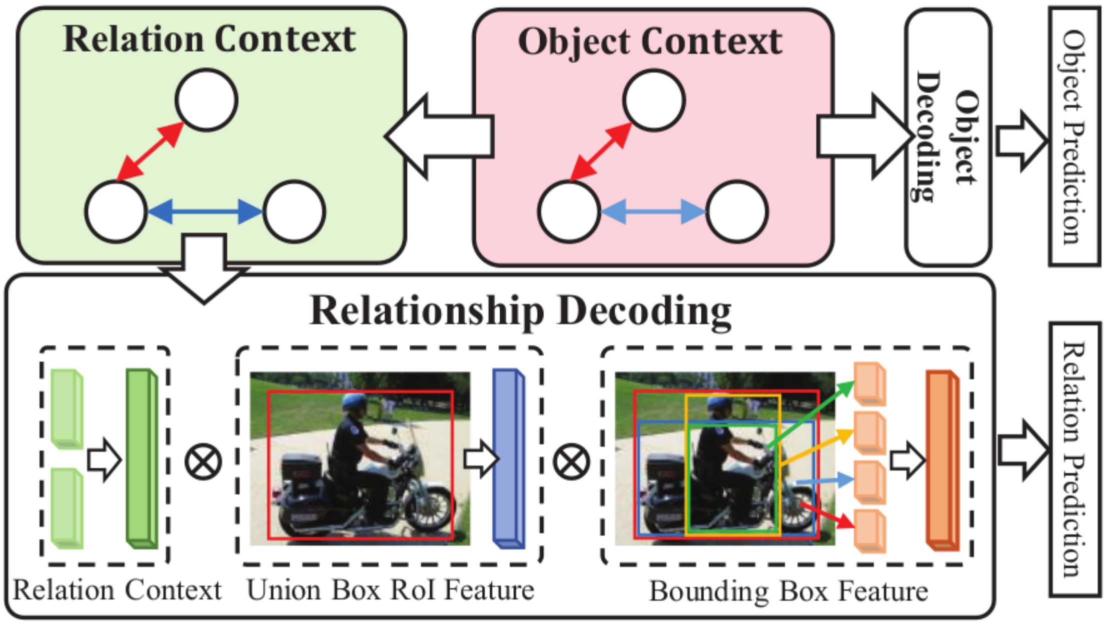


Figure 3.5: The structure of BiTreeLSTM used in VCTree

Therefore, each element of **S** is defined as follows

$$\begin{cases} \boldsymbol{S}_{ij} = f\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \cdot g\left(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{q}\right), \\ f\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sigma\left(\mathrm{MLP}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)\right), \\ g\left(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{q}\right) = \sigma\left(h\left(\boldsymbol{x}_i, \boldsymbol{q}\right)\right) \cdot \sigma\left(h\left(\boldsymbol{x}_j, \boldsymbol{q}\right)\right), \end{cases} \quad - \tag{3.15}$$

where $f\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ is the product of the object correlation and $g\left(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{q}\right)$ is the pairwise task-dependency, $\sigma(\cdot)$ is the sigmoid function and $\boldsymbol{q}$ is the task feature. The overall **S** matrix is assumed to be symmetric, then the maximum spanning tree is obtained using the Prim's algorithm. Note that the object detection is done before the construction of VCTree. Once that VCTree has been constructed, a BiTreeLSTM[39] is adopted as context encoder-decoder to learn object and relation context as summarized in the Figure 3.5.

Iterative Message Parsing (IMP)

Introduced by Xu et. al[40], IMP is one of the earliest example that employs Message Parsing in SGG to capture contextual information. Let be $\mathcal{B}_I$ the set of object bounding box proposals from the image I detected by Region Proposal Network, then for each $\mathcal{B}_I$ are associated three types of variables, that must be inferred, which are:

1. x_i^{cls} the object class label.

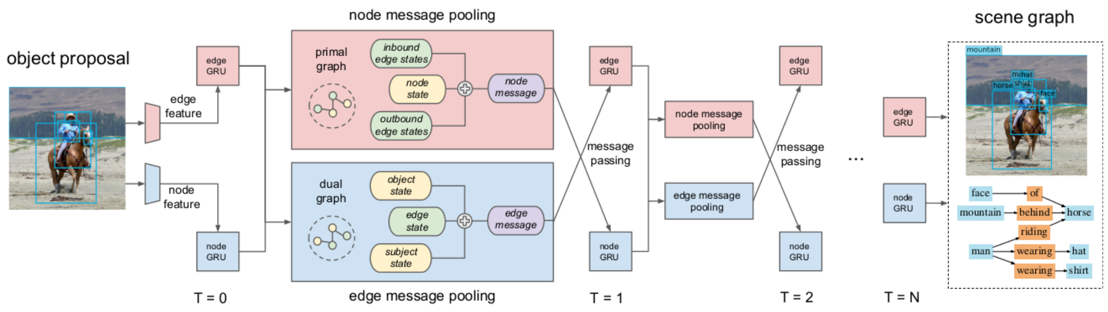


Figure 3.6: IMP schema

2. x_i^{bbox} the box coordinates.

3. $x_{i\to j}$ the relationship between all possible pairing of boxes

Let x done the set of all the aforementioned variables, then the solution of SGG problem can be formalized as optimal $x^* = \arg\max_x P(x|I, \mathcal{B}_I)$. Let be $Q(x|\cdot)$ the probability of each variable x, which at each iteration depends on the current state (denoted by h_i) of nodes and edges, Then, modeling the probability by means of mean field, the overall expression is:

$$Q(x|I, B_I) = \prod_{i=1}^{n} Q(x_i^{\text{cls}}, x_i^{\text{bbox}}|h_i)Q(h_i|f_i^v) \prod_{i=1}^{n} Q(x_{i\to j}|h_{i\to j})Q(h_{i\to j}|f_{i\to j}^e) \tag{3.16}$$

where f_i^v and $f_{i\to j}$ are respectively the visual features of i_{th} node and the edge from the i-th node to the j-th node. Note that this distribution is learnt through the GRUs. Message parsing is employed as follows: firstly, the visual features of nodes(f^v) and edges(f^e) are passed to the GRU units(see Figure 3.6); then, each time a new GRU hidden state is formulated based on incoming messages and the previous hidden state. GRU's weights are shared only between the same set, i.e. the GRUs nodes and GRUs edges. The overall structure creates a bipartite graph, where each subgraphs is the dual of others: the primal graph, the set of nodes, creates messages from edge GRUs to be injected into node GRUs, while the dual graph, the set of edges defines channels, does the opposite. Finally, an innovative message pooling method is introduced to dynamically combine the hidden states of the GRUs into messages through a process of iterative weight updates.

3.1.6 Prior Knowledge as a solver for biased predictions

Here are summarized the the most relevant techniques employed in SGG to tackle unbiased prediction due to tailed-distribution.

3.1.7 Total Direct Effect

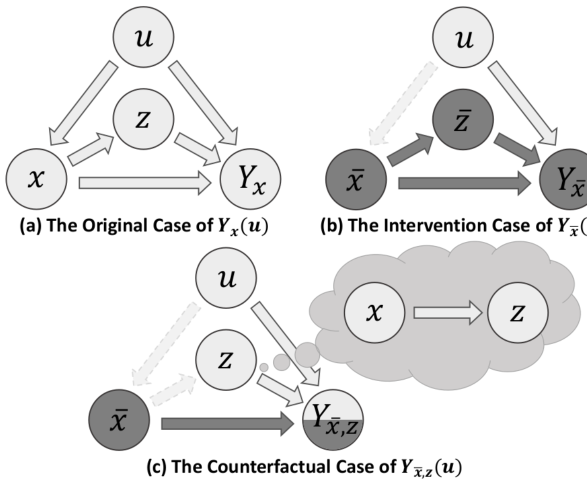

u
z
x
Y_x
(a) The Original Case of $Y_x(u)$
u
$\bar{z}$
$\bar{x}$
$Y_{\bar{x}}$
(b) The Intervention Case of $Y_{\bar{x}}(u)$
u
z
$\bar{x}$
$Y_{\bar{x},z}$
x
z
(c) The Counterfactual Case of $Y_{\bar{x},z}(u)$

Figure 3.7: Schema of the TDE approach, where $Y_{\bar{x},z}(u) = Y(\text{do}(X = \bar{x})|u)$

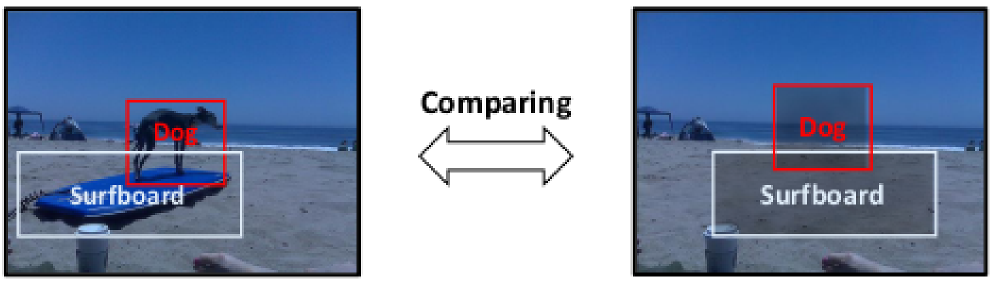


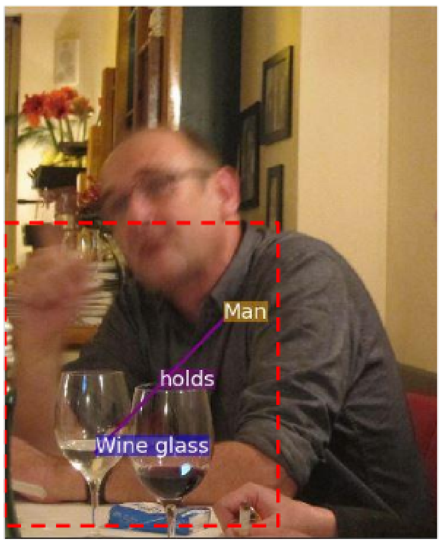
Figure 3.8: Example of Counterfactual Thinking: the visual features of dog and surfboard are removed

To address the issue of biased predicates, Tang et al.[41] introduced an innovative unbiased approach called Total Direct Effect(TDE). The idea behind this model is rooted in the concept of counterfactual causality[42]. Initially, the model retrieves biased predicate predictions using the original scene, which is defined as factual scene; subsequently, the visual features of the objects in the image are removed (this intervention is denoted as $do(\cdot)$) to generate a new scene, the counterfactual

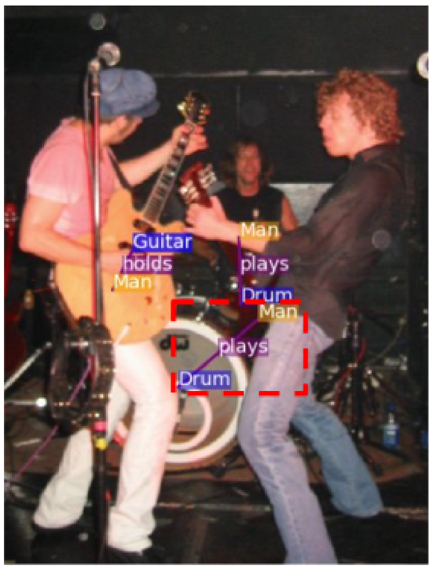
scene, and its related predicate predictions as shown in Figure 3.8. The difference between these predictions aims to remove the biased prediction of predicates, and these resulting logits are the one employed by SGG methods. Thus, TDE can be formalized as follows:
Let be u the scene graph of an image I, x the visual features of a specific object in I and z its object class; then given Y the logits of u and $\bar{x}$ resulting variable after the intervention on x, the unbiased logits $Y_{xe}(u)$ is

$$Y_{xe} = Y_x(u) - Y(\mathrm{do}(X = \bar{x})|u) \tag{3.17}$$

3.1.8 Graphical Contrastive Losses



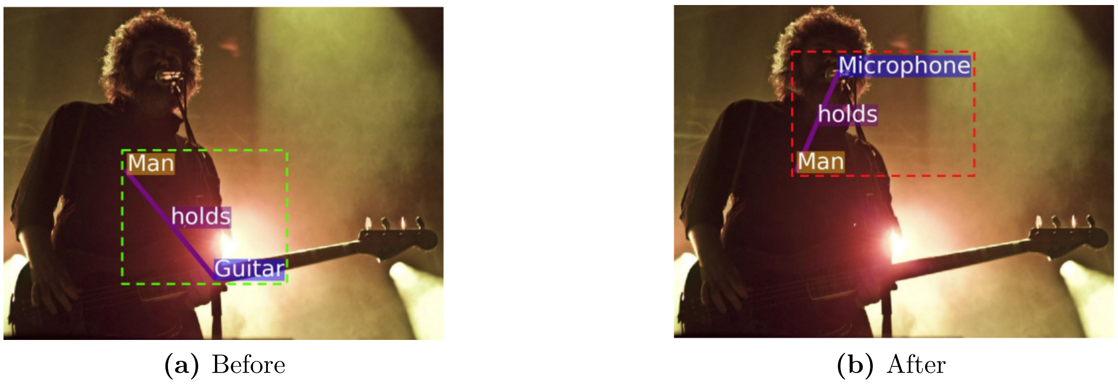
(a) Entity Instance Confusion



(b) Proximal Relationship Ambiguity

Figure 3.9: Two-stage SGG: common issues

In scene graph parsing, given an image, the goal is to deduce a graph with entity categories and their pairwise relationships. Typically, this involves detecting $\langle subject, predicate, object\rangle$ triplets, such as $\langle man, holds, guitar\rangle$ in Figure 3.10. Two-stage approach models often encounter two main issues. The first is **Entity Instance Confusion**, where the model struggles to distinguish between instances of the same class when the subject or object is related to one of many such instances. An example is shown in 3.9a, where the model correctly identifies the man holding a wine glass but fails to determine which of the three visually similar wine glasses he is holding. The second type of error is **Proximal Relationship Ambiguity**,



(a) Before **(b)** After

Figure 3.10: SGG before and after Graph Constrastive Losses

which arises when the image contains multiple subject-object pairs with similar interactions, and the model cannot identify the correct pairing (see Figure 3.9b as an example). [43] proposes novel Graphical Contrastive Losses to address the two aforementioned issues:

Class Agnostic Loss contrasts positive and negative entity pairs regardless of their relation.

Entity Class Aware Loss deals with entity instance confusion by maximizing the margins between instances of the same entity class, i.e. subject or object.

Predicate Class Aware Loss deals with proximal relationship ambiguity by focusing on entity pairs with the same potential predicate.

Note that the authors define the affinity for the Constrastive Learning term as:

$$\Phi(s, o) = 1 - p(pred = \varnothing | s, o) \tag{3.18}$$

where $p(pred|s, o)$ is output distribution over predicate classes conditioned on a subject (s) and object (o) pair and $\varnothing$ is the class symbol representing no relationship. In figure 3.10 it is possible to see a graphical outcome from this approach.

3.1.9 Label Semantic Knowledge Distillation

Most existing SGG models adopt a common training approach, treating object and predicate classification as single-label tasks with one-hot target labels. This training paradigm overlooks two key dataset characteristics:

1. Positive samples may involve multiple reasonable predicates for specific subject-object instances.

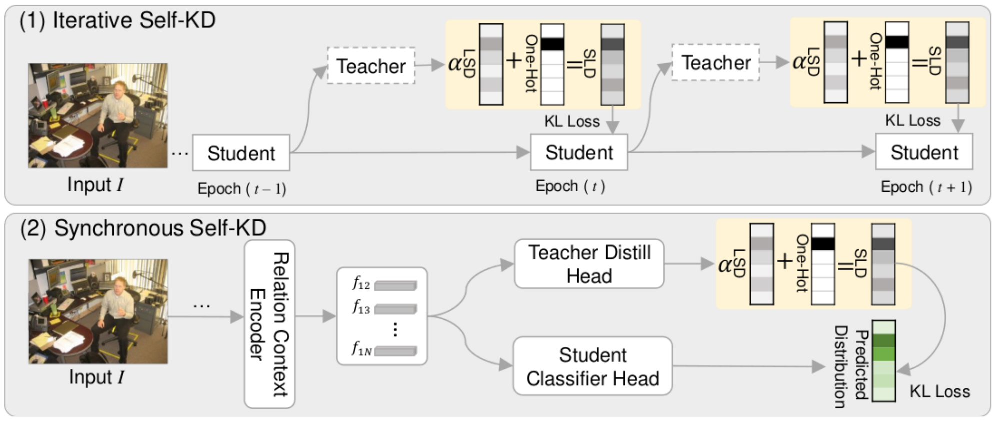


Figure 3.11: The pipeline of two self-Knowledge Distillation learning strategies. When training in t epoch, student model trained at $t-1$ epoch becomes the teacher model. Dotted lines indicate that the model weights are frozen, while f are the object features

2. Negative samples often lack annotations.

Despite these challenges, SGG models are prone to confusion and errors. To address this, Li at al.[44] introduce a novel model-agnostic method called Label Semantic Knowledge Distillation (LS-KD) for unbiased SGG(see Figure 3.11). LS-KD dynamically generates *soft labels* for subject-object instances by merging predicted Label Semantic Distributions (LSD) with original one-hot labels and uses a Softmax function to create Simulated Label Distributions (SLD). The SLD replaces one-hot vectors as soft target labels for relation classifier training (see Figure 3.12. LSD captures correlations between instances and multiple predicate categories. Besides, two self-KD strategies are proposed: **iterative self-KD**, where the student model becomes its own teacher iteratively to utilize past predictions for informative supervision, and **synchronous self-KD**, where the student and teacher models share the same relation encoder but employ two pseudo-siamese classifier heads.

3.1.10 Relation-aware message parsing

Recent SGG frameworks focus on learning complex object relationships in images. Message Passing Neural Networks (MPNNs) are key representation modules due to their modeling of high-order interactions. However, existing MPNN-based models treat scene graphs as homogeneous graphs, limiting context awareness for visual relations. Homogeneous graphs treat all nodes and edges as a single type, causing relations to strongly depend on associated objects. For example, in the triplet

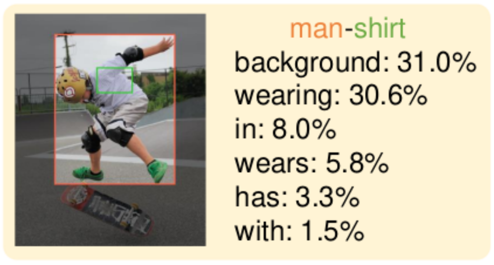


Figure 3.12: An example of the LSD of a missing-annotated triplet

⟨kid, riding, elephant⟩, an elephant does not usually ride a kid because *Human* typically rides *Animal*. Yool et al. propose[45] an unbiased Heterogeneous Scene Graph Generation(HetSGG) framework using Relation-aware Message Passing neural networks (RMP) to capture context by considering predicate types (see Figure 3.13). RMP treats each relation differently, using relation type-specific

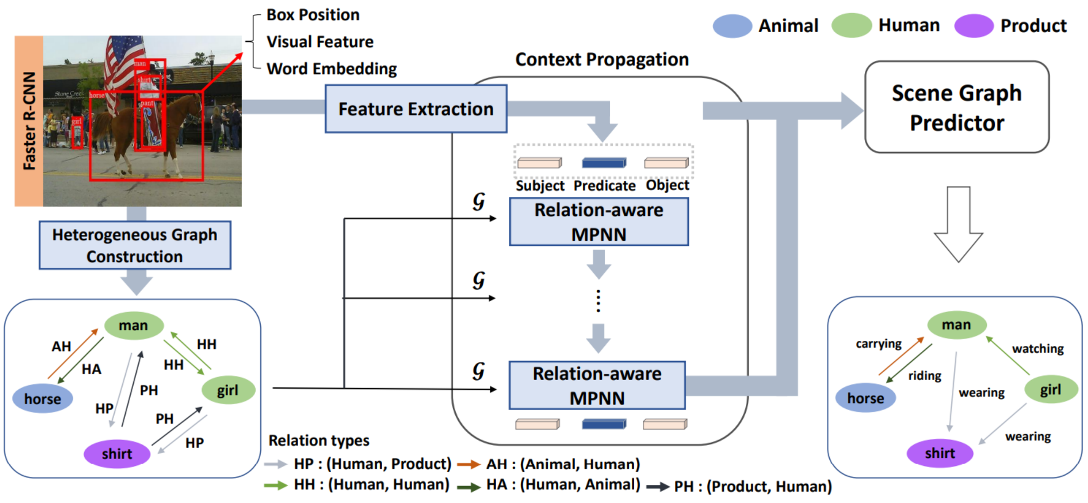


Figure 3.13: HetSGG schema: Given an image, a heterogeneous graph is constructed based on the objects detected by an object detector, i.e. Faster R-CNN, from which feature vectors for objects and predicates are extracted. RMP propagates relation-aware messages to the representations of objects and predicates. Finally, the scene graph predictor generates a heterogeneous scene graph

projection matrices. Its aim is to capture relation-aware context, updating object and relation representations. It comprises two steps:

Edge-wise update for relations where RMP generates relation-specific messages to refine relation representations.

Node-wise update for objects where RMP aggregates messages based on relation types (intra-relation aggregation) and combines relation type-specific object representations to obtain final object representations.

Visual Linguistic Prior into Transformer

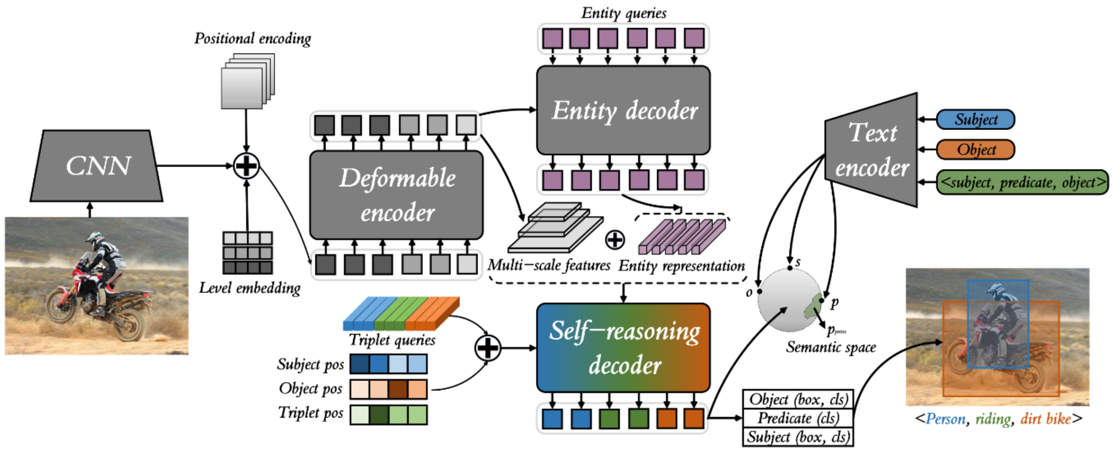


Figure 3.14: SrTR schema

One-stage scene graph generation methods are highly efficient, inferring effective relations using sparse proposals and few queries. However, they often overlook subject-predicate and predicate-object relationships and lack self-reasoning abilities. Furthermore, they tend to neglect linguistic modality knowledge, which is crucial for enhancing reasoning capabilities. To address these limitations, Zhang et al. [46] propose a Self-reasoning Transformer with Visual-linguistic Knowledge (SrTR). SrTR consists of three components:

Backbone and Entity Decoder encodes multi-scale visual context, denoted as M_e, using a CNN and Deformable DETR encoder[47]. It is then delivered to the Deformable DETR decoder, which interacts with the entity query Q_e to produce entity representations H_e and their corresponding bounding boxes B_e.

Self-reasoning Decoder to obtain multi-scale entity visual features, the bounding boxes B_e are mapped back to multi-scale space. Triplet queries, initiated through cross-attention with entity visual features F_{ea_e} to embed visual and location information, are then split into subject query Q_s, predicate query Q_p, and object query Q_o. These queries are input into a self-reasoning decoder for two-by-one self-reasoning training.

Visual-linguistic Alignment the class names of subjects and objects, along with the triplets they form with predicates, create a semantic space with visual-linguistic priors using a CLIP encoder[48]. A visual-linguistic alignment strategy is designed to map the triplet representation to the semantic space, facilitating semantic prior auxiliary relational reasoning.

Chapter 4

Datasets

In Scene Graph Generation the main benchmark datasets are Visual Relationship Detection[11] and Visual Genome[4]. None of those have sufficient labels to be employed in Autonomous Driving scenario and there is a significant lack of spatial relationships; all these issues lead to the creation of another benchmark dataset, Traffic Genome[10]. PandaSet is chosen as KG of TranSE due its richness and variety of information in the Autonomous Driving domain.

4.0.1 Knowledge Graph

PandaSet

PandaSet[49] is a multimodal dataset featuring high-precision sensors with a 360-degree field of view. It is the world's largest open-source dataset to include both mechanical spinning and forward-facing LiDARs and cameras. The dataset offers 28 annotation classes and 37 semantic segmentation labels, meticulously labeled through multi-sensor fusion. In Figures 4.2, 4.3 and 4.4 it is possible to see the label distribution. PandaSet captures complex urban driving environments, including various challenges like traffic, pedestrians, construction zones, and changing lighting conditions with the aim to cover all driving conditions necessary to reach SAE Level 4 and 5. Its variety of information makes it suitable to generated a Knowledge Graph in the Autonomous Driving domain.

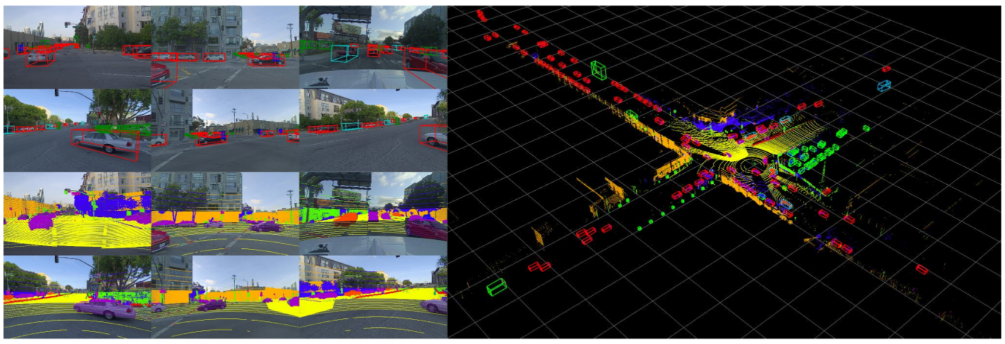


Figure 4.1: A sample from the PandaSet dataset. Left: Camera images with multimodal projection retrieved by the LiDAR's pointcloud. Right: Point cloud from sensor fusion of 2 LiDARs

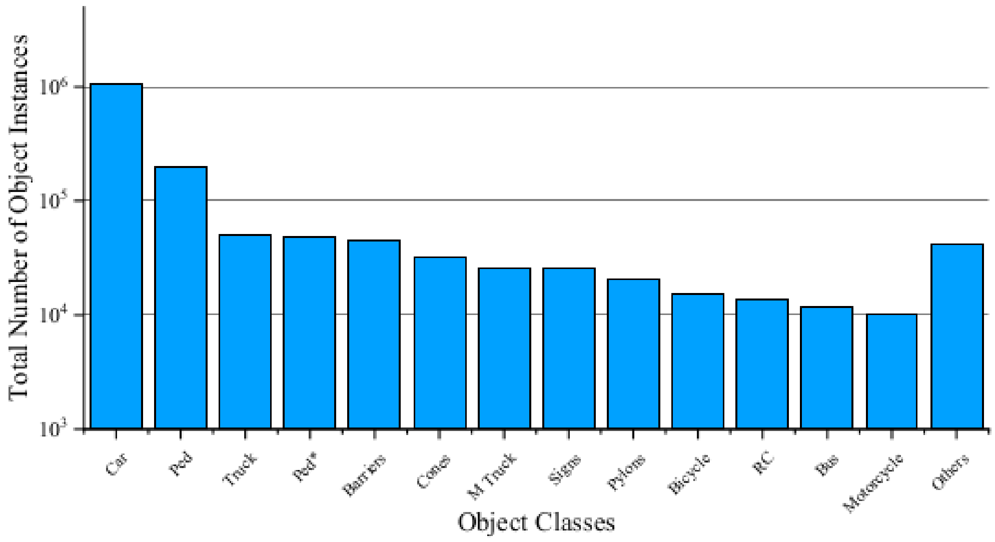


Figure 4.2: Total number of object per class in PandaSet. Note that Ped = Pedestrian, while Ped* = Pedestrian with object, M Truck = Medium-Sized Truck; RC = Rolling Container

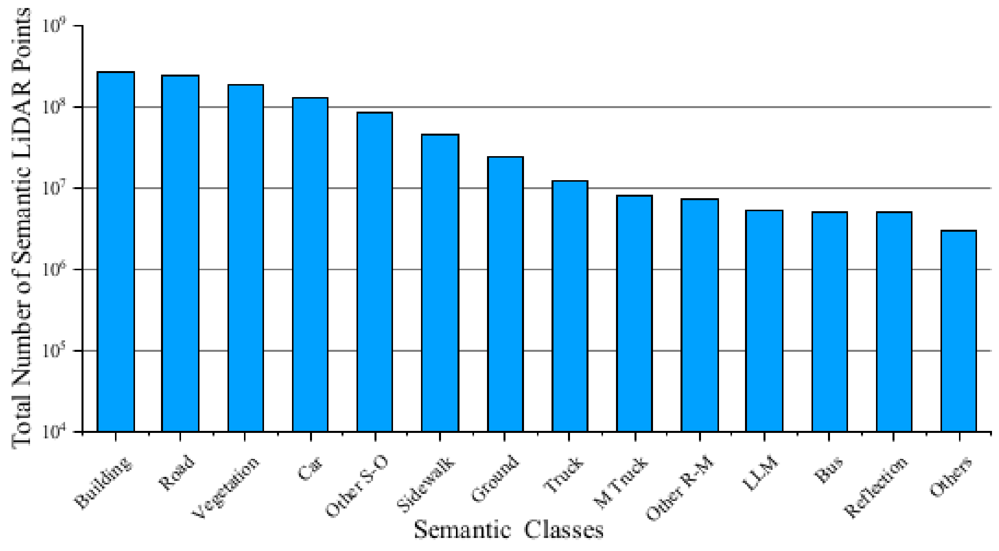


Figure 4.3: Total number of LiDAR points for each semantic segmentation class in PandaSet. Other S-O = Other Static Object; M Truck = Medium-Sized Truck; Other R-M = Other Road Marking, LLM = Lane Line Marking

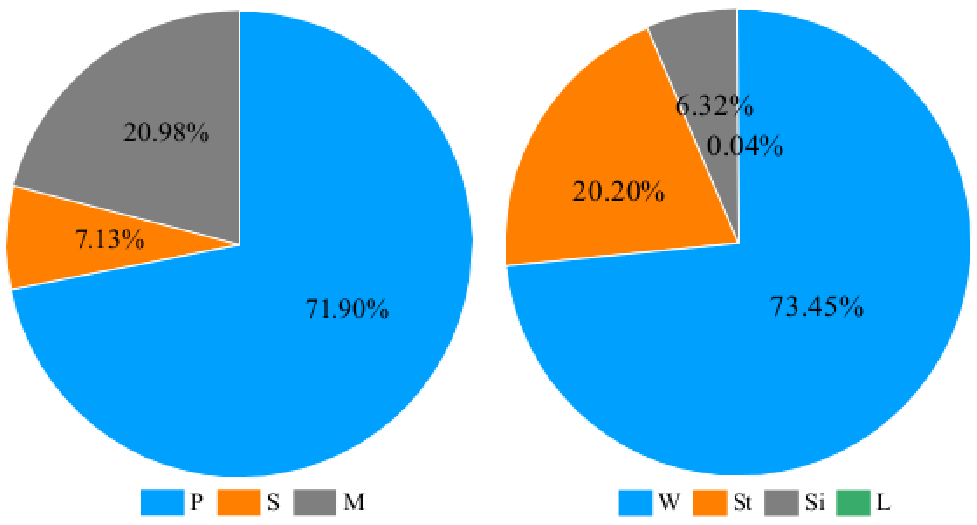


Figure 4.4: Proportion of attribute annotations for Car (left) and Pedestrian(right). Note that Left: P = Parked, S = Stopped, M = Moving, St = Standing, W = Walking, Si = Sitting, L = Lying

KG Generation

Since the PandaSet taxonomy differs from the Traffic Genome one a manual taxonomy alignment has been executed whenever possible, for further details the reader is refereed to the appendix (see Table A.1).

KG Dataset split

As shown in Table 4.1 the low dimension of the dataset leads to $99\% - 1\%$ split ratio respectively for train and test.

	Number
Unique entities	7423
Total entities	580154
Unique Relationships	13
Total relationships	754

Table 4.1: Statistics on diversity inside PandaSet KG

4.0.2 Traffic Genome

Traffic Genome is a traffic scene graph dataset comprised of 1000 scenes selected from Cityscapes dataset[50]. Specifically, there are 34 semantic object classes and 51 relationships. In comparison to Visual Genome dataset, objects and relationships in Traffic Genome are denser and the attribute coverage is higher as shown in Table 4.2, which aims to have an higher perception about relationships in the environment. Note the majority of relationships are focused on spatial relationships (43.85%), such as *in left of*, and area relationships (42.04%), such as *driving on* and *walking on*.

Data-augmentation to mitigate bias

Detailed analysis of Traffic Genome entity distribution shows that only 74% of the entities classes are really used (see Figure 4.6), which degrades training performance during regardless of SGG network employed. Even more, Transformers are data-hungry. Then, it clear that classical data augmentation cannot solve this problem. Thus, Traffic Genome train-set is extended using another dataset, Visual Genome. Since Visual Genome is a dataset with a more general visual domain than Traffic Genome, whose domain it is restricted to Autonomous Driving, pruning and taxonomy alignment are required. Since a manual inspection was not feasible a heuristic rule is defined: a scene belongs to an autonomous driving scenario

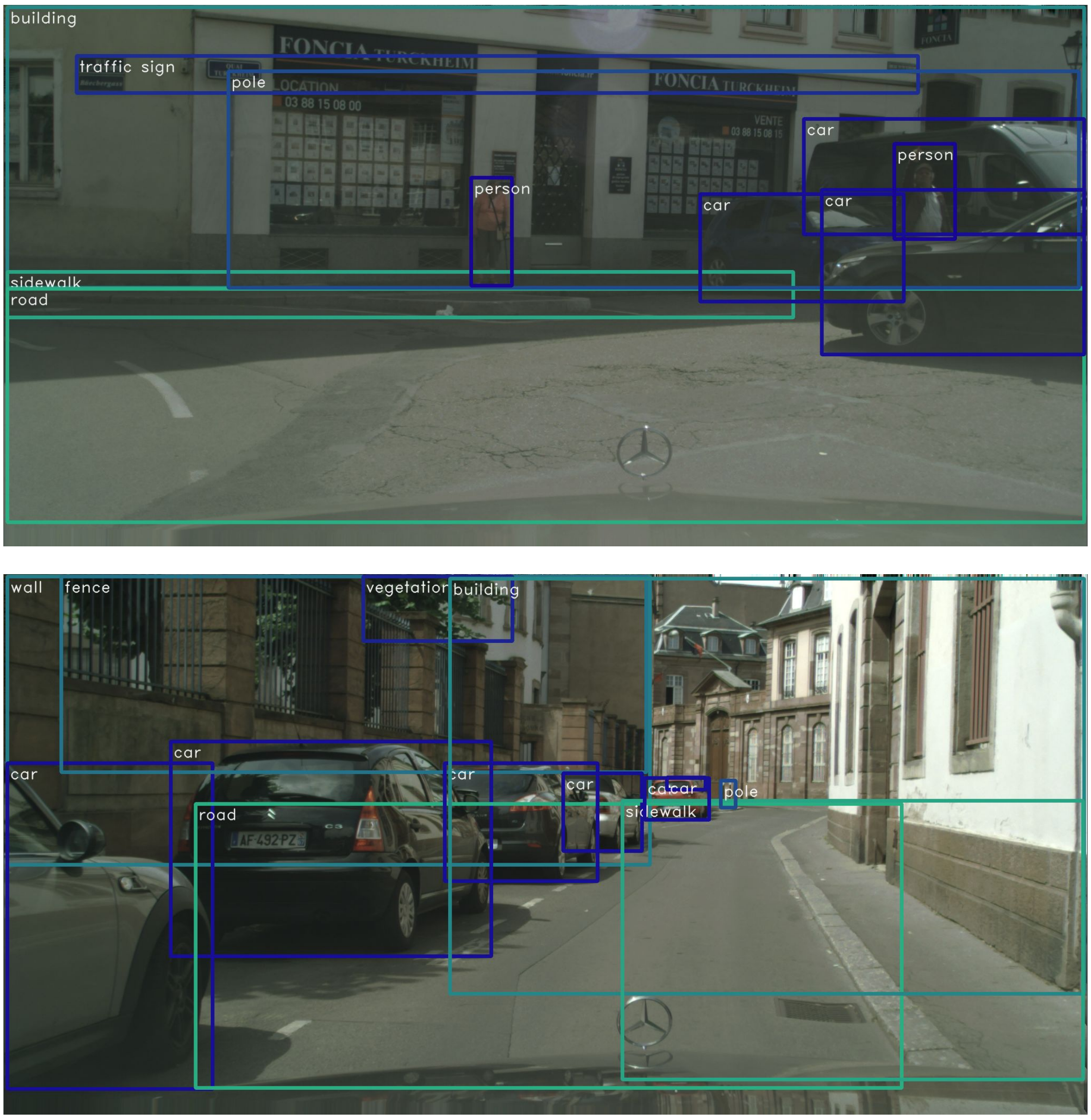


Figure 4.5: Image samples from Traffic Genome with bounding box annotations

whenever at least specific valid entity classes are presented. Valid entity classes are bike, bus, car, motorcycle, sidewalks, truck and vehicle. Valid relationship classes are: above, against, along, at, attached to, behind, belonging to, between, carrying, covered in, growing on, hanging from, has, holding, in, in front of, looking at, lying on, mounted on, near, on, on back of, over, parked on, part of, riding, sitting on, standing on, under, using, walking on, watching, with. For detailed information about the taxonomy alignment rules the reader is referred to the appendix (see Table A.2). As it shown by Figures 4.7 and 4.8 data-augmentation

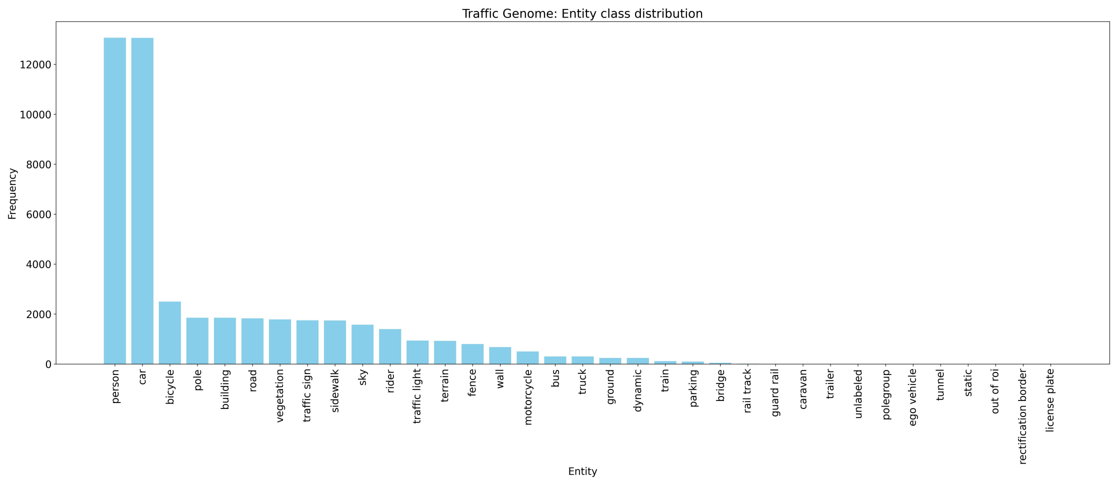
Dataset	Traffic Genome	Visual Genome
Images	1000	108077
Object Categories	34	33877
Total Instances	25146	3843636
Total Instances in Scene Graph	19,291	2254357
Percentage of Instances in Scene Graph	**76.71%**	58.65%
Instances in Scene Graph per Image	**19.29**	20.85
Relationship	51	36550
Total Relationship	29191	1531448
Relationship per image	**29.19**	14.17
Relationships per Instance in Scene Graph	**3.02**	1.36

Table 4.2: A comparison between Traffic Genome and Visual Genome from [10]

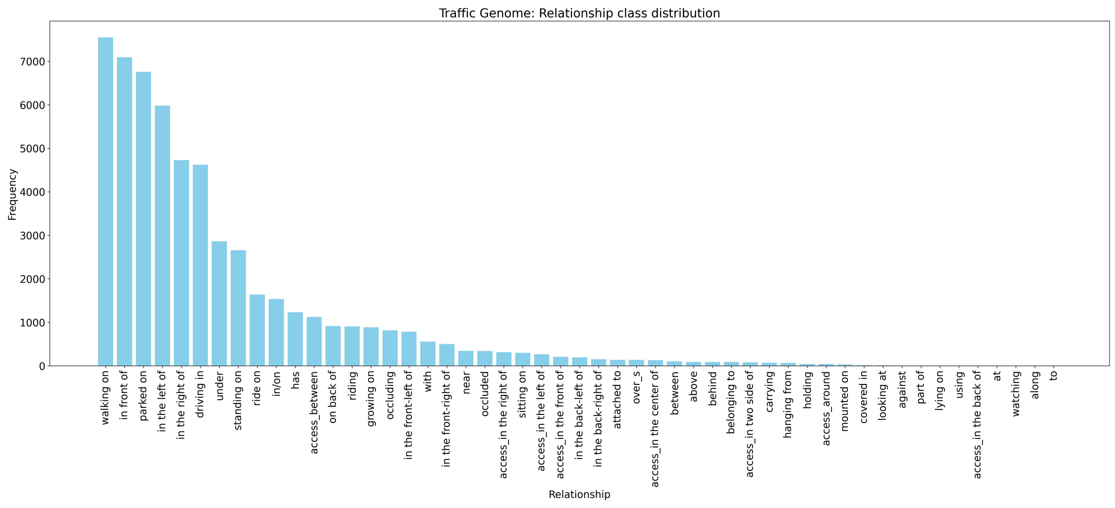
does not eliminated bias since there is still a tailed-distribution both for entities and relationships, though less pronounced. Besides, this issue is useful to determine the effectiveness of the neuro-symbolic approaches. In addition, gamma correction, color jitter and normalization are used.

Set	**Dataset**	**Image**	**Relationship**
training	Pruned Visual Genome	2387	4287
	Traffic Genome	630	18805
	Augmented	3017	23092
validation	Pruned Visual Genome	NA	NA
	Traffic Genome	70	1671
	Augmented	NA	NA
test	Pruned Visual Genome	NA	NA
	Traffic Genome	300	643
	Augmented	NA	NA

Table 4.3: Statistics of Traffic Genome after data-augmentation

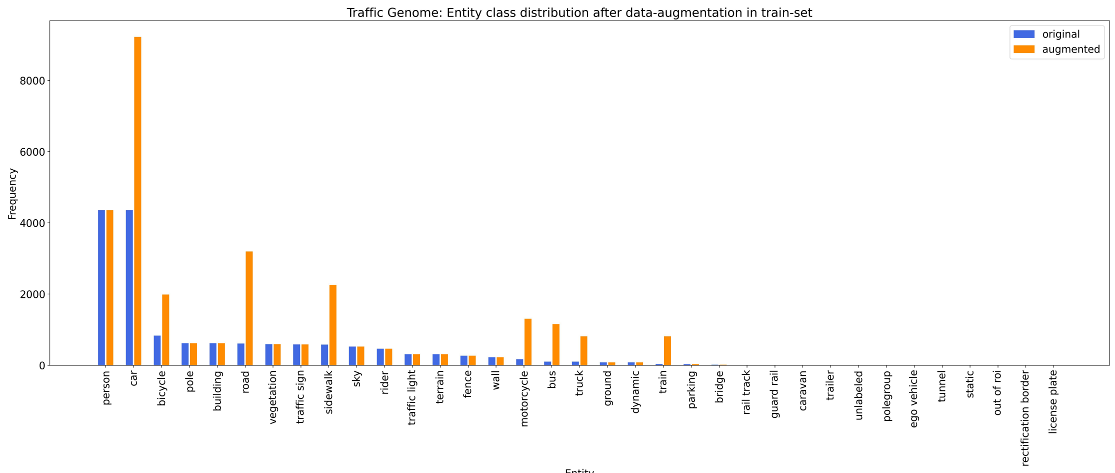


(a) Entity distribution

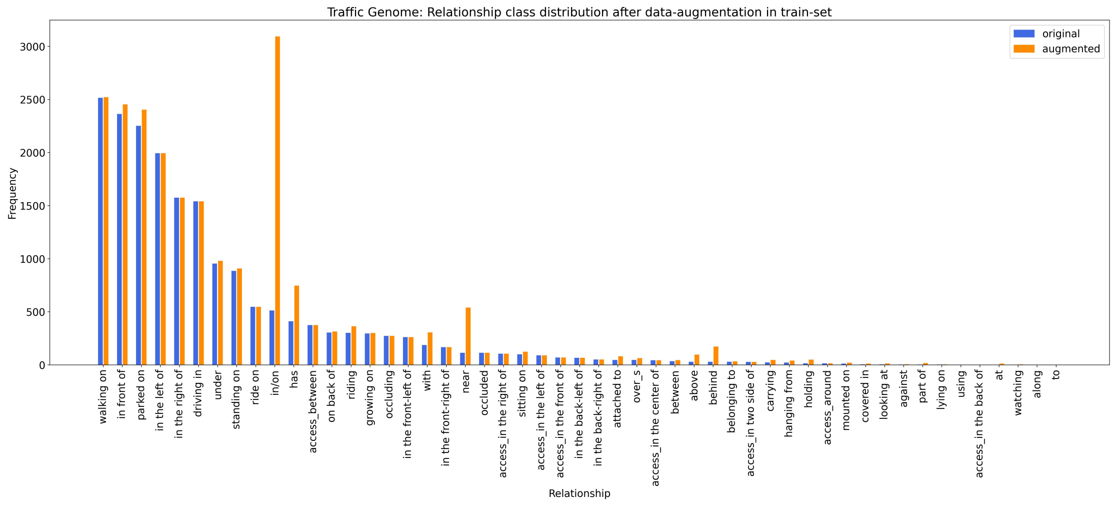


(b) Relationship distribution

Figure 4.6: Distribution of Traffic Genome dataset

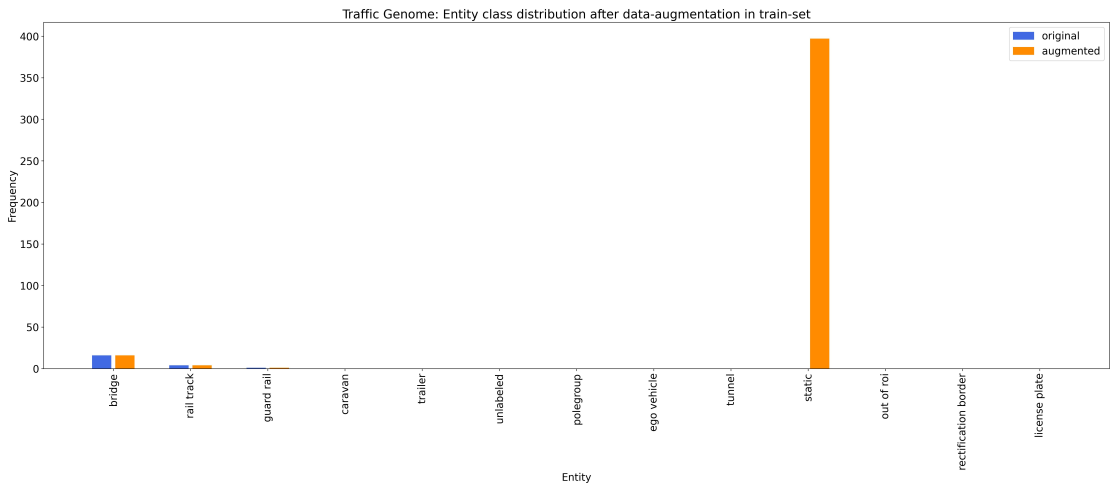


(a) Entity distribution

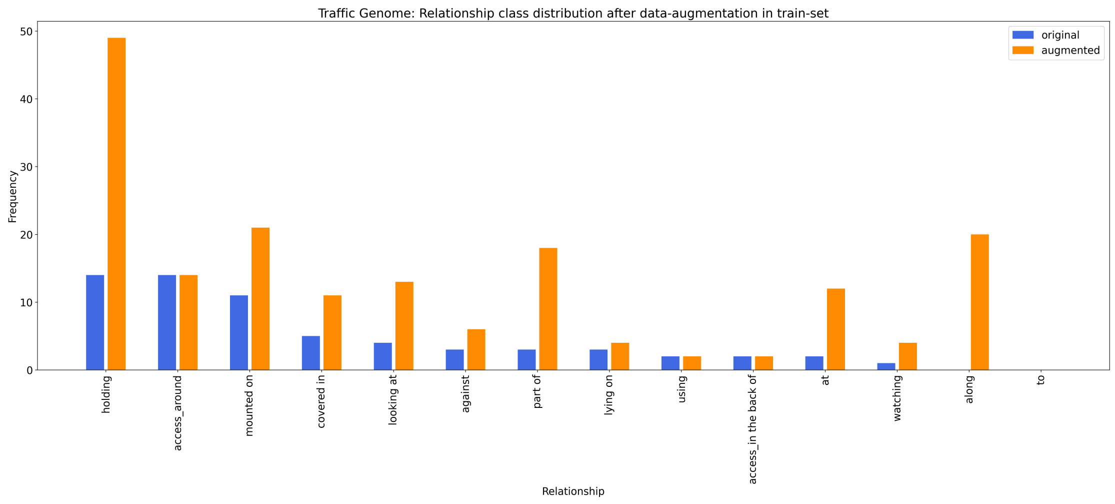


(b) Relationship distribution

Figure 4.7: Effect of data augmentation on the training set



(a) Entity distribution



(b) Relationship distribution

Figure 4.8: Effect of data augmentation into train-set for labels with low frequency

Chapter 5

Methods

In this chapter the different neuro-symbolic techniques for SGG are illustrated. Specifically, RelTR is adopted as the baseline thanks to its states-of-the-art results in one-stage approach. The proposed neuro-symbolic approach aims to introduce a new loss to improve the overall network performance:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{RelTR}} + \alpha \mathcal{L}_{\text{NeSy}} \tag{5.1}$$

where α is an hyperparameter that control how much the neuro-symbolic loss $\mathcal{L}_{\text{NeSy}}$ influences training.

5.0.1 Relational TRansformer (RelTR)

RelTR[51] uses an encoder-decoder architecture based on DETR, where the encoder reasons about the visual feature context and the decoder infers a fixed-size set of triplets subject-predicate-object using different types of attention mechanisms. One key aspect of RelTR is its one-stage approach, which directly predicts sparse scene graphs without combining entities and labeling all possible predicates. This approach allows for efficient and rapid inference. RelTR also introduces a set prediction loss that performs matching between the ground truth and predicted triplets, optimizing the triplet prediction-ground truth assignment during training. Specifically, the loss function is formulated as follows:

$$\mathcal{L}_{\text{triplet}} = \mathcal{L}_{\text{sub}} + \mathcal{L}_{\text{obj}} + \mathcal{L}_{\text{cls}}^{\text{prd}} \tag{5.2}$$

$$\mathcal{L}_{\text{sub}} = \sum_{i=1}^{N_t} \Theta \left[\mathcal{L}_{\text{cls}} + \mathbb{1}_{\left\{c_{\text{sub}}^i \neq \phi\right\}} \mathcal{L}_{\text{box}} \right] \tag{5.3}$$

$$\mathcal{L}_{\text{obj}} = \sum_{i=1}^{N_t} \Theta \left[\mathcal{L}_{\text{cls}} + \mathbb{1}_{\left\{c_{\text{obj}}^i \neq \phi\right\}} \mathcal{L}_{\text{box}} \right] \tag{5.4}$$

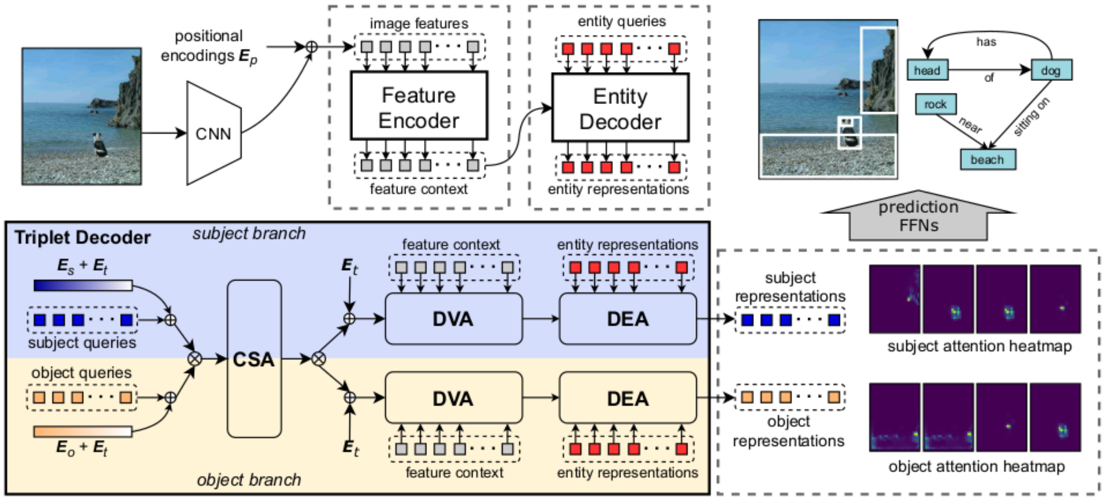


Figure 5.1: RelTR schema

where $\mathcal{L}_{\text{cls}}^{\text{prd}}$ is the cross-entropy loss for predicate classification, $\mathcal{L}_{\text{sub}}$ the subject loss, $\mathcal{L}_{\text{cls}}$ the object loss, c_{sub}^{i} the i-th class object and c_{sub}^{i} the i-th class subject. Note that Θ is 0, when is assigned to the subject or object but the label is predicted correctly and the box overlaps with the ground truth IoU above a predefined threshold; in other cases, Θ is 1. The success of RelTR is based on 3 modules:

Coupled Self-Attention (CSA) captures the context between triplet proposals and the dependencies between subjects and objects in scene graph generation using their latent encoding E_t, E_s, E_o, which are learnt during training.

Decoupled Visual Attention (DVA) extracts visual features independently for subject and object queries representations in scene graph generation. DVA operates in a decoupled manner, where the computations of subject and object representations are independent of each other. This approach allows for the extraction of fine-grained visual information and enhances the localization and classification of subjects and objects.

Decoupled Entity Attention (DEA) improves the localization and classification of subjects and objects by utilizing entity detection results from the entity decoder.

5.1 Knowledge Graph Embedding

Dataset are biased, especially in a Autonomous Driving scenario where scenes usually are extracted from videos, which associate to a scene several frames and their labeling is biased by human annotator. Furthermore, recordable entities and relationships depend on multiple factors such as traffic, weather and privacy-issue, making difficult to capture of all nuances. In the realm of Computer Vision, various researches have been conducted to mitigate bias by introducing external knowledge as Knowledge Graph Embedding. In this case, the most relevant prior works are [52] and [53]: the first tries to tackle bias by imposing semantic relationship between KGE and DETR's embedding; the latter shows a practical example of KGE integration in the Autonomous Driving domain. In this work KGE is used to align Entity Decoder embedding to prior-knowledge whenever the predicted entities from the RelTR are in the external knowledge. In this way, the network is highly penalized when the entities associated to a relationship are not compliant with prior-knowledge: in the latent space of the embedding, distance between entities depends on the relationships in the knowledge base; then, learning the proper entity representation ease the learning of subject and object queries representation by Multi-Head Attention (MHA) modules.
The Knowledge Graph is generated from the PandaSet, utilizing First-Order Logic (FOL) representation within the Resource Description Framework (RDF) language. This Knowledge Graph is used by TransE to generate the Knowledge Graph Embedding.

5.1.1 Loss formulation

The alignment of Entity Decoder to KGE is based on the concept of anchor and candidates. Here, the anchor is defined as the entity prediction given by KGE, while candidates are the entity predictions retrieved by the Entity Decoder, which can be positive or negative depending on the anchor matching. Two loss functions are employed to determine the most effective alignment approach:

Cosine Embedding Loss (CEL) measures the cosine similarity between the anchor and candidates, introducing a margin as a hyperparameter to regulate the similarity threshold.

Hinge Embedding Loss (HEL) based on the Hinge Loss, it calculates the similarity between anchor and candidates that is defined according to the margin, which serves as a hyperparameter.

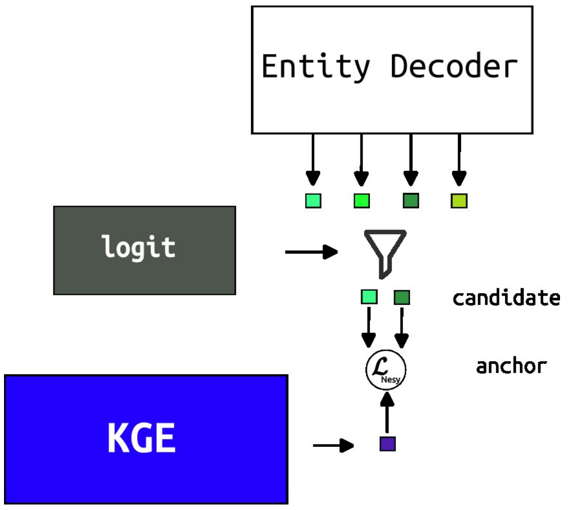


Figure 5.2: Schema of KGE: given the entity logits the filter select the candidate to be aligned to KGE anchor

5.2 Knowledge base of LTN

In this section the knowledge base employed by the LTN is presented. Given an image I, the grounding of the subject x, object y and relationship z done by the corresponding logits of RelTR network:

$$\mathcal{G}(x) = f_x(I) = \hat{x} \tag{5.5}$$

$$\mathcal{G}(y) = f_y(I) = \hat{y} \tag{5.6}$$

$$\mathcal{G}(z) = f_z(I) = \hat{z} \tag{5.7}$$

here f_x, f_y, f_z denote respectively the logits of subjects, objects and relationships. Two types of predicates are defined: **IsOfClass** to evaluate the class membership and **InSet** to evaluate the set-membership. Grounding of predicates is done as follows:

$$\mathcal{G}(\textbf{IsOfClass}) : x, l \rightarrow l^{\intercal}\textbf{softmax}(x) \tag{5.8}$$

$$\mathcal{G}(\textbf{InSet}) : x, \mathcal{S} \rightarrow \sum_{s \in \mathcal{S}} s^{\intercal}\textbf{softmax}(s) \tag{5.9}$$

where x, l represent the predicted class and one-hot enconding label, while $\mathcal{S}$ the set of admissible labels that are express through one-hot encoding. The class membership of relationship z to its ground-truth l_z is evaluated by **relationship axiom**:

$$\forall \, \mathrm{Diag}\,(z, l_z)\,(\textbf{IsOfClass}(\mathcal{G}(z), l_z)) \tag{5.10}$$

To evaluate the set-membership to specific subjects and objects given a relationship, two type of axioms are introduced: **positive constraints** and **negative constraints**. Note both **positive constraints** and **negative constraints** are range constraints that respectively focus on positive and negative domain. Once defined the sets of subjects $\mathcal{S}_x$ and object $\mathcal{S}_y$ that a relationship z can have and the sets of subjects $\mathcal{W}_x$ and object $\mathcal{W}_y$ that it cannot have, positive constraints can be verified through the axiom:

$$\forall \operatorname{Diag}(x, y, z, l_z, \mathcal{S}_x, \mathcal{S}_y) \\ \Big(\textbf{IsOfClass}(\mathcal{G}(z), l_z) \implies \textbf{InSet}(x, \mathcal{S}_x) \wedge \textbf{InSet}(x, \mathcal{S}_y)\Big) \tag{5.11}$$

Conversely, negative constraints satisfaction can be verified through the axiom:

$$\forall \operatorname{Diag}(x, y, z, l_z, \mathcal{W}_x, \mathcal{W}_y) \\ \Big(\textbf{IsOfClass}(\mathcal{G}(z), l_z) \implies \neg\textbf{InSet}(x, \mathcal{W}_x) \vee \neg\textbf{InSet}(x, \mathcal{W}_y)\Big) \tag{5.12}$$

Note that this constraint relaxes penalty of wrong association: it is required to respect at least one InSet predicate to have high-degree of truth. In this way, it possible to investigate which set of relationships is difficult to learn and act accordingly. Even more, an alternative version of negative constraints is formulated to study the effect of above sub-optimal constraint:

$$\forall \operatorname{Diag}(x, y, z, l_x, l_y, \mathcal{W}_z) \\ \Big(\textbf{IsOfClass}(\mathcal{G}(x), l_x) \wedge \textbf{IsOfClass}(\mathcal{G}(y), l_y) \implies \neg\textbf{InSet}(\mathcal{G}(z), \mathcal{W}_z)\Big) \tag{5.13}$$

here $\mathcal{W}_z$ denotes set of wrong relationships for a specific couple of subject x, object y. To verify axioms 5.11 and 5.12 the sets of subjects and objects have been defined accordingly to ground-truth triplets. Even in case of axiom 5.13 the set of wrong relationships has been defined accordingly to the ground-truth triplets: due to high combination of subject and object for a specific wrong relationship, we limited them to same subject and object classes used during the generation of entities to be verified by axiom 5.12. For the sake of readability the First Negative Constraints formulation proposed is denoted as FNC, while the second one as SNC. If the network predicts a triplet <sky, watching, road> the positive constraint is not met since the set of subjects for relationship watching is composed of human-being, i.e. person and rider in Traffic Genome labeling, though road belongs to set of watchable objects. Even more, in this case while FNC would be met since at least the object is not associated to wrong set, SNC would not because the relationship does not hold for those subject and object.
The fuzzy connectors used are Diagonal Quantification, stable Reichenbach Implies, Łukasiewicz And and Or. Positive and negative constraints are employed under two policies:

Ground-truth Policy evaluates negative and positive constraints based on the ground-truth subject and object, i.e. if the predicted triplet is <person, over, road>, but the ground-truth is <car, over, road>, during satisfiability the network evaluates that car and road are possible subject and object for the relationship over.

Prediction Policy evaluates negative and positive constraints based on the predicted subject and object, i.e. if the predicted triplet is <person, over, road>, but the ground-truth is <car, over, road>, during satisfiability the network evaluates whether person and road are possible subject and object for the relationship over. Non Max Suppression (NMS) is applied to restrict evaluation only of significantly different bounding boxes.

Thus, while the ground-truth policy focuses on learning coherent labels with respect to the ground truth, the prediction policy focuses on matching coherent subject and object predictions with respect to the relationship. Of course, the two policies coincide whenever the predictions for object and subject are correct, in other words whenever the network predicts correct classes for each bounding box. All axioms have been evaluated through LTNTorch framework[54].
Note that while the relationship axiom is evaluated on all relationship classes, negative and constraints are restricted to specific ones. Here two versions are analysed: the first one included 41 relationships arbitrary chosen, while the second only relevant tailed relationships in Traffic Genome (see Figure 4.6). In Table 5.1 are reported the details.

Constraint set	Relationships under verification
large	above , access_around , access_between, access_in the center of, access_in the front of, access_in the left of, access_in the right of, access_in two side of, along, attached to, behind, belonging to, between, carrying, driving in, growing on, hanging from, has, holding, in front of, in the back-left of, in the back-right of, in the front-left of, in the front-right of, in the left of, in the right of, in/on, near, occluded, occluding, on back of, over_something, part of, ride on, riding, sitting on, standing on, to, under, watching, with
tail	above, access_around, access_between, access_in the center of, access_in the front of, access_in the left of, access_in the right of, access_in two side of, along, attached to, behind, belonging to, between, carrying, growing on, hanging from, has, holding, in the back-left of, in the back-right of, in the front-left of, in the front-right of, near, occluded, occluding, on back of, over_something, part of, ride on, riding, sitting on, standing on, to, under, watching, with

Table 5.1: Constraints sets used in LTN-based approach

5.3 Experiments

Here we briefly discuss the data-augmentation and setup used. For precise details about hyper-parameters selection the reader is refereed to section 6.1.

5.3.1 Data-augmentation

Data-augmentation is applied only in Traffic Genome, as extensively show in chapter 4. In addition, gamma correction, color jitter and normalization are used.

5.3.2 Setup

For the experiments number of workers is set to 2, while the batch size is 10. TranSE training is done on a Intel Core i9-9940X CPU. While, hyper-parameter selection of RelTR is performed on a NVIDIA(c) V100 GPU provided by HPC@POLITO, neuro-symbolic training are performed on NVIDIA(c) RTX 4090. The detailed description of hyper-parameters is reported in Section 6.1.

5.3.3 Metric

The metrics used during the experiments varies according to the task: Hit for KGE generation; mean Average Precision (mAP) and mean Recall(mR) for SGG. Note that mR is evaluated according to entities' Intersection over Union (IoU), i.e. if a relationship is right, but entities generated are wrong under IoU=0.5, the relationship is discarded. Finally, Predicate CLaSsification (PredCLS) is employed to compare our methods to the ones available in literature.

Chapter 6

Results

In this chapter after discussing the selection of the hyper-parameters of RelTR, the performances of neuro-symbolic approaches are shown. Eventually, the best approaches are compared to the methods available in literature.

6.1 Hyper-parameter selection

Label	**Mean**	**Max**
Entity	25	131
Relationship	29	135

Table 6.1: Statistics on Traffic Genome label distribution

Based on statistics reported in Table 6.1, the following parameter are fixed:

Entity queries of the Entity Decoder are fixed to 140.

Number of triplet are set to 140 to size the dimension of embeddings of subject and object queries according to number of triplets available in the dataset.

Introducing a number of labels (entities and triplets) higher than the dataset keeps a safe margin between maximum number of labels in the dataset and the ones that be predicted. From RelTR configuration reported in Table 6.2, hyper-parameters effect is individually evaluated for 100 epochs; then, combinations of parameters are evaluated till there is clear overfitting. In this phase the metric used for comparison is mAP@50. Here, the Matcher is the module responsible for assigning each ground-truth triplet to a prediction based on the Hungarian algorithm. Hereby, λ_x denotes the weight of a specific loss x.

Parameter	**Value**
Learning rate schedule	Linear
Learning rate dropout	100 epoch
Learning rate backbone	1e-5
Weight decay	1e-4
Positional Embedding	sine
Feature map dimension	2048
Hidden dimension of entity decoder	256
Number of Encoder Layers	6
Number of Decoder Layers	6
Number of Entities	140
Number of Triplets	140
Matcher's cost class	1
Matcher's cost bounding-box	5
Matcher's cost GIoU	2
Matcher's IoU Threshold	0.7
λ_{bbox}	2
λ_{GIoU}	2
λ_{rel}	1
w_{EOS}	0.2

Table 6.2: RelTR configuration 0

6.1.1 Analysis

Starting from *configuration 0* several parameters variations have been explored.

<no-object> class misclassification importance of <no-object> misclassfication weight is studied under moderate variation to avoid prioritizing its prediction over actual classes present in the dataset. Table 6.3 shows that this parameter improves entity detection, when its contribution is small; in the range adopted 0.4 shows better performance.

Experiment	w_{EOS}	**best epoch**	mAP_{val}@0.50
configuration 0	0.1	36	13.6
EXP-1	0.4	11	13.7
EXP-2	0.6	25	11.7
EXP-3	0.7	NO	11.5
EXP-4	0.8	5	4.78

Table 6.3: Performance under End-Of-Sequence (EOS) weight variation under 100 epoch

Attention Head in Encoder-Decoder modules Table 6.4 shows that increasing the complexity model in terms of attention head leads the network to prioritize approximation error over the estimation error of dataset distribution. We believe this fact is caused by the small dataset dimension that makes difficult to generalize when model complexity increases.

Experiment	l_{encoder}	l_{decoder}	**best epoch**	mAP_{val}@0.50
configuration 0	6	6	36	13.6
EXP-5	4	6	NO	12.5
EXP-6	4	4	70	11.9
EXP-7	6	8	54	4.26
EXP-8	8	6	35	11.9
EXP-9	8	8	25	5.94

Table 6.4: Performance under variation of Feature Encoder l_{encoder} and Entity Decoder l_{decoder} layers during 100 epochs

Relationship loss weight an exponential increasing of the relationship loss weight does not improve performance (see Table 6.5), even when Multi-Head Attention varies (see Table 6.6), indicating that the association of relationships does not linearly affect entity detection.

Experiment	$w_{\mathcal{L}_{\text{rel}}}$	**Best epoch**	mAP_{val}@0.50
configuration 0	1	36	13.6
EXP-10	2	14	11.2
EXP-11	4	26	12.3
EXP-12	8	11	10.7

Table 6.5: Performance under variation of the relationship loss coefficient during 100 epochs

Experiment	l_{encoder}	λ_{rel}	**Best epoch**	mAP_{val}@0.50
EXP-12	6	1	204	19.0
EXP-13	6	2	160	20.1
EXP-14	6	4	130	19.6
EXP-15	4	1	130	19.6
EXP-16	4	2	106	21.7
EXP-17	4	4	101	20.8

Table 6.6: Performance under relationship loss coefficient variation λ_{rel} when $w_{\text{EOS}} = 0.4$ and cost bounding box set to 10

Cost assignment of bounding box to classes since entity detection is highly correlated with the association of entity classes to bounding boxes, increasing the cost assignment of bounding boxes to their rightful classes improves the mAP score.

Experiment	K_{bbox}	**Best epoch**	mAP_{val}@0.50
EXP-4	5	11	13.7
EXP-18	10	150	21.6

Table 6.7: Performance by duplicating Matcher's bounding box cost K_{bbox} when $w_{\text{EOS}} = 0.4$

Influence of learning rate the choice of cyclic learning rate is preferable over a linear one, even when overfit happens early, to reach higher performance as shown by Table 6.8.

Experiment	**Learning rate**	**Best epoch**	mAP_{val}@0.50
EXP-1	linear	150	21.6
EXP-19	cyclic	106	22.0

Table 6.8: Performance with different learning rate when $w_{\text{EOS}} = 0.4$ and K_{bbox}=10

Thanks all above considerations, we found a trade-off solution between model complexity and the relevance of loss weights, which constitutes the **baseline** model for neuro-symbolic comparison (see Table 6.9). In Table 6.10 the performance are reported with respect to mean recall. Finally, Table 6.11 summarizes all parameter's values of the baseline. Figure 6.1 shows the performance on validation set.

Experiment	l_{encoder}	λ_{rel}	**Best epoch**	mAP_{val}@0.50
EXP-20	5	1	195	22.7
EXP-21(**baseline**)	5	1.5	170	23.3

Table 6.9: Performance under relationship loss coefficient variation λ_{rel} when $w_{\text{EOS}} = 0.4$ and K_{bbox}=8

mR_{val}@20	mR_{val}@50	mR_{val}@100
15.5	20.9	23.2

Table 6.10: mean recall of the baseline

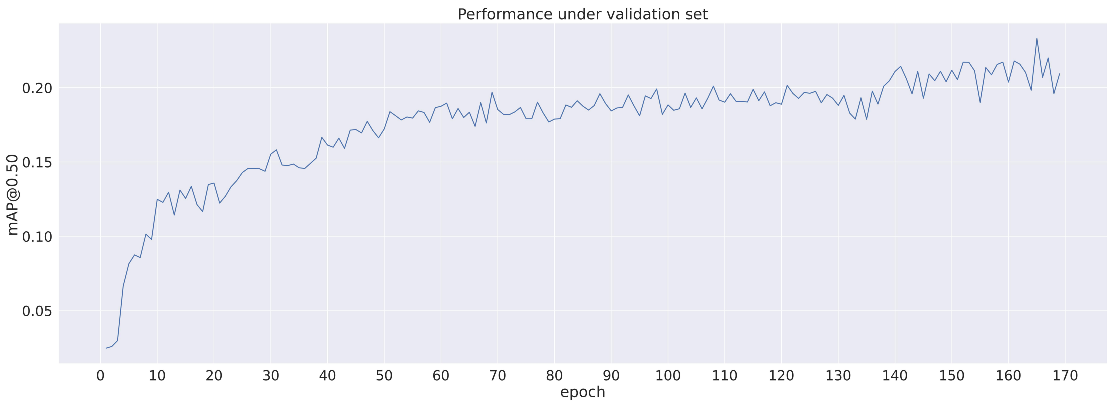
Parameter	**Value**
Learning rate schedule	Cyclic
Minimum Learning rate	1e-5
Maximum Learning rate	1e-3
Learning rate backbone	1e-5
Weight decay	1e-4
Positional Embedding	sine
Feature map dimension	2048
Hidden dimension of entity decoder	256
Number of Encoder Layers	5
Number of Decoder Layers	6
Number of Entities	140
Number of Triplets	140
Matcher's cost class	1
Matcher's cost bounding-box	8
Matcher's cost GIoU	2
Matcher's IoU Threshold	0.7
λ_{bbox}	5
λ_{GIoU}	2
λ_{rel}	1.5
w_{EOS}	0.4

Table 6.11: RelTR baseline: configuration

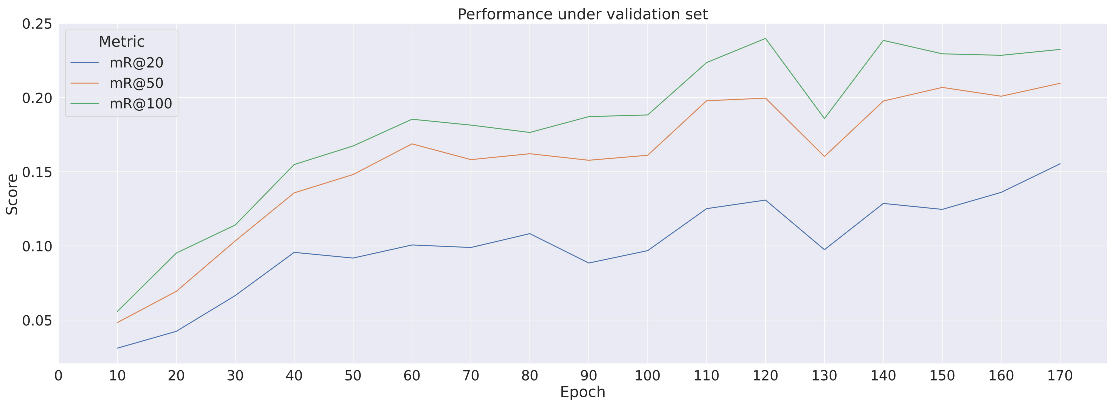
The experiments on hyper-parameter selection have been executed on a NVIDIA V100 using a seed s=18095109307583507207, PyTorch 2.1 and batch size of 12. Instead, neuro-symbolic training was performed on an NVIDIA RTX 4090. Due to its limited RAM capacity, the batch size had to be reduced to 10. To ensure consistent comparisons within the employed architecture, the baseline was also trained on the RTX 4090, revealing noticeable differences only in terms of mAP@0.50 and mR@50 (see Table 6.12).

Experiment	**GPU**	**RAM (GB)**	**CUDA Version**	**batch size**	mAP_{val}@0.50	mR_{val}@20	mR_{val}@50	mR_{val}@100
baseline	V100	32	11.8	12	23.3	15.5	20.9	23.2
baseline	RTX 4090	24	11.8	10	20.3	13.5	19.8	23.2

Table 6.12: Performance under different GPUs



(a) mR score



(b) mAP score

Figure 6.1: Best configuration: validation set performance

6.2 Neuro-symbolic training

In the baseline overfitting becomes pronounced at the 171st epoch. To assess performance systematically within this critical epoch range, the number of training epochs is set to 170. This allows for a comprehensive comparison of different neuro-symbolic methods to identify the most effective in enhancing performance. Metrics exceeding baseline values are highlighted in bold for clarity. In this section both mAP and mR are analysed to see how the neuro-symbolic approaches affect the visual relationship detection tasks in all his task, i.e. entity and relationship detection.

6.2.1 Knowledge alignment

Experiment	**Loss**	α	**Margin**	mAP_{val}@0.5	mR_{val}@100
KGE-1	CEL	1	0.22	19.7	**24.9**
KGE-2	CEL	1	0.42	**20.5**	22.5
KGE-3	CEL	1	0.52	19.7	**24.4**
KGE-4	CEL	0.1	0.52	**21.0**	**23.9**
KGE-5	CEL	1	0.70	20.0	22.8
KGE-6	HEL	1	0.22	19.9	**24.9**
KGE-7	HEL	1	0.42	**20.7**	22.6
KGE-8	HEL	1	0.52	**21.1**	**26.0**
KGE-9	HEL	0.1	0.52	14.7	22.1
KGE-10	HEL	1	0.70	19.4	21.7
baseline	/	/	/	20.3	23.2

Table 6.13: Performance under different KGE-based strategies

In this section, we introduce various KGE-based methods. The comparison in Table 6.13 reveals a decrease in mAP score when aligning the Entity Decoder closely with KGE. Figures 6.2 and 6.3 visually demonstrate this effect, showing that relaxing the Attention mechanism for rare entities expands the analyzed area to predict relationships that better align with external knowledge. RelTR achieves a high mR score under specific conditions, particularly when the margin is small or contained, regardless of the loss function used. This highlights the challenge faced by RelTR in distinguishing between the true class and the <no-object> entity, a common occurrence due to the standard deviation in entity distribution per image, influenced by diverse scenarios captured by Traffic Genome. Table 6.14 compares recall per class between the best-performing KGE method and the baseline, emphasizing improvements and deterioration driven by the Knowledge

Graph. Spatial relationships, frequently represented in the Knowledge Graph, show enhancement, while relationships involving entities with lower frequency exhibit degradation, underscoring the Knowledge Graph's limitations in representing less common scenarios.

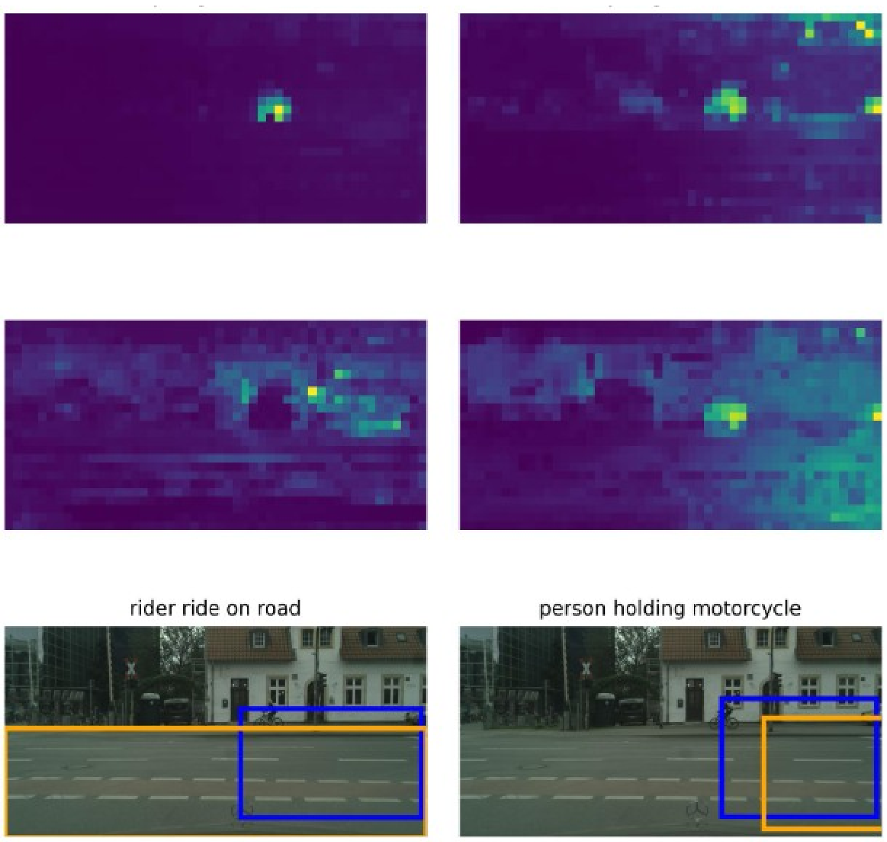


Figure 6.2: Attention Heat-map of Entity Queries for critical example 0: images report baseline (left) and KGE-8 (right) inference

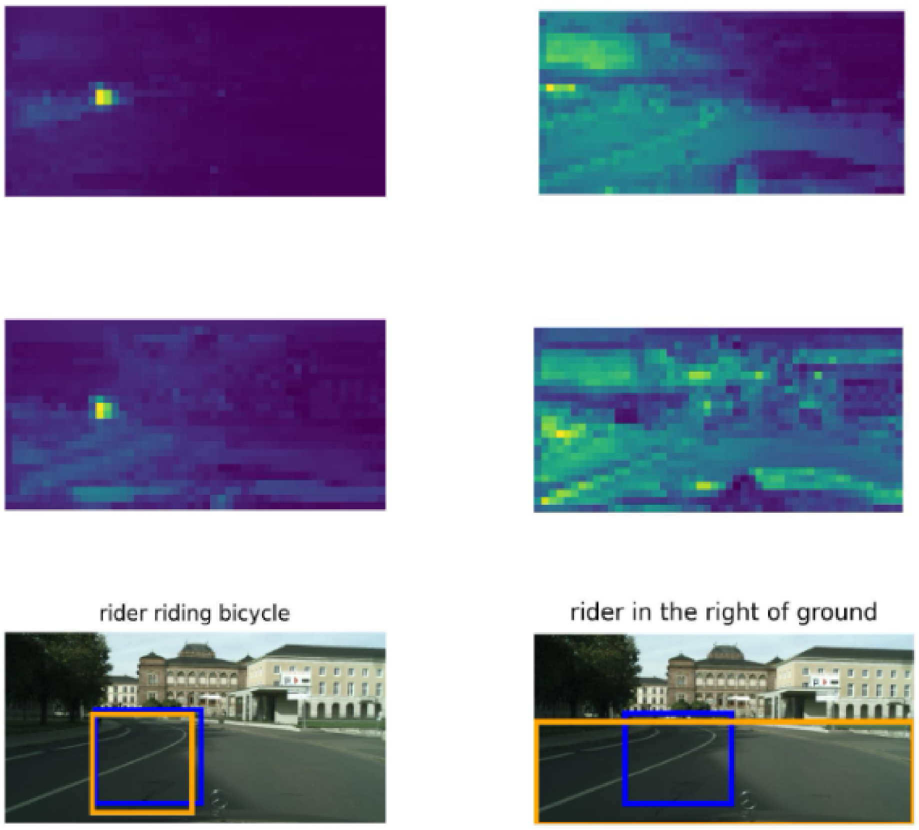


Figure 6.3: Attention Heat-map of Entity Queries for critical example 2: images report baseline (left) and KGE-8 (right) inference

Relationship	$R_{baseline}$@100	R_{KGE}@100
access_around	0.0	0.0
access_between	70.7	85.9
access_in the back of	0.0	0.0
access_in the center of	0.0	0.0
access_in the front of	27.8	11.1
access_in the left of	12.5	12.5
access_in the right of	30.8	38.5
access_in two side of	0.0	100.0
against	0.0	0.0
along	0.0	0.0
at	0.0	0.0
attached to	0.0	0.0
behind	0.0	0.0
belonging to	0.0	0.0
between	0.0	0.0
carrying	0.0	0.0
covered in	0.0	0.0
driving in	65.1	69.5
growing on	53.9	34.2
hanging from	0.0	0.0
has	50.0	48.3
holding	0.0	0.0
in front of	50.8	47.9
in the back-left of	0.0	0.0
in the back-right of	0.0	0.0
in the front-left of	0.0	0.0
in the front-right of	0.0	11.1
in the left of	5.8	27.2
in the right of	27.9	21.6
in/on	50.6	50.6
looking at	0.0	0.0
lying on	0.0	0.0
mounted on	0.0	0.0
near	0.0	20.0
occluded	0.0	0.0
occluding	75.0	73.5
on back of	0.0	0.0
over_something	0.0	0.0
parked on	50.8	49.8
part of	0.0	0.0
ride on	37.5	44.4
riding	73.6	63.9
sitting on	0.0	0.0
standing on	52.8	47.7
to	0.0	0.0
under	30.5	41.6
using	0.0	0.0
walking on	67.2	67.5
watching	0.0	0.0
with	50.0	20.0

Table 6.14: R@100: baseline versus KGE-8

6.2.2 Satisfiability of logical constraints

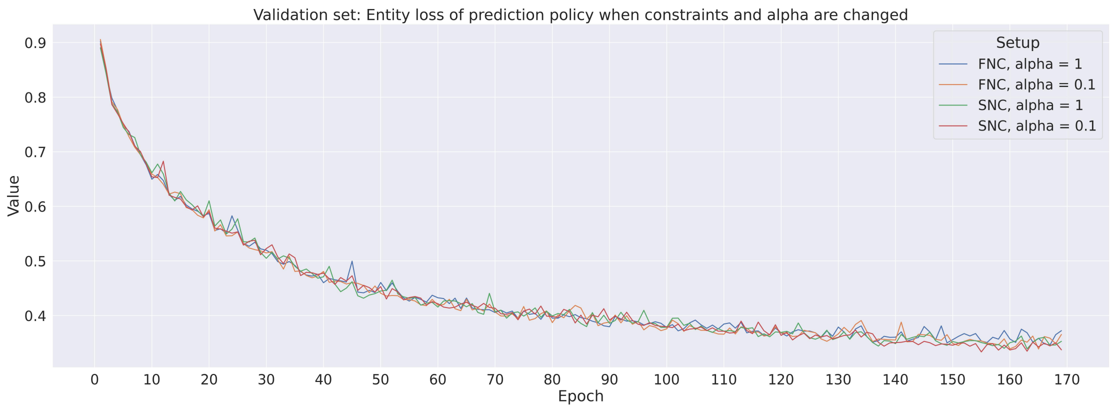
In this section, the influence of LTN during training is examined through two distinct policies: the ground-truth policy and the prediction policy. They are individually analyzed varying the loss' weight and then compared to determine the more suitable approach, if any. In all experiments, the aggregator norm for satisfiability is fixed at 2, while the Aggregator norm for the Universal Quantifier (AUQ) is incremented by 2 at every 50 epochs. Two version of constraints set are employed: a large version, where 41 relationships are evaluated and a tailed one, where only tailed relationships are verified. Two negative constraints versions are compared: FNC and SNC. NMS threshold is set 0.5 to discard similar bounding boxes to refine only the better one, which are supposed to get closer to ground-truth labels during training.

Policy comparison

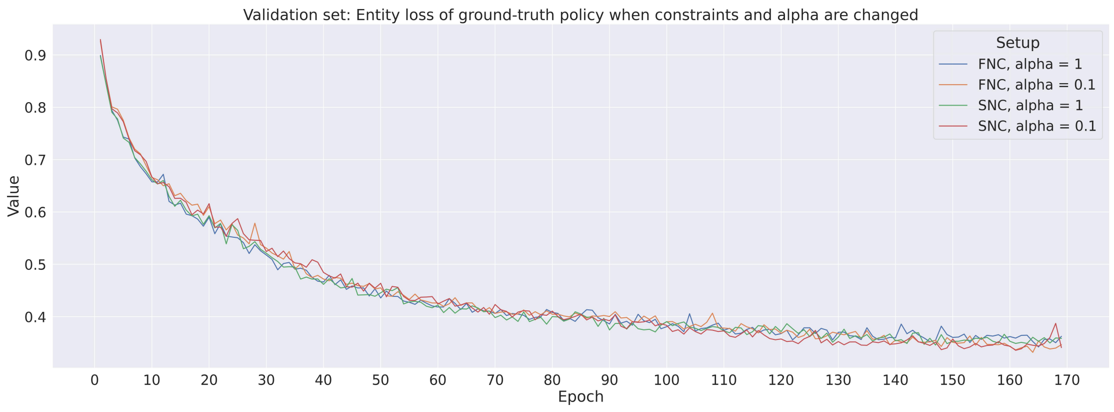
In order to establish an effective experimental strategy, a thorough comparison is conducted between ground-truth and predictions by varying the weights of the loss function and the set of axioms for verification. The diverse behaviors exhibited by mAP and mR during training are illustrated in Table 6.15. This phenomenon stems from the neuro-symbolic formulation, where the axioms emphasize relationship verification over entities. Notably, the verification of entity class-membership occurs solely concerning triplets rather than the ground-truth. Conversely, relationship verification is conducted in both cases. The approach assumes that the cross-entity loss of the RelTR model is potent enough to enforce entity classification, which is proven not to be the case. During optimization, RelTR prioritizes relationship detection over entity detection to achieve an optimal solution. This phenomenon becomes evident in the validation loss (see Figures 6.4 and 6.5), where the cross-entropy loss of entities continues to decrease while the relationship loss overfits. Given these observations, policy decisions and adjustments to loss weights are based solely on the mR metric. Consequently, the neuro-symbolic loss parameter α is set to 0.1 and policy to predict for all subsequent experiments, as it has shown superior performance in terms of mR scores across various thresholds.

Experiment	α	**Policy**	**Constraints' version**	mAP_{val}@0.50	mR_{val}@20	mR_{val}@50	mR_{val}@100
LTN-1	1	prediction	FNC	20.0	**14.0**	**22.2**	**24.9**
LTN-2	1	prediction	SNC	**21.1**	**14.8**	**23.5**	**26.4**
LTN-3	1	ground-truth	FNC	20.3	**15.1**	**21.8**	**24.8**
LTN-4	1	ground-truth	SNC	**21.1**	**13.9**	18.9	21.6
LTN-5	0.1	prediction	FNC	**20.8**	13.3	**20.0**	22.4
LTN-6	0.1	prediction	SNC	**21.0**	**15.6**	**24.2**	**26.6**
LTN-7	0.1	ground-truth	FNC	**21.6**	**15.4**	**20.5**	23.0
LTN-8	0.1	ground-truth	SNC	**20.5**	**14.5**	19.8	**23.3**
baseline	/	/	/	20.3	13.5	19.8	23.2

Table 6.15: mAP and mR scores under different policy



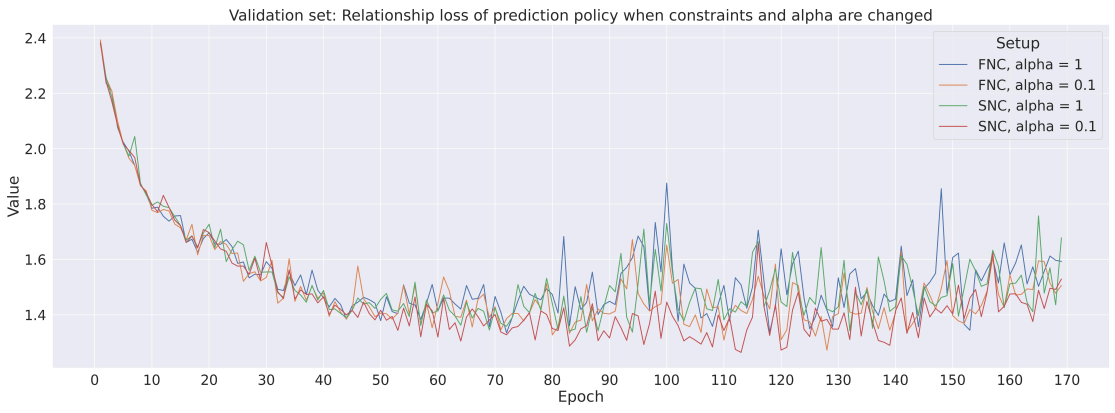
(a) Prediction Policy



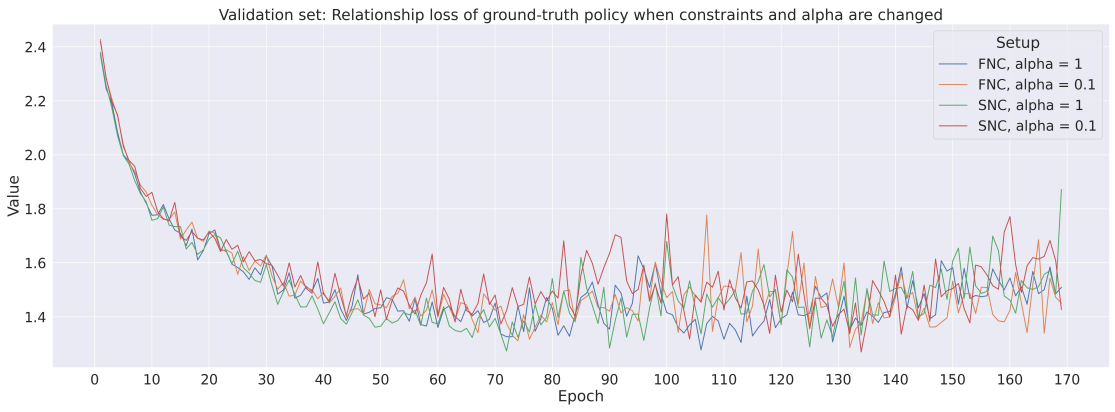
(b) Ground-truth Policy

Figure 6.4: Entity Loss under different setups

To comprehend the behavior of LTN under different policies, we compute the satisfiability of negative and positive constraints during training. It is important to note that these satisfiability measurements are not included in the final loss but serve to assess how the network learns triplet predictions.



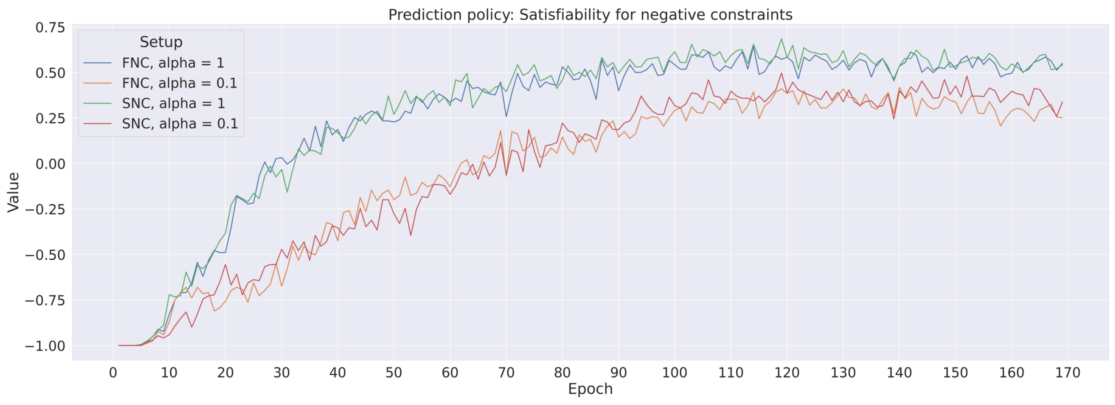
(a) Prediction Policy



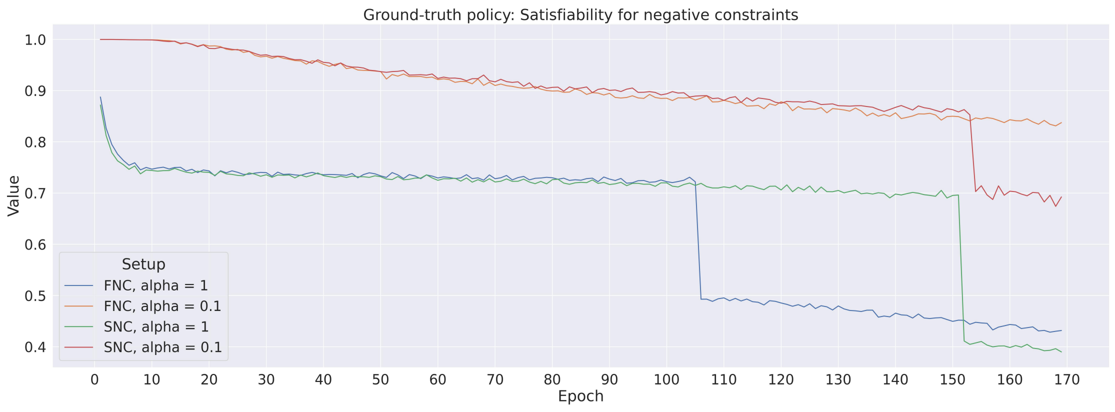
(b) Ground-truth Policy

Figure 6.5: Relationship Loss under different setups

To monitor the elimination of NMS by subjects and objects that are not highly confident in the batch size, we redefine satisfiability by assigning -1 when this phenomenon happens in the batch. Thus, when the satisfiability is lower than 0, it indicates that several bounding boxes are highly unconfident. Figure 6.6 demonstrates that the sub-optimality of FNC affects both the satisfiability of negative and positive constraints, resulting in inferior performance compared to SNC. Only the ground-truth policy is able to partially mitigate this when the neuro-symbolic loss is 1. The ground-truth policy's constraints satisfiability significantly degrades when AUQ is high. In most case the same phenomenon occurs for the satisfiability of the relationship axiom (see Figure 6.8). A possible interpretation could be that emphasizing only prediction coherence to the ground-truth and not governing the rules themselves is not suitable to influence network learning optimally. No significant difference in the satisfiability of the relationship axiom is noticeable in Figure 6.8a under the prediction policy when changing α. This observation arises because the network primarily focuses on negative and positive constraints.

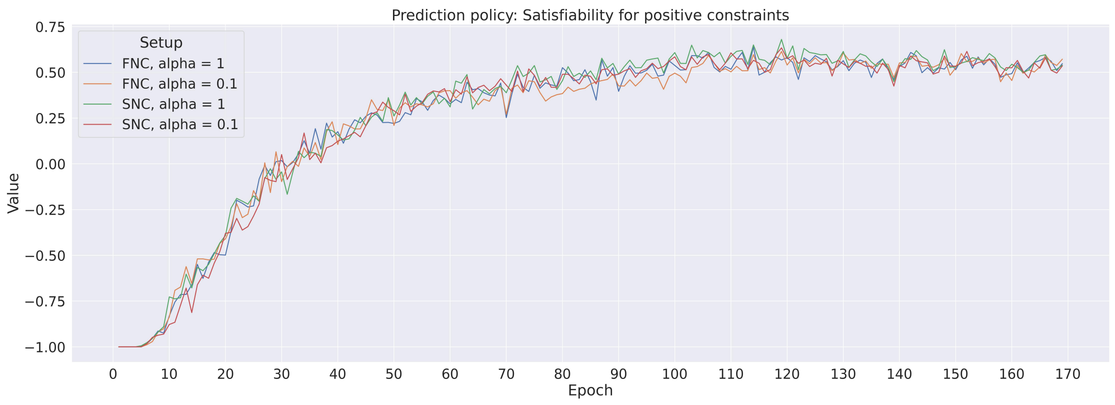


(a) Prediction Policy

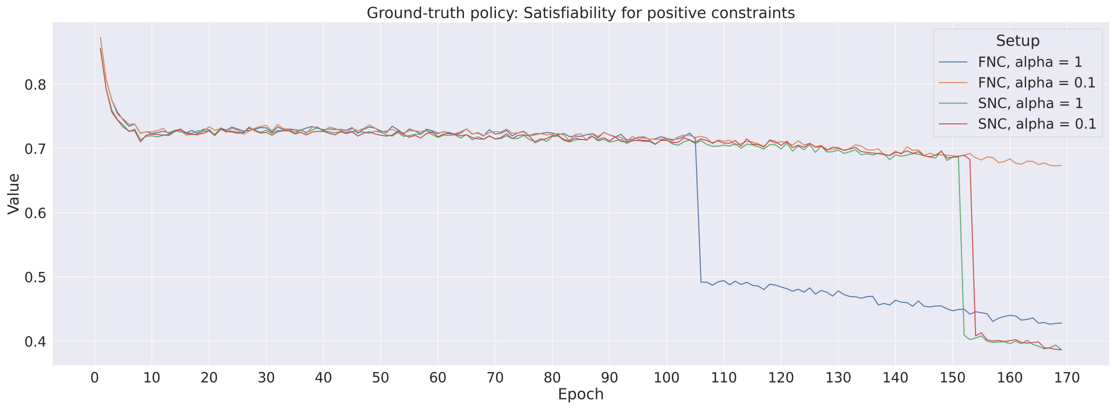


(b) Ground-truth Policy

Figure 6.6: Satisfiability of different negative constraints formulation

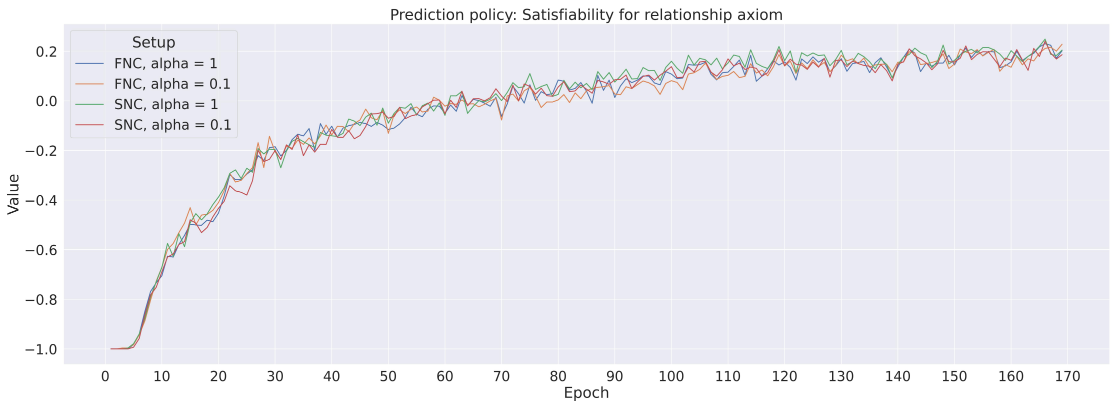


(a) Prediction Policy

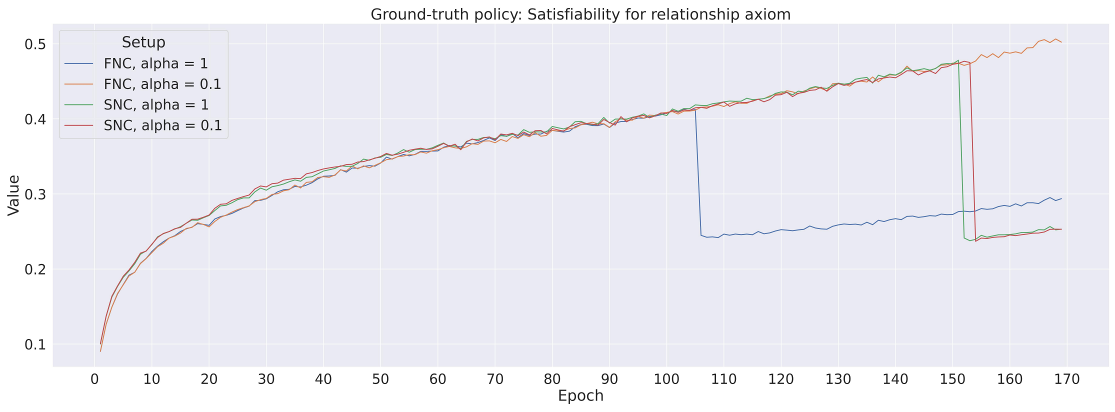


(b) Ground-truth Policy

Figure 6.7: Satisfiability of positive constraints under different Negative Constraints formulations



(a) Prediction Policy



(b) Ground-truth Policy

Figure 6.8: Satisfiability of relationship axiom under different negative constraints formulations

Constraints' set variation

In this section, the effect of constraints set variation is studied under AUQ to understand to determine which is the most effective. For all experiments α, the weight of neuro-symbolic loss, is set to 0.1. Only prediction policy is analyzed. Particular attention is given to mitigate sub-optimality formulation of FNC. Specifically, two methods are analyzed:

Diversity among universal quantification the analysis of individual relationship recall, as detailed in Table 6.16, reveals varying degrees of learning difficulty for different relationships. Consequently, AUQ of positive and negative constraints is differentiate according to learning difficulty, i.e. $p_{AUQ,0}$ is higher when relationships are difficult to respect. Here, Hard Constraints (HC) denotes the constraints on relationship difficulty to respect, while Easy Constraints (EC), the others. The approach is applied only on large constraints set. Note that for the sake of readability only relationships available in validation set are reported in Table 6.16.

Reduction to tailed constraints to understand whether the reduction of constraints to rare relationship helps learning.

Table 6.16 illustrates that performance enhancement is achieved by stratifying constraints based on the ease of relationship detection. However, prioritizing tailed relationships proves to be more effective. By adopting a strategy in which the network exclusively concentrates on challenging examples from the beginning, the learning process becomes more efficient and effective compared to dividing examples based on their difficulty. In fact, the satisfiability error difference in the validation set among AUQ based on constraints difficulty is minimal, as shown in Figure 6.9.

Experiment	**Constraints' set**	$p_{AUQ,0}$ - EC	$p_{AUQ,0}$ - HC	mAP_{val}@0.50	mR_{val}@100
LTN-6	large	2	2	**20.8**	22.4
LTN-9	large	2	4	**21.3**	21.7
LTN-10	tail	2	2	**20.4**	**23.8**
LTN-11	tail	4	4	**20.6**	20.9
baseline	/	/	/	20.3	23.2

Table 6.17: Performance under different constraints' strategy

Relationship	$R_{val,baseline}$@100	$R_{val,LTN-1}$@100	$R_{val,LTN-2}$@100
access_around	0	0	0
access_between	90.2	91.3	90.2
access_in the back of	0	0	0
access_in the front of	27.0	37.0	31.5
access_in the left of	25.0	37.5	37.5
access_in the right of	46.2	46.2	46.2
access_in two side of	0	100.0	0
against	0	0	0
along	0	0	0
at	0	0	0
attached to	0	0	0
belonging to	0	0	0
carrying	0	0	0
driving in	69.1	75.4	75.5
growing on	64.5	61.8	61.8
hanging from	0	0	0
has	50.0	51.7	56.9
in front of	52.8	52.5	53.4
in the back-right of	0	0	0
in the front-left of	7.7	7.7	11.5
in/on	60.0	61.0	60.0
looking at	0	0	0
lying on	0	0	0
mounted on	0	0	0
near	40.0	20.0	20.0
occluded	0	0	0
on back of	11.1	22.2	27.8
over_something	0	37.5	12.5
part of	0	0	0
ride on	41.7	50.0	44.4
sitting on	0	0	0
standing on	59.1	64.4	60.5
to	0	0	0
under	35.6	44.1	35.2
using	0	0	0
walking on	79.8	85.5	84.9
watching	0	0	0
with	58.3	31.7	35.0

Table 6.16: Recall analysis across relationships with large constraints set; reporting only the best values

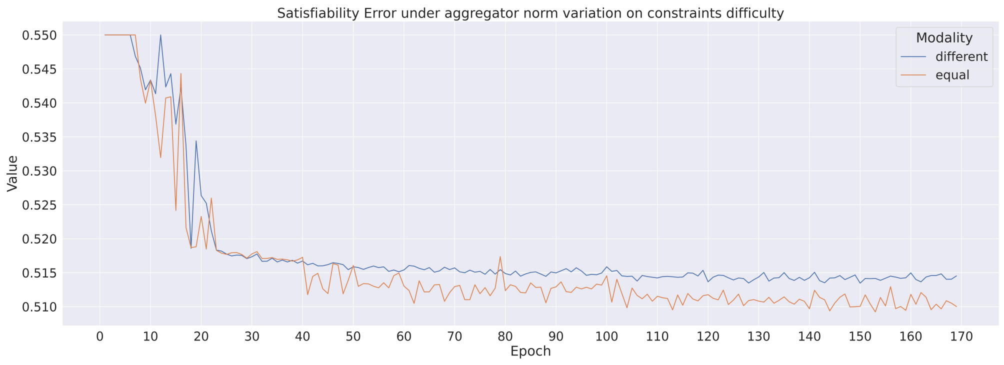


Figure 6.9: Satisfiability error in large constraints set

Furthermore, an analysis is conducted to assess the performance of the two versions of constraints under the condition of severe penalization of low degrees of truth for the axioms, utilizing a high value for the AUQ. Given the substantial improvement demonstrated by the tailored constraints' set, the analysis is confined to this specific set of constraints. In this configuration, the value $p_{AUQ,0}$ is set to 6. Generally, performance degrades around 150, specifically when p_{AUQ} is set to 12, as evidenced by an increase in satisfiability error during validation (see Figure 6.10). This suggests that in this particular setting, the network encounters difficulties in the learning process.
It is noteworthy that sub-optimal constraints within the FNC are inferior to those in the SNC. The inherent complexity of the learning task is further complicated by relaxing constraints, resulting in increased confusion and hindering the network's capacity for improvement. Notably at 160-th epoch the network is unable to generate valid bounding boxes, producing zero-point boxes, rendering the Hungarian algorithm and the overall training unfeasible. In Table 6.18 the maximum score registered during these training sessions is reported.

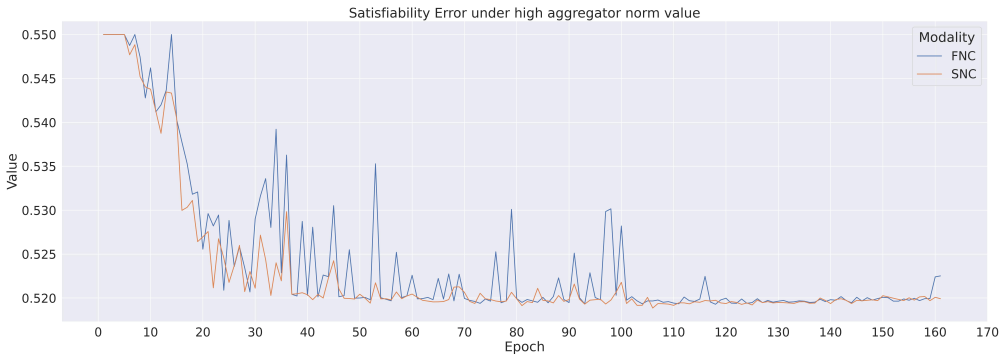


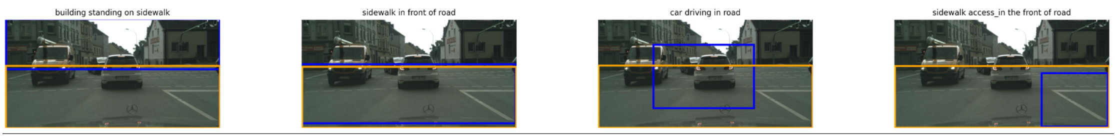
Figure 6.10: Satisfiability error in large constraints set under high AUQ

Constraints' version	$p_{AUQ,0}$	mAP_{val}@0.50@epoch	mR_{val}@100@epoch
FNC	6	21.3@151	23.5@130
SNC	6	19.6@170	24.7@170

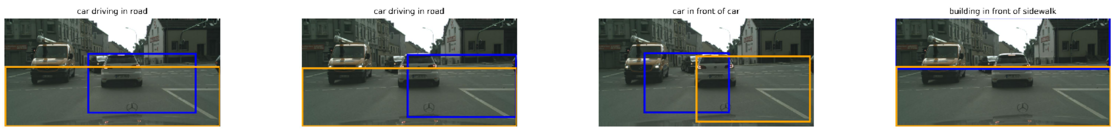
Table 6.18: Maximum scores under different negative constraints' formulation when p_{AUQ} is high

6.3 Qualitative Results

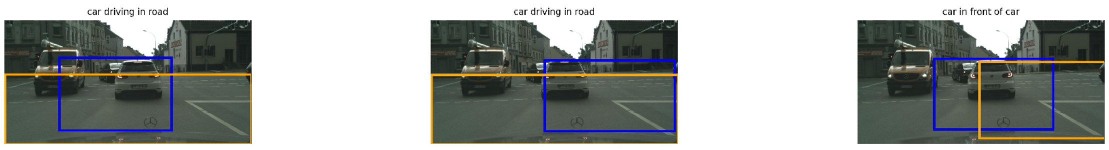
In the section a qualitative analysis is performed on the best configurations found. For sake of readability, we denote the configuration LTN-12 as RelTR and KGE-8 as RelTR-KGE. In this context, a confidence threshold of 0.3 is applied, and the analysis focuses on the top 4 predictions. The results illustrated in Figures 6.11 and 6.12 indicate challenges in the network's ability to accurately resize bounding boxes, particularly the primary one, given the low mAP. All methods are able to predict coherent triplets, but only RelTR-LTN select bounding boxes which are the most reliable. It is worth noting that in the absence of a vehicle, both neuro-symbolic approaches successfully capture coherent triplets, while the baseline falls short in this regard.


building standing on sidewalk
sidewalk in front of road
car driving in road
sidewalk access_in the front of road

(a) RelTR

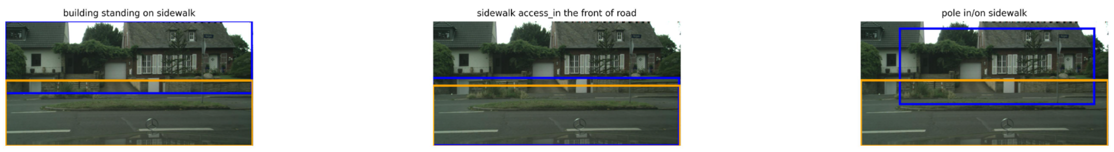


(b) RelTR-KGE

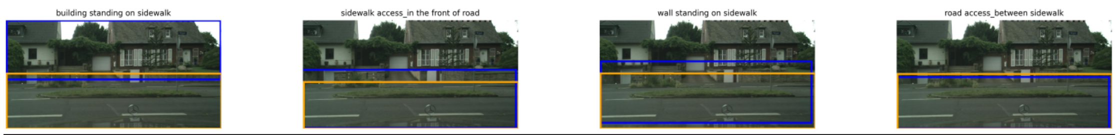


(c) RelTR-LTN

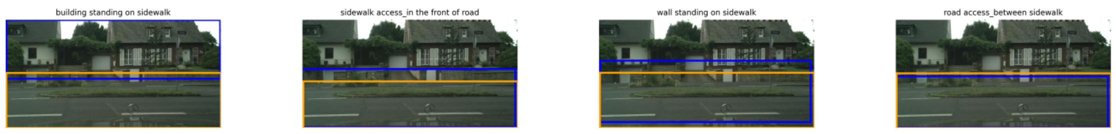
Figure 6.11: Visual Relationship Detection of traffic scene


building standing on sidewalk
sidewalk access_in the front of road
pole in/on sidewalk

(a) RelTR


building standing on sidewalk
sidewalk access_in the front of road
wall standing on sidewalk
road access_between sidewalk

(b) RelTR-KGE


building standing on sidewalk
sidewalk access_in the front of road
wall standing on sidewalk
road access_between sidewalk

(c) RelTR-LTN

Figure 6.12: Visual Relationship Detection of scene without vehicles

6.4 Comparison with the start-of-the-art

In this section, we compare the best-performing neuro-symbolic methods with the SOTA model reported in [10], which reports exclusively their Predicate CLassification (PredCLS). Compared to the two-stage approaches, in our formulation RelTR is not enable to surpass all SOTA methods, though the employment of NeSy reduces this gap, specially in the LTN-based approach. A possible interpretation arises from the model complexity of RelTR, where object and relationship detection occur simultaneously in a transformer-based architecture. Introducing prior knowledge only to the relationship detection task is insufficient, as the network prioritizes it over the ensemble framework of Visual Relationship Detection. Extending neuro-symbolism to modules responsible for visual feature extraction, such as the Feature Encoder, may enhance Attention on visual features.

Method		Loss	PredCLS		
			mR@20	mR@50	mR@100
two-stage	IMP[10]	Cross Entropy	7.03	11.62	15.44
	Motif[10]	Cross Entropy	12.13	18.55	21.64
	VCTree[10]	Cross Entropy	13.48	19.79	24.85
	VCTree - TDE[10]	Cross Entropy	12.82	18.33	22.79
one-stage	RelTR (ours)	Cross Entropy (Baseline)	11.66	16.61	18.81
		Cross Entropy + KGE	11.83	17.97	20.81
		Cross Entropy + LTN	11.10	17.42	22.61

Table 6.19: Comparison with state-of-the-art methods on Traffic Genome test-set

Chapter 7

Conclusion

In this study, for first time we explore the feasibility of employing neuro-symbolic one-stage Scene Graph Generation methods within the domain of Autonomous Driving. The results obtained from the neuro-symbolic models reveal that incorporating prior knowledge can enhance performance. However, the proposed formulations need refinement for practical applicability. Transformer models exhibit a high data dependency, and the integration of external labeled data alone is insufficient to achieve acceptable performance in real scenarios, as demonstrated. This underscores the essential role of prior knowledge. The Attention mechanism of Transformer is notably sensitive to prior knowledge and its introduction shows a divergent behaviour respect to task involved: introducing prior-knowledge only in relationship detection shows that during learning the network prioritizes relationship detection over entity detection. A possible solution could be the integration of a trainable layers in the Feature Encoder which imposes constraint on visual features extraction to improve the features context, which is used by the other Attention modules. Thus, these trainable layers would refine prior knowledge directly before evaluating misclassifications as done in [29]. Furthermore, KGE are biased with respect to facts which are rares in their knowledge base; then, the contribution of KGE should be leveraged accordingly. LTN, on the other hand, exclusively relies on the quality of the constraints design. This distinctive characteristic may more effectively counterbalance biases present in the dataset, favoring its applicability in Autonomous Driving, where safety is a crucial aspect.

Appendix A

Appendix

A.1 KGE Generation

The hidden dimension of KGE has been fixed to 256, the dimension of RelTR's Entity Decoder, to employ KGE alignment. Hit@k is evaluated at k=10 to retrieve a robust KGE model. After a trail and error procedure the number of total number of training epoch is set to 200 due to convergence (see Figure.A.1).

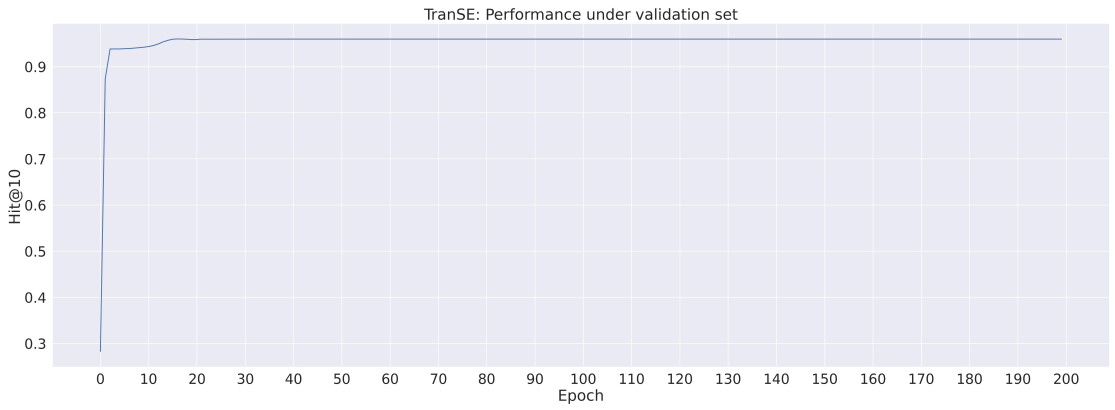


Figure A.1: TransE performance during training

A.2 Taxonomy alignment

Here are reported the taxonomy alignments used during the creation of the Knowledge Graph.

Type	**Label**	**Frequency**
Category	Car	1,040,268
	Truck	75,987
	Road	120,834
	Static	66,914
	Person	159,450
	Object	48,575
	Rider	1
	Bicycle	15,047
	Dynamic	17,111
	Bus	11,552
	Caravan	3,598
	Motorcycle	10,119
	Fence	53,920
Relationship	Parked	843,722
	Translation	4,361,898
	Size	4,361,898
	Rotation	1,453,966
	Stopped	85,305
	Moving	244,471
	In/On	120,834
	Walking	106,346
	Sitting	2,488
	Standing	26,281
	With	56,061
	Without	17,680
	Lying	80

Table A.1: PandaSet attribute after taxonomy alignment

Visual Genome Label	**Traffic Genome Label alignment**
bike	bike
building	building
bus	bus
car	car
fence	fence
motorcycle	motorcycle
pole	pole
sidewalk	sidewalk
sign	traffic sign
street	road
train	train
truck	truck
vehicle	dynamic
woman	person
child	person
girl	person
kid	person
lady	person
man	person
men	person
person	person
people	person
cat	static
cow	static
dog	static
rock	static
sheep	static
tree	static
vegetable	static
wire	Static

Table A.2: Visual Genome Entity taxonomy alignment

Bibliography

[1] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. «Image retrieval using scene graphs». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 3668–3678 (cit. on p. 1).

[2] Eren Erdal Aksoy, Alexey Abramov, Florentin Wörgötter, and Babette Dellen. «Categorizing object-action relations from semantic scene graphs». In: *2010 IEEE International Conference on Robotics and Automation* (2010), pp. 398–405 (cit. on p. 1).

[3] Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermüller, and Yiannis Aloimonos. «From Images to Sentences through Scene Description Graphs using Commonsense Reasoning and Knowledge». In: *ArXiv* abs/1511.03292 (2015) (cit. on p. 1).

[4] Ranjay Krishna, Yuke Zhu, Oliver Groth, and et al. «Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations». In: *International Journal of Computer Vision* 123 (2017), pp. 32–73 (cit. on pp. 1, 3, 28).

[5] SAE Taxonomy. «Definitions for terms related to driving automation systems for on-road motor vehicles. Publication J3016_202104». In: *Society of Automotive Engineers* (2021) (cit. on p. 1).

[6] Huijie Wang et al. «OpenLane-V2: A Topology Reasoning Benchmark for Unified 3D HD Mapping». In: *NeurIPS*. 2023 (cit. on p. 1).

[7] Tianyu Li et al. «Graph-based Topology Reasoning for Driving Scenes». In: *arXiv preprint arXiv:2304.05277* (2023) (cit. on p. 1).

[8] Arnav Vaibhav Malawade, Shih-Yuan Yu, Brandon Hsu, Harsimrat Kaeley, Anurag Karra, and Mohammad Abdullah Al Faruque. «roadscene2vec: A tool for extracting and embedding road scene-graphs». In: *Knowledge-Based Systems* 242 (2022), p. 108245. ISSN: 0950-7051. DOI: `https://doi.org/10.1016/j.knosys.2022.108245`. URL: `https://www.sciencedirect.com/science/article/pii/S0950705122000739` (cit. on p. 1).

[9] Arnav Vaibhav Malawade, Shih-Yuan Yu, Junyao Wang, and Mohammad Abdullah Al Faruque. «RS2G: Data-Driven Scene-Graph Extraction and Embedding for Robust Autonomous Perception and Scenario Understanding». In: *arXiv preprint arXiv:2304.08600* (2023) (cit. on p. 1).

[10] Zhixuan Zhang, Chi Zhang, Zhenning Niu, Le Wang, and Yuehu Liu. «GeneAnnotator: A Semi-automatic Annotation Tool for Visual Scene Graph». In: (2021). arXiv: `2109.02226 [cs.CV]` (cit. on pp. 2, 28, 33, 66).

[11] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. «Visual Relationship Detection with Language Priors». In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 852–869 (cit. on pp. 3, 28).

[12] Artur S. d'Avila Garcez and L. Lamb. «Neurosymbolic AI: the 3rd wave». In: *Artificial Intelligence Review* (2023) (cit. on p. 4).

[13] Lotfi A. Zadeh. «Fuzzy Sets». In: *Inf. Control.* 8 (1965), pp. 338–353 (cit. on p. 4).

[14] Erich Peter Klement, Radko Mesiar, and Endre Pap. «Triangular norms. Position paper I: basic analytical and algebraic properties». In: *Fuzzy Sets Syst.* 143 (2004), pp. 5–26 (cit. on pp. 5, 6).

[15] R. H. Richens. «General program for mechanical translation between any two languages via an algebraic interlingua». In: *Mechanical Translation* 3 (1956), p. 37 (cit. on p. 7).

[16] R G Quillian. «A notation for representing conceptual information. An application to semantics and mechanical English paraphrasing». In: 1963 (cit. on p. 7).

[17] Jeffrey Travers and Stanley Milgram. «An Experimental Study of the Small World Problem». In: *Sociometry* 32.4 (1969), pp. 425–443 (cit. on p. 7).

[18] Marvin Minsky. *A Framework for Representing Knowledge.* Tech. rep. 306. (1974). Santa Monica: MIT-AI Memo, 1974, p. 76 (cit. on p. 7).

[19] Ronald J. Brachman. «A Structural Paradigm for Representing Knowledge». PhD thesis. Harvard University, 1977 (cit. on p. 7).

[20] William A. Woods. «What's in a Link: Foundations for Semantic Networks». In: *Representation and Understanding.* Ed. by Daniel G. Bobrow and Allan Collins. Elsevier, 1975, pp. 35–82 (cit. on p. 7).

[21] John Sowa. «Semantics of Conceptual Graphs». In: *17th Annual Meeting of the Association for Computational Linguistics.* Ed. by Norman K. Sondheimer. University of California at San Diego, La Jolla, CA, USA: The Association for Computational Linguistics, 1979, pp. 39–44. URL: `https://www.aclweb.org/anthology/P79-1010/` (cit. on p. 7).

[22] Aidan Hogan et al. «Knowledge Graphs». In: *ACM Computing Surveys (CSUR)* 54 (2020), pp. 1–37 (cit. on p. 7).

[23] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. «Knowledge Graph Embedding: A Survey of Approaches and Applications». In: *IEEE Transactions on Knowledge and Data Engineering* 29 (2017), pp. 2724–2743 (cit. on pp. 7, 8).

[24] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. «Translating Embeddings for Modeling Multi-relational Data». In: *NIPS*. 2013 (cit. on p. 8).

[25] Luciano Serafini and Artur S. d'Avila Garcez. «Learning and Reasoning with Logic Tensor Networks». In: *AI*IA 2016 Advances in Artificial Intelligence*. Ed. by Giovanni Adorni, Stefano Cagnoni, Marco Gori, and Marco Maratea. Cham: Springer International Publishing, 2016, pp. 334–348 (cit. on p. 9).

[26] Francesco Manigrasso, Filomeno Davide Miro, Lia Morra, and Fabrizio Lamberti. «Faster-LTN: A Neuro-Symbolic, End-to-End Object Detection Architecture». In: *Artificial Neural Networks and Machine Learning – ICANN 2021*. Ed. by Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter. Cham: Springer International Publishing, 2021, pp. 40–52 (cit. on p. 11).

[27] Simone Martone, Francesco Manigrasso, Fabrizio Lamberti, and Lia Morra. «PROTOtypical Logic Tensor Networks (PROTO-LTN) for Zero Shot Learning». In: *2022 26th International Conference on Pattern Recognition (ICPR)*. 2022, pp. 4427–4433. DOI: `10.1109/ICPR56361.2022.9956239` (cit. on p. 11).

[28] Ivan Donadello and Luciano Serafini. «Compensating Supervision Incompleteness with Prior Knowledge in Semantic Image Interpretation». In: *IJCNN*. IEEE, 2019, pp. 1–8 (cit. on p. 11).

[29] Alessandro Daniele and Luciano Serafini. «Knowledge Enhanced Neural Networks for Relational Domains». In: *AIxIA 2022 – Advances in Artificial Intelligence*. Ed. by Agostino Dovier, Angelo Montanari, and Andrea Orlandini. Cham: Springer International Publishing, 2023, pp. 91–109 (cit. on pp. 11, 67).

[30] John D Lafferty, Andrew McCallum, and Fernando CN Pereira. «Conditional random fields: Probabilistic models for segmenting and labeling sequence data». In: *Proceedings of the 18th International Conference on Machine Learning*. 2001, pp. 282–289 (cit. on p. 13).

[31] Xiaojun Chang and al. «A Comprehensive Survey of Scene Graphs: Generation and Application». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2023), pp. 1–26 (cit. on pp. 13, 14).

[32] Thomas N. Kipf and Max Welling. «Semi-Supervised Classification with Graph Convolutional Networks». In: *International Conference on Learning Representations (ICLR)*. 2017 (cit. on p. 13).

[33] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. ICS 8504. San Diego, California: Institute for Cognitive Science, University of California, Sept. 1985 (cit. on p. 14).

[34] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667 (cit. on p. 15).

[35] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. «Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation». In: *Conference on Empirical Methods in Natural Language Processing*. 2014 (cit. on p. 15).

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is All you Need». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017 (cit. on p. 15).

[37] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». In: *arXiv:1506.01497 [cs]* (June 2015). arXiv:1506.01497. 1, 2, 4, 5 (cit. on p. 17).

[38] Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. «Learning to Compose Dynamic Tree Structures for Visual Contexts». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 18).

[39] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. «Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks». In: *ArXiv* (2015) (cit. on p. 19).

[40] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. «Scene Graph Generation by Iterative Message Passing». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3097–3106 (cit. on p. 19).

[41] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. «Unbiased Scene Graph Generation From Biased Training». In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 3713–3722 (cit. on p. 21).

[42] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018 (cit. on p. 21).

[43] Ji Zhang, Kevin J. Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. «Graphical Contrastive Losses for Scene Graph Parsing». In: *CVPR*. 2019 (cit. on p. 23).

[44] Lin Li, Jun Xiao, Hanrong Shi, Wenxiao Wang, Jian Shao, An-An Liu, Yi Yang, and Long Chen. «Label Semantic Knowledge Distillation for Unbiased Scene Graph Generation». In: *IEEE Transactions on Circuits and Systems for Video Technology* (2023) (cit. on p. 24).

[45] Kanghoon Yoon, Kibum Kim, Jinyoung Moon, and Chanyoung Park. «Unbiased Heterogeneous Scene Graph Generation with Relation-aware Message Passing Network». In: *AAAI*. 2023 (cit. on p. 25).

[46] Yuxiang Zhang, Zhenbo Liu, and Shuai Wang. «SRTR: Self-Reasoning Transformer with Visual-Linguistic Knowledge for Scene Graph Generation». In: (Dec. 2022). arXiv preprint. arXiv: `2212.09329 [cs.CV]` (cit. on p. 26).

[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. «Deformable DETR: Deformable Transformers for End-to-End Object Detection». In: *arXiv preprint arXiv:2010.04159* (2020) (cit. on p. 26).

[48] Alec Radford, Karthik Narasimhan, Tim Rocktäschel, Lenny Wu, Jong Wook Kim, and Ilya Sutskever. «Learning Transferable Visual Models From Natural Language Supervision». In: *arXiv preprint arXiv:2103.00020* (2021) (cit. on p. 27).

[49] Pengchuan Xiao et al. «PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving». In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3095–3101 (cit. on p. 28).

[50] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. «The Cityscapes Dataset for Semantic Urban Scene Understanding». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3213–3223 (cit. on p. 31).

[51] Yuren Cong, Michael Ying Yang, and Bodo Rosenhahn. «Reltr: Relation transformer for scene graph generation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023) (cit. on p. 37).

[52] Christopher Lang, Alexander Braun, and Abhinav Valada. «Contrastive Object Detection Using Knowledge Graph Embeddings». In: *ArXiv* abs/2112.11366 (2021) (cit. on p. 39).

[53] Alessandro Oltramari, Jonathan M Francis, Cory Andrew Henson, Kaixin Ma, and Ruwan Wickramarachchi. «Neuro-symbolic Architectures for Context Understanding». In: *Knowledge Graphs for eXplainable Artificial Intelligence.* 2020 (cit. on p. 39).

[54] Tommaso Carraro. *LTNtorch: PyTorch implementation of Logic Tensor Networks.* Version 1.0.0. Mar. 2022. DOI: `10.5281/zenodo.6394282`. URL: `https://doi.org/10.5281/zenodo.6394282` (cit. on p. 42).