Consistent Scene Graph Generation by Constraint Optimization

Boqi Chen
Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada
boqi.chen@mail.mcgil.ca

Kristóf Marussy
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
marussy@mit.bme.hu

Sebastian Pilarski
Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada
sebastian.pilarski@mail.mcgill.ca

Oszkár Semeráth
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
semerath@mit.bme.hu

Dániel Varró
Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada
daniel.varro@mcgill.ca

ABSTRACT

Scene graph generation takes an image and derives a graph representation of key objects in the image and their relations. This core computer vision task is often used in autonomous driving, where traditional software and machine learning (ML) components are used in tandem. However, in such a safety-critical context, valid scene graphs can be further restricted by consistency constraints captured by domain or safety experts. Existing ML approaches for scene graph generation focus exclusively on relation-level accuracy but provide little to no guarantee that consistency constraints are satisfied in the generated scene graphs. In this paper, we aim to complement existing ML-based approaches by a post-processing step using constraint optimization over probabilistic scene graphs that can (1) guarantee that no consistency constraints are violated and (2) improve the overall accuracy of scene graph generation by fixing constraint violations. We evaluate the effectiveness of our approach using well-known, and novel metrics in the context of two popular ML datasets augmented with consistency constraints and two ML-based scene graph generation approaches as baselines.

CCS CONCEPTS

• **Mathematics of computing** → *Computing most probable explanation*; • **Computing methodologies** → **Neural networks**; **Knowledge representation and reasoning**.

KEYWORDS

scene graph generation, constraint optimization, consistency constraints, machine learning, probabilistic logic programming

ASE '22, October 10–14, 2022, Rochester, MI, USA

ACM ISBN 978-1-4503-9475-8/22/10.
https://doi.org/10.1145/3551349.3560433

ACM Reference Format:
Boqi Chen, Kristóf Marussy, Sebastian Pilarski, Oszkár Semeráth, and Dániel Varró. 2022. Consistent Scene Graph Generation by Constraint Optimization. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3551349.3560433

1 INTRODUCTION

Context. In computer vision, scene graph generation (SGG) [13] takes a camera image as input and derives a scene graph with nodes representing relevant objects (with their attributes) and edges capturing key relations between objects. A safety-critical usage scenario for scene graphs is autonomous driving [89] where machine learning (ML) components are often used in combination with traditional software components (e.g., traffic simulators like Carla [26]). There are several robotics applications of SGG by combining automated software engineering and ML [48].

However, safety standards (such as [29, 30]) require justified evidence that safety requirements are satisfied. Many of such requirements can be formulated by domain experts as consistency constraints over scene graphs (e.g., using first-order logic or the Object Constraint Language [34]), and used in tools like Scenic for test scenario generation [31]. Similar consistency constraints have been actively used in model-driven engineering at design-time in complex modeling tools (e.g., Capella, Artop, SysML) and validation tools [41, 65] for checking design rules for digitalization.

Problem statement. In this paper, we define the problem of *consistent scene graph generation* where the derived scene graphs also need to satisfy a set of consistency constraints available as a priori domain knowledge. Our paper investigates to what an extent consistency can be ensured for SGG.

In an autonomous driving context, such constraints may include physical or geometrical constraints (e.g., an autonomous vehicle needs to be on the road), or domain-specific restrictions (e.g., turning counter-clockwise in a roundabout, traffic signs should be consistent). From a safety perspective, the precise boundaries of safe behavior need to be precisely defined [4], e.g., (1) ML components in

an autonomous vehicle should guarantee safe behavior when they operate in a *consistent* environment (e.g. by starting from a consistent training set), and (2) they should gracefully degrade when some constraints are violated by human drivers in the environment (e.g. by handling extreme cases later).

Most existing ML approaches for SGG (1) *ignore the handling of consistency constraints* with the following rationale: assuming that the real scene respects the constraints, if a scene graph can be accurately derived from an image, then all constraints will automatically be satisfied. (2) A few approaches [28, 72, 85] have attempted to *incorporate constraints into loss functions* to penalize cases during training where constraints are not satisfied. Alternatively, (3) *constraints can be incorporated by extra layers* in an ML architecture [40, 80], which turn hard constraints into soft constraints (by introducing differentiable functions) to solve an optimization problem that minimizes the number of constraint violations.

Existing research in SGG only provides a best effort solution to ensure constraint satisfaction where any constraint can be violated with a low (but non-zero) probability, which does not provide any strong consistency guarantees for unseen data points. Moreover, incorporating a new constraint may require manually redesigning the ML architecture and restarting the training from scratch.

Furthermore, existing metrics used by the ML community to assess the performance of SGG operate *on the element level* (e.g., triple based recall). They calculate the total number of correctly identified objects and relations across all scenes. However, there are *no metrics to measure the consistency* or *accuracy of entire scene graphs*. Thus, even if a SGG approach achieves very high recall, the derived scene graphs may still have a low score for consistency.

Objectives. The main objective of the current paper is to complement existing ML-based SGG techniques by post-processing their probabilistic output by constraint optimization techniques to derive scene graphs with guaranteed consistency. On a general level, we aim to demonstrate how formal reasoning techniques (widely used in automated software engineering) can bring benefits to the ML challenge of scene graph generation to ensure consistency.

Contributions. Given a domain specification (with a metamodel and a set of consistency constraints), this paper presents a novel technique to improve the performance of scene graph generation by enforcing domain-specific constraints on the output of a vision-based ML component. As a conceptual novelty, our approach takes *the most probable scene graph which satisfy all the constraints* while obtaining the probabilities from existing ML-based SGG approaches.

- We provide novel metrics for evaluating the consistency and semantic (graph-level) accuracy of existing SGG techniques.
- We describe the consistent scene graph generation challenge as a constraint optimization problem to ensure the satisfaction of domain-specific first-order logic constraints.
- We adapt various deep learning techniques that separate the SGG problem into object recognition, attribute extraction, and relation detection to obtain a probabilistic scene graph.
- Given a probabilistic scene graph as input obtained from any ML model for SGG, we propose a post-processing method to derive consistent scene graphs as most probable explanation by probabilistic logic programming.
- We evaluate our approach on two benchmarks (CLEVR [43] and Blocksworld [7]) frequently used for evaluating the effectiveness of various vision-based ML techniques, and provide an initial study on a real-world dataset [46].

Added value. A main benefit of our approach is that it can significantly improve the performance of various existing ML-based SGG techniques that provide probabilities for individual objects and relations in a post-processing step. Moreover, we provide theoretical guarantees and empirical evidence for full (100%) consistency.

2 BACKGROUND

2.1 Vision based scene graph generation

Scene graph generation (SGG) is a classical task in computer vision. Given an image, SGG aims to extract key objects on the image, the attributes of the objects, and relations between objects as a graph.

Scene graphs. The structure of a scene graph can be defined by modeling techniques, e.g. a metamodel or a type graph. To simplify the presentation, let $\mathbf{R}$ denote a finite set of binary relation symbols (types) between objects, and $\mathbf{A}$ be a finite set of attribute symbols (types) while object types are handled as special attributes. let $V : \mathbf{A} \to \mathbf{V}_a$ map an attribute type $a \in \mathbf{A}$ to its finite possible values $\mathbf{V}_a$. In this paper, we focus only on discrete-valued attributes.

A scene graph $SG = (\mathbf{N}, \mathbf{E}, attr)$ can be represented as labeled and attributed graph [44] where (1) $\mathbf{N}$ is the set of objects (or nodes), (2) $\mathbf{E} : \mathbf{N} \times \mathbf{R} \times \mathbf{N}$ is the set of relations (or edges) labeled from $\mathbf{R}$. As a notational shortcut, $\overrightarrow{st}^r$ captures a relation of type r from node s to t, i.e., we write $\overrightarrow{st}^r \in SG$ iff $\langle s, r, t\rangle \in \mathbf{E}$. The notation $\bar{r}$ will denote the complement of r, i.e., $\overrightarrow{st}^{\bar{r}} \in SG$ iff $\overrightarrow{st}^r \notin SG$. Furthermore, (3) $attr\colon (\mathbf{N} \times \mathbf{A}) \to \mathbf{V}_a$ is a mapping where $\overrightarrow{nv}^a$ means object n has value v of attribute type a, i.e., we write $\overrightarrow{nv}^a \in SG$ iff $attr(n, a) = v$.

Scene graph generation. Existing SGG approaches fall into three main categories (see also Section 7). Here, we provide a high-level overview of *multi-step generation*, which is more data- and memory-efficient [88]. Multi-step SGG consists of three components, namely, object detection, attribute detection, and relationship extraction [62, 88], which are all neural networks trained individually.

During *training*, the object detector is trained first with the training images while using object masks (i.e. a binary matrix where 1 means that a particular pixel is part of the object) as ground truth. Then the attribute detector and relationship extractor are trained from the images using attributes and relationships as ground truth, respectively, while the object detector provides the predicted object masks for the input images. This can provide better approximations for a scene when the ground truth is no longer available.

During *operation* (i.e. actual scene graph generation), the prediction is similar to training: (1) The object detector predicts the object masks. (2) The masks are fed to the attribute and relation extractor to predict the output scene graph. (3) Since the predicted attributes and relations are probability distributions over the possible values, a probabilistic scene graph is first obtained as the output. (4) The final scene graph G_{ml} is chosen by maximum likelihood decision, i.e., by selecting the most probable value individually for each attribute and relation prediction.

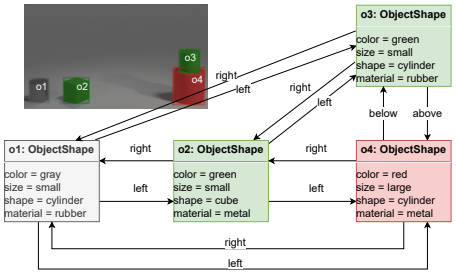


Figure 1: Sample scene with scene graph from Blocksworld

Running example. Blocksworld [7] is used as a running example in the paper, which is a popular ML dataset with basic object shapes (such as cube, sphere and cylinder). Each object has the same attributes $\mathbf{A} = \{\textbf{Shape}, \textbf{Color}, \textbf{Material}, \textbf{Size}\}$. Each attribute type has a finite set of possible values, e.g., $\mathbf{V}_{\text{"Size"}} = \{\text{"small"}, \text{"large"}\}$. Objects can be placed on top of another object to form a tower, and there are multiple of such towers in an image. Altogether, the objects placed on towers form the four basic relationships in this domain $\mathbf{R} = \{\textbf{Left}, \textbf{Right}, \textbf{Above}, \textbf{Below}\}$. Naturally, certain relations may correlate with other relations. For example, two objects stacked on each other should both be left/right from a third object. Moreover, an attribute value may also constrain the relations of an object, e.g., no objects can be stacked on top of a sphere. Figure 1 shows a sample image from Blocksworld together with its scene graph. Note that Blocksworld is derived from the CLEVR dataset [43], which will serve as a further case study in Section 5.

2.2 Hard constraints

In addition to core attributes and relations, a scene graph specification can also include a set of domain-specific hard constraints Φ that all valid objects, attributes and relations must satisfy in each and every scene graph derived from an image. In automotive design practice, similar constraints are captured by domain experts and checked by validation tools [41, 65]. Metamodel and constraints (captured in various specification languages) has been used in research on testing autonomous vehicles [1, 2, 8].

Constraints formulation. In this paper, we assume that all such hard constraints can be represented as first-order logic (FOL) formulae, which can normally be derived from high-level constraint languages (like OCL or VIATRA-Query) [10, 67]. Domain-specific atomic predicates in these formulae are derived in accordance with the relations and attributes of the metamodel, e.g., the mapping of symbols from scene graph to FOL is defined as: $\overrightarrow{st}^r \rightarrow \mathbf{r}(s,t)$, $\overrightarrow{nv}^a \rightarrow \mathbf{a}(n,v)$, $attr(n,a) \rightarrow \mathbf{a}(n)$. Complex FOL formulae are then assembled using the regular Boolean operations ($\wedge, \vee, \neg, \iff$) where quantifiers ($\exists, \forall$) are ranging over objects. For example, $\textbf{Color}(o1, \text{"red"}) \wedge \textbf{Right}(o1, o2)$ specifies there exists an object $o1$ which has a red color and is right to some other object $o2$. We classify domain-specific constraints in the Blocksworld (Figure 1) into geometric (physical) constraints and logical constraints.

Geometrical and physical constraints. Physical and geometrical constraints capture rules that need to be satisfied in order to obey the laws of physics. While in our work, we primarily focus on physical constraints for relations, attribute constraints can be incorporated similarly. A sample geometrical constraint captures that if an object $o1$ is left (right) from another object $o2$, then $o2$ must be right (left) from $o1$, which can be expressed in FOL as $\forall o1, o2 : \textbf{Left}(o1, o2) \iff \textbf{Right}(o2, o1)$. As a sample physical constraint, one can formulate that a sphere cannot hold anything on top of it, formally $\forall o1 : \textbf{Shape}(o1, Sphere) \Rightarrow \neg\exists o2 : \textbf{Above}(o2, o1)$. Altogether, we identified a total of 10 constraints for Blocksworld.

Metric Type	Formula	Metrics
Recall-based	$\frac{size(\mathbf{T}_{pred} \cap \mathbf{T}_{gt})}{size(\mathbf{T}_{gt})}$	SGGen [52], SGGen+ [86]
Scene Accuracy	$\frac{\sum_{i=0}^{N} \mathbf{1}(\mathbf{S}_{pred}[i] \cong \mathbf{S}_{gt}[i])}{N}$	SA
Consistency	$\frac{\sum_{i=0}^{N} \mathbf{1}(\forall \phi \in \Phi: \mathbf{S}_{pred}[i] \models \phi)}{N}$	Con

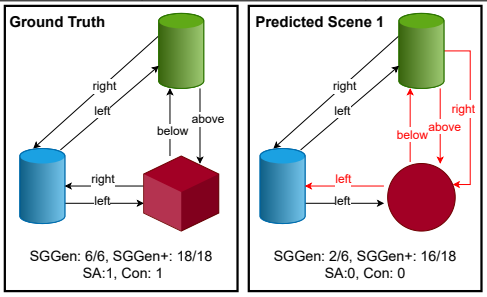


Figure 2: Metrics: ground truth vs. predicted scene graph

Logic constraint. Additional logic constraints may further restrict the set of consistent scene graphs (once physical constraints are already satisfied). In a real scenario, such logic constraints are formulated by experts from traffic rules and safety regulations. To illustrate the effects of such external regulations, we identified and included four constraints for our *Blocksworld* running example. In addition to the fulfillment of physical constraints, these constraints also need to be satisfied by valid towers of objects. The set of logic constraints used in our *Blocksworld* running example is provided in Table 1. For example, a logic constraint expresses that an object cannot hold a larger object on top of it.

3 METRICS FOR SCENE GRAPH GENERATION

The performance of SGG is usually assessed using various metrics that evaluate how many relations are correctly identified by SGG. Moreover, we also propose novel metrics that measure performance on the entire scene level (see Figure 2 for an overview of all metrics).

Recall-based metrics. These metrics are frequently used in evaluating SGG tasks on real-world (or photorealistic) scene images where exhaustive annotations for all objects that appear in such images are hard to obtain [52]. Such metrics (Row 1 in Figure 2) measure how many ground truth triples appear in the predicted scene, as the ratio of triples appearing both in the prediction $\mathbf{T}_{gt}$ and the ground truth $\mathbf{T}_{gt}$ with respect to the number of triples in the ground truth $\mathbf{T}_{gt}$. As such, this metric can ignore irrelevant (or too small) objects in the background of an image. Various metrics in this category differ on how the triples are collected and matched.

SGGen recall [52] is a widely adopted metrics based on counting the number of ground truth ⟨*subject, predicate, object*⟩ triples (i.e. relations) recognized in SGG. A predicted triple matches the ground

C1:An object cannot hold another object larger than it $\forall x, y : \textbf{Size}(y, \text{"Small"}) \wedge \textbf{Above}(x, y) \Rightarrow \textbf{Size}(x, \text{"Small"})$
C2: A large cube has to hold something on top of it $\forall x : \textbf{Size}(x, \text{"Large"}) \wedge \textbf{Shape}(x, \text{"Cube"}) \Rightarrow \exists y : \textbf{Above}(y, x)$
C3: Two neighouring stacks bottom of the stacks cannot have the same color $\forall x, y : \textbf{Bottom}(x) \wedge \textbf{Bottom}(y) \wedge \textbf{next}(x, y) \Rightarrow \textbf{Color}(x) \neq \textbf{Color}(y)$ where $\textbf{Bottom}(x) := \neg\exists y : \textbf{Below}(y, x)$ $\textbf{leftNext}(x, y) := \textbf{Left}(x, y) \wedge \neg(\exists z : \textbf{Left}(x, z) \wedge \textbf{Left}(z, y))$ $\textbf{next}(x, y) := \textbf{leftNext}(\mathbf{x}, \mathbf{y}) \vee \textbf{leftNext}(y, x)$
C4: Each yellow object there has to have something on the right $\forall x : \textbf{Color}(x, \text{"Yellow"}) \Rightarrow \exists y : \textbf{Right}(y, x,)$

(a) Blocksworld constraints

C1: Between any pairs of objects with the same color, there must be an object with a different color placed in between horizontally $\forall x, y : \textbf{Color}(x) = \textbf{Color}(y) \Rightarrow \exists z : (\textbf{Color}(z) \neq \textbf{Color}(x)) \wedge \textbf{inBetween}(z, x, y)$, where: $\textbf{inBetween}(z, x, y) := (\textbf{Left}(x, y) \wedge \textbf{Left}(x, z) \wedge \textbf{Left}(z, y)) \vee \textbf{Left}(y, y) \wedge \textbf{Left}(y, z) \wedge \textbf{Left}(z, x))$
C2: A metal object cannot be behind a rubber object $\forall x : \textbf{Material}(x) = \text{"Metal"} \Rightarrow (\forall y : (\textbf{Material}(y) = \text{"Rubber"}) \Rightarrow \textbf{Behind}(y, x))$
C3: For each large cube, there must be a small cylinder behind it $\forall x : (\textbf{Shape}(x) = \text{"Cube"} \wedge \textbf{Size}(x) = \text{"Large"}) \Rightarrow \exists y : \textbf{Size}(y) = \text{"Small"} \wedge \textbf{Shape}(y) = \text{"Cylinder"} \wedge \textbf{Behind}(y, x)$
C4: If there are a cyan metal and a red sphere object, then there cannot another object behind them. $\exists x, y : \textbf{Color}(x) = \text{"Cyan"} \wedge \textbf{Material}(()x) = \text{"Metal"} \wedge \textbf{Color}(y) = \text{"Red"} \wedge \textbf{Shape}(y) = \text{"Sphere"} \Rightarrow \neg\exists z : z! = x, z! = y \wedge (\textbf{Behind}(z, x) \vee \textbf{Behind}(z, y))$

(b) CLEVR constraints

Table 1: Domain-specific logic constraints of Blocksworld and CLEVR

truth if it detects the correct predicate (relation) and correctly recognizes the two objects. For an object to be correctly recognized, SGG needs to provide the correct attributes of the object and localize the object in the image. IoU (Intersection over Union) is used to measure the localization of objects, which calculates the fraction of the overlap between the ground truth object mask and predicted object mask over the union of the two. The predicted object matches the ground truth if the IoU value is over a certain threshold.

Note that **SGGen** significantly penalizes object mismatch, e.g. if an object is incorrectly identified, all of its adjacent triples are considered to be incorrect. To alleviate this effect, **SGGen+** [86] separates the relation triples with object detection. With this separation, relation triples only check the localization of objects and object types and attributes are considered additional relation triples.

Scene accuracy. Triple-based metrics help evaluate the overall performance of SGG. However, these metrics do not always reflect the SGG performance on a single scene level. When using SGG in a safety-critical scenario, it is much more important to predict if the output of SGG can be fully trusted for a particular scene, i.e. whether *all relations* in a given scene are correctly identified. For that purpose, we propose novel *scene-level metrics* for SGG.

Scene Accuracy (SA) measures the proportion of predicted scenes that completely match a ground truth scene. Let S_{pred} and S_{gt} be the set of predicted and ground truth scene graphs, respectively. Then SA is the fraction of the number of predicted scenes isomorphic to the ground truth (using $\cong$) over the total number of scenes N (Row 2 in Figure 2) where $\mathbf{1}$ is the indicator function (which evaluates to 1 for graphs where a particular condition holds). We assume that a node matcher (e.g. the object mask) pairs the nodes of the two scenes such that graph isomorphism can be checked by validating the appropriate relations between the nodes.

Furthermore, in principle, the predicted scenes should satisfy all hard constraints. We propose to measure the level of **Consistency (Con)**, i.e. the number of scenes that fully satisfy all hard constraints (regardless of whether the scene matches the ground truth or not). Let $\mathbf{S}_{pred}[i] \models \varphi$ denote that the predicted scene satisfies a constraint φ, then the **Con** metric (shown in Row 3 of Figure 2) counts the ratio of such scenes.

Example 1. Figure 2 shows an example evaluation of metrics for a predicted scene graph (right) against the ground truth (left). We assume that (1) all ground-truth objects are localized correctly in the predicted scene graph using object masks, and (2) the material and size of all objects are correct and only focus on shape and color.

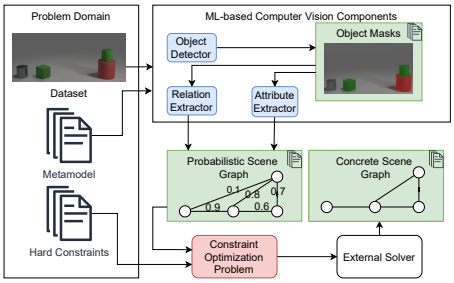


Figure 3: Overview of the approach

We can calculate different metric scores by comparing the ground truth with predicted scene. First, since the red object has a wrong shape in the predicted graph, all relations are considered wrong in SGGen. Thus, the predicted scene graph has 2/6 relations correctly. In SGGen+, we only consider object masks in the relation paring and add attribute tuples in the measurement. We have 6 relation triples and 12 attribute tuples in the ground truth scene graph. Compared to the ground truth, the predicted scene graph has 1 incorrect relation and 1 incorrect attribute, so it has an SGGen+ score of 16/18. Finally, the scene accuracy is 0, and the scene also violates the hard constraints (along the highlighted edges).

4 APPROACH

4.1 Preliminaries: MPE and MAXSAT

In probabilistic logic programming, given a set of probabilistic facts (i.e., a truth assignment valued random variable) F and evidences/constraints E, the *most probable explanation* [61] $MPE(F, E) = \arg\max_f P(F = f \mid F \models E)$ finds the most likely realization f of F that satisfies E, where $P(\cdot \mid \cdot)$ denotes conditional probability.

Example 2. In the context of our running example, we let $P(F \models \textbf{Right}(o_1, o_2)) = 0.3$, $P(F \models \textbf{Left}(o_1, o_2)) = 0.8$, and $E = \{\forall X, Y : \textbf{Right}(X, Y) \Rightarrow \textbf{Left}(Y, X) \wedge \neg\textbf{Right}(Y, X)\}$.
In this case, the MPE is $f = \{\textbf{Left}(o_1, o_2), \textbf{Right}(o_2, o_1)\}$.

Maximum satisifiability (MAXSAT) is an optimization problem aiming to find the maximum subset of clauses which can be satisfied in a Boolean formula in a conjunctive normal form (CNF). When some clauses have weights, it is a weighted partial MAXSAT. Formally, given a set of clauses C_i with weights w_i and a set of hard

constraints Φ, the weighted partial MAXSAT is formulated as:

$$\begin{aligned} &\neg\phi : -\infty \text{ for each } \phi \in \Phi \\ &C_i : w_i \text{ for } i \in [0, n] \\ &\max_x \sum_i w_i \cdot \mathbf{1}(x \models C_i) \text{ subject to } x \models \phi \text{ for each } \phi \in \Phi \end{aligned} \tag{1}$$

A solution to this problem satisfies all clauses with $-\infty$ weights while maximizing the weights of remaining clauses. In the rest of the paper, MAXSAT refers to the weighted partial variant.

To handle FOL constraints Φ, one can eliminate the quantifiers by *instantiating* (unfolding) the constraints with the (finite-domain) attribute and relation variables to obtain their CNF [75].

Example 3. $\Phi = E$ from Example 2 unfolds into $\Phi' = \{\neg\mathbf{Right}(o_1, o_2) \vee \mathbf{Left}(o_2, o_1),\ \neg\mathbf{Right}(o_1, o_2) \vee \neg\mathbf{Right}(o_2, o_1), \neg\mathbf{Right}(o_2, o_2) \vee \mathbf{Left}(o_1, o_2), \neg\mathbf{Right}(o_2, o_2) \vee \neg\mathbf{Right}(o_1, o_2)\}$.

4.2 Overview

A main conceptual contribution of our paper is to provide consistent scene graph generation by augmenting the output of traditional ML-based SGG approaches (which do not handle hard constraints) with reasoning as a MAXSAT problem.

Problem formulation. Given an image and a set of constraints Φ, we formulate *consistent scene graph generation* as an optimization problem to find a ML model $\mathcal{M}$ that maximizes the probability of the predicted scene graph SG being isomorphic with the ground truth SG_{gt} while ensuring that all hard constraints $\phi \in \Phi$ hold:

$$\max_{\mathcal{M}} P(SG \cong SG_{gt} \mid \mathcal{M}) \text{ subject to } SG \models \Phi. \tag{2}$$

Unfortunately, the objective function in Equation 2 cannot be optimized directly, since the distribution of the ground truth SG_{sg} is not known in advance. Instead, given a training set of size N, the probability in the objective function can be approximated with the use of *scene accuracy* as

$$\max_{\mathcal{M}} \frac{\sum_{i=1}^{N} \mathbf{1}(SG[i] \cong SG_{gt}[i])}{N} \text{ subject to } \forall i\colon SG[i] \models \Phi. \tag{3}$$

The solution to this optimization problem corresponds to a model $\mathcal{M}$ which outputs the *most likely scene subject to the hard constraints*. For that purpose, we complement existing SGG techniques with probabilistic logic reasoning to ensure that consistent scene graphs are derived. Our approach handles domain-specific constraints captured in FOL while treating all constraints as hard constraints. As such, a conceptual benefit of our approach (compared to purely ML-based SGG approaches) is that our scene graphs will have absolutely no constraint violations. This provides a very strong guarantee for safety-critical applications such as autonomous vehicles.

Architecture. Figure 3 shows an overview of our approach. The input for a *problem domain* includes *datasets* (i.e. images with ground truth annotations) for training and testing of vision-based ML components, a *metamodel* (or type graph) to describe the structure of scene graphs and a set of *hard constraints* that needs to be satisfied by all consistent scene graphs of the domain. After successful training, our framework derives a consistent scene graph capturing the key objects and relations visible on an image.

Given the individual probabilities of relations, the essence of our approach is to generate the most likely graph subject to hard constraints. To achieve this goal, we formulate this generation as a *weighted partial MAXSAT* from the probabilities. Finally, we use an *external solver* to solve this MAXSAT using constraint optimization in order to create a (concrete) scene graph, which is guaranteed to satisfy all hard constraints.

While this paper focuses exclusively on multi-step scene graph generators, our approach can be integrated with other ML-based scene graph generation techniques that provide probabilities for relations (triples), which is the case for most SGG models [13].

4.3 Extracting scene graphs

Multi-step scene graph generation involves three components: *object detector*, *attribute extractor* and *relationship extractor*.

Object detector. Given an image, the object detector aims to derive a set of objects in the image together with location defined as bounding boxes [63] or object masks [37]. In this paper, we rely on [88], which uses Mask-RCNN [37] to derive object masks from the images. This component only detects the existence of an object, while object attributes are extracted separately.

Attribute extractor. Given the original image and the object masks from the object detector, the attribute extractor identifies attributes, such as the size, shape, and color of an object. In this paper, we rely on [88] which uses a convolutional neural network (CNN) to extract the latent features of each object by combining the original image and the object mask. The object feature is provided to a feed-forward neural network (FNN) to perform classification on discrete attributes and regression on continuous attributes.

Relation extractor. Given the original image and the object masks from the object detector as input, the relation extractor derives relations between a pair of objects as output. First, we extract latent object features with CNN, similar to the attribute extractor. Then the object features for a pair of objects are concatenated as a single feature vector to an FNN. The FNN performs classification and gives an output probability for each relation type, representing if the pair of objects have that relation type.

Formalization. Given a problem domain with attribute types $\mathbf{A}$, attribute domains V and relation types $\mathbf{R}$, we assume that computer vision components provide a probabilistic graph $\mathbb{G} = (\mathbf{N}, P_R, P_A)$ as output where $\mathbf{N}$ is the set of nodes, while P_R and P_A assign probabilities to attributes and relations. Note that $\mathbb{R}$ contains each relation symbol $r \in \mathbf{R}$ and its complement type $\overline{r}$ where $\overrightarrow{n_1 n_2^{\overline{r}}}$ means that no edge $\overrightarrow{n_1 n_2^{r}}$ of type r leads from nodes n_1 to n_2.

- $P_R\colon (\mathbf{N} \times \mathbb{R} \times \mathbf{N}) \to [0, 1]$ gives the probability that an edge $\overrightarrow{n_1 n_2^{r}} \in \mathbb{E}$ of type $r \in \mathbb{R}$ is leading from node n_1 to node n_2 where $P_R(\overrightarrow{n_1 n_2^{r}}) + P_R(\overrightarrow{n_1 n_2^{\overline{r}}}) = 1$.
- $P_A\colon (\mathbf{N} \times \mathbf{A} \times \mathbf{V_a}) \to [0, 1]$ gives the probability that the value of an attribute $a \in \mathbf{A}$ at a node $n \in \mathbf{N}$ is $v \in \mathbf{V_a}$ such that $\sum_{v \in V_a} P_A(n, a, v) = 1$.

Informally, we ensure that the probabilities of all possible values for an attribute of an object sum up to 1, and the probabilities of an edge and its complement edge also add up to 1.

In multi-stage generation, object detection derives $\mathbf{N}$, attribute extraction gives P_A and relation extraction yields E and P_R.

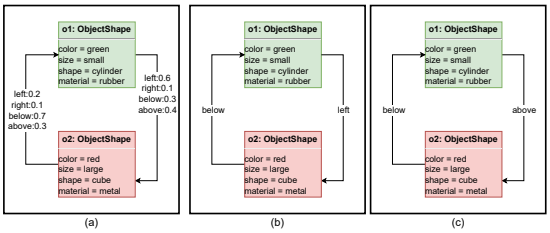


Figure 4: (a) a probabilistic graph $\mathbb{G}$, (b) the maximum likelihood graph G_{ml}, (c) graph G produced by our approach

Example 4. The left part of Figure 4 shows an extract of a probabilistic scene graph with two objects from Blocksworld. For simplicity, we only depict probabilities for relations P_R (next to the relation type), but omit $\bar{r}$. Furthermore, we assume that $P_A = 1$ for all attributes that appear in the diagram. For example, $P_R(\overrightarrow{o_1 o_2}^{\textbf{above}}) = 0.4$, $P_R(\overrightarrow{o_2 o_1}^{\textbf{below}}) = 0.7$, $P_A(\overrightarrow{o_1 \text{"red"}}^{\textbf{color}}) = 1$.

4.4 Constraint optimization problem

ML components used for scene graph extraction are trained to maximize the probability of generating scenes that are close to the ground truth while disregarding the constraints. Given such an ML component that outputs a probabilistic graph $\mathbb{G}$, we approximate $P(SG \mid \mathbb{G}) = P(SG \cong SG_{gt} \mid \mathcal{M})$ in (2) with a Naive Bayes assumption (i.e., each value in P_A and P_R is independent) as

$$\log P(SG \mid \mathbb{G}) = \sum_{\overrightarrow{n_1 n_2}^r \in SG} \log P_R(\overrightarrow{n_1 n_2}^r) + \sum_{\overrightarrow{n v}^a \in SG} \log P_A(\overrightarrow{n v}^a). \quad (4)$$

Finding the most likely scene graph SG subject to hard constraints Φ constitutes a MPE inference task $SG = MPE(P(\cdot \mid \mathbb{G}), \Phi)$. Compared to traditional *MPE*, where the probabilistic model P is fixed and the constraints Φ serve as a *query* for which evidence shall be provided, we use a *prescribed* Φ and extract P from each input image using vision-based ML components.

Along with [58], we solve the *MPE* task by deriving a weighted partial MAXSAT problem. We introduce three types of weighted clauses (with $x_{\overrightarrow{n v}^a}$ / $x_{\overrightarrow{n_1 n_2}^r}$ as attribute / edge variables) according to the the hard constraints Φ and the probabilistic graph $\mathbb{G}$:

$$\begin{aligned} \neg\varphi &: -\infty \text{ for each } \varphi \in \Phi, \\ x_{\overrightarrow{n v}^a} &: \log P_R(\overrightarrow{n v}^a) \text{ for each } \overrightarrow{n v}^a \in \mathbb{G}, \\ x_{\overrightarrow{n_1 n_2}^r} &: \log P_E(\overrightarrow{n_1 n_2}^r) \text{ for each } \overrightarrow{n_1 n_2}^r \in \mathbb{G}. \end{aligned} \quad (5)$$

We eliminate the quantifiers in the constraints with unfolding (see Example 3). Moreover, we extend Φ with constraints over attribute and relation variables such that: (a) exactly one of $x_{\overrightarrow{n_1 n_2}^r}$ and $x_{\overrightarrow{n_1 n_2}^{\bar{r}}}$ is true for each relation r and pair of objects n_1, n_2, and (b) exactly one of $x_{\overrightarrow{n v_1}^a}, \ldots, x_{\overrightarrow{n v_k}^a}$ holds for each attribute a with domain $\mathbf{V}_a = \{v_1, \ldots, v_k\}$ and object n.

We use log probability as the weights for the attribute and relations, so that the sum of the weights corresponds to likelihood $P(SG \mid \mathbb{G})$ of the concrete graph SG from (4).

Finally, we express the MAXSAT problem derived from a probabilistic graph representing a scene as a linear constraint optimization problem [5, 33], and we use the external Gurobi solver [36] to obtain a concrete scene graph as solution.

Example 5. Given a probabilistic scene graph in Figure 4(a), and a set of constraints $\{\forall X, Y : \textbf{Above}(X, Y) \iff \textbf{Below}(Y, X)\}$, an *extract* of the corresponding weighted partial MAXSAT problem is:

$$(\neg x_{\overrightarrow{o_1 o_2}^{\text{below}}} \vee \neg x_{\overrightarrow{o_1 o_2}^{\text{above}}}) \wedge (x_{\overrightarrow{o_1 o_2}^{\text{below}}} \vee x_{\overrightarrow{o_1 o_2}^{\text{above}}}) : -\infty$$
$$x_{\overrightarrow{o_1 o_2}^{\text{above}}} : 0.4, x_{\overrightarrow{o_2 o_1}^{\text{below}}} : 0.7, x_{\overrightarrow{o_1 \text{"red"}}^{\text{color}}} : 1$$

Example 6. Figure 4 (b) and (c) shows solutions obtained using maximum likelihood and our approach. The former (b) produces a graph by individually choosing the most likely edge or attribute, in our case, $\overrightarrow{o_1 o_2}^{\textbf{Left}}$ and $\overrightarrow{o_2 o_1}^{\textbf{Below}}$. However, this graph violates the geometric constraints that left/right and above/below should be opposite edges, as well as the domain-specific constraint **C3** in 1. Using our approach (c), the derived graph has no constraint violations and it will be also closer to the ground truth.

Our approach provides two theoretical guarantees formalized in Theorem 1: (1) a solution scene graph will always satisfy all constraints, and (2) the scene accuracy with respect to the ground truth is at least as good as what is provided by the ML model.

Theorem 1. *Let there be an ML algorithm for SGG that takes an image as input and derives a probabilistic scene graph $\mathbb{G}$, and let G_{ml} be the maximum likelihood graph derived from $\mathbb{G}$. Furthermore, given a set of constraints Φ, if the optimization problem described above has a concrete scene graph G as a solution, then G will satisfy:*

1. *$\forall \varphi \in \Phi : G \models \varphi$ (i.e. G is consistent);*
2. *$\boldsymbol{SA}(G, G_{gt}) \geq \boldsymbol{SA}(G_{ml}, G_{gt})$, where G_{gt} is the ground truth, i.e., our approach is* hippocratic *[69] as it does not change valid scene graphs.*

We provide the proof in the supplementary material.

5 EVALUATION

We conduct experiments to address four research questions:

- **RQ1**: What is the impact of using a constraint-biased data set for training?
- **RQ2**: How effective is our constraint optimization approach in deriving consistent scene graphs?
- **RQ3**: How does the generation of consistent scene graphs affect other metrics with different training set size?
- **RQ4**: How does the runtime of optimization scale with increasing number of objects and constraints?

5.1 Setup

Target domains. We evaluate our approach in the context of two synthetic datasets frequently used to evaluate computer vision models for SGG to address the above research questions.

CLEVR [43] is a visual reasoning dataset with simple geometric shapes placed on a plane, and questions about the scene. Each scene contains 3-10 objects with different shape, material, color, and size attributes. The benchmark has been designed to evaluate the effectiveness of ML models on visual question answering tasks, and it has also been used in image modification [50] and image generation [50, 66]. We also evaluate our approach on the Blocksworld dataset (**BLOCK**) introduced as the running example in Section 2.

The publicly available generator of each datasets derives random scenes, renders their image, and creates a scene graph with precise

object attributes, coordinates and relations as ground truth. The default configuration of the generators is used to derive new scenes for the evaluation: the resolution of each image is 480×320, and each scene contains 3-10 objects. Furthermore, we extended the tool chain to produce the instance segmentation [22], and specified 4 extra constraints to be enforced in addition to the geometric and physical constraints already present in each dataset (see Table 1).

Compared ML approaches. We evaluate the effectiveness of our approach compared to two different baseline ML approaches often used in the context of SGG. The two baselines rely on different information to derive a scene graph from an image. They represent the common view that (a) if each real scene (thus each image) used for training respects the hard constraints, and (b) if scene graphs can be accurately derived for each image, then (c) the constraints will also be satisfied when evaluated on the derived scene graphs.

RM: The *relational* multi-stage SGG approach discussed in Section 4.3 is used as a first baseline (without constraint optimization). This case represents a realistic setup for using SGG in autonomous driving when relations are annotated in the training set.

PM: The second baseline is a *physical* SGG approach where the exact coordinates of each object are available for training as extra information in the form of attributes. Thus, the training only consists of object detection and attribute extraction, while relations are automatically derived by evaluating physical and geometrical constraints over the coordinates. This approach has been used by [88] for **CLEVR**. Such precise coordinates are rarely available in training sets for autonomous driving scenarios (e.g. [71]), but this is still a realistic setting in other SGG scenarios [78].

Our constraint optimization based approach (**OP**) is then integrated with both the **RM** and **PM** ML components resulting in two further approaches to compare, namely, **RM**+**OP** and **PM**+**OP**.

Compared metrics. We aim to compare the effectiveness of the various approaches using the metrics introduced in Section 2, which includes recall-based (global) metrics, such as **SGGen** and **SGGen+**, and novel scene-based metrics **SA** and **Con**. During evaluation, a predicted object is considered to match the ground truth object if their intersection of union (IoU) value exceeds 0.5.

The recall-based SGG metrics originate from information retrieval tasks where *recall@K* is evaluated, i.e. the recall is measured on the top-K most probable relation triples. However, this paper aims to directly measure the difference between the generated scene graph and the ground truth, thus we do not explicitly rank the triples but consider *all* triples retrieved by the model.

5.2 RQ1: Constraint-biased training set

Rationale. While logic constraints in SGG are restrictions that should be obeyed, inconsistent scenes may appear in the training data (e.g. unlawful behavior by the human drivers) in real world scenarios. We call a dataset *constraint biased* if all data samples satisfy the logic constraints of the domain. In this RQ, we aim to measure the effect of inconsistent scenes in the dataset for training.

Setup. To generate an unconstrained training set (**-Cons**) for this experiment, we use the original generators. Then for a constraint-biased training set (**+Cons**), we rerun the generation and keep only data samples satisfying all logic constraints (including the extra constraints 1-4 in Table 1) while keeping the same distribution of images. In the end, we obtain a total of four training sets for the two domains; each containing 4000 scenes.

During an initial investigation with the constraint-biased data set (**+Cons**), we observed that certain implication constraints (of the form *premise* $\implies$ *consequence*) are trivially satisfied when a complex premise evaluates to false. For example, **C2** in **BLOCK**, which requires that a large cube must have something on top of it, is trivially satisfied if the scene contains no large cubes. To better understand the role of logic constraints in the domains, we selected two constraints with a complex premise from each domain (**C2**, **C4** in **BLOCK** and **C3**, **C4** in **CLEVR**). Then, we also generated test sets for each constraint such that all samples in the domain satisfied the *premises* of that constraint while still ensuring consistency.

We explore the influence of different training set on the two baseline ML approaches **RM** and **PM**. We train each component to near convergence: 5 epochs for object detector, 50 for attribute extractor and 24 for relation extractor. We also include a complete configuration for training in the supplementary materials. We train for each configuration multiple times and keep the best model. We keep the object detector the same for both approaches to allow better focus on the quality of relation and attribute extraction.

Analysis of results. Table 2 show different metrics (in percentage) for the two ML approaches trained with unconstrained and constraint-biased datasets. The different metrics in columns are grouped by the test sets. For example, **Cons** is a constraint-biased dataset and **Cons(C2)** is a constraint-biased dataset in which all data points satisfy the premise of constraint **C2**. The rows show the performance of **RM** and **PM** using the different training sets.

Overall, **PM** model performs better than **RM**. The improvement mainly comes from incorporating physical coordinates in training. However, we acknowledge that physical coordinates cannot capture all relations and they are not commonly available in real-world datasets. Thus, we focus on the comparison of the different setups for **RM** and **PM** rather than directly comparing the two approaches.

We carry out statistical analysis to compare the performance in each setup. We group the generated scenes into 200-sample subgroups and calculate the metrics value for each subgroup. Then we calculate the Cohen-d effective size [18] of **(+Cons)** and **(-Cons)** and highlight the ones (in bold) with larger than 0.8 Cohen-d value.

In general, the model trained with constrained-biased data performs better in the **BLOCK** dataset, dominating 9/12 cases for **RM**, while there is no statistically significant difference for **PM**. However, the model trained with an unbiased training set performs better in **CLEVR** (for 9/12 cases in **RM** 10/12 cases in **PM**).

We suspect that the difference comes from the various complexity of the two domains. The tower structure in **BLOCK** is harder to learn for **RM** compared to **CLEVR**, which results in lower overall accuracy. On the other hand, the model trained with an unbiased dataset achieves near-perfect performance on **CLEVR** (over 95% in **SA**) when the benefit of a constraint-biased training set is minimal.

RQ1: *Our results are mixed. A constraint-biased training set may improve the performance of a ML model when the base performance of the ML model trained on the unbiased training set of the task is low. However, such a constraint-biased training set may decrease performance if the base performance of the model is already high.*

Approach	BLOCK Test Set											
	Cons				Cons (C2)				Cons (C4)			
	SGGen	SGGen+	SA	Con	SGGen	SGGen+	SA	Con	SGGen	SGGen+	SA	Con
RM+CONS	**99.19**	**99.67**	**89.80**	92.50	**99.01**	**99.58**	**85.40**	92.85	**99.20**	**99.69**	**89.50**	95.85
RM-CONS	98.86	99.36	85.10	92.45	98.45	99.11	78.65	**94.85**	98.93	99.36	83.65	**96.90**
PM+CONS	99.63	99.97	98.75	99.95	99.63	99.98	98.65	100	99.43	99.97	98.40	100
PM-CONS	99.63	99.95	98.85	100	99.63	99.98	98.80	100	99.39	99.95	98.35	100

Approach	CLEVR Test Set											
	Cons				Cons (C3)				Cons (C4)			
	SGGen	SGGen+	SA	Con	SGGen	SGGen+	SA	Con	SGGen	SGGen+	SA	Con
RM+CONS	99.55	99.94	98.05	99.60	98.80	99.79	93.9	99.10	99.70	99.96	98.75	99.80
RM-CONS	**99.92**	**99.98**	**99.50**	99.75	**99.61**	**99.93**	**98.30**	99.30	**99.97**	**99.99**	**99.7**	99.95
PM+CONS	99.20	99.96	98.05	99.90	97.68	99.81	93.85	98.50	99.12	99.94	98.05	99.55
PM-CONS	**99.60**	**99.98**	**99.10**	99.90	**98.56**	**99.90**	**96.10**	98.25	**99.71**	**99.99**	**99.35**	**99.90**

Table 2: Comparison of two ML approaches on different test sets for BLOCK and CLEVR (values in % with two decimal places)

5.3 RQ2: Effectiveness for consistency

Rationale. In this RQ, we explore the impact of constraint optimization for restoring consistency. Thus, we will measure the ratio of relations and attributes misidentified by the base ML technique, and check how many of them can be fixed with logic reasoning. We investigate training sets of different size to assess how consistency decreases with less amount of data (e.g. in few shot learning [81]).

Setup. We train the ML models using the best performing training set for each domain (see the previous RQ). For training the models, the same initial weights are used, while we gradually increase the size of the training set (from 1000 images as 1/4 to 4000 images as 4/4). To avoid potential overfitting on smaller dataset size, we use 10% from the dataset as validation set and take the model performing best on the validation set for evaluation. Then we apply our approach on the probabilistic graph derived by the various ML models to fix the misidentified relations and attributes. We evaluate the consistency **Con** of the generated scenes for the baseline **RM** and **PM** and their fixed counterparts **RM**+**OP** and **PM**+**OP** using images of the **Cons** test set.

Analysis of results. The last columns in Table 3 show the **Con** values for each domain and approach (Table 3a for **BLOCK** and Table 3b for **CLEVR**). In the tables, we present the original metric and the metric obtained after logic reasoning **OP** (format: original metric → metric after logic reasoning).

While both **RM** and **PM** approaches achieve high consistency, full 100% consistency is achieved only in rare cases. As such, scenes derived by the ML models still dominantly have some constraint violations. We also observe that the level of consistency is particularly problematic for smaller training sets (83.15% in **BLOCK RM** and 65.40% in **CLEVR**). Moreover, there is no consistency guarantee for ML models in case of unseen data points.

On the other hand, the scenes produced with our approach **OP** achieved full consistency in all cases, regardless to the dataset size or base ML approach. Even when the base consistency is poor (65.40% in **CLEVR PM** with 1/4 dataset size), the scenes become fully consistent after the post-processing by **OP**. This empirical evidence reinforces the theoretical guarantee of Theorem 1; the scenes generated by **OP** will be consistent for unknown data points.

For our two case studies, solving the MAXSAT problem in **OP** has only minimal overhead in addition to the ML components. We could process more than 30 scenes per second on an average personal computer with a 3.80GHz CPU and 32GB RAM.

RQ2: *Despite a high-level of consistency, only very few scene graphs derived by baseline ML models **RM** and **PM** are fully consistent; likewise, they provide no consistency guarantees. Our approach **OP** derives fully consistent scene graphs using different ML models as basis with improvements up to 35% , and strong consistency guarantees even for unseen data points.*

5.4 RQ3: Effectiveness for performance metrics

Rationale. To assess the relevance of our approach from ML perspective, we explore (1) how fixing inconsistencies may affect other performance metrics, (2) how performance is influenced by the size of the training set. Such data efficiency is important in a real world setting as we may not a priori know how much data is required for training to achieve target performance. Thus, we measure performance along training sets of increasing size.

Setup. We use the same setup as in RQ2 to measure the recall-based metrics (**SGGen**, **SGGen+**), and the scene-based metric **SA**.

Analysis of results. The first three columns of each case study in Table 3 show the measured metric values of the different approaches using increasing training set size.

In general, performance improves across all metrics as dataset size increases, which is unsurprising as larger training sets tend to improve ML performance. In particular, **SA** is near 0 when trained on 1/4 of **CLEVR** data. We found the root cause of this observation: the ML model significantly overfit when producing the color attribute of objects. Thus, all metrics that require strict identity for objects (such as **SGGen** and **SA**) decrease significantly.

Due to property 2 of Theorem 1, **SA** with **OP** is as good as **SA** without **OP**. While we only have formal guarantees for **SA**, we observe that our approach improves the performance for all other metrics. Our approach increased the performance of the underlying ML model in 73% of cases and achieved the same performance in all other cases. The increase is particularly impressive for **RM**, with up to 3% in **SGGen**, 2% in **SGGen+** and over 20% in **SA**.

With sufficient amount of data, the base performance for **RM** is generally much lower than **PM**, which also uses precise physical

Size		BLOCK Test Set			
		SGGen	SGGen+	SA	Con
4/4	RM	98.41→99.65	99.19→99.97	82.50→98.85	90.90→100
	PM	99.42→99.42	99.96→99.96	98.20→98.20	100→100
3/4	RM	97.27→99.50	98.59→99.94	78.30→98.60	87.15→100
	PM	99.13→99.13	99.94→99.94	97.60→97.60	99.95→100
2/4	RM	97.17→99.48	99.94→99.97	75.95→98.20	86.50→100
	PM	99.27→99.27	99.94→99.94	97.75→97.75	100→100
1/4	RM	95.95→99.17	97.83→99.85	74.30→95.90	83.15→100
	PM	98.11→98.11	99.84→99.84	94.80→94.90	99.80→100

(a) Results for BLOCK

CLEVR Test Set			
SGGen	SGGen+	SA	Con
99.91→99.97	99.98→99.99	99.30→99.90	99.90→100
99.49→99.58	99.97→99.98	99.00→99.20	99.85→100
99.78→99.81	99.98→99.98	99.00→99.45	99.80→100
99.43→99.45	99.97→99.97	98.75→98.85	99.75→100
99.81→99.92	99.96→99.99	98.45→99.45	99.70→100
98.98→99.08	99.95→99.95	97.80→98.00	99.75→100
99.37→99.60	99.90→99.97	97.05→98.80	99.45→100
14.55→15.96	93.36→93.45	03.10→04.25	65.40→100

(b) Results for CLEVR

Table 3: Comparison of the different approaches with increasing dataset size (values in % with two decimals)

coordinates. However, fixing violations of domain-specific and physical constraints on top of a relational ML model **RM+OP** achieves better results than **PM** in all cases. This is a very important finding, since relations can be easily annotated by manual processing of images, precise object coordinates are typically unavailable in SGG scenarios [71]. In addition, our approach can also improve the performance even when the base metric value is already near perfect in **CLEVR** (99.30% → 99.90%), which is normally difficult to achieve with pure ML-based methods.

Despite a marginal gain, the effect of fixing inconsistencies for **PM** is less significant. We suspect this situation is because when transforming the coordinates from **PM** to a probabilistic scene graph, the relations are deterministically obtained from the coordinates. Thus, **OP** can only help deriving attribute values using the logical constraints, but the physical constraints (which restrict base relations) cannot provide more information in this case.

Moreover, Base **SA** values are in general much lower than **SGGen** and **SGGen+**, which indicates that our novel **SA** metric is more strict. For example, in **BLOCK**, the **SGGen** value is 95.95% for **RM** trained with 1/4 of the data while the **SA** value is only 74.30%. As mentioned in Section 3, **SA** measures the proportion of scenes isomorphic to the ground truth. The large difference on the value of the tuple and scene-based metrics means the errors are generally spread across scenes. At the same time, **OP** significantly improves **SA** in most cases and achieves near perfect (99.9%) in **CLEVR RM** with the full training set. This improvement means more scenes generated by **OP** contains are isomorphic to the ground truth, which is particularly beneficial for safety-critical applications such as autonomous driving, as a reliable (isomorphic) scene graph implies that any query for the scene will provide a justifiably correct answer. Since ground truth scenes are mostly consistent, fixing inconsistencies of a derived scene graph can improve all other metrics.

RQ3: *Our **OP** approach increases the overall performance of underlying ML models in each case, especially, for small datasets. A relational ML baseline with constraint optimization **RM+OP** outperformed **PM+OP**, which is particularly promising when only relations (but no precise object coordinates) are available for SGG training. Scene accuracy **SA** is stricter metric than traditional recall-based SGG metrics, and **OP** significantly improves **SA**.*

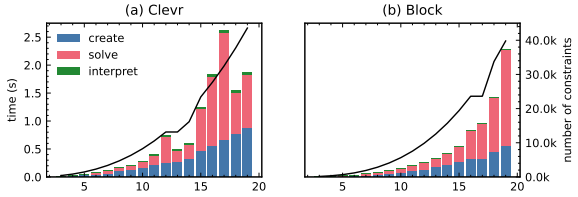


Figure 5: Time analysis for (a) CLEVR and (b) BLOCK

5.5 RQ4: Runtime analysis

Rationale. In this section, we perform runtime analysis for the optimization problem **OP**. Specifically, we aim to analyze the trend of runtime with the number of objects in the ground truth scenes.

Setup. We increase the number of objects from 3 to 19 for both **BLOCK** and **CLEVR** where the maximum is close to the average number of objects in a real-world dataset [46]. For each data point, we randomly generate an image with the specific number of objects and run it through the relational model **RM**. Then, we send the resulting probabilistic graph to **OP** and measure the time needed for each step of **OP** (create the problem, solve the problem and interpret the solution into scene graph). We also count the total number of unfolded constraints involved in the optimization problem. We measure for each data point ten times and take the average.

Analysis of results. Figure 5 shows the runtime analysis for both (a) **CLEVR** and (b) **BLOCK**. The bars show the time taken by **OP** and the lines show the total number of constraints created by **OP**. The flat line fragment in the chart indicates that the **RM** used for this analysis is imperfect, i.e. it may miss an object in the scene However, it does not influence the analysis of the overall trend.

The number of constraints in a scene and the optimization runtime grow polynomially with the increasing number of objects. Interestingly, the time for **OP** evolves almost linearly with respect to the number of constraints. This trend may indicate that most constraints are easy to be solved and the probabilities provided by the vision-based component gives a good hint in finding the result.

At the same time, while time needed to create the optimization problem is monotonically increasing, the time needed to solve the problem actually fluctuates. Again, this fluctuation may be attributed to the prediction quality from the visual-based component.

RQ4: *Our approach scales to average scene size of real world dataset. The time needed to solve the problem is also influenced by the quality of output from the vision-based component.*

5.6 Threats to validity

Internal validity. The ML models usually have random initial weights and the training is normally non-deterministic. We use best practice and the same number of training examples in training the base ML components as in [88]. We manually selected seeds to initialize the model to avoid different local maximum with different initialization and keep training deterministic. At the same time, we compared our approach with the baselines on different training sets of different size to minimize the influence of non-deterministic in training. We use 1/3 of all data for testing which is a typical setup in machine learning [23]. We use two popular independent benchmarks and get consistent results.

External validity. We selected simple and complex FOL constraints for both case studies which represent physical limitations in the real world as well as domain-specific restrictions (such as traffic rules). While the case studies are motivated by safety-critical self-driving applications, our measurements do not provide evidence about the direct applicability of our approach in such scenarios. Obviously, for SGG problems in other domains (with different constraint sets), our technique may result in different amount of actual improvements. Furthermore, in real applications, the performance of our approach relies on the precise formulation of the constraints by domain experts and the complexity of the constraints themselves. Nevertheless, our approach already works with an incomplete set of constraints, see Section 6. While similar constraints are often present in industrial validation tools [41, 65], the expressiveness of constraint languages may be a further limitation.

Construct validity. We measured standard performance metrics widely used in SGG scenarios, and we proposed a novel metric (scene accuracy), which is more strict than existing ones.

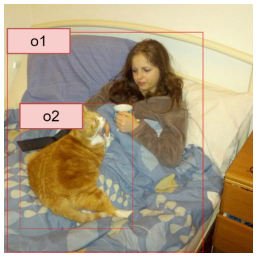
6 APPLICATION TO REAL WORLD DATASET

Finally, we demonstrate how to adopt our approach to Visual Genome (**VG**) [46], a real-world SGG dataset with complex object and relation types. For this initial study, we chose one physical constraint for relations and one logical constraint on object types.

For the physical constraints, we choose four relation types **Behind**, **Above**, **Under** and **In** and formulate one simple physical constraint on the four relation types: no loops with two or three objects. For the object type, we set a constraint that each scene must have a person-like object (i.e. "Person", "Man" or "Woman").

We filter the **VG** test set such that the ground truth scene contains at least one relation with the above types and one person-like object, which results in 4597 images. Furthermore, we check the consistency metric **Con** on the *ground truth* scenes and found that about 99.85% scenes are consistent with the physical constraint. The few inconsistent examples may be resulting from some noise in the labeling process. Nevertheless, to make our analysis closer to real-life scenario, we do not filter out such inconsistent scenes.

We choose a recent SGG model pretrained on the **VG** dataset [73] as our vision-based component. To simplify the comparison, we perform *scene graph classification* where the ground truth object bounding box is provided, and the task is to find the object and relation types. We compare the result by getting the most likely predicate for each object pair (without considering the constraints) and the graph generated by our approach. We skip the **SA** metric as the ground truth scenes in the dataset are not complete [46].



Bed(o1) → Woman(o1)
Under(o2, o1) → Above(o2, o1)

Metric	Improvement
SGGen	33.04 → 33.15
SGGen+	63.44→63.48
Con	64.43→100

Figure 6: An example scene [25], fixes, and metrics from VG

Figure 6 shows the improvement in the metrics by our approach for scene graph classification. Our approach still ensures 100% consistency over all test scenes, and, it gradually improves on other metrics (as the ground truth scenes are dominantly consistent).

7 RELATED WORK

7.1 Vision-based scene graph generation

End-to-end generation. The goal of end-to-end scene graph generation is to train deep neural networks which build scene graphs in one pass. A comprehensive overview of vision-based end-to-end scene graph generation approaches can be found in [13]. Convolutional neural networks (CNNs) are often used to learn encodings of objects and use them to extract attributes and relationships [42, 86]. Such methods have been extended to use Recurrent Neural Networks (RNNs) for relationship detection, e.g., Neural Motifs [91] and IMP [84]. Recently, some methods integrate graph neural networks (GNNs) due to their advantages for graph structures [15, 92].

End-to-end methods can generate scene graphs from real-world images with many relation and object types They are, however, computationally intensive and difficult to train. Moreover, unlike our approach, they largely do not support domain-specific constraints and provide no guarantee of consistency in generated scenes.

Some end-to-end scene generation methods incorporate domain-specific external knowledge during training. For instance, Cui et al. [19] and Liang et al. [49] improve relationship detection by using language priors to exploit similarities between words representing object and relation types. Further approaches [16, 35, 38, 90] include ConceptNet [68] extract information from an external knowledge base to improve generalization on rare relation and object types.

Our work is different from these techniques in two ways. (1) We address constraint satisfaction during post-processing rather than training and (2) we capture more expressive constraints by formulating such external knowledge as first-order predicates.

Multi-step generation. Multi-step generation breaks the end-to-end process into discrete steps to make the generation process more computationally efficient, as well as interpretable. However, such approaches are less represented in literature. Notably, Yi et al. [88]

and Reich et al. [62] recognize objects as a first step, then determine attributes, and finally generate relationships. In contrast, Tian et al. [74] first conduct end-to-end scene generation to create an intuitive graph and then refine it using multi-step reasoning.

We follow the same multi-step generation pipeline as authors of [62, 74], but extend it by incorporating constraint and external knowledge representation in a novel post-processing step.

7.2 Handling of constraints in deep learning

Explicitly including constraints during training can often improve the performance and generalization of ML models [28]. Dash et al. [21] and Borghesi et al. [11] provide comprehensive overviews.

Data augmentation. Augmenting data with additional in-distribution examples makes ML models more likely to capture constraints during training. Data augmentation has been successfully applied within image classification [59], scene graph generation [45], and natural language processing [47]. This method may improve constraint capturing, but provides no guarantees.

Loss function. Recent work has explored converting constraints into differentiable logic terms and integrating them into a loss function as 'regularization terms' [28, 72, 85]. This can significantly improve the consistency of ML model outputs, but we are unaware of any approaches that guarantees satisfaction of all constraints.

Architecture. Constraints can also be incorporated directly within deep learning architectures to ensure constraint satisfaction. Towell et al. [76] present an approach which determines most suitable architectures from a set of constraints. Wan et al. [79] add indicators as auxiliary input to reflect logic within the data to improve reasoning. Wang et al. [80] propose the integration of a MAXSAT layer to capture (physical) constraints. MultiplexNet [40] augments deep network output layers with multiplexors of logic constraints. In general, such approaches rely on relaxing hard constraints into soft constraints as differentiable continuous functions.

Such architecture-based approaches often consider only single object instances during training for problems such as image classification. Our approach can handle complex constraints which include multiple objects and relationships via post-processing after scene graph generation. Furthermore, our approach treats hard constraints strictly without relaxation.

Neurosymbolic logic reasoning. The research area of neurosymbolic logic reasoning [70] aims to combine traditional rule-based logic reasoning with deep learning. Approaches such as [24, 54, 55, 87] learn logic concepts and rules from scratch. In contrast, we leverage existing domain models and logic constraints to improve model performance ensure that the constraints are always satisfied.

Probabilistic logic programming [61] and abductive reasoning [20] were used in conjunction with neural networks to train models end-to-end with logical information as supervision and provide logically consistent output [53, 77, 82]. However, these approaches do not handle SSG where the number of objects in the scene is not known in advance. We use most probable evidence inference not as part of supervision, but as a post-processing step, which allows the separation of concerns between the model and logic reasoning and enables considering multi-step SSG tasks.

7.3 ML in Model-Driven Engineering (MDE)

There exist a growing number of works using ML techniques such as graph kernels [17] and reinforcement learning [27, 83] within MDE — where graph models and consistency constraints have been used extensively. Several works apply natural language processing for requirements engineering [3, 12, 39], e.g., for automated metamodel generation [6]. Likewise, ML is being used in cyber-physical systems [56, 60], model transformation [9], software artifacts mining [32, 64], and metamodel classification [51, 57].

In contrast, this paper exploits formal modeling techniques (metamodels, graphs, constraints, graph reasoning) for improving the consistency and overall performance ML components used in SGG.

8 CONCLUSIONS

In this paper, we investigated the challenge of consistent scene graph generation which aims to derive a scene graph (with key objects, attributes and relations) from a photorealistic image which fully satisfies a set of complex constraints (physical, geometrical, or domain-specific). While SGG is often used in an autonomous driving context, existing work addresses this challenge without consideration of consistency constraints. This is a major obstacle for future certification of SGG techniques in a safety-critical setting where justifiable evidence is necessitated.

We proposed a complementary technique to existing SGG approaches based on formal modeling (in particular, constraint optimization and probabilistic logic programming) by post-processing the probabilistic output of existing ML models used for SGG. Our approach provides theoretical guarantees that the output scene graph will satisfy complex, relational first-order logic constraints of the domain. We also proposed scene accuracy as a novel strict metric to help evaluate the performance of SGG.

We carried out experimental evaluation on two popular machine learning benchmarks. By fixing inconsistencies in the probabilistic graph, our approach showed significantly improvements (up to 35% points) compared to baseline ML techniques along all performance metrics. We achieved an overall scene accuracy of 98.85% and 99.90% for the two case studies on 2000 images as test sets. We also demonstrated that our approach scales to average scene size of Visual Genome, a real world dataset, and fixed inconsistent scene graphs also in that setting. All material for the experiments are available at [14].

ACKNOWLEDGMENTS

This paper is partially supported by the NSERC RGPIN-2022-04357 project, the FRQNT-B2X project (file number: 319955), the National Research, Development, and Innovation Fund of Hungary under Grant TKP2021-EGA-02, and the ÚNKP-21-4 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund. Project no. 2019-1.3.1-KK-2019-00004 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme.

REFERENCES

[1] Raja Ben Abdessalem, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2018. Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 1016–1026. https://doi.org/10.1145/3180155.3180160

[2] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2018. Testing autonomous cars for feature interaction failures using many-objective search. In *ASE 2018 - Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Vol. 18. Association for Computing Machinery, Inc, New York, New York, USA, 143–154. https://doi.org/10.1145/3238147.3238192

[3] Sallam Abualhaija, Chetan Arora, Mehrdad Sabetzadeh, Lionel C Briand, and Michael Traynor. 2020. Automated demarcation of requirements in textual specifications: a machine learning-based approach. *Empirical Software Engineering* 25, 6 (2020), 5454–5497.

[4] Rob Alexander, Hamid Asgari, Rob Ashmore, Andrew Banks, Rajiv Bongirwar, Ben Bradshaw, John Bragg, John Clegg, Jane Fenn, Chris Harper, et al. 2020. Safety assurance objectives for autonomous systems.

[5] Carlos Ansótegui and Joel Gabas. 2013. Solving (Weighted) Partial MaxSAT with ILP.. In *CPAIOR*, Vol. 13. 403–409.

[6] Chetan Arora, Mehrdad Sabetzadeh, Shiva Nejati, and Lionel Briand. 2019. An active learning approach for improving the accuracy of automated domain model extraction. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 28, 1 (2019), 1–34.

[7] Masataro Asai. 2018. Photo-Realistic Blocksworld Dataset. *ArXiv* abs/1812.01818 (2018).

[8] Aren A Babikian, Oszkár Semeráth, Anqi Li, Kristóf Marussy, and Dániel Varró. 2021. Automated generation of consistent models using qualitative abstractions and exploration strategies. *Software and Systems Modeling* (2021), 1–25.

[9] Islem Baki and Houari Sahraoui. 2016. Multi-step learning and adaptive search for learning complex model transformations from examples. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 25, 3 (2016), 1–37.

[10] Gábor Bergmann. 2014. Translating OCL to Graph Patterns. In *MoDELS (Lecture Notes in Computer Science, Vol. 8767)*. Springer, 670–686.

[11] Andrea Borghesi, Federico Baldo, and Michela Milano. 2020. Improving Deep Learning Models via Constraint-Based Domain Knowledge: a Brief Survey. *ArXiv* abs/2005.10691 (2020).

[12] Orlando Amaral CEJAS, Sallam Abualhaija, Damiano Torre, Mehrdad Sabetzadeh, and Lionel Briand. 2021. AI-enabled Automation for Completeness Checking of Privacy Policies. *IEEE Transactions on Software Engineering* (2021).

[13] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alexander G Hauptmann. 2021. A Comprehensive Survey of Scene Graphs: Generation and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[14] Boqi Chen. 2022. Clevr-Relational. https://github.com/20001LastOrder/Clevr-Relational.

[15] Long Chen, Hanwang Zhang, Jun Xiao, Xiangnan He, Shiliang Pu, and Shih-Fu Chang. 2019. Counterfactual Critic Multi-Agent Training for Scene Graph Generation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), 4612–4622.

[16] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. 2019. Knowledge-Embedded Routing Network for Scene Graph Generation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 6156–6164.

[17] Robert Clarisó and Jordi Cabot. 2018. Applying graph kernels to model-driven engineering problems. In *Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis*. 1–5.

[18] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Routledge.

[19] Zhen Cui, Chunyan Xu, Wenming Zheng, and Jian Yang. 2018. Context-dependent diffusion network for visual relationship detection. In *Proceedings of the 26th ACM international conference on Multimedia*. 1475–1482.

[20] Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. 2019. Bridging Machine Learning and Logical Reasoning by Abductive Learning. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[21] Tirtharaj Dash, Sharad Chitlangia, Aditya Ahuja, and Ashwin Srinivasan. 2022. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports* 12 (2022).

[22] DIYer22. 2022. bpycv. https://github.com/DIYer22/bpycv.

[23] Kevin K Dobbin and Richard M Simon. 2011. Optimally splitting cases for training and testing high dimensional classifiers. *BMC medical genomics* 4, 1 (2011), 1–8.

[24] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural Logic Machines. In *ICLR*.

[25] Matteo Doni. 2012. . flickr. https://www.flickr.com/photos/todoleo/8310164456/

[26] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

[27] Martin Eisenberg, Hans-Peter Pichler, Antonio Garmendia, and Manuel Wimmer. 2021. Towards Reinforcement Learning for In-Place Model Transformations. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 82–88.

[28] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin T. Vechev. 2019. DL2: Training and Querying Neural Networks with Logic. In *ICML*.

[29] ISO: International Organization for Standardization. 2018. ISO 26262-1, Road vehicles — Functional safety.

[30] ISO: International Organization for Standardization. 2019. ISO/PAS 21448, Road vehicles - Safety of the intended functionality.

[31] Daniel J Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L Sangiovanni-Vincentelli, and Sanjit A Seshia. 2019. Scenic: a language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 63–78.

[32] Khouloud Gaaloul, Claudio Menghi, Shiva Nejati, Lionel C Briand, and David Wolfe. 2020. Mining assumptions for software components using machine learning. In *Proc. of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 159–171.

[33] Geoffrey Gordon, Sue Ann Hong, and Miroslav Dudík. 2012. First-order mixed integer linear programming. *arXiv preprint arXiv:1205.2644* (2012).

[34] Object Modeling Group. 2005. *Object Constraint Language Specification, version 2.0*. Object Modeling Group.

[35] Jiuxiang Gu, Handong Zhao, Zhe L. Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. 2019. Scene Graph Generation With External Knowledge and Image Reconstruction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 1969–1978.

[36] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[37] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[38] Roei Herzig, Moshiko Raboh, Gal Chechik, Jonathan Berant, and Amir Globerson. 2018. Mapping Images to Scene Graphs with Permutation-Invariant Structured Prediction. In *NeurIPS*.

[39] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F Tichy. 2020. NoRBERT: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 169–179.

[40] Nicholas Hoernle, Rafael-Michael Karampatsis, Vaishak Belle, and Ya'akov Gal. 2021. MultiplexNet: Towards Fully Satisfied Logical Constraints in Neural Networks. *ArXiv* abs/2111.01564 (2021).

[41] IncQuery. 2022. IncQuery Model Validator. https://incquery.io/solutions/incquery-model-validator

[42] Sandeep Inuganti and Vineeth N. Balasubramanian. 2020. Assisting Scene Graph Generation with Self-Supervision. *arXiv: Computer Vision and Pattern Recognition* (2020).

[43] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 1988–1997.

[44] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3668–3678.

[45] Boris Knyazev, Harm de Vries, Cătălina Cangea, Graham W. Taylor, Aaron C. Courville, and Eugene Belilovsky. 2021. Generative Compositional Augmentations for Scene Graph Prediction. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), 15807–15817.

[46] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. https://arxiv.org/abs/1602.07332

[47] Bohan Li, Yutai Hou, and Wanxiang Che. 2021. Data Augmentation Approaches in Natural Language Processing: A Survey. *ArXiv* abs/2110.01852 (2021).

[48] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. 2021. Igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272* (2021).

[49] Xiaodan Liang, Lisa Lee, and Eric P. Xing. 2017. Deep Variation-Structured Reinforcement Learning for Visual Relationship and Attribute Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 4408–4417.

[50] Nan Liu, Shuang Li, Yilun Du, Joshua B. Tenenbaum, and Antonio Torralba. 2021. Learning to Compose Visual Relations. *ArXiv* abs/2111.09297 (2021).

[51] José Antonio Hernández López, Javier Luis Cánovas Izquierdo, and Jesús Sánchez Cuadrado. 2021. ModelSet: a dataset for machine learning in model-driven engineering. *Software and Systems Modeling* (2021), 1–20.

[52] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. 2016. Visual relationship detection with language priors. In *European conference on computer vision*. Springer, 852–869.

[53] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. DeepProbLog: Neural Probabilistic Logic Programming. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc.

[54] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. 2019. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *ICLR*.

[55] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. 2020. Relational Neural Machines. In *ECAI (Frontiers in Artificial Intelligence and Applications, Vol. 325)*. IOS Press, 1340–1347.

[56] Shiva Nejati. 2019. Testing cyber-physical systems via evolutionary algorithms and machine learning. In *2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST)*. IEEE, 1–1.

[57] Phuong T Nguyen, Juri Di Rocco, Ludovico Iovino, Davide Di Ruscio, and Alfonso Pierantonio. 2021. Evaluation of a machine learning classifier for metamodels. *Software and Systems Modeling* 20, 6 (2021), 1797–1821.

[58] James D. Park. 2002. Using Weighted MAX-SAT Engines to Solve MPE. In *AAAI*. AAAI Press / The MIT Press, 682–687.

[59] Luis Perez and Jason Wang. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *ArXiv* abs/1712.04621 (2017).

[60] Sebastian Pilarski, Martin Staniszewski, Matthew Bryan, Frederic Villeneuve, and Dániel Varró. 2021. Predictions-on-chip: model-based training and automated deployment of machine learning models at runtime. *Software and Systems Modeling* 20, 3 (2021), 685–709.

[61] Luc De Raedt and Angelika Kimmig. 2015. Probabilistic (logic) programming concepts. *Mach. Learn.* 100, 1 (2015), 5–47.

[62] Daniel Reich, Felix Putze, and Tanja Schultz. 2021. Adventurer's Treasure Hunt: A Transparent System for Visually Grounded Compositional Visual Question Answering based on Scene Graphs. *ArXiv* abs/2106.14476 (2021).

[63] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2015), 1137–1149.

[64] Safdar Aqeel Safdar, Tao Yue, Shaukat Ali, and Hong Lu. 2020. Using multi-objective search and machine learning to infer rules constraining product configurations. *Automated Software Engineering* 27, 1 (2020), 1–62.

[65] SAIC. 2022. Digital Engineering Validation Tool. https://www.saic.com/digital-engineering-validation-tool

[66] Axel Sauer, Kashyap Chitta, Jens Muller, and Andreas Geiger. 2021. Projected GANs Converge Faster. *ArXiv* abs/2111.01007 (2021).

[67] Oszkár Semeráth, Ágnes Barta, Ákos Horváth, Zoltán Szatmári, and Dániel Varró. 2017. Formal validation of domain-specific languages with derived features and well-formedness constraints. *Softw. Syst. Model.* 16, 2 (2017), 357–392.

[68] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *ArXiv* abs/1612.03975 (2017).

[69] Perdita Stevens. 2010. Bidirectional model transformations in QVT: semantic issues and open questions. *Softw. Syst. Model.* 9, 1 (2010), 7–20.

[70] Zachary Susskind, Bryce Arden, Lizy K. John, Patrick Stockton, and Eugene B. John. 2021. Neuro-Symbolic AI: An Emerging Class of AI Workloads and their Characterization. *CoRR* abs/2109.06133 (2021). arXiv:2109.06133 https://arxiv.org/abs/2109.06133

[71] Zsolt Szalay, Zoltán Hamar, and Peter Simon. 2018. A multi-layer autonomous vehicle and simulation validation ecosystem axis: Zalazone. In *International Conference on Intelligent Autonomous Systems*. Springer, 954–963.

[72] Naoya Takeishi and Yoshinobu Kawahara. 2019. Knowledge-based regularization in generative modeling. *arXiv preprint arXiv:1902.02068* (2019).

[73] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. 2020. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3716–3725.

[74] Hongshuo Tian, Ning Xu, An-An Liu, Chenggang Yan, Zhendong Mao, Quan Zhang, and Yongdong Zhang. 2021. Mask and Predict: Multi-step Reasoning for Scene Graph Generation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 4128–4136.

[75] Emina Torlak and Daniel Jackson. 2007. Kodkod: A Relational Model Finder. In *TACAS (Lecture Notes in Computer Science, Vol. 4424)*. Springer, 632–647.

[76] Geoffrey G. Towell, Jude W. Shavlik, and Michiel O. Noordewier. 1990. Refinement of approximate domain theories by knowledge-based neural networks. In *AAAI 1990*.

[77] Efthymia Tsamoura, Timothy M. Hospedales, and Loizos Michael. 2021. Neural-Symbolic Integration: A Compositional Perspective. In *AAAI*. AAAI Press, 5051–5060.

[78] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Niessner. 2019. RIO: 3D Object Instance Re-Localization in Changing Indoor Environments. In *Proceedings IEEE International Conference on Computer Vision (ICCV)*.

[79] Fang Wan and Chaoyang Song. 2018. A neural network with logical reasoning based on auxiliary inputs. *Frontiers in Robotics and AI* 5 (2018), 86.

[80] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*. PMLR, 6545–6554.

[81] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a Few Examples: A Survey on Few-shot Learning. *ACM Comput. Surv.* 53, 3 (2020), 63:1–63:34.

[82] Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. 2021. DeepStochLog: Neural Stochastic Logic Programming. *ArXiv* abs/2106.12574 (2021).

[83] Zhengkai Wu, Evan Johnson, Wei Yang, Osbert Bastani, Dawn Song, Jian Peng, and Tao Xie. 2019. REINAM: reinforcement learning for input-grammar inference. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 488–498.

[84] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. 2017. Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5410–5419.

[85] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. 2018. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *ICML*.

[86] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. 2018. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*. 670–685.

[87] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. 2020. CLEVRER: Collision Events for Video Representation and Reasoning. In *ICLR*.

[88] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. 2018. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *NeurIPS*.

[89] Shih-Yuan Yu, Arnav Vaibhav Malawade, Deepan Muthirayan, Pramod P Khargonekar, and Mohammad Abdullah Al Faruque. 2021. Scene-graph augmented data-driven risk assessment of autonomous vehicle decisions. *IEEE Transactions on Intelligent Transportation Systems* (2021).

[90] Alireza Zareian, Svebor Karaman, and Shih-Fu Chang. 2020. Bridging Knowledge Graphs to Generate Scene Graphs. In *ECCV*.

[91] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. 2018. Neural Motifs: Scene Graph Parsing with Global Context. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 5831–5840.

[92] Ji Zhang, Kevin J. Shih, A. Elgammal, Andrew Tao, and Bryan Catanzaro. 2019. Graphical Contrastive Losses for Scene Graph Generation. *ArXiv* abs/1903.02728 (2019).