Hochschule für angewandte Wissenschaften Coburg
Fakultät Elektrotechnik und Informatik

Studiengang: Informatik

Masterarbeit

Data Fusion for Scene Graph Generation: Bridging Simulated and Real Datasets

Kromm, Edward

Betreut durch:

Prof. Dr. Thomas Wieland, Hochschule Coburg
Sabrina Benassou, Forschungszentrum Jülich GmbH
Prof. Dr. Stefan Kesselheim, Forschungszentrum Jülich GmbH

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AdamW	Adam with decoupled Weight decay
ADAS	Advanced Driver-Assistance Systems
AI	Artificial Intelligence
BERT	Bidirectional Transformers for Language Understanding
CARLA	Car Learning to Act
CNN	Convolutional Neural Network
CSA	Coupled Self Attention
DEA	Decoupled Entity Attention
DETR	Detection Transformer
DS	Dataset Switch
DVA	Decoupled Visual Attention
ELE	Entity Layers Enabled
FED	Frozen Entity Detection
FFN	Feed Forward Network
GPU	Graphics Processing Unit
GTA	Grand Theft Auto
HPC	High Performance Computing
IoU	Intersection over Union
JSC	Jülich Supercomputing Centre
LiDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptrons
NeRF	Neural Radiance Fields
NLP	Natural Language Processing
nxtAIM	NXT GEN AI METHODS - Generative Methods for Perception, Prediction and Planning
PhrDet	Phrase Detection
PR	Precision-Recall
PredCls	Predicate Classification

RelTR	Relation Transformer
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SGCls	Scene Graph Classification
SGDet	Scene Graph Detection
SGG	Scene Graph Generation
Sim2Real	Simulation-to-Reality
VQA	Visual Question Answering

Abstract Deutsch

Die Generierung von Szenengraphen bietet ein leistungsstarkes Werkzeug für das KI-gestützte visuelle Verständnis von Bildern. Sie ermöglicht nicht nur die Erkennung von Objekten in einem Bild, sondern auch die Vorhersage von Beziehungen zwischen diesen, wie beispielsweise *Auto–hält an–Ampel* und *Fußgänger–überquert–Straße*. Diese Fähigkeit erlangt eine besondere Relevanz im Kontext des autonomen Fahrens, in dem der relationale Kontext zwischen Verkehrsteilnehmern und Infrastruktur eine entscheidende Rolle spielt. Allerdings wird die Anwendung der Szenengraphenerstellung in diesem Bereich durch den Mangel an annotierten Datensätzen behindert. Fahrsimulatoren wie CARLA bieten eine skalierbare Alternative, die im Vergleich zur manuellen Annotation eine effiziente Datengenerierung ermöglicht. Allerdings lassen sich Modelle, die ausschließlich auf simulierten Daten trainiert wurden, aufgrund der erheblichen Domänenlücke zwischen beiden oft nicht auf reale Daten übertragen.

In dieser Arbeit wird der Herausforderung, die sich aus der Kombination simulierter und realer Datensätze zur Erstellung autonomer, fahrspezifischer Beziehungsannotationen sowie der Schließung der Domänenlücke für Vorhersagen in der realen Welt stellt, aufgegriffen. Zu diesem Zweck wird ein neuartiges Datenfusionsframework vorgeschlagen. Die vorliegende Arbeit präsentiert die vollständige Pipeline, einschließlich der Datensatzerzeugung in der Simulation, der Anpassung öffentlich verfügbarer Ressourcen und Augmentationsstrategien. Das Relation Transformer-Modell wird eingehend analysiert, wobei besonderes auf die Interpretation der internen Mechanismen des Modells Wert gelegt wird. Zu diesem Zweck wird die gelernte Attention als Heatmaps visualisiert. Dies liefert Erkenntnisse darüber, ob sich das Modell bei der Vorhersage von Beziehungen auf semantisch bedeutsame Bereiche konzentriert. Aufbauend auf diesen Erkenntnissen werden zwei neue Ansätze vorgestellt, die eine Inferenz auf realen Daten ermöglichen und gleichzeitig das in der Simulation erworbene relationale Wissen übertragen. Eine Ablationsstudie quantifiziert darüber hinaus den Einfluss der Domänenlücke auf die Modellleistung und zeigt die Stärken und Grenzen der vorgeschlagenen Methoden auf. Die Resultate demonstrieren, dass einer der konzipierten Ansätze die Diskrepanz zwischen Simulation und Realität signifikant reduziert und konkrete Empfehlungen werden aufgeführt für die Fortentwicklung dieser Technik in Richtung der Verwendung als Werkzeug für das KI-gestützte visuelle Verständnis von Bildern im automotive Kontext gibt.

Abstract English

Scene graph generation has emerged as a powerful tool for AI-driven visual understanding of images by not only detecting objects in an image but also predicting the relationships between them, such as *car–stops at–traffic light* or *pedestrian–crosses–street*. This capability is particularly important for autonomous driving, where relational context between road users and infrastructure plays a critical role. However, the application of scene graph generation in this domain is hindered by the scarcity of annotated datasets. Driving simulators such as CARLA provide a scalable alternative, enabling efficient data generation compared to manual annotation. Yet models trained exclusively on simulated data often fail to generalize to real-world data due to the substantial domain gap between the two.

This thesis addresses this challenge by proposing a novel data fusion framework that combines simulated and real datasets to construct autonomous driving–specific relationship annotations and subsequently bridge the domain gap for real-world prediction. The work presents the complete pipeline, including dataset generation in simulation, adaptation of publicly available resources, and augmentation strategies. The Relation Transformer model is analyzed in depth, and particular attention is given to interpreting its internal mechanisms by visualizing the learned attention maps as heatmaps. This analysis provides insights into whether the model focuses on semantically meaningful regions when predicting relationships. Building on this understanding, two new approaches are introduced to enable inference on real data while transferring relational knowledge acquired in simulation. An ablation study further quantifies the impact of the domain gap on model performance and highlights the strengths and limitations of the proposed methods. Results demonstrate that one of the developed approaches effectively mitigates the simulation-to-reality gap and concrete suggestions for advancing this technique toward further uses for AI-driven visual understanding of images in the automotive context are provided.

1 Introduction

Cognitive tasks, such as describing images and answering questions are highly challenging for an Artificial Intelligence (AI) to learn. They require recognizing visual elements and reasoning about their complex relationships and contextual meanings [KZG+17]. Inspired by the human ability to effortlessly interpret and understand visual scenes, visual scene understanding has become a highly relevant research topic in computer vision [LZZ+24]. Unlike instance-level tasks, which focus solely on detecting and recognizing individual objects, scene understanding goes beyond mere object localization by capturing the semantic relationships between objects. These relationships provide richer contextual information, enabling a more comprehensive representation of a scene.

Understanding driving scenarios is critical to autonomous driving because it enables developers to design more effective and explainable Advanced Driver-Assistance Systems (ADAS). However, the complexity and highly dynamic nature of real-world driving environments, where objects interact in structured yet unpredictable ways, present a significant challenge. To interpret situations such as *a pedestrian crossing in front of a vehicle* or *a vehicle overtaking a bicycle*, autonomous systems must go beyond object detection and incorporate relational reasoning [YMM+22a, ZQL+24, MYH+22].

One way to structure such information is through a scene graph, which is a graphical representation in which nodes correspond to object instances and directed edges encode the relationships between these objects. By explicitly modeling interactions, such as *a man sitting on a bench* or *a car parked next to a building*, scene graphs facilitate high-level reasoning about an image's content. Deep learning models that can infer these structured representations from images are known as Scene Graph Generation (SGG) models [JKS+15].

SGG models are designed to process an image and generate a scene graph that describes its semantic content. See Figure 1 for reference. These models have received significant attention due to their potential in various downstream applications, including image captioning and Visual Question Answering (VQA) [CRX+23]. Scene graphs act as a bridge between visual and textual representations, making them useful for multimodal AI applications. One of the largest data sets for this application: Visual Genome, presents this concept [KZG+17].

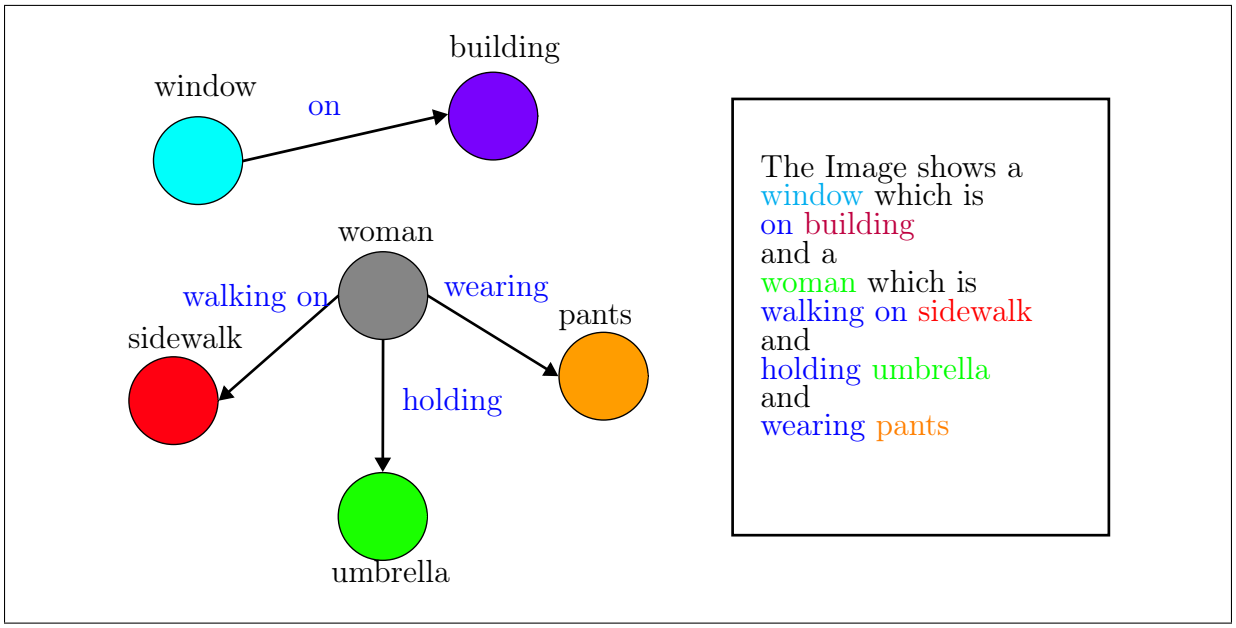


Figure 1: From Scene Graph to textual description. Adapted from [CYR23]

1.1 Motivation and Objectives

In one of the earlier reports [Edw24] associated with this thesis, the potential use of scene graphs as a semantic metric was analyzed. A semantic metric measures the distance (or difference) between two images based on their semantic content. To develop this metric, a SGG model was applied to two images, specifically the Relation Transformer (RelTR) [CYR23]. The outputs were then compared using the Bidirectional Transformers for Language Understanding (BERT) score [ZKW+20], a metric that quantifies the semantic similarity of these texts.

This research is conducted within the domain of autonomous driving as part of the NXT GEN AI METHODS - Generative Methods for Perception, Prediction and Planning (nxtAIM) project [Rei24]. The goal of nxtAIM is to adapt and apply generative methods to autonomous driving by leveraging technological advancements and large data sets to advance the field [Rei24]. However, supervised learning in this field is hindered by the limited availability of annotated data, which is a particular challenge in scene graph generation due to the costliness of relationship annotations. For reference, the Visual Genome data set [KZG+17] relied on contributions from over 33,000 unique crowd workers to achieve broad coverage. Despite such efforts, no data set currently exists that provides detailed relationship annotations specifically tailored to autonomous driving scenarios. The conclusion of the previous report emphasized that, in order to effectively use RelTR as a semantic metric for autonomous driving data, a richly annotated data set with domain-specific relationships is essential.

To address this challenge, this thesis builds upon a self-created annotation framework. Specifically, a semi-automatic annotation tool was designed to efficiently generate a richly annotated data set with minimal human effort. This tool leverages the Car Learning to Act (CARLA) simulation software [DRC+17] to create autonomous driving images along with automatically annotated relationships.

Figure 2: Example from the CARLA Simulator

However, training RelTR solely on a simulated data set presents a significant challenge. The CARLA simulator is built on the Unreal Engine 4 game engine [Epi], resulting in images that resemble synthetic video game graphics rather than real-world scenes (see Figure 2). Although the visual appearance differs considerably, the semantic relationships between objects remain valid. Overcoming this domain gap is the central focus of this thesis and leads to the overarching research objective: *How can the domain gap between simulated and real-world data be addressed, particularly through the use of a transformer architecture that leverages an encoder-decoder pipeline to bridge the gap in the latent space?*

This master's thesis is conducted in cooperation with Jülich Supercomputing Centre (JSC), which provides access to powerful High Performance Computing (HPC) resources. These systems will be used extensively throughout this research.

1.2 Problem Definition

The lack of richly annotated large-scale real-world data sets remains a major bottleneck in training high-performance deep learning models, particularly for vision-based applications in autonomous driving. Tasks such as image recognition (e.g., detecting vehicles or pedes-

trians), path planning and navigation (e.g., identifying drivable versus occupied areas), and target tracking (e.g., maintaining the identity of an object across frames) all depend on deep learning models and large amounts of high-quality training data [ZCC+25]. However, acquiring and annotating real-world data at scale is both costly and time consuming.

To address this challenge, recent research has increasingly explored the use of synthetic data generated in simulation environments and video games, such as Grand Theft Auto (GTA) V [RVRK16a]. These virtual platforms offer richly detailed scenes, complete control over environmental conditions, and the ability to generate large amounts of labeled data with minimal manual effort [RVRK16b]. Synthetic data sets have demonstrated significant promise in various perception tasks, such as object detection, semantic segmentation, and scene understanding. This is particularly evident in domains like autonomous driving, where the demand for diverse and accurately labeled data is critical.

However, a persistent challenge in the use of synthetic data is the Simulation-to-Reality (Sim2Real) domain gap — the performance degradation observed when models trained on synthetic data are evaluated on real-world imagery [RVRK16b, WU18]. This gap stems from significant differences in image statistics, lighting conditions, textures, noise characteristics, and semantic distributions between the simulated and real world domains. Despite ongoing efforts, the fundamental causes and mechanisms behind this generalization gap remain only partially understood [RDD24].

This problem becomes even more pronounced in the task of SGG, which extends object detection by modeling pairwise relationships between entities. In the context of autonomous driving, scene graphs offer a structured and interpretable representation of the environment, allowing reasoning about interactions such as *car turning at intersection* or pedestrian crossing road in front of vehicle. Despite their promise, no existing real-world data set provides fine-grained relationship annotations tailored to driving scenarios. The annotation process for scene graphs is significantly more complex than for bounding boxes or labels, making real-world SGG data sets particularly scarce [KZG+17].

Synthetic environments, in contrast, offer not only efficient data capture, but also enable semi-automated annotation pipelines. Access to internal simulation metadata, such as object locations, class IDs, and scene structure, allows the automatic generation of high-quality relationship annotations. This makes synthetic data an attractive alternative to overcome data scarcity, provided that the domain gap can be effectively bridged.

1.2.1 Research Hypotheses

This thesis investigates whether the Sim2Real domain gap for a transformer-based SGG model, specifically RelTR [CYR23], can be mitigated by decoupling the training process and aligning the learned latent space representations.

The proposed approach assumes that the RelTR encoder, trained on real-world data, can learn robust domain-relevant visual features, while the relationship classification head, trained on synthetic data, can take advantage of this latent space to predict relationships on the real data domain.

Feature extraction: The encoder of the RelTR model is trained on real-world driving data to capture relevant and domain-specific image features.

Relationship Classification: The reasoning layers responsible for predicting pairwise relationships are trained on synthetic data, which provides dense and accurate annotations via automated labeling.

Due to the absence of a richly annotated real-world SGG data set, joint fine-tuning of both components is currently infeasible. Instead, this thesis explores model merging strategies that combine these separately trained components into a unified, end-to-end SGG model capable of generalizing to real-world driving scenes.

The data set created in the course of this thesis does not claim completeness. Instead, it focuses specifically on road lane–related relationships, which are missing or underrepresented in existing data sets. Given this, the question arises whether those selected relationships are even learnable to infer even without a domain shift. The scope of this research is strictly limited to the RelTR model; comparisons with other SGG architectures are considered out of scope.

Finally, due to the computational cost associated with training RelTR, only a limited set of hyperparameter adjustments was made compared to the original implementation in [CYR23]. This decision reflects the assumption that the published hyperparameters are sufficiently well tuned and allows this thesis to focus on evaluating the proposed approaches. Extensive hyperparameter optimization, including methods such as K-Fold cross-validation, is therefore excluded from the scope of this work.

1.3 Thesis Structure

This thesis is organized into several key sections that build upon each other to address the research objective of bridging the Sim2Real domain gap in SGG for autonomous driving.

It begins with a comprehensive introduction to Convolutional Neural Network (CNN) and the transformer architecture, providing the necessary background for understanding the model analyzed in this work. This is followed by an in-depth examination of the RelTR model architecture, including its core components: the CNN backbone, the encoder-decoder architecture, and the feedforward classification network. The section concludes with a detailed explanation of the employed loss functions.

After establishing the theoretical and technical foundations, the thesis introduces its core research concept. This includes a clear statement of the main research objectives, a description of the two proposed approaches to address the domain gap, and an explanation of how these approaches are evaluated. The motivation and relevance of the research are also discussed to provide context and define the scope of the work.

Subsequently, a comprehensive review of the related literature is presented. It begins with an overview of previous efforts to bridge the Sim2Real gap, outlining the challenges and solutions explored in existing research. The focus then shifts to the use of simulation environments and scene graphs in autonomous driving, further emphasizing the relevance of this work within the broader research landscape. And concludes with differencing this work from existing publications.

With the context established, the thesis proceeds to describe the data sets developed and used in the experiments. Both data sets are analyzed in terms of class distributions and frequency of occurrence. For the relationship data set, the distribution of relation types is also examined. Each data set section concludes with a baseline training of the respective model component to serve as a benchmark for subsequent evaluations.

The core of the thesis focuses on the two proposed approaches to address the Sim2Real domain adaptation problem. For each method, the experimental setup is described, followed by a detailed evaluation. Performance is first assessed on simulated data, with results compared against established baselines. Average and class-wise performances are analyzed. Subsequently, the models were evaluated on a small, manually annotated real-world data set. This includes qualitative analyses and inference examples, with a particular focus on attention maps to provide insight into model behavior.

After analyzing the Sim2Real approaches, an ablation study is conducted to better understand the strengths and limitations of the proposed approaches. The best-performing strategy is applied to a third data set that is visually closer to the simulated domain. This allows for distinguishing between limitations inherent to the approach itself and those caused by the domain gap.

The thesis concludes with a comprehensive discussion of the research, highlighting key contributions and findings, and addressing the initial research questions. Following this,

a dedicated section on future work outlines potential directions to expand and enhance the study, suggesting avenues for further investigation and improvements. Finally, the concluding chapter synthesizes the overall outcomes.

2 Prior Knowledge

To establish a foundation for the subsequent analysis, this section outlines the key concepts and architectures relevant for this thesis. It begins with an overview of CNNs and the transformer architecture, followed by a detailed examination of the RelTR model, which forms the core of the methodology employed.

2.1 Convolutional Neural Networks

CNN are a core architectural paradigm in deep learning, especially suited for processing grid-like data, such as images. Introduced as a biologically inspired approach to visual perception, CNNs have become the foundation for most modern computer vision systems [AZH+21, PMR+23].

A typical CNN architecture begins with a convolutional layer followed by a non-linear activation function, and often a pooling layer. This basic pattern can be grouped into blocks and repeated throughout the network. In classification-focused CNNs, the final layers are usually one or more fully connected layers, as illustrated in Figure 3.

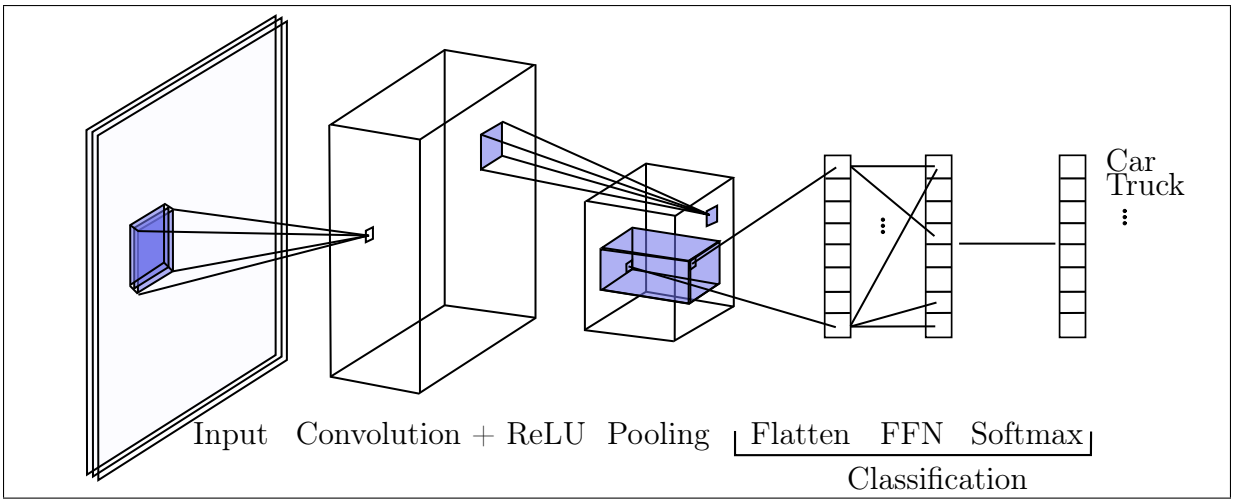


Figure 3: Standard CNN architecture. Adapted from [PMR+23].

The convolutional layer applies a kernel (or filter) to compute feature maps from the input data. This kernel typically has the same number of channels c and a square spatial dimension, resulting in a shape of $w \times h \times c$. The filter slides over the input image, computing the sum of element-wise multiplications between the kernel weights and the input values. This produces a two-dimensional activation map, as defined in Equation 2.1[PMR+23].

$$\text{Activation map} = \sum_{y=0}^{columns} \times \left(\sum_{x=0}^{rows} \text{Input}(x-p, y-q)\ \text{Filter}(x, y) \right) \tag{2.1}$$

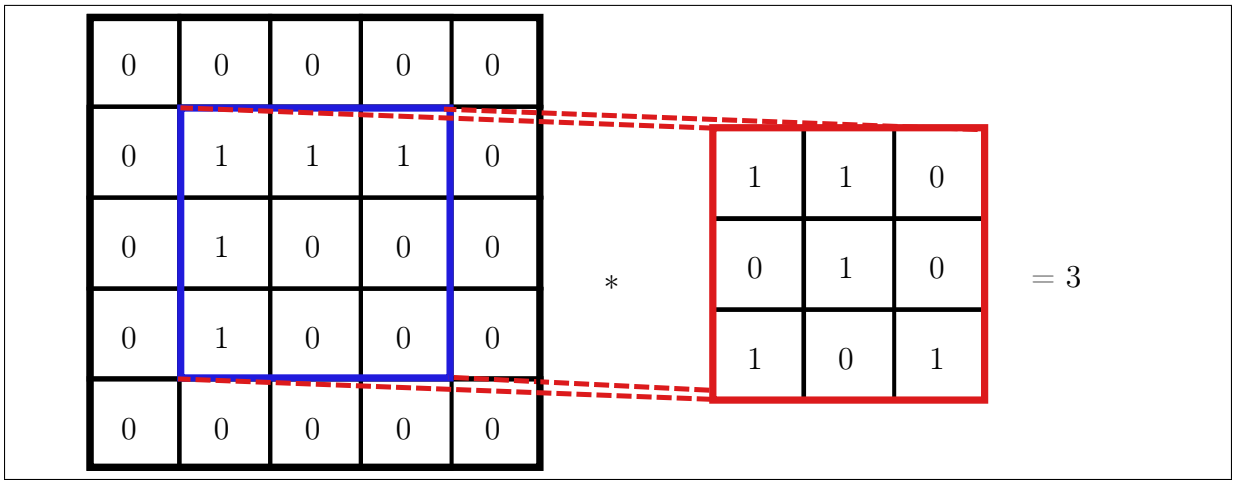


Figure 4: Convolutional layer operation. Adapted from [PMR+23].

The resulting activation map generally differs in spatial dimensions from the input. According to Equation 2.2, the output dimensions depend on the kernel size K, stride S and padding P. Padding, often implemented by zero-padding the input, can be used to preserve spatial dimensions. Alternatively, increasing the stride leads to a greater spatial reduction.

$$W_{out} \times H_{out} = \left(\lfloor \frac{W_{in} + 2P - K}{S} \rfloor + 1 \right) \times \left(\lfloor \frac{H_{in} + 2P - K}{S} \rfloor + 1 \right) \tag{2.2}$$

A visual representation of this convolutional operation is shown in Figure 4.

After convolution, the activation map is passed through a non-linear activation function. These functions are essential in neural networks to introduce non-linearity, allowing them to learn complex and abstract patterns [DSC22]. Following activation, pooling layers reduce spatial dimensions by aggregating neighboring values using operations such as maximum or average pooling. This reduces the computation and helps mitigate overfitting [PMR+23].

2.2 Transformer

The transformer is a deep learning architecture originally proposed for sequence modeling tasks in Natural Language Processing (NLP) [VSP+17]. Since then, the transformer architecture has also found wide application in computer vision and scene understanding [CMS+20, CYR23, DBK+21].

The standard transformer consists of an encoder that maps an input sequence of symbol representations $x_1, ..., x_n$ to a sequence of continuous representations $z = (z_1, ..., z_n$. The decoder then generates an output sequence $y_1, ..., y_n$, typically one token at a time. Unlike

previous neural sequence transduction models such as Recurrent Neural Network (RNN) or Long Short-Term Memory (LSTM), the transformer relies solely on the attention mechanism, without recurrence or convolution.

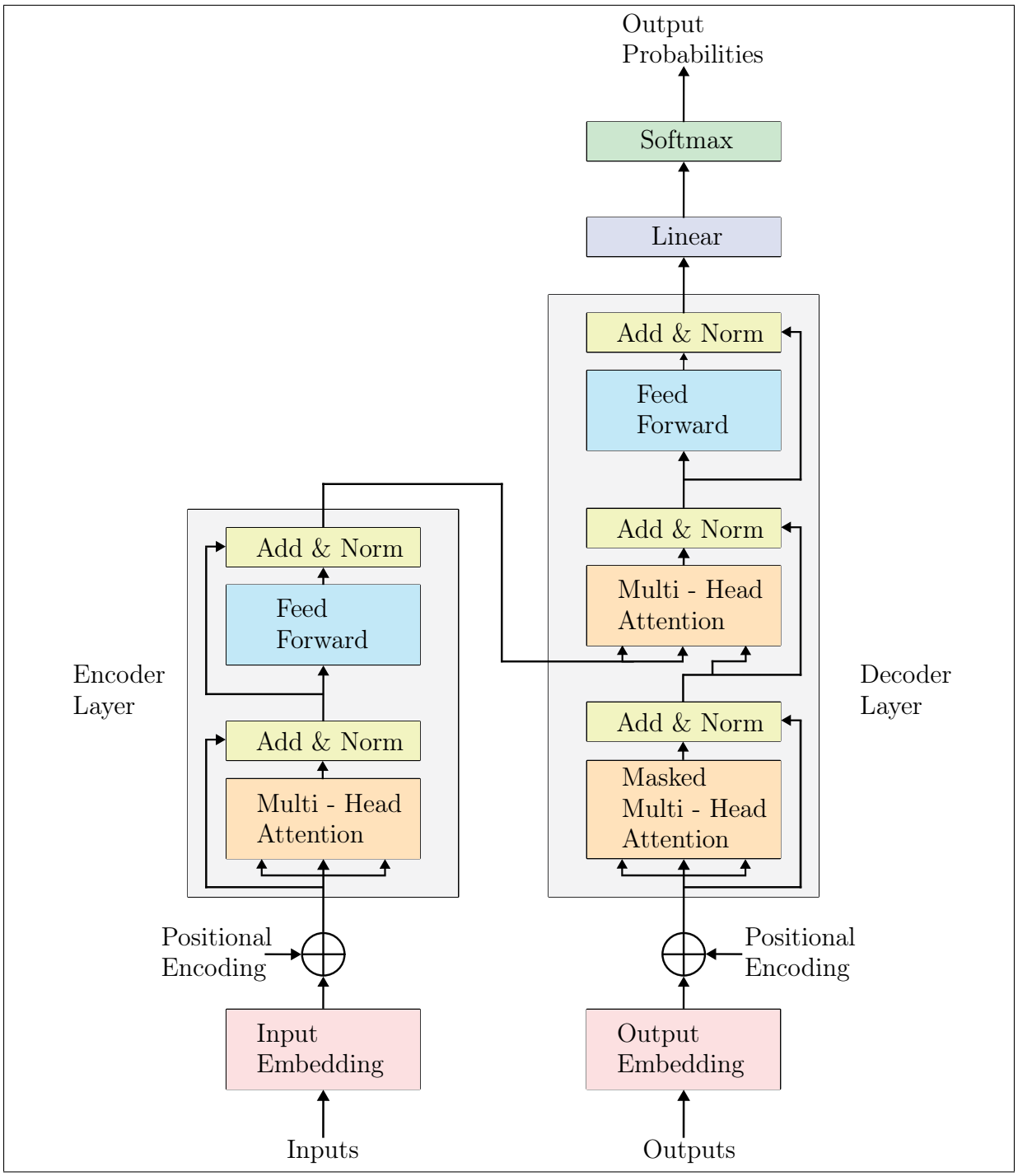


Figure 5: Transformer Architecture. Adapted from [VSP+17]

The core component is the attention function, which maps a query and a set of key-value pairs to an output. Specifically, the query q_i key k_i and the value v_i are vector

representations derived from the same input vector x_i using different learned weight matrices W_q, W_k, W_v respectively:

$$q_i = W_q \times x_i k_i = W_k \times x_i v_i = W_v \times x_i \tag{2.3}$$

These vectors are typically stacked into matrices Q, K, V for parallel computation. The scaled dot-product attention is computed by taking the dot product of the queries and keys. This is then scaled by the square root of the key dimension d_k. The result is normalized using the softmax function and multiplied by the value matrix V:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.4}$$

with the softmax function defined as:

$$softmax(x_i) = \frac{e^{V_i(x)}}{\sum_k e^{V_k(x)}} \tag{2.5}$$

which normalizes each input $V_i(x)$ into the range of $(0, 1)$ with the constraint that all inputs $V_k(x)$ sum up to 1 [Bri89].

The attention operation allows each position in the input sequence to attend to all other positions, making it possible to capture long-range dependencies.

To increase model expressiveness and allow the network to learn from different representation subspaces at different positions, the transformer introduces multi-head attention. Instead of performing a single attention operation, the input is linearly projected into multiple lower-dimensional subspaces via separate projection matrices W_i^Q, W_i^K and W_i^K, for each attention head i. These projections are processed in parallel and their outputs concatenated. A final projection matrix W^O is applied to combine the heads into the final output:

$$\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.6a}$$

$$\text{where} \quad \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.6b}$$

$$W_i^Q, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v} \tag{2.6c}$$

For a graphical overview of the original transformer see Figure 5.

As shown in Figure 5, the transformer integrates positional encodings to preserve spatial structure. Since the attention mechanism itself is permutation invariant, it lacks any inher-

ent notion of spatial or sequential order [VSP+17]. One common approach to incorporating positional information in transformers is to add fixed sinusoidal encodings of varying frequencies to the input embeddings:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.7a}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.7b}$$

Alternatively, positional encodings can also be learned during training, allowing the model to adapt the positional representation to the specific task.

2.3 Overview of RelTR Architecture

This section provides a brief description of how the RelTR model operates and outlines its main architectural components. The discussion begins with the CNN backbone for feature extraction, followed by the transformer encoder–decoder architecture, and concludes with the inference procedure and loss functions used for training.

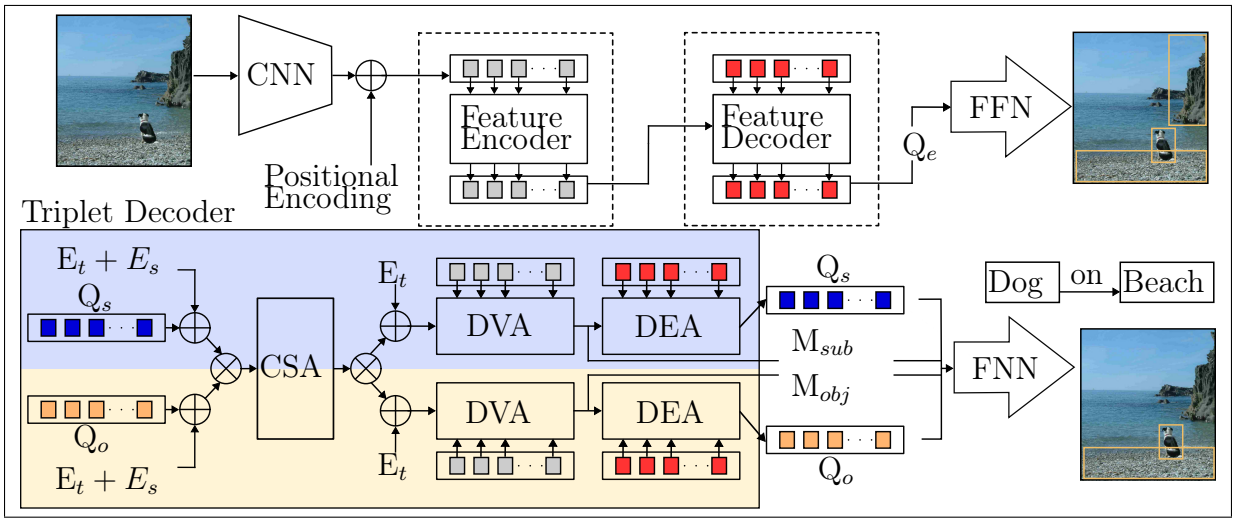


Figure 6: Architecture overview of RelTR. Adapted from [CYR23]

2.3.1 CNN Backbone Architecture

Several well-known CNN architectures build on the same principles, including AlexNet [KSH17], VGGNet [SZ15], and ResNet [HZRS16]. Since the authors of RelTR chose ResNet as their backbone, this architecture is the focus of this thesis.

In [HZRS16], the authors analyzed the training and testing error behavior of networks with increasing depth. They observed that deeper networks can suffer from higher training

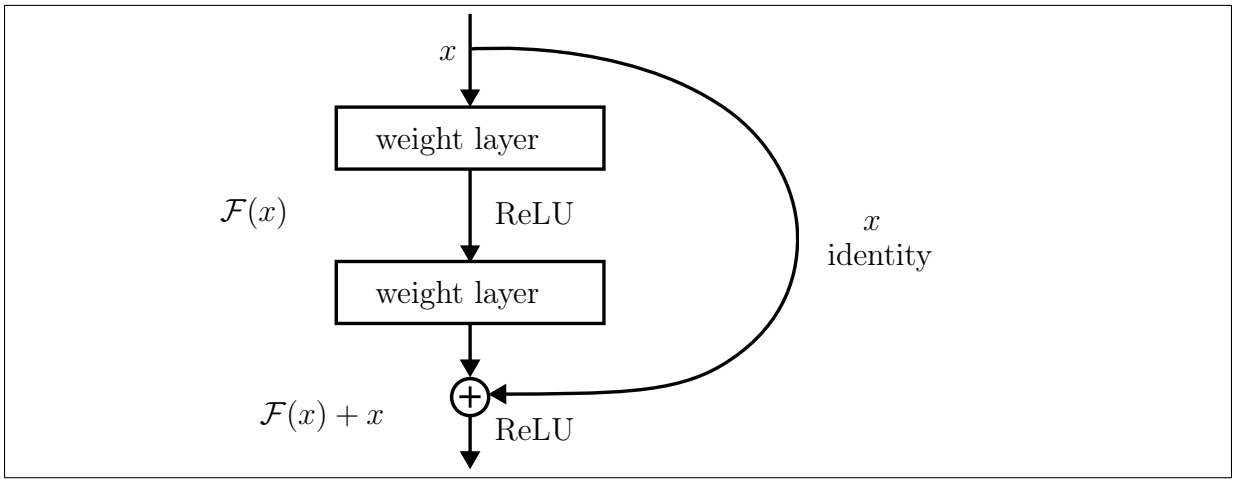


Figure 7: Residual block with identity shortcut. Cite [HZRS16].

error due to the vanishing or exploding gradient problem, where gradients either become negligibly small or grow uncontrollably due to repeated multiplications during backpropagation. To address this, they introduced the deep residual learning framework, which is the basis of ResNet.

Instead of learning a direct mapping $\mathcal{H}(x)$, residual networks learn a residual function $\mathcal{F}(x) = \mathcal{H}(x) - x$, so that the actual mapping becomes:

$$\mathcal{H}(x) = \mathcal{F}(x) + x \tag{2.8}$$

These residual blocks typically consist of two convolutional layers with Rectified Linear Unit (ReLU) activations in between and one around the residual sum. The ReLU function is defined as:

$$ReLU(x) = \max(0, x) \; for \; x \in \mathbb{R} \tag{2.9}$$

This structure, shown in Figure 7, allows for the training of significantly deeper networks without degrading performance. If gradients through $\mathcal{F}(x)$ vanish, the identity shortcut x ensures that at least the input is propagated, effectively skipping the block. This principle is not only used for CNN architectures, it is also used in transformer-based architectures [VSP+17].

The authors propose several variants of the ResNet architecture, differentiated primarily by their depth. These include ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, where the number indicates the total number of layers in the network. In the case of RelTR, the ResNet-50 variant is used as the convolutional backbone to extract visual features from the input images.

To adapt ResNet for use as the backbone in RelTR, the classification layers at the end are removed. This allows the network to output a dense feature map (often referred to as feature embedding) instead of class scores. This output is then passed on to the transformer component, as described in the following section.

2.3.2 Encoder

The encoder of RelTR largely follows the structure of a standard transformer encoder but is adapted to process image-based feature maps rather than one-dimensional token sequences. Each encoder layer consists of a multi-head self-attention mechanism and a position-wise Feed Forward Network (FFN), both surrounded by residual connections Equation 2.8 and layer normalization. The FFN applies two linear transformations with a ReLU Equation 2.9 activation in between, as defined in Equation 2.10.

$$FFN(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{2.10}$$

For a schematic overview of the encoder, see Figure 5.

Adapting the Transformer Encoder to Images Unlike traditional transformers that operate on one-dimensional sequences of tokens, RelTR is designed to handle two-dimensional spatial features extracted from images. A similar adaptation is discussed in [DBK+21], where image inputs are divided into flattened two-dimensional patches and reshaped into a sequence:

$$x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}, \tag{2.11}$$

where C denotes the number of channels, (P, P) is the resolution of each patch, and $N = HW/P^2$ is the total number of patches for an image of resolution (H, W). These patches are then projected into the model's embedding space using a trainable linear layer.

In RelTR, a hybrid approach is used [DBK+21]. Instead of working with raw image patches, the model first passes the image through a CNN backbone, which generates a feature map $Z_0 \in \mathbb{R}^{HW \times d}$. This feature map is then transformed using a convolutional layer to align with the transformer's input embedding dimension.

Positional Encodings for two-dimensional Spatial Information For image-based inputs, a two-dimensional positional encoding is necessary. Following the approach in [DBK+21], separate sine/cosine encodings are computed for the X and Y spatial axes.

Each encoding has a dimensionality of $d_{model}/2$, and they are concatenated to form the final two-dimensional positional embedding. This enables the model to capture both horizontal and vertical spatial relationships.

Query, Key, and Value Calculation in RelTR In RelTR, the query $Q_{encoder}$ and key $K_{encoder}$ matrices are derived from the CNN feature map output. The classification layers of the backbone are removed (as discussed in Section 2.3.1), producing a feature activation tensor $f \in \mathbb{R}^{C \times H \times W}$. This feature map is processed by a convolutional layer, and two-dimensional positional encodings E_p are added to compute the attention inputs:

$$Q_{encoder}, K_{encoder} = \text{Conv}(f) + E_p. \tag{2.12}$$

Because the self-attention mechanism requires identical queries and keys, $Q_{encoder}$ and $K_{encoder}$ share the same values. The value $V_{encoder}$, however, is derived from the convolutional layer without the addition of positional encodings.

This design ensures that spatial structure is preserved throughout the attention process, while enabling the model to reason about interactions across the entire image. By encoding both content and location, the encoder can learn meaningful object and relation features that are crucial for downstream SGG tasks [CYR23].

2.3.3 Decoder

The RelTR's decoder stacks three key attention mechanisms: Coupled Self Attention (CSA), Decoupled Visual Attention (DVA), and Decoupled Entity Attention (DEA), on top of the Entity Detection Layer from the Detection Transformer (DETR) [CMS+20]. The three key attention mechanisms are referred to as the relationship detection module in this thesis.

Entity Detection Layer The Entity Detection Layer forms the first component of each decoder block in RelTR. Although it follows the general transformer decoder architecture proposed in [VSP+17], it incorporates specific adaptations for parallel object decoding. Specifically N_e learnable embeddings of dimension d are processed via a multi-head self-attention mechanism (cf. Equation 2.6a) followed by a position-wise FFN.

Unlike the original transformer, which decodes one token at a time, DETR [CMS+20] introduces parallel decoding of N objects. RelTR adopts this parallel decoding paradigm to represent multiple entities simultaneously.

A crucial distinction between the encoder and decoder is the presence of a cross-attention mechanism, which connects the decoder's entity representations with the encoder's visual features. While the mathematical operations of cross-attention are identical to those of self-attention, the key difference lies in the origin of the inputs:

1. Self-attention: Query Q_{self}, Key K_{self} and Value V_{self} are all derived from the same sequence (i.e., the entity embeddings).
2. Cross-attention: Query Q_{cross} is obtained from the previous decoder block, while K_{cross} and V_{cross} originate from the encoder output, also referred to as *memory* or the latent space.

Implementation in RelTR In RelTR, the queries Q_{self} and keys K_{self} are computed from the sum of the entity representations and their corresponding positional embeddings. The values V_{self} in contrast, are derived directly from the entity representations without positional information.

For the cross-attention operation:

1. Q_{cross} is obtained similarly to Q_{self}, but uses the updated entity representations from the preceding self-attention layer, such that $Q_{cross} \neq Q_{self}$
2. K_{cross} and V_{cross} are sourced from the encoder memory. Only the keys are augmented with positional encodings.

The output of this layer is passed through a FFN:

$$Q_e = FFN(Q_e + Q_e) \tag{2.13}$$

The FFN consists of two linear transformations with a ReLU activation in between, followed by residual connections and layer normalization, which ensure stable gradient propagation. The resulting entity embeddings Q_e, are forwarded to the triplet decoding layers for subject-object reasoning.

Coupled Self-Attention The CSA layer enables the decoder to learn the semantic roles of entities specifically, whether they act as subjects or objects in relational triplets. This step is essential for SGG, as it lays the foundation for identifying semantic dependencies between entities [CYR23].

To achieve this, two types of learnable queries are introduced:

1. Subject queries $Q_s \in \mathbb{R}^{N_t \times d}$
2. Object queries $Q_q \in \mathbb{R}^{N_t \times d}$

However, since self-attention is permutation-invariant (i.e., treats all elements symmetrically), explicit differentiation between triplet components is required. This is addressed by introducing learnable triplet encodings $E_t \in \mathbb{R}^{N_t \times d}$, along with subject-specific $\mathbb{E}_s$ and object-specific $\mathbb{E}_o$ role embeddings.

The input to the self-attention layer is constructed as:

$$Q = K = [Q_s + E_s + E_t, Q_o + E_o + E_t] \tag{2.14}$$

The concatenated representation is then processed through a self-attention mechanism:

$$[Q_s, Q_o] = Att_{CSA}(Q, K, [Q_s, Q_o]) \tag{2.15}$$

After attention is applied, the representations retain the same symbols Q_s and Q_o for brevity. The updated embeddings are then split back into subject and object branches.

Decoupled Visual Attention The DVA module extracts fine-grained visual features from the encoder's output Z_{memory} enabling each entity to attend to relevant image regions. Unlike CSA, DVA treats the subject and object branches independently to prevent mutual interference.

The decoder queries Q_s, Q_o are individually combined with the triplet encodings and used to query the encoder memory, which is augmented with positional encodings:

$$Q = Q_{s/o} + E_t, K = Z_{memory} + E_p \tag{2.16}$$

Where:

1. $Q_{s/o}$ represents the subject or object query, processed independently.
2. E_t is the learnable triplet encoding, which helps distinguish between different triplets.
3. E_p originated from the CNN backbone and was computed alongside the feature map Z_0.

The attention operation produces updated entity queries as well as spatial attention maps:

$$Q_{s/o}, M_{(sub/obj)} = Att_{DVA}^{(sub/obj)}(Q, K, Z_{memory}) \tag{2.17}$$

The attention maps ($M_{(sub/obj)} \in \mathbb{R}^{N_t \times HW}$ indicate which regions of the input feature map each subject or object attends to. These will later be used to predict the coordinates of the subject and object (see Section 2.3.4).

Decoupled Entity Attention The final module in each decoder block, DEA, refines the subject and object representations by leveraging the global entity embeddings Q_e. Unlike CSA and DVA, DEA does not distinguish between semantic roles, enabling information flow between the entity-level and triplet-level representations.

The DEA operation is defined as:

$$Q_{(s/o)} = Att_{DEA}^{sub/obj}(Q_s + E_t, Q_e, Q_e) \tag{2.18}$$

The updated representations are subsequently passed through a FFN:

$$Q_{(s/o)} = FFN(Q_{(s/o)} + Q_{(s/o)} \tag{2.19}$$

As in earlier layers, this FFN consists of two linear layers with ReLU activation, residual connections, and layer normalization. The result is a refined embedding of subject and object queries that can be used for predicting relational triplets.

2.3.4 Final Inference

The final output of RelTR is a list of predicted relational triplets. Each triplet consists of a predicate label, the class labels of both subject and object, and their corresponding bounding box coordinates.

Output	**Symbol**	**Produced By**
Target Entity Representations	Q_e	Output of Entity Layer
Target Triplet Representations	$[Q_s, Q_o]$	CSA $\rightarrow$ DVA $\rightarrow$ DEA
Subject Attention Map	M_{sub}	DVA
Object Attention Map	M_{obj}	DVA

Table 1: Key outputs of the transformer model at different stages.

Alternative Entity-Based Inference The model can perform classification and bounding box regression using global entity representations Q_e, produced by the Entity Layer, as an alternative inference mode. This mode is useful when training the entity detection module independently.

$$\text{Class}_{entity} = \text{softmax}(\text{FFN}(Q_e)) \tag{2.20a}$$

$$\text{BBox}_{entity} = \sigma(\text{MLP}(Q_e)) \tag{2.20b}$$

This allows for a decoupled approach where entity localization and classification are handled independently of triplet reasoning.

Object and Subject Inference The class labels for the subject and object are predicted using two independently trained FFNs, each consisting of a single linear layer. These classifiers take the respective triplet representations Q_s and Q_o as input (cf. Table 1):

$$\text{Class}_{sub/obj} = \text{softmax}(\text{FFN}(Q_{s/o})) \tag{2.21}$$

Bounding box coordinates for both subject and object are predicted via separate Multi-Layer Perceptrons (MLP)s. Each MLP takes its respective entity representation as input and outputs four values representing the normalized center coordinates x, y and the width and height of the bounding box:

$$\text{BBox}_{sub/obj} = \sigma(\text{MLP}(Q_{s/o})) \tag{2.22}$$

Predicate Prediction The predicate (i.e., relationship) between the subject and object is predicted by another MLP. It takes as input the concatenation of the refined subject and object representations Q_s and Q_o as well as a spatial feature vector V_{spa}:

$$\hat{p}_{prd} = \text{softmax}(\text{MLP}([Q_s, Q_o, V_{spa}])) \tag{2.23}$$

The spatial feature vector V_{spa} is computed by concatenating the attention heatmaps for the subject and object M_{sub} and M_{obj} and passing the result through a CNN, as illustrated in Figure 8.

The softmax function in Equation 2.23 yields a probability distribution over all predicate classes, from which the most likely predicate is selected.

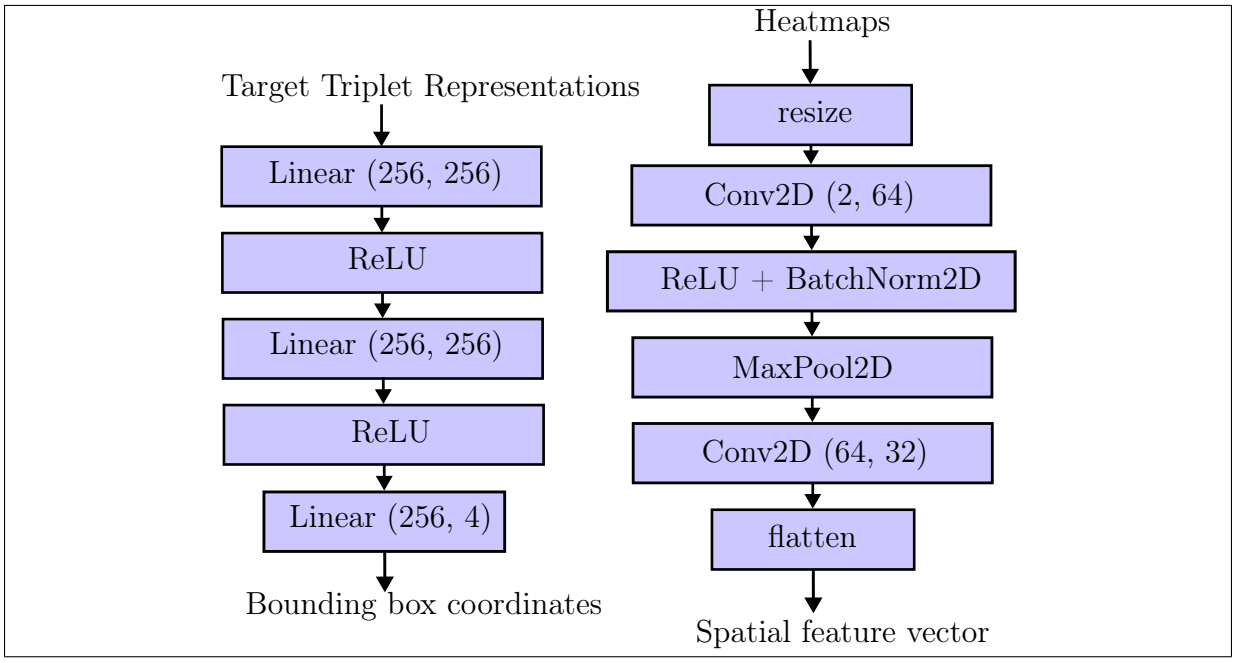


Figure 8: Left: Architecture to generate bounding box coordinates. Right: Architecture to generate spatial feature vector V_{spa}. Adapted from [CYR23]

2.3.5 Loss Functions

Like DETR [CMS+20], RelTR outputs a fixed number N of predictions, where N is set to be significantly larger than the typical number of objects in an image. In order to compute the loss between the set of N predictions, denoted as $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$, and the corresponding ground truth y, the following three steps are performed:

1. Extend the ground truth set y to length N by appending no-object $\varnothing$ entries.
2. Determine the optimal assignment between predicted and ground truth elements.
3. Calculate the loss based on this assignment.

Optimal matching is achieved via the Hungarian algorithm [CYR23, CMS+20]. A classical use case of the Hungarian algorithm (see Algorithm 1) is the Job Assignment Problem.A toy example is provided to illustrate the algorithmic principles using the minimization variant required for loss computation.

Assume a set of employees $A_{\text{employee}} = (0, 1, 2, 3)^T$ and a set of jobs $B_{\text{jobs}} = (0, 1, 2, 3)$, along with a cost matrix indicating the cost for assigning employee e to job j. The goal is to find a one-to-one assignment between employees and jobs that minimizes the total cost, subject to the constraint that each employee is assigned to exactly one job and no job is assigned to more than one employee. A graphic showing the algorithm in action for a cost matrix M_{cost} can be found in Figure 9.

Algorithm 1 Hungarian Algorithm

```
1: M_cost of shape (N, N)
2: M ← M_cost                                   ▷ Create working copy of cost matrix
3: procedure INITIALIZATION(M)
4:     for each row r in M do
5:         Subtract min(r) from each element in r
6:     end for
7:     for each column c in M do
8:         Subtract min(c) from each element in c
9:     end for
10:    COVERING(M)
11: end procedure
12: procedure COVERING(M)
13:    Cover all zeros in M using the minimum number of horizontal and vertical lines
14:    U ← set of uncovered elements
15:    D ← set of elements covered twice (horizontally or vertically)
16:    if number of covering lines needed= N then
17:        OutputOptimum(M)
18:    else
19:        Augmentation(U, D, M)
20:    end if
21: end procedure
22: procedure AUGMENTATION(U, D, M)
23:    m ← min(U)
24:    for each e ∈ U do
25:        e ← e − m
26:    end for
27:    for each e ∈ D do
28:        e ← e + m
29:    end for
30:    Covering(M)
31: end procedure
32: procedure OUTPUTOPTIMUM(M)
33:    opt ← 0
34:    coveredColumns ← []
35:    for each row index i in M do
36:        Z ← list of column indices j such that M[i][j] = 0
37:        j* ← arg min_{j∈Z} M_cost[i][j]  &  j ∉ coveredColumns
38:        append j to coveredColumns
39:        opt ← opt + M_cost[i][j*]
40:    end for
41:    return opt
42: end procedure
```

Analogously, in the case of RelTR, the optimal assignment between ground truth elements and predictions is determined by constructing a cost matrix, where each entry quantifies the cost of matching a predicted element $\hat{y}_i$ with a ground truth label y_j.

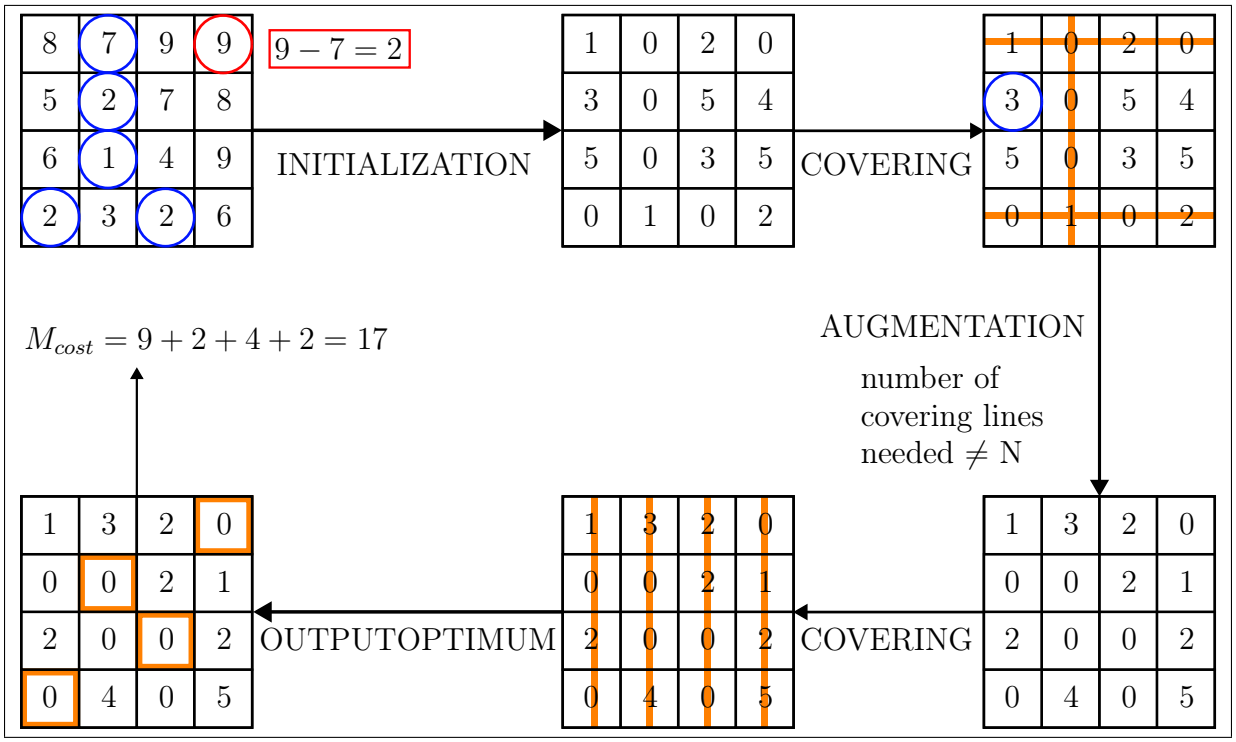


Figure 9: Example of the Hungarian algorithm. Blue circles indicate the row-wise minima. Red circles mark the minima among the column-wise minima after subtracting the row-wise minima. Column-wise minima equal to zero were excluded for clarity

As described in Section 2.3.4, RelTR generates five prediction outputs: (1) entity class probabilities ($Class_{\text{entity}}$) and (2) entity bounding boxes ($BBox_{\text{entity}}$) for object-level inference, as well as (3) subject/object class probabilities ($Class_{\text{sub/obj}}$), (4) subject/object bounding boxes ($BBox_{\text{sub/obj}}$), and (5) predicate probabilities ($\hat{p}_{\text{pred}}$) for relationship triplet inference.

Entity-Based Matching In Entity-Based Matching, each entry of the Entity Cost Matrix $ECM_{i,j}$, with $(i, j) \in N_{\text{entity}}$, where N_{entity} denotes the number of predictions made by the Entity Detection Layer (see Section 2.3), is computed as a linear combination of the negative log-likelihood of the class prediction and the bounding box loss. This results in the following equation:

$$\mathcal{L}_{\text{HungarianEntity}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right], \tag{2.24}$$

where $\hat{\sigma}(i)$ is computed as a pair-wise matching cost between ground truth y_i and a prediction with index $\sigma(i)$:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}), \tag{2.25}$$

The negative log-likelihood term $-\log \hat{p}_{\hat{\sigma}(i)}(c_i)$ is computed by selecting the predicted class probability corresponding to the ground truth label c_i from the model output at index $\hat{\sigma}(i)$. This is done by using the output of the final layer of the FFN applied to $Q_{s/o}$ before the softmax (see Equation 2.20a) and then applying the logarithm.

The bounding box loss $\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$ measures the distance between the predicted bounding box $\hat{b}_\sigma(i)$ and the ground truth box b_i. It combines the l_1 distance and the generalized Intersection over Union (IoU) loss $\mathcal{L}_{\text{giou}}$ [RTG+19], resulting in the following equation:

$$\mathcal{L}_{\text{box}}(b_i, \hat{b}_\sigma(i)) = \lambda_{\text{iou}}\mathcal{L}_{\text{giou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}}\|b_i - \hat{b}_\sigma(i)\|_1, \tag{2.26}$$

where $\lambda_{\text{iou}}, \lambda_{\text{L1}} \in \mathbb{R}$ are hyperparameters that balance the two loss terms [CMS+20].

Algorithm 2 Generalized IoU. Adapted from [RTG+19]

Predicted Box with $B^p = (x_1^p, y_1^p, x_2^p, y_2^p)$
Ground Truth Box with $B^g = (x_1^g, y_1^g, x_2^g, y_2^g)$
Calculating area of $B^p : A^p = (x_2^p - x_1^p) \times (y_2^p - y_1^p)$
Calculating area of $B^g : A^g = (x_2^g - x_1^g) \times (y_2^g - y_1^g)$
Calculating intersection $\mathcal{I}$ between B^p and B^g:
$x_1^{\mathcal{I}} = max(x_1^p, x_1^g)$, $x_2^{\mathcal{I}} = max(x_2^p, x_2^g)$,
$y_1^{\mathcal{I}} = max(y_1^p, y_1^g)$, $y_2^{\mathcal{I}} = max(y_2^p, y_2^g)$
$\mathrm{I} = \begin{cases} (x_2^{\mathcal{I}} - x_1^{\mathcal{I}})(y_2^{\mathcal{I}} - y_1^{\mathcal{I}}), & \text{if } x_2^{\mathcal{I}} > x_1^{\mathcal{I}} \text{ and } y_2^{\mathcal{I}} > y_1^{\mathcal{I}} \\ 0 & \text{otherwise} \end{cases}$
Finding the coordinate of smallest enclosing box Box^c:
$x_1^c = min(x_1^p, x_1^g)$, $x_2^c = min(x_2^p, x_2^g)$,
$y_1^c = min(y_1^p, y_1^g)$, $y_2^c = min(y_2^p, y_2^g)$
Calculating area of $B^c : A^c = (x_2^c - x_1^c) \times (y_2^c - y_1^c)$
$IoU = \frac{I}{U}, where\ U = A^p + A^g - I$
$GIoU = IoU - \frac{A^c - U}{A^c}$
return $1 - GIoU$

Relationship Triplet Matching In Relationship Triplet Matching, each entry of the triplet cost matrix $TCM_{(i,j)}$, with $(i, j) \in N_{\text{triplet}}$, where N_{triplet} denotes the number of relationship triplet predictions made by the DVA and DEA modules (see Section 2.3), is computed as a linear combination of the negative log-likelihoods of the subject, predicate, and object class labels, as well as the corresponding bounding box losses for the subject and object.

Final Loss Calculation After applying the Hungarian algorithm to match predictions with the ground truth, the final loss can be calculated. This loss is then minimized by an optimizer through weight updates in the model.

For training the RelTR model, the authors employ three primary loss components [CYR23]:

1. Classification loss for entity, subject, and object class labels, denoted as $\mathcal{L}_{\text{labels}}$,
2. Bounding box regression loss for entity, subject, and object boxes, denoted as $\mathcal{L}_{\text{boxes}}$,
3. Relation classification loss, denoted as $\mathcal{L}_{\text{relations}}$.

(1) Label Loss: The ground truth labels y_{labels} consist of the class labels of all entities in the image, as well as the entities acting as subject and object in each relation triplet. The predicted labels $\hat{y}_{\text{labels}}$ are obtained from the outputs of Equation 2.20a and Equation 2.21. Since the matching ensures that each ground truth entry at index i corresponds to its respective prediction at index i, the classification loss is computed using the standard cross-entropy function [COR98]:

$$\mathcal{L}_{\text{labels}}(y_{\text{labels}}, \hat{y}_{\text{labels}}) = -\sum_{i=1}^{C} y_{\text{labels},i} \log(\hat{y}_{\text{labels},i}) \tag{2.27}$$

where C is the number of classes.

(2) Bounding Box Loss: The ground truth bounding boxes y_{boxes} contain the coordinates (x_1, y_1, x_2, y_2) for all entities, including those acting as subjects and objects in the relation triplets. The box loss $\mathcal{L}_{\text{boxes}}$ combines the generalized IoU loss and the ℓ_1 loss, similar to the DETR model:

$$\mathcal{L}_{\text{boxes}}(y_{\text{boxes}}, \hat{y}_{\text{boxes}}) = \mathcal{L}_{\text{giou}}(y_{\text{boxes}}, \hat{y}_{\text{boxes}}) + \|y_{\text{boxes}} - \hat{y}_{\text{boxes}}\|_1 \tag{2.28}$$

(3) Relation Loss: The predicted relations $\hat{y}_{\text{relations}}$ and their corresponding ground truth $y_{\text{relations}}$, are used to compute the relation classification loss. This is done again using cross-entropy.

$$\mathcal{L}_{\text{relations}}(y_{\text{relations}}, \hat{y}_{\text{relations}}) = -\sum_{i=1}^{C} y_{\text{relations},i} \log(\hat{y}_{\text{relations},i}) \tag{2.29}$$

Combined Loss: The total loss used to train the model is a weighted sum of the three components described above:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{labels}} + \lambda_2 \mathcal{L}_{\text{boxes}} + \lambda_3 \mathcal{L}_{\text{relations}} \tag{2.30}$$

Where, λ_1, λ_2, and λ_3 are hyperparameters that control the relative contribution of each loss term to the overall training objective [CYR23].

2.3.6 Optimizer

The optimizer used to minimize the loss functions presented in Section 2.3.5 by updating the parameters of the RelTR model is Adam with decoupled Weight decay (AdamW) [CYR23, LH17]. AdamW is a variant of the Adam optimizer that incorporates weight decay, to the momentum terms of Adam, by decoupling it from the gradient computation.

The optimization process begins by computing the gradients of the model parameters with respect to the loss (Algorithm 3 Line 8). These gradients are then used to update two running averages, commonly referred to as the first and second momentum terms (Algorithm 3 Line 9-10). To account for initialization bias, both momentum terms are corrected (Algorithm 3 Line 11-12). After applying the adaptive update based on the momentum terms, the separate weight decay term is added (Algorithm 3 Line 14) [LH17].

Algorithm 3 Adam with decoupled weight decay [LH17]

Require: Learning rate $\alpha = 10^{-4}$
Require: Decay and regularization $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\lambda = 10^{-4}$
1: $t \leftarrow 0$
2: $\theta_{t=0} \in \mathbb{R}^n$ ▷ Parameters to optimize
3: $\eta_{t=0} \in \mathbb{R}$ ▷ Schedule multiplier
4: $m_{t=0} \leftarrow 0$ ▷ First moment vector
5: $v_{t=0} \leftarrow 0$ ▷ Second moment vector
6: **while** *stopping cirterion not met* **do**
7: $t \leftarrow t + 1$
8: $g_t \leftarrow \nabla f_t(\theta_{t-1})$
9: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
10: $v_t \leftarrow \beta_2 v_{t-1} + 1(1 - \beta_2) g_t^2$
11: $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
12: $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
13: $\eta_t \leftarrow SetScheduleMultiplier(t)$
14: $\theta_t \leftarrow \theta_{t-1} - \eta_t (\alpha \hat{m}_t / (\sqrt{\hat{v}_t + \epsilon}) + \lambda \theta_{t-1}$
15: **end while**

3 Concept

This thesis investigates whether the domain gap arising in SGG tasks—when models are trained on simulated data but evaluated on real-world data—can be effectively bridged by leveraging the learned latent space representations, without altering or adapting the input data. Specifically, the study examines whether relational knowledge learned in simulation can transfer to real-world scenes purely through architectural and training strategies, without requiring explicit domain adaptation techniques at the input or data level.

The primary research objectives are:

1. To explore whether latent-space learning can bridge the domain gap in SGG.
2. To identify challenges that arise in such a transfer, including whether certain relationship types (e.g., spatial vs. functional) are inherently harder to generalize.
3. To evaluate the practical utility and scalability of this approach for future research or applications.

The RelTR model [CYR23] is selected as the foundation due to its transformer-based architecture and efficient inference performance. While inference speed is not the central focus of this thesis, it was an important factor in prior research, where SGG models were used to define semantic similarity metrics for image comparison—such as loss functions in generative models. In the context of the nxtAIM project [Rei24], training generative models—particularly Stable Diffusion—relies on extensive Graphics Processing Unit (GPU) resources within the cluster infrastructure of Forschungszentrum Jülich. Maximizing GPU utilization is critical to ensure training efficiency and to avoid unnecessary resource consumption. Consequently, if SGG models are employed as part of a loss function in such training pipelines, its computation must be as fast as possible to prevent it from becoming a bottleneck [Cite Sabrina].

The architectural design of RelTR makes it particularly well-suited for modular training, as it is built upon the DETR [CMS+20], allowing for independent training of the object detection and relationship prediction components. This modularity enables a hybrid training strategy, where the object detection module can be trained on real-world data sets (to preserve visual grounding), and the relationship module can be trained on rich, fully-annotated synthetic data.

The model is divided into two functional components:

1. Entity Detection Module: Responsible for detecting objects, predicting bounding boxes, and classifying entities. This component will be trained on a real-world data set such as CityScapes to ensure robust grounding in natural visual scenes.

2. Relationship Detection Module: Responsible for predicting pairwise relationships between detected entities (e.g., car - on right lane of - road, person - on - sidewalk). This module will be trained on synthetic data generated in the CARLA simulator, which provides dense and consistent relationship annotations.

Two key strategies will be implemented and compared to investigate how relational features can generalize across domains:

1. Sequential Freeze Strategy:

   1.1 The entity detection module is first trained on real-world data and then frozen.

   1.2 The relationship prediction module is subsequently trained on synthetic data.

   1.3 A variant of this approach will also be explored where only the encoder is frozen, and the decoder layers (which integrate entity and relationship features) are fine-tuned during relationship training.

2. Data set Switch Strategy:

   2.1 A conditional training loop is implemented: real-world samples are used to update only the entity module, while synthetic data updates both modules. This strategy encourages a gradual alignment of latent feature spaces between the domains.

   2.2 Additionally, a semi-supervised variant will be introduced using pseudo-labels generated from the Sequential Freeze model to augment real-world relationship annotations.

The evaluation will proceed in two phases:

1. Simulated Test Set Evaluation: Each model variant will first be evaluated on a held-out CARLA test set to establish an upper bound for relationship prediction performance.

2. Real-World Benchmark Evaluation: A custom, hand-annotated data set of real-world images will be created, annotated with the same relationship ontology as the CARLA data set. Although small in size, this data set will serve as the main benchmark to assess how well the model transfers relational knowledge to real-world domains.

Standard SGG metrics will be used for evaluation, including:

1. Recall@K: Measures whether ground truth relationships are ranked among the top-K predicted ones, for each class.
2. Mean Recall@K: Evaluates the model's ability to predict a diverse set of relationships, for all classes.

Furthermore, attention map visualization and analysis will be performed on the relationship prediction module to interpret how and where the model focuses during inference, helping to identify failure cases and interpret learned features.

4 Related Work

Sim2Real transfer is a widely researched approach in machine learning, particularly in Reinforcement Learning (RL), computer vision and robotics. Central challenges in this field include bridging the domain gap between synthetic and real-world data, addressing limited access to labeled real data, and improving the generalization of learned models across domains. Prominent strategies include transfer learning, domain adaptation, domain randomization, and hybrid training with synthetic and real data.

4.1 Generalization and Domain Adaptation

A key line of research focuses on generalization via simulation. For instance, [KBK+19] propose a model that first learns perception and task-specific policies in simulation, then transfers the learned perception layers to a real-world reward predictor. Their method, tested on autonomous drone navigation, significantly outperformed alternative transfer learning strategies. Similarly, [ZQW20] provide a comprehensive survey of Sim2Real transfer techniques in deep RL, covering approaches such as zero-shot transfer, domain adaptation, and domain randomization. In zero-shot transfer, models trained in high-fidelity simulators are directly deployed in real-world scenarios. Domain adaptation seeks to explicitly align source (simulated) and target (real) domains, often via adversarial training or feature space alignment. Domain randomization enhances robustness by introducing diverse variations—such as lighting, textures, and noise—into the simulation, encouraging models to generalize better to unseen real-world data.

4.2 Transfer Learning with Synthetic Data

Transfer learning has proven especially effective when real-world annotated data is scarce. In fault diagnosis, [ALZ+23] show that models trained solely on simulated data can generalize effectively to real-world scenarios. In manufacturing, [TGH+18] train models on synthetic process simulations before fine-tuning on real experimental data, significantly accelerating learning and improving performance.

More generally, [Rog23] analyze the effectiveness of synthetic data for model training. Their findings suggest that while synthetic data is often sufficient for training, hybrid strategies—combining real and simulated data—typically yield better performance. Similarly, in financial anomaly detection, [SVN23] demonstrate that synthetic data can mitigate class imbalance and data sparsity, though they also caution against overfitting to synthetic distributions or adversarial exploitation.

4.3 Domain Randomization

Domain randomization is a widely adopted technique to narrow the Sim2Real gap. [TPA+18] train object detection models using highly varied synthetic images, randomizing lighting, object textures, and camera poses. They show that fine-tuning on a small amount of real-world data can result in accuracy levels comparable to models trained purely on real data. Likewise, [TFR+17] demonstrate that domain-randomized simulations can enable deep models trained entirely in simulation to perform well on real-world object grasping and detection tasks.

4.4 Simulation in Autonomous Driving

Simulation plays a central role in autonomous driving research, offering a safe, controllable, and repeatable environment for the development and evaluation of decision-making algorithms. One such simulator, also used in this thesis, is CARLA [DRC+17]. Driving simulators like CARLA enable the definition of diverse traffic scenarios and the generation of synthetic sensor data, relying on underlying physics engines to approximate realistic vehicle and environmental behavior. Despite their utility, these simulators generally produce deterministic outcomes and may lack the variability present in real-world driving situations [CZY+24].

Beyond traffic simulation, such platforms have proven useful for tasks like scene graph extraction. For instance, [YMM+22b] proposed an extraction pipeline—conceptually similar to the one preliminarily developed in this thesis—that takes advantage of the simulator's internal access to ground-truth metadata (e.g., precise object locations and class labels) to construct structured representations of traffic scenes.

In addition to scenario testing, simulation is increasingly recognized as a scalable and safe approach to data generation. [ZWBR+24] explore the use of generative AI to synthesize training data that bridges the gap between simulated and real domains. Similarly, [YLWX18] present a real-to-virtual domain unification strategy, where real-world driving data is mapped into a simplified virtual representation. This transformation reduces visual complexity and supports domain-invariant learning for downstream tasks such as control prediction.

Recent advances in generative AI have further expanded the simulation toolbox. For example, [ZLX+24] demonstrate how Neural Radiance Fields (NeRF) can be employed to reconstruct Light Detection and Ranging (LiDAR) point clouds, offering a promising approach to increase diversity and realism in sensor data. These developments highlight

the growing relevance of simulation and generative methods in addressing safety and data annotation challenges in autonomous driving research.

4.5 Distinction from Existing

This thesis differentiates itself from the previously mentioned related work due to the nature of SGG models. Traditional SGG methods typically follow a multi-stage pipeline, decomposing the task of SGG into sub-tasks such as object detection, relation graph construction, and relation prediction. In recent years, however, one-stage methods have emerged that aim to directly predict scene graphs from images [LZZ+24]. Nevertheless, in the case of the RelTR model, which is considered a one-stage method, the following holds true: the model first detects objects in an image and then predicts the relationships between them [CYR23]. This thesis leverages this characteristic to bridge the Sim2Real gap. Rather than relying on techniques to make simulated data usable throughout the entire training process, this thesis focuses on the transition point in the model, from object detection to relationship prediction, and uses this stage to bridge the gap between real and simulated input data.

To date, the challenge of domain transfer in SGG has received limited attention. The only directly relevant work found is by [PDL+21], who also attempt to align real and simulated domains using the latent space of an encoder. However, their method relies on modifying the simulated input appearance to mimic real-world images, and their relationship ontology is limited to simple positional predicates (e.g., left of, behind, in front of). In contrast, this thesis focuses on preserving the input space and instead aligning latent relational representations between domains.

5 Data set

In this thesis, two distinct data sets were utilized for training the model, each selected to fulfill a specific purpose in assessing its performance and generalization capabilities. The following subsections present a comprehensive description of their composition and annotation methodologies, followed by an outline of the corresponding baseline model training and evaluation procedures.

5.1 Real-World Data set

The CityScapes data set [COR+16] is a large-scale benchmark focused on semantic understanding of urban street scenes. It provides high-resolution RGB images with dense pixel-level annotations for 30 visual classes, including both semantic and instance-level labels. The data set comprises 5,000 finely annotated images and an additional 20,000 images with coarse annotations, collected across 50 different cities during daytime and under favorable weather conditions.

Although originally designed for semantic segmentation, where each pixel is assigned a semantic class label, this thesis repurposes the CityScapes data set for object detection using the RelTR model. In contrast to segmentation, object detection requires predicting tight bounding boxes around individual object instances, each associated with a class ID. In this work, CityScapes is used to train the first part of the RelTR model, which is responsible for entity (object) detection.

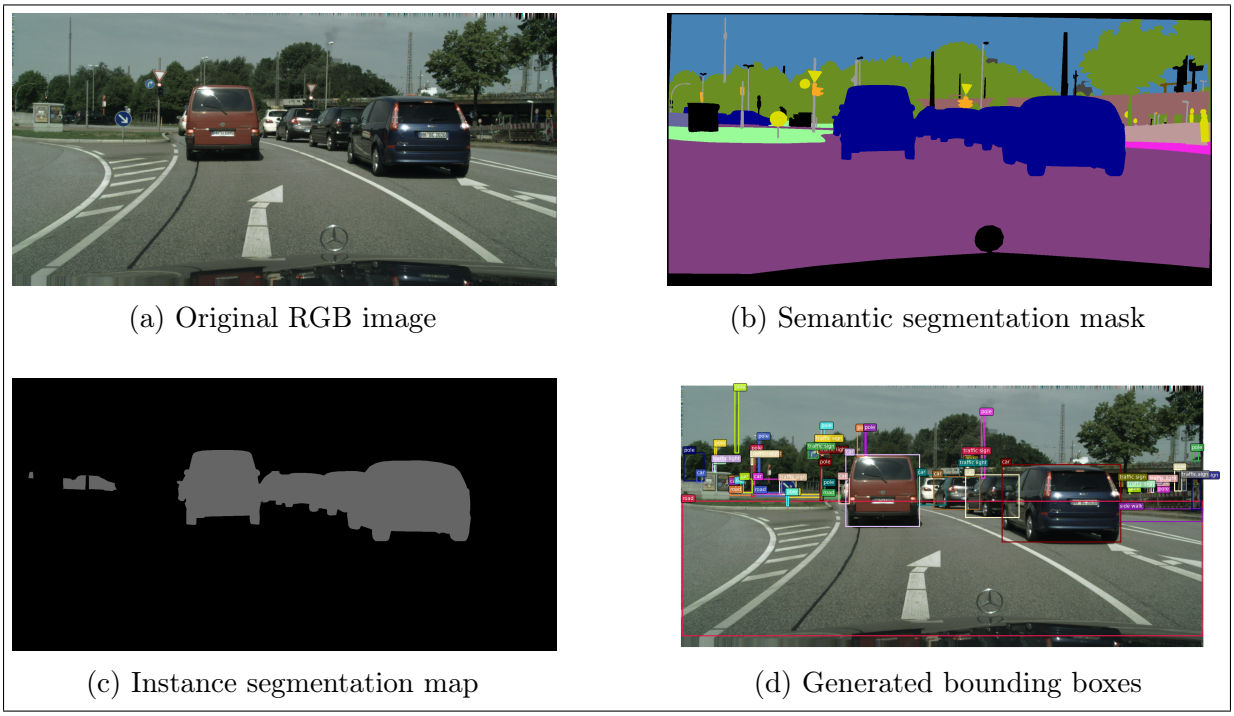
Other data sets such as KITTI [GLU12] and nuScenes [CBL+19] were considered. However, these contain a limited set of object categories and often exclude critical classes such as *road*, *sidewalk*, or *ground*. To overcome these limitations, the CityScapes segmentation maps were leveraged to generate bounding box annotations.

The conversion process is illustrated in Figure 10. Starting from the original RGB image (a), the semantic (b) and instance (c) segmentation masks are used to generate bounding boxes (d) via contour detection and color-based class extraction. The following algorithm was implemented to generate the bounding boxes:

Algorithm 4 Bounding Box Generation from Semantic and Instance Masks

Require: Paths to semantic mask image P_{class} and instance mask image $P_{instance}$

$class\_image \leftarrow \text{ReadImage}(P_{class})$
$instance\_image \leftarrow \text{ReadImage}(P_{instance})$
$instance\_ids \leftarrow \text{UniqueValues}(instance\_image)$
$bounding\_boxes \leftarrow [\,]$
for each $instance\_id$ in $instance\_ids$ **do**
    $mask \leftarrow (instance\_image == instance\_id)$
    $contours \leftarrow \text{FindContours}(mask)$
    for each $contour$ in $contours$ **do**
        $(x, y, w, h) \leftarrow \text{BoundingRect}(contour)$
        $color\_rgb \leftarrow class\_image[\text{row}(contour), \text{col}(contour)]$
        $class\_label \leftarrow \text{CITYSCAPES\_CLASSES}[color\_rgb]$
        if $class\_label \neq$ "void" **then**
            Append $\{\text{instance\_id}, \text{class\_label}, (x, y, w, h)\}$ to $bounding\_boxes$
        end if
    end for
end for
return $bounding\_boxes$



(a) Original RGB image

(b) Semantic segmentation mask

(c) Instance segmentation map

(d) Generated bounding boxes

Figure 10: Conversion of CityScapes annotations into bounding boxes. Starting from the raw RGB input (a), semantic and instance masks (b) and (c) are used to generate tight bounding boxes for object detection (d).

Figure 10 shows the full pipeline, including the generated bounding boxes used for training.

5.1.1 Class Filtering and Renaming

To ensure compatibility between CityScapes and the custom data set generated using the CARLA simulator, additional pre-processing steps were applied. Specifically, classes not represented in the CARLA data set were removed to reduce noise and prevent the model from learning irrelevant categories.

Furthermore, some class labels were renamed to match the taxonomy used in CARLA. For example, the *parking* class was re-labeled as *ground*, and the *rider* class was merged into the more general *person* category. These changes ensured consistency across data sets and reduced semantic ambiguity.

All class mapping decisions are summarized in Table 2.

Table 2: Mapping of CityScapes classes to Carla-compatible labels

Original CityScapes Label	**Mapped Carla Label**
parking	ground
rider	person
caravan	*removed*
trailer	*removed*
train	*removed*
terrain	*removed*
wall	*removed*
fence	*removed*
sky	*removed*
building	*removed*
vegetation	*removed*
tunnel	*removed*
rail track	*removed*
guard rail	*removed*

5.1.2 Train-Val-Test Split

While CityScapes provides a predefined train-validation-test split tailored for semantic segmentation, this split was not optimal for object detection in the context of this thesis, because the test data set does not include the necessary labeling to extract bounding box information as described in Section 5.1. Therefore, a custom split was used: all cities, from the train and validation data sets were combined into the training set, except for Weimar and Zurich, which were used for validation and testing, respectively.

5.1.3 Initial Evaluation on CityScapes

Training the first stage of the RelTR model exclusively on CityScapes resulted in limited qualitative and quantitative performance. A metric evaluation framework was implemented based on standard object detection practices, including precision and recall [NZS23]:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.1a}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.1b}$$

Predictions were matched to ground truth boxes using IoU. A prediction was considered a true positive if its IoU exceeded a threshold (e.g., 0.50) and its class label matched the ground truth. False positives and false negatives were defined accordingly.

Precision-Recall (PR) curves were computed for IoU thresholds ranging from 0.50 to 0.95 by varying the prediction confidence threshold. Figure 11 shows these PR curves, which reveal the model's performance under varying localization strictness.

A clear performance drop with increasing IoU thresholds suggests that the model often makes semantically correct predictions, but has difficulty with precise localization and capturing all relevant objects in the scene. High precision and low recall indicate a conservative prediction style. The low overall recall suggests that many objects are missed.

Visual comparisons in Figure 12 and Figure 13 further confirm this trend, showing sparse predictions, but aligned attention for the most confident predictions.

5.1.4 Data set Expansion with BDD100K and Mapillary

To improve detection performance, two additional data sets with instance segmentation were integrated:

1. BDD100K [YCW+20], contributing 8,000 converted images.
2. Mapillary Vistas [NOBK17], adding 20,000 images.

The same conversion process (Algorithm 4) was applied to these data sets. Since high-quality data were now abundantly available, the 20,000 coarsely annotated CityScapes images were excluded.

This resulted in a curated data set of 33,000 finely annotated images used for training, validation, and testing.

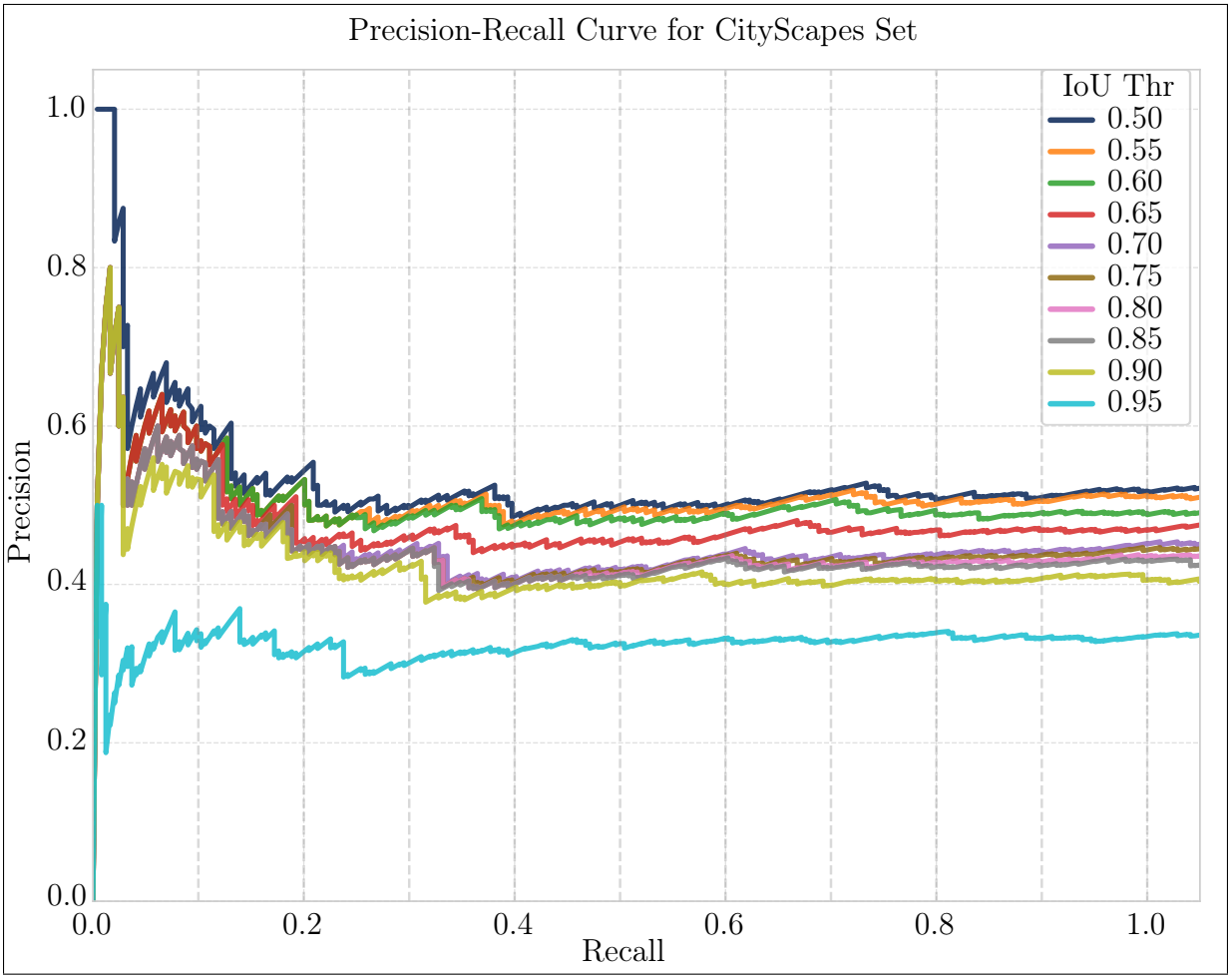
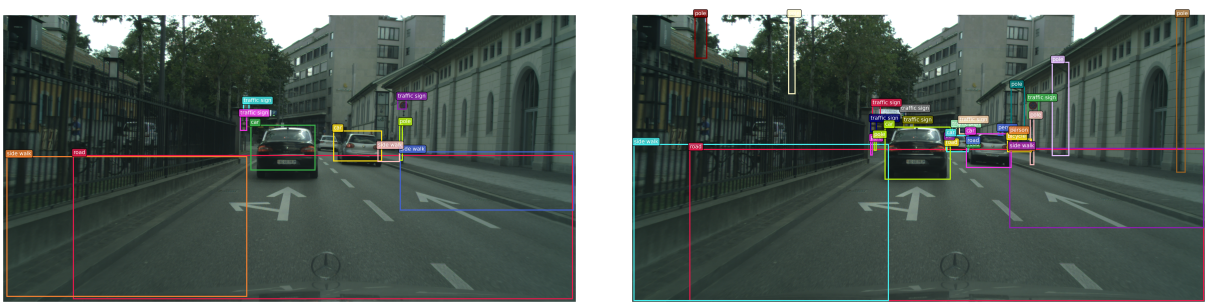


Figure 11: Precision-Recall curves for the CityScapes test set at various IoU thresholds ranging from 0.50 to 0.95.



(a) Prediction Output

(b) Ground Truth

Figure 12: Comparison of prediction and ground truth bounding boxes.

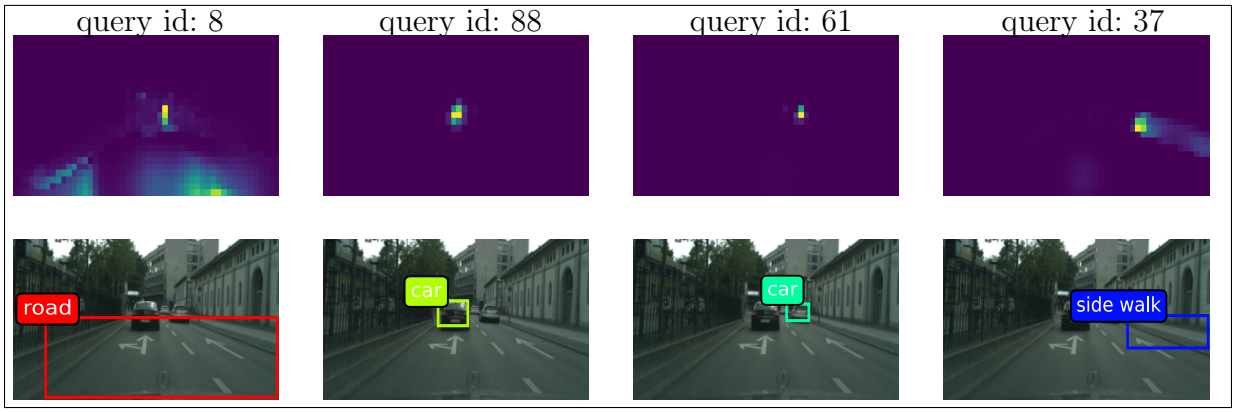


Figure 13: Attention Heatmaps on CityScapes

5.1.5 Data set Distribution Analysis

Figure 21 shows the distribution of object instances and images per class across the training, validation, and test sets. Classes like *pole*, *car*, and *person* dominate the object instance distribution. While classes such as *truck* and especially *bridge* are underrepresented.

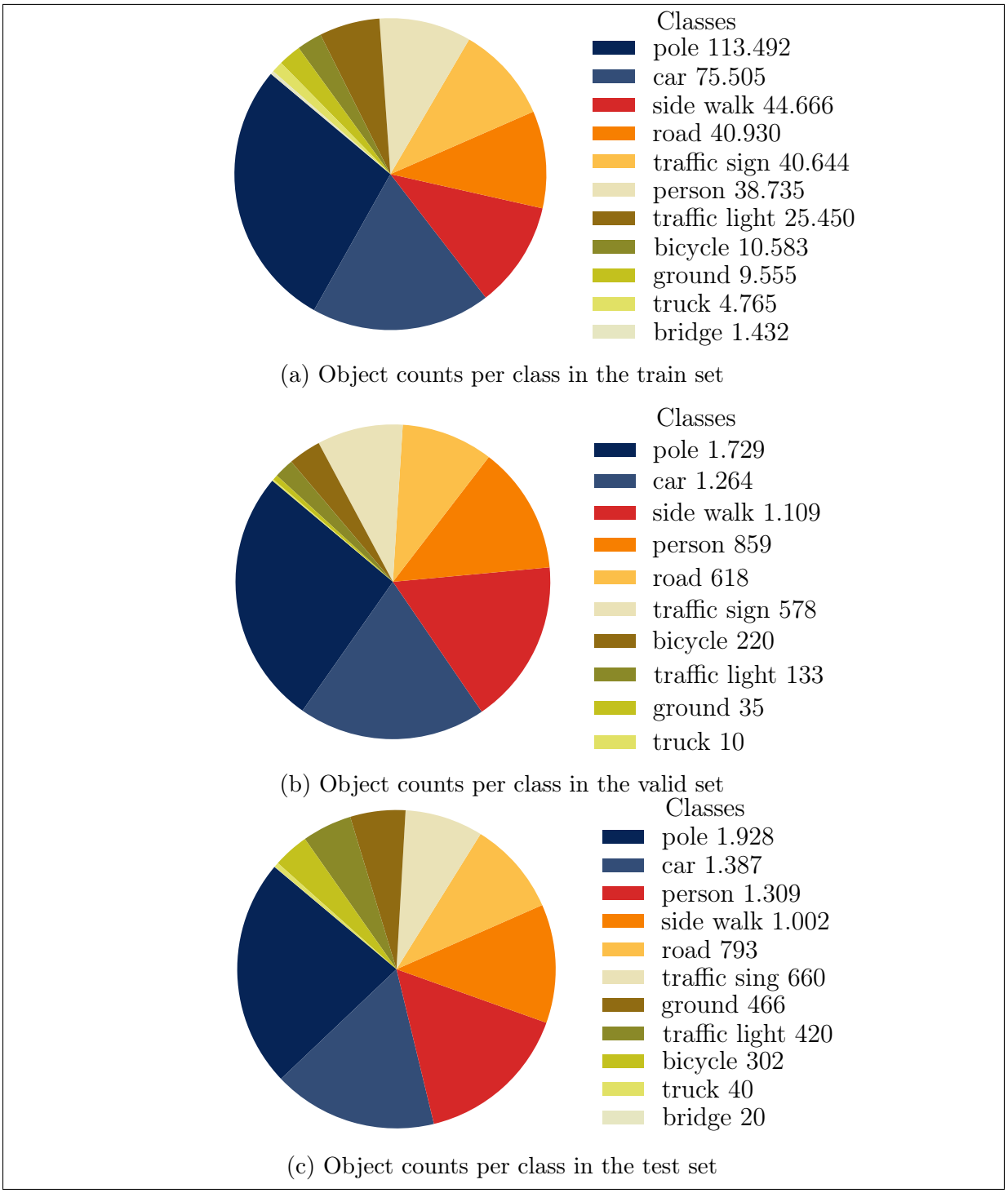
Most images contain at least one instance of *road*, *pole*, or *sidewalk*, confirming their ubiquity in urban scenes.

This distribution pattern is consistent with expectations, given that the data set is automotive-focused. Naturally, objects like *cars*, *traffic signs*, *roads*, and *pedestrians* are highly prevalent, while less commonly encountered elements in traffic scenes — such as *bridges* or *trucks* — occur less frequently.

5.1.6 Final Training Setup

Training was conducted using the default hyperparameters provided in the original RelTR paper [CYR23], with no additional tuning. Learning rate scheduling followed the strategy used in DETR [CMS+20], including a step drop after 400 of the 500 total training epochs. A detailed configuration overview, including system specifications and batch sizes, is provided in Table 3.

Using the same evaluation method as described earlier, the model's performance on the real-world data set is shown in Figure 16, where PR curves are plotted across IoU thresholds from 0.50 to 0.95. For thresholds between 0.50 and 0.90, precision remains consistently high, exceeding 0.8 across most recall levels. Each curve starts at a precision of 1.0 and gradually declines toward 0.9 as recall increases, indicating that the model produces reliable, high-confidence predictions and maintains accuracy even as more predictions are considered by lowering the confidence threshold.



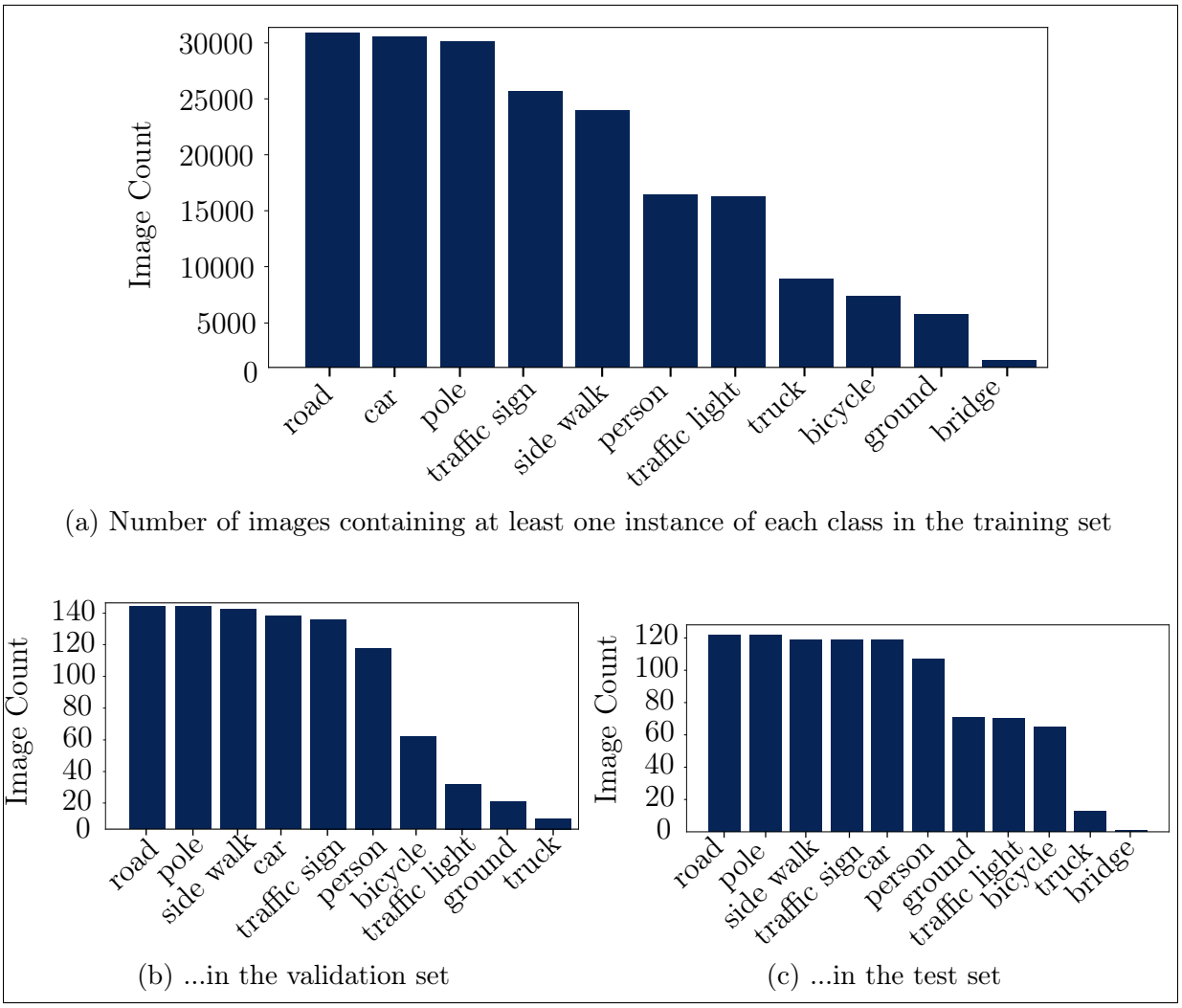
(a) Object counts per class in the train set

(b) Object counts per class in the valid set

(c) Object counts per class in the test set

Figure 14: Distribution of object instances per class in the real-world data set. The top plot shows the class distribution in the training set, while the two plots below represent the validation and test sets, respectively

At an IoU threshold of 0.95, a distinct deviation from this trend is observed. The curve begins at a substantially lower precision of around 0.3 and rises slowly to approximately 0.7. This behavior suggests that, under such a strict spatial alignment requirement, even the most confident predictions often fall short in terms of bounding box accuracy, although



(a) Number of images containing at least one instance of each class in the training set

(b) ...in the validation set

(c) ...in the test set

Figure 15: Image distribution per object class in the real-world data set. The plots indicate how many images contain at least one instance of each object class in the training (top), validation (bottom left), and test sets (bottom right).

classification remains largely correct. Since the target application does not require highly precise localization but rather correct detection and classification, IoU thresholds between 0.50 and 0.80 are more relevant for evaluating practical performance.

Overall, the model exhibits robust detection behavior within the practically relevant IoU range. High precision across these thresholds indicates a low false positive rate, while the performance drop at 0.95 IoU is not indicative of a critical flaw but rather reflects an evaluation criterion that surpasses the necessary spatial granularity for the intended task.

Visual comparisons in Figure 17 show the improvement made by the expansion of the data set.

Parameter	Value
Encoder layers	6
Decoder layers	6
Hidden dimension	256
Feedforward dimension	2048
Number of attention heads	8
Number of entity queries	300
Number of triplets	200
Dropout	0.1
Optimizer	AdamW
Learning rate (base)	1×10^{-4}
Learning rate (backbone)	1×10^{-5}
Weight decay	1×10^{-4}
Epochs	500
Learning rate drop epoch	400
Batch size	12
GPUs	8 × NVIDIA A100 (40 GB)
Framework	PyTorch 2.1.2
System	JUWELS Booster (Forschungszentrum Jülich)

Table 3: Training configuration and hyperparameters used for training RelTR model on the curated data set.

5.2 CARLA Data set

One of the main contributions of this thesis is the creation of a new data set based on the CARLA simulator [DRC+17]. This data set captures semantic relationships between traffic participants and static infrastructure elements in the form of subject-predicate-object triplets. The data set was generated using a custom annotation tool developed as part of a previous student research project at Coburg University. While the internal workings of the tool lie beyond the scope of this thesis, its core functionalities are briefly outlined below.

The data set was generated using the CARLA simulator, specifically the ScenarioRunner [CAR23] extension, which provides a range of predefined traffic scenarios. These include vehicle-following sequences, turning maneuvers at intersections, and interactions with cross-traffic. Each scenario was recorded from seven different camera perspectives: rear view (centered), rear-left and rear-right, side views (left and right), and front-left and front-right. In addition to RGB data, segmentation, instance, and depth information were also recorded to support the annotation pipeline.

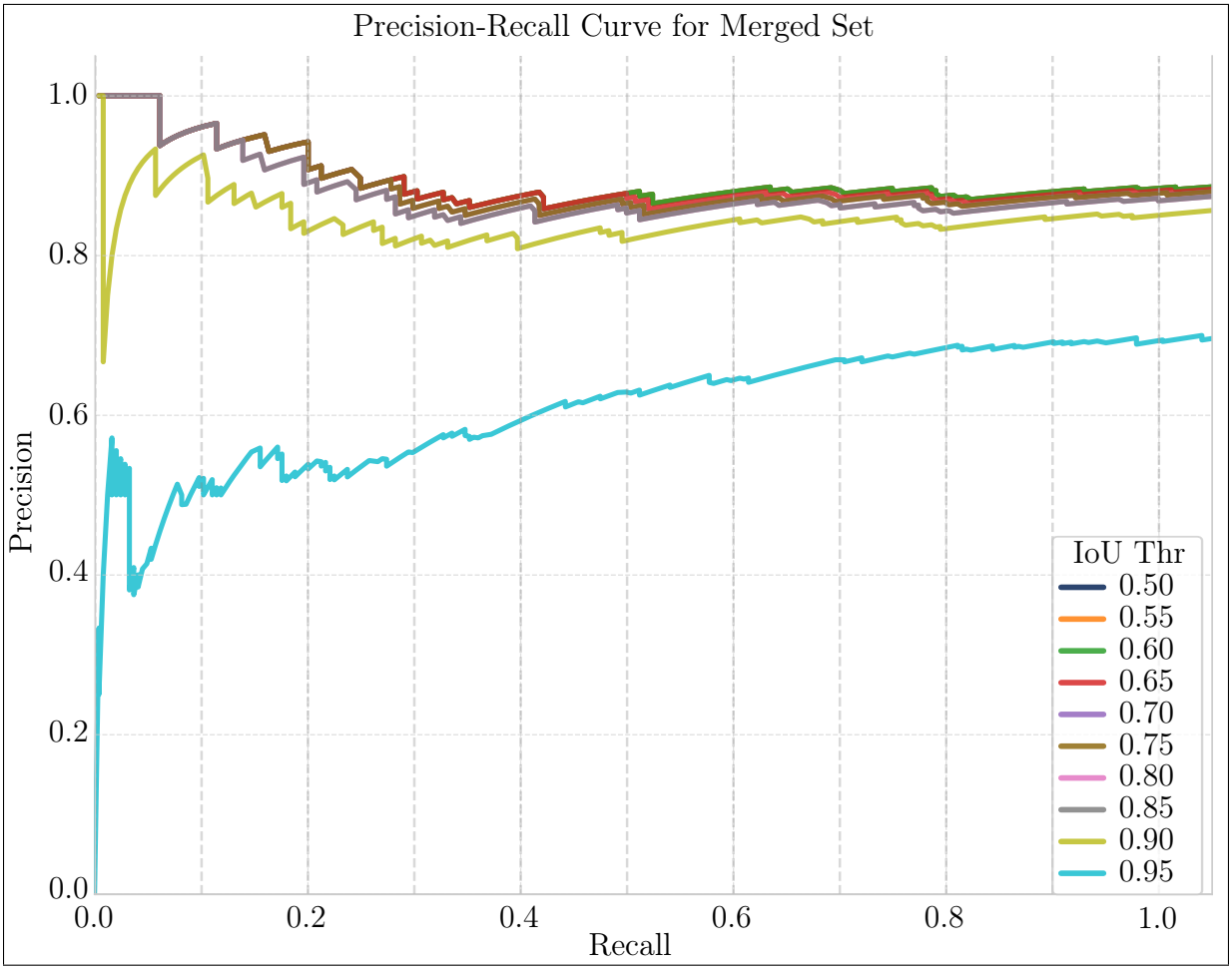
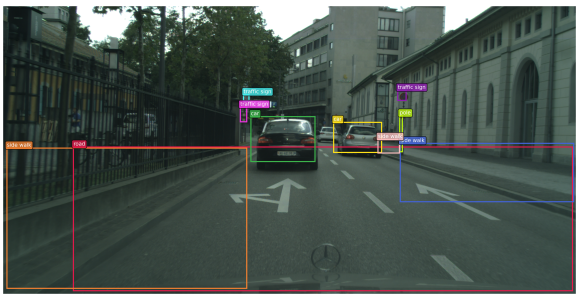
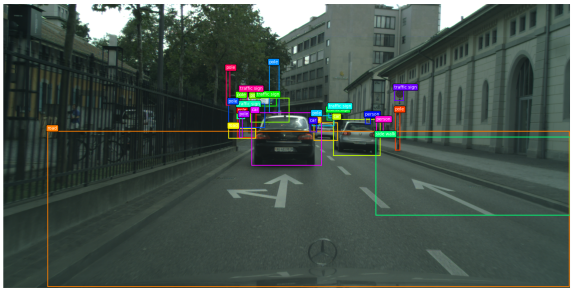


Figure 16: Precision-recall curves for the real-world test set at various IoU thresholds ranging from 0.50 to 0.95.



(a) Prediction output before expansion



(b) Prediction output after expansion

Figure 17: Comparison of prediction before and after expansion

The annotation tool processes each recorded frame by extracting metadata from the simulator, computing two-dimensional bounding boxes, and assigning semantic labels based on object categories and positions. When instance IDs are unavailable, image masks are used to determine object extent. The tool further refines bounding boxes for certain object types and enforces annotation consistency using a predefined lookup table.

All annotations are exported in structured JSON format, enabling efficient access and integration with machine learning pipelines.

The definition of annotated relationships was guided by the overarching research goal of Work Package 1.1 of the funded project nxtAIM [Rei24], which seeks to establish semantic metrics for comparing real and generated images in the autonomous driving domain. The central question driving this effort was: *Given a model that generates photorealistic images, which semantic relationships between objects must be preserved to consider the image valid or useful for downstream tasks?*

This guiding question led to the formulation of specific relational predicates, each motivated by typical traffic scenarios and spatial constraints:

Spatial placement of infrastructure elements: For instance, the relationship *pole on side walk* ensures that poles appear in plausible positions.

Attachment relationships: *traffic sign attached to pole* and *traffic light attached to pole* ensure that signs and lights appear mounted to supporting structures.

Vehicle-road alignment: *car on right/left side of road* and
car on left/middle/right lane of road enforce proper lane placement.

Vehicle-vehicle alignment: The relationship *car on same road line as* ensures that vehicles traveling in a platoon are aligned.

Relative positions: *object in front of car* and *object behind car* capture critical proximity relationships relevant for collision avoidance.

Driving maneuvers: *car turning left/right on road* represents dynamic interactions during turning events.

Collisions: The relationship *car is hitting object* encodes the occurrence of physical contact and serves as a marker for critical failure cases.

These relational labels collectively capture both structural and dynamic elements of realistic traffic scenes, making the data set suitable for evaluating SGG in autonomous driving contexts.

5.2.1 Data set Statistics

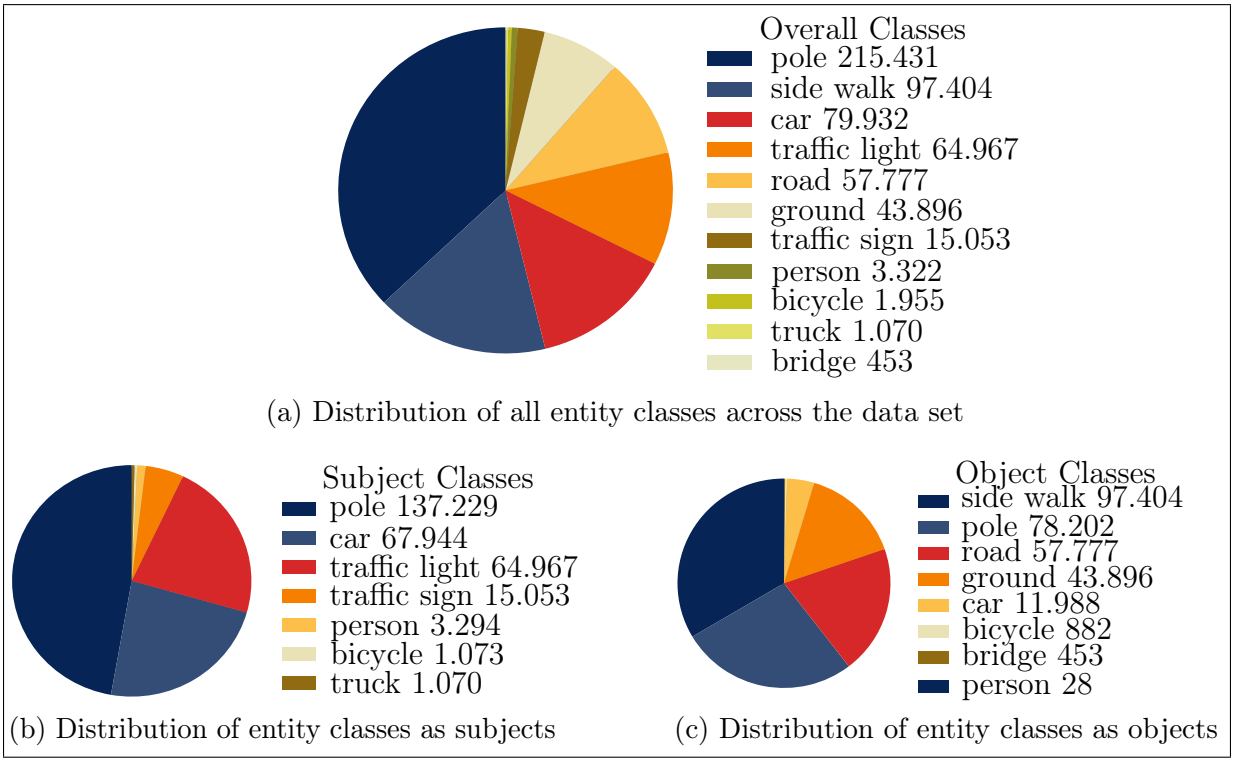
To provide a deeper understanding of the generated data set, several statistics are presented and critically analyzed below.

Figure 18(a) shows the overall distribution of all entity classes across the data set. It is evident that static infrastructure elements dominate the scenes, with *pole* instances

representing 37.06% of all entities, followed by *sidewalks* and *cars*. Other prominent classes include *traffic light* (11.18%) and *road* (9.94%). In contrast, dynamic elements such as *person* (0.57%), *bicycle* (0.34%), and *truck* (0.18%) are considerably less frequent. This skew towards static elements reflects the nature of urban scenarios in CARLA but reduces the data set's overall diversity.

Figure 18(b) displays the distribution of entity classes used as subjects in annotated relationships. Here, *pole* accounts for 47.2% of all subject entities, followed by *car* (23.4%) and *traffic light* (22.4%). Dynamic entities such as *person* (1.1%), *truck* (0.4%) and *bicycle* (0.4%) remain underrepresented, suggesting that the majority of relational dynamics are based on static infrastructure and vehicle interactions, rather than complex human-object or vehicle-vehicle relations.

Figure 18(c) illustrates the distribution of classes serving as objects in relationships. *Sidewalk* (33.51%), *pole* (26.91%), and *road* (19.88%) dominate, reinforcing the prevalence of static environmental elements. Dynamic objects such as *car* (4.12%) and *bicycle* (0.3%) make up only a minor fraction. This strong imbalance is expected but noteworthy for later model evaluation.



(a) Distribution of all entity classes across the data set

(b) Distribution of entity classes as subjects

(c) Distribution of entity classes as objects

Figure 18: Entity class distributions across the data set

Figure 19 presents the frequency distribution of all annotated relationship types. The most frequent relationships involve static interactions, such as *pole on sidewalk* and *traffic light attached to pole*. Relationships involving dynamic objects are, as anticipated, much less common. Given the scenario design, this imbalance is acceptable.

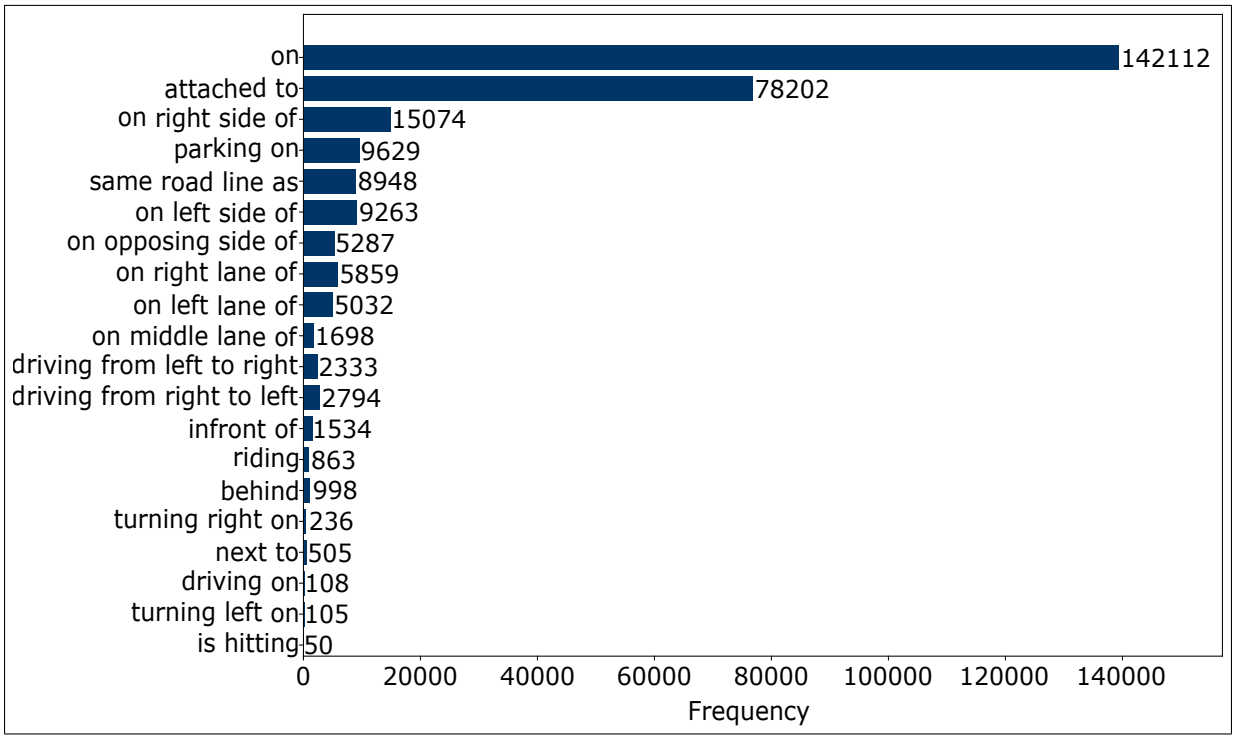


Figure 19: Frequencies of all annotated relationship types

A more concerning imbalance is observed among lane-specific relationships. *On left lane of* (5032 occurrences) and *on right lane of* (5859 occurrences) are significantly more frequent than *on middle lane of* (1698 occurrences), which could bias the model towards side lanes in multi-lane scenes.

Relationships describing turning maneuvers are particularly underrepresented, with only 236 instances of *turning right on* and 105 instances of *turning left on*. These low frequencies pose a risk that the model may not adequately learn such interactions, despite their relevance in real-world scenarios.

Further imbalance is found among road-side placement relations. While *on right side of* (15,074) and *on left side of* (9263) are relatively common, *on opposing side of* is annotated only 5287 times. Nevertheless, the visual distinction between these categories may reduce the impact of this imbalance on model performance.

Lastly, the relationship *is hitting* is extremely rare, with just 50 annotations. This class was deliberately included to illustrate that synthetic simulation environments can represent ethically challenging interactions which would be impractical or unethical to capture in real-world data sets.

In summary, while the data set provides a comprehensive foundation for modeling scene graphs in urban environments, it suffers from significant class imbalances—particularly between static and dynamic elements, and across relationship types. These imbalances

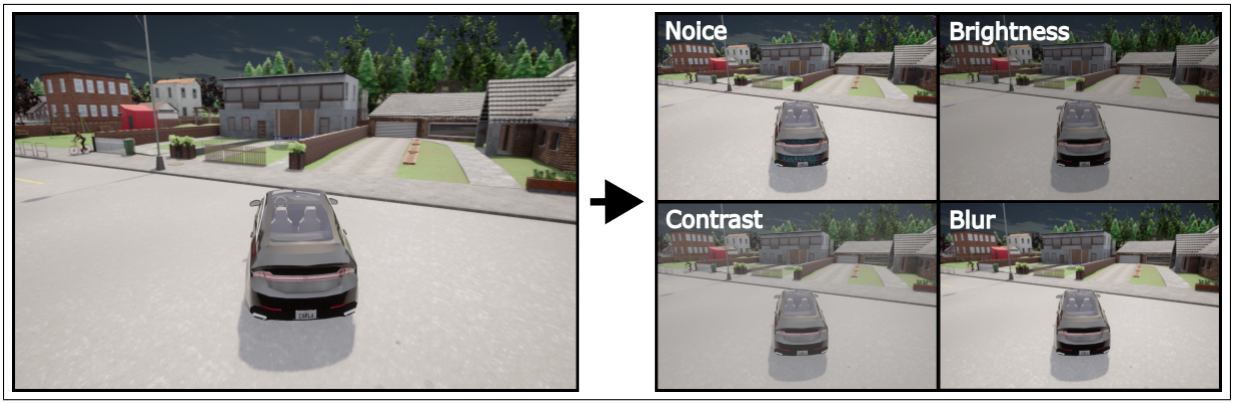


Figure 20: Visual example of the applied augmentation techniques

motivated the application of targeted augmentation strategies, which are discussed in the next section.

5.2.2 Data set Augmentation

To address the data set's class imbalance—particularly concerning underrepresented relationship classes such as *turning left on*, *turning right on*, and *on middle lane of*—a targeted data augmentation strategy was implemented.

The process starts by identifying all frames containing at least one instance of a rare relationship. For each of these frames, a set of augmentation techniques was applied independently to the original image, aiming to increase data diversity while preserving the underlying semantic structure. The following transformations were used:

- Addition of random Gaussian noise
- Random brightness adjustment
- Random contrast adjustment
- Application of a random blur filter

Each transformation produced one augmented variant of the original image, resulting in four new samples per frame. By applying these augmentations independently, cumulative distortions were avoided, and the integrity of spatial relationships within the scene was maintained.

Figure 20 illustrates an example from the augmentation process. The original image is shown alongside its four augmented versions, highlighting the visual diversity introduced through this pipeline.

5.2.3 Impact of Augmentation on Data set Distribution

The targeted data augmentation strategy led to measurable improvements in the distribution of both entity classes and relationship types.

For the overall entity distribution, a slight increase in dynamic object categories was observed. The proportion of *person* instances rose from 0.57% to 0.86%, *bicycle* instances increased from 0.34% to 0.75%, and *truck* instances grew from 0.18% to 0.38%. These shifts reflect the focus on augmenting frames containing underrepresented dynamic elements.

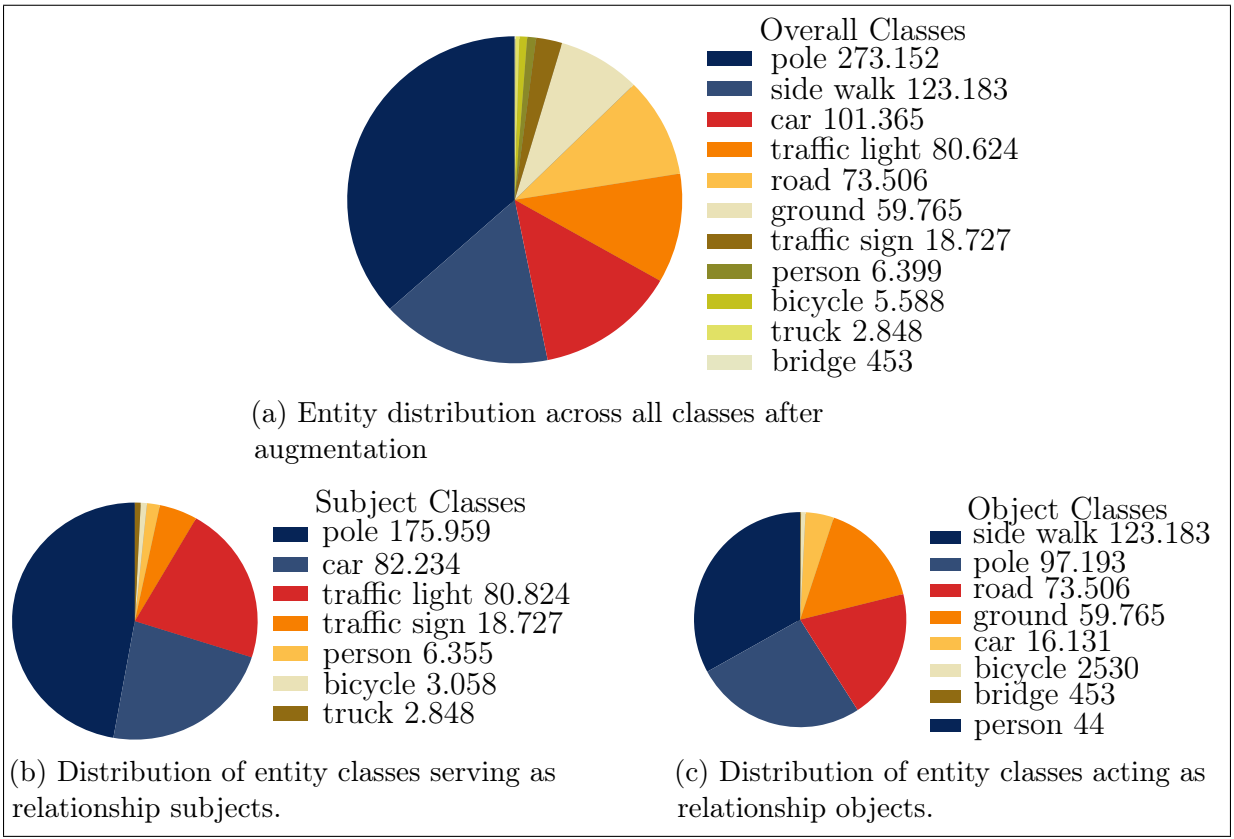
More substantial effects were achieved in the distribution of relationship classes. The number of instances of the *on middle lane of* relationship more than doubled, increasing from 1,608 to 3,957 occurrences. Similarly, the frequencies of the previously rare *turning right on* and *turning left on* relationships increased from 236 to 828 and from 105 to 348, respectively. Meanwhile, frequently occurring classes such as *on right lane of* (5,859 occurrences) and *on left lane of* (5,032 occurrences) remained stable, as they were not selected for augmentation.

Although it would have been technically feasible to further expand the data set by repeatedly augmenting the same scenes with additional variations, this approach was intentionally avoided. Excessive duplication would have increased the data set size without contributing meaningful new information and might have introduced redundancy, reducing the data set's effectiveness during model training. The chosen augmentation scope therefore strikes a balance—enhancing diversity in critical areas without inflating the data set unnecessarily.

Overall, the adjustments successfully mitigated class imbalance, particularly for dynamic interactions and lane-specific relationships. The resulting data set is better equipped to support robust learning of rare but semantically important relational patterns, thereby improving model generalization.

5.2.4 Baseline Performance on Simulated Data

To establish a performance baseline for subsequent Sim2Real experiments, the complete RelTR model was trained and evaluated exclusively on the simulated data set. The training followed the same hyperparameter configuration as outlined in Table 3. Evaluation was performed using the widely adopted Recall@K metric for SGG, which measures the fraction of ground truth relationships that appear among the top-K highest-confidence predictions [LZZ+24]. Although the Precision@K metric is also commonly reported [CYR23], it is omitted in this thesis, as the primary objective is to determine whether and how domain gap closure can be achieved, rather than to assess the absolute precision of the approaches.



(a) Entity distribution across all classes after augmentation

(b) Distribution of entity classes serving as relationship subjects.

(c) Distribution of entity classes acting as relationship objects.

Figure 21: Updated entity distributions after targeted augmentation

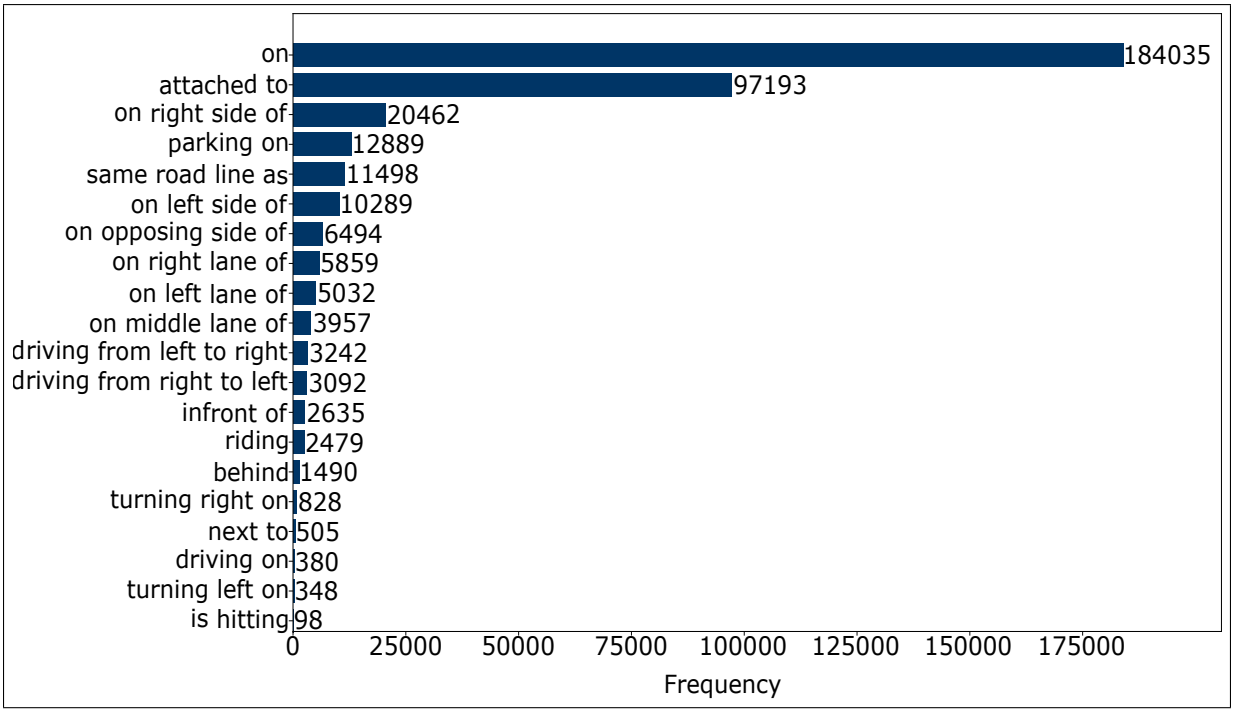


Figure 22: Relationship class frequencies after augmentation

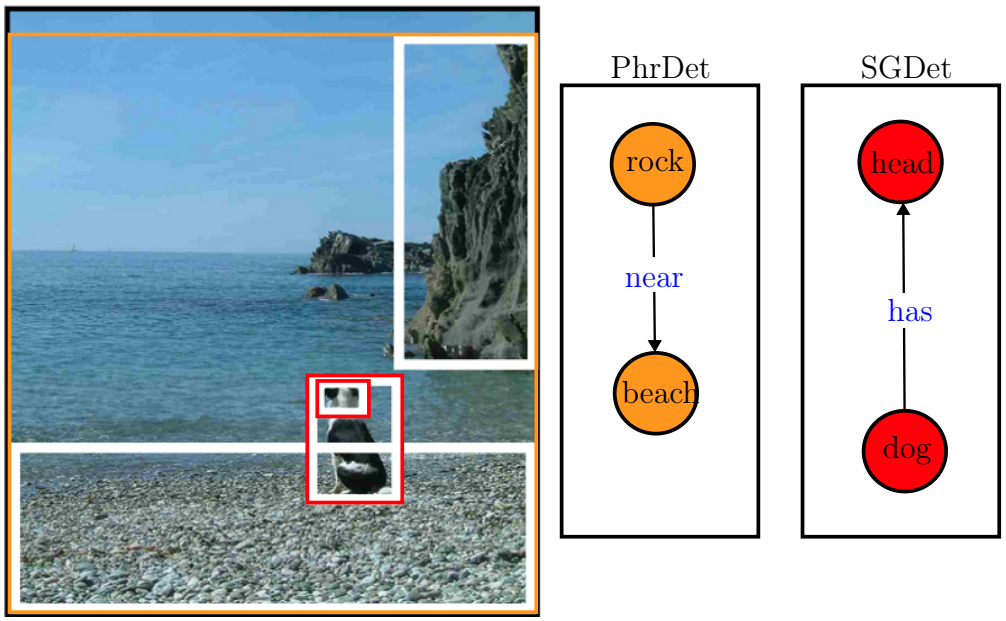


Figure 23: Visual representation of the criteria for a correct prediction. For PhrDet *rock near beach* (orange) and SGDet *dog has head* (red)

Common values for K include 50 and 100; this work additionally includes the more stringent setting of K=20 to enable direct comparison with the original RelTR paper [CYR23].

SGG models are typically evaluated across four standard tasks:

1. Phrase Detection (PhrDet): Predicts a subject–predicate–object triplet and localizes the entire relationship with a bounding box that overlaps the ground truth by at least 50%.

2. Predicate Classification (PredCls): Predicts the relationship between object pairs, given ground truth object locations and labels.

3. Scene Graph Classification (SGCls): Classifies both objects and relationships, given ground truth object locations.

4. Scene Graph Detection (SGDet): Detects and classifies both objects and relationships, requiring predicted object bounding boxes to overlap with the ground truth by at least 50%. While referred to as SGG in literature [LZZ+24], this thesis adopts the SGDet naming convention used in [CYR23] to better distinguish between the class of models and the task.

Visual representations of the criteria for PhrDet and SGDet are given in Figure 23.

This evaluation focuses on tasks that do not rely on ground truth object input, specifically PhrDet and SGDet. Results are compared against those reported in the original RelTR work [CYR23], as well as expectations derived from the distribution of relationship classes in the simulated data set.

Recall@K	**PhrDet**		**SGDet**	
	Original	**Simulated**	**Original**	**Simulated**
20	Not reported	63	21.2	40
50	34.5	68	27.5	41
100	39.8	69	Not reported	41

Table 4: Comparison of Recall@K for PhrDet and SGDet between the original RelTR model and the version trained on the simulated data set.

The results indicate that PhrDet consistently achieves higher recall than SGDet, which is expected due to its less stringent evaluation criteria. Although overall recall remains moderate, the model performs competitively given the smaller number of object and relationship classes, as well as the reduced data set size compared to [CYR23]. Furthermore, recall scores demonstrate minimal improvement as K increases. This suggests that the model's high-confidence predictions dominate the output and that adding lower-ranked predictions provides little additional value.

To further investigate model behavior, per-class recall was computed for selected relationship types.

Several notable patterns emerge from the per-class recall. The relationship *on* shows a pronounced gap between PhrDet and SGDet. This likely stems from bounding box flexibility for broad entities such as roads or sidewalks. As illustrated in Figure 24 (top) and Figure 24 (bottom), predictions may semantically align with the ground truth but fail to meet the stricter SGDet threshold.

Positional relationships such as *on right side of* and *on middle lane of* achieve near-perfect accuracy in both tasks. In contrast, relationships involving relative motion direction—such as *turning right on* and *turning left on*—are predicted poorly. This likely reflects the difficulty of inferring motion direction from static images. Interestingly, recall for *turning right on* is zero across all K values. While the class is underrepresented, as shown in Figure 22, frequency alone does not fully account for the discrepancy.

Notably, rare relationships can still be learned effectively. The relationship *is hitting* appears only 98 times in the data set but achieves perfect PhrDet recall. However, SGDet recall remains low due to bounding box inaccuracies. This suggests that even infrequent relational patterns can be captured by the model if visually distinct.

Table 5: Per-class recall across tasks and K values

Class	K	SGDet	PhrDet
on	20	15.0	51.9
	50	17.2	61.7
	100	17.7	66.4
on right side of	20	87.8	98.1
	50	87.9	99.7
	100	87.9	99.8
on middle lane of	20	98.9	100
	50	98.9	100
	100	98.9	100
turning right on	20	0.0	0.0
	50	0.0	0.0
	100	0.0	0.0
turning left on	20	16.6	33.9
	50	16.6	33.9
	100	16.6	33.9
is hitting	20	2.6	100
	50	2.6	100
	100	2.6	100
same road line as	20	16.1	21.2
	50	16.1	46.3
	100	16.1	47.0

In addition to recall-based evaluation, qualitative analysis of the model's behavior was performed by visualizing attention maps. Following the methodology in [CYR23], the cross-attention layer of the DVA module (Equation 2.17) was analyzed. This component directly influences predicate prediction (Equation 2.23) and integrates visual and relational cues, making it particularly informative for understanding prediction decisions.

Figure 25 presents attention maps for a simulated scene. For query ID 161, predicting *on right side of*, attention is sharply focused on road markings on the right, supporting the plausibility of the prediction. Query ID 16 predicts *on* between a pole and a sidewalk, with correct relationships and bounding boxes, also attention appears to be focused on the correct side walk. Query ID 179 demonstrates a clear failure: attention is placed on a car, but the model predicts a relationship (*riding*) between a person and a bicycle, neither of which is present.

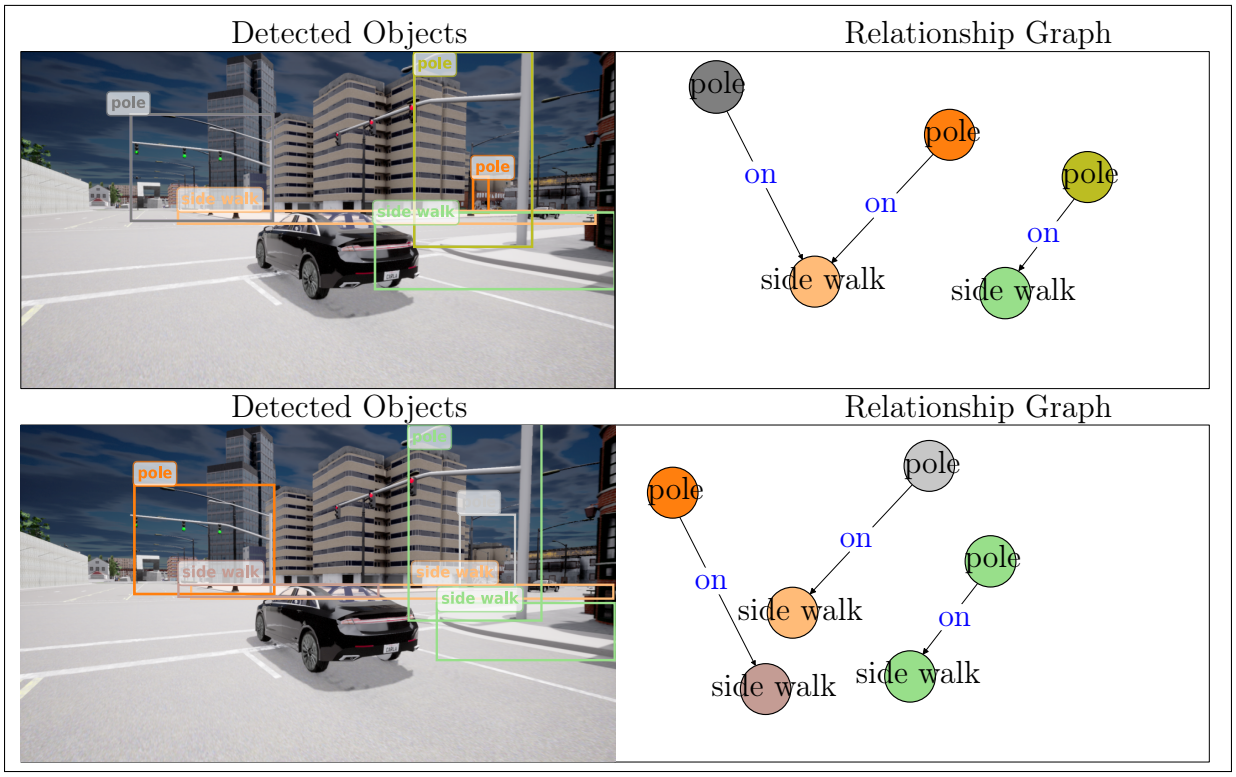


Figure 24: Ground truth (top) against inference (bottom) for the baseline model. Bounding boxes and relationship graph where filtered by hand to only show the relationship triplet for *pole on side walk* for the relevant subjects and objects

5.2.5 Conclusion

The baseline evaluation of the RelTR model trained exclusively on the simulated data set reveals both strengths and limitations in its ability to learn and generalize relational reasoning in structured visual environments. Quantitatively, the model achieves moderate performance on PhrDet and lower scores on SGDet, with minimal improvements as K increases on Recall@K metrics. This indicates that while the model is confident in its top predictions, it frequently fails to produce sufficiently accurate object localizations to satisfy the stricter criteria of SGDet.

Per-class analysis demonstrates that spatially grounded and visually salient relationships such as *on right side of* and *on middle lane of* are learned reliably, achieving near-perfect recall even under SGDet conditions. In contrast, relationships requiring inference of dynamic properties (e.g., *turning right on*) or subtle positional semantics (e.g., *same road line as*) remain challenging. These difficulties stem from a combination of data set imbalance, visual ambiguity, and the inherent limitations of static imagery for representing motion-related interactions. Notably, the model achieves high recall for visually distinct but infrequent relationships such as *is hitting* in the PhrDet task, indicating that even rare classes can be learned when sufficient visual cues are present.

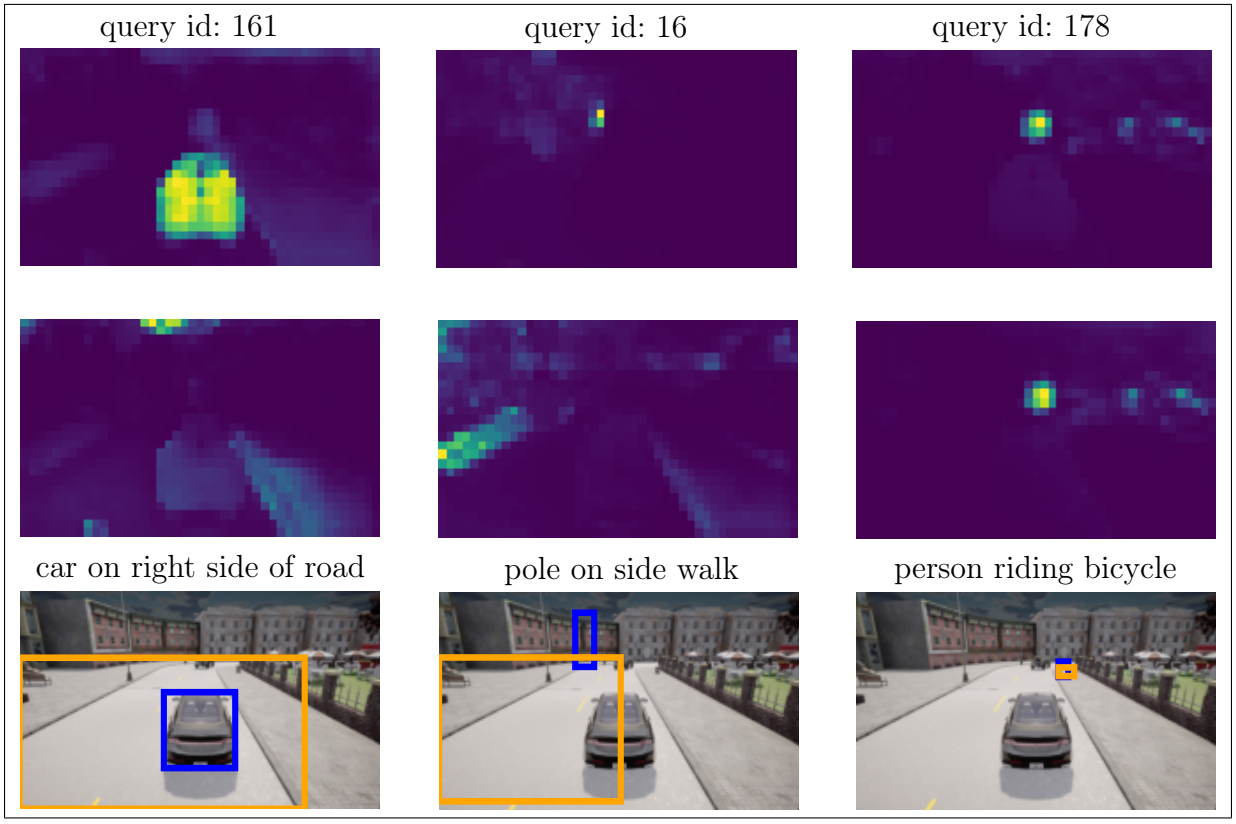


Figure 25: Attention heatmaps of the baseline model for simulated scene. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

Qualitative inspection of cross-attention maps supports these observations. Accurate predictions are typically accompanied by focused attention on semantically meaningful regions (e.g., lane markings, object contours). Hallucinated predictions—such as relationships involving absent entities—further highlight the potential for overfitting to simulation-specific artifacts or insufficient variation in training viewpoints.

Taken together, these findings provide a robust performance baseline for subsequent Sim2Real transfer investigations. Importantly, they also address the first research question posed in this thesis: *Are the annotated relationships in the simulated data set learnable by the selected model?* The empirical results demonstrate that this question can be answered affirmatively. The RelTR model is capable of learning both frequent and infrequent relationships, provided that they are visually grounded and sufficiently represented within the data set. This confirms the effectiveness of the annotation scheme and supports the validity of using the simulated data set for relational reasoning tasks in downstream applications. The only notable exception is the *turning right on* relationship, which remains difficult for the model to learn reliably—likely due to its inherent visual ambiguity and lower representation in the data set, despite augmentation efforts.

6 Sim To Real Approaches

To address the gap between simulation and the real-world domain, two transfer learning strategies were employed, each implemented in multiple variants. These approaches differ in the extent to which pre-trained knowledge is retained or adapted during training on real-world data. The strategies, along with their respective variants, are described and evaluated in detail in the subsequent sections.

6.1 Sequential Freeze Strategy

In this approach, the model is initialized with weights pre-trained on a real-world data set (Section 5.1), maintaining a strict separation between the entity detection part of the model (feature extractors and shared components) and the task-specific layers, thus this approach is entitled as Frozen Entity Detection (FED). To ensure the task-specific layers receive meaningful input, simulated data annotated only with entity classes and bounding boxes were added to the real-world data set. During training, the model was validated exclusively on a test set containing real data, and the validation loss never exceeded that of a model trained solely on real-world data.

Pre-trained weights are loaded into the full model; only parameters not included in the checkpoint are marked as trainable, while all matching parameters are explicitly frozen by setting `requires_grad = False`. To further stabilize the frozen components, their corresponding submodules are set to evaluation mode (e.g., `model.backbone`, `transformer.encoder`, and entity-related decoder layers). This prevents stochastic layers such as dropout or batch normalization (if present) from modifying the learned representations during training. Only the task-specific layers related to relationship detection are updated.

Trainable parameters—those not found in the pre-trained checkpoint—are reinitialized using kaiming initialization for convolutional and linear layers, and constant initialization for biases and normalization layers. This controlled freezing strategy preserves the core visual and semantic representations learned from real-world data while enabling effective adaptation to the simulated domain through the task-specific components. For clarity on which parameters are frozen or reinitialized, see Table 6, where the module names correspond directly to those introduced in Section 2.

Training configurations are summarized in Table 3.

Before evaluating on real-world inference data, the model is first tested on the baseline training data set to enable a controlled comparison assessing the effect of pretraining the encoder on real data.

Table 6: Overview of frozen and reinitialized modules in the sequential freeze strategy.

Module	Status
Feature Extraction	
CNN Backbone	**Frozen**
Transformer Encoder	
Encoder	**Frozen**
Transformer Decoder	
Entity Layer	**Frozen**
Coupled Self-Attention	**Initialized**
Decoupled Visual Attention	**Initialized**
Decoupled Entity Attention	**Initialized**
Prediction Heads	
Entity Class	**Frozen**
Entity Bounding Boxes	**Frozen**
Subj/Obj Class	**Initialized**
Subj/Obj Bounding Boxes	**Initialized**

Table 7: Comparison of mean recall at K for PhrDet and SGDet between the baseline and FED model

Task	K	Baseline	FED
PhrDet	20	63	45
	50	68	48
	100	69	48
SGGDet	20	40	42
	50	41	43
	100	41	43

From Table 7, two key observations emerge: (1) Pre-training the entity detection component on real-world data improves object localization accuracy, as reflected by higher recall in the SGDet task, which requires both subject and object bounding boxes to have at least 50% overlap with ground truth. (2) Conversely, performance on the PhrDet task decreases, possibly because while more precise localization benefits strict spatial relationships, the frozen entity encoder may lack flexibility to detect a broader variety of relationship instances.

To investigate further, a per-class recall analysis was conducted for both SGDet and PhrDet tasks (see Table 8).

Table 8: Per-class recall values for SGDet and PhrDet across different K values on simulated data of the baseline and FED model

Class	K	SGDet		PhrDet	
		Baseline	FED	Baseline	FED
on	20	15.0	44.0	51.9	86.9
	50	17.2	45.8	61.7	88.9
	100	17.7	46.2	66.4	89.4
on right side of	20	87.8	90.2	98.1	97.3
	50	87.9	90.3	99.7	97.4
	100	87.9	90.3	99.8	97.4
on middle lane of	20	98.9	0.0	100	0.0
	50	98.9	0.0	100	0.0
	100	98.9	0.0	100	0.0
turning right on	20	0.0	0.0	0.0	0.0
	50	0.0	0.0	0.0	0.0
	100	0.0	0.0	0.0	0.0
turning left on	20	16.6	94.5	33.9	98.8
	50	16.6	94.5	33.9	98.8
	100	16.6	94.5	33.9	98.8
is hitting	20	2.6	0.0	100	0.0
	50	2.6	0.0	100	0.0
	100	2.6	0.0	100	0.0
same road line as	20	16.1	0.0	21.2	18.1
	50	16.1	20.5	46.3	21.2
	100	16.1	20.5	47.0	21.2

The results in Table 8 support the hypothesis: for relationship classes where the FED model still produces predictions, recall often improves compared to the baseline. For example, classes such as *on*, *on right side of*, and *turning left on* show substantial recall gains. However, some classes (e.g., *on middle lane of* and *is hitting*) lose all detection capability. This suggests that the pre-trained encoder may not have captured certain semantic nuances necessary for these relationships, likely because these concepts were not relevant during pre-training on the entity detection task.

A similar but less consistent trend is observed in the PhrDet task. For instance, recall for the *same road line as* class deteriorates despite the model still making predictions.

It is important to note that the main motivation for including real data in the encoder and freezing it was not to improve performance on simulated data but to transfer learned knowledge for inference on real data.

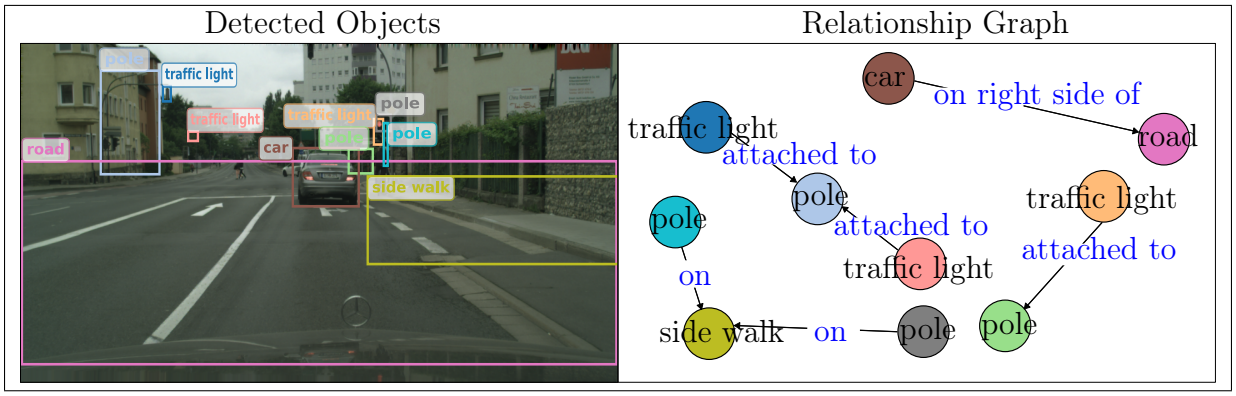


Figure 26: Inference results on a CityScapes data set image using the FED model. The relationship graph was cleaned by hand for better visuals.

Figure 26 illustrates inference by the frozen model on a real-world image from the CityScapes data set. While the model is not flawless—hallucinations are especially noticeable in the pole class where it predicts more poles than present—the detected relationships are mostly accurate and semantically meaningful.

To further assess whether the model maintains meaningful attention patterns, attention maps were compared with those produced by a model trained solely on simulated data. As shown in Figure 27, the frozen model produces plausible attention maps. However these are less sharply focused than the purely simulation-trained model's attention maps (e.g., Figure 25).

Table 9 presents the performance of the model on a small, manually annotated real-world data set containing relationship annotations. On the SGDet task, performance remains generally low across all classes, indicating that the predicted bounding boxes often do not meet the required IoU thresholds for valid triplet matches. This suggests that the localization component of the model struggles to generalize well to the real-world domain. In contrast, the PhrDet metric shows comparatively better results, particularly for the classes *on*, *on right side of*, and *on left side of*. These road-specific spatial relations appear to transfer reasonably well from simulation to reality, suggesting that the model is able to learn domain-invariant features for these more common or visually simpler relationships. However, a significant drop in performance is observed for lane-specific relationships such as *on right lane of* and *on left lane of*. In most cases, the PhrDet score is near zero, regardless of K. This indicates that either (1) the latent space representations required to detect such fine-grained spatial relations are not easily transferable between simulation and real domains, or (2) the amount of annotated training data for these classes is insufficient to allow the model to generalize effectively.

Table 9: Per-class recall values for SGDet and PhrDet across different K values on real data using the FED model

Class	K	PhrDet	SGDet
on	20	41.8	7.3
	50	44.8	7.3
	100	51.3	7.3
attached to	20	5.2	2.9
	50	5.2	2.9
	100	8.5	2.9
on right side of	20	42.7	10.8
	50	49.5	10.8
	100	50.1	10.8
parking on	20	4.1	0.0
	50	22.8	0.0
	100	56.5	0.0
on left side of	20	71.7	24.7
	50	74.0	28.2
	100	74.0	28.2
on right lane of	20	0.4	0.0
	50	26.0	6.6
	100	26.0	6.6
on middle lane of	20	0.0	0.0
	50	0.0	0.0
	100	0.0	0.0
on left lane of	20	0.0	0.0
	50	2.6	0.0
	100	2.6	0.0

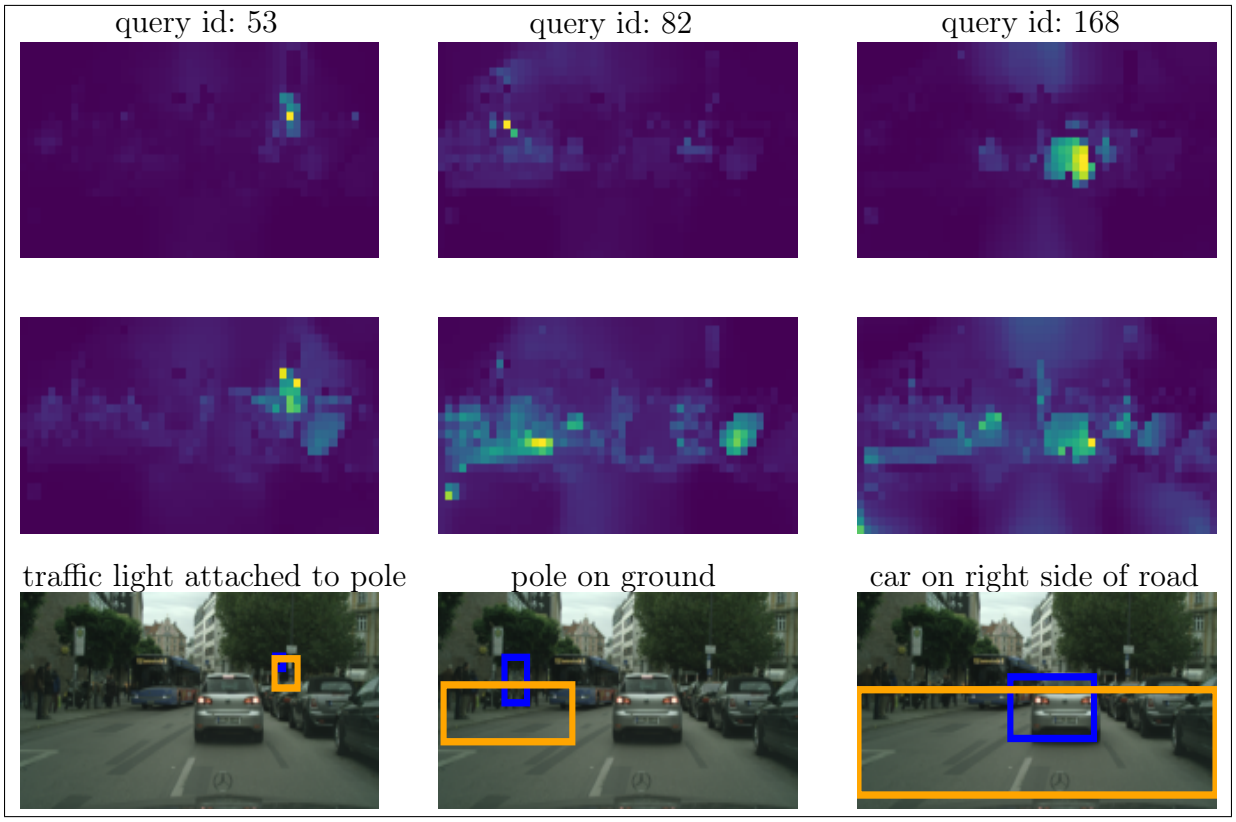


Figure 27: Heatmap results on a CityScapes data set image using the FED model. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

6.1.1 Post-Processing

In addition to the standard RelTR inference pipeline, a post-processing step was developed to refine bounding box predictions. This step operates on the outputs of the prediction networks within the entity detection module $Class_{entity}$ and $BBox_{entity}$. For each predicted bounding box produced by the relationship detection module $BBox_{sub/obj}$, the algorithm checks whether there exists a corresponding bounding box from the entity detection module with an IoU threshold of at least 0.5 and the same class identifier. If such a match is found, the bounding box from the entity detection module replaces the original prediction.

This refinement procedure was applied to all visual inference outputs of RelTR presented in this thesis. However, it is important to note that this post-processing step does not influence any reported metric results. Since the evaluation protocol already counts predictions with an $\tau \geq 0.5$ as correct, replacing bounding boxes under these conditions neither improves nor degrades quantitative performance.

Algorithm 5 Bounding box refinement post-processing for RelTR

Predicted bounding boxes from relationship detection module $BBox_{sub/obj}$
Predicted bounding boxes from entity detection module $BBox_{entity}$
IoU threshold $\tau = 0.5$
for each $bbox_{rel} \in BBox_{sub/obj}$ **do**
    for each $bbox_{entity} \in BBox_{entity}$ **do**
        if IoU$(bbox_{rel}, bbox_{entity}) \geq \tau$ **and** class$(bbox_{rel})$ = class$(bbox_{entity})$ **then**
            Replace $bbox_{rel}$ with $bbox_{entity}$
            break
        end if
    end for
end for
return Updated $BBox_{sub/obj}$

6.1.2 Entity Layers Enabeld

This configuration builds upon the setup described in Section 6.1. The model is initialized with weights pre-trained on a real-world data set. These weights are loaded into the full architecture, and, in addition to parameters not covered by the checkpoint, the decoder modules responsible for entity detection are set as trainable. This configuration is referred to as Entity Layers Enabled (ELE), whereas the encoder remains frozen. An overview of the resulting parameter configuration is provided in Table 10, while the training setup and hyperparameters are identical to those in Table 3.

Prior to evaluating on real-world data, the model is first assessed on the synthetic data set and compared to both the baseline and the configuration from Section 6.1. As shown in Table 11, this approach leads to a performance decline across all K-values in the SGDet evaluation task. The ELE variant underperforms compared to both the FED and baseline setups. But outperformes FED on the PhrDet evaluation task. This could suggest that the ELE setup resulted in a model outputting more, but potentially less accurate predictions.

This result suggests that changing weights responsible for extracting fine-grained visual features from the encoder's output (see Section 2.2) enables to extract potentially more relevant visual features but decreases accuracy on simulated data input.

Further insight is gained through a per-class evaluation shown in Table 12. While ELE performs comparably or even better in certain relationships—such as *on and on right side of*—some classes such as *on middle lane of*, *is hitting*, and *turning right* on show no improvement, with recall remaining at zero. These failure cases likely require deeper semantic understanding or low-level spatial features not provided by the frozen encoder.

Interestingly, ELE substantially outperforms the FED setup *on same road line as*, but underperforms on *on right side of*. This suggests a trade-off effect caused by the difference

Table 10: Overview of frozen and reinitialized modules in the sequential freeze strategy, with enabled entity detection layer. Pretrained indicates that those weights are pre-trained on a real-world data set and are then further changed during training on simulated data

Module	Status
Feature Extraction	
CNN Backbone	**Frozen**
Transformer Encoder	
Encoder	**Frozen**
Transformer Decoder	
Entity Layer	**Pretrained**
Coupled Self-Attention	**Initialized**
Decoupled Visual Attention	**Initialized**
Decoupled Entity Attention	**Initialized**
Prediction Heads	
Entity Class	**Pretrained**
Entity Bounding Boxes	**Pretrained**
Subj/Obj Class	**Initialized**
Subj/Obj Bounding Boxes	**Initialized**

Table 11: Comparison of mean recall at K for PhrDet and SGDet across Baseline, FED, and ELE settings

Task	K	Baseline	FED	ELE
PhrDet	20	63	45	51
	50	68	48	53
	100	69	48	53
SGDet	20	40	42	40
	50	41	43	40
	100	41	43	40

in distribution between simulated and real data—improvement in some relation classes may come at the cost of others.

Figure 28 shows real-world inference. While both FED and ELE approaches yield correct predictions, ELE produces more detections overall—some correct, such as identifying the lane the vehicle is driving on, and some erroneous, such as hallucinated poles in the distance.

Table 12: Per-class recall for SGDet and PhrDet tasks across different model variants and K values on simulated data.

Class	K	SGDet			PhrDet		
		Baseline	FED	ELE	Baseline	FED	ELE
on	20	15.0	44.0	50.6	51.9	86.9	85.3
	50	17.2	45.8	51.9	61.7	88.9	89.9
	100	17.7	46.2	52.2	66.4	89.4	90.9
on right side of	20	87.8	90.2	96.0	98.1	97.3	96.4
	50	87.9	90.3	96.1	99.7	97.4	96.7
	100	87.9	90.3	96.1	99.8	97.4	96.7
on middle lane of	20	98.9	0.0	0.0	100	0.0	0.0
	50	98.9	0.0	0.0	100	0.0	0.0
	100	98.9	0.0	0.0	100	0.0	0.0
turning right on	20	0.0	0.0	0.0	0.0	0.0	0.0
	50	0.0	0.0	0.0	0.0	0.0	0.0
	100	0.0	0.0	0.0	0.0	0.0	0.0
turning left on	20	16.6	90.0	90.0	33.9	94.5	100
	50	16.6	98.8	90.0	33.9	99.5	100
	100	16.6	98.8	90.0	33.9	99.5	100
is hitting	20	2.6	0.0	0.0	100	0.0	0.0
	50	2.6	0.0	0.0	100	0.0	0.0
	100	2.6	0.0	0.0	100	0.0	0.0
same road line as	20	16.1	18.1	1.8	21.2	0.0	50.2
	50	16.1	20.5	2.5	46.3	21.2	62.6
	100	16.1	20.5	2.5	47.0	21.2	62.6

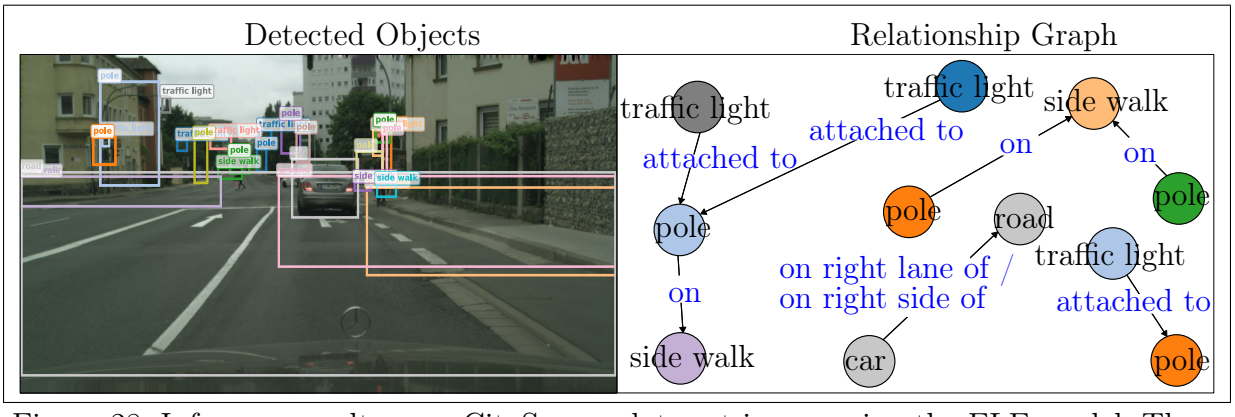


Figure 28: Inference results on a CityScapes data set image using the ELE model. The relationship graph was cleaned by hand for better visuals. Bounding boxes where not altered to display the amount of objects that where predicted to have relationships

Differences in attention behavior are visualized in Figure 29 and Figure 27. Compared to the FED configuration, attention maps in ELE appear more scattered and less concentrated. For example, query ID 168 in both approaches leads to the same prediction (*car on right side of road*), but the attention in ELE includes irrelevant regions such as the sky, while the FED model focuses more clearly on the street surroundings. Similarly, query ID 86 in ELE reveals inaccurate attention around the sidewalk, leading to misplaced bounding boxes.

Both models exhibit hallucinations of distant objects, e.g., poles or traffic lights that are not actually present. These are visible in query IDs 53 (FED) and 134 (ELE), both incorrectly predicting a traffic light in the same image region.

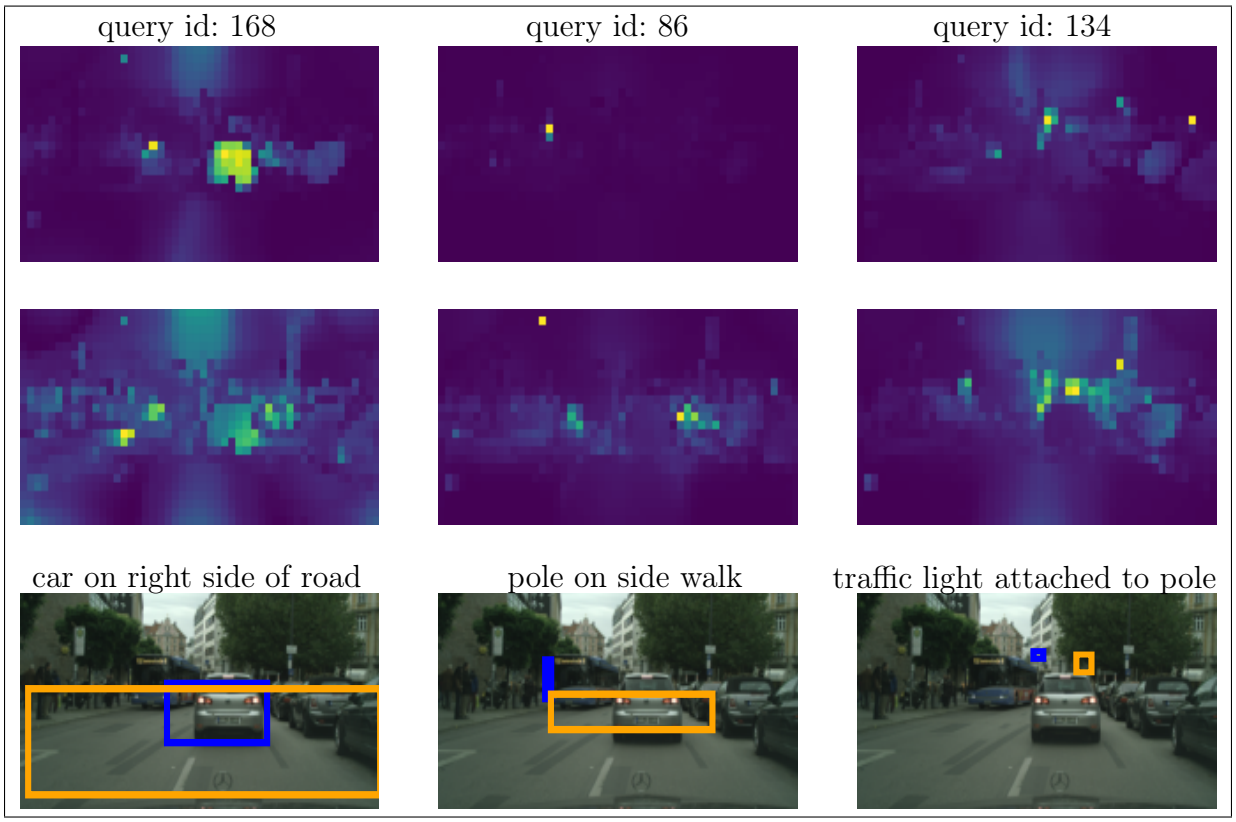


Figure 29: Heatmap results on a CityScapes data set image using the ELE model. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

Table 13 presents the per-class mean recall at K (mR@K) values for the SGDet and PhrDet metrics across various K values (20, 50, 100) on the real-world data set. The results compare the performance of the ELE model, in which entity layers were enabled for training, against the FED model with frozen encoder and detection layers.

The comparison reveals that enabling the entity layers led to improved recall in certain classes, while performance degraded in others. Notably, the ELE model achieves substan-

Table 13: Per-class recall values for SGDet and PhrDet across different K values on real data.

Class	K	PhrDet		SGDet	
		ELE	FED	ELE	FED
on	20	37.2	41.8	8.9	7.3
	50	46.5	44.8	11.6	7.3
	100	50.3	51.3	11.8	7.3
attached to	20	5.1	5.2	0.0	2.9
	50	13.8	13.7	2.9	2.9
	100	13.8	13.7	2.9	2.9
on right side of	20	27.3	42.7	9.2	10.8
	50	37.8	49.5	15.9	10.8
	100	38.1	50.1	16.1	10.8
parking on	20	38.5	4.1	2.3	0.0
	50	38.5	22.8	2.3	0.0
	100	38.5	56.5	2.3	0.0
on left side of	20	68.9	71.7	8.2	24.7
	50	83.9	74.0	19.7	28.2
	100	83.8	74.0	19.7	28.2
on right lane of	20	2.3	0.4	2.0	0.0
	50	57.6	26.0	6.3	6.6
	100	76.0	26.0	6.3	6.6
on middle lane of	20	0.0	0.0	0.0	0.0
	50	0.0	0.0	0.0	0.0
	100	0.0	0.0	0.0	0.0
on left lane of	20	0.0	0.0	0.0	0.0
	50	0.0	2.6	0.0	0.0
	100	0.0	2.6	0.0	0.0

tially higher PhrDet recall for the class *on right lane of* and shows a marked improvement in SGDet recall for the class *on right side of*. Similarly, the class *on left side of* benefits from enabling entity layers, with consistently higher PhrDet recall values across all K values.

These mixed results suggest that while enabling training for the entity layers can enhance performance for specific relationships, but is detection more false positives as shown in Figure 28. This observation underscores the challenge posed by the domain gap between synthetic and real-world data. It highlights the need for further investigation into which components of the model architecture should be adapted or fine-tuned during transfer

learning. A more selective or adaptive training strategy may be required to achieve consistent performance gains across all relationship classes.

6.2 Data set Switch Strategy

The Dataset Switch (DS) setting explores a novel training paradigm in which the model is trained jointly on both real and simulated data. To mitigate the domain gap and prevent catastrophic forgetting, adaptation was restricted such that only the entity detection part of the model was updated when processing real data. This was achieved by setting the relationship detection loss to zero during real-data batches, effectively freezing that component of the model through zero gradients.

This mechanism required modifying the data loader to provide placeholder relation annotations for real samples. During training, the loss function detected whether the input originated from the real domain and replaced the placeholder labels with the model's own predictions. This substitution caused the relation prediction loss to evaluate to zero, ensuring that no gradients were propagated for the relation decoder.

A challenge arose in the classification loss of subject and object bounding boxes. Since this component relied on cross-entropy loss (see Section 2.3.5), aligning predicted and target labels was insufficient to produce zero loss since it would only occur if the model predicted the correct class with 100% confidence. To circumvent this, the corresponding loss terms were explicitly hardcoded to zero during real-data batches.

This manual loss manipulation introduced conflicts with PyTorch's distributed training. Specifically, parameters that did not receive gradients were flagged as unused, which caused failures in the backward pass. This was resolved by enabling the `find_unused_parameters=True` flag, which allowed the training to proceed at the cost of increased computational overhead.

A critical implementation detail was ensuring synchronized data set switching across all GPU ranks. Without consistent domain alignment between GPUs, `torch.optim` would hang due to inconsistent parameter updates. To avoid this, training alternated strictly by one epoch on real data followed by one epoch on simulated data. Attempts to switch domains at every training step instead of per epoch led to diminished performance on the simulated test set, suggesting that excessive alternation may hinder stable learning.

In terms of performance (Table 14), the DS training strategy performs comparably to the baseline on the PhrDet task, indicating effective retention of relational knowledge. However, it underperform on the SGDet task, highlighting weaknesses in accurately predicting the spatial configuration of objects when training is decoupled in this manner.

Table 14: Comparison of mean recall at K for PhrDet and SGDet across different training strategies on simulated data.

Task	K	Baseline	FED	ELE	DS
PhrDet	20	63	45	42	45
	50	68	48	44	46
	100	69	48	44	47
SGDet	20	40	42	28	38
	50	41	43	29	38
	100	41	43	29	38

Table 15: Per-class recall for SGDet and PhrDet tasks across different model variants and K values on simulated data.

Class	K	SGDet			PhrDet		
		Baseline	FED	DS	Baseline	FED	DS
on	20	15.0	44.0	11.5	51.9	86.9	63.8
	50	17.2	45.8	12.7	61.7	88.9	68.4
	100	17.7	46.2	12.8	66.4	89.4	70.8
on right side of	20	87.8	90.2	87.9	98.1	97.3	96.4
	50	87.9	90.3	87.9	99.7	97.4	96.4
	100	87.9	90.3	87.9	99.8	97.4	96.4
on middle lane of	20	98.9	0.0	97.7	100	0.0	100
	50	98.9	0.0	97.7	100	0.0	100
	100	98.9	0.0	97.7	100	0.0	100
turning right on	20	0.0	0.0	0.0	0.0	0.0	0.0
	50	0.0	0.0	0.0	0.0	0.0	0.0
	100	0.0	0.0	0.0	0.0	0.0	0.0
turning left on	20	16.6	90.0	81.5	33.9	94.5	81.5
	50	16.6	98.8	81.5	33.9	99.5	81.5
	100	16.6	98.8	81.5	33.9	99.5	81.5
is hitting	20	2.6	0.0	0.0	100	0.0	0.0
	50	2.6	0.0	0.0	100	0.0	0.0
	100	2.6	0.0	0.0	100	0.0	0.0
same road line as	20	16.1	18.1	0.0	21.2	0.0	0.0
	50	16.1	20.5	0.0	46.3	21.2	0.0
	100	16.1	20.5	0.0	47.0	21.2	0.0

Per-class results (Table 15) reveal further nuances. The DS approach shows a consistent decline in performance across most classes compared to other strategies and even fails to predict certain classes such as *same road line as*. Interestingly, however, it recovers

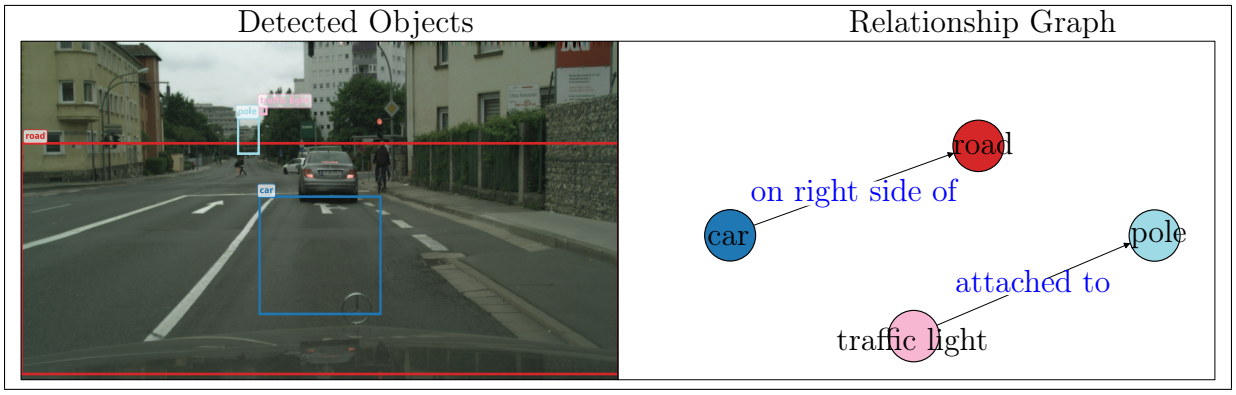


Figure 30: Inference results on a CityScapes data set image using the DS model. The relationship graph was cleaned by hand for better visuals.

perfect recall for *on middle lane of*, a class where the baseline already performed well. This suggests that the frozen encoder may have never encoded sufficient information for predicting that relationship, and once learning was re-enabled, the model was able to acquire it from scratch.

Turning to qualitative results, inference on real data reveals critical limitations. As shown in Figure 30, the DS model generates very few predictions—none of which are correct. Notably, to even elicit these outputs, the confidence threshold had to be lowered from 70% to 30%. Moreover, the predicted bounding box for the car appears in the same location across all input images, implying that the entity detection module failed to generalize.

An inspection of intermediate checkpoints revealed a puzzling trend: the model initially improved over time, but then suddenly degraded in performance, stagnating at the failure mode shown in Figure 30. This suggests a potential collapse or instability during optimization.

Attention map visualizations (Figure 31) further confirm this hypothesis. The attention is diffusely distributed with no clear focus on objects of interest. In fact, the brightest areas of attention tend to cluster around the sky, indicating that the model is attending to irrelevant regions and failing to encode meaningful object-centric information.

The adaptation restrictions imposed by the DS approach appear to have caused the model to operate in two distinct modes, depending on whether the input originates from the simulated or real domain. This phenomenon is illustrated in Figure 32, which shows attention maps from the FED model during inference on both real and simulated data. The heatmaps reveal no significant differences between the two domains: the subject branch exhibits clearly focused attention, while the object branch attends more broadly—particularly for classes such as road and sidewalk, where a wider contextual understanding is necessary to correctly infer relationships.

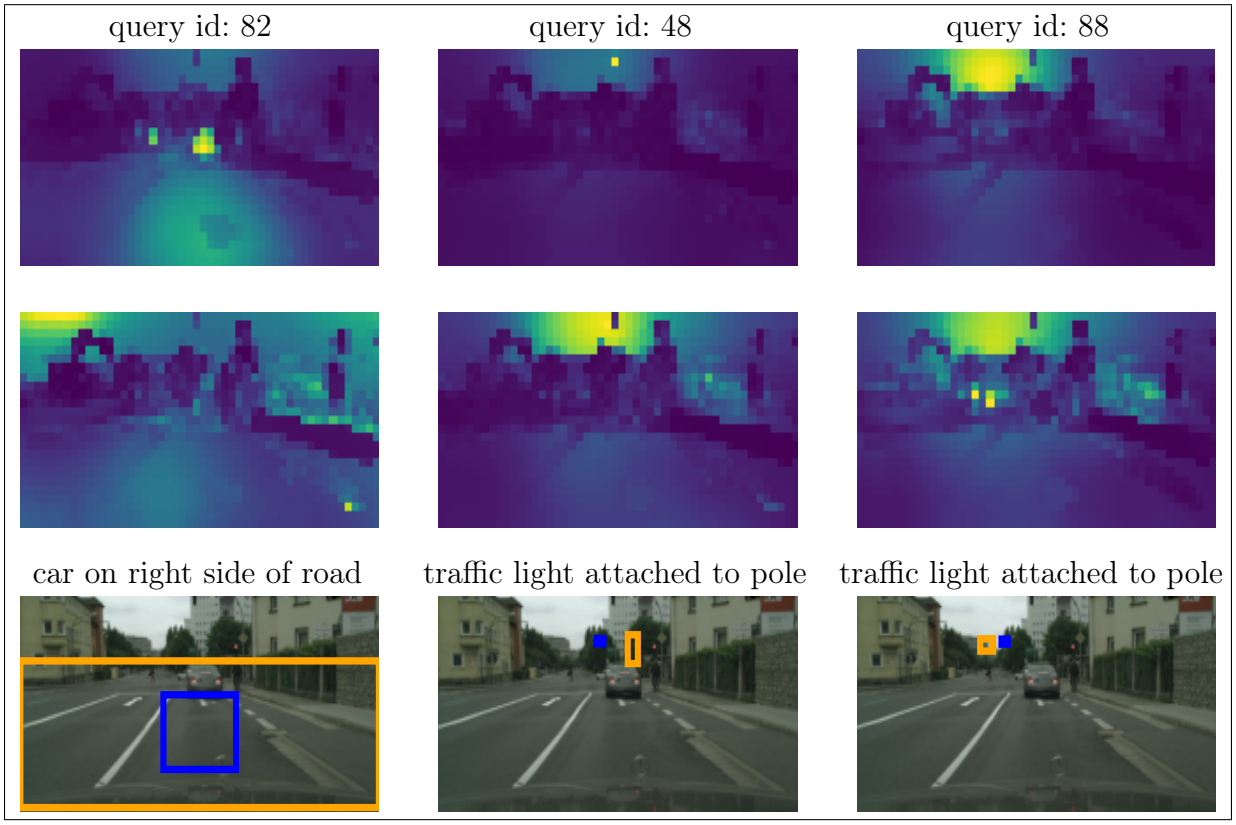


Figure 31: Heatmap results on a CityScapes data set image using the DS model. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

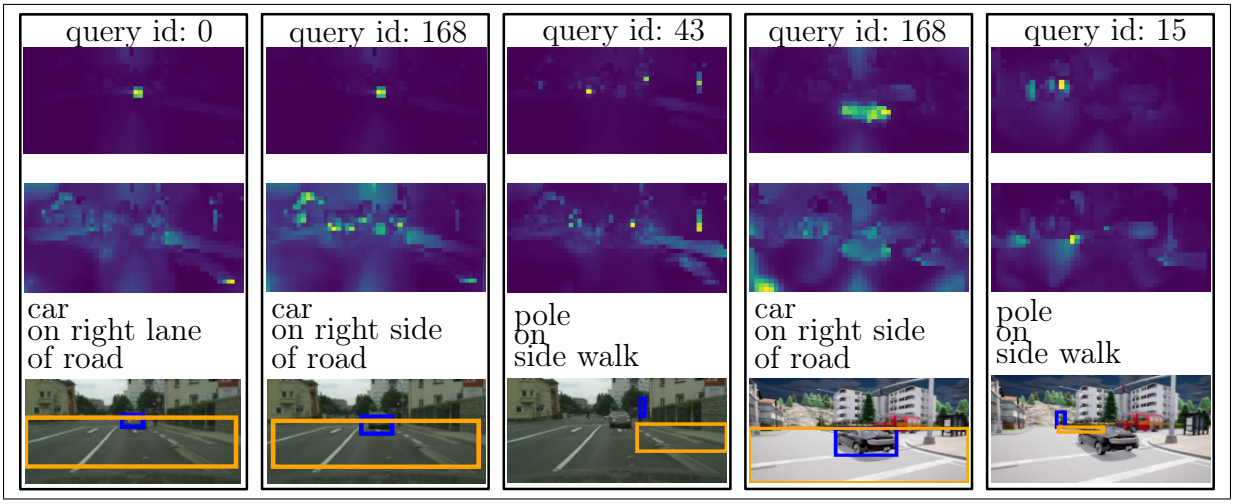


Figure 32: Attention maps on real and simulated data of the FED model. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

In contrast, Figure 33 presents attention maps from the DS model under the same conditions. Here, a distinct difference in attention is observed depending on the domain. On simulated data, the heatmaps resemble those of the FED model, albeit with reduced focus in the subject branch and an even broader spread in the object branch. However, for real data, the attention becomes incoherent, showing no meaningful or consistent pattern. In many cases, the attention is diffused across irrelevant regions such as the sky, indicating a breakdown in the model's ability to generalize.

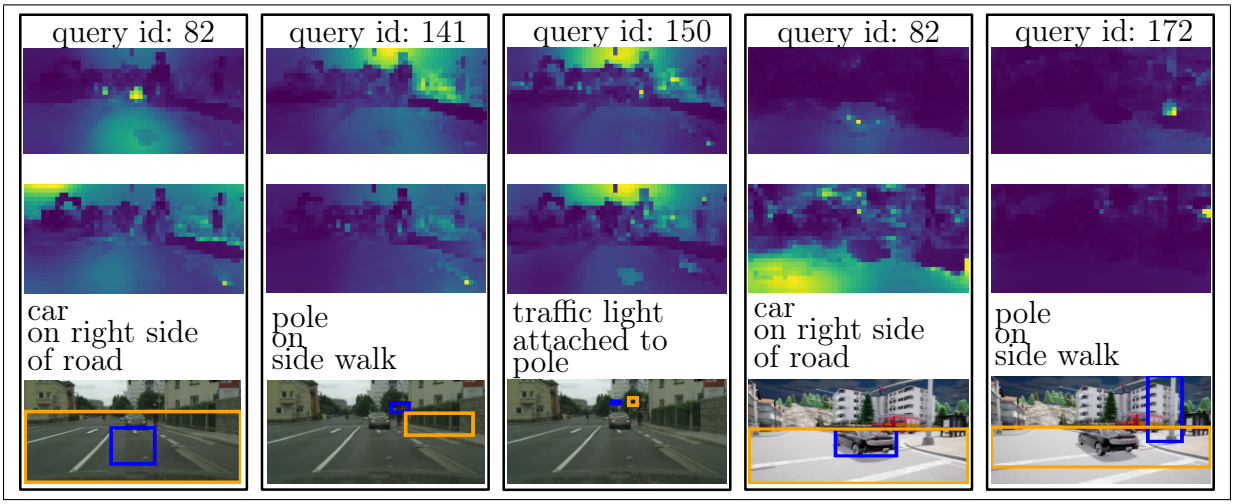


Figure 33: Attention maps on real and simulated data of the DS model. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

Figure 34 compares the relationship detection outputs of both models. While the DS model performs relationship inference on simulated data, the predicted bounding boxes are noticeably less precise than those of the FED model, pointing to a degradation in spatial grounding. On the other hand, Figure 35 compares the entity detection outputs of both models. Although the DS model produces less accurate and occasionally misaligned bounding boxes compared to the FED model, the results show that this component remains functional to some extent across both domains.

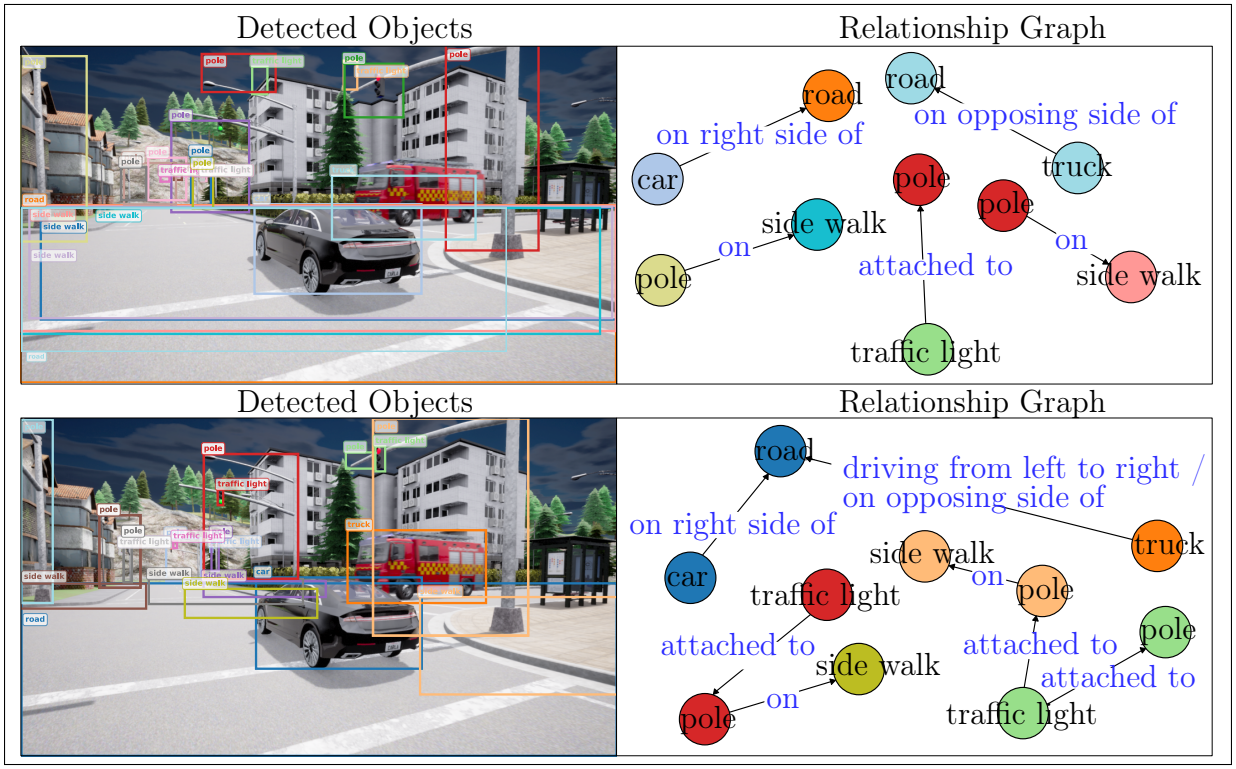


Figure 34: Relationship detection on simulated data using the DS model (top) and the FED model (bottom). The relationship graph was cleaned by hand for better visuals. Bounding boxes where not altered to display the amount of objects that where predicted to have relationships

These observations support the hypothesis that the adaptation restrictions in the DS model led to partial domain specialization: the model appears capable of performing entity detection on both simulated and real data, but is only able to infer relationships in the simulated domain. This domain-specific behavior mirrors the restricted training conditions and highlights the challenge of achieving robust generalization under such constraints.

Overall, the DS strategy illustrates the potential of modular domain adaptation but also its pitfalls. While it retains some relational reasoning capability, it struggles with consistent object localization and exhibits optimization instability. Addressing these challenges requires refining the loss masking strategy, stabilizing training across domains,

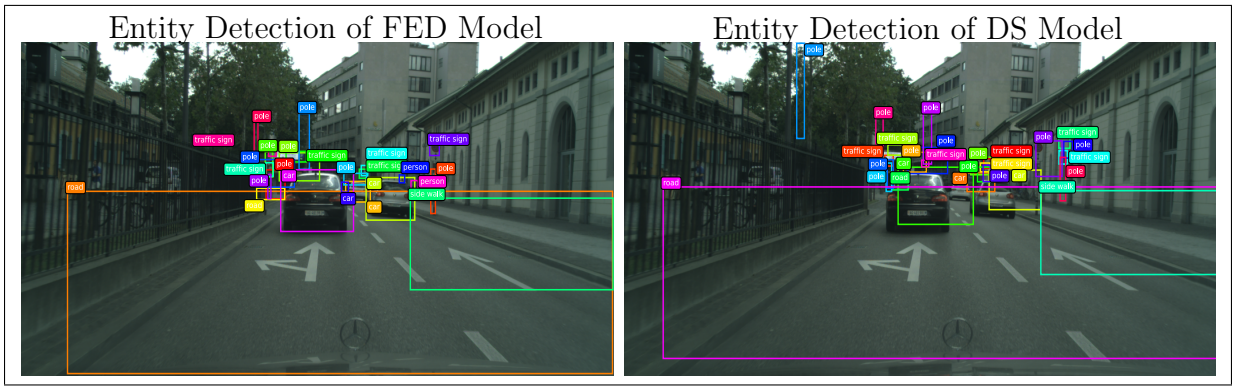


Figure 35: Entity detection on real data using the FED model (left) and the FED model (right). Labels where replaced by hand for better visuals.

and improving robustness to per-class performance drops. The following section will describe one attempt of solving this.

6.2.1 Semi-Supervised Variant

This configuration builds upon the previously introduced setup and incorporates a strategy to refine the loss computation, as described in Section 6.2. Rather than setting the loss to zero for samples originating from the real domain, pseudo-labels are employed to compute a meaningful loss. However, since these pseudo-labels are inherently noisy, the loss contribution is downscaled by a factor of $\alpha = 0.4$.

In this thesis, pseudo-labels are used to distill knowledge from one model into another. Although [KRAD24] notes that this process is not traditionally classified as pseudo-labeling, the term will be used in this context. The model providing the pseudo-labels is the one trained using the FED strategy (Section 6.1), as the ELE strategy tends to produce a higher number of false positives (see Figure 28).

The selection of pseudo-labels is based on confidence-thresholding [KRAD24], with a threshold set to $\tau = 0.85$. It was hypothesized that the model trained under this semi-supervised setting would perform comparably to the FED model, given that its training signal was derived from it. However, the results shown in Figure 36 deviate from this expectation. While the failure mode of the DS strategy (as detailed in Section 6.2) is effectively mitigated, the performance still falls significantly short of the FED baseline (see Figure 26).

Further insights are provided by the attention maps in Figure 37, which reveal even more distorted attention in the DVA module than observed in Figure 31. These findings suggest that the most effective strategy for bridging the domain gap remains the two-stage training approach—first pre-training the entity detection module of RelTR, followed by training the

Table 16: Per-class recall values for SGDet and PhrDet across different K values on real data.

Class	K	PhrDet		SGDet	
		DS	FED	DS	FED
on	20	11.8	41.8	0.0	7.3
	50	17.7	44.8	0.0	7.3
	100	18.7	51.3	0.0	7.3
attached to	20	0.0	5.2	0.0	2.9
	50	0.0	13.7	0.0	2.9
	100	0.0	13.7	0.0	2.9
on right side of	20	90.7	42.7	0.0	10.8
	50	90.7	49.5	0.0	10.8
	100	90.7	50.1	0.0	10.8
parking on	20	0.0	4.1	0.0	0.0
	50	0.0	22.8	0.0	0.0
	100	0.0	56.5	0.0	0.0
on left side of	20	11.5	71.7	0.0	24.7
	50	11.5	74.0	0.0	28.2
	100	11.5	74.0	0.0	28.2
on right lane of	20	0.0	0.4	0.0	0.0
	50	0.0	26.0	0.0	6.6
	100	0.0	26.0	0.0	6.6
on middle lane of	20	0.0	0.0	0.0	0.0
	50	0.0	0.0	0.0	0.0
	100	0.0	0.0	0.0	0.0
on left lane of	20	0.0	0.0	0.0	0.0
	50	0.0	2.6	0.0	0.0
	100	0.0	2.6	0.0	0.0

relationship detection module. Consequently, further exploration of the semi-supervised variant was deemed unnecessary.

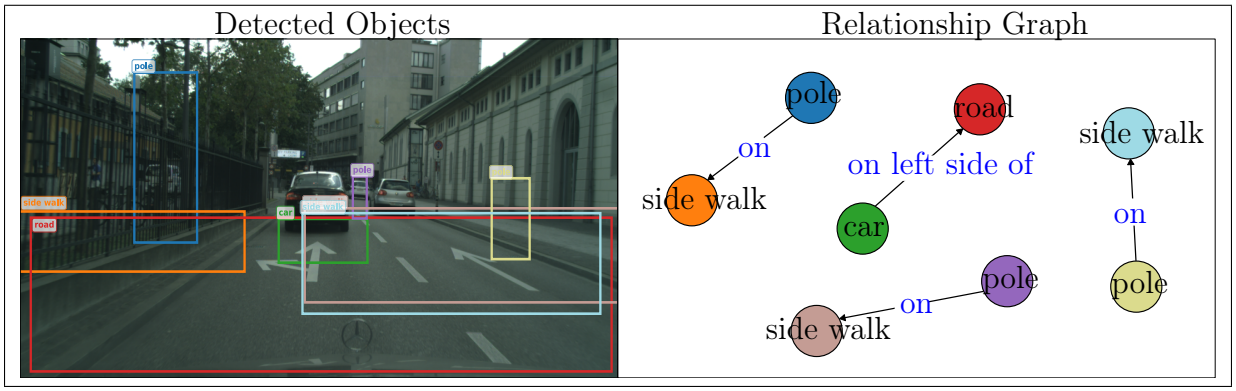


Figure 36: Inference results on a CityScapes data set image using the semi-supervised Variant of the DS model. The relationship graph was cleaned by hand for better visuals.

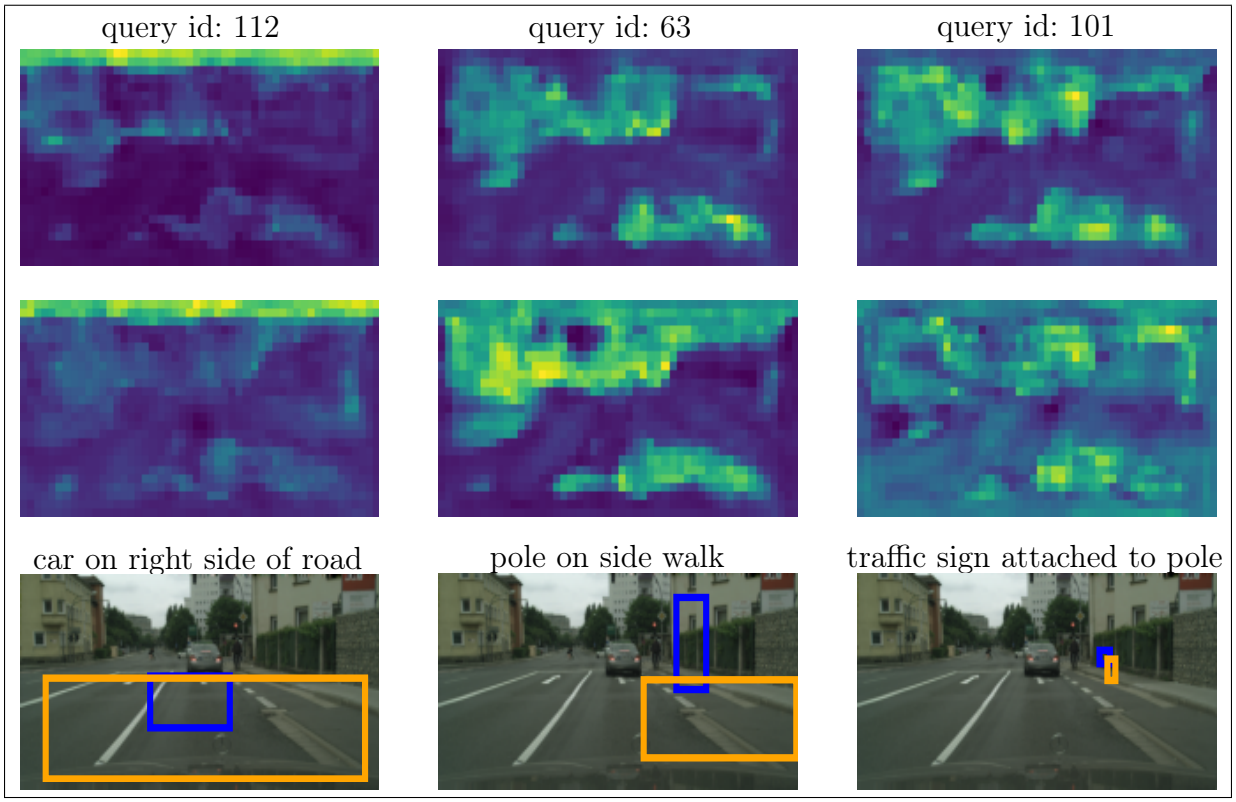


Figure 37: Heatmap results on a CityScapes data set image using the semi-supervised variant of the DS model. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

7 Ablation Study

Following the evaluation on real-world data, the following ablation study investigates the impact of domain gaps within the FED training approach. Two options can be considered to reduce this domain gap: (1) increasing the realism of the simulator used to generate relationship annotations, or (2) selecting a goal domain that is visually more similar to the simulator.

The first option is currently not feasible. While CARLA has been updated to Unreal Engine 5, which significantly improves visual realism, the annotation tool used in this thesis—based on ScenarioRunner [CAR23]—does not, to the date of this thesis, support this newer version of CARLA. As a result, the second approach is pursued. For this, a goal domain with greater visual similarity to CARLA is selected. The GTA data set [RVRK16a], which, like CARLA, is based on a game engine, fulfills this criterion. Figure 38 illustrates the optical similarity between the two data sets.

Figure 38: Visual similarity across synthetic domains: Example images from the GTA (left) and CARLA (right) data sets

As demonstrated in Section 6, the most effective configuration is the FED approach, where the entity detection module of RelTR is first pre-trained on a mixture of the goal and start domains with object annotations, followed by training the relationship detection module on start domain data annotated with relationships. The same approach is adopted for the current ablation study.

The influence of the domain gap becomes immediately apparent when comparing entity detection models trained exclusively on different goal domains. Figure 39 presents results from models trained only on GTA images and only on real images, both evaluated on CARLA input. For the GTA-trained model, incorrect predictions could be almost completely eliminated by applying a confidence threshold of $\tau_{\text{GTA}} = 0.7$. In contrast, the real-trained model continued to produce a large number of false positives even at a threshold of $\tau_{\text{Real}} = 0.9$. This behavior, previously observed in Section 6.1.2, can now be attributed to the larger domain gap between real-world images and the CARLA simulator.

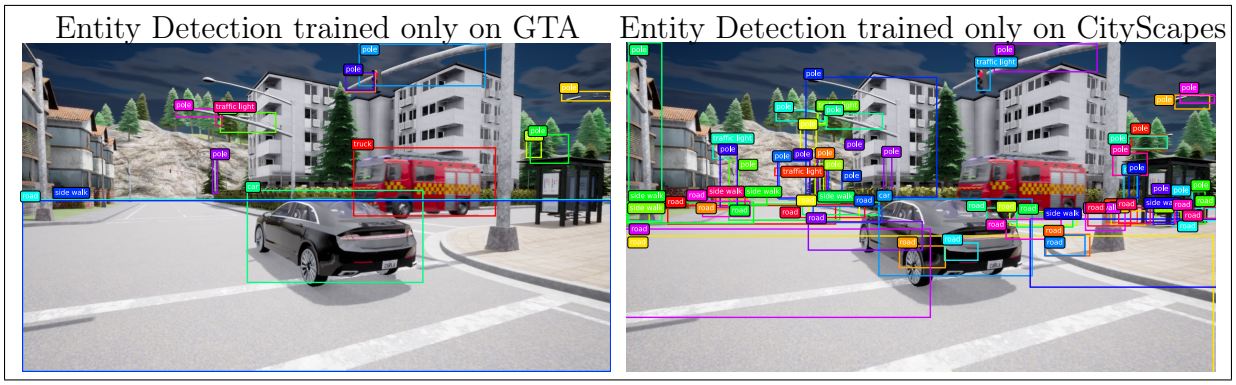


Figure 39: Entity detection performance on CARLA images. Model trained only on GTA images (left) and model trained only on real images (right)

Table 17: Comparison of mean Recall at K for PhrDet and SGDet between the baseline and FED models.

Task	K	Baseline	FED Real	FED GTA
PhrDet	20	63	45	58
	50	68	48	63
	100	69	48	64
SGDet	20	40	42	46
	50	41	43	48
	100	41	43	49

Table 17 compares mean recall at K (mR@K) for PhrDet and SGDet between baseline and FED models trained with different goal domains. Two key findings emerge:

1. Training on a visually closer goal domain (GTA) improves object localization, as evidenced by higher SGDet recall scores.

2. The FED model trained on GTA also outperforms the version trained on real data in the PhrDet task and approaches the performance of the baseline trained only on CARLA.

Table 18 provides a per-class analysis on simulated data. The FED model trained on GTA consistently performs similarly to or better than the baseline across multiple classes, including *on*, *on right side of*, and *same road line as*. Notably, the class *on middle lane of*, previously not predicted by the FED model trained on real data, is now accurately recognized. This suggests that prior assumptions about missing encoded information can instead be explained by the larger domain gap.

However, the GTA-trained model does underperform the real-trained model on certain classes (e.g., *turning left on*). This may be attributed to overfitting in the real-trained model due to annotation imbalance. On the other hand, the GTA-trained model successfully

Table 18: Per-class recall values for SGDet and PhrDet across different K values on simulated data using the FED model trained on real and GTA data.

Class	K	SGDet			PhrDet		
		Baseline	Real	GTA	Baseline	Real	GTA
on	20	15.0	44.0	38.8	51.9	86.9	74.7
	50	17.2	45.8	45.2	61.7	88.9	79.9
	100	17.7	46.2	49.0	66.4	89.4	83.6
on right side of	20	87.8	90.2	95.9	98.1	97.3	96.1
	50	87.9	90.3	96.0	99.7	97.4	96.4
	100	87.9	90.3	96.0	99.8	97.4	96.4
on middle lane of	20	98.9	0.0	99.8	100	0.0	100
	50	98.9	0.0	99.8	100	0.0	100
	100	98.9	0.0	99.8	100	0.0	100
turning right on	20	0.0	0.0	0.0	0.0	0.0	0.0
	50	0.0	0.0	0.0	0.0	0.0	0.2
	100	0.0	0.0	0.2	0.0	0.0	0.4
turning left on	20	16.6	98.8	60.2	33.9	94.5	60.2
	50	16.6	98.8	60.2	33.9	94.5	60.2
	100	16.6	98.8	60.2	33.9	94.5	60.2
is hitting	20	2.6	0.0	0.0	100	0.0	0.0
	50	2.6	0.0	0.0	100	0.0	0.0
	100	2.6	0.0	0.0	100	0.0	0.0
same road line as	20	16.1	18.1	52.0	21.2	0.0	67.2
	50	16.1	20.5	54.2	46.3	21.2	85.0
	100	16.1	20.5	54.2	47.0	21.2	91.3

predicts classes that the baseline could not (*turning right on*) and fails on others like *is hitting*, possibly due to their rarity (see Figure 22).

Figure 40 and Figure 41 provide visualizations of the final relationship inference and attention maps for the DVA module. Compared to the FED model trained on real data (Figure 27), attention is more clearly focused on relevant regions, highlighting the improved bridging capability when training with a visually similar domain.

Finally, Table 19 compares the performance of the models in their respective goal domains using a small, manually annotated evaluation data set. The model trained on GTA outperforms the model trained on real data in three of the four most frequent relationship classes: *on*, *attached to*, and *on right side of*. *On opposing side of* is also better predicted by the GTA-trained model. Underperformance in some other classes can be explained by class imbalance (see Figure 22). Importantly, these results show a clearer correlation

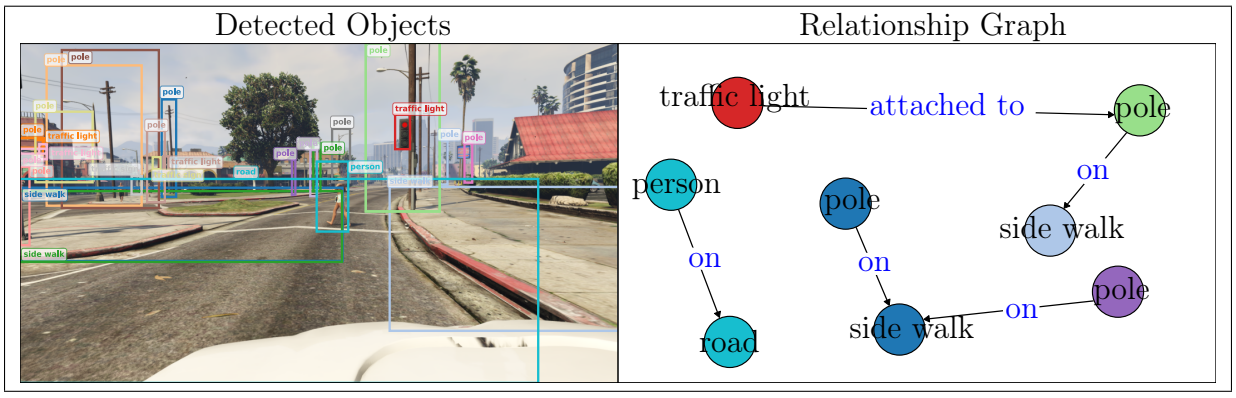


Figure 40: Final relationship predictions from the FED model trained on GTA data. The Relationship Graph was cleaned by hand for better visuals. Bounding Boxes where not altered to display the amount of objects that where predicted to have relationships

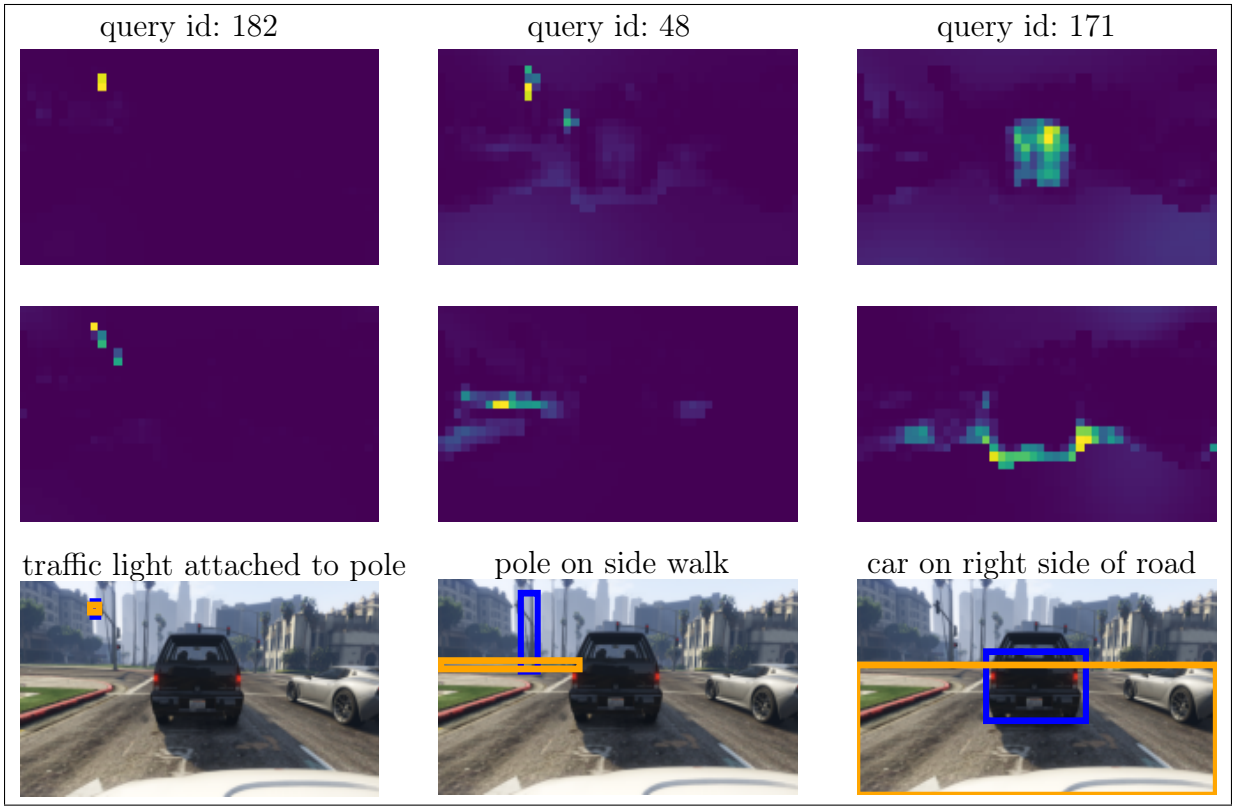


Figure 41: Attention maps of the DVA module for the FED model trained on GTA data. Query ID refers to the index of the query in the transformer decoder output. Heatmaps correspond to the attention maps produced by the DVA Module (Equation 2.17), where the attention map on the top is the subject and the lower one is the object branch.

Table 19: Per-class recall values for SGDet and PhrDet across different K values of the FED models trained on real and GTA data

Class	K	PhrDet		SGDet	
		FED GTA	FED Real	FED GTA	FED Real
on	20	73.0	41.8	47.6	7.3
	50	82.4	44.8	51.1	7.3
	100	85.3	51.3	53.9	7.3
attached to	20	81.7	5.2	16.4	2.9
	50	82.9	13.7	17.0	2.9
	100	82.9	13.7	17.0	2.9
on right side of	20	66.8	42.7	44.4	10.8
	50	69.4	49.5	47.9	10.8
	100	69.4	50.1	48.1	10.8
on left side of	20	29.1	71.7	17.7	24.7
	50	37.1	74.0	23.0	28.2
	100	37.1	74.0	23.0	28.2
on right lane of	20	0.0	0.4	0.0	0.0
	50	3.1	26.0	0.0	6.6
	100	9.9	26.0	0.0	6.6
on middle lane of	20	0.0	0.0	0.0	0.0
	50	0.0	0.0	0.0	0.0
	100	0.0	0.0	0.0	0.0
on left lane of	20	0.0	0.0	0.0	0.0
	50	0.0	2.6	0.0	0.0
	100	0.0	2.6	0.0	0.0
on opposing side of	20	85.4	0.0	13.5	0.0
	50	85.4	25.4	19.8	0.0
	100	85.4	52.9	19.8	0.0

between annotation density and performance for GTA, supporting the hypothesis that domain similarity enables better generalization from increased data volume.

8 Discussion & Future Work

This section reflects on the findings of the conducted experiments, analyzing the observed results in the context of the research objectives. Strengths, limitations, and potential implications of the proposed approach are discussed, with particular emphasis on the factors influencing model performance in both simulated and real-world domains. Furthermore, avenues for future work are outlined, highlighting possible methodological improvements, additional experiments, and broader applications that could extend the contributions of this thesis.

8.1 Baseline

The baseline evaluation of RelTR on the simulated data set shows that the model can learn the relationships between stationary and dynamic objects effectively. This validates the quality of the data set's annotations. High recall in relationships such as *on right side of* and *on middle lane of* under both PhrDet and SGDet conditions suggests that the transformer-based architecture is well-suited to capturing structured relationships in the image. However, the consistently lower SGDet scores and limited performance gains with increasing K indicate that precise object localization remains a bottleneck, constraining the model's ability to meet stricter relational grounding criteria. This weakness is most pronounced in relationships that are even hard for humans to classify based of only an image-such as *turning right on*-which likely suffer from a combination of data set imbalance and inherent visual ambiguity. Interestingly, high recall in visually distinctive but infrequent relationships (e.g. *is hitting*) demonstrates that class frequency alone is not a limiting factor if the models train and goal domains are identical. Cross-attention map analysis further reveals a close alignment between model focus and semantically meaningful image regions for correct predictions, while wrong predictions often arise from overfitting to simulation-specific artifacts or insufficient viewpoint diversity, see the relationship prediction of *person riding bicycle* in Figure 25. Together, these findings confirm that the simulated data set provides a viable training ground for relationship predictions and establish a benchmark for evaluating the domain transfer strategies. At the same time, however, they underscore the necessity of a more diverse data set.

8.2 Frozen Entity Detection

The FED approach demonstrates the potential to leverage pre-trained entity representations learned from real-world scenes. By freezing the entity detection module initialized with real-data weights, the model benefits from robust and semantically meaningful object features,

which provide a stable foundation for relationship prediction in the simulated training domain. Although the domain gap restricts adaptation during relationship learning, visible by the lost capability of predicting the class *on middle lane of*, the approach validates the feasibility of transferring learned relational knowledge from simulation to reality. Inference results on real-world images show that the model can detect relevant relationships with semantically coherent predictions, especially for common relations such as *on* and *on right side of*. These findings indicate that key road-level spatial relationships can be represented in a domain-invariant manner, facilitating Sim2Real transfer despite visual domain gaps. Although localization accuracy on real-world test data remains a challenge, the PhrDet task scores suggest that the presence of relationships can be detected reliably when the overlap thresholds of the bounding box are not required to be fully met. The marked drop in detection for fine-grained lane-specific relationships highlights areas where domain discrepancies and insufficient simulated-world annotations limit transferability.

The ELE configuration of the FED approach enables the model to adjust fine-grained visual feature extraction relevant to entity localization and relationship detection beyond what the frozen encoder provides. A key effect of this increased flexibility is that the model generates more relationship predictions during inference that are above a high confidence threshold. Although this leads to improved detection rates for certain spatial relationships-such as *on right lane of* and *on left side of*-the model simultaneously produces a number of false positives. This behavior is clearly visible in the results of the real-world inference, where the ELE model identifies more correct relationships but also incorrectly predicts entities such as distant poles that are not present.

Although the quantitative results reported in this thesis appear modest, they are competitive compared to the original RelTR paper's performance metrics [CYR23]. However, the exhibits limitations regarding the scope of scene contexts it can effectively handle. Specifically, the model requires that the scene content during inference be reasonably represented within the training data. Scenes with complex or highly varied interactions-where many entities and relationships occur simultaneously-pose challenges for the model, resulting in decreased prediction accuracy. This indicated that the model has not yet learned to generalize robustly across diverse scene content, but rather specializes in the types of environment encountered during training. This limitation is understandable. Although the model demonstrates a promising ability to generalize across domains (from simulated to real-world data), achieving true generalization across varying scene configurations will require expanding the training data to encompass a broader range of interactions and scene complexities.

However, the overall results of the FED approach provide encouraging evidence that a simulation-trained RelTR model, when combined with a real-data pre-trained and frozen

entity detection module, can perform inference on real-world scenes with meaningful relational understanding. This outcome confirms the viability of the Sim2Real pipeline central to this thesis.

8.3 Data set Switch Strategy

The DS approach aimed to jointly train the model on simulated and real data by updating only the entity detection layers during real-data batches and updating the whole model during simulated-data batches. However, this strategy led to significant optimization instability and domain-specific failure modes. Qualitative results reveal that the DS model produces very few and mostly incorrect predictions on real data. Attention maps show highly diffuse and unfocused activations, with the model attending to irrelevant regions such as the sky, indicating a failure to learn meaningful object-centric representations for the real domain. This discrepancy in domain-specific attention patterns contradicts the more consistent patterns observed in the FED model.

Overall, the imposed training restrictions caused partial domain specialization where the model could detect entities across domains but only infer relationships reliably in simulation. This highlights the difficulty of stabilizing joint domain training with selective adaptation, leading to degraded localization and relationship prediction on real data. The DS strategy thus demonstrates the challenges of modular adaptation and suggests that more stable and nuanced training methods, such as FED, are necessary to achieve robust Sim2Real transfer.

8.4 Ablation Study

The ablation study reveals the significant impact that the domain gap has on model performance, an influence greater than initially anticipated. Compared to models pre-trained on real-world data, models pre-trained on domains visually closer to CARLA, such as GTA, demonstrate better prediction of relationships learned from CARLA data when evaluated on CARLA or GTA inputs. Metric scores correlate directly with the amount of annotated data available per class, giving confidence to conclude that with increasing annotated data-especially within visually similar domains-can improve the accuracy of the FED approach. These insights suggest that many performance limitations previously attributed to model capacity or semantic understanding are actually driven by the domain gap's size, characteristics, and annotation scarcity.

8.5 Future Work

The findings of this study highlight several promising directions for future research aimed at improving Sim2Real transfer in SGG. One key area is the expansion and diversification of the simulated data set, particularly focusing on increasing the representation of relationship classes with low annotation counts. A richer and more balanced data set will help reduce overfitting and better capture the complexity of real-world scenes. Encouragingly, this expansion is readily achievable thanks to the semi-automatic annotation toolchain developed as part of this work, which significantly lowers the barrier for efficient data set growth.

Alongside data expansion, advanced domain adaptation techniques, such as feature-level alignment, adversarial training, and style transfer methods, offer the potential to bridge the visual and semantic gap between simulated and real domains more effectively.

Lastly, the use of next-generation simulation platforms, such as the latest version of CARLA utilizing Unreal Engine 5's enhanced photorealism and complex scene dynamics, promises to provide higher-fidelity synthetic data. This increased realism could substantially narrow the domain gap, improving the transferability of learned models to real-world scenarios, and advancing the overall robustness of relationship detection across domains.

9 Conclusion

The results of this work demonstrate that the selected annotated relations between objects are generally learnable and predictable by the RelTR model. Among the investigated strategies for reducing the domain gap between real and simulated images, the FED approach proved to be the most effective overall. However, potential weaknesses were also revealed: certain relations, such as *on middle lane of*, could not be reliably predicted, suggesting that the frozen encoder may not encode features necessary for these relations.

To examine whether such information might be present in the entity detection layers, the ELE approach was implemented, allowing these layers to be trainable. While this resulted in a higher number of predictions, they included both correct and incorrect ones. Furthermore, an issue of model confidence emerged, likely caused by the domain gap. Reliable prediction of the previously problematic relations was still not achieved.

The third approach, the DS, which enabled full training of the model and sequential learning on either real or simulated data, did not improve performance. This confirmed the FED approach as the most effective among those tested.

The final ablation study provided critical insights: when the domain gap was smaller, the relations previously deemed unlearnable – including *on middle lane of* – could be reliably predicted. This disproves the assumption that the Frozen Entity Detection approach is inherently unsuitable for these relations and indicates that the primary cause lies in the magnitude of the domain gap. Results on simulated test data are either close to the baseline or exceed it in some cases. Performance correlates almost perfectly with the number of annotated samples per class, suggesting that increasing annotations could further enhance the accuracy of the approach.

A further major contribution of this work lies in efficient data set creation. In contrast to existing data sets with relationship annotations, neither crowdsourcing nor a lengthy manual annotation campaign was required. By leveraging the CARLA simulator, a suitable data set was created within two months by a single person. The developed semi-automatic annotation tool significantly reduced manual effort, making the data set far more scalable than comparable existing resources.

References

[ALZ+23] Ai, Ting ; Liu, Zhiliang ; Zhang, Jiyang ; Liu, Honghao ; Jin, Yaqiang ; Zuo, Mingjian: Fully Simulated-Data-Driven Transfer-Learning Method for Rolling-Bearing-Fault Diagnosis. In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), S. 1–11. *http://dx.doi.org/10.1109/TIM.2023.3301901*. – DOI 10.1109/TIM.2023.3301901

[AZH+21] Alzubaidi, Laith ; Zhang, Jinglan ; Humaidi, Amjad J. ; Al-Dujaili, Ayad ; Duan, Ye ; Al-Shamma, Omran ; Santamaría, J. ; Fadhel, Mohammed A. ; Al-Amidie, Muthana ; Farhan, Laith: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. In: *Journal of Big Data* 8 (2021), Nr. 1, 53. *http://dx.doi.org/10.1186/s40537-021-00444-8*. – DOI 10.1186/s40537–021–00444–8. – ISSN 2196–1115

[Bri89] Bridle, John: Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters. In: Touretzky, D. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 2, Morgan-Kaufmann, 1989

[CAR23] CARLA Simulator scenario_runner contributors: *ScenarioRunner for CARLA*. *https://github.com/carla-simulator/scenario_runner/tree/master*, 2023. – Accessed August 15, 2025

[CBL+19] Caesar, Holger ; Bankiti, Varun ; Lang, Alex H. ; Vora, Sourabh ; Liong, Venice E. ; Xu, Qiang ; Krishnan, Anush ; Pan, Yuxin ; Baldan, Giancarlo ; Beijbom, Oscar: nuScenes: A Multimodal Dataset for Autonomous Driving. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 11618-11628. *https://api.semanticscholar.org/CorpusID:85517967*

[CMS+20] Carion, Nicolas ; Massa, Francisco ; Synnaeve, Gabriel ; Usunier, Nicolas ; Kirillov, Alexander ; Zagoruyko, Sergey: End-to-End Object Detection with Transformers. In: Vedaldi, Andrea (Hrsg.) ; Bischof, Horst (Hrsg.) ; Brox, Thomas (Hrsg.) ; Frahm, Jan-Michael (Hrsg.): *Computer Vision – ECCV 2020*. Cham : Springer International Publishing, 2020. – ISBN 978–3–030–58452–8, S. 213–229

[COR98] Cybenko, George ; O'Leary, Dianne P. ; Rissanen, Jorma: *The mathematics of information coding, extraction and distribution*. Bd. 107. Springer Science & Business Media, 1998

[COR+16] Cordts, Marius ; Omran, Mohamed ; Ramos, Sebastian ; Rehfeld, Timo ; Enzweiler, Markus ; Benenson, Rodrigo ; Franke, Uwe ; Roth, Stefan ; Schiele, Bernt: *The Cityscapes Dataset for Semantic Urban Scene Understanding*. https://arxiv.org/abs/1604.01685. Version: 2016

[CRX+23] Chang, Xiaojun ; Ren, Pengzhen ; Xu, Pengfei ; Li, Zhihui ; Chen, Xiaojiang ; Hauptmann, Alex: A Comprehensive Survey of Scene Graphs: Generation and Application. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2023), Nr. 1, S. 1–26. http://dx.doi.org/10.1109/TPAMI.2021.3137605. – DOI 10.1109/TPAMI.2021.3137605

[CYR23] Cong, Yuren ; Yang, Michael Y. ; Rosenhahn, Bodo: RelTR: Relation Transformer for Scene Graph Generation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2023), Nr. 9, S. 11169–11183. http://dx.doi.org/10.1109/TPAMI.2023.3268066. – DOI 10.1109/TPAMI.2023.3268066

[CZY+24] Chen, Di ; Zhu, Meixin ; Yang, Hao ; Wang, Xuesong ; Wang, Yinhai: Data-Driven Traffic Simulation: A Comprehensive Review. In: *IEEE Transactions on Intelligent Vehicles* 9 (2024), Nr. 4, S. 4730–4748. http://dx.doi.org/10.1109/TIV.2024.3367919. – DOI 10.1109/TIV.2024.3367919

[DBK+21] Dosovitskiy, Alexey ; Beyer, Lucas ; Kolesnikov, Alexander ; Weissenborn, Dirk ; Zhai, Xiaohua ; Unterthiner, Thomas ; Dehghani, Mostafa ; Minderer, Matthias ; Heigold, Georg ; Gelly, Sylvain ; Uszkoreit, Jakob ; Houlsby, Neil: *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. https://arxiv.org/abs/2010.11929. Version: 2021

[DRC+17] Dosovitskiy, Alexey ; Ros, German ; Codevilla, Felipe ; Lopez, Antonio ; Koltun, Vladlen: *CARLA: An Open Urban Driving Simulator*. https://arxiv.org/abs/1711.03938. Version: 2017

[DSC22] Dubey, Shiv R. ; Singh, Satish K. ; Chaudhuri, Bidyut B.: Activation functions in deep learning: A comprehensive survey and benchmark. In: *Neurocomput.* 503 (2022), September, Nr. C, 92–108. http://dx.doi.org/10.1016/j.neucom.2022.06.111. – DOI 10.1016/j.neucom.2022.06.111. – ISSN 0925–2312

[Edw24] Edward, Kromm: Project Report Semantic Metrics. In: *Unpublished Report for Funded project nxtAIM* (2024)

[Epi] Epic Games: *Unreal Engine*. https://www.unrealengine.com

[GLU12] Geiger, Andreas ; Lenz, Philip ; Urtasun, Raquel: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, S. 3354–3361

[HZRS16] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, S. 770–778

[JKS+15] Johnson, Justin ; Krishna, Ranjay ; Stark, Michael ; Li, Li-Jia ; Shamma, David ; Bernstein, Michael ; Fei-Fei, Li: Image Retrieval Using Scene Graphs. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015

[KBK+19] Kang, Katie ; Belkhale, Suneel ; Kahn, Gregory ; Abbeel, P. ; Levine, Sergey: Generalization through Simulation: Integrating Simulated and Real Data into Deep Reinforcement Learning for Vision-Based Autonomous Flight. In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), 6008-6014. https://api.semanticscholar.org/CorpusID:60440689

[KRAD24] Kage, Patrick ; Rothenberger, Jay C. ; Andreadis, Pavlos ; Diochnos, Dimitrios I.: A Review of Pseudo-Labeling for Computer Vision. In: *ArXiv* abs/2408.07221 (2024). https://api.semanticscholar.org/CorpusID:271865922

[KSH17] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet classification with deep convolutional neural networks. In: *Commun. ACM* 60 (2017), Mai, Nr. 6, 84–90. http://dx.doi.org/10.1145/3065386. – DOI 10.1145/3065386. – ISSN 0001–0782

[KZG+17] Krishna, Ranjay ; Zhu, Yuke ; Groth, Oliver ; Johnson, Justin ; Hata, Kenji ; Kravitz, Joshua ; Chen, Stephanie ; Kalantidis, Yannis ; Li, Li-Jia ; Shamma, David A. ; Bernstein, Michael S. ; Fei-Fei, Li: Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. In: *International Journal of Computer Vision* 123 (2017), May, Nr. 1, 32–73. http://dx.doi.org/10.1007/s11263-016-0981-7. – DOI 10.1007/s11263–016–0981–7. – ISSN 1573–1405

[LH17] Loshchilov, Ilya ; Hutter, Frank: Decoupled Weight Decay Regularization. In: *International Conference on Learning Representations*, 2017

[LZZ+24] Li, Hongsheng ; Zhu, Guangming ; Zhang, Liang ; Jiang, Youliang ; Dang, Yixuan ; Hou, Haoran ; Shen, Peiyi ; Zhao, Xia ; Shah, Syed Afaq A. ; Bennamoun, Mohammed: Scene Graph Generation: A comprehensive survey. In: *Neurocomput.* 566 (2024), Januar, Nr. C. *http://dx.doi.org/10.1016/j.neucom.2023.127052*. – DOI 10.1016/j.neucom.2023.127052. – ISSN 0925–2312

[MYH+22] Malawade, Arnav V. ; Yu, Shih-Yuan ; Hsu, Brandon ; Muthirayan, Deepan ; Khargonekar, Pramod P. ; Faruque, Mohammad Abdullah A.: Spatiotemporal Scene-Graph Embedding for Autonomous Vehicle Collision Prediction. In: *IEEE Internet of Things Journal* 9 (2022), Nr. 12, S. 9379–9388. *http://dx.doi.org/10.1109/JIOT.2022.3141044*. – DOI 10.1109/JIOT.2022.3141044

[NOBK17] Neuhold, Gerhard ; Ollmann, Tobias ; Bulò, Samuel R. ; Kontschieder, Peter: The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, S. 5000–5009

[NZS23] Naidu, Gireen ; Zuva, Tranos ; Sibanda, Elias M.: A Review of Evaluation Metrics in Machine Learning Algorithms. In: Silhavy, Radek (Hrsg.) ; Silhavy, Petr (Hrsg.): *Artificial Intelligence Application in Networks and Systems*. Cham : Springer International Publishing, 2023. – ISBN 978–3–031–35314–7, S. 15–25

[PDL+21] Prakash, Aayush ; Debnath, Shoubhik ; Lafleche, Jean F. ; Cameracci, Eric ; State, Gavriel ; Law, Marc T.: *Sim2{SG}: Sim-to-Real Scene Graph Generation for Transfer Learning*. *https://openreview.net/forum?id=wbQXW1XTq_y*. Version: 2021

[PMR+23] Purwono, Purwono ; Ma'arif, Alfian ; Rahmaniar, Wahyu ; Imam, Haris ; Fathurrahman, Haris Imam K. ; Frisky, Aufaclav ; Haq, Qazi Mazhar U.: Understanding of Convolutional Neural Network (CNN): A Review. In: *International Journal of Robotics and Control Systems* 2 (2023), 01, S. 739–748. *http://dx.doi.org/10.31763/ijrcs.v2i4.888*. – DOI 10.31763/ijrcs.v2i4.888

[RDD24] Rüter, Joachim ; Durak, Umut ; Dauer, Johann C.: Investigating the Sim-to-Real Generalizability of Deep Learning Object Detection Models. In: *Journal of Imaging* 10 (2024), Nr. 10. *http://dx.doi.org/10.3390/jimaging10100259*. – DOI 10.3390/jimaging10100259. – ISSN 2313–433X

[Rei24] Reichardt, Jörg: *nxtAIM*. *https://nxtaim.de/projekt/*. Version: 2024. – Accessed: 2025-02-06

[Rog23] Rogstadkjernet, Magnus: *Utilization of synthetic and simulated cardiac imaging data in the training of artificial intelligence models - a literature review*, University of Oslo, Diplomarbeit, 2023. *https://www.duo.uio.no/handle/10852/103109*. – Accessed: 2025-02-07

[RTG+19] Rezatofighi, Hamid ; Tsoi, Nathan ; Gwak, JunYoung ; Sadeghian, Amir ; Reid, Ian ; Savarese, Silvio: Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, S. 658–666

[RVRK16a] Richter, Stephan R. ; Vineet, Vibhav ; Roth, Stefan ; Koltun, Vladlen: Playing for Data: Ground Truth from Computer Games. In: Leibe, Bastian (Hrsg.) ; Matas, Jiri (Hrsg.) ; Sebe, Nicu (Hrsg.) ; Welling, Max (Hrsg.): *European Conference on Computer Vision (ECCV)* Bd. 9906, Springer International Publishing, 2016 (LNCS), S. 102–118

[RVRK16b] Richter, Stephan R. ; Vineet, Vibhav ; Roth, Stefan ; Koltun, Vladlen: Playing for Data: Ground Truth from Computer Games. In: Leibe, Bastian (Hrsg.) ; Matas, Jiri (Hrsg.) ; Sebe, Nicu (Hrsg.) ; Welling, Max (Hrsg.): *Computer Vision – ECCV 2016*. Cham : Springer International Publishing, 2016. – ISBN 978–3–319–46475–6, S. 102–118

[SVN23] Selvaraj, Akila ; Venkatachalam, Deepak ; Namperumal, Gunaseelan: Synthetic Data for Financial Anomaly Detection: AI-Driven Approaches to Simulate Rare Events and Improve Model Robustness. In: *Journal of Artificial Intelligence Research and Applications* 2 (2023), Jan., Nr. 1, 373–425. *https://aimlstudies.co.uk/index.php/jaira/article/view/221*

[SZ15] Simonyan, K ; Zisserman, A: Very deep convolutional networks for large-scale image recognition, Computational and Biological Learning Society, 2015, S. 1–14

[TFR+17] Tobin, Josh ; Fong, Rachel ; Ray, Alex ; Schneider, Jonas ; Zaremba, Wojciech ; Abbeel, Pieter: Domain randomization for transferring deep neural networks from simulation to the real world. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, S. 23–30

[TGH+18] Tercan, Hasan ; Guajardo, Alexandro ; Heinisch, Julian ; Thiele, Thomas ; Hopmann, Christian ; Meisen, Tobias: Transfer-Learning: Bridging the Gap between Real and Simulation Data for Machine Learning in Injection Molding. In: *Procedia CIRP* 72 (2018), 185-190. *http://dx.doi.org/https://doi.org/10.1016/j.procir.2018.03.087*. – DOI https://doi.org/10.1016/j.procir.2018.03.087. – ISSN 2212–8271. – 51st CIRP Conference on Manufacturing Systems

[TPA+18] Tremblay, Jonathan ; Prakash, Aayush ; Acuna, David ; Brophy, Mark ; Jampani, Varun ; Anil, Cem ; To, Thang ; Cameracci, Eric ; Boochoon, Shaad ; Birchfield, Stan: Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization . In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA : IEEE Computer Society, Juni 2018, 1082-10828

[VSP+17] Vaswani, Ashish ; Shazeer, Noam ; Parmar, Niki ; Uszkoreit, Jakob ; Jones, Llion ; Gomez, Aidan N. ; Kaiser, Łukasz ; Polosukhin, Illia: Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA : Curran Associates Inc., 2017 (NIPS'17). – ISBN 9781510860964, S. 6000–6010

[WU18] Wrenninge, Magnus ; Unger, Jonas: *Synscapes: A Photorealistic Synthetic Dataset for Street Scene Parsing*. *https://arxiv.org/abs/1810.08705*. Version: 2018

[YCW+20] Yu, Fisher ; Chen, Haofeng ; Wang, Xin ; Xian, Wenqi ; Chen, Yingying ; Liu, Fangchen ; Madhavan, Vashisht ; Darrell, Trevor: *BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning*. *https://arxiv.org/abs/1805.04687*. Version: 2020

[YLWX18] Yang, Luona ; Liang, Xiaodan ; Wang, Tairui ; Xing, Eric: Real-to-Virtual Domain Unification for End-to-End Autonomous Driving. In: Ferrari, Vittorio (Hrsg.) ; Hebert, Martial (Hrsg.) ; Sminchisescu, Cristian (Hrsg.) ; Weiss, Yair (Hrsg.): *Computer Vision – ECCV 2018*. Cham : Springer International Publishing, 2018. – ISBN 978–3–030–01225–0, S. 553–570

[YMM+22a] Yu, Shih-Yuan ; Malawade, Arnav V. ; Muthirayan, Deepan ; Khargonekar, Pramod P. ; Faruque, Mohammad Abdullah A.: Scene-Graph Augmented Data-Driven Risk Assessment of Autonomous Vehicle Decisions. In: *IEEE Transactions on Intelligent Transportation Systems* 23 (2022), Nr.

7, S. 7941–7951. *http://dx.doi.org/10.1109/TITS.2021.3074854*. – DOI 10.1109/TITS.2021.3074854

[YMM+22b] Yu, Shih-Yuan ; Malawade, Arnav V. ; Muthirayan, Deepan ; Khargonekar, Pramod P. ; Faruque, Mohammad Abdullah A.: Scene-Graph Augmented Data-Driven Risk Assessment of Autonomous Vehicle Decisions. In: *IEEE Transactions on Intelligent Transportation Systems* 23 (2022), Nr. 7, S. 7941–7951. *http://dx.doi.org/10.1109/TITS.2021.3074854*. – DOI 10.1109/TITS.2021.3074854

[ZCC+25] Zhang, Jingyu ; Cao, Jin ; Chang, Jinghao ; Li, Xinjin ; Liu, Houze ; Li, Zhenglin: Research on the Application of Computer Vision Based on Deep Learning in Autonomous Driving Technology. In: Siarry, Patrick (Hrsg.) ; Jabbar, M. A. (Hrsg.) ; Cheung, Simon King S. (Hrsg.) ; Li, Xiaolong (Hrsg.): *Proceedings of the 2023 International Conference on Wireless Communications, Networking and Applications*. Singapore : Springer Nature Singapore, 2025. – ISBN 978–981–96–2409–6, S. 82–91

[ZKW+20] Zhang*, Tianyi ; Kishore*, Varsha ; Wu*, Felix ; Weinberger, Kilian Q. ; Artzi, Yoav: BERTScore: Evaluating Text Generation with BERT. In: *International Conference on Learning Representations*, 2020

[ZLX+24] Zheng, Zehan ; Lu, Fan ; Xue, Weiyi ; Chen, Guang ; Jiang, Changjun: LiDAR4D: Dynamic Neural Fields for Novel Space-time View LiDAR Synthesis. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, S. 5145–5154

[ZQL+24] Zhang, Yunpeng ; Qian, Deheng ; Li, Ding ; Pan, Yifeng ; Chen, Yong ; Liang, Zhenbao ; Zhang, Zhiyao ; Zhang, Shurui ; Li, Hongxu ; Fu, Maolei ; Ye, Yun ; Liang, Zhujin ; Shan, Yi ; Du, Dalong: GraphAD: Interaction Scene Graph for End-to-end Autonomous Driving. In: *ArXiv* abs/2403.19098 (2024). *https://api.semanticscholar.org/CorpusID:268732895*

[ZQW20] Zhao, Wenshuai ; Queralta, Jorge P. ; Westerlund, Tomi: Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, S. 737–744

[ZWBR+24] Zhao, Haonan ; Wang, Yiting ; Bashford-Rogers, Thomas ; Donzella, Valentina ; Debattista, Kurt: *Exploring Generative AI for Sim2Real in Driving Data Synthesis*. *https://arxiv.org/abs/2404.09111*. Version: 2024