Knowledge-Based Systems 242 (2022) 108245

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

ROADSCENE2VEC: A tool for extracting and embedding road scene-graphs

Arnav Vaibhav Malawade [*,1], Shih-Yuan Yu [1], Brandon Hsu, Harsimrat Kaeley, Anurag Karra, Mohammad Abdullah Al Faruque

Department of Electrical Engineering & Computer Science, University of California - Irvine, Irvine, CA 92697, USA

ARTICLE INFO

Article history:
Received 2 September 2021
Received in revised form 30 December 2021
Accepted 19 January 2022
Available online 29 January 2022

Keywords:
Scene-graph
Autonomous vehicles
Graph embedding
Vehicle safety
Graph learning
Knowledge graph

ABSTRACT

Recently, road scene-graph representations used in conjunction with graph learning techniques have been shown to outperform state-of-the-art deep learning techniques in tasks including action classification, risk assessment, and collision prediction. To enable the exploration of applications of road scene-graph representations, we introduce ROADSCENE2VEC: an open-source tool for extracting and embedding road scene-graphs. The goal of ROADSCENE2VEC is to enable research into the applications and capabilities of road scene-graphs by providing tools for generating scene-graphs, graph learning models to create spatio-temporal scene-graph embeddings, and tools for visualizing and analyzing scene-graph-based methodologies. The capabilities of ROADSCENE2VEC include (i) customized scene-graph generation from either video clips or data from the CARLA simulator, (ii) multiple configurable spatio-temporal graph embedding models and baseline CNN-based models, (iii) built-in functionality for using graph and sequence embeddings for risk assessment and collision prediction applications, (iv) tools for evaluating transfer learning, and (v) utilities for visualizing scene-graphs and analyzing the explainability of graph learning models. We demonstrate the utility of ROADSCENE2VEC for these use cases with experimental results and qualitative evaluations for both graph learning models and CNN-based models. ROADSCENE2VEC is available at https://github.com/AICPS/roadscene2vec.

1. Introduction

Autonomous Vehicles (AVs) are expected to revolutionize personal mobility, logistics, and road safety [1]. However, recent accidents involving Tesla Autopilot and Uber's self-driving cars indicate that the development of safe and robust AVs remains a difficult challenge [2–4]. Current statistics indicate that perception and prediction errors were factors in over 40% of driver-related crashes between conventional vehicles [5], leading both researchers and industry leaders to race to address these problems via advanced AV perception systems. Until recently, most AV perception architectures relied entirely on either (i) deep learning techniques using Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) [6–9]; or (ii) model-based methods, which use known road geometry information and vehicle trajectory models to estimate the state of the road scene [10,11]. Although these approaches have been successful in typical use cases, they cannot obtain a high-level, human-like understanding of complex road scenarios. This limitation is due to their inability to explicitly capture inter-object relationships or the overall structure of the road scene.

Research has suggested that humans rely on cognitive mechanisms to identify the structure of a scene and reason about inter-object relations when performing complex tasks (e.g., identifying risk) [12]. As such, the capability to capture and identify the complex relationships between road objects is critical in designing an effective human-like AV perception system. To address the limitations of existing AV perception methods, several groups have adopted a variant of knowledge graphs known as *scene-graphs* to model the road state and the relationships between objects [13–16]. A *scene-graph* representation encodes rich semantic information of an image or observed scene, essentially bringing an abstraction of objects and their complex relationships as illustrated in Fig. 1. While each of these related works proposes a different form of *scene-graph* representation, all demonstrate significant performance improvements over conventional perception methods. In [15], the authors propose a 3D-aware egocentric spatio-temporal interaction framework that uses both an *Ego-Thing* graph and an *Ego-Stuff* graph, which together encode how the ego vehicle interacts with both moving and stationary objects in a scene, respectively. In [14], the authors

* Corresponding author.
E-mail addresses: malawada@uci.edu (A.V. Malawade), shihyuay@uci.edu (S.-Y. Yu), bdhsu@uci.edu (B. Hsu), kaeleyh@uci.edu (H. Kaeley), karraa@uci.edu (A. Karra), alfaruqu@uci.edu (M.A. Al Faruque).

[1] Both authors contributed equally to this research.

https://doi.org/10.1016/j.knosys.2022.108245

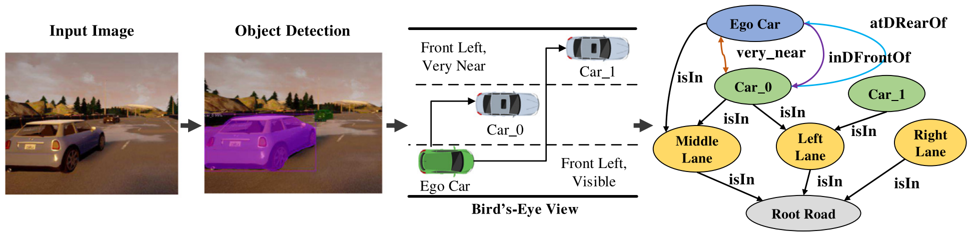

Fig. 1. How camera data can be used to construct a road *scene-graph* representation.

propose a pipeline using a multi-relational graph convolutional network (MR-GCN) for classifying the driving behaviors of traffic participants. The MR-GCN combines spatial and temporal information, including relational information between moving objects and landmark objects. Our prior work has demonstrated that a spatio-temporal *scene-graph* embedding can be used to assess the subjective risk of driving maneuvers more effectively than state-of-the-art methods [13,16]. In addition, our method can better transfer knowledge and is more explainable.

Although a wide range of *scene-graph* based AV perception approaches have been proposed, each method was developed from scratch, requiring significant time and resource investment by each research group. Although tools exist to perform pre-processing and graph learning (e.g., Pytorch and Pytorch Geometric), to the best of our knowledge, there exists no tool for systematically converting road scenes into *scene-graphs* in this field. As a result, each research group must start developing their *scene-graph* construction methodology from the ground up, wasting time and effort that could be better spent using the resultant *scene-graph* representations to solve more complex research problems. To address this problem, we propose ROADSCENE2VEC: a tool for systematically extracting and embedding road *scene-graphs*. ROADSCENE2VEC enables researchers to quickly and easily extract scene graphs from camera data, evaluate different graph construction methodologies, and use several different graph learning and machine learning algorithms to generate spatio-temporal graph embeddings for a wide range of AV tasks. We envision ROADSCENE2VEC to serve the following use cases:

- Converting image-based datasets as well as datasets generated by the CARLA simulator [17] into *scene-graphs*.
- Enabling the exploration of different *scene-graph* construction methodologies for a given application via a flexible, reconfigurable, and user-friendly *scene-graph* extraction framework.
- Allowing researchers to explore various spatio-temporal graph embedding methods, supporting customized algorithms for further design exploration.
- Providing a set of baselines drawn from state-of-the-art works for different AV applications (CNN and CNN-LSTM based algorithms).
- We provide *scene-graph* visualization utilities to enhance design space exploration for graph construction.

We target camera data since images are the most rich and detailed modality, providing high-resolution details about the scene as well as color information. This information can be used for better identifying the context of the scene and relations between participants. If other modalities are added, it is unlikely that much more information will be added to the scene graph; only the robustness of the system and precision of the graph will be improved. Besides, current state-of-the-art AV perception architectures utilizing sensor fusion still have shortcomings [18]. Furthermore, most publicly available AV datasets primarily contain image data.

1.1. Novel contributions

Our novel contributions for this research community are:

1. We present ROADSCENE2VEC: a flexible *scene-graph* construction and embedding framework that allows researchers to experiment with different graph extraction formulations to find the best one for their problem.
2. We provide an end-to-end graph learning framework for modeling the *scene-graph* representations. Our framework enables automated experimentation and metrics logging over a wide range of graph learning AV applications. We also provide a graph learning model template defining the core structure and functions used by our framework to facilitate users defining their own models and problems.
3. We provide many visualization tools and utilities for inspecting and understanding the *scene-graphs* including attention maps, color-coding by classes or relation type, birds-eye view projection, embedding projection, etc. These tools enable users to interpret their results easily without having to design their own visualizer.
4. We provide state-of-the-art CNN-based models drawn from recent AV papers for cross-comparison with graph-learning-based techniques.

1.2. Paper organization

The rest of our paper is laid out as follows. In Section 2 we discuss related works. In Section 3 we introduce the core functionality of our tool and its methodology. In Section 4 we provide usage examples. In Section 6 we demonstrate the practical, real-world value of our tool by evaluating it on several common use cases. Finally, in Section 7 we present our conclusions.

2. Related work

In this section, we begin by describing general AV design philosophies. Then we elaborate on graph-based approaches used in scene understanding. Lastly, we briefly discuss existing tools and libraries.

2.1. AV design methodologies

The two common design approaches for AV systems are (i) end-to-end deep learning architectures [19] and (ii) modular architectures. Modular approaches are implemented as a pipeline of separate components for performing each sub-task of the AV (e.g., perception, localization, planning, control). In contrast, end-to-end approaches generate actuator outputs (e.g., steering, brake, accelerator) directly from their sensory inputs [7]. One advantage of a modular design approach is the division of a task into an easier-to-solve set of sub-tasks that have been addressed in other fields such as robotics, computer vision, and vehicle dynamics, from which prior knowledge can be leveraged. However, one disadvantage of such an approach is the complexity of implementing, running and validating the complete pipeline [19]. End-to-end approaches can achieve good performance with smaller network size and low implementation costs because they perform feature extraction from sensor inputs implicitly through the network's hidden layers [7]. However, the authors in [20] point out that the needed level of supervision is too weak for the end-to-end model to learn critical control information (e.g., from image to steering angle), so it can fail to handle complicated driving maneuvers or be insufficiently robust to disturbances.

A third approach called the *direct perception* approach was first proposed by DeepDriving [20]. In this approach, a set of *affordance indicators*, such as the distance to lane markings and other cars in the current and adjacent lanes, are extracted from an image and serve as an intermediate representation (IR) for generating the final control output. They show that this IR improves performance for simple driving tasks such as lane following and enables better generalization to real-world environments. Similarly, [21] uses a collection of filtered images as the IR. They state that the IR used in their approach allows the training to be conducted on either real or simulated data, facilitating testing and validation in simulations before testing on a real car. Moreover, they show that it is easier to synthesize perturbations to the driving trajectory in the IR than at the raw sensor inputs themselves, enabling them to produce non-expert behaviors such as off-road driving and collisions. The authors in [6] use Mask-RCNN [22] to color the vehicles in each input image, producing a form of IR. In contrast to the works mentioned above, ROADSCENE2VEC utilizes a *scene-graph* IR that encodes the spatial and semantic relations between all the traffic participants in a frame. This form of representation is similar to a knowledge graph with the key distinction that *scene-graphs* explicitly encode knowledge about a visual scene.

2.2. Graph-based driving scene understanding

Several works have applied graph-based formulations for road scene understanding. In [15], the authors propose a 3D-aware egocentric Spatio-temporal interaction framework that uses both an *Ego-Thing* graph and an *Ego-Stuff* graph to encode how the ego vehicle interacts with both moving and stationary objects in a scene, respectively. In [14], the authors propose a pipeline using a multi-relational graph convolutional network (MR-GCN) for classifying the driving behaviors of traffic participants. The MR-GCN combines spatial and temporal information, including relational information between moving and landmark objects. In [23], the authors propose extracting road scene graphs in a manner that includes pose information for scene layout reconstruction. A similar approach was also proposed in [24]. Authors in [25] propose using a probabilistic graph approach for explainable traffic collision inference. In our prior work, we demonstrated that a *scene-graph* representation used with an MRGCN leads to state-of-the-art performance at assessing the subjective risk of driving maneuvers [13] and collision prediction [16]. Our tool implements examples of multi-relational graph learning models (MRGCN and MRGIN) and model skeletons, enabling users to evaluate other graph learning model formulations more easily.

2.3. Graph extraction and graph learning libraries

Other libraries for extracting *scene-graphs* from input images have been proposed. [26] proposed the Graph R-CNN model, which extracts scene graphs by identifying the set of individual objects in the image before identifying the spatial relations between the objects. With this process, Graph R-CNN can extract the spatial features of the scene in the form of a *scene-graph*. [27] provides a benchmark for evaluating several kinds of *scene-graph* generation models on image datasets. The *scene-graph* representations extracted by these tools are then used for semantic understanding and labeling tasks, such as image captioning and visual question answering. Although these tools and models are successful at these tasks, they do not incorporate specific domain knowledge relevant to the AV problem space. Autonomous driving is a highly complex problem on its own, so AV algorithms must utilize domain knowledge, including driving rules, road layout, and markings, as well as light and sign information. Furthermore, AV algorithms must account for temporal factors; the tools mentioned above operate on individual images and thus do not account for these safety-critical temporal factors.

Regarding graph learning tools and libraries, several tools such as GraphGYM [28], DGL [29], and OGB [30] exist for quickly and easily evaluating several graph learning models on problems including node/graph classification and regression. However, none of these pre-existing tools enable *scene-graph* generation; they can only be used with existing graph data. Our proposed tool is the only tool that enables both the extraction and learning of AV-specific *scene-graphs*.

3. ROADSCENE2VEC Architecture

This section introduces ROADSCENE2VEC's architecture, features, and intended workflow. Our ROADSCENE2VEC is implemented as a Python library, integrating various external packages such as APIs from PyTorch, PyTorch Geometric, Detectron2, and CARLA. ROADSCENE2VEC consists of four key modules: (i) data generation (**data.gen**) and preprocessing (**data.proc**), (ii) *scene-graph* extraction (**scene_graph**), (iii) model training and evaluation (**learning**), and (iv) visualization (**util**). We detail each module in the following subsections.

3.1. Dataset generation tools (data.gen)

The module **data.gen** in ROADSCENE2VEC allow researchers to synthesize driving data for their research. To successfully handle complex and long-tail driving scenarios, deep learning approaches typically train their models on large datasets that contain a wide range of "corner cases". However, generating such datasets is expensive and time-consuming in the real-world [17]. Thus, most researchers instead use synthesized datasets containing plenty of these corner cases to evaluate their research ideas.

For this purpose, ROADSCENE2VEC integrates the open-source driving simulator, CARLA [17], which allows users to generate driving data by controlling a vehicle (either in manual mode or autopilot mode) in simulated driving scenarios. On top of that, ROADSCENE2VEC also integrates the CARLA Scenario Runner, which contains a set of atomic controllers that enable users to automate the execution of complex driving maneuvers.

In ROADSCENE2VEC, **data.gen** produces each driving clip in CARLA's simulated world by (i) selecting one autonomous car randomly, (ii) switching its mode to manual mode, and (iii) using the Scenario Runner to command the vehicle to change lanes. In addition, the data generation tool in ROADSCENE2VEC manipulates the various presets in CARLA to specify the number of cars, pedestrians, weather and lighting conditions, etc., for making the generated driving data more diverse. Moreover, through the APIs provided by the Traffic Manager (TM) of the CARLA simulator, the tool can customize the driving characteristics of every autonomous vehicle in the simulated world, such as the intended speed considering the current speed limit, the chance of ignoring the traffic lights, or the chance of neglecting collisions with other vehicles. Overall, the tool allows users to simulate a wide range of very realistic urban driving environments and

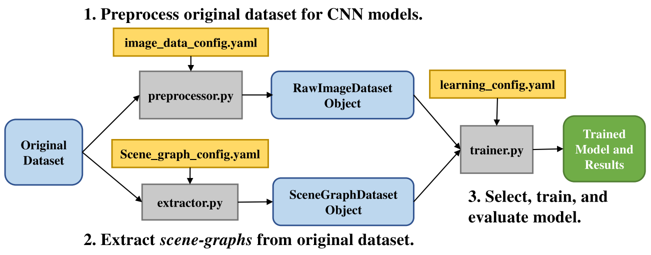

Fig. 2. Workflow for using ROADSCENE2VEC to preprocess a dataset; extract *scene-graphs* from the dataset; and select, train, and evaluate a model on the dataset.

generate synthesized datasets suitable for training and testing a model.

Using the CARLA Python API and the CARLA Scenario Runner, we implemented a tool in the **data.gen** module for extracting the road scene's state information as well as the corresponding ego-centric camera images directly from the CARLA simulator for use in ROADSCENE2VEC. For each frame in a driving sequence, we store the attributes of all the objects as a Python dictionary. These attributes include object type, location, rotation, lane assignment, acceleration, velocity, and light status. For static objects such as traffic lights and signage, we store the type of object, its location, and light state (light color) or sign value (e.g., speed limit). We refer to the datasets in this format as CARLA datasets. In addition, our tool supports using image-based datasets, such as the camera data extracted from CARLA or the Honda Driving Dataset [31] used in our experiments. The code provided in our data.gen module can be modified to support other driving actions, such as turning, accelerating, braking, and overtaking.

Under the **data.gen** module, ROADSCENE2VEC also provides an annotation tool for quickly and easily labeling both CARLA datasets and image datasets. The annotator offers a graphical user interface (GUI) that enables users to view, label, exclude or trim specific driving sequences. Our annotator enables users to assign one label for each sequence and supports averaging multiple independent labelers' decisions. Our annotators GUI is shown in Fig. 3. In comparison to popular annotation tools, such as CVAT [32] and VoTT [33], our annotator offers a more streamlined approach for video clip labeling. These other annotation tools typically iterate through only a single image at a time as they are primarily designed for tasks such as object detection and semantic segmentation. In contrast, our annotator implements a broader range of video playback controls (play, pause, replay, ignore, etc.), facilitating risk analysis over a complete video sequence. Currently, our annotator supports sequence-level risk labels from 1–5. Still, it can be extended to support other label types for problems such as scenario classification and rare event identification. In addition to the annotation tool, we also provide dataset utilities such as train-test splitting, k-fold cross-validation, and downsampling as part of the trainers in the **learning.util** module.

3.2. Data preprocessing (roadscene2vec.data.proc)

The data storage and preprocessing functions are implemented through the **data.proc** module of ROADSCENE2VEC. To use a new dataset with ROADSCENE2VEC, it must first have the correct directory structure defined in our repository. Next, the input dataset can go through one of the two workflows shown in Fig. 2: (i) the dataset is preprocessed into a "RawImageDataset" to be used with CNNs and other image processing models directly, or (ii) the dataset is sent to the corresponding *scene-graph* extractor to generate *scene-graph* representations of every frame in the dataset (discussed in Section 3.3). The preprocessing step is necessary

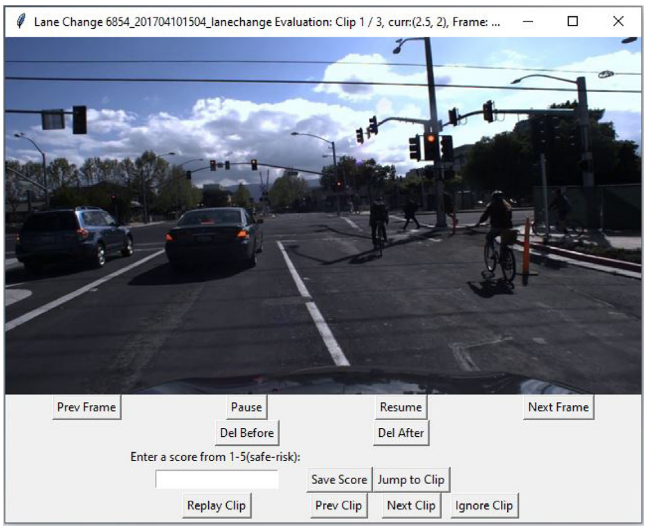
Lane Change 6854_201704101504_lanechange Evaluation: Clip 1 / 3, curr:(2.5, 2), Frame: ...
Prev Frame
Pause
Resume
Next Frame
Del Before
Del After
Enter a score from 1-5(safe-risk):
Save Score
Jump to Clip
Replay Clip
Prev Clip
Next Clip
Ignore Clip

Fig. 3. The user interface of the annotator tool, used to label, filter, and trim datasets.

for the conventional deep-learning models; the input images often need to be resized, reshaped, or sub-sampled before being trained with models to meet memory and space constraints. After preprocessing, the RawImageDataset object stores the sets of driving video clips as image sequences, the labels associated with the video clips, and metadata (such as sequence name/action type). For each image in each clip in the dataset, the image preprocessor loads the image using OpenCV, resizes and recolors the image according to the configuration settings, and stores the image as a PyTorch Tensor. The resulting RawImageDataset object is then serialized and stored as a pickle (.pkl) file.

3.3. Road *scene-graph* extraction (roadscene2vec.scene_graph.extraction)

Here, we describe how an input dataset is converted into a "SceneGraphDataset" object via our *scene-graph* extraction framework. We first describe how the entities and relations in the *scene-graph* are defined and configured before discussing the specific steps needed to extract *scene-graphs* from both CARLA and image-based datasets.

3.3.1. Entity and relation extraction

A list of ROADSCENE2VEC's user-configurable *scene-graph* extraction settings is shown in Table 1. In our formulation, each "actor" (object) in the *scene-graph* is assigned a type from the set {car, motorcycle, bicycle, pedestrian, lane, light, sign}, matching those defined by CARLA. Users can reconfigure the set of object types to support other dataset types, applications, or ontologies.

The default relation extraction pipeline we implement identifies three kinds of pair-wise relations: *proximity* relations (e.g. *visible*, *near*, *very_near*, etc.), *directional* relations (e.g. *Front_Left*, *Rear_Right*, etc.), and *belonging* relations (e.g. car_1 *isIn* left_lane). Two objects are assigned the *proximity* relation, $r \in$ {*Near_Collision* (4 ft.), *Super_Near* (7 ft.), *Very_Near* (10 ft.), *Near* (16 ft.), *Visible* (25 ft.)} provided the objects are physically separated by a distance that is within that relation's threshold. The *directional relation*, $r \in$ {*Front_Left*, *Left_Front*, *Left_Rear*, *Rear_Left*, *Rear_Right*, *Right_Rear*, *Right_Front*, *Front_Right*}, is assigned to a pair of objects, in this case between the ego-car and another car in the view, based on their relative orientation and only if they are within the *near* threshold distance from one another. Additionally, the *isIn* relation identifies which vehicles are on which

Table 1
Scene graph configuration options and their descriptions. Each of these parameters can be reconfigured by the user to produce custom *scene-graphs*.

Parameter	Description
actor_names	The list of object types. The default list is based on the actor types defined by the CARLA simulator.
relation_names	The list of all implemented relation types.
car_names /moto_names /bicycle_names/etc.	Object names defined in the CARLA simulator. These lists are used to cross-reference the object type for a given CARLA vehicle name.
directional_thresholds	Defines the set of enabled directional relations and their thresholds in degrees.
directional_relation_list	Defines the pairs of object types for which directional relations will be extracted.
proximity_thresholds	Defines the set of enabled distance relations and their thresholds in feet.
proximity_relation_list	Defines the pairs of object types for which proximity relations will be extracted.
lane_threshold	Represents 50% of the width of a lane in feet. If an object is more than this distance from the ego car's center, it is considered to be in the left or right lane.

lanes (see Fig. 1). We use each vehicle's horizontal displacement relative to the ego vehicle to assign vehicles to either the *Left Lane*, *Middle Lane*, or *Right Lane* using the known lane width. Our current abstraction only considers three-lane areas, and, as such, we map vehicles in all left lanes and all right lanes to the same *Left Lane* node *Right Lane* node, respectively. If a vehicle overlaps two lanes (i.e., during a lane change), it is mapped to both lanes.

The set of possible entity types, relation types, relation thresholds, and valid object pairs is defined in the *scene_graph_config* file. These settings are entirely user re-configurable, enabling broad design space exploration of different graph extraction methodologies. After graph extraction is completed, the set of all *scene-graph* sequences, metadata, and labels are saved as a SceneGraphDataset.

3.3.2. CARLA scene-graph extraction

Since the CARLA datasets contain a dictionary with a list of objects and their attributes, we directly use this dictionary to initialize the nodes in the *scene-graph*. Each node is assigned its type label from the set of actor_names and its corresponding attributes (e.g., position, angle, velocity, current lane, light status, etc.) for relation extraction. Once all nodes are added to the *scene-graph*, we extract relations between each pair of objects in the scene.

3.3.3. Image scene-graph extraction

To extract *scene-graphs* from image-based datasets, we first need to identify the set of objects in each image along with their attributes. We use Mask-RCNN [22] to extract the set of objects in the image as well as their bounding boxes. Next, we compute the inverse-perspective mapping transformation of the image, yielding a top-down 'birds-eye view' (BEV) projection of the scene. By generating this projection and projecting the bounding box coordinates from the original image into the birds-eye view, we can estimate the position of each vehicle relative to the ego-vehicle with reasonably high fidelity. This position information, along with the object class information, is used to construct the *scene-graphs*. However, the BEV projection needs to be re-calibrated for each dataset, as typically, each dataset uses a different camera angle and camera configuration. To facilitate this calibration step, we provide a BEV calibration utility in **scene_graph.extraction.bev**. This utility provides an interactive way for the user to select the road area and calibrate the BEV projection for a new dataset with a single step.

3.3.4. Scene-graph visualization

Our *scene-graph* visualization tool, located in the **roadscene2vec.util** module, consists of a GUI that simultaneously displays an input image side by side with its corresponding *scene-graph*, as is shown in Fig. 8. This tool enables researchers to experiment with a wide range of relation types and distance thresholds and quickly optimize their *scene-graph* extraction settings for their specific application or dataset.

3.4. Scene-graph embedding (roadscene2vec.learning)

The **learning** module contains our framework for splitting datasets as well as training, testing, and scoring models at various tasks. It also contains our graph learning models, baseline deep learning models, and a graph learning model template to enable users to define their own graph models for use with our framework.. The **model** submodule contains the model definitions while the **util** submodule contains the training, evaluation, and scoring functions. The training code supports implementing k-fold cross-validation, a user-definable train-test split, and downsampling and class weighting to correct dataset imbalances. The model specification, training hyperparameters, and dataset configuration settings are loaded from the *learning_config* file, which is user-modifiable. Next, we introduce the models available in ROADSCENE2VEC.

3.4.1. Graph learning models (roadscene2vec.learning.model)

The graph learning models we provide in ROADSCENE2VEC enable various configurations of both spatial modeling and temporal modeling components as shown in Fig. 4. The spatial modeling components that are configurable include (i) graph convolution layers, (ii) graph pooling and graph attention layers, and (iii) graph readout operations. The configurable temporal modeling components include (i) temporal modeling layers and (ii) temporal attention layers. Our experiments use MRGCN and MRGIN models that are identical in structure and differ only in the type of spatial modeling used. Next, we discuss these components in more detail.

Spatial modeling (SPATIAL_MODEL). We provide two multi-relational graph convolution implementations based on (i) graph convolutional networks (GCNs) [34] and (ii) graph isomorphism networks (GINs) [35]. These layers propagate node embeddings across edges via graph convolutions, resulting in a new set of node embeddings. The two implementations differ in how data is propagated through successive graph convolutions. Graph pooling filters the set of node embeddings in the graph to only those most useful for the task. We enable two types of graph pooling layers extended for multi-relational use cases: Self-Attention Graph Pooling (SAGPool) [36] and Top-K Pooling (TopkPool) [37]. After pooling, a global readout operation is used to collect the set of pooled node embeddings into a unified graph embedding. We implement *max*, *mean*, and *add* readout operations.

Temporal modeling (TEMPORAL_MODEL). The temporal model we implement uses Long Short-Term Memory (LSTM) layers to convert the sequence of *scene-graph* embeddings to either (i) one spatio-temporal embedding (for sequence classification tasks) or (ii) a sequence of spatio-temporal embeddings (for graph classification/prediction tasks). For graph classification/collision prediction tasks, the output from an LSTM layer for each input *scene-graph* embedding is collected as a sequence of spatio-temporal *scene-graph* embeddings P that is sent to an MLP layer to produce the final set of model outputs. For sequence classification tasks, a temporal readout operation is applied to P to compute a single spatio-temporal sequence embedding $\mathbf{z}$ by (i) extracting only the last hidden state of the LSTM p_T (LSTM-last), (ii) taking the sum over $\mathbf{P}$, or (iii) using a temporal attention layer (LSTM-attn) to compute an attention-weighted sum of the different elements of P as described in [38].

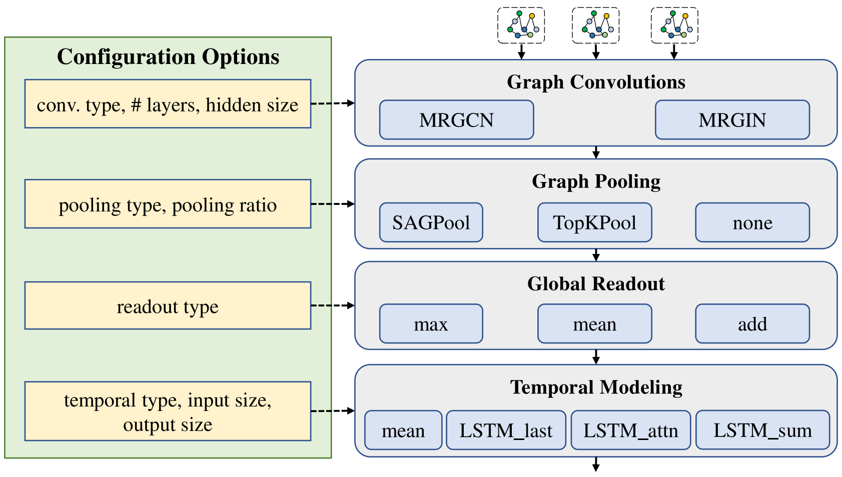

Fig. 4. Graph learning model configuration options provided in ROADSCENE2VEC.

3.4.2. *Baseline models (roadscene2vec.learning.model)*

In addition to the graph learning models that are core to ROADSCENE2VEC, we also provide a set of baseline deep learning models for quickly and easily comparing to typical image-processing approaches. These baselines include (i) a ResNet-50 [39] CNN classifier and (ii) a CNN+LSTM classifier [6]. The motivation for using these baselines stems from their prevalence in AV image processing tasks, such as risk assessment [6]. Users can easily use other graph or deep-learning models with our framework as long as they follow the same, typical PyTorch model structure.

3.4.3. *Performance evaluation and hyperparameter optimization*

To enable live monitoring of training runs and in-depth analysis of the effects of different hyperparameter settings on performance, we integrate our library with Weights and Biases (W&B).[2] W&B is a free, publicly available tool for tracking experiments, visualizing performance, identifying hyperparameter importance, and organizing results. We believe this integration will enable researchers to identify trends in the data and optimize model performance more quickly.

4. Usage examples

In this section, we describe some of ROADSCENE2VEC's use-cases. First, Section 4.1 exhibits a fundamental use-case in which an image frame I is converted into a *scene-graph* g and then into a fixed-length embedding h_g. Next, the use cases of ROADSCENE2VEC for two risk-based autonomous driving applications (subjective risk assessment and collision prediction) are described in Section 4.2 and Section 4.3, respectively. In Section 4.4, we discuss how ROADSCENE2VEC can be used for performing and evaluating transfer learning. Finally, in Section 4.5, we show how ROADSCENE2VEC can be used to analyze the explainability of the graph learning models.

4.1. *Use Case 1: Converting an ego-centric observation into a scene-graph*

Our high-level algorithm for converting an input image into a *scene-graph* is shown in Algorithm 1. Let us walk through a typical workflow for converting an image dataset into a set of *scene-graph* embeddings. First, the preprocessor processes the image to set the dataset format and image sizing. These *scene-graphs* can then be visualized using the visualizer tool we provide. The following script streamlines the execution of this use case:

```
> python examples/use_case_1.py
```

These scripts take configuration information directly from the data_config and scene_graph_config files in the **config** module. The config files indicate which type of dataset is being used (CARLA or image-based) and the location and extraction settings for the dataset. The scene_graph_config file also allows the reconfiguration of the relation extraction settings as shown in Table 1. The choice of relation extraction settings changes the *scene-graph* structure, which can change how the graph learning model processes the data.

Algorithm 1: Use Case 1 - Extracting a sequence of *scene-graphs* from a driving clip.

```
1  Input: A sequence of images from a driving video clip $I$.
2  Output: A sequence of scene graphs $G$ for $I$.
3  def IMG2GRAPH($I_t$):
4      $O_t \leftarrow$ OBJ_DETECTION($I_t$)
5      $A_t \leftarrow$ ATTR_EXTRACTION($I_t, O_t$)
6      $G_t \leftarrow$ GRAPH_EXTRACTION($O_t, A_t$)
7      return $G_t$
8  def EXTRACT_SEQ($I$):
9      $G \leftarrow \{\}$
10     for $I_t$ in $I$ do
11         $G_t \leftarrow$ IMG2GRAPH($I_t$)
12     end
13     return $G$
14 $G \leftarrow$ EXTRACT_SEQ($I$)
```

4.2. *Use Case 2: Subjective risk assessment*

In prior AV research, attempts to improve vehicle safety have involved modeling either the *objective risk* or the *subjective risk* of driving scenes [40–42]. The *objective risk* is defined as the objective probability of an accident occurring and is typically determined by statistical analysis [40]. In contrast, *subjective risk* refers to the driver's own perceived risk and is an output of the driver's cognitive process [41,42]. Since subjective risk accounts for the human behavior perspective and its critical role in anticipating risks [41–43], it has the potential to assess contextual risk better than objective methods and thus better assure passenger safety. Further, studies such as [40,44] provide direct evidence that a driver's subjective risk assessment is inversely related to the risk of traffic accidents. Within this context, AVs must support understanding driving scenes and quantify the subjective risk of driving decisions.

[2] https://wandb.ai/.

Given this motivation, we show that the graph learning models available in ROADSCENE2VEC can be used to convert these extracted *scene-graphs* into spatio-temporal *scene-graph* embeddings for the task of subjective risk assessment, as was done in our prior work [13].

4.2.1. Problem formulation

In our prior work [13], and here, we make the same assumption used in [6] that the set of driving sequences can be partitioned into two jointly exhaustive and mutually exclusive subsets: risky and safe. We denote the sequence of images of length T by $\mathbf{I} = \{I_1, I_2, I_3, \dots, I_T\}$. We assume the existence of a spatio-temporal function f that outputs whether a sequence of driving actions x is safe or risky via a risk label y, as given in Eq. (1).

$$y = f(\mathbf{I}) = f(\{I_1, I_2, I_3, \dots, I_{T-1}, I_T\}), \tag{1}$$

where

$$y = \begin{cases} (1,0), & \text{if the driving sequence is safe} \\ (0,1), & \text{if the driving sequence is risky.} \end{cases} \tag{2}$$

Overall, the goal of the model is to learn to approximate the function f. Our algorithmic implementation of this use case is shown in Algorithm 2.

Algorithm 2: Use Case 2 - *Scene-graph* embedding for risk assessment

```
Input: A sequence of images from a driving video clip $I$.
Output: Risk assessment $\hat{Y}$.
def SEQ2VEC($G$):
    $h_G \leftarrow \{\ \}$
    for $G_t$ in $G$ do
        $\mathbf{h}_{G_t} \leftarrow$ SPATIAL_MODEL($G_t$)
    end
    $Z \leftarrow$ TEMPORAL_MODEL($\mathbf{h}_G$)
    $\hat{y}_0, \hat{y}_1 \leftarrow$ ACTIVATION(MLP($Z$))
    if $\hat{y}_1 \geq \hat{y}_0$ then
        return 1
    else if $\hat{y}_0 > \hat{y}_1$ then
        return 0
def RISK_ASSESS($I$):
    $G \leftarrow$ EXTRACT_SEQ($I$)
    $\hat{Y} \leftarrow$ SEQ2VEC($G$)
    return $\hat{Y}$
$\hat{Y} \leftarrow$ RISK_ASSESS($I$)
```

4.2.2. Training

To achieve this goal, we train the graph learning model using the extracted sequences of *scene-graphs* as inputs and the subjective risk labels given by human annotators for each sequence. As such, the problem becomes a simple sequence classification problem, where the goal is to classify a given sequence of images as *risky* or *safe*. The configuration settings for training the model are available in the learning_config file in the **config** module. The following command can be used to train the model for risk assessment:

```
> python examples/use_case_2.py
```

4.3. Use Case 3: Collision prediction

In our third use case, we demonstrate how ROADSCENE2VEC can be used to study approaches for predicting future vehicle collisions, as was done in our prior work [16]. In contrast to Use Case 2, which is a sequence classification problem, collision prediction has safety-critical time constraints. It uses the history of prior *scene-graphs* to make predictions about the state of future graphs. Current statistics indicate that perception and prediction errors were factors in over 40% of driver-related crashes between conventional vehicles [5]. However, a significant number of reported AV collisions are also the result of these errors [45,46]. With this motivation, we show that *scene-graphs* can be used to represent road scenes and model inter-object relationships to improve perception and scene understanding. An example of our methodology is shown in Fig. 6.

4.3.1. Problem formulation

We formulate the problem of collision prediction as a time-series classification problem where the goal is to predict if a collision will occur in the *near future*. Our goal is to accurately model the spatio-temporal function f, where

$$\mathbf{Y_n} = f(\{I_1, \dots, I_{n-1}, I_n\}), \mathbf{Y_n} \in \{0, 1\}, \text{for } n > 2, \tag{3}$$

where $\mathbf{Y_n} = 1$ implies a collision in the near future and $\mathbf{Y_n} = 0$ otherwise. Here the variable I_n denotes the image captured by the on-board camera at time n. The interval between each frame varies with the camera sampling rate. Our implementation of Use Case 3 is shown in Algorithm 2.

Algorithm 3: Use Case 3 - *Scene-graph* embedding for collision prediction

```
Input: A sequence of images from a driving video clip $I$.
Output: Sequence of collision likelihood predictions: $\hat{Y}$.
def GRAPH2VEC($G_t$, $p_{t-1}$, $c_{t-1}$):
    $\mathbf{h}_{G_t} \leftarrow$ SPATIAL_MODEL($G_t$)
    $p_t, c_t \leftarrow$ TEMPORAL_MODEL($\mathbf{h}_{G_t}, p_{t-1}, c_{t-1}$)
    $\hat{y}_0, \hat{y}_1 \leftarrow$ ACTIVATION(MLP($p_t$))
    if $\hat{y}_1 \geq \hat{y}_0$ then
        return 1, $p_t$
    else if $\hat{y}_0 > \hat{y}_1$ then
        return 0, $p_t$
def COLLISION_PRED($I$):
    $G \leftarrow$ EXTRACT_SEQ($I$)
    $p_0, c_0 \leftarrow$ [0, 0, ..., 0] , [0, 0, ..., 0]
    $\hat{Y} \leftarrow \{\ \}$
    for $G_t$ in $G$ do
        $\hat{Y}_t, p_t \leftarrow$ GRAPH2VEC($G_t$, $p_{t-1}$, $c_{t-1}$)
        $t \leftarrow t + 1$
    end
    return $\hat{Y}$
$\hat{Y} \leftarrow$ COLLISION_PRED($I$)
```

4.3.2. Training

To train a model for this application, we adjust the model to produce one output per graph instead of one output per sequence. For the application of collision prediction, we also assign each frame in a video clip a label identical to the entire clip's label to train the model to identify the preconditions of a future collision and predict it as early as possible. The following command can be used to train the model for collision prediction:

```
> python examples/use_case_3.py
```

4.4. Use Case 4: Transfer learning

Models trained on simulated datasets must be able to transfer their knowledge to real-world driving scenarios as they can differ significantly from simulations. One key advantage of using *scene-graphs* is that they are a form of Intermediate Representation (IR), meaning that they provide a higher level of abstraction compared to image data alone. This abstraction means that *scene-graphs* are generally better able to transfer knowledge across datasets and domains, such as from simulated data to real-world driving data. Since this is a key benefit of using a graph-based approach and is a critical use case for validating AV safety, ROADSCENE2VEC supports running transfer learning experiments between any two datasets. To implement this use case, we use the original dataset to train the model and use the user-specified transfer dataset to test the model. No additional domain adaptation is performed. The workflow for Use Case 4 is shown in Algorithm 4. The following script runs an example of transfer learning.

```
> python examples/use_case_4.py
```

Algorithm 4: Use Case 4 - Transfer learning evaluation

```
1  Input: Source dataset $D_S$, transfer dataset $D_T$, model $m$, and training epochs $E$.
2  Output: Transfer learning result $R_T$.
3  def TRAIN($D$, $m$, $E$):
4      for epoch in $E$ do
5          $X, Y \leftarrow D$
6          $O \leftarrow m(X)$
7          $L \leftarrow$ LOSS_FUNCTION($O, Y$)
8          $m \leftarrow$ UPDATE_MODEL($L, m$)
9      end
10     return $m$
11 def EVALUATE($D$, $m$):
12     $X, Y \leftarrow D$
13     $O \leftarrow m(X)$
14     $R \leftarrow$ SCORE($O, Y$)
15     return $R$
16 def TRANSFER_KNOWLEDGE($D_S$, $D_T$, $m$, $E$):
17     $m' \leftarrow$ TRAIN($D_S, m, E$)
18     $R_T \leftarrow$ EVALUATE($D_T, m'$)
19     return $R_T$
20 $R_T \leftarrow$ TRANSFER_KNOWLEDGE($D_S, D_T, m, E$)
```

4.5. Use Case 5: Explainability analysis

Explainability refers to the ability of a model to communicate the factors that influenced its decision-making process for a given input, particularly those that might lead the model to make incorrect decisions [47,48]. Since deep-learning models are typically black-boxes, they are difficult to diagnose and adjust when failures occur. Thus, models which can better explain their decision-making process are easier to verify, debug, and make safer. Our library enables users to analyze the explainability of different model architectures by visualizing the node attention scores of a graph learning model for a given input. The workflow of this use case is shown in Algorithm 5. First, using a pre-trained graph learning model, we run inference on a dataset and record the model's spatial and temporal attention scores for each sequence to a CSV file. Then, we visualize the node attention scores for each *scene-graph* and color code the nodes according to their attention score. For a given graph, the nodes with higher attention scores had a more significant impact on the decision made by the model.

5. Experiments

In this section, we present results from running each use case presented in Section 4 as well as details on the datasets and metrics used to evaluate each model.

Algorithm 5: Use Case 5 - Explainability analysis of *scene-graph* risk assessment

```
1  Input: A sequence of images from a driving video clip $I$, trained model $m$.
2  Output: Risk assessment result $\hat{Y}$, node attention scores $\alpha_t$ and temporal attention score $\beta_t$ for each graph in $G$.
3  def SEQ2VEC_ATTN($G$):
4      $h_G, \alpha \leftarrow \{\}, \{\}$
5      for $G_t$ in $G$ do
6          $\mathbf{h}_{G_t}, \alpha_t \leftarrow$ SPATIAL_MODEL($G_t$)  // $\alpha_t$ from SAGPool layer
7      end
8      $Z, \beta \leftarrow$ TEMPORAL_MODEL($\mathbf{h}_G$)  // $\beta$ from LSTM-attn layer
9      $\hat{y}_0, \hat{y}_1 \leftarrow$ ACTIVATION(MLP($Z$))
10     if $\hat{y}_1 \geq \hat{y}_0$ then
11         return $1, \alpha, \beta$
12     else if $\hat{y}_0 > \hat{y}_1$ then
13         return $0, \alpha, \beta$
14 def GET_ATTENTION_SCORES($I$):
15     $G \leftarrow$ EXTRACT_SEQ($I$)
16     $\hat{Y}, \alpha, \beta \leftarrow$ SEQ2VEC_ATTN($G$)
17     return $\hat{Y}, \alpha, \beta$
18 $\hat{Y}, \alpha, \beta \leftarrow$ GET_ATTENTION_SCORES($I$)
```

5.1. Dataset preparation

For experiments, we prepared two types of driving datasets: (i) synthesized lane-changing datasets (*271-syn* and *1043-syn*), and (ii) typical real-world driving datasets (*571-honda* and *1361-honda*). We labeled all of the datasets using our annotator tool as described in Section 3.1. More details on the datasets as well as the labeling process can be found in [13]. We randomly split each dataset into a training set and a testing set by the ratio 7:3 such that the split is stratified, i.e., the proportion of risky to safe lane change clips in the training and testing sets is the same. The models are first trained on the training set before being evaluated on the testing set. The final score of a model on a dataset is computed by averaging over the testing set scores for five different stratified train-test splits.

5.2. Model configuration

In our experiments, we use two graph learning architectures denoted MRGCN and MRGIN. Both models consist of the following structure: two graph convolution layers of size 64, one SAGPooling layer with 0.5 pooling ratio, one *add* readout layer, and one problem-specific temporal model as defined in Fig. 5. The two architectures only differ in the way successive graph convolutions are processed, as discussed in Section 3.4.1. As for the baselines, we evaluate the ResNet-50 CNN classifier and the CNN+LSTM classifier in our experiments. All models were evaluated using 5-fold cross-validation with the average test performance over the five folds presented as the final result.

5.3. Use Case 1 evaluation: Scene-graph extraction

In Fig. 7, we show an example where two scene graphs are extracted from the same input image with different relation extraction settings. The graph at the bottom contains relations between all pairs of vehicles in the scene; for each pair of vehicles, if the two vehicles are within some distance threshold, the distance

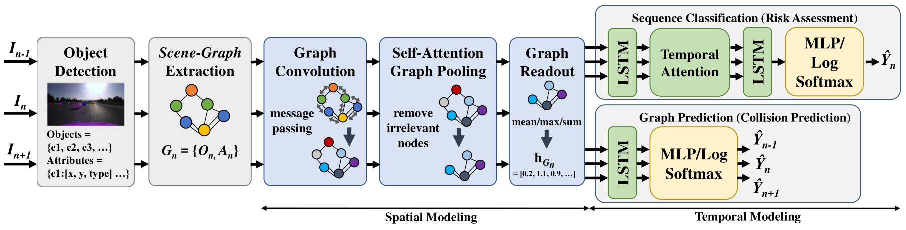

Fig. 5. The architecture of our configurable *scene-graph* based AV perception model. Our two pre-implemented temporal modeling pipelines for specific AV tasks are shown (sequence classification and graph prediction). However, users can remove or replace these model components for performing other AV tasks such as graph classification or scenario classification.

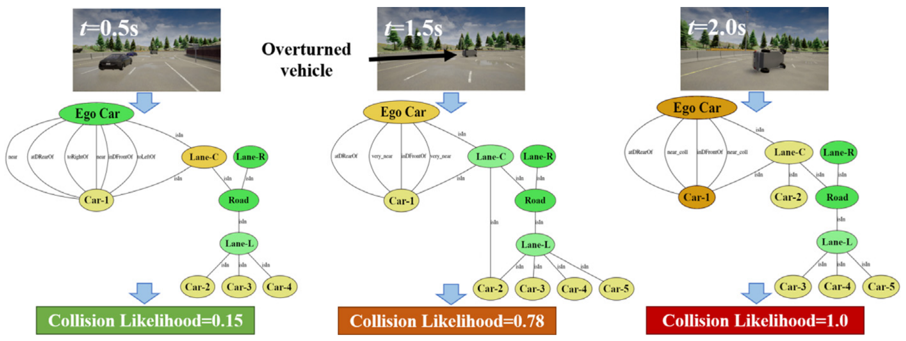

Fig. 6. Demonstration of collision prediction using *scene-graphs*. Each node's color indicates its attention score (importance to the collision likelihood) from orange (high) to green (low).

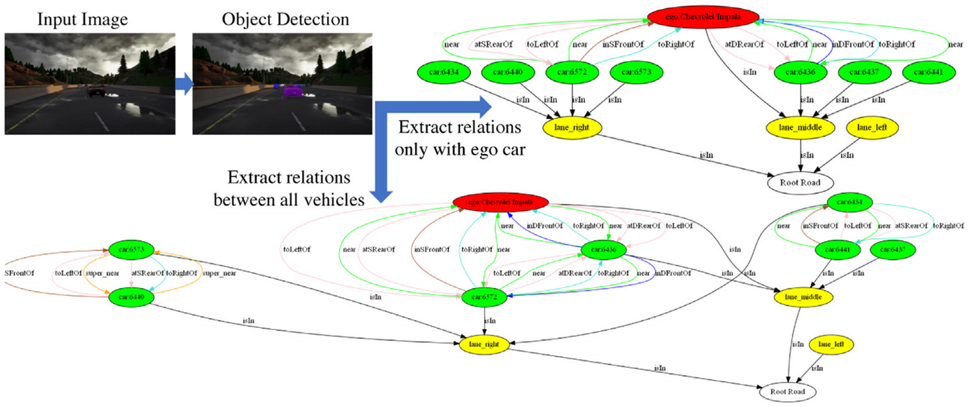
Input Image
Object Detection
Extract relations only with ego car
Extract relations between all vehicles

Fig. 7. Demonstration of *scene-graph* extraction with two different relation extraction settings. Zoom in for details.

and direction relations are constructed. The graph at the top left is similar. However, it only contains relations between the ego vehicle and each other vehicle. This figure shows one example of how our tool enables flexible graph construction for different applications. A demonstration of our visualizer tool is shown in Fig. 8. As shown, our visualizer allows the user to inspect how objects detected in the input image translate to the objects and relations in the *scene-graph*.

5.4. Use Case 2 evaluation: Subjective risk assessment

Here, we demonstrate how ROADSCENE2VEC can be used to train and evaluate several models for the subjective risk assessment use case. We used classification accuracy and the Area Under the Curve (AUC) [49] of the Receiver Operating Characteristic (ROC) to score the models. AUC, sometimes referred to as a balanced accuracy measure [50], measures the probability that a binary classifier ranks a positive sample more highly than a random negative sample. This is a more balanced measure for measuring accuracy, especially with imbalanced datasets (i.e., *271-syn*, *1043-syn*, *571-honda*).

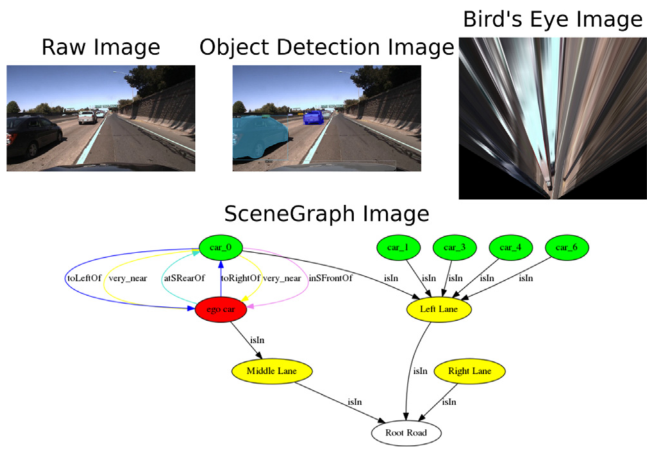

Fig. 8. A demonstration of our *scene-graph* visualization tool that enables the user to inspect: (i) an original input image, (ii) the object detection results, (iii) the birds-eye view projection of the image, and (iv) the resultant *scene-graph*.

Table 2
Risk assessment result for MRGCN, MRGIN, ResNet-50, and CNN+LSTM.

Metric	Dataset	MRGCN	MRGIN	ResNet-50	CNN+LSTM [6]
Accuracy	271-syn	**0.9320**	0.8561	0.6938	0.8033
	1043-syn	**0.9580**	0.8784	0.9053	0.7742
	571-honda	**0.8710**	0.8310	0.7689	0.6041
	1361-honda	**0.8655**	0.7245	0.6839	0.7158
AUC	271-syn	**0.9620**	0.9437	0.7371	0.8394
	1043-syn	**0.9780**	0.9591	0.9616	0.8221
	571-honda	**0.9105**	0.8903	0.8343	0.6670
	1361-honda	**0.9124**	0.8164	0.7340	0.7560

Table 3
Collision prediction accuracy, AUC, and MCC for different models in ROADSCENE2VEC.

Metric	Dataset	MRGCN	MRGIN	ResNet-50	CNN+LSTM [6]
Accuracy	271-syn	**0.8812**	0.8028	0.7039	0.7184
	1043-syn	**0.9095**	0.7803	0.8080	0.8029
	571-honda	0.6922	0.7230	**0.7340**	0.5606
AUC	271-syn	**0.9457**	0.8724	0.7564	0.7607
	1043-syn	**0.9477**	0.8826	0.9026	0.8493
	571-honda	0.7775	**0.7844**	0.7802	0.5871
MCC	271-syn	**0.5145**	0.3046	0.3320	0.1474
	1043-syn	**0.5385**	0.2852	0.4602	0.2436
	571-honda	0.2142	0.1908	**0.3547**	0.1347

Table 2 shows a comparison between MRGCN, MRGIN, ResNet-50, and CNN+LSTM [6] models for driving scene risk assessment. The results show that the MRGCN based approach consistently outperforms the other models across all the datasets in terms of both classification accuracy and AUC. We found that the performance difference between the *scene-graph* based approaches and the CNN-based approaches increased when the training datasets were smaller, indicating that the graph-based methods could likely learn a good representation with fewer data.

5.5. Use Case 3 evaluation: Collision prediction

Next, we evaluated the models in ROADSCENE2VEC at collision prediction using classification accuracy, AUC, and Matthews Correlation Coefficient (MCC) [51]. MCC is considered a balanced performance measure for binary classification, even on datasets with significant class imbalances. The MCC score outputs a value between −1.0 and 1.0, where 1.0 corresponds to a perfect classifier, 0.0 to a random classifier, and −1.0 to an always incorrect classifier. The results from our evaluation are shown in Table 3.

Once again, MRGCN outperforms the other models on the synthetic datasets. However, on the 571-honda dataset, the ResNet-50 model outperforms MRGCN across all metrics. Upon deeper inspection of the results, we found that the ResNet-50 model had a higher FNR than the MRGCN and a lower FPR than the MRGCN, suggesting that the ResNet-50 model is less sensitive than the MRGCN. Given that collision prediction is a safety-critical application, this behavior may not necessarily be desirable; however, decision boundary tuning could be used to fine-tune the sensitivity for the final application's requirements.

On both Use Case 2 and 3, MRGIN underperforms MRGCN, likely because MRGCN is a more general framework while MRGIN is designed to perform well at graph topology analysis problems, such as graph isomorphism testing. MRGIN may outperform MRGCN on different problem formulations or graph construction formulations if they play to these strengths of MRGIN.

5.6. Use Case 4 evaluation: Transfer learning

Here, we demonstrate how ROADSCENE2VEC can be used to evaluate each model's ability to transfer the knowledge learned from simulated datasets to real-world datasets. As part of this use case, ROADSCENE2VEC uses the model weights and parameters learned from training on the simulated dataset (*271-syn* or *1043-syn* in this case) directly for testing on the real-world driving dataset (*571-honda*) with no domain adaptation steps. We show the results of this evaluation for the MRGCN, ResNet-50, and CNN+LSTM models in Table 4.

Table 4
The results of comparing transferability between MRGCN, ResNet-50, and CNN+LSTM [6]. In this experiment, we trained each model on both the *271-syn* dataset and *1043-syn* dataset. Then we evaluated the accuracy of the trained model on both original dataset and *571-honda* dataset without any domain adaptation.

Experiment	Model	Original Acc.	Transfer Acc.
271-syn to *571-honda*	ResNet-50	0.7039	0.1899 (−0.514)
	CNN+LSTM [6]	0.8033	0.5244 (−0.279)
	MRGCN	**0.9040**	**0.8690 (−0.035)**
1043-syn to *571-honda*	ResNet-50	0.8080	0.1725 (−0.636)
	CNN+LSTM [6]	0.7742	0.6010 (−0.173)
	MRGCN	**0.9520**	**0.8870 (−0.065)**

As expected, the performance of all models degrades when tested on *571-honda* dataset. However, as Table 4 shows, the accuracy of the MRGCN only drops by 3.5% and 6.5% when the model is trained on *271-syn* and *1043-syn*, respectively, while the CNN+LSTM's performance drops by 27.9% and 17.3%, respectively. Furthermore, the MRGCN achieves a higher accuracy score than the CNN+LSTM when transferring from the smaller *271-syn* dataset, once again indicating that *scene-graph* models can better model the problem even when trained on smaller amounts of data. The ResNet-50 model performs worst and classifies most of the sequences as risky, resulting in an accuracy score nearly equivalent to the proportion of risky sequences in the *571-honda* dataset (17.25%). These results suggest that the *scene-graph* models can transfer knowledge more effectively than the CNN-based models.

5.7. Use Case 5 evaluation: Explainability analysis

To demonstrate ROADSCENE2VEC's tools for evaluating explainability, we run our explainability analysis tool on the MRGCN model trained for risk assessment on the 271-syn dataset. The result from analyzing one of the sequences from the dataset is shown in Fig. 9. As shown, the attention scores are highest on the nodes which present the highest degree of risk. Additionally, the graph with the highest attention score for the other vehicle is also the graph corresponding to the collision with the other vehicle.

6. Discussion

6.1. Practicality

Although ROADSCENE2VEC is intended to be a tool that benefits the research community, its practicality and carryover to real-world applications are equally important. As shown with Use Case 4, ROADSCENE2VEC enables researchers to directly evaluate the ability of models trained on synthetic data to transfer their knowledge to real-world driving scenes. Many research papers often overlook this critical problem, leading to a disconnect between simulated trials and real-world performance. Our tool better enables the study of this crucial problem area and allows researchers to analyze the real-world practicality of various graph-based methodologies. Furthermore, we show that ROADSCENE2VEC is directly compatible with both the real-world Honda driving dataset [31] as well as the popular open-source driving simulator, CARLA [17], making our tool useful for a wide range of AV applications.

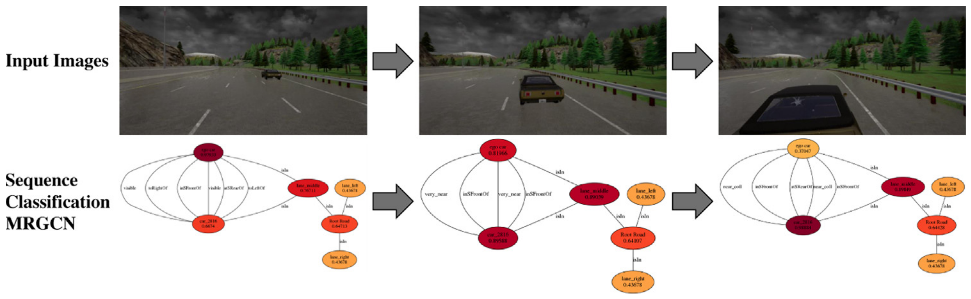

Fig. 9. A demonstration of how Use Case 5 enables *explainability* analysis. For this driving sequence, it can be clearly seen how the node attention scores shift to give higher weight to the approaching vehicle as its distance to the ego car reduces.

6.2. Limitations and future work

Although ROADSCENE2VEC provides a suite of tools for training and evaluating both scene-graph-based and CNN-based models, there are some limitations to its capabilities. For example, ROADSCENE2VEC currently only supports input data in the format of ground-truth data from the CARLA simulator or image data from a forward-facing camera; it currently does not support radar, lidar, or multi-camera data. We selected image data and CARLA data as the primary input modalities because these data types are the ones most used by AV researchers currently. Although radar and lidar data are valuable and well-studied in specific applications such as localization and sensor fusion, most AV research papers exploring perception and control methodologies use camera-based inputs. However, this limitation can be overcome by implementing preprocessors for extracting (or fusing) *scene-graphs* from these different modalities. Thus, ROADSCENE2VEC does not currently support multiple sensing modalities but could support them as part of future work. Furthermore, our tool does not implement more than a few common perception algorithms and use cases. However, our tool is designed to be modular and re-configurable to support custom models and problem formulations. We expect that researchers will develop custom architectures and models for the various well-studied problems in the AV domain and provide instructions in our repository for integrating the custom models with ROADSCENE2VEC's workflow. Thus, we leave the study of other AV applications and model architectures as future work. We also welcome outside contributions to our open-source tool to further improve its utility for the research community.

7. Conclusion

It is clear from current research as well as the examples shown in this paper that *scene-graph* representations of road scenes can be beneficial for a wide range of AV applications. In this paper, we introduced and demonstrated our tool for exploring and studying the applications of road *scene-graphs*, named ROADSCENE2VEC. We showed that our re-configurable graph-construction methodology enables the study of different graph layouts for various problems. We also demonstrated performance evaluations for conventional CNN architectures and graph-based models for two common AV perception use cases: risk assessment and collision prediction. Furthermore, we showed how our tool facilitates studying the transferability and explainability of graph-based AV models for both synthetic and real-world data. We believe our open-source tool fills a significant gap in the research community and will enable a deeper study of the applicability and practicality of graph-based solutions for AV problems.

CRediT authorship contribution statement

Arnav Vaibhav Malawade: Conceptualization, Methodology, Software, Writing – original draft, Investigation, Visualization. **Shih-Yuan Yu:** Conceptualization, Methodology, Software, Writing – original draft, Investigation, Visualization. **Brandon Hsu:** Software, Data curation, Validation, Visualization, Investigation. **Harsimrat Kaeley:** Software, Data curation, Validation, Visualization, Investigation. **Anurag Karra:** Software, Data curation, Validation, Visualization, Investigation. **Mohammad Abdullah Al Faruque:** Supervision, Writing – review & editing, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the National Science Foundation (NSF), USA under award CMMI-1739503 and by Graduate Assistance in Areas of National Need (GAANN), USA under award P200A180052. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency.

References

[1] Todd Litman, Autonomous Vehicle Implementation Predictions, Victoria Transport Policy Institute Victoria, Canada, 2017.
[2] National Transportation Safety Board, Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian, Technical Report NTSB/HAR-19/03, National Transportation Safety Board, 2019.
[3] National Transportation Safety Board, Collision Between a Sport Utility Vehicle Operating With Partial Driving Automation and a Crash Attenuator, Technical Report NTSB/HAR-20/01, National Transportation Safety Board, 2020.
[4] National Transportation Safety Board, Collision Between Car Operating with Partial Driving Automation and Truck-Tractor Semitrailer, Technical Report NTSB/HAB-20/01, National Transportation Safety Board, 2020.
[5] Alexandra S. Mueller, Jessica B. Cicchino, David S. Zuby, What humanlike errors do autonomous vehicles need to avoid to maximize safety? J. Saf. Res. (2020).
[6] Ekim Yurtsever, Yongkang Liu, Jacob Lambert, Chiyomi Miyajima, Eijiro Takeuchi, Kazuya Takeda, John H.L. Hansen, Risky action recognition in lane change video clips using deep spatiotemporal networks with segmentation mask transfer, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, IEEE, 2019, pp. 3100–3107.
[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al., End to end learning for self-driving cars, 2016, arXiv preprint arXiv:1604.07316.
[8] Chongben Tao, Haotian He, Fenglei Xu, Jiecheng Cao, Stereo priori RCNN based car detection on point level for autonomous driving, Knowl.-Based Syst. 229 (2021) 107346.

[9] Degui Xiao, Xuefeng Yang, Jianfang Li, Merabtene Islam, Attention deep neural network for lane marking detection, Knowl.-Based Syst. 194 (2020) 105584.

[10] Sebastian Sontges, Markus Koschi, Matthias Althoff, Worst-case analysis of the time-to-react using reachable sets, in: 2018 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2018, pp. 1891–1897.

[11] David Nistér, Hon-Leung Lee, Julia Ng, Yizhou Wang, The safety force field, 2019, NVIDIA White Paper.

[12] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al., Relational inductive biases, deep learning, and graph networks, 2018, arXiv preprint arXiv:1806.01261.

[13] Shih-Yuan Yu, Arnav Vaibhav Malawade, Deepan Muthirayan, Pramod P. Khargonekar, Mohammad Abdullah Al Faruque, Scene-graph augmented data-driven risk assessment of autonomous vehicle decisions, IEEE Trans. Intell. Transp. Syst. (2021).

[14] Sravan Mylavarapu, Mahtab Sandhu, Priyesh Vijayan, K. Madhava Krishna, Balaraman Ravindran, Anoop Namboodiri, Towards accurate vehicle behaviour classification with multi-relational graph convolutional networks, 2020, arXiv preprint arXiv:2002.00786.

[15] Chengxi Li, Yue Meng, Stanley H Chan, Yi-Ting Chen, Learning 3D-aware egocentric spatial-temporal interaction via graph convolutional networks, 2019, arXiv preprint arXiv:1909.09272.

[16] Arnav V. Malawade, Shih-Yuan Yu, Brandon Hsu, Deepan Muthirayan, Pramod P. Khargonekar, Mohammad A. Al Faruque, Spatio-temporal scene-graph embedding for autonomous vehicle collision prediction, IEEE Internet of Things Journal (2022) http://dx.doi.org/10.1109/JIOT.2022.3141044, 1-1.

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, Vladlen Koltun, CARLA: An open urban driving simulator, 2017, arXiv preprint arXiv:1711.03938.

[18] Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, Bo Li, et al., Invisible for both camera and LiDAR: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks, 2021, arXiv preprint arXiv:2106.09249.

[19] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, Kazuya Takeda, A survey of autonomous driving: Common practices and emerging technologies, 2019, arXiv preprint arXiv:1906.05113.

[20] Chenyi Chen, Ari Seff, Alain Kornhauser, Jianxiong Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2722–2730.

[21] Mayank Bansal, Alex Krizhevsky, Abhijit Ogale, Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst, 2018, arXiv preprint arXiv:1812.03079.

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, Mask r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.

[23] Yafu Tian, Alexander Carballo, Ruifeng Li, Kazuya Takeda, Road scene graph: A semantic graph-based scene representation dataset for intelligent vehicles, 2020, arXiv preprint arXiv:2011.13588.

[24] Lars Kunze, Tom Bruls, Tarlan Suleymanov, Paul Newman, Reading between the lanes: Road layout reconstruction from partially segmented scenes, in: 2018 21st International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2018, pp. 401–408.

[25] Xiaoming Liu, Yu Lan, Yadong Zhou, Chao Shen, Xiaohong Guan, A real-time explainable traffic collision inference framework based on probabilistic graph theory, Knowl.-Based Syst. 212 (2021) 106442.

[26] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, Devi Parikh, Graph r-cnn for scene graph generation, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 670–685.

[27] Kaihua Tang, A scene graph generation codebase in PyTorch, 2020, https://github.com/KaihuaTang/Scene-Graph-Benchmark.pytorch.

[28] Jiaxuan You, Rex Ying, Jure Leskovec, Design space for graph neural networks, in: NeurIPS, 2020.

[29] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, et al., Deep graph library: Towards efficient and scalable deep learning on graphs, 2019.

[30] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, Jure Leskovec, Open graph benchmark: Datasets for machine learning on graphs, 2020, arXiv preprint arXiv:2005.00687.

[31] Vasili Ramanishka, Yi-Ting Chen, Teruhisa Misu, Kate Saenko, Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning, in: Conference on Computer Vision and Pattern Recognition, 2018.

[32] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, D. Hoff, D. Kruchinin, A. Zankevich, D. Sidnev, Opencv/CVAT: v1. 1.0, 2020, Zenodo, Aug.

[33] Microsoft, Vott, 2021, GitHub Online URL https://github.com/microsoft/VoTT. (Accessed 29 December 2021).

[34] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.

[35] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka, How powerful are graph neural networks? 2018, arXiv preprint arXiv:1810.00826.

[36] Junhyun Lee, Inyeop Lee, Jaewoo Kang, Self-attention graph pooling, 2019, arXiv preprint arXiv:1904.08082.

[37] Hongyang Gao, Shuiwang Ji, Graph u-nets, 2019, arXiv preprint arXiv:1905.05178.

[38] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv preprint arXiv:1409.0473.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition, 2016, pp. 770–778.

[40] G.B. Grayson, G. Maycock, J.A. Groeger, S.M. Hammond, D.T. Field, Risk, Hazard Perception and Perceived Control, TRL Report TRL560, TRL Ltd., Crowthorne, UK, 2003.

[41] Ray Fuller, Towards a general theory of driver behaviour, Accid. Anal. Prev. 37 (3) (2005) 461–472.

[42] Naren Bao, Dongfang Yang, Alexander Carballo, Ümit Özgüner, Kazuya Takeda, Personalized safety-focused control by minimizing subjective risk, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, IEEE, 2019, pp. 3853–3858.

[43] Naren Bao, Alexander Carballo, Chiyomi Miyajima, Eijiro Takeuchi, Kazuya Takeda, Personalized subjective driving risk: Analysis and prediction, J. Robot. Mechatronics 32 (3) (2020) 503–519.

[44] Ulrich Tränkle, Christhard Gelau, Thomas Metker, Risk perception and age-specific accidents of young drivers, Accid. Anal. Prev. 22 (2) (1990) 119–125.

[45] Brandon Schoettle, Michael Sivak, A Preliminary Analysis of Real-World Crashes Involving Self-Driving Vehicles, University of Michigan Transportation Research Institute, 2015.

[46] Chengcheng Xu, Zijian Ding, Chen Wang, Zhibin Li, Statistical analysis of the patterns and characteristics of connected and autonomous vehicle involved crashes, J. Saf. Res. 71 (2019) 41–47.

[47] Amina Adadi, Mohammed Berrada, Peeking inside the black-box: A survey on explainable artificial intelligence (XAI), IEEE Access 6 (2018) 52138–52160.

[48] Boris Knyazev, Graham W Taylor, Mohamed Amer, Understanding attention and generalization in graph neural networks, in: Advances in Neural Information Processing Systems, 2019, pp. 4202–4212.

[49] Andrew P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (7) (1997) 1145–1159.

[50] Marina Sokolova, Guy Lapalme, A systematic analysis of performance measures for classification tasks, Inf. Process. Manage. 45 (4) (2009) 427–437.

[51] Davide Chicco, Giuseppe Jurman, The advantages of the matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, BMC Genomics 21 (1) (2020) 6.