Scene-Graph Embedding for Robust Autonomous Vehicle Perception

Shih-Yuan Yu, Arnav Vaibhav Malawade, and Mohammad Abdullah Al Faruque

1 Introduction

Automotive CPS, also called *Autonomous Vehicles* (AV), aims to revolutionize personal mobility, logistics, and road safety [17]. However, accidents involving perception errors in modern self-driving cars are still a regular occurrence, highlighting that the development of safe and robust AVs remains a difficult challenge [26–28]. What is worse is that these perception errors often seem completely irrational from our perspective. In one crash, a self-driving car failed to perceive a semi-truck that was utterly obstructing the highway [28]. Another crash involved a vehicle steering directly into the freeway divider in broad daylight [27]. These events cast serious doubt on the ability of current AV perception systems to understand the state of the road. According to a statistic [24], perception and prediction errors were the primary factors in over 40% of driver-related crashes between conventional vehicles. In complex urban environments, navigation is particularly challenging because the scenarios are highly variable and involve pedestrians and bicyclists, heavy traffic, blind driveways, blocked roadways, etc. [23, 30, 46]. Within this context, the effectiveness of understanding the driving scenes becomes particularly crucial, leading researchers and industry leaders to race to address these problems via more advanced AV perception systems.

One might ask, how are humans able to perceive the state of the road effectively without succumbing to the common mistakes of AV perception systems? Recent research suggests that humans rely on cognitive mechanisms to identify the structure of a scene and reason about inter-object relations when performing complex tasks such as identifying risk during driving [5]. However, existing AV perception

S.-Y. Yu · A. V. Malawade (✉) · M. A. Al Faruque
The University of California, Irvine, Irvine, CA, USA
e-mail: shihyuay@uci.edu; malawada@uci.edu; alfaruqu@uci.edu

V. K. Kukkala, S. Pasricha (eds.), *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, https://doi.org/10.1007/978-3-031-28016-0_18

architectures use road geometry information and vehicle trajectory models for estimating the state of the road scene (model-based methods) [29, 37]. More recently, architectures using deep learning techniques that leverage *Convolutional Neural Networks* (CNNs), *Long-Short Term Memory Networks* (LSTMs), or *Multi-Layer Perceptrons* (MLPs) [6, 19, 20, 38, 41, 47] have proven effective at capturing features essential for modeling subjective risk in both spatial and temporal domains [47]. However, these approaches cannot obtain a high-level, human-like understanding of complex road scenarios due to their inability to explicitly capture inter-object relationships or the overall structure of the road scene. Failing to capture these relationships can result in poor perception performance in complex scenarios. Overall, designing a robust perception system for automotive CPSs using data-driven approaches poses the following challenges:

1. Designing a reliable method that can handle a wide range of complex and unpredictable traffic scenarios,
2. Building a model that is transferable from the simulation setting to the real-world setting because the real-world datasets for supervised training are limited,
3. Building a model that can provide explainable decisions.

Take risk assessment tasks as an example. To overcome the first challenge, deep learning-based methods must be trained on large datasets covering a wide range of "corner cases" (especially risky driving scenarios), which are expensive and time-consuming to generate [9]. In this case, many researchers resort to using synthesized datasets containing many examples of these corner cases to address this issue. However, as mentioned in the second challenge, for these to be valuable, a model must be able to transfer the knowledge gained from simulated training data to real-world situations. A standard method for measuring a model's ability to generalize is *transferability*, where a model's accuracy on a dataset different from the training dataset is evaluated. Suppose a model can effectively transfer the knowledge gained from a simulated training set to a real-world testing set. In that case, it will likely perform better in unseen real-world scenarios. Even if these existing methods can transfer knowledge well, the predictions of such methods lack *explainability*, which is crucial for establishing trust between ADSs and human drivers [1, 2, 4]. In the third research challenge, *Explainability* refers to the ability of a model to effectively communicate the factors that influenced its decision-making process for a given input, particularly those that might lead the model to make incorrect decisions [1, 13]. Suppose a model can give attention to the aspects or entities in a traffic scene that make the scenario risky or non-risky. In that case, it can improve its decision, and its decisions become more explainable [39].

2 Scene-Graph Representation of Road Scenes

2.1 ADS Design Philosophies and Intermediate Representation

Many design philosophies for ADS have been proposed over the years, such as the *modular* design and the *end-to-end* design. Most *modular* design approaches comprise a pipeline of separate components from the sensory inputs to the actuator outputs. In contrast, *end-to-end* approaches generate output directly from their sensory inputs [6, 31]. 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 [14], computer vision [12, 19, 20] and vehicle dynamics [32]. As a result, prior knowledge from these fields can be leveraged when designing the components corresponding to the sub-tasks. However, one disadvantage of such an approach is the complexity of the whole pipeline [46]. *End-to-end* design approaches can achieve good performance with a smaller network size because they perform feature extraction from sensor inputs implicitly through the network's hidden layers [6, 18]. However, the authors in [8] point out that the needed level of supervision is too weak for the end-to-end model to learn critical controlling information (e.g., from image to steering angle), so it can fail to handle complicated driving maneuvers.

Recently, few methodologies have leveraged the benefits of an *intermediate representation* (IR). *DeepDriving* [8], called the *direct perception*, was one of the first approaches to use an IR methodology. In their methodology, a set of *affordance* indicators, such as the distance to lane markings and cars in the current and adjacent lanes, are extracted from an image and serve as an IR for generating the final control output. The authors of [8] prove that the use of this IR is effective for simple driving tasks such as lane following and for generalizing the learned knowledge from simulation to real-world environments, thus improving *transferability*. Authors in [3] use a collection of filtered images, each representing a piece of distinct information, as the IR. They state that the IR used in their methodology allows the training to be conducted on 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 at the mid-level representations than at the level of raw sensors, enabling them to produce non-expert behaviors such as off-road driving and collisions. As such, the capability to capture and identify the complex relationships between road objects is critical in designing an effective human-like perception system for automotive CPS.

2.2 Graph-Based Driving Scene Understanding

In literature, several groups have adopted a variant of *Knowledge Graphs* known as *scene-graphs* to model the road state and the relationships between objects [16,

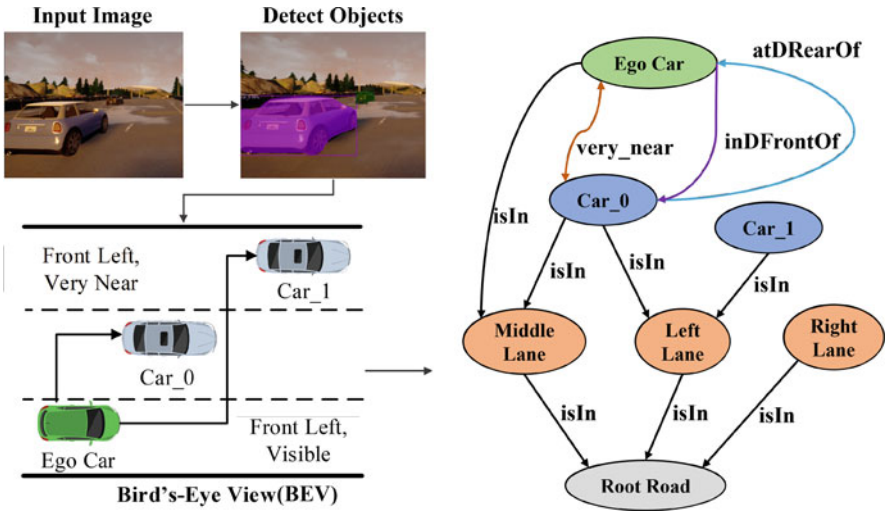

Fig. 1 How camera data can be used to construct a road *scene-graph* representation

21, 22, 25, 45]. 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 [16], 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 [25], 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 objects and landmark objects. Our prior work has demonstrated that the use of spatio-temporal *scene-graph* embeddings improves performance at subjective risk assessment and collision prediction versus state-of-the-art methods [21, 22, 45]. In addition, our method can better transfer knowledge and is more explainable.

2.3 Scene-Graph Extraction from Driving Scenes

In literature, several approaches have been proposed for extracting *scene-graphs* from images by detecting the objects in a scene and then identifying their visual relationships [42, 44]. However, these works focus on extracting *scene-graphs* for single general images for tasks like automated image captioning instead of modeling

these graphs to maximize performance over a temporally-correlated sequence of images as are typically used for autonomous driving. Thus, we adopted a partially rule-based process to extract objects and their attributes from images. Object attributes and bounding boxes are extracted directly from images using state-of-the-art image processing techniques. As Fig. 1 shows, we first convert each image I_t into a collection of objects O_t using Faster RCNN [34], a state of the art object detection algorithm in the *Detectron2* [40] computer vision library. Next, we use OpenCV's perspective transformation library to generate a top-down perspective of the image, commonly known as a "birds-eye view" projection [7]. This projection lets us approximate each object's location relative to the road markings and the ego vehicle. Next, for each detected object in O_t, we use its estimated location and class type (cars, motorcycles, pedestrians, lanes, etc.) to compute the attributes required in building the *scene-graph*.

After collecting the list of objects in each image and their attributes, we can begin constructing the corresponding *scene-graphs*. For each image I_t, we denote the corresponding *scene-graph* by $G_t = \{O_t, A_t\}$ and model it as a directed multi-graph where multiple types of edges connect nodes. The nodes of a *scene-graph*, denoted as O_t, represent the objects in a scene such as lanes, roads, traffic signs, vehicles, pedestrians, etc. The edges of G_t are represented by the adjacency matrix A_t, where each value in A_t represents the type of the corresponding edge in G_t. The edges between two nodes represent the different kinds of relations between them (e.g., near, Front_Left, isIn, etc.). For assessing the risk of driving behaviors, we consider both distance and directional relations between traffic participants useful. We assume that one object's local proximity and positional information will influence the other's motion only if they are within a certain distance. Therefore, in this work, we extract only the location information for each object and adopt a simple rule to determine the relations between the objects using their attributes (e.g., relative location to the ego car), as shown in Fig. 1. For distance relations, we assume two objects are related by one of the relations $r \in \{$*Near_Collision* (4 ft.), *Super_Near* (7 ft.), *Very_Near* (10 ft.), *Near* (16 ft.), *Visible* (25 ft.)$\}$ if the objects are physically separated by a distance that is within that relation's threshold. In the case of the directional relations, we assume two objects are related by the relation $r \in \{$*Front_Left*, *Left_Front*, *Left_Rear*, *Rear_Left*, *Rear_Right*, *Right_Rear*, *Right_Front*, *Front_Right*$\}$ based on their relative positions if they are within the *Near* threshold distance from one another.

In addition to directional and distance relations, we also implement the *isIn* relation that connects vehicles with their respective lanes. Specifically, we use each vehicle's horizontal displacement relative to the ego vehicle to assign cars to either the *Left Lane*, *Middle Lane*, or *Right Lane* based on known lane width. Our abstraction only includes these three-lane areas, and, as such, we map vehicles in all left lanes to the same *Left Lane* node and all vehicles in right lanes to the *Right Lane* node. If a vehicle overlaps two lanes (i.e., during a lane change), we assign it an *isIn* relation to both lanes. Figure 1 illustrates an example of resultant *scene-graph*.

3 Spatio-Temporal Scene-Graph Embedding Approach for Robust Automotive CPS Perception

To tackle the research challenges, we propose a *scene-graph* augmented data-driven approach for assessing the subjective risk of driving maneuvers, where the *scene-graphs* serve as intermediate representations (IR) as shown in Fig. 1. The key advantage of using *scene-graph* as IR is that they allow us to model the relationships between the participants in a traffic scene, thus potentially improving the model's understanding of a scene. Our proposed architecture consists of three major components: (1) a pipeline to convert the images of a driving clip to a sequence of *scene-graphs*, (2) a Multi-Relational Graph Convolution Network (MR-GCN) to convert each of the *scene-graphs* to an embedding (a vectorized representation), and (3) an LSTM for temporally modeling the sequence of embeddings of the respective *scene-graphs*. Our model also contains multiple attention layers: (1) a node attention layer before the embedding of a *scene-graph* is computed, and (2) an attention layer on top of the LSTM, both of which can further improve its performance and explainability.

3.1 Problem Formulation

For training the model, we formulate the problem of subjective risk assessment as a supervised *scene-graph* sequence classification problem. Our approach makes the same assumption used in [47] 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 $\mathbf{I} = \{I_1, I_2, I_3, \ldots, I_T\}$. 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).

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

where

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

The goal of our approach is to propose a suitable model for approximating the function f. Here, the first step is the extraction of the *scene-graph* G_t from each image I_t of the video clip $\mathbf{I}$. This step is achieved by a series of processes that we collectively call the *Scene-Graph Extraction Pipeline* (described in Sect. 2.3). In the second step, these *scene-graphs* are passed through graph convolution layers and an attention-based graph pooling layer. The graph-level embeddings of each *scene-graph*, $\mathbf{h}_{G_t}$, are then calculated using a graph readout operation. Next, these

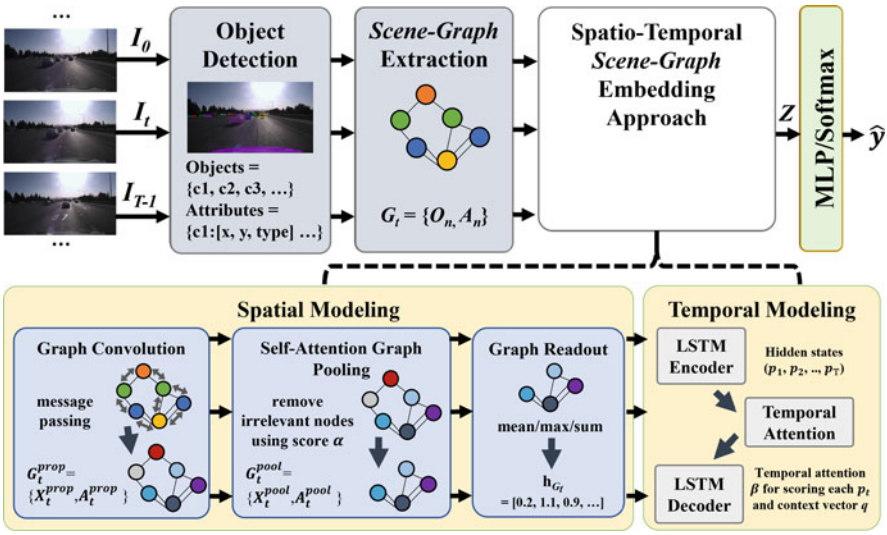

Fig. 2 An illustration of spatio-temporal *scene-graph* embedding approach

scene-graph embeddings are passed sequentially to LSTM cells to acquire the *spatio-temporal* representation, denoted as $\mathbf{Z}$, of each *scene-graph* sequence. Lastly, we use a Multi-Layer Perceptron (MLP) layer with a *Softmax* activation function to acquire the final inference, denoted as $\hat{y}$, of the risk for each driving sequence $\mathbf{I}$.

To sum up, the model of our approach consists of three major components: a spatial model, a temporal model, and a risk inference component. The spatial model outputs the embedding h_{G_t} for each scene-graph G_t. The temporal model processes the sequence of *scene-graph* embeddings $\mathbf{h}_I = \{h_{G_1}, h_{G_2}, \ldots, h_{G_T}\}$ and produces the spatio-temporal embedding $\mathbf{Z}$. The risk inference component outputs each driving clip's final risk assessment, denoted as $\hat{y}$, by processing the Spatio-temporal embedding $\mathbf{Z}$. The overall network architecture is shown in Fig. 2. We discuss each of these components in detail below.

3.2 Spatial Modeling

The spatial model we propose uses MR-GCN layers to compute the embedding for a *scene-graph*. The use of MR-GCN allows us to capture multiple types of relations on each *scene-graph* $G_t = \{O_t, A_t\}$. In the *Message Propagation* phase, a collection of node embeddings and their adjacency information serve as the inputs to the MR-GCN layer. Specifically, the l-th MR-GCN layer updates the node embedding, denoted as $\mathbf{h}_v^{(l)}$, for each node v as follows:

$$\mathbf{h}_v^{(l)} = \mathbf{\Phi}_0 \cdot \mathbf{h}_v^{(l-1)} + \sum_{r \in \mathbf{A_t}} \sum_{u \in \mathbf{N}_r(v)} \frac{1}{|\mathbf{N}_r(v)|} \mathbf{\Phi}_r \cdot \mathbf{h}_u^{(l-1)}, \tag{3}$$

where $N_r(v)$ denotes the set of neighbor indices of node v with the relation $r \in A_t$. $\mathbf{\Phi}_r$ is a trainable relation-specific transformation for relation r in MR-GCN layer. Since the information in $(l-1)$-th layer can directly influence the representation of the node at l-th layer, MR-GCN uses another trainable transformation $\mathbf{\Phi}_0$ to account for the self-connection of each node using a special relation [35]. Here, we initialize each node embedding $\mathbf{h}_v^{(0)}$, $\forall v \in O_t$, by directly converting the node's type information to its corresponding one-hot vector.

Typically, the node embedding becomes more refined and global as the number of graph convolutional layers, L, increases. However, the authors in [43] also suggest that the features generated in earlier iterations might generalize the learning better. Therefore, we consider the node embeddings generated from all the MR-GCN layers. To be more specific, we calculate the embedding of node v at the final layer, denoted as $\mathbf{H}_v^L$, by concatenating the features generated from all the MR-GCN layers, as follows,

$$\mathbf{H}_v^L = \mathbf{CONCAT}(\{\mathbf{h}_v^{(l)}\} | l = 0, 1, \ldots, L). \tag{4}$$

We denote the collection of node embeddings of *scene-graph* G_t after passing through L layers of MR-GCN as $\mathbf{X}_t^{prop}$ (L can be 1, 2 or 3).

The node embedding $\mathbf{X}_t^{prop}$ is further processed with an attention-based graph pooling layer. As stated in [13], such an attention-based pooling layer can improve the explainability of predictions and is typically considered a part of a unified computational block of a graph neural network (GNN) pipeline. In this layer, nodes are pooled according to the scores predicted from either a trainable simple linear projection [10] or a separate trainable GNN layer [15]. We denote the graph pooling layer that uses the **SCORE** function in [10] as *TopkPool* and the one that uses the **SCORE** function in [15] as *SAGPool*. The calculation of the overall process is presented as follows:

$$\boldsymbol{\alpha} = \mathbf{SCORE}(\mathbf{X}_t^{prop}, \mathbf{A_t}), \tag{5}$$

$$\mathbf{P} = \text{top}_k(\boldsymbol{\alpha}), \tag{6}$$

where $\boldsymbol{\alpha}$ stands for the coefficients predicted by the graph pooling layer for nodes in G_t and $\mathbf{P}$ represents the indices of the pooled nodes, which are selected from the top k of the nodes ranked according to α. The number k of the nodes to be pooled is calculated by a pre-defined pooling ratio, pr, and using $k = pr \times |O_t|$, where we consider only a constant fraction pr of the embeddings of the nodes of a scene-graph to be relevant (i.e., 0.25, 0.5, 0.75). We denote the node embeddings and edge adjacency information after pooling by $\mathbf{X}_t^{pool}$ and $\mathbf{A_t^{pool}}$ and are calculated as follows:

$$\mathbf{X}_t^{pool} = (\mathbf{X}_t^{prop} \odot \tanh(\boldsymbol{\alpha}))_{\mathbf{P}}, \tag{7}$$

$$\mathbf{A}_{\mathbf{t}}^{\mathbf{pool}} = \mathbf{A}_{\mathbf{t}}^{\mathbf{prop}}{}_{(\mathbf{P},\mathbf{P})}. \tag{8}$$

where $\odot$ represents an element-wise multiplication, $()_{\mathbf{P}}$ refers to the operation that extracts a subset of nodes based on P, and $()_{(\mathbf{P},\mathbf{P})}$ refers to the formation of the adjacency matrix between the nodes in this subset.

Finally, our model aggregates the node embeddings of the graph pooling layer, $\mathbf{X}_t^{pool}$, using a graph **READOUT** operation, to produce the final graph-level embedding $\mathbf{h}_{G_t}$ for each *scene-graph* G_t as given by

$$\mathbf{h}_{G_t} = \mathbf{READOUT}(\mathbf{X}_t^{pool}), \tag{9}$$

where the **READOUT** operation can be either summation, averaging, or selecting the maximum of each feature dimension, over all the node embeddings, known as *sum-pooling*, *mean-pooling*, or *max-pooling*, respectively. The process until this point is repeated across all images in $\mathbf{I}$ to produce the sequence of embedding, $\mathbf{h}_I$.

3.3 Temporal Modeling

The temporal model we propose uses an LSTM for converting the sequence of scene-graph embeddings $\mathbf{h}_I$ to the combined spatio-temporal embedding $\mathbf{Z}$. For each timestamp t, the LSTM updates the hidden state p_t and cell state c_t as follows,

$$p_t, c_t = \mathbf{LSTM}(\mathbf{h}_{G_t}, c_{t-1}), \tag{10}$$

where $\mathbf{h}_{G_t}$ is the final *scene-graph* embedding from timestamp t. After the LSTM processes all the scene-graph embeddings, a temporal readout operation is applied to the resultant output sequence to compute the final Spatio-temporal embedding Z given by

$$\mathbf{Z} = \mathbf{TEMPORAL\_READOUT}(p_1, p_2, \ldots, p_T) \tag{11}$$

where the **TEMPORAL_READOUT** operation could be extracting only the last hidden state p_T (LSTM-last), or be a temporal attention layer (LSTM-attn).

In [2], adding an attention layer b to the encoder-decoder based LSTM architecture is shown to achieve better performance in Neural Machine Translation (NMT) tasks. For the same reason, we include *LSTM-attn* in our architecture. *LSTM-attn* calculates a context vector q using the hidden state sequence $\{p_1, p_2, \ldots, p_T\}$ returned from the LSTM encoder layer as given by

$$q = \sum_{t=1}^{T} \beta_t p_t \tag{12}$$

where the probability β_t reflects the importance of p_t in generating q. The probability β_t is computed by a *Softmax* output of an energy function vector e, whose component e_t is the energy corresponding to p_t. Thus, the probability β_t is formally given by

$$\beta_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)}, \tag{13}$$

where the energy e_t associated with p_t is given by $e_t = b(s_0, p_t)$. The temporal attention layer b scores the importance of the hidden state p_t to the final output, which in our case is the risk assessment. The variable s_0 in the temporal attention layer b is computed from the last hidden representation p_T. The final Spatio-temporal embedding for a video clip, Z, is computed by feeding the context vector q to another LSTM decoder layer.

3.4 Risk Inference

The last piece of our model is the risk inference component that computes the risk assessment prediction $\hat{Y}$ using the spatio-temporal embedding $\mathbf{Z}$. This component is composed of a MLP layer followed by a *Softmax* activation function. Thus, the prediction $\hat{Y}$ is given by

$$\hat{Y} = Softmax(\mathbf{MLP}(Z)) \tag{14}$$

During training, the loss for the prediction is calculated as follows,

$$\mathbf{CrossEntropyLoss}(Y, \hat{Y}) \tag{15}$$

For training our model, we use a mini-batch gradient descent algorithm that updates its parameters by training on a batch of *scene-graph* sequences. To account for label imbalance, we apply class weighting when calculating loss. Besides, several dropout layers are inserted into the network to reduce overfitting.

4 Experimental Results

To illustrate the benefits of our *scene-graph* augmented approach, we present experimental results for assessing the risk of several common driving tasks, including

lane changes, turns, and merges into (merging) and out of (branching) the traffic flow. We also evaluate a state-of-the-art SMT+CNN+LSTM based risk assessment model [47] on these tasks to serve as the *baseline*. We evaluate several different aspects of performance, including risk assessment accuracy, capability to transfer knowledge from synthetic data to real-world data, and explainability. Next, let us discuss the experimental setup.

4.1 Experimental Setup

We prepare two types of datasets for the experiments (1) synthesized datasets and (2) real-world driving datasets. To create the synthesized datasets, we collected data from various driving conditions simulated in the CARLA driving simulator.[1] We generated the real-world dataset by extracting various driving actions from the Honda Driving Dataset (HDD) [33]. We generated a wide range of simulated lane changes using the various presets in CARLA that allowed us to specify the number of cars, pedestrians, weather and lighting conditions, driver behavior, etc. The lane changes that resulted in collisions, near collisions, or otherwise dangerous conditions are considered our *risky* samples, while the safe lane changes are labeled as *safe*. Common factors that can affect the risk of a driving action include the distance to other cars and the side curbs, the speed relative to other vehicles, the sizes of adjacent vehicles, the presence of bikers or pedestrians, and the traffic light status.

We generated two synthesized datasets: a *271-syn* dataset and a *1043-syn* dataset, containing 271 and 1043 lane-changing clips, respectively. In addition, we subsampled the *271-syn* and *1043-syn* datasets further to create two balanced datasets that have a 1:1 distribution of risky to safe lane changes: *96-syn* and *306-syn*. Our synthesized driving datasets are available online in both raw image and *scene-graph* format [11]. For real driving datasets, we processed the HDD dataset to create a dataset called *1361-honda* composed of 571 lane changing, 350 turning, 297 branching, and 149 merging video clips. For evaluating the capability of the model to transfer knowledge after training on the synthesized lane change datasets, we subsampled *1361-honda* to create a lane-changing dataset that contains 571 real-world lane changing clips, called *571-honda*. The final score of a model on a dataset is computed by averaging over the testing set scores for ten different train-test splits, where 30% of the dataset is reserved as the testing set.

In our experiments, we trained each model for 500 epochs. From our experimentation, we found that the best configuration of our model consisted of two MR-GCN layers with 64 hidden units, a SAGPool pooling layer with a ratio of 0.5, *sum-pooling* for graph readout operation, and *LSTM-attn* for temporal modeling.

[1] https://github.com/carla-simulator/carla.

4.2 Experiments on Risk Assessment

We evaluate each model's performance on each dataset by measuring its classification accuracy and the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC). The classification accuracy is the ratio of the number of correct predictions on the test set of a dataset to the total number of samples in the testing set. AUC, sometimes referred to as a *balanced* accuracy measure [36], quantifies the likelihood that a binary classifier ranks a positive sample more highly than a random negative sample. This metric is especially useful for imbalanced datasets (i.e. *271-syn*, *1043-syn*, *571-honda*).

Figure 3 shows the comparison between our model's performance and the baseline [47] for the synthetic datasets. The results show that our approach performs best across all datasets.

The results also show that the performance difference between our approach and the baseline increased when the training datasets were smaller. This result indicates that our approach can learn an accurate model even from a smaller dataset, likely resulting from its use of a *scene-graph* based IR. We also found that our approach performs better than the baseline on balanced datasets, meaning that our approach is better at discriminating between the two classes in general. For context, the datasets *271-syn* and *306-syn* contain roughly the same number of clips but differ in the distribution of safe to risky lane changes (2.30:1 for *271-syn* vs. 1:1 for *306-syn*).

Although these results are impressive, we must ask, how much does each component in the model contribute to the overall performance? One easy way to answer this question is with an *ablation study*, where we measure the performance of our model after adding each modeling component one at a time, as is shown in Table 1. From Table 1 we find that the simplest of the models, with no MR-GCN layer (replaced with an MLP layer) and a simple average of the embeddings in $\mathbf{h}_I$ for the temporal model (denoted as *mean* in Table 1), achieves a relatively low classification accuracy of 75%. Starting from this base model, we find that replacing *mean* with an LSTM layer for temporal modeling yields a 10.5% increase in performance. Next, we try adding a single MR-GCN layer with 64 hidden units and *sum-pooling* to the base model, resulting in a 14.8% performance gain. The performance gain achieved by including the MR-GCN layer alone demonstrates the effectiveness of explicitly modeling the relations between objects. Now, we try the single MR-GCN layer with *sum-pooling* and the LSTM model together, which yields the maximum performance gain of 18.1% over the simplest model. This result clearly illustrates that our model's spatial and temporal components are both crucial for maximizing performance.

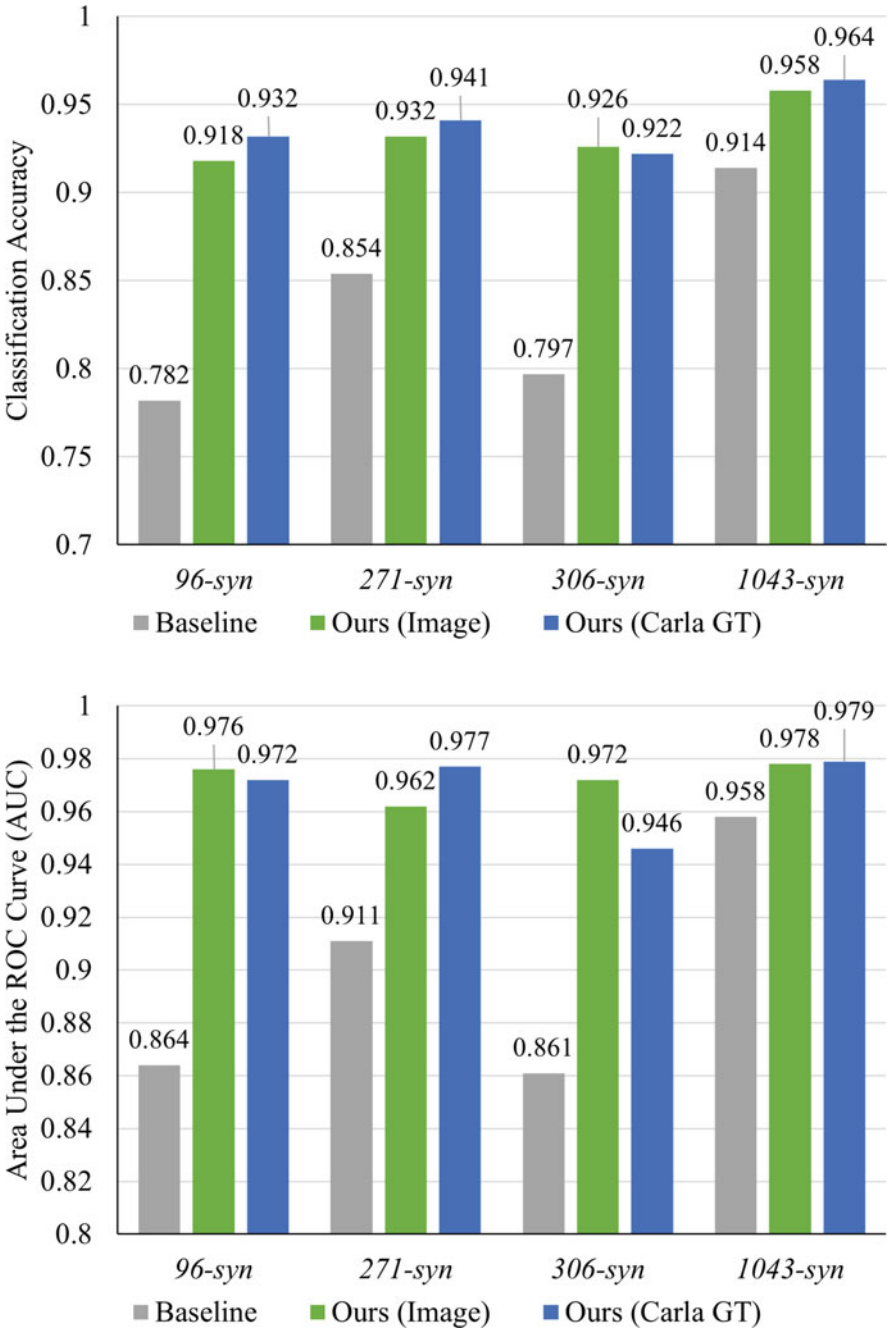

Fig. 3 Accuracy and AUC comparison between our approaches (Real Image and CARLA GT) and [47] on different datasets. Our approach outperforms the baseline across datasets

Table 1 The results of the CARLA GT approach on *1043-syn* dataset with various spatial and temporal modeling settings. In these experiments, we used MR-GCN layers with 64 hidden units and *sum-pooling* as the graph readout operation. The bolded numbers indicate the highest performing configuration in terms of Average Accuracy (Avr. Acc.) and Average AUC Score (Avr. AUC) for the grouping indicated by the leftmost column

	Spatial modeling	Temporal modeling	Avr. Acc.	Avr. AUC
Ablation study	No MR-GCN	*mean*	0.762	0.823
	No MR-GCN	*LSTM-last*	0.867	0.929
	1 MR-GCN	*mean*	0.910	0.960
	1 MR-GCN	*LSTM-last*	**0.943**	**0.977**
Temporal attention	No MR-GCN	*LSTM-last*	0.867	0.929
	No MR-GCN	*LSTM-attn*	0.868	0.928
	1 MR-GCN	*LSTM-last*	0.943	0.977
	1 MR-GCN	*LSTM-attn*	**0.950**	**0.977**
Spatial attention	1 MR-GCN	*mean*	0.910	0.960
	1 MR-GCN, *TopkPool*	*mean*	0.886	0.930
	1 MR-GCN, *SAGPool*	*mean*	**0.937**	**0.968**

4.3 Evaluation of Attention Mechanisms on Risk Assessment

Next, we evaluate the various attention components of our proposed model. To evaluate the benefit of attention over the spatial domain, we tested our model with three different graph attention methods: no attention, *SAGPool*, and *TopkPool*. To evaluate the impact of attention on the temporal domain, we tested our model with the following temporal models: *mean*, *LSTM-last*, and *LSTM-attn*. The results of this analysis are also shown in Table 1.

For evaluating the benefits of graph attention, we start with an attention-free model: one MR-GCN layer with *sum-pooling* + *mean*. In comparison, the model that uses *SAGPool* for attention on the graph shows a 2.7% performance gain over the attention-free model because using attention over both nodes and relations allows *SAGPool* to better filter out irrelevant nodes from each *scene-graph*. We found that the model using *TopkPool* as the graph-attention layer became relatively unstable, resulting in a 2.4% performance drop compared to the attention-free model. This drop is likely because *TopkPool* ignores the relations between nodes when calculating α.

For evaluating the impact of attention on the temporal model, we assessed the effects of adding a temporal attention layer to the following two models: (1) with no MR-GCN layers and no temporal attention and (2) with one MR-GCN layer and no temporal attention. Our model with no MR-GCN and no temporal attention performed nearly the same as our model with no MR-GCN and *LSTM-attn*. We also find that adding *LSTM-attn* to the model with one MR-GCN layer increases its performance by 0.7% over the same model with no temporal attention. These results demonstrate that the inclusion of temporal attention improves performance, though only marginally compared to the benefits of spatial attention. This might be because

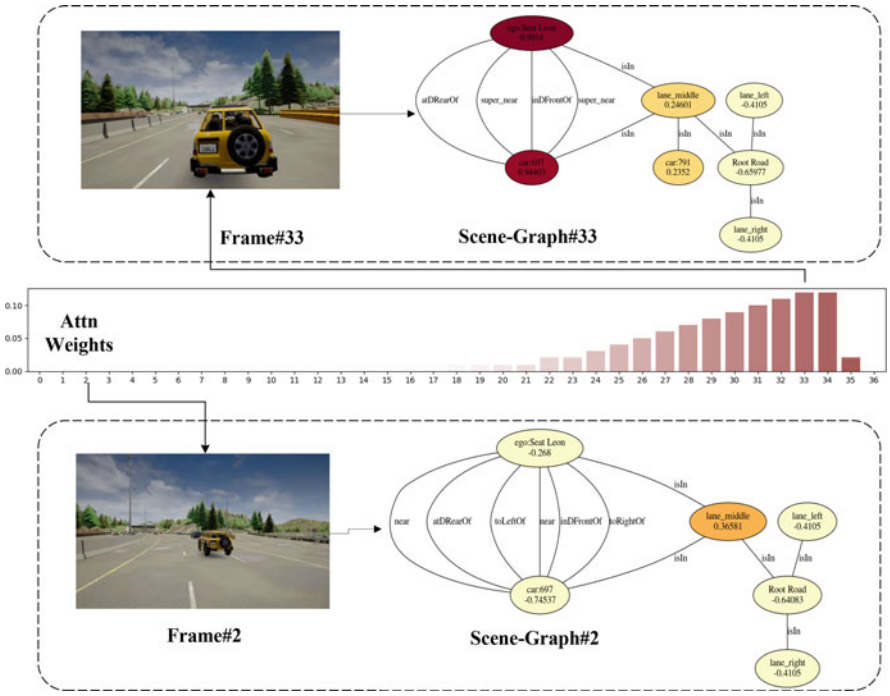

Fig. 4 The visualization of attention weights in both spatial (α) and temporal (β) domains using a risky lane changing clip as an example. We used a gradient color from light yellow to red to visualize each node's projection score indicating its relative importance. The white to red bar chart visualizes the temporal attention scores of each frame

LSTM-*last* learns a good enough temporal model that LSTM-*attn* can only slightly improve on it.

Figure 4 demonstrates how we can use the attention weights of our model to pinpoint the critical factors related to driving risk in both temporal and spatial domains, thus enabling it to *explain* its decisions. As described previously in Eq. (7), the node attention weights α are used by our graph pooling layer to filter out the objects in a *scene-graph* that are less relevant to the overall risk of the scene. Meanwhile, the temporal attention weights, β, allow the LSTM encoder to score each intermediate hidden state (p_t) and retain only the most useful information in Z for the final risk assessment. We demonstrate our model's capability to explain its decisions better using the visualization of both spatial and temporal attention shown in Fig. 4. The figure shows a clearly increasing trend of temporal attention scores $\beta_1, \beta_2, \ldots \beta_T$ as the lane-changing scenario becomes riskier over time. Intuitively, the frames with higher attention scores are weighted more heavily when calculating Z and thus contribute more to the final risk assessment decision. In this risky lane changing example, the temporal attention scores progressively increase between frames 19 and 32 during the lane change; and the highest frame attention weights

appear in frames 33 and 34, which are the frames immediately before the collision occurs. Figure 4 also shows the projection scores for the node attention layer, where a higher score for a node indicates that it contributes more to the final decision of risk assessment. As shown in this example, as the ego car approaches the yellow vehicle, the node attention weights for the ego car and the yellow vehicle are increased proportionally to the scene's overall risk. In the first few frames, the risk of collision is low; thus, the node attention weights are low; however, in the last few frames, a collision between these two vehicles is imminent; thus, the attention weights for the two cars are much higher than for any other nodes in the graph. This example clearly shows how graph representations and models, when used with attention, can effectively explain their decision-making process. This capability can be valuable for debugging edge cases at design time, thus reducing the chances of ADS making unexpected, erroneous decisions in real-world scenarios and improving human trust in the system.

4.4 Transferability from Virtual To Real Driving

This section demonstrates our approach's capability to effectively transfer the knowledge learned from a simulated dataset to a real-world dataset. As mentioned previously, this capability is vital since little real-world data exists for rare scenarios. Models must primarily rely on simulation data to improve driving safety in the real world. To demonstrate this capability, we use the model weights and parameters learned from training on the *271-syn* dataset or the *1043-syn* dataset directly for testing on the real-world driving dataset: *571-honda*. We also compare the transferability of our model with that of the baseline method [47]. The results are shown in Fig. 5.

As expected, the performance of both our approach and the baseline degrades when tested on *571-honda* dataset. However, as Fig. 5 shows, the accuracy of our approach only drops by 6.7% and 3.5% when the model is trained on *271-syn* and *1043-syn*, respectively, while the baseline's performance drops drastically by a much higher 21.3% and 14.9%, respectively. The results show that our proposed model can transfer knowledge more effectively than the baseline.

4.5 Risk Assessment By Action Type

This section shows results from evaluating our model's performance on other kinds of driving scenarios available in the HDD besides lane changes: turning, branching, merging, etc. The results for training and evaluating our model on the *1361-honda* dataset are shown in Table 2. From Table 2, we can see that our graph-based approach significantly outperforms [47] in both overall accuracy (0.86 v.s. 0.58) and overall AUC (0.91 v.s. 0.61), indicating that our approach can better

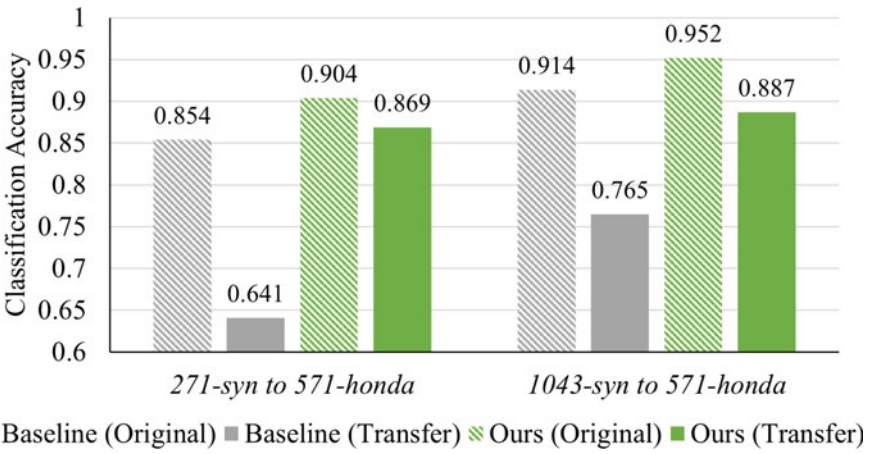

Fig. 5 Transferability comparison between our real image model and the baseline [47]. In this experiment, we trained each model on both *271-syn* dataset and *1043-syn* dataset. Then we tested the accuracy of each model on both original dataset and *571-honda* dataset

Table 2 Breakdown of risk assessment performance by driver action types (Lane Changing, Merging, Branching, and Turning) evaluated on *1361-Honda* dataset. The bolded numbers indicate the highest score between *Ours* and the baseline [47] on each of the categories of driver actions (rows)

Metric	Action type	**Ours**	[47]
Accuracy	Overall	**0.8655**	0.5844
	Lane changing	**0.8710**	0.5714
	Merging	**0.8462**	0.5854
	Branching	**0.9101**	0.5556
	Turning	**0.8211**	0.6218
AUC	Overall	**0.9124**	0.6078
	Lane changing	**0.9105**	0.5877
	Merging	**0.9395**	0.6526
	Branching	**0.9462**	0.5807
	Turning	**0.8645**	0.6400

assess risk across diverse driving scenarios and driving action types. In Table 2 we also show the performance for each action type. The results show that our approach also outperforms [47] on each class of driving action. Our approach slightly under-performs on turning scenarios compared to its performance on other action types. This discrepancy is likely because turning scenarios are intrinsically more complicated than straight-road driving scenarios (lane change, branch, merge). Another reason could be that the heading of vehicles is a more significant factor in complicated scenarios, while the *scene-graph* used in our work contains only distance and directional relations.

5 Conclusion

In this chapter, we discovered how the expressive power of graph representations of data could be leveraged to significantly improve the perception performance of automotive CPS. There were clear improvements across experiments and datasets, with our graph-based approach outperforming conventional CNN-based methods in terms of accuracy, explainability, and transferability. All of these benefits can be attributed to the explicit modeling of inter-object relationships via the graph's topology, thus improving the model's ability to semantically *understand* each scene. Although the approach presented here was effective at modeling risk, several other problems in the AV domain remain unsolved, including motion prediction, object detection, and control. When adapted to fit these problems, graph-based methods could potentially provide the same benefits over existing methods.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE Access **6**, 52138–52160 (2018)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. Preprint (2014). arXiv:14090473
3. Bansal, M., Krizhevsky, A., Ogale, A.: Chauffeurnet: learning to drive by imitating the best and synthesizing the worst. Preprint (2018). arXiv:181203079
4. Bao, N., Yang, D., Carballo, A., Özgüner, Ü., Takeda, K.: Personalized safety-focused control by minimizing subjective risk. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 3853–3858. IEEE (2019)
5. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.: Relational inductive biases, deep learning, and graph networks. Preprint (2018). arXiv:180601261
6. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. Preprint (2016). arXiv:160407316
7. Bradski, G.: The OpenCV Library. Dr Dobb's Journal of Software Tools (2000)
8. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision, pp 2722–2730 (2015)
9. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: an open urban driving simulator. Preprint (2017). arXiv:171103938
10. Gao, H., Ji, S.: Graph u-nets. Preprint (2019). arXiv:190505178
11. Hsu, B., Yu, S.Y., Malawade, A.V., Muthirayan, D., Khargonekar, P., Al Faruque, M.A.: Scene-graph-risk-assessment dataset (2021). https://doi.org/10.21227/c0z9-1p30
12. Jain, R., Kasturi, R., Schunck, B.G.: Machine Vision, vol. 5. McGraw-Hill, New York (1995)
13. Knyazev, B., Taylor, G.W., Amer, M.: Understanding attention and generalization in graph neural networks. In: Advances in Neural Information Processing Systems, pp 4202–4212. Curran Associates, Inc. (2019)
14. Laumond, J.P., et al.: Robot Motion Planning and Control, vol 229. Springer, Berlin (1998)
15. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. Preprint (2019). arXiv:190408082
16. Li, C., Meng, Y., Chan, S.H., Chen, Y.T.: Learning 3d-aware egocentric spatial-temporal interaction via graph convolutional networks. Preprint (2019). arXiv:190909272

17. Litman, T.: Autonomous Vehicle Implementation Predictions. Victoria Transport Policy Institute, Victoria (2017)
18. Malawade, A.V., Odema, M., Lajeunesse-DeGroot, S., Al Faruque, M.A.: Sage: a split-architecture methodology for efficient end-to-end autonomous vehicle control. ACM Trans. Embedd. Comput. Syst. **20**(5s), 1–22 (2021)
19. Malawade, A.V., Mortlock, T., Faruque, M.A.A.: Ecofusion: energy-aware adaptive sensor fusion for efficient autonomous vehicle perception. In: Design Automation Conference (DAC). ACM (2022)
20. Malawade, A.V., Mortlock, T., Faruque, M.A.A.: Hydrafusion: context-aware selective sensor fusion for robust and efficient autonomous vehicle perception. In: International Conference on Cyber-Physical Systems (ICCPS). IEEE (2022)
21. Malawade, A.V., Yu, S.Y., Hsu, B., Kaeley, H., Karra, A., Al Faruque, M.A.: roadscene2vec: a tool for extracting and embedding road scene-graphs. Knowl.-Based Syst. **242**, 108245 (2022)
22. Malawade, A.V., Yu, S.Y., Hsu, B., Muthirayan, D., Khargonekar, P.P., Al Faruque, M.A.: Spatio-temporal scene-graph embedding for autonomous vehicle collision prediction. IEEE Internet Things J. **9**(12), 9379–9388 (2022) https://doi.org/10.1109/JIOT.2022.3141044
23. Montgomery, W.D., Mudge, R., Groshen, E.L., Helper, S., MacDuffie, J.P., Carson, C.: Securing America's future energy. 52 p. Available: https://avworkforce.secureenergy.org/
24. Mueller, A.S., Cicchino, J.B., Zuby, D.S.: What humanlike errors do autonomous vehicles need to avoid to maximize safety? J. Saf. Res. **75**, 310–318 (2020) ISSN 0022-4375, https://doi.org/10.1016/j.jsr.2020.10.005
25. Mylavarapu, S., Sandhu, M., Vijayan, P., Krishna, K.M., Ravindran, B., Namboodiri, A.: Towards accurate vehicle behaviour classification with multi-relational graph convolutional networks. Preprint (2020). arXiv:200200786
26. 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
27. 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
28. 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
29. Nistér, D., Lee, H.L., Ng, J., Wang, Y.: The safety force field. NVIDIA White Paper (2019)
30. Paden, B., Čáp, M., Yong, S.Z., Yershov, D., Frazzoli, E.: A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans. Intell. Veh. **1**(1), 33–55 (2016)
31. Pomerleau, D.A.: Alvinn: an autonomous land vehicle in a neural network. In: Advances in Neural Information Processing Systems, pp 305–313. Morgan Kaufmann Publishers (1989)
32. Rajamani, R.: Vehicle Dynamics and Control. Springer Science & Business Media, New York (2011)
33. Ramanishka, V., Chen, Y.T., Misu, T., Saenko, K.: Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In: Conference on Computer Vision and Pattern Recognition (2018)
34. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp 91–99. Curran Associates, Inc. (2015)
35. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference, pp 593–607. Springer (2018)
36. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. **45**(4), 427–437 (2009)
37. Sontges, S., Koschi, M., Althoff, M.: Worst-case analysis of the time-to-react using reachable sets. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp 1891–1897. IEEE (2018)

38. Tao, C., He, H., Xu, F., Cao, J.: Stereo priori rcnn based car detection on point level for autonomous driving. Knowl.-Based Syst. **229**, 107346 (2021)
39. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp 5998–6008. Curran Associates, Inc. (2017)
40. Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, V.: Facebook AI research (FAIR). Available: https://github.com/facebookresearch/detectron2. (2019)
41. Xiao, D., Yang, X., Li, J., Islam, M.: Attention deep neural network for lane marking detection. Knowl. Based Syst. **194**, 105584 (2020)
42. Xu, D., Zhu, Y., Choy, C.B., Fei-Fei, L.: Scene graph generation by iterative message passing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 5410–5419 (2017)
43. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? Preprint (2018). arXiv:181000826
44. Yang, J., Lu, J., Lee, S., Batra, D., Parikh, D.: Graph r-cnn for scene graph generation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 670–685 (2018)
45. Yu, S.Y., Malawade, A.V., Muthirayan, D., Khargonekar, P.P., Al Faruque, M.A.: Scene-graph augmented data-driven risk assessment of autonomous vehicle decisions. IEEE Trans. Intell. Trans. Syst. **23**(7), 7941–7951 (2021) https://doi.org/10.1109/TITS.2021.3074854
46. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: common practices and emerging technologies. Preprint (2019). arXiv:190605113
47. Yurtsever, E., Liu, Y., Lambert, J., Miyajima, C., Takeuchi, E., Takeda, K., Hansen, J.H.: Risky action recognition in lane change video clips using deep spatiotemporal networks with segmentation mask transfer. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp 3100–3107. IEEE (2019)