Scene-Graph Augmented Data-Driven Risk Assessment of Autonomous Vehicle Decisions

Shih-Yuan Yu, Arnav Vaibhav Malawade, *Graduate Student Member, IEEE*, Deepan Muthirayan, Pramod P. Khargonekar, *Fellow, IEEE*, and Mohammad Abdullah Al Faruque, *Senior Member, IEEE*

***Abstract*— There is considerable evidence that evaluating the subjective risk level of driving decisions can improve the safety of Autonomous Driving Systems (ADS) in both typical and complex driving scenarios. In this paper, we propose a novel data-driven approach that uses scene-graphs as intermediate representations for modeling the subjective risk of driving maneuvers. Our approach includes a Multi-Relation Graph Convolution Network, a Long-Short Term Memory Network, and attention layers. To train our model, we formulate subjective risk assessment as a supervised scene classification problem. We evaluate our model on both synthetic lane-changing datasets and real-driving datasets with various driving maneuvers. We show that our approach achieves a higher classification accuracy than the state-of-the-art approach on both large (96.4% vs. 91.2%) and small (91.8% vs. 71.2%) lane-changing synthesized datasets, illustrating that our approach can learn effectively even from small datasets. We also show that our model trained on a lane-changing synthesized dataset achieves an average accuracy of 87.8% when tested on a real-driving lane-changing dataset. In comparison, the state-of-the-art model trained on the same synthesized dataset only achieved 70.3% accuracy when tested on the real-driving dataset, showing that our approach can transfer knowledge more effectively. Moreover, we demonstrate that the addition of spatial and temporal attention layers improves our model's performance and explainability. Finally, our results illustrate that our model can assess the risk of various driving maneuvers more accurately than the state-of-the-art model (86.5% vs. 58.4%, respectively).**

***Index Terms*— Autonomous vehicle, risk assessment, scene understanding, graph convolutional neural network.**

I. Introduction

AUTONOMOUS Driving Systems (ADSs) have advanced significantly in recent years. However, navigation is still challenging in complex urban environments since the scenarios are highly variable and complex [1]–[3]. The continued reports of autonomous vehicle crashes only highlight these challenges [4]–[7]. A risk-based approach for autonomous driving has the potential to address this challenge and better assure driving safety. Within this context, the effectiveness of understanding the driving scenes and quantifying the risk of driving decisions becomes particularly crucial for ADSs.

Manuscript received August 31, 2020; revised February 19, 2021; accepted April 17, 2021. This work was supported in part by the NSF under Award CMMI-1739503 and Award ECCS-1839429. The Associate Editor for this article was L. M. Bergasa. *(Corresponding author: Shih-Yuan Yu.)*

The authors are with the Department of Electrical Engineering and Computer Science, University of California at Irvine, Irvine, CA 92697 USA (e-mail: shihyuay@uci.edu; malawada@uci.edu; dmuthira@uci.edu; pramod.khargonekar@uci.edu; alfaruqu@uci.edu).

Digital Object Identifier 10.1109/TITS.2021.3074854

Several papers have leveraged state-of-the-art deep learning architectures for modeling subjective risk [2], [8]. Such methods typically use Convolutional Neural Networks (CNNs) and Long-Short Term Memory Networks (LSTMs) and have proven effective at capturing features essential for modeling subjective risk in both spatial and temporal domains [8]. However, it is unclear whether these methods can capture critical higher-level information, such as the relationships between traffic participants in a given scene. Failing to capture these relationships can result in poor ADS performance in complex scenarios.

To address these limitations, 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 Figure 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: (i) a pipeline to convert the images of a driving clip to a sequence of *scene-graphs*, (ii) a Multi-Relational Graph Convolution Network (MR-GCN) to convert each of the *scene-graphs* to an embedding (a vectorized representation), and (iii) an LSTM for temporally modeling the sequence of embeddings of the respective *scene-graphs*. Our model also contains multiple attention layers: (i) a node attention layer before the embedding of a *scene-graph* is computed, and (ii) an attention layer on top of the LSTM, both of which can further improve its performance and explainability. For training the model, we formulate the problem of subjective risk assessment as a supervised *scene-graph* sequence classification problem.

A. Research Challenges

Overall, designing a risk assessment system for ADSs using data-driven approaches presents 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.

To overcome Research Challenge 1, deep learning based subjective risk assessment methods must be trained on large

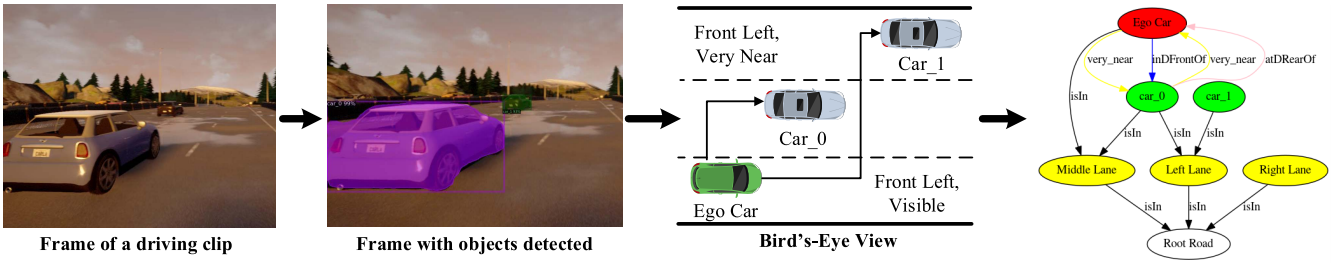

Fig. 1. An illustration of *scene-graph* extraction using the *Real Image Pipeline*. In this process, the first step is to detect a list of objects on each frame of a clip. Then, we project each frame to its bird's-eye view to better approximate the spatial relations between objects. Finally, we construct a *scene-graph* using the list of detected objects and their attributes.

datasets covering a wide range of "corner cases" (especially risky driving scenarios), which are expensive and time-consuming to generate [9]. Many researchers resort to using synthesized datasets containing many examples of these corner cases to address this issue. However, for these to be valuable, a model must be able to transfer the knowledge gained from simulated training data to real-world situations (Research Challenge 2). 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. If a model can transfer the knowledge gained from a simulated training set to a real-world testing set effectively, 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 [10]–[12]. *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 [12], [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 [14] (Research Challenge 3).

B. Our Novel Contributions

Our key contributions are as follows:

1) We present a novel *scene-graph* augmented data-driven approach for assessing the risk of driving actions in autonomous vehicles.
2) We demonstrate that our approach outperforms existing methods at risk assessment across a wide range of scenarios using lane change as a use case.
3) We demonstrate that the use of multi-level attention in our proposed approach provides better explainability.
4) We demonstrate that our *scene-graph* based approach can better transfer knowledge gained from simulated environments to real-world risk assessment tasks.

C. Paper Organization

The rest of the paper is structured as follows: In Section II, we discuss related works. In Section III we introduce our *scene-graph* augmented approach. In Section IV, we discuss our experimental results. Finally, in Section V, we present our conclusions.

II. Related Work

A. Design Philosophies in ADSs

Two broad approaches for designing ADSs are (i) modular design, (ii) end-to-end design [2]. Most modular approaches comprise a pipeline of separate components from the sensory inputs to the actuator outputs, while end-to-end approaches generate output directly from their sensory inputs [15], [16]. 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 [17], computer vision [18] and vehicle dynamics [19]. Therefore, 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 [2]. End-to-end 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 [16]. 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 controlling information (e.g., from image to steering angle), so it can fail to handle complicated driving maneuvers.

A third approach was first proposed by DeepDriving [20], called the *direct perception* approach. In their 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 IR for generating the final control output. They prove that the use of this IR is effective for simple driving tasks such as lane following and generalizing to real-world environments. Authors in [21] use a collection of filtered images, each representing a piece of distinct information, as the IR. They state that the IR used in their approach 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. The authors in [8] use Mask-RCNN [22] to color the vehicles in each input image, producing a form of IR. In contrast to the works

mentioned above, our approach uses a *scene-graph* IR that encodes the spatial and semantic relations between all the traffic participants in a frame.

B. Graph-Based Driving Scene Understanding

Several papers have applied graph-based formulations for driving scene understanding [23]–[25]. In [24], the authors propose a 3D-aware egocentric Spatio-temporal interaction framework that uses both an *Ego-Thing* graph and an *Ego-Stuff* graph, which encode how the ego vehicle interacts with both moving and stationary objects in a scene, respectively. In [23], [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 is constructed by combining spatial and temporal information, including relational information between moving objects and landmark objects. Our work is primarily inspired by [23], [25] but differs in the application and network architecture. These papers focus on predicting each object's behavior in the *scene-graph* while our work focuses on assessing the subjective risk of the entire scenario. Consequently, we propose a network architecture that implements more components such as node-attention, graph pooling layers, and readout operations.

C. Risk Assessment

In prior research, the problem of risk assessment for autonomous driving has been tackled by modeling either the *objective risk* or the *subjective risk* [10], [26], [27]. The *objective risk* is defined as the objective probability of an accident occurring and is usually determined by statistical analysis [26]. Some works have focused on minimizing the *objective risk* by modeling the trajectories of vehicles [28], [29] to guarantee safe driving. *Subjective risk* refers to the driver's own perceived risk and is an output of the driver's cognitive process [10], [27]. One primary reason why assessing subjective risk is important is because it accounts for the human behavior perspective and its critical role in anticipating risks, as many works point out [10], [27], [30]. Further, studies such as [26], [31] provide direct evidence that a driver's subjective risk assessment is inversely related to the risk of traffic accidents. Similarly, [10] suggests that augmenting an objective risk assessment system with *subjective risk* techniques can improve overall risk assessment performance. For these reasons, our goal in this work is to build a model for subjective risk assessment. In Figure 2, we present examples of a lane change that is both *subjectively* and *objectively* risky and a lane change that is *subjectively safe* but *objectively risky* from our driving dataset. Both an objective and subjective approach would likely identify the obvious risk factor of the close-proximity, high-speed vehicle in (a). However, an objective risk assessment approach may incorrectly consider (b) to be risky because of the ego car's perceived trajectory while this is, in fact, a safe lane change.

Several works have studied subjective risk assessment for autonomous driving systems [8], [10], [27], [32]. In [32], Hidden Markov Models (HMMs) and Language Models are used to detect unsafe lane change events. The approach taken in [8] is the most related to our work as it infers the risk-level of overall driving scenes with a deep Spatio-temporal neural network architecture. Using Mask-RCNN [22] to generate an IR for each image, their approach achieves a 3% performance gain in risk assessment. They show that the architecture with Semantic Mask Transfer (SMT) + CNN + LSTM can perform 25% better than the architecture with Feature Transfer (FT) + Frame-by-Frame (FbF). This result indicates that capturing the spatial and temporal features from a single camera can be useful in modeling subjective risk. However, this approach only considers the spatial features (the latent vector output of the CNN layers) of a frame instead of the relations between all the traffic participants. Our work uses *scene-graphs* as IRs to capture the high-level relationships between all the traffic participants of a scene.

Fig. 2. An example of (a) a lane change that is both *subjectively* and *objectively* risky as well as (b) a *subjectively safe* but *objectively risky* lane change from our driving dataset. In (a), the ego car starts a safe lane change, but a high-speed vehicle suddenly appears in the ego car's blind spot and nearly collides with it. In (b), the ego car appears to drive directly towards the adjacent vehicle but is actually making a safe lane change on a curved road.

III. Scene-Graph Augmented Approach for Risk Assessment

In this section, we discuss our proposed approach for *scene-graph* augmented risk assessment. In our work, we make the same assumption used in [8] 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 Equation 1.

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

where

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

In this section, we propose a suitable model for approximating the function f. In the model we propose, 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 Section III-A. 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 *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 $\mathbf{I}$. We describe more details regarding each of our model's components in Section III-B.

A. Scene-Graph Extraction Pipeline

Several approaches have been proposed for extracting scene-graphs from images by detecting the objects in a scene and then identifying their visual relationships [33], [34]. However, these works have focused on single general images instead of a sequence of images as it arises in autonomous driving, where higher accuracy is demanded. Thus, we adopted a partially rule-based process to extract objects and their attributes from images called the *Real Image Pipeline*. Besides, to evaluate how our approach performs with *scene-graphs* containing ground truth information, we use the *Carla Ground Truth (GT) Pipeline* as a surrogate for the ideal situation where the attributes for each object can be correctly extracted. After the objects and attributes have been extracted, the *scene-graphs* are constructed as described in III-A.3. We discuss these two approaches in detail below.

1) Real Image Pipeline: In this pipeline, object attributes and bounding boxes are extracted directly from images using state-of-the-art image processing techniques. As Figure 1 shows, we first convert each image I_t into a collection of objects O_t using Detectron2, a state-of-the-art object detection model based on Faster RCNN [35], [36]. 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 [37]. This projection allows us to 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*.

2) Carla Ground Truth Pipeline: Object detection and location estimation with solely a monocular camera can be unstable because of factors such as weather and camera position [38], which can impact the correctness of our image-based *scene-graph* construction pipeline and thus our approach's performance. To evaluate our methodology under the assumption that object attributes can be extracted without error, we build our *scene-graphs* using the ground-truth location and class information for each vehicle in the *Carla GT Pipeline*. We extract this information directly from Carla simulator [9] without using any image processing steps.

3) Graph Construction: After collecting the list of objects in each image and their attributes, we begin constructing the corresponding *scene-graphs* as follows. 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.).

In assessing the risk of driving behaviors, traffic participants' relations that we consider useful are the distance relations and the directional relations. The assumption made here is that the local proximity and positional information of one object 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 Figure 1. to ed 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. For the *Carla GT Pipeline*, we extract the ground-truth lane assignments for each vehicle from the simulator directly. For the *Real Image Pipeline*, 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* based on a 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*.

B. Scene-Graph Sequence Based Risk Assessment Model

The model we propose 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 Figure 3. We discuss each of these components in detail below.

1) 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

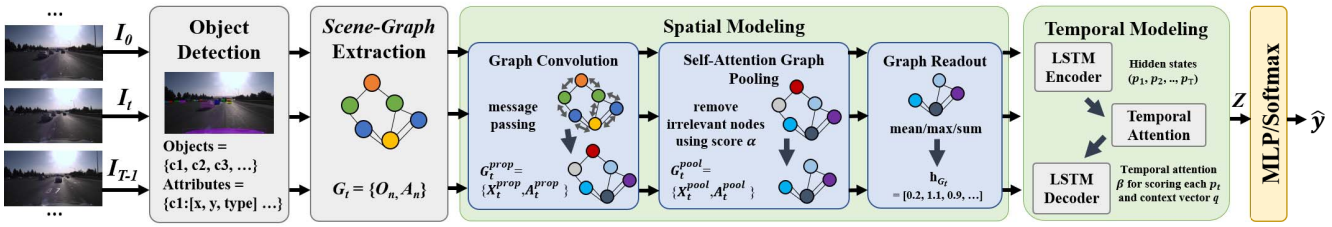

Fig. 3. An illustration of our model's architecture. First each image $I_t \in \mathbf{I}$ is converted to a *scene-graph* G_t via the *Scene-Graph Extraction Pipeline*. Next each *scene-graph* G_t is converted to its corresponding *scene-graph* embedding $\mathbf{h}_{G_t}$ via the graph convolution, pooling, and readout operations in the Spatial Modeling block. Then, the resulting *scene-graph* embeddings are sequentially processed by LSTM and temporal attention layers to acquire the spatio-temporal representation $\mathbf{Z}$ for a *scene-graph* sequence. Finally, the risk inference $\hat{y}$ of the sequence is calculated from $\mathbf{Z}$ using an MLP with a *Softmax* activation function.

embedding, denoted as $\mathbf{h}_v^{(l)}$, for each node v as follows:

$$\mathbf{h}_v^{(l)} = \Phi_0 \cdot \mathbf{h}_v^{(l-1)} + \sum_{r \in \mathbf{A_t}} \sum_{u \in \mathbf{N}_r(v)} \frac{1}{|\mathbf{N}_r(v)|} \Phi_r \cdot \mathbf{h}_u^{(l-1)}, \quad (3)$$

where $N_r(v)$ denotes the set of neighbor indices of node v with the relation $r \in A_t$. Φ_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 Φ_0 to account for the self-connection of each node using a special relation [39]. 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 [40] 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,

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

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 [41] or a separate trainable GNN layer [42]. We denote the graph pooling layer that uses the **SCORE** function in [41] as *TopkPool* and the one that uses the **SCORE** function in [42] as *SAGPool*. The calculation of the overall process is presented as follows:

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

$$\mathbf{P} = \mathrm{top}_k(\alpha), \quad (6)$$

where α 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:

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

$$\mathbf{A_t^{pool}} = \mathbf{A_t^{prop}}_{(\mathbf{P},\mathbf{P})}. \quad (8)$$

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

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

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, $\mathbf{h}_I$.

2) 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,

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

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

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

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 [11], 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) 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

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

The loss for the prediction is calculated as follows,

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

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.

IV. Experimental Results

In this section, we provide extensive experimental results to illustrate the accuracy of our model and its ability to transfer knowledge (transferability) for a specific driving maneuver: lane changes. This task by itself is crucial, given that 7.62% of all traffic accidents between light vehicles can be attributed to improper execution of lane changes [43]. Besides, we also evaluate our model's capability for turning as well as entering or leaving the traffic flow of a road (merging and branching, respectively). We evaluate our approach by providing comparisons between our model and a state-of-the-art SMT+CNN+LSTM based risk assessment model [8]. We refer to this model as the *baseline*. Besides, we provide results for our model's best hyper-parameter setting and perform an ablation study to evaluate the contribution of each major component in our model.

A. Dataset Preparation

We prepared two types of datasets for our experiments (i) synthesized lane-changing datasets and (ii) real-world driving datasets. To create the synthesized datasets, we developed a tool to generate lane-changing clips using the Carla[1] and Carla Scenario Runner.[2] We generated the real-world dataset by extracting lane change clips and other driving actions from the Honda Driving Dataset (HDD) [44].

Carla is an open-source driving simulator [9] that allows users to control a vehicle in either manual mode or autopilot mode. The Carla Scenario Runner contains a set of atomic controllers that enable users to control a car in a driving scene and perform complex driving maneuvers. We modified the user script in Carla so that it can (i) select one autonomous car randomly and switch its mode to manual mode, and then (ii) utilize Scenario Runner's function to force the vehicle to change lanes.

The data generating tool allows us to fabricate lane changing clips directly instead of extracting them from long driving clips. 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, etc. Also, through the APIs provided by the Traffic Manager (TM) of the Carla simulator, we were able to customize the driving characteristics of every autonomous vehicle, 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, this allowed us to simulate a wide range of very realistic urban driving environments and generate synthesized datasets suitable for training and testing a model.

We generated two synthesized lane-changing datasets: a *271-syn* dataset and a *1043-syn* dataset, containing 271 and 1,043 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*. For real driving datasets, we processed the HDD dataset to create a dataset called *1361-honda*. Specifically, *1361-honda* contains 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 *271-syn* and *1043-syn* datasets, we subsampled *1361-honda* to create a real-driving lane-changing dataset which contains 571 lane changing clips, denoted as the *571-honda* dataset.

To label the video clips in both the real-world and synthesized datasets, we performed an annotation process similar to the one used in [8]. The process starts with multiple human

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

[2] https://github.com/carla-simulator/scenario_runner

annotators assigning a risk score to each clip that ranges from -2 to 2, where 2 implies a highly risky driving scenario and -2 implies the safest driving scenario. Then for each video clip, the risk labels of all the annotators are averaged and converted to a binary label y as follows: if the average is ≤ 0, then the label $y = 0$ (safe) is assigned, else label $y = 1$ (risky) is assigned.

In our work, we used five different anonymous annotators. After the annotation process, the annotators were required to write down the criteria and rationale they used when annotating the video clips. This process ensured that the annotators paid attention while labeling, reducing the odds of trivial misjudgment of a driving scene's risk level. The risk factors common among the five annotators were 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. Besides, the sudden or random appearance of objects in the scene was also a critical factor in determining a driving scene's risk level.

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 each of the splits 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 ten different stratified train-test splits of the dataset.

B. Training and Model Specification

Our models were implemented using *PyTorch* and *PyTorch-Geometric* [45], [46]. We used the ADAM optimizer for the training algorithm. We considered three learning rates: $\{0.0005, 0.0001, 0.00005\}$, and a weight decaying rate of 5×10^{-4} per epoch. We used a batch size of 16 sequences for each training epoch. In our experiments, we trained each model for 200 epochs. Regarding the setting of hyper-parameters, we considered the options described in Section III-B. From our experimentation, we found that the best option for the hyper-parameters of our model is a mini-batch size of 16 sequences, a learning rate of 0.00005, two MR-GCN layers with 100 hidden units, a SAGPool pooling layer with a ratio of 0.5, *sum-pooling* for graph readout operation and *LSTM-attn* for temporal modeling.

To ensure a fair comparison between our model and the baseline, we reported the performance of the model configurations with the lowest validation loss throughout the training in the results section. All the experiments were conducted on a server with one NVIDIA TITAN-XP graphics card and one NVIDIA GeForce GTX 1080 graphics card. For implementing the baseline model [8], we used the source code available at their open-source repository.[3] The source code and *scene-graph* datasets used in our experiments are open sourced at https://github.com/louisccc/sg-risk-assessment.

[3]https://github.com/Ekim-Yurtsever/DeepTL-Lane-Change-Classification

C. Experiments on Risk Assessment

We evaluate each model's performance by measuring its classification accuracy and the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) for each dataset. 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 [47], 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*).

Figure 4 shows the comparison between our model's performance and the baseline model [8] for all the synthetic datasets. The results show that our approach consistently outperforms [8] across all the datasets in terms of both classification accuracy and AUC. Particularly, on the *1043-syn* dataset, our Image-based and GT pipelines outperform [8] in classification accuracy by 4.4% and 5% respectively (i.e., an accuracy of 95.8% and 96.4% compared to 91.4% for the baseline).

We found that the performance difference between our approach and the baseline increased when the training datasets were smaller. Figure 4 shows that the difference in the accuracy between our approach using the GT pipeline and the baseline [8] is 5% for the *1043-syn* dataset and 8.7% for the *271-syn* dataset. This result indicates that our approach can learn an accurate model even from a smaller dataset. We postulate this is a direct result of its use of a *scene-graph* based IR.

We also found that our approach performs better than the baseline on balanced datasets. Among the datasets used for evaluation of the models, 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*). We found that the performance difference between our image-based approach and the baseline on these datasets is 12.9% on the *306-syn* dataset compared to 7.8% on the *271-syn* dataset, indicating that our approach can discriminate between the two classes better than the baseline.

We also evaluated the contribution of each functional component in our proposed model by conducting an ablation study. The results of the study are shown in Table I. From Table I 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 I), achieves a classification accuracy of 75%. We find that replacing *mean* with an LSTM layer for temporal modeling yields a 10.5% increase in performance. We also find that including a single MR-GCN with 64 hidden units and *sum-pooling* to the simplest model results in a 14.8% performance gain over the simplest model. The performance gain achieved by just including the MR-GCN layer suggests the effectiveness of considering the relations between objects. Finally, we find that the model with one MR-GCN with 64 hidden units and *sum-pooling* plus the LSTM layer for temporal modeling yields the maximum gain of 18.1% over

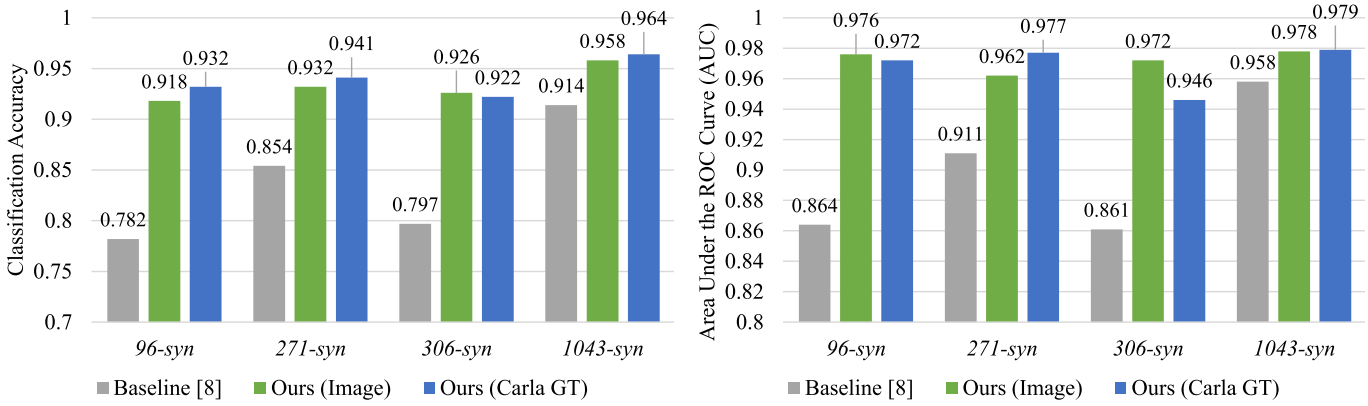

Fig. 4. Accuracy and AUC comparison between our approaches (Real Image and Carla GT) and [8] on different datasets. In these experiments, we trained the model using the hyper-parameter settings reported in Section IV-C.

TABLE I

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

	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**

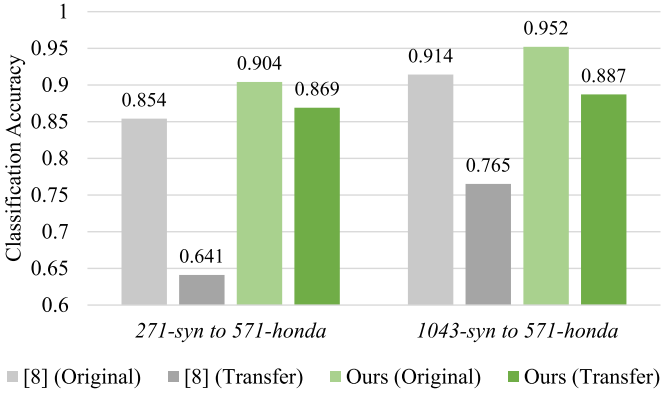

Fig. 5. The results of comparing transferability between our Real Image model and [8]. In this experiment, we trained our model using our best hyper-parameters on both *271-syn* dataset and *1043-syn* dataset. Then we tested the accuracy of our approach on both original dataset and *571-honda* dataset. We followed the same procedure to train and test [8].

the simplest model. These results clearly demonstrate the importance of each component in the model we propose.

D. Evaluation of Attention Mechanisms on Risk Assessment

In this section, 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 detailed results that elucidate the effectiveness of these different attention mechanisms are presented in Table I.

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. This result indicates that the use of 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 is likely because *TopkPool* ignores the relations between nodes when calculating α. Another reason for this instability could be the random initialization of weights in *TopkPool*, which can exponentially affect the overall performance as stated in [13].

For evaluating the impact of attention on the temporal model, we evaluated the effects of adding a temporal attention layer to the following two models: (i) with no MR-GCN layers and no temporal attention and (ii) with one MR-GCN layer and no temporal attention. Compared to the model with no MR-GCN layer and no temporal attention, the model's performance with no MR-GCN and *LSTM-attn* was found to be 0.1% higher. We also found 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 does improve the performance, though only marginally. The reason why we only see a marginal improvement can be that the temporal attention layer is less relevant with the dataset that our model was trained on. When preparing these datasets, we manually removed the frames that are irrelevant to a lane change, exactly the set of frames that temporal attention would have given less attention to, thus minimizing its effect.

Figure 6 demonstrates our model's capability to pinpoint the critical factors in assessing driving risk in both temporal and spatial domains. As Equation 7 shows, the node attention weights α are used by our graph pooling layer to filter the

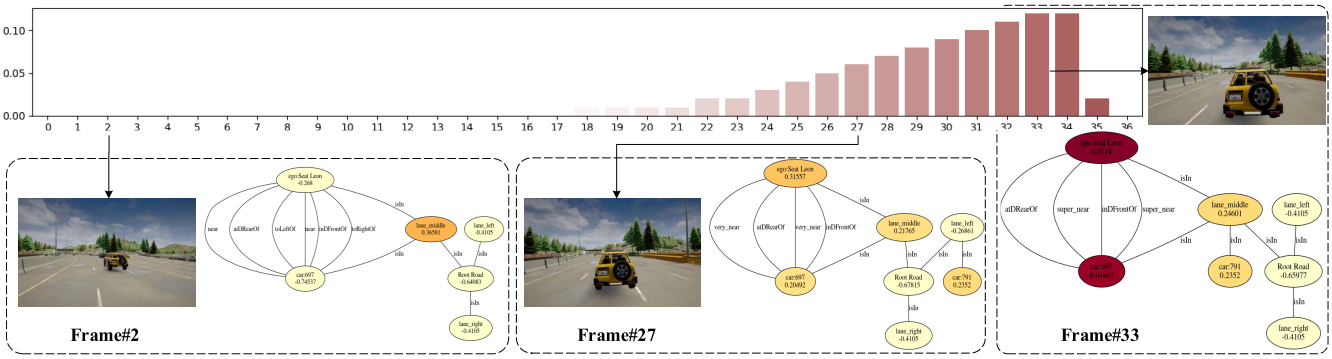

Fig. 6. The visualization of attention weights in both spatial (α) and temporal (β) domains using a risky lane changing clip as an example. We used a gradient color from light yellow to red for visualizing each node's projection score that indicates its importance in calculating a *scene-graph* embedding. We also used a gradient colored (white to red) bar chart to visualize the temporal attention coefficients $\beta_1, \beta_2, \ldots \beta_{36}$ used for calculating the context vector c.

objects in a scene-graph that are less significant for assessing the risk. In the temporal domain, the attention weights, β, allow the LSTM encoder to score each intermediate hidden state (p_t) and retain only the useful information in Z for the final risk assessment. Table I demonstrates the benefit of applying attention layers in risk assessment. Specifically, it shows that the addition of the *SAGPool* attention layer improves the accuracy of the baseline model (1 MR-GCN + mean) by 2.7% and that including *LSTM-attn* to the baseline model (1 MR-GCN + *LSTM-last*) increases the performance by 0.3%.

In addition to improving our model's performance, the inclusion of graph and temporal attention improves the explainability of our model's risk assessment decisions. We demonstrate this capability using the visualization of both graph attention and temporal attention provided in Figure 6. Figure 6 shows the trend of the attention scores $\beta_1, \beta_2, \ldots \beta_T$ for a risky lane changing clip. Intuitively, the frame with a higher attention score α_t contributes more to the context vector c (shown in Equation 12), thus playing a more critical role in calculating h_{G_t} and contributing 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 6 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 demonstrates our model's capability to pinpoint the critical factors in a *scene-graph* that contributed to its risk assessment decision. 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.

E. Transferability From Virtual to Real Driving

In this section, we demonstrate our approach's capability to effectively transfer the knowledge learned from a simulated dataset to a real-world dataset. 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 [8]. The results are shown in Figure 5.

As expected, the performance of both our approach and the baseline degrades when tested on *571-honda* dataset. However, as Figure 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 categorically show that our proposed model can transfer knowledge more effectively than the baseline.

F. Risk Assessment by Action Type

In this section, we show results from evaluating our model's performance on various other driving scenarios available in the HDD: turning, branching, merging, etc. The results for training and evaluating our model on the *1361-honda* dataset are shown in Table II. From Table II, we can observe that our approach significantly outperforms [8] 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 assess risk across various driving scenarios. In Table II we also show the performance for each action type. The results show that our approach also outperforms [8] on each individual type of driving scenario. 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 contributing factor in complicated scenarios, while the *scene-graph* used in our work contains only distance and directional relations.

TABLE II

BREAKDOWN OF RISK ASSESSMENT PERFORMANCE BY DRIVER ACTION TYPES (LANE CHANGING, MERGING, BRANCHING, AND TURNING) EVALUATED ON *1361-Honda* DATASET

Metric	Action Type	**Ours**	[8]
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

V. CONCLUSION

Subjective risk assessment is a challenging, safety-critical problem that requires a good semantic understanding of many possible road scenarios. Our results show that our scene-graph augmented approach outperforms state-of-the-art techniques at risk assessment tasks in terms of accuracy (95.8% vs. 91.4%) and AUC (0.978 vs. 0.958). We also show that our approach can learn with much less training data than these techniques, as our approach achieves 91.8% accuracy on the *96-syn* dataset compared to 78.2% accuracy achieved by [8]. Additionally, our results show that our approach can better transfer knowledge gained from simulated datasets to real-world datasets (5.0% avg. acc. drop for our approach vs. 18.1% avg. acc. drop for [8]). We also show that the use of spatial and temporal attention components improves our approach's performance and explainability. Finally, our results illustrate that our model can more accurately assess the risk of diverse driving maneuvers than the state-of-the-art model (86.5% for our approach vs. 58.4% for [8]).

ACKNOWLEDGMENT

The authors thank the colleagues of the Embedded & Cyber-Physical Systems (AICPS) Lab, from University of California, Irvine, particularly Emily Sing Yen Lam, Aung Myat Thu, and Brandon Hsu for their contribution during the course of this research. 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] R. Mudge, D. Montgomery, E. Groshen, J. Groshen, S. Helper, and C. Carson, "America's workforce and the self-driving future realizing productivity gains and spurring economic growth," Securing Amer. Future Energy, Washington, DC, USA, Tech. Rep., Jun. 2018. [Online]. Available: https://trid.trb.org/view/1516782

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *arXiv preprint arXiv:1906.05113*, 2019.

[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[4] D. Lavrinc, "This is how bad self-driving cars suck in rain," Tech. Rep., 2014. [Online]. Available: https://jalopnik.com/this-is-how-bad-self-driving-cars-suck-in-the-rain-1666268433

[5] A. Davies, "Google's self-driving car caused its first crash," *Wired*, to be published.

[6] M. McFarland. (2016). *Who's Responsible When an Autonomous car Crashes*. CNN. [Online]. Available: https://money.cnn.com/2016/07/07/technology/tesla-liability-risk/index.%html

[7] T. B. Lee, "Autopilot was active when a tesla crashed into a truck, killing driver," *ARS Technica*, to be published. [Online]. Available: https://arstechnica.com/cars/2019/05/feds-autopilot-was-active-during-d%eadly-march-tesla-crash/

[8] E. Yurtsever *et al.*, "Risky action recognition in lane change video clips using deep spatiotemporal networks with segmentation mask transfer," in *Proc. IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 3100–3107.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," 2017, *arXiv:1711.03938*. [Online]. Available: http://arxiv.org/abs/1711.03938

[10] N. Bao, D. Yang, A. Carballo, Ü. Özgüner, and K. Takeda, "Personalized safety-focused control by minimizing subjective risk," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3853–3858.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[12] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[13] B. Knyazev, G. W. Taylor, and M. Amer, "Understanding attention and generalization in graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 4202–4212.

[14] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[15] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 305–313.

[16] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: http://arxiv.org/abs/1604.07316

[17] J.-P. Laumond *et al.*, *Robot Motion Planning Control*, vol. 229. Berlin, Germany: Springer, 1998.

[18] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, vol. 5. New York, NY, USA: McGraw-Hill, 1995.

[19] R. Rajamani, *Vehicle Dynamics and Control*. Berlin, Germany: Springer, 2011.

[20] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2722–2730.

[21] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst," 2018, *arXiv:1812.03079*. [Online]. Available: http://arxiv.org/abs/1812.03079

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2961–2969.

[23] S. Mylavarapu, M. Sandhu, P. Vijayan, K. M. Krishna, B. Ravindran, and A. Namboodiri, "Towards accurate vehicle behaviour classification with multi-relational graph convolutional networks," 2020, *arXiv:2002.00786*. [Online]. Available: http://arxiv.org/abs/2002.00786

[24] C. Li, Y. Meng, S. H. Chan, and Y.-T. Chen, "Learning 3D-aware egocentric spatial-temporal interaction via graph convolutional networks," 2019, *arXiv:1909.09272*. [Online]. Available: http://arxiv.org/abs/1909.09272

[25] S. Mylavarapu, M. Sandhu, P. Vijayan, K. M. Krishna, B. Ravindran, and A. Namboodiri, "Understanding dynamic scenes using graph convolution networks," 2020, *arXiv:2005.04437*. [Online]. Available: http://arxiv.org/abs/2005.04437

[26] G. Grayson, G. Maycock, J. Groeger, S. Hammond, and D. Field, "Risk, hazard perception and perceived control," TRL Ltd., Crowthorne, U.K., Tech. Rep. TRL560, 2003.

[27] R. Fuller, "Towards a general theory of driver behaviour," *Accident Anal. Prevention*, vol. 37, no. 3, pp. 461–472, May 2005.

[28] C. Laugier *et al.*, "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety," *IEEE Intell. Transp. Syst. Mag.*, vol. 3, no. 4, pp. 4–19, Oct. 2011.

[29] H. Woo *et al.*, "Advanced adaptive cruise control based on collision risk assessment," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 939–944.

[30] N. Bao, A. Carballo, C. Miyajima, E. Takeuchi, and K. Takeda, "Personalized subjective driving risk: Analysis and prediction," *J. Robot. Mechtron.*, vol. 32, no. 3, pp. 503–519, 2020.

[31] U. Tränkle, C. Gelau, and T. Metker, "Risk perception and age-specific accidents of young drivers," *Accident Anal. Prevention*, vol. 22, no. 2, pp. 119–125, Apr. 1990.
[32] E. Yurtsever *et al.*, "Integrating driving behavior and traffic context through signal symbolization for data reduction and risky lane change detection," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 3, pp. 242–253, Sep. 2018.
[33] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5410–5419.
[34] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph R-CNN for scene graph generation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 670–685.
[35] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," Tech. Rep., 2019. [Online]. Available: https://github.com/facebookresearch/detectron2
[36] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[37] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, to be published.
[38] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 11621–11631.
[39] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.* Cham, Switzerland: Springer, 2018, pp. 593–607.
[40] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826* . [Online]. Available: https://arxiv.org/abs/1810.00826
[41] H. Gao and S. Ji, "Graph U-Nets," 2019, *arXiv:1905.05178*. [Online]. Available: https://arxiv.org/abs/1905.05178
[42] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," 2019, *arXiv:1904.08082*. [Online]. Available: https://arxiv.org/abs/1904.08082
[43] W. G. Najm *et al.*, "Pre-crash scenario typology for crash avoidance research," Nat. Highway Traffic Saf. Admin., Washington, DC, USA, Tech. Rep. DOT HS 810 767, 2007.
[44] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko, "Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7699–7707.
[45] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. NIPS-W*, 2017, pp. 1–4.
[46] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," 2019, *arXiv:1903.02428*. [Online]. Available: https://arxiv.org/abs/1903.02428
[47] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009.

Shih-Yuan Yu received the B.S. and M.S. degrees in computer science and information engineering from National Taiwan University (NTU), Taipei, Taiwan in 2014. He is currently pursuing the Ph.D. degree with the University of California at Irvine. He worked at MediaTek for four years. His current research interest is about the design automation of embedded systems using data-driven system modeling approaches. It covers incorporating machine learning methods to identify potential security issues in systems.

Arnav Vaibhav Malawade (Graduate Student Member, IEEE) received the B.S. degree in computer science and engineering and the M.S. degree in computer engineering from the University of California at Irvine (UCI) in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree with UCI, under the supervision of Prof. Mohammad Al Faruque. His research interests include the design and security of cyber-physical systems in connected/autonomous vehicles, manufacturing, the IoT, and healthcare.

Deepan Muthirayan received the B.Tech. and M.Tech. degrees from the Indian Institute of Technology Madras in 2010 and the Ph.D. degree from the University of California at Berkeley in 2016. He is currently a Post-Doctoral Researcher with the Department of Electrical Engineering and Computer Science, University of California at Irvine (UC Irvine). His doctoral thesis work focused on market mechanisms for integrating demand flexibility in energy systems. Before his term at UC Irvine, he was a Post-Doctoral Associate with Cornell University, where his work focused on online scheduling algorithms for managing demand flexibility. His current research interests include control theory, machine learning, topics at the intersection of learning and control, online learning, online algorithms, game theory, and their application to smart systems.

Pramod P. Khargonekar (Fellow, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Bombay, India, in 1977, and the M.S. degree in mathematics and the Ph.D. degree in electrical engineering from the University of Florida in 1980 and 1981, respectively. He was the Chairman of the Department of Electrical Engineering and Computer Science from 1997 to 2001 and held the position of the Claude E. Shannon Professor of Engineering Science with the University of Michigan. From 2001 to 2009, he was the Dean of the College of Engineering and an Eckis Professor of Electrical and Computer Engineering with the University of Florida till 2016. After serving briefly as the Deputy Director of Technology at the ARPA-E for the term 2012–2013, he was appointed by the National Science Foundation (NSF) to serve as the Assistant Director for the Directorate of Engineering (ENG) in March 2013, a position he held till June 2016. He is currently the Vice Chancellor for Research and a Distinguished Professor of electrical engineering and computer science with the University of California at Irvine. His research and teaching interests are centered on theory and applications of systems and control. He has received numerous honors and awards, including the IEEE Control Systems Award, the IEEE Baker Prize, the IEEE CSS Axelby Award, the NSF Presidential Young Investigator Award, and the AACC Eckman Award. He is also a Fellow of IFAC and AAAS.

Mohammad Abdullah Al Faruque (Senior Member, IEEE) received the B.Sc. degree in computer science and engineering (CSE) from the Bangladesh University of Engineering and Technology (BUET) in 2002, the M.Sc. degree in computer science from Aachen Technical University, Germany, in 2004, and the Ph.D. degree in computer science from the Karlsruhe Institute of Technology, Germany, in 2009. He is currently an Associate Professor with the University of California at Irvine (UCI) and directing the Embedded and Cyber-Physical Systems Laboratory. He served as the Emulex Career Development Chair from October 2012 to July 2015. Before, he was with Siemens Corporate Research and Technology, Princeton, NJ, USA, as a Research Scientist. He is also the author of two published books. Besides more than 120 IEEE/ACM publications in the premier journals and conferences, he holds nine U.S. patents. His current research is focused on the system-level design of embedded and cyber-physical-systems (CPS) with a special interest in low-power design, CPS security, and data-driven CPS design. He is also an ACM Senior Member. Besides many other awards, he was the recipient of the School of Engineering Mid-Career Faculty Award for Research 2019, the IEEE Technical Committee on Cyber-Physical Systems Early-Career Award 2018, the IEEE CEDA Ernest S. Kuh Early Career Award 2016, the UCI Academic Senate Distinguished Early-Career Faculty Award for Research 2017, and the School of Engineering Early-Career Faculty Award for Research 2017.