Spatio-Temporal Scene-Graph Embedding for Autonomous Vehicle Collision Prediction

Arnav V. Malawade*, Shih-Yuan Yu*, Brandon Hsu, Deepan Muthirayan,
Pramod P. Khargonekar, Mohammad A. Al Faruque

Department of Electrical Engineering & Computer Science, University of California - Irvine, Irvine, CA 92697,
USA {malawada, shihyuay, bdhsu, dmuthira, pramod.khargonekar, alfaruqu}@uci.edu

Abstract—In autonomous vehicles (AV), early warning systems rely on collision prediction to ensure occupant safety. However, state-of-the-art methods using deep convolutional networks either fail at modeling collisions or are too expensive/slow, making them less suitable for deployment on AV edge hardware. To address these limitations, we propose SG2VEC, a spatio-temporal *scene-graph* embedding methodology that uses Graph Neural Network (GNN) and Long Short-Term Memory (LSTM) layers to predict future collisions via visual scene perception. We demonstrate that SG2VEC predicts collisions 8.11% more accurately and 39.07% earlier than the state-of-the-art method on synthesized datasets, and 29.47% more accurately on a challenging real-world collision dataset. We also show that SG2VEC is better than the state-of-the-art at transferring knowledge from synthetic datasets to real-world driving datasets. Finally, we demonstrate that SG2VEC performs inference 9.3x faster with an 88.0% smaller model, 32.4% less power, and 92.8% less energy than the state-of-the-art method on the industry-standard Nvidia DRIVE PX 2 platform, making it more suitable for implementation on the edge.

Index Terms—scene graph, collision prediction, autonomous vehicles, graph learning, ADAS

I. INTRODUCTION

The synergy of Artificial Intelligence (AI) and the Internet of Things (IoT) has accelerated the advancement of Autonomous Vehicle (AV) technologies, which is expected to revolutionize transportation by reducing traffic and improving road safety [1], [2]. However, recent reports of AV crashes suggest that there are still significant limitations. For example, multiple fatal Tesla Autopilot crashes can primarily be attributed to perception system failures [3], [4]. Additionally, the infamous fatal collision between an Uber self-driving vehicle and a pedestrian can be attributed to perception and prediction failures by the AV [5]. These accidents (among others) have eroded public trust in AVs, and nearly 50% or more of the public have expressed their mistrust in AVs [6]. Current statistics indicate that perception and prediction errors were factors in over 40% of driver-related crashes between conventional vehicles [7]. However, a significant number of reported AV collisions are also the result of these errors [8], [9]. Thus, in this paper, we aim to improve the safety and acceptance of AVs by incorporating *scene-graphs* into the perception pipeline of collision prediction systems to improve scene understanding.

For the past several years, automotive manufacturers have begun equipping consumer vehicles with statistics-based collision avoidance systems based on calculated Single Behavior Threat Metrics (SBTMs) such as Time to Collision (TTC), Time to React (TTR), etc. [10], [11]. However, these methods lack robustness since they make significant assumptions about the behavior of vehicles on the road. A very limiting assumption they make is that vehicles do not diverge from their current trajectories [11]. SBTMs can also fail in specific scenarios. For example, TTC can fail when following a vehicle at the same velocity within a very short distance [11]. As a result, these methods are less capable of generalizing and can perform poorly in complex road scenarios. Moreover, to reduce false positives, these systems are designed to respond at the last possible moment [12]. Under such circumstances, the AV control system can fail to take timely corrective actions [13] if the system fails to predict a collision or estimates the TTC inaccurately.

More effective collision prediction methods using Deep Learning (DL) have also been proposed in the literature. However, these approaches can be limited because they do not explicitly capture the relationships between the objects in a traffic scene. Understanding these relationships could be critical as it is suggested that a human's ability to understand complex scenarios and identify potential risks relies on cognitive mechanisms for representing structure and reasoning about inter-object relationships [14]. These models also require large datasets that are often costly or unsafe to generate. Synthetic datasets are typically used to augment the limited real-world data to train the models in such cases [15]. However, these trained models must then be able to transfer the knowledge gained from synthetic datasets to real-world driving scenarios. Furthermore, DL models contain millions of parameters and require IoT edge devices with significant computational power and memory to run efficiently. Likewise, hosting these models on the cloud is infeasible because it requires persistent low-latency internet connections.

In summary, the key research challenges associated with autonomous vehicle collision prediction are:

1) Capturing complex relationships between road participants.
2) Detecting future collisions early enough such that the AV can take corrective actions.
3) Generalizing to a wide range of traffic scenarios.

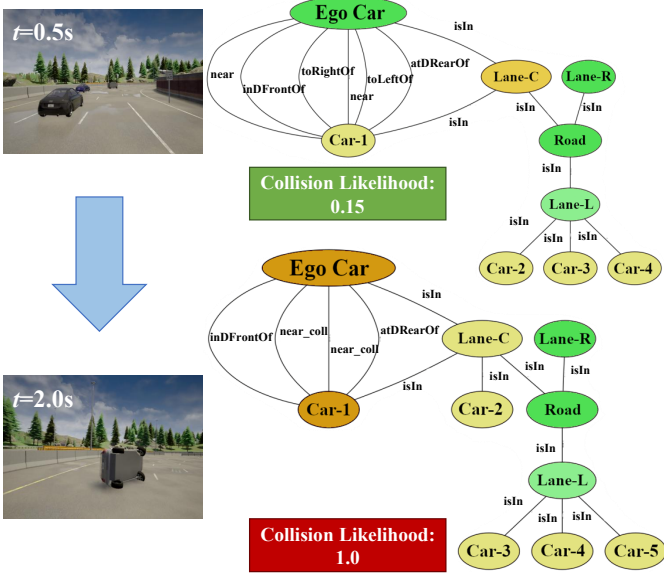* Arnav V. Malawade and Shih-Yuan Yu contributed equally to this article.

Fig. 1: How SG2VEC predicts collisions using *scene-graphs*. Each node's color indicates its attention score (importance to the collision likelihood) from orange (high) to green (low).

4) Developing algorithms that can run efficiently on automotive IoT edge devices.

In this work, we propose the use of *scene-graphs* to represent road scenes and model inter-object relationships as we hypothesize that this will improve perception and scene understanding. Recently, several works have shown that graph-based methods that capture and model complex relationships between entities can improve performance at high-level tasks such as behavior classification [16], [17] and semantic segmentation [18]. *Scene-graphs* are used in many domains to abstract the locations and attributes of objects in a scene and model inter-object relationships [16], [17], [19]. In our prior work, [20], we demonstrated that a *scene-graph* sequence classification approach can better assess the subjective risk of driving clips compared to a conventional CNN+LSTM-based approach; our approach could also better generalize and transfer knowledge across datasets. This paper extends the approach presented in [20] to collision prediction by enabling the prediction of future road states via changes to the temporal modeling components of the architecture and changes in the problem formulation. The *scene-graph* representation we propose represents traffic objects as nodes and the relationships between them as edges. The novelty of our *scene-graph* representation lies in our graph construction technique that is specifically designed for higher-level AV decisions such as collision prediction. One of our key contributions is showing that *a graph-based intermediate representation can be very effective and efficient for higher-level AV tasks directly related to driving decisions*.

Our proposed methodology for collision prediction, SG2VEC, is shown in Figure 1. It combines the *scene-graph* representation with a graph-embedding architecture to generate a sequence of *scene-graph* embeddings for the sequence of visual inputs perceived by an AV. The graph embedding technique we use is based on the core MR-GCN framework [16] adapted for the collision prediction problem. The sequence of graph embeddings is then input to a Long Short-Term Memory (LSTM) network to make the final prediction on the possibility of a future collision. To the best of our knowledge, our work is the first to propose using *scene-graphs* for early collision prediction.

Our paper makes the following key research contributions:

1) We demonstrate that our SG2VEC collision prediction methodology significantly outperforms the current state of the art on simulated lane-change datasets and a very challenging real-world collision dataset containing a wide range of driving actions, collision types, and weather/road conditions.
2) We demonstrate that SG2VEC can transfer knowledge gained from simulated data to real-world driving data more effectively than the state-of-the-art method.
3) We show that SG2VEC performs faster inference and requires less power than the state-of-the-art method on the industry-standard Nvidia DRIVE PX 2 autonomous driving hardware platform, used in all 2016-2018 Tesla models for their Autopilot system [21].

II. RELATED WORK

A. Early Collision Prediction

Since collision prediction is key to the safety of AVs, a wide range of solutions have been proposed by academia and industry. As mentioned earlier, current consumer vehicles use statistics-based SBTMs for collision prediction but can perform poorly in complex situations [10], [11] or react too late to avoid collisions [12], [13]. Expanding on these approaches, companies like Mobileye and Nvidia have proposed more comprehensive mathematical models for ensuring AV safety, namely Responsibility-Sensitive Safety (RSS) [22] and Nvidia Safety Force Field [23], respectively. However, these models are heavily rule-based and can thus be fragile in complex situations with high uncertainty. Additionally, computing future trajectory constraints with RSS is non-trivial and can require vehicle-specific calibration [24].

Model-based probabilistic and deep learning approaches for collision prediction have also been proposed. For example, [25] proposes a model-based probabilistic technique that uses the roadway geometry, ego trajectory, and position/velocity of road objects to predict future object positions. However, this model is highly conservative and is likely to have a high false-positive rate. Similarly, [26] and [27] use model-based approaches but require significant domain knowledge about the driving scene, such as road geometry information as well as accurate vehicle position and velocity information. [28] proposes a deep learning collision prediction approach. Still, due to its use of pre-processed trajectory data captured from cameras overlooking a highway, it is not ego-centric and cannot be practically used for on-vehicle collision prediction. In a different approach, [29] proposes a Deep Predictive Model (DPM) that used a Bayesian Convolutional LSTM for collision risk assessment where image data, vehicle telemetry data, and driving inputs were all factors in the risk assessment decision.

However, this approach was only evaluated on simula
scenes containing two vehicles and no other dynami
Thus, DPM's performance may suffer when evaluated
complex road scenarios.

In contrast to these existing works, we propose
which captures structural and relational information
scene in a *scene-graph* representation and computes
temporal embedding to predict collisions. Additio
perform experiments that were not done in many pri
such as evaluating each model's capability to transf
edge, efficiency on AV hardware, performance on a
real-world crash dataset, and ability to predict collisi
We primarily compare our methodology with the DP
the state-of-the-art data-driven collision prediction fi
for AVs that considers both spatial and tempora
Although the DPM uses multiple modalities for ser
results in [29] show that it achieves an accuracy (of
that is just 0.24% less using just the image sensing modality.
In this work, we compare our proposed SG2VEC methodology
and the DPM on image-only datasets, which is fair because
the DPM's performance does not vary much with the inclusion
of other modalities.

B. AV Scene-Graphs and Optimization Techniques

Several works have proposed graph-based methods for scene
understanding. For example, [16] proposed a multi-relational
graph convolutional network (MR-GCN) that uses both spatial
and temporal information to classify vehicle driving behavior.
Similarly, in [17], an *Ego-Thing* and *Ego-Stuff* graph are
used to model and classify the ego vehicle's interactions with
moving and stationary objects, respectively. In our prior work,
we demonstrated that a *scene-graph* sequence embedding
approach assesses driving risk better than the state-of-the-art
CNN-LSTM approach [20]. In [20], we utilized an architecture
consisting of MR-GCN layers for spatial modeling and an
LSTM with attention for temporal modeling; however, this
architecture was only capable of performing binary sequence-
level classification over a complete video clip. Thus, although
our prior architecture could accurately assess the subjective
risk of complete driving sequences, it was not capable of
predicting the future state of a scene.

Current autonomous driving systems consume a substantial
amount of power (up to 500 Watts for the Nvidia DRIVE AGX
Pegasus), demanding more robust cooling and power delivery
mechanisms. Thus, many have tried to optimize AV tasks
for efficiency without sacrificing performance. Existing ap-
proaches have proposed methods for jointly optimizing power
consumption and latency for localization [30], perception [31],
and control [32]. However, to the best of our knowledge, no
work has explored this optimization for AV safety systems,
such as collision prediction systems.

III. SCENE-GRAPH EMBEDDING METHODOLOGY

In SG2VEC, we formulate the problem of collision predic-
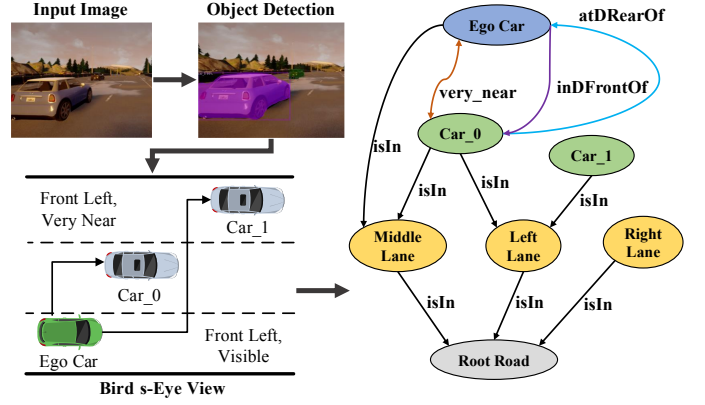tion as a time-series classification problem where the goal is

Fig. 2: An illustration of our *scene-graph* extraction process.

to predict if a collision will occur in the *near future*. Our goal
is to accurately model the spatio-temporal function f, where

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

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

SG2VEC consists of two parts (Figure 3) : (i) the *scene-
graph* extraction, and (ii) collision prediction through spatio-
temporal embedding, described in Section III-A and Sec-
tion III-B respectively.

A. Scene-Graph Extraction

The first step of our methodology is the extraction of *scene-
graphs* for the images of a driving scene. The extraction
pipeline forms the *scene-graph* for an image as in [33], [34]
by first detecting the objects in the image and then identifying
their relations based on their attributes. The difference from
prior works lies in the construction of a *scene-graph* that is
designed for higher-level AV decisions. We propose extracting
a *minimal* set of relations such as directional relations and
proximity relations. From our design space exploration we
found that adding many relation edges to the *scene-graph* adds
noise and impacts convergence while using too few relation
types reduces our model's expressivity. The best approach
we found across applications involves constructing mostly
ego-centric relations for a moderate range of relation types.
Figure 2 shows an example of the graph extraction process.

We denote the extracted *scene-graph* for the frame I_n
by $G_n = \{O_n, A_n\}$. Each *scene-graph* G_n is a directed,
heterogeneous multi-graph, where O_n denotes the nodes and
A_n is the adjacency matrix of the graph G_n. As shown in
Fig. 2, nodes represent the identified objects such as lanes,
roads, traffic signs, vehicles, pedestrians, etc., in a traffic scene.
The adjacency matrix A_n indicates the pair-wise relations
between each object in O_n. The extraction pipeline first
identifies the objects O_n by using Mask R-CNN [35]. Then,
it generates an inverse perspective mapping (also known as
a "birds-eye view" projection) of the image to estimate the
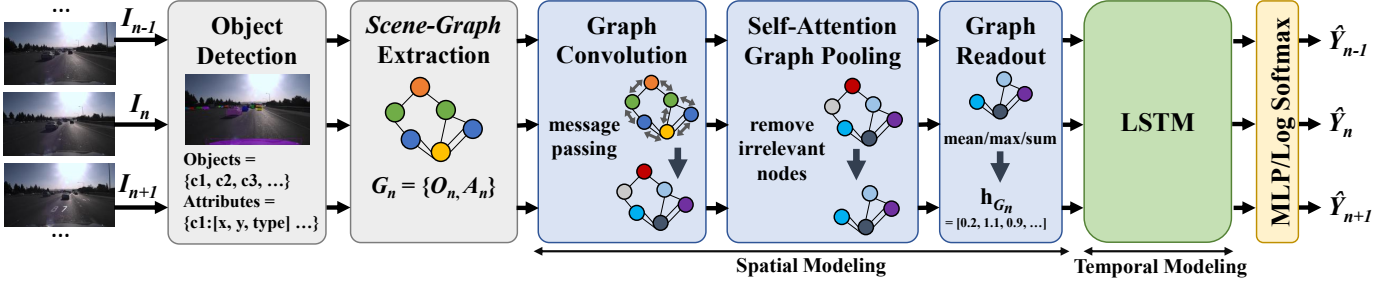locations of objects relative to the ego car, which are used to

Fig. 3: An illustration of SG2VEC's architecture.

construct the pair-wise relations between objects in A_n. For each camera angle, we calibrate the birds-eye view projection settings using known fixed distances, such as the lane length and width, as defined by the highway code. This enables us to estimate longitudinal and lateral distances accurately in the projection. For datasets captured by a single vehicle, this step only needs to be performed once. However, for datasets with a wide range of camera angles such as the *620-dash* dataset introduced later in the paper, this process needs to be performed once per vehicle. With a human operator, we found that this calibration step takes approximately 1 minute per camera angle on average.

The extraction pipeline identifies three kinds of pair-wise relations: *proximity* relations (e.g. *visible*, *near*, *very_near*, etc.), *directional* (e.g. *Front_Left*, *Rear_Right*, etc.) relations, and *belonging* (e.g. car_1 *isIn* left_lane) relations. Two objects are assigned the *proximity* relation, $r \in \{Near\_Collision$ (4 ft.), *Super_Near* (7 ft.), *Very_Near* (10 ft.), *Near* (16 ft.), *Visible* (25 ft.)$\}$ provided the objects are physically separated by a distance that is within that relation's threshold. The *directional relation*, $r \in \{Front\_Left, Left\_Front, Left\_Rear, Rear\_Left, Rear\_Right, Right\_Rear, Right\_Front, Front\_Right\}$, is assigned to a pair of objects, in this case between the ego-car and another car in the view, based on their relative orientation and only if they are within the *Near* threshold distance from one another. Additionally, the *isIn* relation identifies which vehicles are on which lanes (see Fig. 2). We use each vehicle's horizontal displacement relative to the ego vehicle to assign vehicles to either the *Left Lane*, *Middle Lane*, or *Right Lane* using the known lane width. Our abstraction only considers three-lane areas, and, as such, we map vehicles in all left lanes and all right lanes to the same *Left Lane* node *Right Lane* node respectively. If a vehicle overlaps two lanes (i.e., during a lane change), it is mapped to both lanes.

B. Collision Prediction

As shown in Figure 3, in our collision prediction methodology, each image I_n is first converted into a *scene-graph* $G_n = \{O_n, A_n\}$ with the pipeline mentioned in Section III-A. Each node $v \in O_n$ is initialized by a one-hot vector (*embedding*), denoted by $\mathbf{h}_v^{(0)}$. Then, the MR-GCN [36] layers are used to update these embeddings via the edges in A_n. Specifically, the l-th MR-GCN layer computes the node embedding for each node v, denoted as $\mathbf{h}_v^{(l)}$, as follows:

$$\mathbf{h}_v^{(l)} = \mathbf{\Phi}_0 \cdot \mathbf{h}_v^{(l-1)} + \sum_{r \in \mathbf{A_n}} \sum_{u \in \mathbf{N}_r(v)} \frac{1}{|\mathbf{N}_r(v)|} \mathbf{\Phi}_r \cdot \mathbf{h}_u^{(l-1)}, \quad (2)$$

where $N_r(v)$ denotes the set of neighbors of node v with respect to the relation $r \in A_n$, $\mathbf{\Phi}_r$ is a trainable relation-specific transformation for relation r, and $\mathbf{\Phi}_0$ is the self-connection for each node v that accounts for the influence of $\mathbf{h}_v^{(l-1)}$ on $\mathbf{h}_v^{(l)}$ [36]. After multiple MR-GCN layers, the output of each layer is concatenated to produce the final embedding for each node, denoted by $\mathbf{H}_v^L = \mathbf{CONCAT}(\{\mathbf{h}_v^{(l)}\}|l = 0, 1, ..., L)$, where L is the index of the last layer.

The final embeddings for *scene-graph* G_n, denoted by $\mathbf{X}_n^{prop}$, are then passed through a self-attention graph pooling (SAGPooling) layer that filters out irrelevant nodes from the graph, creating the pooled set of node embeddings $\mathbf{X}_n^{pool}$ and their edges $\mathbf{A}_n^{pool}$. In this layer, we use a graph convolution layer to predict the score $\alpha = \mathbf{SCORE}(\mathbf{X}_n^{prop}, \mathbf{A}_n^{prop})$ and then use α to perform *top-k* filtering to filter out the irrelevant nodes in the *scene-graph* [37].

Then, for each *scene-graph* G_n, the corresponding $\mathbf{X}_n^{pool}$ is passed through the graph readout layer that condenses the node embeddings (using operations such as sum, mean, max, etc.) to a single graph embedding $\mathbf{h}_{G_n}$. Then, this spatial embedding $\mathbf{h}_{G_n}$ is passed to the temporal model (LSTM) to generate a spatio-temporal embedding z_n as follows:

$$z_n, s_n = \mathbf{LSTM}(h_{G_n}, s_{n-1}) \quad (3)$$

The hidden state s_n of the LSTM is updated for each timestamp n. Lastly, each spatio-temporal embedding z_n is then passed through a Multi-Layer Perceptron (MLP) that outputs each class's confidence value. The two outputs of the MLP are compared, and $\hat{Y}_n$ is set to the index of the class with the greater confidence value (0 for no-collision or 1 for collision). During training, we calculate the cross-entropy loss between each set of non-binarized outputs $\hat{Y}_n$ and the corresponding labels for backpropagation.

IV. EXPERIMENTAL RESULTS

This section provides extensive experimental results to demonstrate SG2VEC's performance, efficiency, and transferability compared to the state-of-the-art collision prediction model, DPM [29]. For SG2VEC, we used 2 MR-GCN layers, each of size 64, one *SAGPooling* layer with a pooling ratio of 0.25, one *add-readout* layer, one LSTM layer with hidden size 20, one MLP layer with an output of size 2, and a LogSoftmax to generate the final confidence value for each class. For the DPM, we followed the architecture used in [29], which uses one 64x64x5 Convolutional LSTM (ConvLSTM) layer, one

32x32x5 ConvLSTM layer, one 16x16x5 ConvLSTM layer, one MLP layer with output size 64, one MLP layer with output size 2, and a Softmax to generate the final confidence value. For both models, we used a dropout of 0.1 and ReLU activation. The learning rates were 0.00005 for SG2VEC and 0.0001 for DPM. We ran the experiments shown in Sections IV-B and IV-C on a Windows PC with an AMD Ryzen Threadripper 1950X processor, 16 GB RAM, and an Nvidia GeForce RTX 2080 Super GPU.

A. Dataset Preparation

We prepared three types of datasets for our experiments: (i) synthesized datasets, (ii) a typical real-world driving dataset, and (iii) a complex real-world driving dataset. Examples from each dataset are shown in Figure 4. Our synthetic datasets focus on the highway lane change scenario as it is a common AV task. To evaluate the transferability of each model from synthetic datasets to real-world driving, we prepared a typical real-world dataset containing lane-change driving clips. Finally, we prepared the complex real-world driving dataset to evaluate each model's performance on a challenging dataset containing a broad spectrum of collision types, road conditions, and vehicle maneuvers. All datasets were collected at a 1280x720 resolution, and each clip spans 1-5 seconds.

1) Synthetic Datasets: To synthesize the datasets, we developed a tool using CARLA [15], an open-source driving simulator, and CARLA Scenario Runner[1] to generate lane change video clips with/without collisions. We generated a wide range of simulated lane changes with different numbers of cars, pedestrians, weather and lighting conditions, etc. We also customized each vehicle's driving behavior, such as their intended speed, probability of ignoring traffic lights, or the chance of avoiding collisions with other vehicles. We generated two synthetic datasets: a *271-syn* dataset and a *1043-syn* dataset, containing 271 and 1,043 video clips, respectively. These datasets have no-collision:collision label distributions of 6.12:1 and 7.91:1, respectively. In addition, we sub-sampled the *1043-syn* dataset to create *306-syn*: a balanced dataset that has a 1:1 distribution. Our synthetic *scene-graph* datasets[2] and our source code[3] are open-source and available online.

2) Typical Real-World Driving Dataset: This dataset, denoted as *571-honda*, is a subset of the Honda Driving Dataset (HDD) [38] containing 571 lane-change video clips from real-world driving with a distribution of 7.21:1. The HDD was recorded on the same vehicle during mostly safe driving in the California Bay Area.

3) Complex Real-World Driving Dataset: Our complex real-world driving dataset, denoted as *620-dash*, contains very challenging real-world collision scenarios drawn from the Detection of Traffic Anomaly dataset [39]. This dataset contains a wide range of drivers, car models, driving maneuvers, weather/road conditions, and collision types, as recorded by on-board dashboard cameras. Since the original dataset contains only collision clips, we prepared *620-dash* by splitting

[1]https://github.com/carla-simulator/scenario_runner
[2]https://dx.doi.org/10.21227/c0z9-1p30
[3]https://github.com/AICPS/sg-collision-prediction

Dataset	Model	Accuracy	AUC	MCC
271-syn	SG2VEC (5-frames)	**0.8979**	**0.9541**	**0.5362**
271-syn	SG2VEC	0.8812	0.9457	0.5145
271-syn	DPM	0.8733	0.8939	0.2160
306-syn	SG2VEC (5-frames)	0.7946	0.8653	0.5790
306-syn	SG2VEC	**0.8372**	**0.9091**	**0.6812**
306-syn	DPM	0.6846	0.6881	0.3677
1043-syn	SG2VEC (5-frames)	**0.9142**	**0.9623**	0.5323
1043-syn	SG2VEC	0.9095	0.9477	**0.5385**
1043-syn	DPM	0.8834	0.9175	0.2912
620-dash	SG2VEC (5-frames)	0.6534	0.7113	0.3053
620-dash	SG2VEC	**0.7007**	**0.7857**	**0.4017**
620-dash	DPM	0.4890	0.4717	-0.0366

TABLE I: Classification accuracy, AUC, and MCC for SG2VEC (Ours) and DPM.

each clip in the original dataset into two parts: (i) the beginning of the clip until 1 second before the collision, and (ii) from 1 second before the collision until the end of the collision. We then labeled part (i) as 'no-collision' and part (ii) as 'collision.' The *620-dash* dataset contains 315 collision video clips and 342 non-collision driving clips.

4) Labeling and Pre-Processing: We labeled the synthetic datasets and the *571-honda* dataset using human annotators. The final label assigned to a clip is the average of the labels assigned by the human annotators rounded to 0 (no collision) and 1 (collision/near collision). Each frame in a video clip is given a label identical to the entire clip's label to train the model to identify the preconditions of a future collision.

For SG2VEC, all the datasets were pre-processed using the *scene-graph* extraction pipeline mentioned in Section III-A to construct the *scene-graphs* for each video clip. For a given sequence, SG2VEC can leverage the full history of prior frames for each new prediction. For the DPM, the datasets were pre-processed to match the input format used in its original implementation [29]. Thus, the DPM uses 64x64 grayscale versions of the clips in the datasets turned into sets of subsequences J_n for a clip of length l defined as follows.

$$J_n = \{I_n, I_{n+1}, I_{n+2}, I_{n+3}, I_{n+4}\}, \text{for } n \in [1, l-4] \quad (4)$$

Since DPM only uses five prior frames to make each prediction, we also present results for SG2VEC using the same length of history, denoted as SG2VEC (5-frames) in the results.

B. Collision Prediction Performance

We evaluated SG2VEC and the DPM using classification accuracy, area under the ROC curve (AUC) [40], and Matthews Correlation Coefficient (MCC) [41]. MCC is considered a balanced measure of performance for binary classification even on datasets with significant class imbalances. The MCC score outputs a value between -1.0 and 1.0, where 1.0 corresponds to a perfect classifier, 0.0 to a random classifier, and -1.0 to an always incorrect classifier. Although class re-weighting helps compensate for the dataset imbalance during training, classification accuracy is typically less reliable for imbalanced datasets, so the primary metric we use to compare the models is MCC. We used stratified 5-fold cross-validation to produce the final results shown in Table I and Figure 5.

Fig. 4: Examples of driving scenes from our a) synthetic datasets, b) typical real-world dataset, and c) complex real-world dataset. In a), all driving scenes occur on highways with the same camera position and clearly defined road markings; lighting and weather are dynamically simulated in CARLA. In b) driving scenes occur on multiple types of clearly marked roads but lighting, camera angle, and weather are consistent across scenes. c) contains a much broader range of camera angles as well as more diverse weather and lighting conditions, including rain, snow, and night-time driving; it also contains a large number of clips on unpaved or unmarked roadways, as shown.

1) Synthetic Datasets: The performance of SG2VEC and the DPM on our synthetic datasets is shown in Table I. We find that our SG2VEC achieves higher accuracy, AUC, and MCC on every dataset, even when only using five prior frames as input. In addition to predicting collisions more accurately, SG2VEC also infers **5.5x** faster than the DPM on average. We attribute this to the differences in model complexity between our SG2VEC architecture and the much larger DPM model. Interestingly, SG2VEC (5-frames) achieves slightly better accuracy and AUC than SG2VEC on the imbalanced datasets and slightly lower overall performance on the balanced datasets. This is likely because the large number of safe lane changes in the imbalanced datasets adds noise during training and makes the full-history version of the model perform slightly worse. However, the full model can learn long-tail patterns for collision scenarios and performs better on the balanced datasets.

The DPM achieves relatively high accuracy and AUC on the imbalanced *271-syn* and *1043-syn* datasets, but suffers significantly on the balanced *306-syn* dataset. This drop indicates that the DPM could not identify the minority class (collision) well and tended to over-predict the majority class (no-collision). In terms of MCC, the DPM scores higher on the *306-syn* dataset than what it scores on the other datasets. This result is because the *306-syn* dataset has a balanced class distribution compared to the other datasets, which could enable the DPM to improve its prediction accuracy on the collision class.

In contrast, the SG2VEC methodology performs well on both balanced and imbalanced synthetic datasets with an average MCC of **0.5860**, an average accuracy of **87.97%**, and an average AUC of **0.9369**. Since MCC is scaled from -1.0 to 1.0, SG2VEC achieves a **14.72%** higher average MCC score than the DPM model.

The results from our SG2VEC ablation study are shown in Table III and support our hypothesis that both spatial modeling with MRGCN and temporal modeling with LSTM are core to SG2VEC's collision prediction performance. However, the MRGCN appears to be slightly more critical to performance than the LSTM. Interestingly the choice of pooling layer (no pooling, Top-K pooling, or SAG Pooling) does not seem to significantly affect performance at this task as long as LSTM is used; when no LSTM is used SAG Pooling presents a clear performance improvement.

2) Complex Real-World Dataset: The performance of both the models significantly drops on the highly complex real-world *620-dash* dataset due to the variations in the driving scenes and collision scenarios. This drop is to be expected as this dataset contains a wide range of driving actions, road environments, and collision scenarios, increasing the difficulty of the problem significantly. We took several steps to try and address this performance drop. First, we improved the birds-eye view (BEV) calibration on this dataset in comparison to the other datasets. Since the varying camera angles and road conditions in this dataset impact our ability to properly calibrate SG2VEC's BEV projection in a single step, we created custom BEV calibrations for each clip in the dataset, which improved performance somewhat. However, as shown in Figure 4c, a significant part of the dataset consists of driving clips on roads without any discernible lane markings, such as snowy, unpaved, or unmarked roadways. These factors make it challenging to correlate known fixed distances (i.e., the width and length of lane markings) with the projections of these clips. To further improve performance on this particular dataset, we performed extensive architecture and hyperparameter tuning. We found that, with one MRGCN layer of size 64, one LSTM layer with hidden size 100, no SAGPooling layer, and a

Dataset		
271-syn	SG2	
271-syn		
1043-syn	SG2	
1043-syn		

TABLE II: Aver

high learning ra
better performan
the beginning of
LSTM layer with
a keep ratio of 0.
features of each
collision likeliho
result, the additi
causing overfittin
The spatial embe
spatial model (1
combined with u
to capture more
and better genera
this dataset and
acquiring more c
with fixed camer
approaches. Hov
are little to no c
meet these requi
outperforms the I
21.17% higher a
higher MCC score. Since DPM achieves a negative MCC
score, its performance on this dataset is worse than that of a
random classifier (MCC of 0.0). Consistent with the synthetic
dataset results, *sg2vec* using all frames performs better on the
balanced *620-dash* dataset than SG2VEC (5-frames). Overall,
these results show that, on very challenging and complex real-
world driving scenarios, SG2VEC can perform much better
than the current state-of-the-art.

3) Time of Prediction: Since collision prediction is a time-
sensitive problem, we evaluated our methodology and the
DPM on their average time-of-prediction (ATP) for video
clips containing collisions. To calculate the ATP, we recorded
the first frame index in each collision clip when the model
correctly predicts that a collision would occur. We then
averaged these indices and compared them with the average
collision video clip length. Essentially, ATP gives an estimate
of how early each model can predict a future collision.
These results are shown in Table II. On the *1043-syn* dataset,
SG2VEC achieves 0.1725 for the ratio of the ATP and the
average sequence length while the DPM achieves a ratio
of 0.2382, indicating that SG2VEC predicts future collisions
39.07% earlier than the DPM on average. In the context of
real-world collision prediction, the average sequence in the
1043-syn dataset represents 1.867 seconds of data. Thus, our
methodology predicted collisions **122.7** milliseconds earlier
than DPM on average. This extra time can be critical for
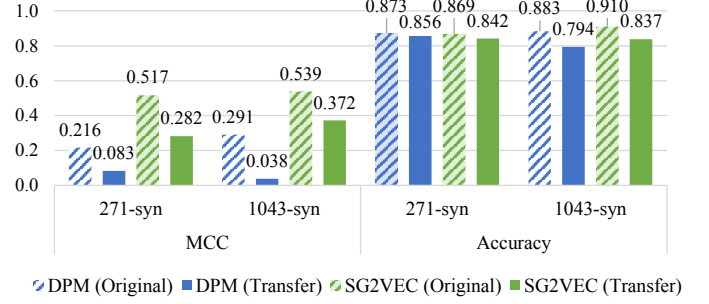ensuring that the AV avoids an impending collision.

Fig. 5: Performance after transferring the models trained on
synthetic *271-syn* and *1043-syn* datasets to the real-world *571-
honda* dataset.

C. Transferability From Synthetic to Real-World Datasets

The collision prediction models trained on simulated
datasets must be transferable to real-world driving as it can
differ significantly from simulations. To evaluate each model's
ability to transfer knowledge, we trained each model on a
synthetic dataset before testing it on the *571-honda* dataset.
No additional domain adaptation was performed. We did not
evaluate transferability to the *620-dash* dataset because it
contains a wide range of highly dynamic driving maneuvers
that were not present in our synthesized datasets. As such,
evaluating transferability between our synthesized datasets and
the 620-dash dataset would yield poor performance and would
not provide insight. Figure 5 compares the accuracy and MCC
for both the models on each training dataset and the *571-honda*
dataset after transferring the trained model.

We observe that the SG2VEC model achieves a significantly
higher MCC score than the DPM model after the transfer,
suggesting that our methodology can better transfer knowledge
from a synthetic to a real-world dataset compared to the state-
of-the-art DPM model. The drop in MCC values observed for
both the models when transferred to the *571-honda* dataset
can be attributed to the characteristic differences between
the simulated and real-world datasets; the *571-honda* dataset
contains a more heterogeneous set of road environments, light-
ing conditions, driving styles, etc., so a drop in performance
after the transfer is expected. We also note that the MCC
score for the SG2VEC model trained on *271-syn* dataset drops
more than the model trained on the *1043-syn* dataset after
the transfer, likely due to the smaller training dataset size.
Regarding accuracy, the SG2VEC model trained on *1043-syn*
achieves 4.37% higher accuracy and the model trained *271-syn*
dataset achieves 1.47% lower accuracy than the DPM model
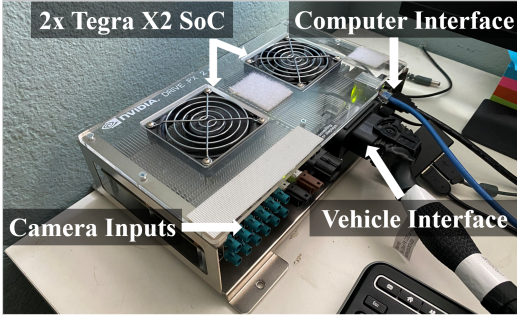trained on the same datasets. The DPM's similar accuracy after

Fig. 6: Our experimental setup for evaluating SG2VEC and DPM on the industry-standard Nvidia DRIVE PX 2 hardware.

transfer likely results from the class imbalance in the *571-honda* dataset. Overall, we hypothesize that SG2VEC's use of an intermediate representation (i.e., *scene-graphs*) inherently improves its ability to generalize and thus results in an improved ability to transfer knowledge compared to CNN-based deep learning approaches.

D. Evaluation on Industry-Standard AV Hardware

To demonstrate that the SG2VEC is implementable on industry-standard AV hardware, we measured its inference time (milliseconds), model size (kilobytes), power consumption (watts), and energy consumption per frame (milli-joules) on the industry-standard Nvidia DRIVE PX 2 platform, which was used by Tesla for their Autopilot system from 2016 to 2018 [21]. Our hardware setup is shown in Figure 6. For the inference time, we evaluated the average inference time (AIT) in milliseconds taken by each algorithm to process each frame. We recorded power usage metrics using a power meter connected to the power supply of the PX 2. To ensure that the reported numbers only reflected each model's power consumption and not that of background processes, we subtracted the hardware's idle power consumption from the averages recorded during each test. For a fair comparison, we captured the metrics for the core algorithm (i.e., the SG2VEC and DPM model), excluding the contribution from data loading and pre-processing. Both models were run with a batch size of 1 to emulate the real-world data stream where images are processed as they are received. For comparison, we also show the AIT on a PC for the two models.

Our results are shown in Table IV. SG2VEC performs inference **9.3x** faster than the DPM on the PX 2 with an **88.0%** smaller model and **32.4%** less power, making it undoubtedly more practical for real-world deployment. Our model also uses **92.8%** less energy to process each frame, which can be beneficial for electric vehicles with limited battery capacity. With an AIT of 0.4828 ms, SG2VEC can theoretically process up to 2,071 frames/second (fps). In contrast, with an AIT of 4.535 ms, the DPM can only process up to 220 fps. In the context of real-world collision prediction, this means that SG2VEC could easily support multiple 60 fps camera inputs from the AV while DPM would struggle to support more than three.

Model	PC AIT (ms)	PX2 AIT (ms)	Size (KB)	Power (W)	Energy/frame (mJ)
SG2VEC	**0.2549**	**0.4828**	**331**	**2.99**	**1.44**
DPM	1.393	4.535	2,764	4.42	20.0

TABLE IV: Performance evaluation of inference on *271-syn* on the Nvidia DRIVE PX 2.

V. CONCLUSION

With our experiments, we demonstrated that our *scene-graph* embedding methodology for collision prediction, SG2VEC, outperforms the state-of-the-art method, DPM, in terms of average MCC (0.5055 vs. 0.2096), average inference time (0.255 ms vs. 1.39 ms), and average time of prediction (39.07% sooner than DPM). Additionally, we demonstrated that SG2VEC could transfer knowledge from synthetic datasets to real-world driving datasets more effectively than the DPM, achieving an average transfer MCC of 0.327 vs. 0.060. Finally, we showed that our methodology performs faster inference than the DPM (0.4828 ms vs. 4.535 ms) with a smaller model size (331 KB vs. 2,764 KB) and reduced power consumption (2.99 W vs. 4.42 W) on the industry-standard Nvidia DRIVE PX 2 autonomous driving platform. In the context of real-world collision prediction, these results indicate that SG2VEC is a more practical choice for AV safety and could significantly improve consumer trust in AVs. Few works have explored graph-based solutions for other complex AV challenges such as localization, path planning, and control. These are open research problems that we reserve for future work.

REFERENCES

[1] S. vom Dorff, B. Böddeker *et al.*, "A fail-safe architecture for automated driving," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 828–833, 2020.

[2] T. Bijlsma, A. Buriachevskyi *et al.*, "A distributed safety mechanism using middleware and hypervisors for autonomous vehicles," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1175–1180, 2020.

[3] National Transportation Safety Board, "Collision Between a Sport Utility Vehicle Operating With Partial Driving Automation and a Crash Attenuator," National Transportation Safety Board, Tech. Rep. NTSB/HAR-20/01, 2020.

[4] ——, "Collision Between Car Operating with Partial Driving Automation and Truck-Tractor Semitrailer," National Transportation Safety Board, Tech. Rep. NTSB/HAB-20/01, 2020.

[5] ——, "Collision between vehicle controlled by developmental automated driving system and pedestrian," National Transportation Safety Board, Tech. Rep. NTSB/HAR-19/03, 2019.

[6] Deloitte Development LLC, "Global Automotive Consumer Study - North America," Deloitte Development LLC, Tech. Rep., 2020.

[7] A. S. Mueller, J. B. Cicchino, and D. S. Zuby, "What humanlike errors do autonomous vehicles need to avoid to maximize safety?" *Journal of Safety Research*, 2020.

[8] B. Schoettle and M. Sivak, "A preliminary analysis of real-world crashes involving self-driving vehicles," University of Michigan Transportation Research Institute, Tech. Rep. UMTRI-2015-34, 2015.

[9] C. Xu, Z. Ding *et al.*, "Statistical analysis of the patterns and characteristics of connected and autonomous vehicle involved crashes," *Journal of safety research*, vol. 71, pp. 41–47, 2019.

[10] H. L. D. Institute, "Volvo collision avoidance features: initial results," *Highway Loss Data Institute Bulletin*, vol. 29, no. 5, 2012.

[11] J. Dahl, G. R. de Campos *et al.*, "Collision avoidance: A literature review on threat-assessment techniques," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 101–113, 2018.

[12] S. Sontges, M. Koschi, and M. Althoff, "Worst-case analysis of the time-to-react using reachable sets," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1891–1897.

[13] J. Nilsson, A. C. Ödblom, and J. Fredriksson, "Worst-case analysis of automotive collision avoidance systems," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 1899–1911, 2015.

[14] P. W. Battaglia, J. B. Hamrick *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[15] A. Dosovitskiy, G. Ros *et al.*, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[16] S. Mylavarapu, M. Sandhu *et al.*, "Towards accurate vehicle behaviour classification with multi-relational graph convolutional networks," *arXiv preprint arXiv:2002.00786*, 2020.

[17] C. Li, Y. Meng *et al.*, "Learning 3d-aware egocentric spatial-temporal interaction via graph convolutional networks," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8418–8424, 2020.

[18] L. Kunze, T. Bruls *et al.*, "Reading between the lanes: Road layout reconstruction from partially segmented scenes," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 401–408, 2018.

[19] S. Mylavarapu, M. Sandhu *et al.*, "Understanding dynamic scenes using graph convolution networks," *arXiv preprint arXiv:2005.04437*, 2020.

[20] S.-Y. Yu, A. V. Malawade *et al.*, "Scene-graph augmented data-driven risk assessment of autonomous vehicle decisions," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[21] "All new Teslas are equipped with NVIDIA's new Drive PX 2 AI platform for self-driving - Electrek," https://electrek.co/2016/10/21/all-new-teslas-are-equipped-with-nvidias-new-drive-px-2-ai-platform-for-self-driving, Oct 2016, [Online; accessed 9. Nov. 2020].

[22] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[23] D. Nistér, H.-L. Lee *et al.*, "The safety force field," *NVIDIA White Paper*, 2019.

[24] B. Gassmann, F. Pasch *et al.*, "Integration of formal safety models on system level using the example of responsibility sensitive safety and carla driving simulator," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2020, pp. 358–369.

[25] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[26] Y. Wang, Z. Liu *et al.*, "Trajectory planning and safety assessment of autonomous vehicles based on motion prediction and model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8546–8556, 2019.

[27] L. Zhang, W. Xiao *et al.*, "Surrounding vehicles motion prediction for risk assessment and motion planning of autonomous vehicle in highway scenarios," *IEEE Access*, vol. 8, pp. 209 356–209 376, 2020.

[28] X. Wang, J. Liu *et al.*, "A real-time collision prediction mechanism with deep learning for intelligent transportation system," *IEEE transactions on vehicular technology*, vol. 69, no. 9, pp. 9497–9508, 2020.

[29] M. Strickland, G. Fainekos, and H. B. Amor, "Deep predictive models for collision risk assessment in autonomous driving," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2018.

[30] B. Asgari, R. Hadidi *et al.*, "Pisces: Power-aware implementation of slam by customizing efficient sparse algebra," *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

[31] S. Baidya, Y.-J. Ku *et al.*, "Vehicular and edge computing for emerging connected and autonomous vehicle applications," *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

[32] C. Huang, S. Xu *et al.*, "Opportunistic intermittent control with safety guarantees for autonomous systems," *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

[33] J. Yang, J. Lu *et al.*, "Graph r-cnn for scene graph generation," *Proceedings of the European conference on computer vision (ECCV)*, pp. 670–685, 2018.

[34] D. Xu, Y. Zhu *et al.*, "Scene graph generation by iterative message passing," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5410–5419, 2017.

[35] K. He, G. Gkioxari *et al.*, "Mask r-cnn," *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[36] M. Schlichtkrull, T. N. Kipf *et al.*, "Modeling relational data with graph convolutional networks," *European Semantic Web Conference*, pp. 593–607, 2018.

[37] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," *arXiv preprint arXiv:1904.08082*, 2019.

[38] V. Ramanishka, Y.-T. Chen *et al.*, "Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7699–7707, 2018.

[39] Y. Yao, X. Wang *et al.*, "When, where, and what? a new dataset for anomaly detection in driving videos," *arXiv preprint arXiv:2004.03044*, 2020.

[40] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[41] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, p. 6, 2020.

Arnav Vaibhav Malawade received a B.S. in Computer Science and Engineering and an M.S. in Computer Engineering from the University of California Irvine (UCI) in 2018 and 2021, respectively. He is currently a Ph.D. student at UCI under the supervision of Professor Mohammad Al Faruque. His research interests include the design and security of cyber-physical systems in connected/autonomous vehicles, manufacturing, IoT, and healthcare.

Shih-Yuan Yu received the B.S. and M.S. degrees in Computer Science and Information Engineering from the National Taiwan University (NTU) in 2014. He worked at MediaTek for 4 years. Currently he is a Ph.D. student in the University of California, Irvine. Now his research interests are about design automation of embedded systems using data-driven system modeling approaches. It covers incorporating machine learning methods to identify potential security issues in systems.

Brandon Hsu received a B.S. degree in Computer Engineering from the University of California Irvine (UCI) in 2021. His research interests lie broadly in machine and statistical learning with current focus on perception, representation learning, and autonomous vehicles.

Deepan Muthirayan is currently a Post-doctoral Researcher in the department of Electrical Engineering and Computer Science at University of California at Irvine. He obtained his Phd from the University of California at Berkeley (2016) and B.Tech/M.tech degree from the Indian Institute of Technology Madras (2010). His doctoral thesis work focused on integrating market mechanisms in energy systems. Before his term at UC Irvine he was a post-doctoral associate at 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 Khargonekar received B. Tech. Degree in electrical engineering in 1977 from the Indian Institute of Technology, Bombay, India, and M.S. degree in mathematics in 1980 and Ph.D. degree in electrical engineering in 1981 from the University of Florida, respectively. He was Chairman of the Department of Electrical Engineering and Computer Science from 1997 to 2001 and also held the position of Claude E. Shannon Professor of Engineering Science at The University of Michigan. From 2001 to 2009, he was Dean of the College of Engineering and Eckis Professor of Electrical and Computer Engineering at the University of Florida till 2016. After serving briefly as Deputy Director of Technology at ARPA-E in 2012-13, he was appointed by the National Science Foundation (NSF) to serve as Assistant Director for the Directorate of Engineering (ENG) in March 2013, a position he held till June 2016. Currently, he is Vice Chancellor for Research and Distinguished Professor of Electrical Engineering and Computer Science at the University of California, Irvine. His research and teaching interests are centered on theory and applications of systems and control. He has received numerous honors and awards including IEEE Control Systems Award, IEEE Baker Prize, IEEE CSS Axelby Award, NSF Presidential Young Investigator Award, AACC Eckman Award, and is a Fellow of IEEE, IFAC, and AAAS.

Mohammad Abdullah Al Faruque (M'06, SM'15) received his B.Sc. degree in Computer Science and Engineering (CSE) from Bangladesh University of Engineering and Technology (BUET) in 2002, and M.Sc. and Ph.D. degrees in Computer Science from Aachen Technical University and Karlsruhe Institute of Technology, Germany in 2004 and 2009, respectively. He is currently with the University of California Irvine (UCI) as an Associate Professor and Directing the Embedded and Cyber-Physical Systems Lab. He served as an Emulex Career Development Chair from October 2012 till July 2015. Before, he was with Siemens Corporate Research and Technology in Princeton, NJ as a Research Scientist. His current research is focused on the system-level design of embedded and Cyber-Physical-Systems (CPS) with special interest in low-power design, CPS security, data-driven CPS design, etc. He is an ACM senior member. He is the author of 2 published books. Besides many other awards, he is 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, and the IEEE CEDA Ernest S. Kuh Early Career Award 2016. He is also the recipient of the UCI Academic Senate Distinguished Early-Career Faculty Award for Research 2017 and the School of Engineering Early-Career Faculty Award for Research 2017. Besides 120+ IEEE/ACM publications in the premier journals and conferences, he holds 9 US patents.